UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

LEXMARK INTERNATIONAL, INC.,

Petitioner

v.

FLEXIWORLD TECHNOLOGIES, INC.,

Patent Owner.

PTAB Case No. IPR 2023-00846

Patent No. 9,965,233

DECLARATION OF DR. MAJID RABBANI IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,965,233

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

TABLE OF CONTENTS

I. INTRODUCTION 1
II. QUALIFICATIONS AND EXPERIENCE 1
A. Qualifications 2
B. Materials Reviewed 13
III. UNDERSTANDING OF LEGAL PRINCIPLES AND PERSPECTIVES 14
A. Types of Claims 14
B. Anticipation and Obviousness 15
C. Claim Construction 17
D. Person of Ordinary Skill in the Art 18
IV. STATE OF THE ART 20
A. Image Encoding and Compression 23
B. Division of Labor and Universal Printing: 38
V. THE '233 PATENT 48
A. Overview of the Alleged Invention 48
B. Prosecution History 54
VI. DETAILED DISCUSSION OF KEY PRIOR ART 54
A. U.S. Patent No. 6,748,471 ("Keeney") (Ex. 1004.) 54
B. U.S. Patent No. 6,091,507 ("Vatland") (Ex. 1005.) 56
VII. GROUNDS FOR CHALLENGE 57
A. Ground 1: Claims 1, 2, 4, 5, 7, 9, 11, 12, 13, 17-21, 27, 29, 30, 35, 36, 47, 48, 50, 52, 57, and 61 are obvious over Keeney. 57
1. Claim 47 58
a. [47.PRE]: 58
b. [47.A]: 62
c. [47.B]: 65
d. [47.C]: 65
e. [47.D]: 68
f. [47.E]: 70
g. [47.F]: 72
h. [47.G]: 77

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

i. [47.H.i]: 79
j. [47.H.ii]: 80
k. [47.I]: 81
2. Dependent claim 48: 82
3. Dependent Claim 61:. 84
a. [1.PRE]: 85
b. [1.A]: 85
c. [1.B]: 85
d. [1.C]: 86
e. [1.D]: 86
f. [1.E]: 87
g. [1.F]: 88
h. [1.G]: 88
i. [1.H]: 88
j. [1.I]: 89
k. [1.J]: 89
l. [1.K]: 90
m. [1.L]: 90
n. [1.M.i]: 90
o. [1.M.ii]: 91
4. Claim 2: 92
a. [2.A]: 92
b. [2.B]: 92
c. [2.C]: 92
d. Claim 4: … 92
5. Claim 6: 93
a. [6.A]: 93
b. [6.B]: 93
6. Claim 7:. 94
7. Claim 9: 95
a. [9.A]: 95
b. [9.B]: 95
8. Claim 11: 96
a. [11.PRE]: 96
b. [11.A]: 96
c. [11.B]: 96
9. Claim 12: 96
10. Claim 13: 97
11. Claim 17:. 97
12. Claim 18: 97

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

a. [18.PRE]: 97
b. [18.A]: 98
c. [18.B]: 99
d. [18.C]: 100
e. [18.D]: 100
f. [18.E]: 102
g. [18.F]: 102
h. [18.G]: 102
i. [18.H]: 103
j. [18.I]: 103
k. [18.J]: 103
l. [18.K]: 104
m. [18.L]: 104
n. [18.M]: 104
13. Claim 19: 104
a. [19.PRE]: 104
b. [19.A]: 105
c. [19.B]: 105
14. Claim 20: 105
15. Claim 21: 105
16. Claim 27: 106
a. [27.PRE]: 106
b. [27.A]: 106
c. [27.B,C]: 107
d. [27.D]: 107
e. [27.E]: 108
f. [27.F]: 108
g. [27.G]: 108
h. Claim 29: 109
i. Claim 30: 109
j. Claim 35: 109
k. Claim 36: 110
17. Claim 50: 111
a. [50.PRE]: 111
b. [50.A]: 112
c. [50.B]: 112
d. [50.C]: 112
e. [50.D,E]: 112
f. [50.F]: 113
g. [50.G]: 113

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

h. [50.H]: 113
i. [50.I]: 113
j. [50.J]: 114
k. [50.K]: 114
l. [50.L]: 114
m. [50.M]: 114
n. [50.N]: 115
o. [50.O]: 115
p. [50.P]: 116
18. Claim 52: 116
19. Claim 57: 116

B. Ground 2: Claims 47, 48, and 61 are obvious over Keeney and Vatland 117
1. Motivation to Combine 117
2. Claim 47 117
a. [47.PRE]: 117
b. [47.A]: 118
c. [47.B]: 118
d. [47.C]: 118
e. [47.D]: 118
f. [47.E]: 118
g. [47.F]: 118
h. [47.G]: 120
i. [47.H.i]: 120
j. [47.H.ii]: 120
k. [47.I]: 120
3. Dependent claim 48: 120
4. Dependent Claim 61: 120

VIII. CONCLUSION 121

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

LIST OF APPENDICES

Appendix [Ref]	Description of the Appended Document
Appendix A	*Curriculum Vitae* of Majid Rabbani
Appendix B	Dictionary Definitions: "Encoding"
Appendix C	"JPEG," Wikipedia, available at https://en.wikipedia.org/wiki/JPEG
Appendix D	Roy Hoffman, DATA COMPRESSION IN DIGITAL SYSTEMS (1997)
Appendix E	"Data Compression," Wikipedia, available at https://en.wikipedia.org/wiki/Data_compression
Appendix F	ITU T.81 Information technology — Digital Compression and Coding of Continuous-Tone Images, 1992
Appendix G	ITU-T Recommendation T.6 Terminal Equipment and Protocols for Telematic Services — Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile, 1993
Appendix H	U.S. Patent No. 5,956,467 to Rabbani, filed December 17, 1996
Appendix I	U.S. Patent No. 6,608,928 to Queiroz, filed November 3, 1999
Appendix J	U.S. Patent No. 6,285,889 to Nykänen, filed August 4, 1997
Appendix K	U.S. Patent No. 7,028,102 to Larsson, filed February 17, 2000
Appendix L	U.S. Patent No. 7,644,018 to Yukie, filed April 4, 2000
Appendix M	U.S. Patent No. 7,212,229 to Parulski, filed March 6, 2001
Appendix N	RFC 2393 - IP Payload Compression Protocol (IPComp), December 1998

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Appendix O	*Compression and Encryption in IPSec*, REDMOND MAGAZINE, June 24, 1998
Appendix P	J. P. McGregor and R. B. Lee, *Performance impact of data compression on virtual private network transactions*, PROCEEDINGS 25TH ANNUAL IEEE CONFERENCE ON LOCAL COMPUTER NETWORKS LCN 2000, 8-10 Nov. 2000, at 500

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

EXHIBITS

Exhibit No.	Description
Ex. 1001	U.S. Patent No. 9,965,233 (’233 patent)
Ex. 1002	Declaration of Dr. Majid Rabbani
Ex. 1003	File History for U.S. Patent No. 9,965,233
Ex. 1004	U.S. Patent No. 6,748,471 (“Keeney”)
Ex. 1005	U.S. Patent No. 6,0910507 (“Vatland”)
Ex. 1006	Docket for *Flexiworld Technologies, Inc. v Lexmark International Inc.*, 22-cv-110, Eastern District of Kentucky
Ex. 1007	’233 Patent Claim Limitation Matrix
Ex. 1008	“Application Software,” IEEE 100: The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition (2000)
Ex. 1009	RFC - 2406 (IPSec), 1998

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

I. INTRODUCTION

1. My name is Dr. Majid Rabbani. I have been retained as an expert consultant by Perkins Coie LLP on behalf of Lexmark International, Inc. ("Petitioner") in this proceeding before the United States Patent and Trademark Office ("PTO") against Flexiworld Technologies, Inc. ("Patent Owner") regarding U.S. Patent No. 9,965,233 ("the '233 patent") (Ex 1001). I have been asked to consider whether certain references disclose or suggest the features recited in claims 1, 2, 4, 5, 7, 9, 11, 12, 13, 17-21, 27, 29, 30, 35, 36, 47, 48, 50, 52, 57, and 61 ("the challenged claims") of the '233 patent. My opinions are set forth below.

2. I am being compensated for my time. I am not receiving any other form of compensation. I have no interest in the outcome of this proceeding, and my compensation is not dependent on the content of my opinions or the outcome of this proceeding.

II. QUALIFICATIONS AND EXPERIENCE

3. I am an independent consultant. All opinions stated in this Declaration are based on my own personal knowledge and professional judgment. In forming my opinions, I have relied on my knowledge and experience relating to the technology referenced in this Declaration.

4. I am over 18 years of age and, if I am called upon to do so, I would be competent to testify as to the matters set forth herein. A copy of my current

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

curriculum vitae, which includes a more detailed summary of my education and professional and academic experience, is provided as Appendix A to this Declaration. The following provides an overview of some of my experience that is relevant to the matters set forth in this Declaration.

A. Qualifications

1. I possess the knowledge, expertise, skills, experience, training and education to form an expert opinion and expert testimony in this matter. I have 43 years of experience in the field of digital information, signal, image, and video processing and encoding and communications. I received my Ph.D. in Electrical Engineering from the University of Wisconsin-Madison in 1983, with a specialty in digital signal and image processing and communications. My thesis was entitled "Restoration of Nonlinearly Distorted Images" and was concerned with deblurring digital images that were captured by partially or completely coherent light.

2. Since fall of 2016, I have been employed first as a visiting full professor of Electrical and Micro-Electronic Engineering at Rochester Institute of Technology (RIT), and currently as a Professor of Practice where I perform research, teach graduate courses in digital signal and image processing and compression and supervise and advise PhD student research in the area of information processing using deep learning. Graduate level courses that I have taught in the past 4 years include Digital Signal Processing, Digital Image Processing, Pattern Recognition,

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Digital Image and Video Compression, Information Theory, and Random Signals and Noise. In addition to my teaching responsibilities, I am currently the co-PI (principle investigator) on a $750,000 grant from the Army Research Labs to conduct research in the area of image and video processing with deep learning.

3. Prior to joining RIT, I was employed by the research labs of Eastman Kodak Company for 33 years, from 1983 (upon graduation with my Ph.D.), until April of 2016, when I retired from Kodak. Throughout my career at Kodak Research Laboratories, I held positions as Research Scientist, Senior Research Scientist, Research Associate, Senior Research Associate, Department Head, and, finally, as a Kodak Fellow, the title with which I retired from Kodak. The title of Kodak Fellow in the digital sector of Kodak has been held by only one other individual in the history of the Kodak Research Labs.

4. The focus of my work over my career has been digital information, signal, image and video processing, signal encoding and compression, and image and video analysis. During my years at Kodak, I have undertaken research and development related to imaging and printing technologies and have developed technologies implemented in printers and digital proofing systems. For example, I invented a method for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, which was

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

patented as U.S. Patent No. 5,956,467, titled "Encoding Color Image Data for Multipass Color Printers."

5. Upon joining Kodak in 1983, I initiated research in analyzing the transfer of signal and noise in medical screen-film systems. The result of my research, which was published both internal and external to Kodak, received the Kodak C. E. K. Mees Award in 1988 among a pool of 150 submitted technical reports. The Mees Award was regarded as Kodak's highest honor in conducting and reporting research. I was awarded the Mees Award for a second time almost two decades later in 2005 for my research on digital watermarking of images and video.

6. In 1985, I formed a research group and initiated research in the area of image and video compression, which was a novel topic at Kodak. The result of my research, which led to several patents and several internal and external publications, was implemented in the Kodak SV9600 Transceiver system in 1988. This product was used by CBS in 1989 to transmit still images from China during the unrest in Tiananmen Square. The Chinese government had shut down satellite links to stop the transmission of electronic images and the only means available to the journalists for transmitting electronic images was dial-up telephone lines. This product, which predated the JPEG standard, used a proprietary still image compression and encoding algorithm developed and patented by me and one of my colleagues. This algorithm was capable of compressing digital color images by an average factor of

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

20:1, which enabled the transmission of such images over dial-up telephone lines in less than one minute given the modem technology of that era and the typical size of a digital image. Because of the extraordinary engineering achievement of this product and its utility in communicating images, the National Academy of Television Arts and Sciences granted Kodak an Engineering Emmy Award in 1990 for outstanding achievement in the technical field of image compression. I received the award on behalf of Kodak during a ceremony in New York City.

7. In 1987, I started participating in the JPEG committee that was responsible for the development of the JPEG image compression standard as the Kodak representative. The JPEG Committee had just been formed in 1986, with the objective of developing methods for the lossy compression of continuous-tone digital still images, particularly, images produced by digital photography. I remained a participant in JPEG for the next 25 years and until 2012, during which I assumed many leadership positions to advance the objectives of the committee. For example, in 1990, when the JPEG committee had reached an impasse due to technical differences between IBM and AT&T, and the approval of the standard had been seriously jeopardized, I was appointed by ISO (International Organization for Standardization) as the technical mediator between the two companies to resolve the selection of the arithmetic coder technology. As another example of my leadership role, I was the Chairman of the Coding Efficiency Group (50+ scientists) of the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

JPEG-2000 committee from 1997-2000. I was interviewed in 2003 by the El Pais Spanish newspaper regarding my leadership role in the JPEG2000 compression standard after delivering a seminar on JPEG2000 in Madrid ("Majid Rabbani: Es major desarrollar estandares abiertos que propietarious," Madrid, Spain, October 2, 2003).

8. In 1992, I started participating at the MPEG Committee as the Kodak representative, a role that I continued for the next 20 years until 2012. In recognition of my dedication and contributions to the JPEG and MPEG committees, I was awarded in 1997 an Engineering Emmy Award for the second time by the National Academy of Television Arts and Sciences for my co-editorship role in the JPEG and MPEG standardization committees.

9. In 1991, I was the principal author of a book on the technical subject of image compression entitled "Digital Image Compression Techniques" that was published by the SPIE Press. This book achieved the highest sales record of any SPIE technical publication and sold more than 6,000 copies in a period of just a few years. In 1992, I edited another book entitled "Image Coding and Compression" as part of the SPIE Milestone Series. Additionally, I have authored several prominent book chapters on the topic of image and video compression and coding including ("JPEG2000 Core Coding system - Part 1" The JPEG2000 Suite, John Wiley and Sons, "JPEG Compression in Medical Imaging," in Handbook of Medical Imaging,

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

SPIE Press, and "Digital Image Compression," in Digital Image Processing: Fundamentals and Applications, Marcel Dekker).

10. My group was responsible for the development of the compression algorithm and the hierarchical file format used in the Kodak Photo-CD product that was launched in 1992. Photo-CD used a high-quality proprietary compression algorithm for encoding a still image in multiple resolutions (referred to as the Image Pac), the lowest of which was intended for thumbnail representation of the image. I was interviewed by the Photo Electronic Imaging Magazine regarding the Photo-CD product ("A Lossless Explanation of Photo-CD," Vol. 37, No 2, 1994).

11. Because of my contributions to the science and technology of image and video compression, I was elected as a Fellow of SPIE (International Society for Optics and Photonics) in 1993. This rank is achieved by less than 7% of all SPIE's 19,000 worldwide members. In 1997, I was elected as a Fellow of IEEE (professional association of Institute of Electrical and Electronics Engineers) for the same technical recognition, a rank granted to less than 3% of IEEE's 420,000 members worldwide. Currently, I am a Life Fellow of both the SPIE and the IEEE technical societies. In 2008, I was awarded the high honor of IEEE Region 1 Technical Award for my pioneering work in image compression. In 2015, I was awarded the Electronic Imaging Scientist of the Year Award by IS&T (Society for Imaging Science and Technology) and SPIE for fundamental contributions to image

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

compression. From 2013-2017, I served as the Chairman of the SPIE Fellows Committee. In 2018, I was awarded the SPIE Directors' Award, which is presented to an individual who, in the opinion of the SPIE Board of Directors, has rendered a significant service of outstanding benefit to the Society.

12. In 1992, I was asked by the Department of Justice to direct the efforts of Los Alamos scientists in the digital enhancement of George Holliday's video of the beating of Rodney King. I subsequently testified as the prosecution's expert witness in the Federal Court in Los Angeles. I was interviewed by the local CBS news in Rochester, NY and appeared in an article in the Rochester's local newspaper, Democrat and Chronicle, for my expert witness testimony. Because of my recognition in this event, I have been frequently hired by the law enforcement community in both US and Canada to train their video analysts in the methods of image and video processing, coding and compression.

13. Throughout my career at Kodak, I was constantly involved in educating the technical community on the subject matters of digital image and video processing and compression. I have held positions as adjunct faculty at both the Rochester Institute of Technology (my current employer) and the University of Rochester. I provided training in digital imaging to Kodak senior VP's and other Kodak internal training programs and taught numerous courses at technical societies such as IEEE, SPIE, IS&T, SMPTE, SID and SIRA, and educational institutions such as VIGC in

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Europe, MIT, Berkeley, and Portland State University. My educational activities are expected to have trained over 5,000 scientists and engineers in the area of digital image and video processing and have been honored by various awards, the most recent of which is the "Electronic Imaging Distinguished Educator Award," awarded by IS&T and SPIE in January of 2012 for 25 years of educational service to the electronic imaging community.

14. I have taught a short course on the various areas of digital image and video processing and compression for 35 consecutive years (1988-2022) at the Electronic Imaging (EI) Symposium held in Northern California Bay Area and regarded as the premier venue for promoting the electronic imaging science and technology. I was asked by the USPTO to spend two weeks in summers of 2007 and 2008 as the Chief Technical Expert to assist in planning and execution of a comprehensive training and mentoring program of patent examiners in the field of digital image and video processing.

15. Because of my expertise in the field of image and video compression, I was elected by the Digital Cinema Initiatives (DCI) Consortium as the sole industry expert to the Digital Cinema Compression Experts Panel during 2003-2004, which evaluated the suitability of compression standards for the motion picture industry.

16. During my career at Kodak, I was responsible for the research and deployment of many technologies and products related to image and video coding

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

and compression. Examples in addition to those cited above are the Kodak Hyper Compression product for storing a color facial image in the 57-byte record provided on the back of a standard credit card, space-qualified compression ASIC's used onboard the Space Imaging Ikonos satellite and Digital Globe QuickBird satellite, the Flashpix file format, various compression algorithms used in Kodak digital copiers, the first commercial Internet application of JPEG2000 by empowering the visualization of aerial imagery on Map Quest servers, and the proprietary compression technology used in the Kodak QT-100 and QT-150 first generation digital cameras that were marketed by Apple.

17. In 1996 I developed an encryption algorithm based on stream cipher technology for the secure transmission of digital visual content (such as high-quality images) over the network of Kodak Kiosks. I presented the technical details of my encryption algorithm to the National Security Agency and obtained the agency's approval for international exportation.

18. In the time span of early 2000, my research team was also in charge of evaluating the various video compression and encoding technologies and codecs for adoption in the Kodak digital camera models. My team, under my supervision and guidance, designed and conducted rigorous tests to evaluate the impact of various encoder implementations of a wide range of video compression standards, such as

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Motion JPEG, MPEG2, and H.264, on the quality and bit rate of the compressed video data.

19. I am an inventor or co-inventor on 44 issued patents, almost all of which pertain to the field of signal, image and video processing. I was awarded the rank of Kodak Distinguished Inventor by my previous employer, Eastman Kodak Company, in recognition of the magnitude and impact of my inventions.

20. I was the General Chair of the Electronic Imaging Symposium in 2012 and a co-chair of that Symposium in 2011 and in 1996. The Electronic Imaging Symposium, which has been held once a year for the past 36 years, consists of 20-24 conferences covering a broad range of technical areas pertaining to the field of electronic imaging and is considered as the premier venue for reporting research and technology in this field. Until 2015, this Symposium was co-sponsored by the SPIE and IS&T professional societies and it is currently sponsored by IS&T.

21. In June of 1995, I was asked by the National Research Council to organize and chair a 2-day technical event in Washington D.C., entitled "Maintaining and Certifying Image and Instrument Integrity." The Proceedings of this event, which was co-authored by me, was used by the Computer Science and Telecommunications Board (CSTB) to identify research topics and allocate government funding.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

22. I have been a plenary or a keynote speaker on more than 50 different occasions throughout my career, all of which pertain to the field of signal, image and video processing.

23. I have been involved in professional service in numerous technical venues concerned with the promotion of digital image and video processing technology. For example, I was the Chair of the SPIE Commercial + Scientific Sensing and Imaging Symposium, Anaheim, CA, in April 2017, which hosted several thousand technical participants. A detailed list of my professional involvements appears in my attached Curriculum Vitae (**Appendix A**).

24. As an example, from 2014 to 2018 I was the Associate Editor for the technical area of image and video compression standards for IEEE Transaction on Circuits and Systems for Video technology (TCSVT). This transaction, which covers the circuits and systems aspects of all video technologies, is a premier venue for publishing state-of-the-art research. Among areas covered are encoding and compression, transmission, communication and networking; and/or hardware and software design and implementation. As another example, for the past 25 years (since 1997) I have been an active member of the Program Committee of the IEEE Data Compression Conference (DCC), which is an international forum for current theoretical and experimental research on data compression and related applications.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B. Materials Reviewed

5. The opinions contained in this Declaration are based on the documents I have reviewed, my professional judgment, and my education, experience, and knowledge relating to digital information, signal, image, and video processing and encoding and communications, and related fields. In forming my opinions, I have reviewed all materials listed in the Exhibit List provided above, all materials filed in IPR2022-01114 by Roku, Inc., challenging the ’233 patent, as well as any other materials and information I reference in this Declaration.

6. My opinions contained in this Declaration are based on the documents I have reviewed and considered, as well as my knowledge and professional judgment. My opinions have been guided by my appreciation of how a person of ordinary skill in the art (“POSITA”) would have understood the state of the art, the prior art, and the claims and the specification of the ’233 patent at the time of the alleged invention.

7. The time of the alleged invention of the ’233 patent encompasses the time frames near and including November 20, 2000. I understand the application for the ’233 patent claims priority to a provisional application filed on November 20, 2000. (’233 patent, Patent Face). My opinions reflect how a POSITA (which I describe below) would have understood the ’233 patent, the prior art to the patent, and the state of the art at the time of the alleged invention.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

8. Based on my experience and expertise, it is my opinion that the prior art disclosed and/or suggested all the limitations recited in challenged claims 1, 2, 4, 5, 7, 9, 11, 12, 13, 17-21, 27, 29, 30, 35, 36, 47, 48, 50, 52, 57, and 61 of the '233 patent, as I discuss in detail below.

III. UNDERSTANDING OF LEGAL PRINCIPLES AND PERSPECTIVES

9. I am not a lawyer and I will not provide legal opinions. Although I am not a lawyer, I have been advised that certain legal standards are to be applied by technical experts in forming opinions regarding the meaning and validity of patent claims.

10. I understand that a patent claim is invalid if it is anticipated or obvious in view of the prior art and that a claim can be unpatentable even if all requirements of the claim cannot be found in a single prior art reference. I further understand that invalidating a claim requires that the claim be anticipated or obvious from the perspective of a POSITA at the time the invention was made.

11. In forming my analyses, opinions, and conclusions expressed in this Declaration, I have applied the legal principles set forth below, which were provided to me by counsel for Petitioners.

A. Types of Claims

12. I understand that there are two types of U.S. patent claims: 1) independent claims and 2) dependent claims. I understand that independent claims

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

only include the aspects stated in the independent claim. I further understand that dependent claims include the aspects stated in that dependent claim, and any other aspects stated in any claim from which that dependent claim depends.

B. Anticipation and Obviousness

13. I understand that a claim is not patentable if it is anticipated or obvious. I understand that anticipation of a claim requires that every element of a claim is disclosed expressly or inherently in a single prior art reference.

14. I further understand that obviousness of a claim requires that the claim be obvious from the perspective of a person of ordinary skill in the relevant art at the time the invention was made. I understand that the earliest patent application leading to the ’233 patent was a U.S. provisional patent application filed on November 20, 2000. I have therefore analyzed the unpatentability of the claims as of that day. I may refer to the relevant time as the “2000 timeframe” in this Declaration, with the understanding that this does not include the time period on or after the filing date of the first application (i.e., November 20, 2000).

15. I understand that to prove that prior art or a combination of prior art renders a patent obvious, it is necessary to:

a) identify the particular references that, singly or in combination, make the patent obvious;

b) specifically identify which elements of the patent claim appear in each of the asserted references; and

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

c) explain why a POSITA would have combined the references, and how they would have done so, to create the inventions claimed in the patent.

16. I further understand that exemplary rationales that may support a conclusion of obviousness include:

a) combining prior art elements according to known methods to yield predictable results;

b) simple substitution of one known element for another to obtain predictable results;

c) use of known technique(s) to improve similar devices (methods or products) in the same way;

d) applying a known technique to a known device (method or product) ready for improvement to yield predictable results;

e) "obvious to try" - choosing from a finite number of identified, predictable solutions with a reasonable expectation of success;

f) known work in one field of endeavor may prompt variations of the work for use in either the same field or a different field based on design incentives or other market forces if the variations are predictable to a POSITA; and

g) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

17. In analyzing obviousness, I understand that it is important to understand the scope of the claims, the level of skill in the relevant art, the scope and content of the prior art, the differences between the prior art and the claims, and any "secondary considerations" (described below). I also understand that if a technique has been used to improve one device, and a POSITA would recognize that it would improve

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill. Although I understand that it is not required, there may also be a specific "teaching, suggestion or motivation" to combine any first prior art reference with a second prior art reference. Such a "teaching, suggestion, or motivation" to combine the first prior art reference with the second prior art reference can be explicit or implicit and may provide further evidence of obviousness.

18. I understand that secondary (or objective) considerations are relevant to the determination of whether a claim is obvious. Such secondary (or objective) considerations can include evidence of commercial success caused by an invention, evidence of a long-felt need that was solved by an invention, evidence that others copied an invention, or evidence that an invention achieved a surprising or unexpected result. I understand that such evidence must have a nexus, or causal relationship to the elements of a claim, to be relevant to the obviousness or non-obviousness of the claim. I am not currently aware of any such evidence for the claims of the '233 patent and reserve my right to review and respond to such evidence if presented.

C. Claim Construction

19. I understand that, for purposes of my analysis, the terms appearing in the patent claims should be interpreted according to their "ordinary and customary

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

meaning" at the time the alleged invention was made. In determining the ordinary and customary meaning, the words of a claim are first given their plain meaning that those words would have had to a POSITA. I understand that the structure of the claims, the specification, and the file history also may be used to in the construction of a claim term. Additionally, treatises and dictionaries may be used, in limited circumstances, to determine the meaning attributed by a POSITA to a claim term at the time of filing.

20. For all claim terms considered in this Declaration, I have applied the ordinary and customary meaning of those terms as they would have been understood by a POSITA at the time the alleged invention was made. To the extent the Patent Owner proposes any constructions or the Board construes any claims, I reserve the right to address such proposals or constructions. Regardless, under any reasonable construction of the claims, the prior art discloses and/or suggests all the limitations recited in challenged claims of the '233 patent.

D. Person of Ordinary Skill in the Art

21. I understand that my assessment and determination of invalidity must be undertaken from the perspective of what would have been known or understood by a POSITA at the time of the earliest patent application supporting the '233 patent.

22. I have been informed and understand that the POSITA is not a real person, but rather a legal construct—a hypothetical person. This person is presumed

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

to have known the relevant art at the time of the invention. I have been informed by counsel that relevant factors in determining the level of ordinary skill include (1) the educational level of the inventor, (2) the type of problems encountered in the art, (3) prior art solutions to those problems, (4) the rapidity with which innovations are made, (5) sophistication of the technology, and (6) the educational level of active workers in the field. I further understand that the person of "ordinary skill" is not a person of extraordinary skill in the art. In other words, this person of ordinary skill is not necessarily an expert in the field.

23. As of the alleged November 2000 priority date, a POSITA would have at least a bachelor's degree in computer science, computer engineering, or electrical engineering (or equivalent experience) and at least two years of practical experience with printer technology. My opinion about the level of ordinary skill would not change if the claims were only entitled to the November 18, 2001 actual filing date of the '233 Patent.

24. As discussed above, I understand that when interpreting the claims of the '233 patent, I should do so based on the perspective of a POSITA as of the relevant priority date. I understand that Patent Owner has not yet taken a position with regard to the priority date of the challenged claims in related litigation. However, on its face, the '233 Patent claims priority to Provisional Application No. 60/252,682, filed on November 20, 2000. I do not take a position in this declaration

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

whether the challenged claims are in fact entitled to this priority date because all of the asserted prior art is prior art to the '233 Patent even in view of this priority date. However, if Patent Owner offers arguments or evidence asserting that the asserted prior art is not prior art to the '233 Patent, I reserve the right to offer additional opinions in rebuttal.

25. At the time of the alleged invention, I was a person of at least ordinary skill in the art, and I have applied the perspective of the POSITA in forming and expressing my opinions below.

IV. STATE OF THE ART

26. The explosive growth of the Internet was partly fueled by advancements in storage, compression, transmission, and data security of digital data, as well as computational capabilities of electronic devices, which had reached a certain level of maturity to allow proliferation of Internet-based applications. Encoding data for transmission over a network has long been an enabling technology for communicating data.

27. By the year 2000, the technology for the compression of all types of digital media had matured and a suite of algorithms had been developed by international organizations such as ISO and ITU to compress text, audio, images, and video digital content.

28. The term "encoding" broadly refers to converting data from one form

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

to another.[1] There are many forms of encoding depending on what is being encoded. An early example is the Morse code developed in 1830's for electrical telegraphy where alphabets and numbers are represented by dots and dashes. ASCII[2] code—another example of encoding—is the most common character encoding format for text data in computers and the internet, where alphabetic, numeric or special characters and control codes such as punctuation marks are uniquely encoded into a sequence of 8 binary digits (bits). The address of a webpage or the Uniform Resource Locator (URL), is transmitted over the internet using the ASCII code.

29. Image compression is another common form of encoding. In 1992, the now ubiquitous JPEG standard was developed to facilitate the efficient storage and transmission of digital images.[3] The JPEG standard outlined implementation guidelines for continuous-tone still image encoding and decoding processes.

30. Another common form of encoding is modulation whereby data is

[1] Appx. B, https://techterms.com/definition/encoding; https://www.merriam-webster.com/dictionary/encoding; https://www.techopedia.com/definition/948/encoding

[2] American Standard Code for Information Exchange (ASCII)

[3] Appx C, https://en.wikipedia.org/wiki/JPEG

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

converted into radio waves by adding information to an electronic or optical sinusoidal waveform called the carrier signal. Modulation is the enabling technology for Infra-red, Bluetooth, and Wi-Fi transmissions among others.

31. Encryption is another form of encoding. While encoding is the general operation of representing the data in a specific format, encryption is a specialized form of encoding where the encoded messages can only be accessed by authorized parties who hold the decryption key.

32. Another example of encoding is formatting of the data for transmission over the Internet using transmission protocols such as the TCP (Transmission Control Protocol). Protocols are rules that define how the data is formatted and transferred over a network. TCP, which is commonly used for transferring files between apps and devices, was developed in the 1970s and was adopted as the protocol standard for ARPANET (the predecessor to the Internet) in 1983. TCP, which is also used in IPSec[4], creates a secure communication line to ensure the reliable transmission of data. When sending packets using TCP/IP, the data portion

[4] IPSec stands for IP Security. It is a standard suite of protocols between two communication points across the IP network that provide data authentication, integrity, and confidentiality. It also defines the encrypted, decrypted and authenticated packets.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

of each IP Packet is formatted as a "TCP segment," where each segment consists of a header with multiple fields of varying sizes in addition to the data.

33. Finally, error control coding (sometimes referred to as channel coding) is another form of encoding used to provide reliable digital data transmission and storage when the communication medium can cause bit errors. For example, wireless communication systems depend on channel coding to ensure that the data received is the same as the data sent because wireless links suffer from interference and fading which cause errors.

A. Image Encoding and Compression

34. Compression is a form of encoding image and print data because it changes the data representation by taking advantage of statistical redundancies to reduce the size of the data representation or file.[5] The textbook "Data Compression in Digital Systems" by Roy Hoffman establishes that at least by 1997, encoding and compression was a mature technology and "not just a household word, but a household servant." (Appx. D at Preface vi). The goal of data compression is to find ways to represent information using as few bits as possible for storage and transmission. (*Id.* at 5) There are four main reasons to use data compression: (1) storage efficiency, (2) transmission bandwidth conservation, (3) transmission time

[5] Appx. E; https://en.wikipedia.org/wiki/Data_compression

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

reduction, increase data access rate. (*Id.* at 5)

35. While storage capability and transmission bandwidth for digital data continues to grow with technological development, so does the amount of data to be stored and transmitted. Geographically distributed databases and new applications involving digitized speech, audio, image and video required increasing amounts of storage and bandwidth. Digital system designers have long known that storage, bandwidth and time are always at a premium and cannot be wasted. As shown in Figure 1.2 below, data compression allowed these resources to be used more efficiently and reduced costs for storage and bandwidth. (*Id.* at 5).

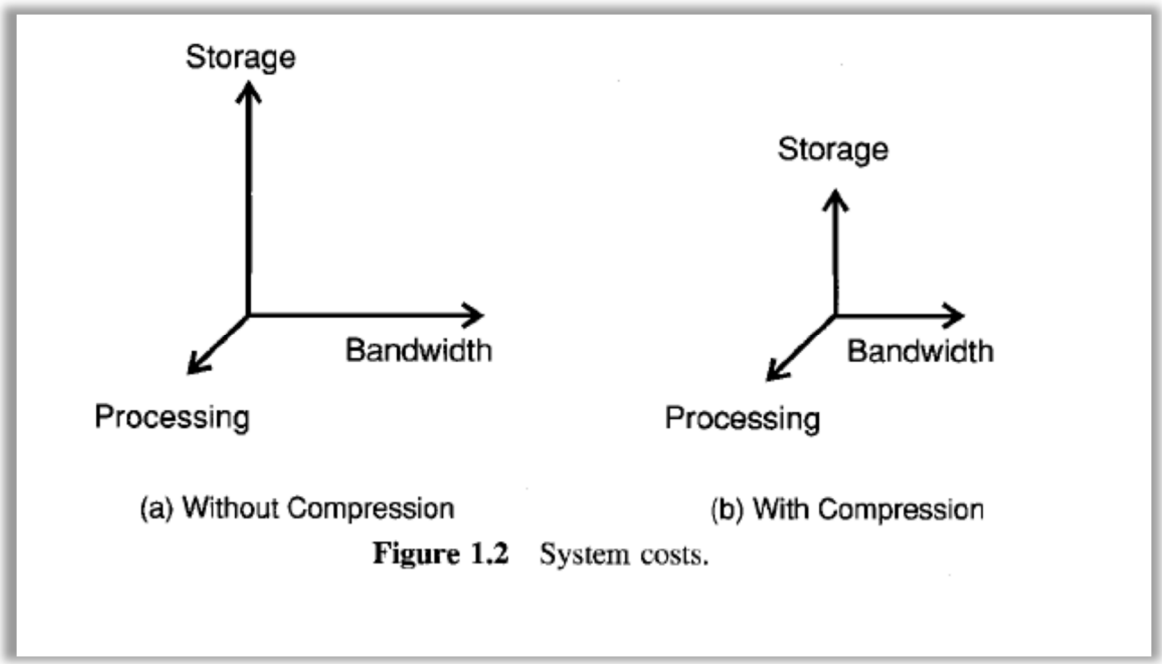


Figure 1.2 System costs.

Appendix D, Figure 1.2

36. Figure 1.4 is a representation of a digital system "that captures, encodes

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

and compresses data which it stores and/or transmits, it also decompresses, decodes, and displays or presents data to end users" (*Id.* at 11-12).

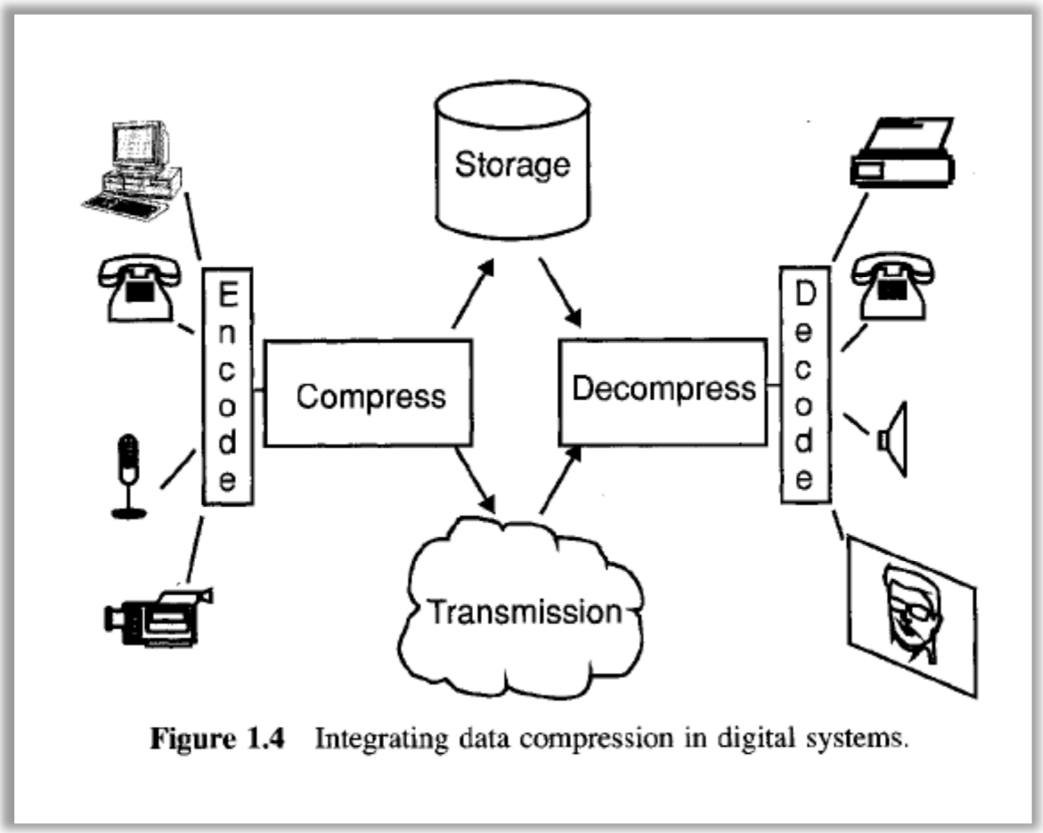


Figure 1.4 Integrating data compression in digital systems.

Appendix D, Figure 1.4

37. Figure 1.5 illustrates a client-server computer system where the client computers are connected to a network that delivers data from a server system.

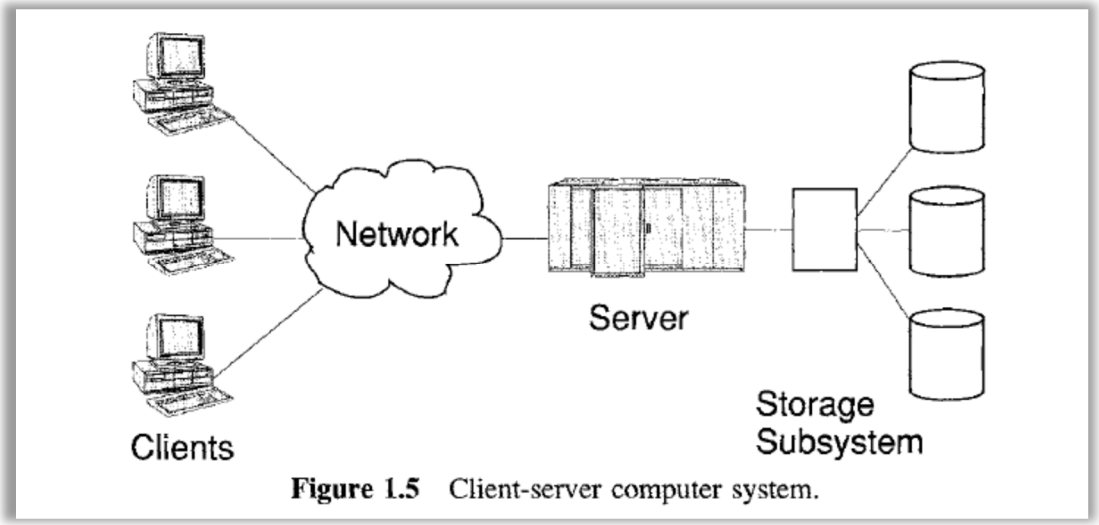


Figure 1.5 Client-server computer system.

Appendix D, Figure 1.5

38. The client server computing architecture was developed in the 1980s allowing client computers to communicate with servers that provided services and access to information stored on databases (such as remote printing). This architecture further enabled the processing workload to be moved off the client system (that is often resource constrained) and onto servers. (*Id.* at 13). In this distributed system, it was common for compression and decompression to take place multiple times between data capture and delivery. (*Id*).

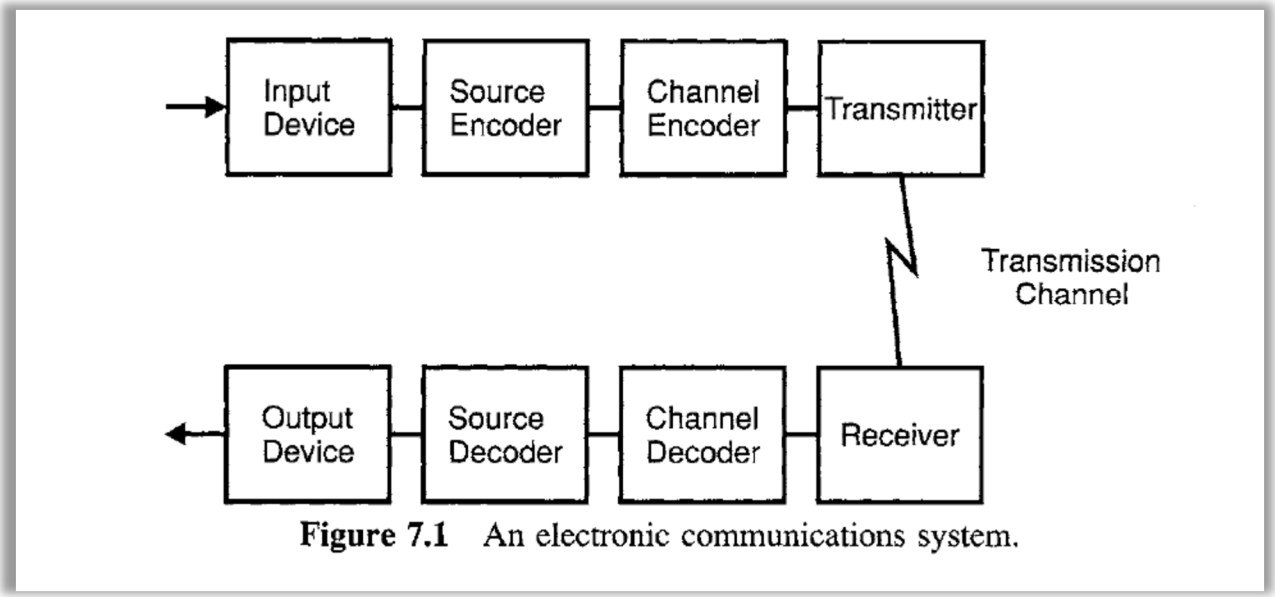


Figure 7.1 An electronic communications system.

Appendix D, Figure 7.1

39. Finally, Figure 7.1 depicted information conveyed along a transmission channel that may be wireless, wired, or a fiber-optic link. As Hoffman explains:

> The first task for any electronic communications system is to convert the information to be transmitted into electrical signals. This is accomplished by input devices such as microphones, computers, scanners, and cameras, where information in the form of voice (speech) or audio, computer data, images, and video are captured. Electrical signals from the input devices are processed in several steps: As previously stated, our interest is limited to digital electronic communications systems (for reasons that will soon become apparent). Digital systems transmit information as a sequence of encoded pulses (whereas analog systems transmit continuously varying signals). ***Therefore, the source encoder must perform analog-to-digital conversion and other transformations-including data compression-to efficiently encode the data.*** Then the data must be further processed by the channel encoder into a form suitable for transmission. This processing includes adding error-detection or error-correction data to compensate for transmission errors and conversion to an electrical signal that the transmitter can accept. In a wired

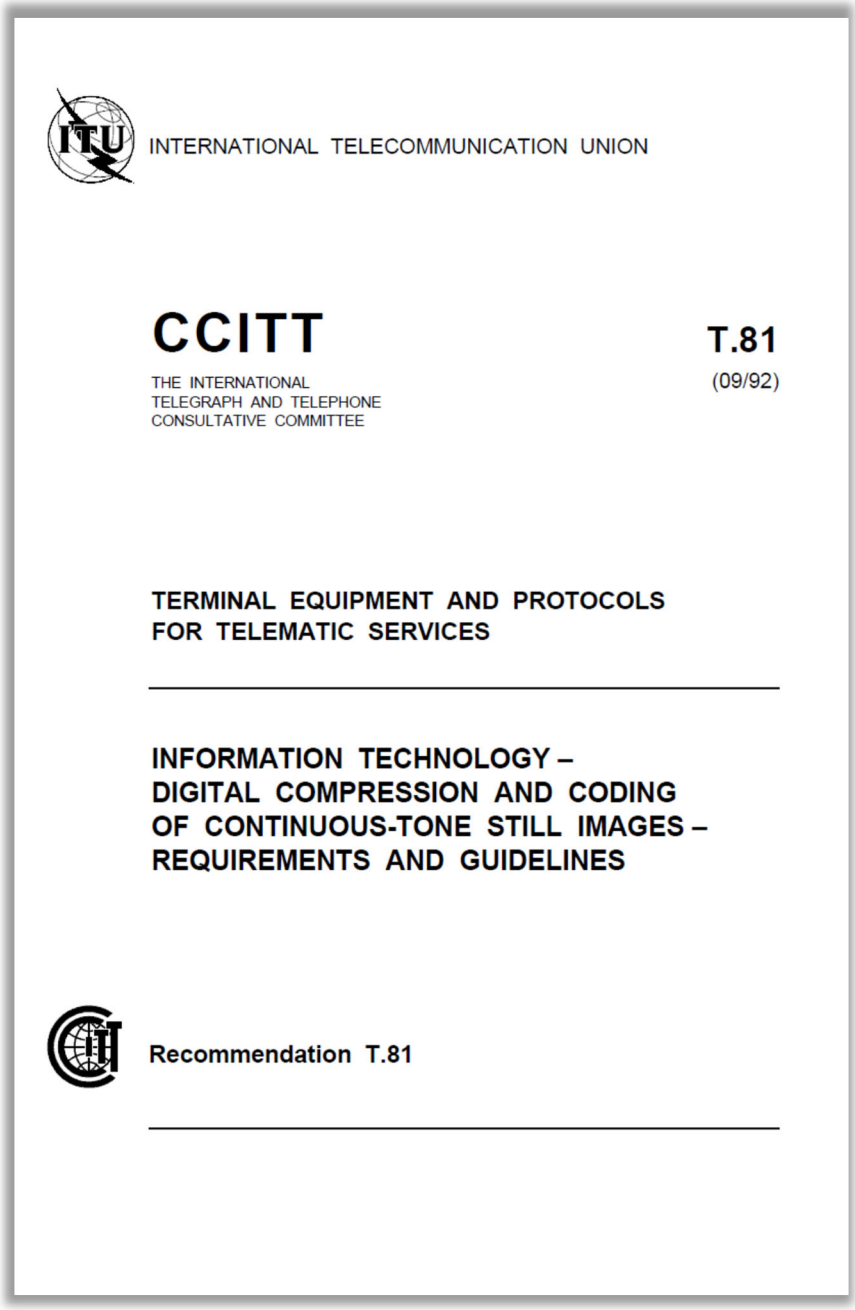
Lexmark, Exhibit 1002
Lexmark v. Flexiworld

> system, the transmitted signal takes the form of an electrical current that travels along a conductor. In a wireless system, the signal is sent through space as an electromagnetic wave launched by the transmitter. In an optical system, the electrical signal is converted to light that is guided by the glass fiber.
>
> At the receiver, the signal is captured and all the processing steps are undone, allowing the transmitted information to be recovered and reproduced. Output devices include computers, loudspeakers, printers, and image or video display devices. If all goes well, the recovered information will be an exact or acceptably faithful reproduction of the original.

(*Id.* at 178).

40. By 2000, various image encoding and compression schemes had been developed and standardized. One of the most recognizable and prominent examples of an international image compression standard is JPEG ("Joint Photographic Experts Group"). JPEG was developed under the joint auspices of the three major international standards organizations, ISO, ITU, and IEC, and was approved in September 1992 as ITU-T Recommendation T.81 and titled "***Digital Compression and Coding of Continuous-Tone Still Images - Requirements and Guidelines***." (Appx. F, JPEG standard T.81).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

T.81
(09/92)

TERMINAL EQUIPMENT AND PROTOCOLS
FOR TELEMATIC SERVICES

INFORMATION TECHNOLOGY –
DIGITAL COMPRESSION AND CODING
OF CONTINUOUS-TONE STILL IMAGES –
REQUIREMENTS AND GUIDELINES

Recommendation T.81

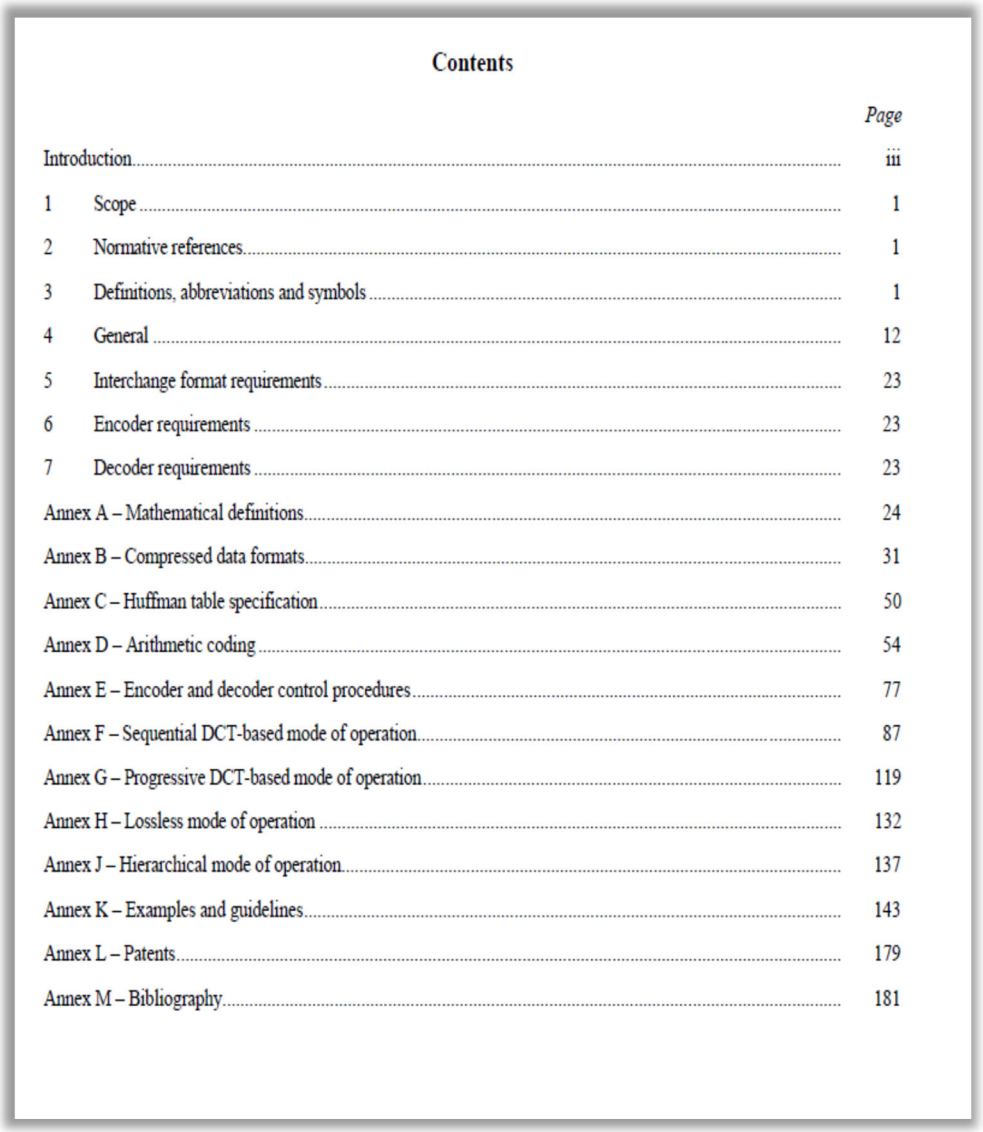
JPEG Standard T.81 Cover (1992)

41. JPEG was developed as a standard for the encoding of continuous tone grayscale and color images. By the year 2000 owing to the economies of scale, inexpensive hardware and software implementations of JPEG were prevalent and

the JPEG standard was almost exclusively used to encode still images in any aspect of digital image acquisition which employed compression. I note here that the very title of the JPEG standard included "compressing" and "coding" establishing that a POSITA would have understood that these two technologies are essentially intertwined, and a POSITA would have understood that compression represents at least one type of data encoding. This is further established by the definitions presented in Section 3.1 where "encoding process" is defined as "***a process which take as its input a continuous-tone image and outputs a compressed image data***." (*Id.* at 7). (emphasis added).

42. Similarly, as shown below, the JPEG standard Table of Contents included: encoder requirements, decoder requirements and compressed data formats. (*Id*. at Table of Contents).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



Contents

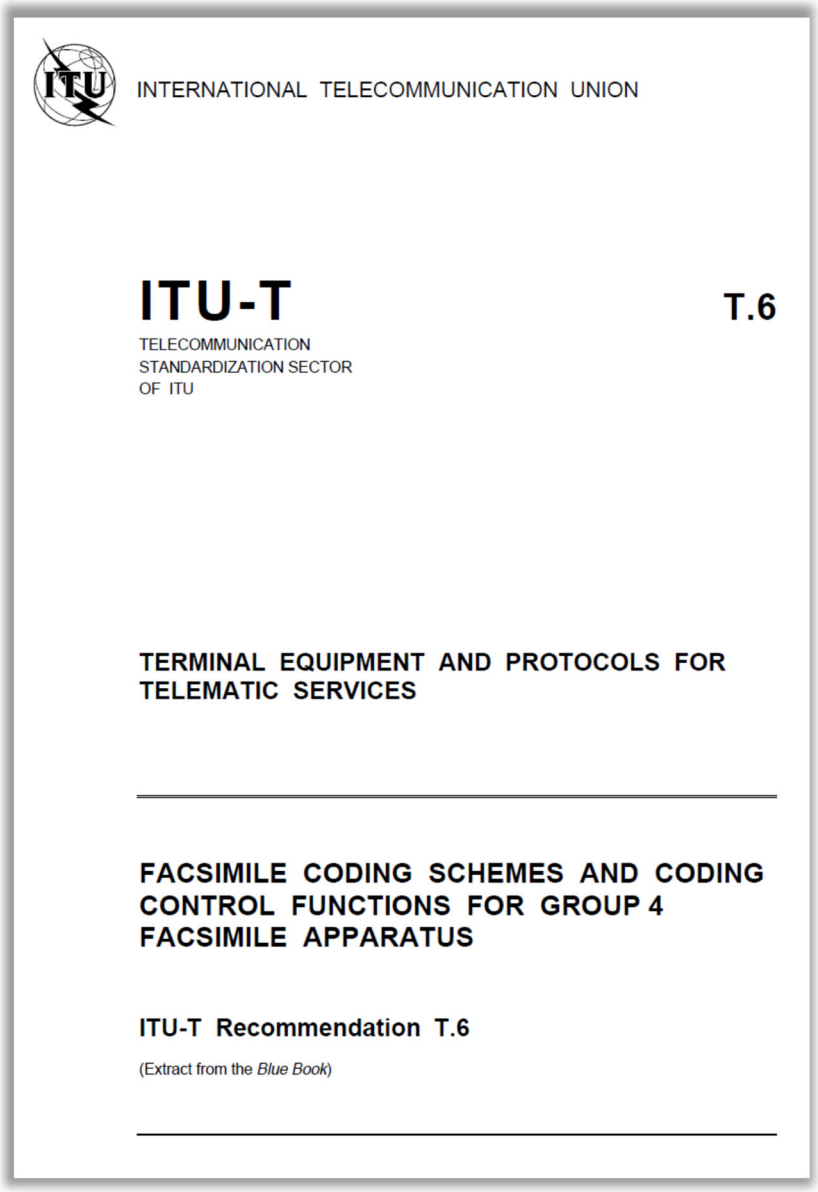
		Page
Introduction		iii
1	Scope	1
2	Normative references	1
3	Definitions, abbreviations and symbols	1
4	General	12
5	Interchange format requirements	23
6	Encoder requirements	23
7	Decoder requirements	23
Annex A – Mathematical definitions		24
Annex B – Compressed data formats		31
Annex C – Huffman table specification		50
Annex D – Arithmetic coding		54
Annex E – Encoder and decoder control procedures		77
Annex F – Sequential DCT-based mode of operation		87
Annex G – Progressive DCT-based mode of operation		119
Annex H – Lossless mode of operation		132
Annex J – Hierarchical mode of operation		137
Annex K – Examples and guidelines		143
Annex L – Patents		179
Annex M – Bibliography		181

JPEG Standard T.81 - Table of Contents (1992)

43. Another coding standard is the 1984 ITU-T Recommendation T.6 CCITT Group 4 "Facsimile Coding Scheme and Coding Control Functions for Group 4 Facsimile Apparatus." (Appx. G). CCITT Group 4 compression is a lossless method of image compression used for facsimile machines and black and white images and is another example where coding and compression are interchangeably

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

used.



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

T.6

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

TERMINAL EQUIPMENT AND PROTOCOLS FOR TELEMATIC SERVICES

FACSIMILE CODING SCHEMES AND CODING CONTROL FUNCTIONS FOR GROUP 4 FACSIMILE APPARATUS

ITU-T Recommendation T.6

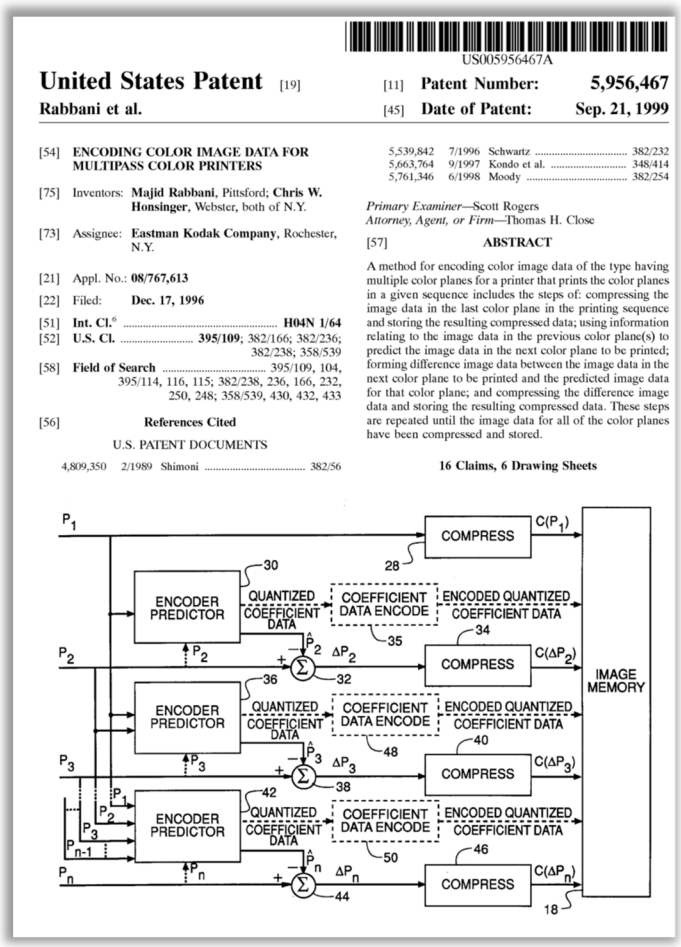
(Extract from the *Blue Book*)

ITU-T Recommendation 1.6 Cover (1984)

44. In addition to these standards, there are countless patents that addressed coding and compression for either storage or transmission of image data in a print environment. For example, during my tenure at Kodak, I was a named inventor on

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

44 patents. One patent in particular as shown below is U.S. Patent No. 5,956,467 filed on December 17, 1996 and titled "Encoding Color Image Data for Multipass Color Printers." (Appx. H at Title).



US005956467A

United States Patent [19]
Rabbani et al.

[11] **Patent Number: 5,956,467**
[45] **Date of Patent: Sep. 21, 1999**

[54] **ENCODING COLOR IMAGE DATA FOR MULTIPASS COLOR PRINTERS**

[75] Inventors: **Majid Rabbani**, Pittsford; **Chris W. Honsinger**, Webster, both of N.Y.

[73] Assignee: **Eastman Kodak Company**, Rochester, N.Y.

[21] Appl. No.: **08/767,613**

[22] Filed: **Dec. 17, 1996**

[51] **Int. Cl.**[6] **H04N 1/64**

[52] **U.S. Cl.** **395/109**; 382/166; 382/236; 382/238; 358/539

[58] **Field of Search** 395/109, 104, 395/114, 116, 115; 382/238, 236, 166, 232, 250, 248; 358/539, 430, 432, 433

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,809,350	2/1989	Shimoni	382/56
5,539,842	7/1996	Schwartz	382/232
5,663,764	9/1997	Kondo et al.	348/414
5,761,346	6/1998	Moody	382/254

Primary Examiner—Scott Rogers
Attorney, Agent, or Firm—Thomas H. Close

[57] **ABSTRACT**

A method for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence includes the steps of: compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data; using information relating to the image data in the previous color plane(s) to predict the image data in the next color plane to be printed; forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane; and compressing the difference image data and storing the resulting compressed data. These steps are repeated until the image data for all of the color planes have been compressed and stored.

16 Claims, 6 Drawing Sheets

Rabbani Patent - Encoding Color Image Data for Multipass Printers

45. This patent is another example of how a POSITA well before 2000 would have understood compression represented a type of data encoding when preparing image data for either storage or transmission. The patent described that both low-cost and high-cost printers encounter problems when printing multiple colors. For example, in low-cost printers significant delay occurs due to data

buffering issues and network bandwidth constraints when providing image data to the printer. (*Id.* at 1:32-51). Similarly, for higher-end color printers containing sufficient memory, it was desirable to "discard the data related to each color plane of the current image once it has been printed, so that the memory thus made available can be reused to download the color planes of the next image in the printer queue. (*Id.* at 1:52-62). As my patent explained, in the prior art each color plane was independently compressed "using a compression scheme such as lossy or lossless JPEG compression standard. (*Id.* at 1:63-2:2). Therefore, I saw a need to improve image compression for the higher quality printers to minimize the cost of printer memory. (*Id.* at 2:35-41). With reference to JPEG, my patent described how the various steps of transformation, quantization, and symbol encoding contributed to the amount of compression and the resulting image quality achieved by JPEG.

> In general, a lossy compression scheme such as the JPEG standard includes three steps: transformation or decomposition, quantization, and symbol encoding. The transformation step reduces the correlation or the redundancy in the data to make it more amenable to compression. The quantization step, which is where the loss in lossy compression is introduced, maps ranges of input values into a single quantized value and thus reduces the size of the data to be encoded. Finally, the symbol encoding Step translates the quantizer output values into a string of bits (binary "1" and "0" digits).

(*Id*. at 4:8-18). According to the invention, color correlation reduction was achieved by ensuring that redundant information in a given color plane was reduced and

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

compressed to improve performance. (*Id*. at 4:19-26). Figures 4 and 5 depicted schematic diagrams of the inventive encoder and decoder.

46. While my '467 patent is directed to encoding and compressing image files for storage in the local printer memory and not transmission, a POSITA would have understood that the same benefits of JPEG compression extended to transmission.

47. The need for compression when transmitting images over a communication link is further described in **U.S. Patent No. 6,608,928** to Queiroz. (Appx. I). For example, the patent explained that:

> 2. Description of Related Art
>
> Data compression is required in data handling processes, where too much data is present for practical applications using the data. Commonly, compression is used in communication links to reduce the transmission time or required bandwidth. Similarly, compression is preferred in image storage systems, including digital printers and copiers, where "pages" of a document to be printed are stored temporarily in precollation memory. The amount of media space on which the image data is stored can be substantially reduced with compression. Generally speaking, scanned images, i.e., electronic representations of hard copy documents, are often large, and thus make desirable candidates for compression.

(*Id*. at 1:10-22)

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

> It should be appreciated that the compression systems and methods of this invention can be applied to any set of compression techniques for separately processing the various types of data contained in a composite image. Accordingly, the compression/decompression methods and systems of this invention can be applied to any number of systems, including digital printers and copiers, that need to provide compressed or decompressed images. Systems implementing this invention can include data or image processing systems capable of compressing or decompressing images. The systems and methods described herein are efficient and result in smoother boundaries between regions of different image data types.

48. (*Id*. at 2:16-29). The patent further explained that the image data is transferred to a channel including a network, Internet or wireless channel or to a storage.

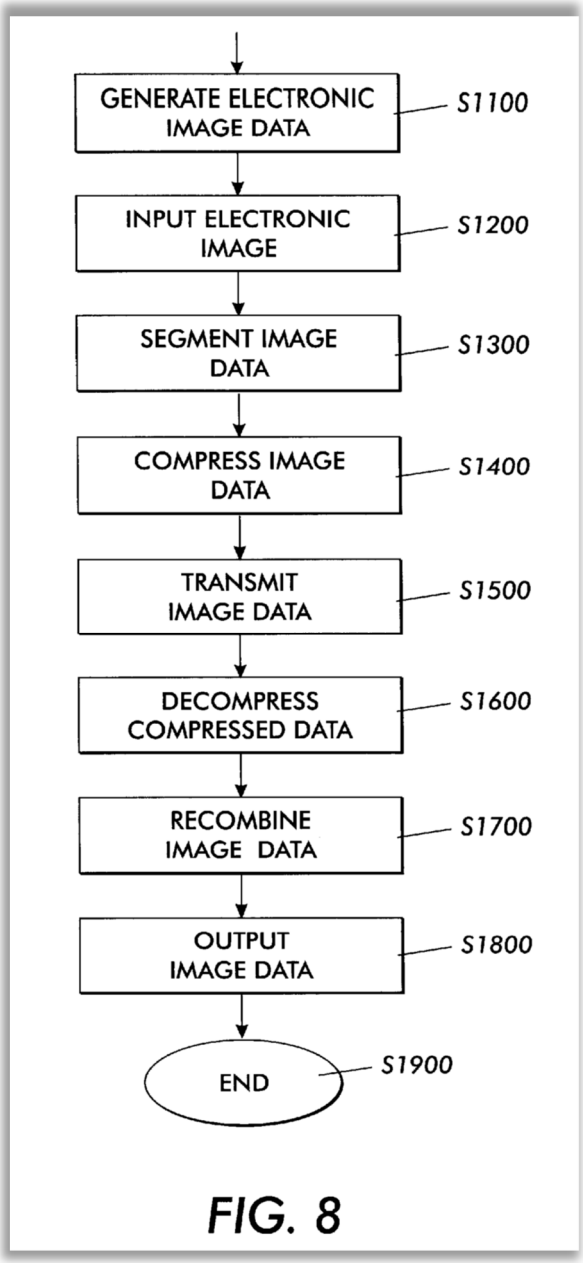
> Once compressed, each plane of the image data then is preferably transferred to the channel or storage device **300**. The channel or storage device **300** can be either or both of a channel device for transmitting the compressed image data to the decoder **500** or a storage device for indefinitely storing the compressed image data until there arises a need to decompress the compressed image data. The channel device can be any known structure or apparatus for transmitting the compressed image data from a first apparatus implementing the encoder **400** according to this invention to a physically remote decoder **500** according to this invention. Thus, the channel device can be a public switched telephone network, a local or wide area network, an intranet, the Internet, a wireless transmission channel, any other distributed network, or the like.

(*Id*. at 3:24-39).

49. Finally, Figure 8 below showed a flowchart outlining an image compression and decompression method of the inventive embodiment. Particularly,

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

step S1400 shows compressed image data is generated and step S1500 the "compressed image data is transmitted and possibly stored prior to transmission to a device for decompressing the compressed data." (*Id*. at 10:64-11:1-2).



Quieroz, Figure 8 - Image Compression Flowchart

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

50. Against this background and evidence, it is my opinion that encoding image data for its efficient transfer over a communication link via compression was well-known and even ubiquitous in the printer and digital data industry long before 2000.

B. Division of Labor and Universal Printing:

51. As discussed above, by 2000 printing technology was quite mature and different printing technologies, such as electrophotographic, dye sublimation, inkjet, and others were part of the printer landscape. With the proliferation of networks and associated technologies, many remote printing options, both wired and wireless, became available. And, as a natural consequence, different options were available for how to divide the print processing operations between the computer (or mobile device), the printer, or some other processing node that could perform the processing operations. Some factors besides the computational complexity of the RIP operations that could impact the distribution of the tasks included the communications range, data transfer rate, error rates, and bandwidth costs of the various connections in the printing network.

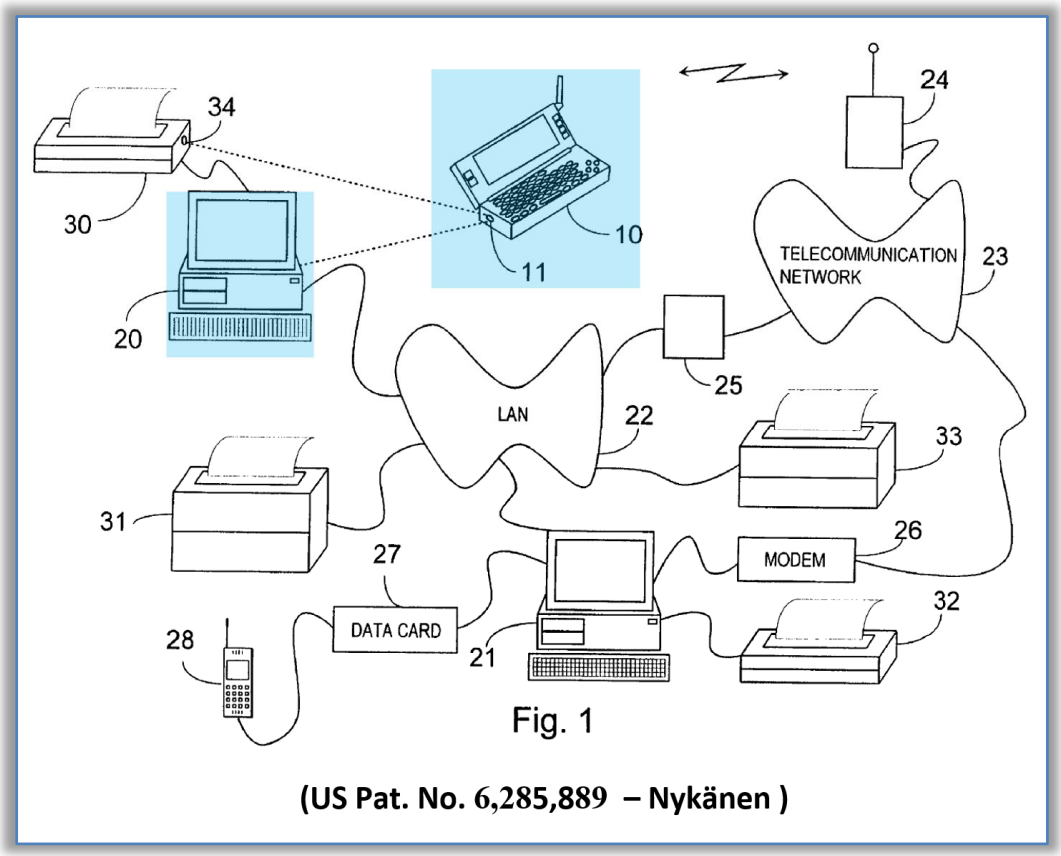
52. Another consideration in remote printing related to knowing which printer, or printer type, was available, and processing the digital information and delivering it to the printer in a format that the printer understood. This was typically done by using a device driver specific to a printer, which allowed ripping the data in

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

a format acceptable to that printer. So, it was also important to have ready access to the proper print driver and/or to be able to obtain the print driver or its functionality.

53. Prior to 2000, both of the above issues had been addressed, some examples of which are described below.

54. **U.S. Patent No. 6,285,889** to Nykänen described a system where data to be printed was converted at mobile device (10) into a data transfer format that was independent of the application and was wirelessly sent to another terminal (20), which converted the data into a form suitable for an output device (30) that could be a printer or a monitor (Appx. J at Abstract, 8:20-35).



Nykänen, Figure 1 - Remote Printing System

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

55. Nykänen expressly described using JPEG-compression methods to facilitate the transfer of large pictures of good quality and for utilizing the network's bandwidth effectively, explaining that:

> An essential part of page description languages is also the possibility to include separate pictures in the pages, which pictures have been coded using some other coding method, such as e.g. JPEG-compression method. JPEG (Joint Photographic Experts Group) is an ISO standard for encoding fixed pictures and it facilitates transferring large pictures of good quality utilizing data transfer band effectively. The compression ratio can be adjusted using picture quality as a criterion, and thus JPEG-standard is very versatile. **JPEG is well suited for information output systems according to the invention utilizing independent data transfer format 44, in which systems the capacity available for data transfer can be very different in different operating Situations: by radio in GSM-system it may be that only 9600 bit/s is available, whereas in infrared connections it is possible to use e.g. speeds 1.152 or 4 Mbit/s.**

(*Id.* at 11-40-55, emphasis added).

56. **U.S. Patent No. 7,028,102** to Larsson, filed on February 17, 2000 and titled "Method and System For Presenting Information," is another example of a remote printing environment. (Appx. K). Larsson taught a networked printing paradigm that consisted of up to 4 nodes: (i) a handheld device, e.g., a cellphone, (ii) a presentation unit, e.g., a printer/display with a communication device, (iii) an information service provider, e.g., a content database, and (iv) a device capable of handling presentation requests.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

57. A user wishing to use his handheld device to send selected content from an information service provider to a presentation device would establish a short-range wireless communication with the printer to acquire its specifications. These specifications, together with a pointer to the selected content, were communicated via a wireless protocol to the device that handles presentation requests, which converts the selected content to a format suitable for the presentation unit (i.e., a device-dependent output) and sent the converted document to the communication device of the presentation unit via a network. Larsson recognized the need to save transmission bandwidth which it achieved by the proposed routing of his invention. (*Id*. at 7:3:36-39).

58. **U.S. Patent No. 7,644,018 to Yukie** is directed to a "System and Method For Providing Publicly Vended Content Via a Wireless Network." (Appx. L). Yukie's Figure 1 disclosed a system having a user terminal, a wireless network connection, a database of publicly vended content, a connect server for communicating with the user terminal and database, and a billing module.

59. Yukie described a system for delivering digitized content in response to user requests and by using Internet Protocols. The user was billed in various ways, such as by content piece, length of time or a subscription (*Id*. at Abstract). A user terminal 32, including a mobile wireless device, communicated over a network path to a connect server 26. The connect server obtained content from a content database

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

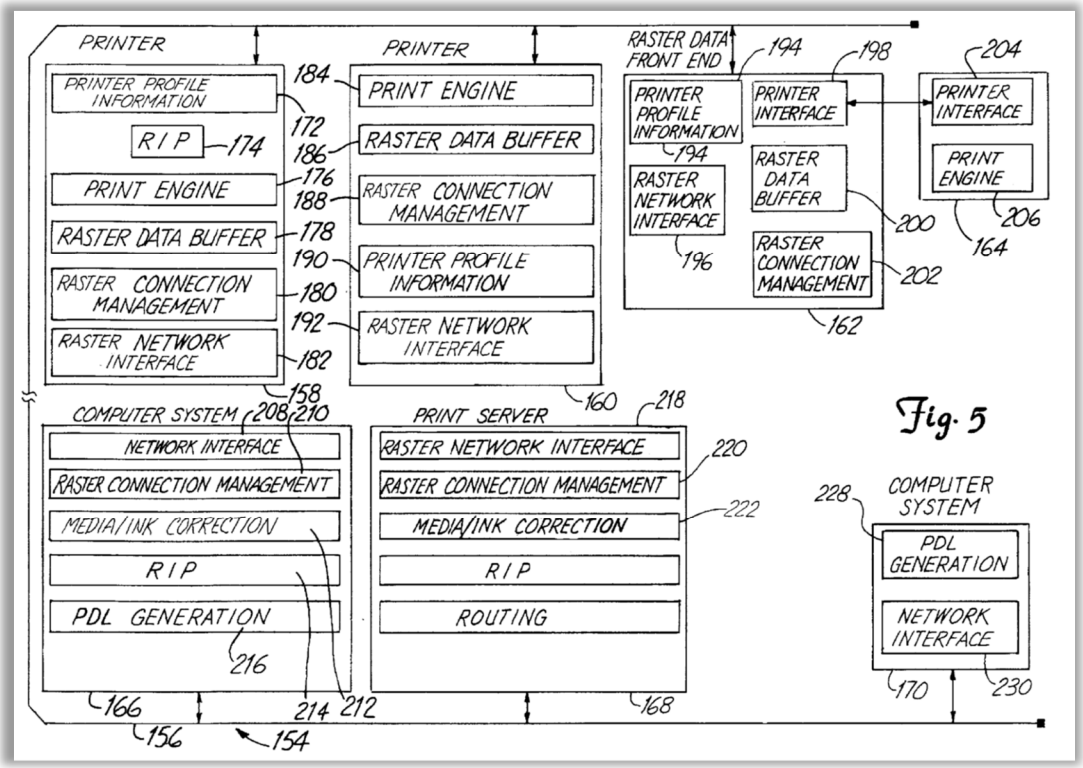
14 or content providers 12 or provided the content itself. Yukie described that the server received and verified logon information including the user's ID and password, determined the type of terminal device the user is using, and provided content at a rate compatible with the terminal. Preferably, the content was provided to the user terminal over the network in IP packets and over a "private, secure tunnel channel." (*Id*. at 6:8-12).

60. The user terminal sent a request to the connect server for accessing digital content. The connect server received the logon and verified the user's identification and password if provided. The content server accessed the database of publicly vended content and fulfilled the request. A billing module is accessible to the connect server for generating accounting data based either on the specific request or, for example, subscription, which is established at least in part by a user profile. The connect server determined the type of device that the user is using and sends the content to the terminal at a rate compatible with that type of device. Preferably, the content was provided to the user terminal over the network in IP packets.

61. **U.S. Patent No. 6,091,507 to Vatland** *et al.* was directed at a method and apparatus for printing a document over a network wherein a host computer generated a raster image from a series of page description language instructions of the document and transferred the raster image over a network to one or more printers where it is printed. (Ex. 1005 at 1:5011). Figure 5 shows a networked computing

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

system having a conventional network and raster network sharing the same physical network 156 along with printers, 158, 160, 164, raster data front end 162, computer systems 166 and 170 and print server 168.



Vatland, Figure 5 - Networked Computing System

62. In reference to the "raster connection management module 210," Vatland explains that it compressed raster data, decompressed raster data, encrypted raster data, and decrypted raster data for transmission over network connections as is known in the art. (*Id*. at 8:19-23, 11:15-30 (stating that "the data may be compressed and decompressed to obtain additional network bandwidth.")).

Printing Workflow

63. Printing usually contains two basic sets of operations. One is the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

generation of a Page Description Language (PDL) file where the content to be printed is converted to a format that defines the page elements independently of the printer technology so that a page's appearance would stay consistent regardless of the specific printer used. Another element is the Raster Image Processing (RIP) set of operations that convert a PDL file to a printer-specific bitmap image ready to be printed.

64. Most printers treat printed pages as a rectangular array of pixels (e.g., a bitmap image) with a certain width and height and bit depth, which could be very large depending on the device type. However, this may not be a convenient representation for storing, editing, and transferring such files over the distributed network. PDLs describe the appearance of a page at a higher level than an array of pixels and serve as a link between computer applications, such as graphics tools and text editors, and output devices, such as printers and displays. PDLs are text (human-readable) or binary data streams, usually intermixed with text or graphics to be printed.

65. Some PDLs were invented by printer manufacturing companies. For example, PCL (HP printer language) was invented by Hewlett-Packard and AFP (IBM and Ricoh printers' language) was devised by IBM. Others were developed by software companies, such as Adobe's PostScript (PS) and PDF (Portable Document Format) languages. Every company that invented its own PDL also developed a

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

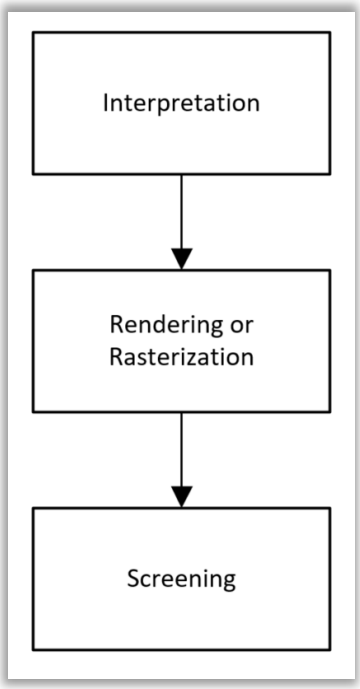
corresponding device specific interpreter, often referred to as a "driver," which translates high-level PDL to an array of pixels to be transferred to the native low-level software of a specific printer.

66. A RIP operation manages the translation of a high-level PDL such as Adobe PostScript, PDF, or HP PCL to printer language, e.g., a bitmap image with the proper spatial resolution, bit depth and color, ready to be printed by a specific printer. In simple terms, it translates graphics files to an image that a printer can understand and print. The RIP set of operations can be performed either in hardware or in software, but they are typically performed in software.

67. The RIP set of operations vary depending on the specific printer but generally include the following. (i) Interpretation: During this stage, the PDL file is decoded into a printer specific internal database of graphical elements that must be placed on the output. (ii) Rendering or "rasterization": During this stage, the image is converted into a rectangular array of dots (pixels) that has a printer specific supported spatial resolution, and color and tonal information are assigned to each dot. In many practical applications, rendering is performed simultaneously with the interpretation stage. (iii) Screening: At this stage, the image is converted into a halftone pattern that can be printed using limited ink colors. For example, since the ink of an inkjet printer cannot be diluted during printing to create a lighter shade, halftones are used to control grayscale and color by printing various sizes of tiny

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

dots, spaced closely together to create the appearance of continuous tones. An example of a popular half-toning algorithm is error diffusion. Finally, the RIP software sends the processed image to the printer for printing.



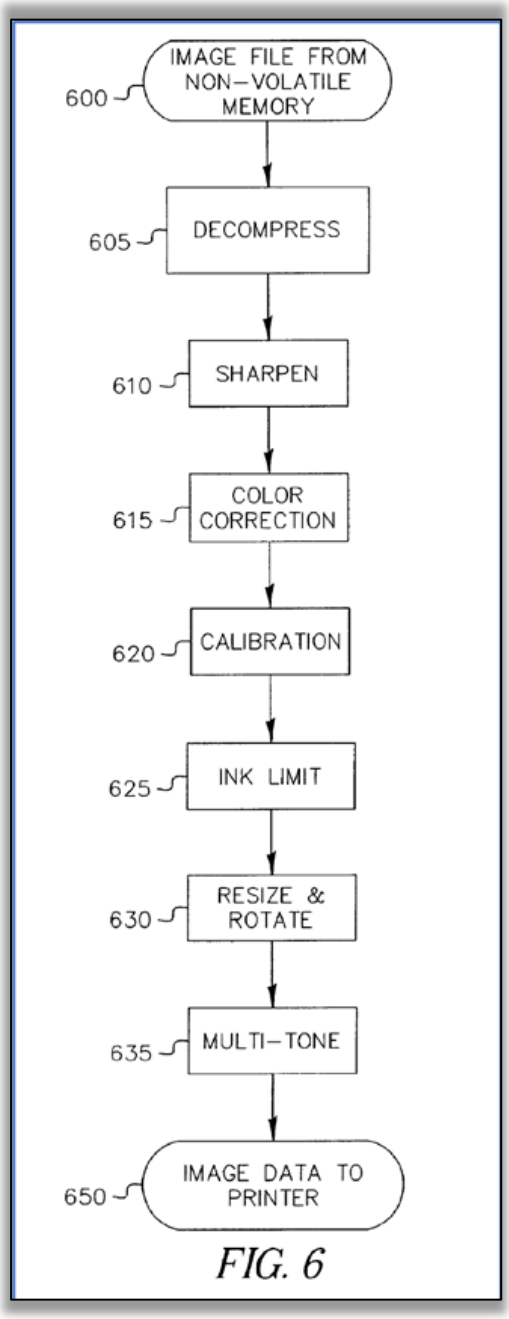
Typical Printing Process Flow

68. The RIP stage can also have in its input images in their native formats such as JPEG or TIFF and it might also include other operations such as encryption/decryption, compression/decompression, encoding/decoding, scaling, segmentation, color space conversion and correction, and contrast and brightness adjustment. In general, the RIP operations are computation intensive.

69. Figure 6 of **U.S. Patent 7,212,229** to Parulski described a typical set of image processing operations to prepare an image for sending to an inkjet printer.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

(Appx. M at 3:37-43, 7:31-10:2). These operations were well understood at the time the ’233 Patent was filed and, as explained below, they could be done by different components of a printing system.



Parulski, Figure 6

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

70. Given the prevailing distributed network paradigms that were known by the year 2000, the need to distribute the computational load of the printing process among the various nodes of the network based on their capabilities would have been known to a POSITA.

V. THE '233 PATENT

A. Overview of the Alleged Invention

71. The '233 patent (Ex. 1001) is based on Application No. 09/992,413 filed on November 18, 2001 with two listed inventors, William Ho Chang and Christina Ying Liu. The Abstract explains that the patent is directed to streaming protected digital content to client or output devices:

> Digital content services provided over the Internet for transmitting or streaming protected digital content to information apparatus or output devices are herein disclosed and enabled. The protected digital content is encrypted and device dependent or specific for rendering at a limited number of devices including smart phones, information pads, audio players, media controllers, televisions, printers etc. To illustrate, an information apparatus may include an application for accessing selected digital content from a subscribed content service or store over the Internet. To access digital content, a client application may send, to the content service or store, authentication information, payment or subscription information, and/or a reference to selected digital content. The service then generates encrypted device dependent output data associated with a device or identification previously registered with the service for transmitting or streaming, to the information apparatus or output

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

> devices (e.g., televisions, speakers, media controllers), the digital content for rendering.

72. The applicants of the '233 patent recognized that information apparatuses, particularly mobile information apparatuses (e.g., PDAs or smartphones) can present limitations in outputting digital data pervasively, because of reduced processing power, small screen size, and limited memory space. '233 patent at 1:11-16. The '233 Patent describes in the Background and Summary of the Invention that output devices can be video monitors, printers or audio devices. '233 31-35.

> "any one or more of fax machines, printers, copiers, image and/or video display devices (e.g., televisions, monitors and projectors), and audio output devices."

73. To overcome these and other limitations of the information apparatus, the alleged invention of the '233 patent allows the information apparatus to output data to an output device so that the user can enjoy the "full richness" of the original document content. *Id.* at 4:4-7.

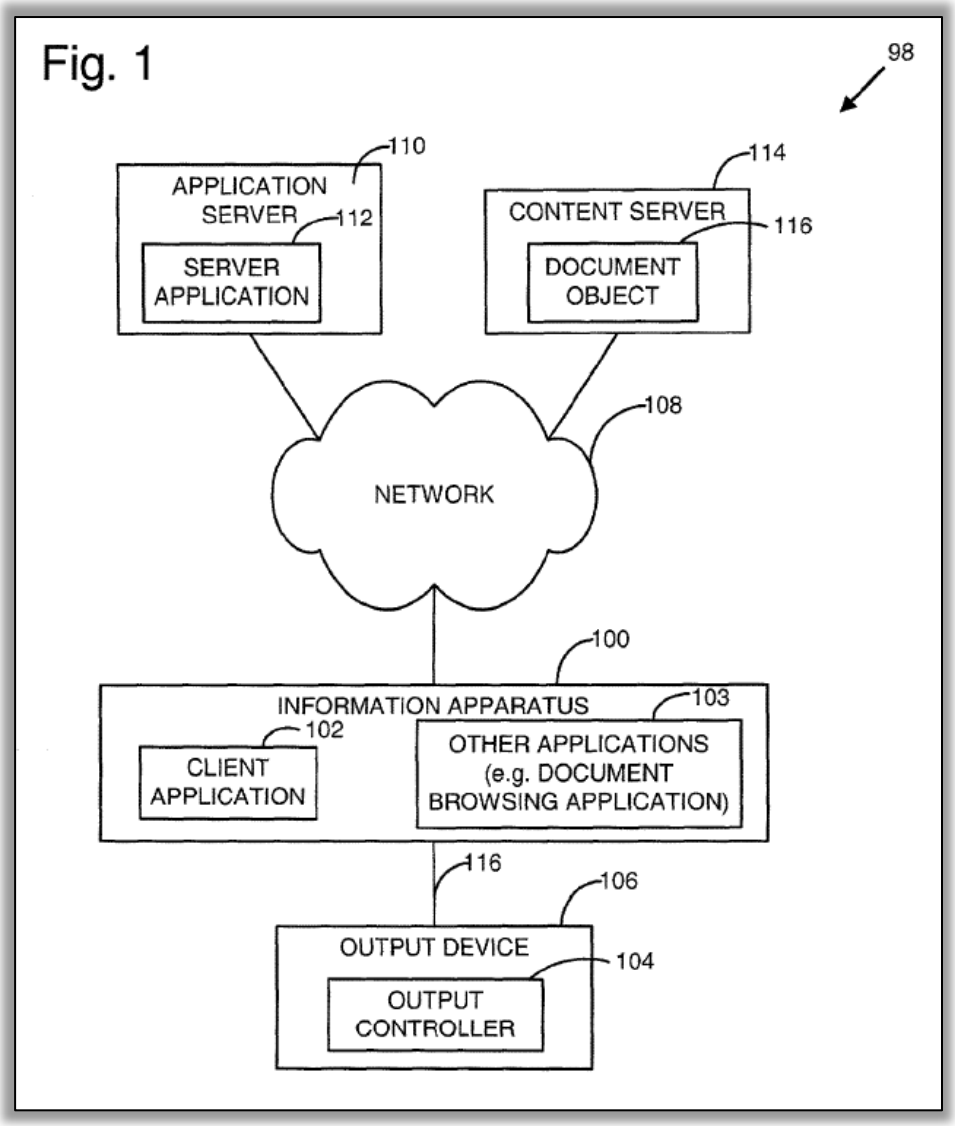
74. As shown in Figure 1 (reproduced below), the alleged invention consists of one or more servers that communicate with an information apparatus via a network. See *id.* at 8:7-15. The information apparatus also communicates with an output device to output data that represents an original document or file. *Id.* at 8:44-47. In the '233 teachings, application server includes software to implement the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

functionalities of the server and provide services to the client applications on the information apparatus. See *id.* at 16:62-66.

75. Server applications may receive authentication information, such as a password, user name, id number, biometric information, digital certificate or security key, when necessary. In the '233 teachings therefore, passwords and user names are types of authentication information. See *id.* at 26:1-14.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



'233 Patent, Figure 1

76. As disclosed in the Abstract, a user operating a client device (i.e., information apparatus) may desire to access protected digital content that is output at an output device. To access the protected digital content, the client device may send to a content service implemented by one or more servers, authentication information, subscription information, and/or a reference or pointer to the digital

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

content selected by the user. The client device may also send one or more device objects that contain information about the output device, such as an IP address. The digital content service generates encrypted device-dependent output data related to the content selected by the user, which is transmitted to the client device that sends the output data to an output device (either with or without any additional processing).

77. To illustrate the alleged invention encompassed by the ‘233 patent, the specification gives examples of the claimed information apparatus and system in the printing context. *Id.* at 1:35-44. For example, a user could access a news article published on the Internet on their information apparatus (e.g., a smart phone), but the small display screen, limited processing power, or lack of needed software or communication protocols could render the information apparatus unable to view the entire article. *Id.* at 32:38-50. The user could access the documents through a document browsing application by indicating to the client application a pointer or reference to the output content. For example, the user could select the output content through a GUI provided by the documents browsing application. Therefore, in the teachings of the ’233 patent, examples of pointers or references include the user selecting content through a GUI. *Id.* at 25:44-56.

78. But the information apparatus may have a printing function available so that the user can print the full article. The user may invoke a print function on the information apparatus, which launches a client application. *Id.* at 32:51-33:11. The

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

client application will search for available printers via a discovery process or allow the user to select a known printer. *Id.* at 33:11-26. If a printer is found or specified, the user may select that printer as the desired output device. *Id.* at 33:26-37. In order to print the news article, the information apparatus may need to provide a reference to the document for output, the selected output device, and authentication and/or subscription information. An output device object may contain one or more attributes which describe the capabilities of a particular device, such as a printer or monitor and define the service provided by the output device. *Id.* at 5:62-67, 6:6-10.

79. A user profile may save payment information or if the service is a subscription collect authorization information. *Id.* at 33:26-34:3. After this information is provided and a job object is submitted, the information apparatus communicates with one or more servers via a composite message which may include a document object, a printer or device object and a job object. The server fetches output content based on the document object provided by the information apparatus. *Id.* at 34:3-32. The server may process or encode the output content, for example perform raster image processing operations, before sending the output data to the printer via the information apparatus, which may or may not perform additional data processing. *Id.* at 34:32-60.

80. The ‘233 patent does not teach any particular encoding techniques nor does it teach any particular encryption techniques done by a server.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B. Prosecution History

81. I have reviewed the prosecution history of the '233 patent. The applicant filed Application No. 09/992,413 on Nov. 18th, 2001. (Ex. 1003.)

82. The prosecution of the '233 Patent was lengthy. The Applicant repeatedly amended the claims after allowance. After a restriction requirement and several sets of claim rejections based primarily on references directed to either printer servers or camera phones, the Examiner finally issued a Notice of Allowance on June 19, 2014. The Notice (at 24-25) identified the primary reason for allowance as the lack of teaching in the prior art of a "content object including information associated with a pointer or reference to a selected digital content… associated with a selection by the user… from at among the at least one or more pointers or references." After the Notice of Allowance, Applicants filed amendments at least another ten times, each time resulting in another Notice of Allowance, before the '233 Patent finally issued. At no time did the Examiner substantively discuss any reference substantially similar to the combination of references discussed in this Petition.

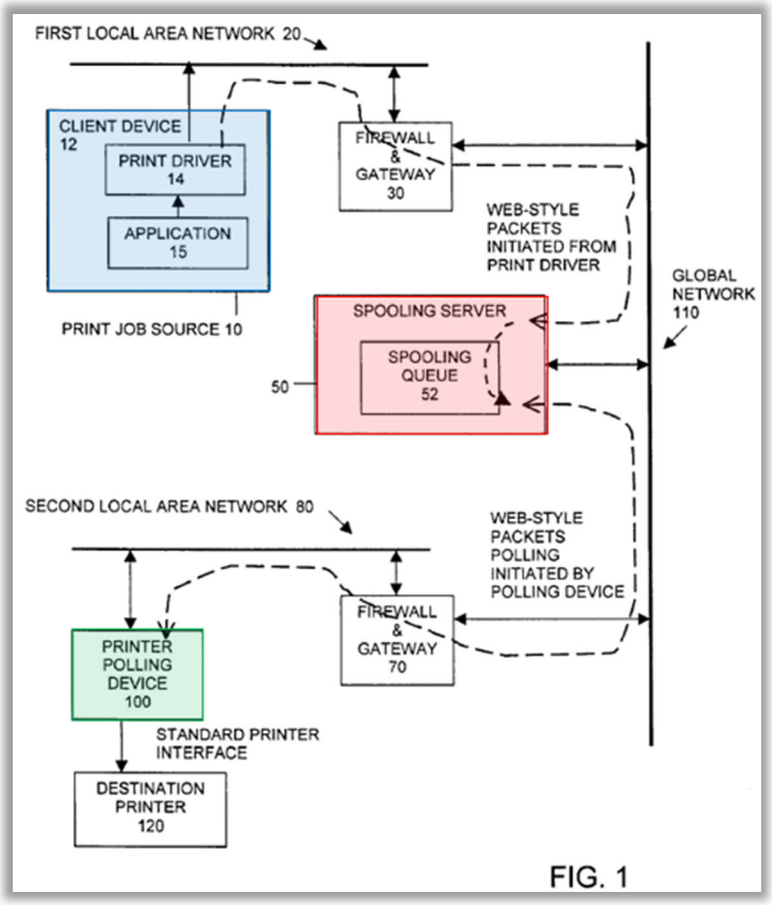
VI. DETAILED DISCUSSION OF KEY PRIOR ART

A. U.S. Patent No. 6,748,471 ("Keeney") (Ex. 1004.)

83. Like the '233 Patent, Keeney described "requesting and receiving print jobs over a communications network." (Keeney, 1:9-10). Figure 1 of Keeney

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

illustrated an embodiment of the system, annotated below:



Keeney, Figure 1 (Annotated)

84. In Keeney's system, a spooling server (red in annotated Figure 1 above) provided "compatibility with virtually all printers that may be connected to [a global communication network]…so that a document can be printed anywhere regardless of a specific printer's requirements." (*Id.,* 6:29; 6:34-37). The spooling server would receive print jobs from a print job source (blue), also referred to in Figure 1 as a client device. (*Id.,* 6:43-44). A printer polling device (green) would "poll[] the spooling server 50 via the network 110 to identify a print job associated with the

printer polling device." (*Id.,* 6:45-47). If the spooling server identified a print job for the printer polling device, "[t]he identified print job [would] be transmitted from the spooling server 50, to the printer polling device 100 and printed at a printer 120 coupled to the printer polling device 100." (*Id.,* 6:66-7:2).

85. The printer polling device would request print jobs from the spooling server by transmitting a "polling query" to the spooling server. (*Id.*, 12:1-10). Each polling query included an identification of a job to be printed at the printer associated with the polling device (*id.,* 12:8-10), as well as an identification of the associated printer (*id.,* 13:65-14:3).

86. The spooling server maintained a queue of print jobs, referenced by a user's personal identification number (PIN) or by a unique job number. (*Id.*, 9:20-22; 9:42-45). When a polling query identifying the PIN or the unique job number was received, the spooling server converted the print job identified by the PIN or the job number into "a form that [could] be understood by the printer." (*Id.*, 13:62-64). Once formatted according to printer-specific information, the spooling server "place[d] the print job 11 into a printer specific data file 805 for delivery to the printer polling device 100 to be printed at the target printer 120." (*Id.*, 14:19-22).

B. U.S. Patent No. 6,091,507 ("Vatland") (Ex. 1005.)

87. Vatland was directed to "a method and apparatus for printing a document over a network." (Vatland, 1:5-6). Vatland explained that network

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

printing suffered from a slow data transfer rate over the network, which resulted in "raster data [not being] transferred to [a] printer fast enough to feed a moderately fast print engine." (*Id.,* 1:62-64). To solve this problem, Vatland provided several techniques for increasing the speed of raster data transfers over a network (*id.,* 11:8-30), including that printer data "may be compressed and decompressed to obtain additional network bandwidth." (*Id.,* 11:25-26).

VII. GROUNDS FOR CHALLENGE

88. As discussed in detail below, the alleged invention of the '233 patent was disclosed in the prior art long before the '233 patent's 2000 priority date.

A. Ground 1: Claims 1, 2, 4, 5, 7, 9, 11, 12, 13, 17-21, 27, 29, 30, 35, 36, 47, 48, 50, 52, 57, and 61 are obvious over Keeney.

89. In the Complaint filed in the Eastern District of Kentucky litigation, Patent Owners allege that Petitioner infringes claims 47, 48, and 61. Therefore, these claims are discussed first.

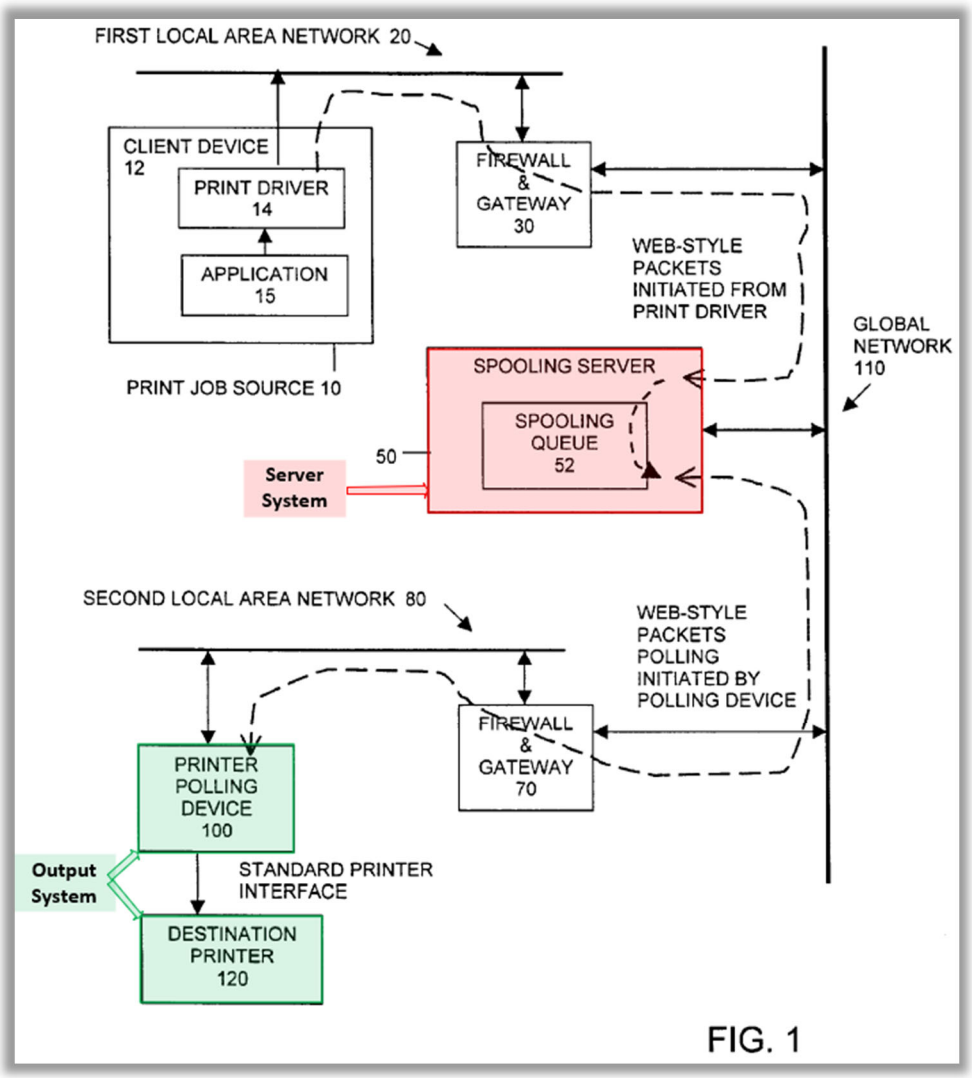
Lexmark, Exhibit 1002
Lexmark v. Flexiworld

1. Claim 47

a. [47.PRE]: In a computer memory or storage device at a server system, containing one or more server applications executable by one or more processors at one or more servers in the server system for transmitting device dependent output data acceptable for rendering at an output system, the output system including one or more devices, and the server system including one or more servers that are accessible by the one or more devices over a network that includes the Internet, the server system being distinct and separate from the one or more devices of the output system, the one or more server applications in the computer memory or storage device, when executed by the one or more processors at the one or more servers, configures the server system to:

90. To the extent the preamble is limiting, Keeney taught it. Keeney disclosed a server system in the form of a "spooling server" and an output system in the form of a "printer polling device" and "destination printer." See annotated Figure 1 of Keeney below.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



Keeney, Figure 1 (Annotated)

91. ***Computer memory or storage device at a server system***: The spooling server according to Keeney's disclosure included a "print job storage 52," also referred to as a "spooling queue 52" or "memory 52." (Keeney, Fig. 1; 8:9; 14:54). Additionally, spooling server 50 is described as storing received print jobs "in memory 52, which may comprise random access memory (RAM), magnetic or optical storage media, or any other read/write memory device." (Keeney, Fig. 9;

14:53-57).

92. Additionally, Keeney's print job source 10 is described as "a computer, a telephone, a personal digital assistant, an Internet appliance or the like" (Keeney, 3: 26-28), each of which a POSITA would understand to include a memory or storage devices.

93. ***Server applications executable by a processor in the server system***: Keeney disclosed that the spooling server included a processor that "enable[d] the spooling server to search for, retrieve, and/or reformat the print job for delivery to the printer polling device." (Keeney, 14:64-67). The spooling server was also configured to encrypt the reformatted print job before transmitting it to the printer polling device. (*Id.*, 8:54-58). The '233 patent broadly defines server applications as "application software to implement the functionalities of the application server and to provide services to its clients." ('233 patent, 16:62-66; 17:1-18:5). Based at least on this description and a POSITA's general knowledge, a POSITA would have found it obvious for the spooling server disclosed by Keeney to include one or more server applications executable by a processor in the spooling server, since it was well known for computing devices to use application software to perform specific

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

functions[6] and such application software would have enabled the spooling server to perform the functions Keeney described it as performing.

94. ***Transmitting device-dependent output data acceptable for rendering at an output system***: When a print job was available for printing, Keeney's spooling server "reformatted [the print job] in a form suitable for the identified printer." (*Id*., 9:57-59).

95. ***Output system including one or more devices***: Keeney's disclosure of the printer polling device and the destination printer – which were both devices – together form an "output system."

96. ***Server system including a server accessible by the device(s) over a network that includes the Internet***: As discussed above, the spooling server taught by Keeney is the claimed server system. The printer polling device disclosed by Keeney (a device that was part of the output system) was "capable of polling the spooling server 50 via the network 110." (*Id.*, 6:45-46). Keeney's network 110 was described as the Internet. (*Id.*, 3:34-39, 7:11-12).

[6] "Application Software" is either "software designed to fulfill specific needs of a user," or "software that is specific to an application and is composed of programs, data, and documentation." (Ex. 1008 - IEEE 100: The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition (2000)).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

97. ***Sever system distinct and separate from the output system device(s)***: As depicted in Keeney's Figure 1 above, the spooling server (the server system) was distinct and separate from the printer polling device and destination printer (the output system devices) because the output devices are remote from and receive content over a network from the connected spooling server.

b. [47.A]: receive, by the server system from the output system, one or more content objects, the one or more content objects including one or more pointers or references to digital content corresponding to one or more selection by a user at the output system, the one or more pointers or references to digital content are related to available digital content at the server system, and the available digital content are provided, at least in part, to the output system by the server system:

98. In Keeney's system, the spooling server maintained a queue of print jobs, referenced by a user's personal identification number (PIN) or by a unique job number. (Keeney, 9:20-22 ("each print job may be stored on the spooling server 50 according to a personal identification number (PIN)"), 9:38-40 ("A plurality of print jobs may be stored on the spooling server 50 according to the PIN 400 (e.g., in spooling queue 52)"), 9:42-45 ("Each print job may additionally be assigned a unique job number…Thereafter, the job number may be used to access the data files associated with that job unambiguously")).

99. To request any documents available for printing at an associated printer, the printer polling device would send a "polling query" to the spooling server.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

(Keeney, 12:1-10). The polling query included either a PIN or a job number to identify the print job to be printed. (*Id.,* 12:8-10). The object in the polling query identifying the PIN or job number was therefore a *content object including a pointer or reference to digital content* available at the spooling server (server system) based on the spooling server's maintenance of a queue of print jobs referenced by PIN and/or job number.

100. A user of Keeney's system could request a print job to be printed by interacting with a user interface at the printer polling device. (*Id*., 9:28-33 ("a small terminal may be provided at each printer polling device 100 to allow a user to interact with the spooling server 50 and request that the user's print jobs be sent to the printer 120 at the user's location")). The user requests print job by inputting the PIN used by the spooling server to index print jobs. (*Id.,* 9:34-38). Accordingly, Keeney disclosed that the PIN or job number in the polling query (a content object containing a pointer or reference to digital content) corresponded to a selection by the user at the printer polling device (the output system).

101. Keeney also described another embodiment of requests to the spooling server that rendered obvious the claimed "content objects." In this alternate embodiment, "an agent program 200 [on a client device] provide[d] a directory of documents 204 to the spooling server 50." (*Id.,* 12:18-20). The agent program made "the contents of some or all of the directories 204 on the client device 12 available

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

to the spooling server 50." (*Id.,* 12:34-36). The directory was communicated to and displayed by the printer polling device, such that "[s]election of a print job 11 from the directory [could] be made (e.g., via a user interface)." (*Id.,* 12:45-48). When the user selected a document in the directory for printing, the spooling server sent the agent program document information, "such as the name of the document that is requested." (*Id.,* 13:6-8). When a user used the printer polling device to select a print job from the directory (a selection by the user at the output system), a POSITA would have understood that the printer polling device would have sent the spooling server (server system) a content object including pointer or reference to the selected print job in order to enable the spooling server to identify the document information that was sent to the agent program.

102. Regardless of how the printer polling device requested a print job, the spooling server described by Keeney (server system) would send the requested print job (digital content) to the printer polling device for printing by the associated printer (output system). (*Id.,* 4:42-45 ("The print job may be printed at the desired print location when the printer polling device at the desired print location polls the spooling server and identifies the print job"); *see also* 9:56-60, 13:14-15).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

c. [47.B]: receive, by the server system from the output system, one or more objects, the one or more objects having information related to the output system that includes the one or more devices:

103. The ’233 patent explains that the term “object” can include “[a]ny entity containing information, descriptions, attributes, data, instructions etc. in any computer-readable form or medium such as hardware, software, files based on or including voice, text, graphics, image, or video information.” (’233 patent, 5:21-25). In Keeney’s disclosure, Keeney described that the polling query generated by the printer polling device could further include an “object having information related to the output system,” consistent with the ’233 patent’s explanation, in the form of a data object that includes an identification of the printer at which a document is to be printed. (Keeney, 13:65-14:3 (“Target printer identification…can be provided by the printer polling device 100 as part of the polling query”)).

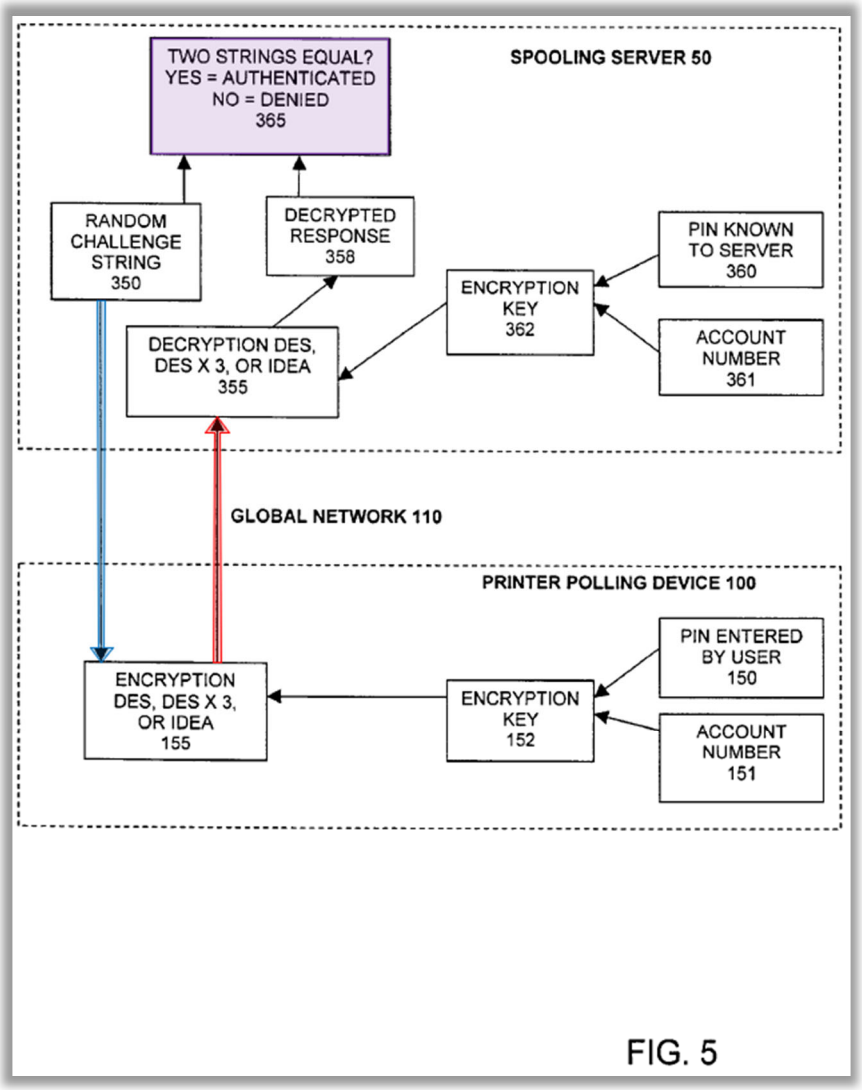
d. [47.C]: receive, by the server system and from the output system, security or authentication information associated with a user operating the output system for authentication of the user:

104. Keeney explained that when a user desires to print documents over a non-secure channel, it was important for the spooling server to “verify that it is indeed talking to the actual user.” (Keeney, 10:9-11). To this end, Keeney disclosed an authentication process in which the spooling server would send a random string of bits to the printer polling device (see step indicated by blue in annotated Figure 5

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

below). (*Id.*, 10:11-16). The printer polling device would in turn encrypt the string using an encryption key generated based on a user's personal identification number (PIN) and return the encrypted string to the spooling server (see step annotated in red). (*Id.*, 10:11-16). The spooling server would decrypt the string using an encryption key generated from a previously received user PIN and "accept[] the user as authenticated" if the decrypted string matched the original string (see step annotated in purple). (*Id.*, 10:16-21).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



Keeney, Figure 5 (Annotated)

105. Thus, Keeney disclosed "security or authentication information associated with a user" in the form of the string of bits encrypted based on the user PIN, which enabled the spooling server to authenticate the user.

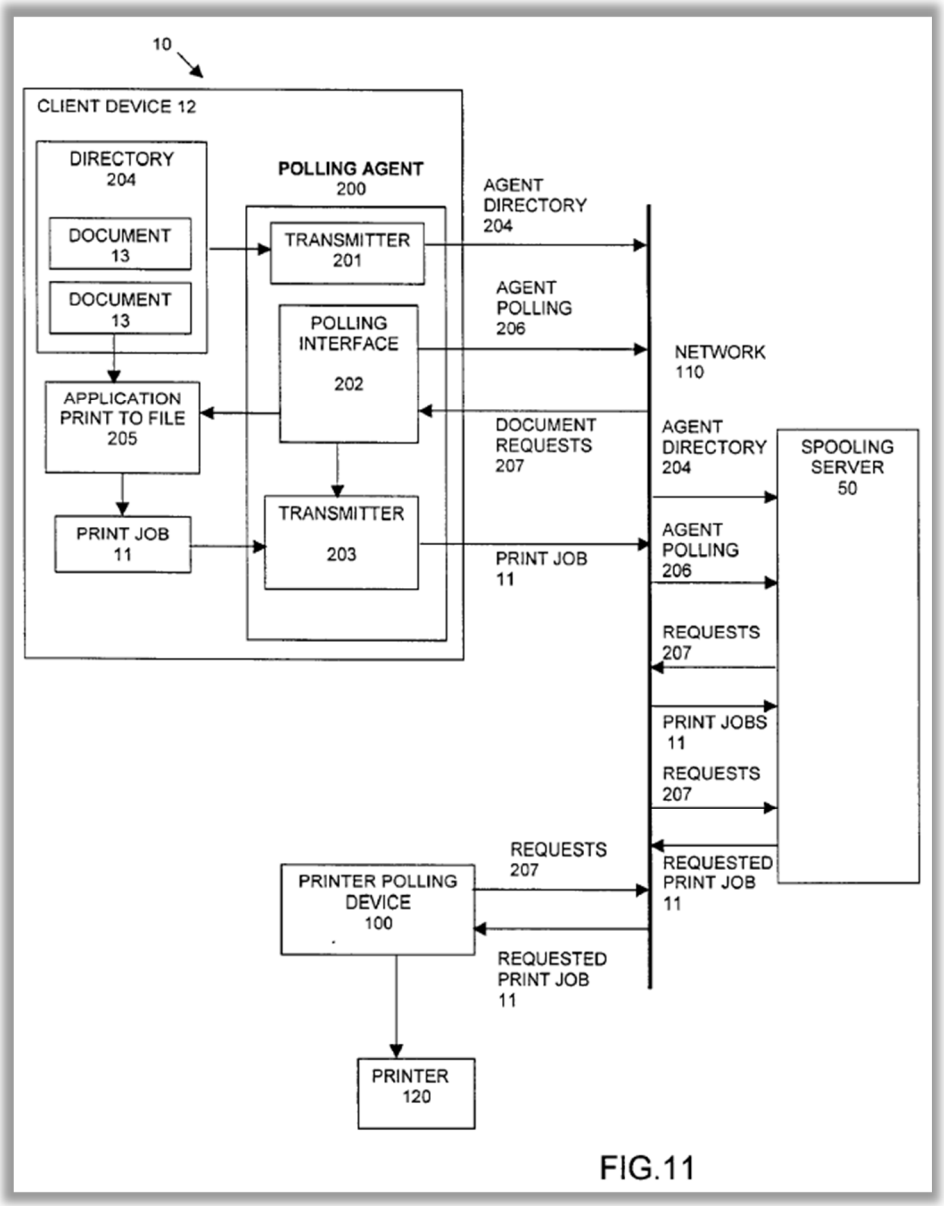
e. [47.D]: retrieve, by the server system, at least part of the digital content from a content server employing, at least in part, the one or more pointers or references to the digital content included in the one or more content objects received from the output system:

106. As described with respect to [47.A], Keeney disclosed a content object including pointers or references to digital content in the form of either a PIN or job number identifying a print job stored in a queue at the spooling server. (Keeney, 8:8-9). Keeney also disclosed using these pointers or references to retrieve digital content from a content server such as a file system on the spooling server[7]. For example, the spooling server would use the PIN or job number to access a requested print job. (*Id.*, 9:20-22 ("each print job may be stored on the spooling server 50 according to…PIN"), 9:44-45 ("the job number may be used to access the data files associated with that job unambiguously")). Once the spooling server identified the print job to be printed, "the data for that print job is retrieved from the file system" (that is, retrieved from a content server). (*Id.,* 9:55-57).

[7] The '233 patent explains that an application server and a content server "may be implemented with different applications executed in a single network node or server." ('233 patent, 18:27-31). Keeney's disclosure of a spooling server and spooling server file system, implemented on the same network node or server, is consistent with this explanation in the '233 patent.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

107. Keeney also disclosed that print jobs could be stored in a directory on a client device (also referred to as a "print job source 10"), where the client device is remote from the spooling server and printer polling device and where the directory functions as a content server. (*Id.,* 12:32-38; Figure 11).



Keeney, Figure 11

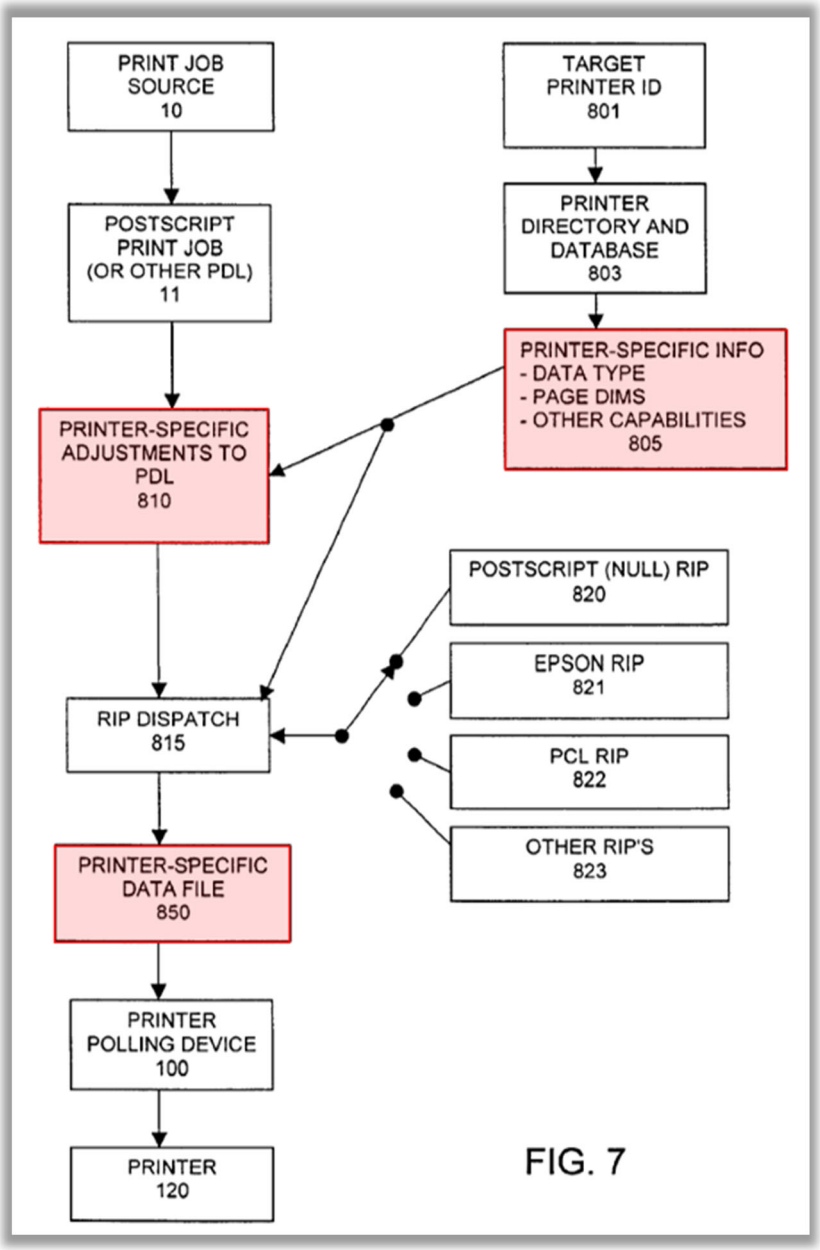
108. In this embodiment, the spooling server would retrieve a print job from the client device by sending "document information, such as the name of the document that is requested," to the agent program on the client device. (*Id.,* 13:6-8). The agent program would "open[] the document 13 using the application it was created in and print[] it to a temporary file 205 to create a print job," which is returned to the spooling server for transmission to the destination printer. (*Id.,* 13:9-15).

f. [47.E]: generate, by the server system, output data associated with at least part of the digital content, the output data having a relationship, at least in part, to the information included in the one or more objects received by the server system:

109. The spooling server in Keeney's disclosure generated output data for a selected print job by converting the print job into "a form that [could] be understood by the printer." (Keeney, 13:62-64). As described with respect to 47[B], the spooling server received an object that included an identification of a target printer for printing a print job (e.g., the polling query object identifying the destination printer). (*Id.*, 14:1-3). The spooling server used the target printer identification to "retrieve[] printer specific information 805 for the target printer 120" (*id.*, 14:4-5) and, "[b]ased on the printer specific information 805 (such as data type, page dimensions, color capabilities, margins, and other print characteristics) the spooling server 50 [would] make printer specific adjustments 810 to the print job 11 at the page description

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

language level prior to rasterization." (*Id.*, 14:6-10). Once formatted according to the printer-specific adjustments, the spooling server "place[d] the print job 11 into a printer specific data file 805 for delivery to the printer polling device 100 to be printed at the target printer 120." (*Id.*, 14:19-22; *see also* annotated Figure 7 below).



Keeney, Figure 7 (Annotated)

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

110. The "printer specific data file" constituted the claimed "output data associated with at least part of the digital content," and it would have been obvious that this data file had a relationship to the information included in the object received by the server system because it was generated according to the specific characteristics of the target printer.

g. [47.F]: apply, by the server system, an encoding operation on at least part of the digital content for transmitting the output data that is associated with at least part of the digital content:

111. It is my opinion that Keeney in combination with the general knowledge of a POSITA renders limitation 47.F obvious in two ways. First, when "encoding . . .the digital content" is viewed as a traditional print flow process where an encoding operation is performed on the digital document itself, Keeney teaches this.

112. To begin with, the '233 patent does not define "encode," "encoding," or "encoding operation." Instead, it generally describes that the digital content may comprise "any format, language, encoding data or combination." ('233 patent, 8:37-41). As I explained above, as POSITA would understand that "encoding" is any function or operation that changes the representation of the data. (*See* ¶¶28-33).

113. In Keeney's disclosure, the spooling server (e.g., server system) transmitted a print job (digital content) to the printer polling device for output to a

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

destination printer. (Keeney, 8:53-57). The printer polling device received the print job from the spooling server via the "network 110" (i*d*., 14:40-42), which was described as including the Internet. (*Id*., 7:11-12).

114. By the time of the '233 patent's filing date, a POSITA would have known that encoding operations such as compression were used to facilitate transmitting print data over a network. (*See* ¶¶26-49). A POSITA would further understand that digital content in the form of a print job could be compressed using various image compression protocols such as JPEG because it was ubiquitously used to preserve system resources by reducing the amount to data to be stored or transmitted over a network. (Appx. F, ¶¶ 40-49).

115. Aside from being ubiquitous in print flow operations, there are multiple additional reasons that the compression (encoding) operation would have been obvious. By the time of filing date of the '233 patent, it was well-recognized that compressing the data before encrypting and transmitting it over a network was beneficial. For example, IP Payload Compression Protocol (IPComp) was a protocol to reduce the size of IP datagrams to increase the overall communication performance between a pair of nodes by compressing the datagram. (Appx. N, RFC 2393, December 1998). It taught that for IP payload compression to be useful it must be applied before encryption and applied to the IP packet payload. (*Id.* at 1). This order of operations (compression before encryption) was preferred for two reasons:

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

(a) the encryption operation can be performed more efficiently on the compressed data because fewer bits need to be encrypted, and (b) encrypting the content first (followed by compression) may not have reduced the size of data because the encryption operation usually randomizes the content, which makes its compression less efficient. (*See, e.g.,* Appx. O ("Compression and Encryption in IPSec," June 24, 1998), recognizing the benefits of compression before encryption for transporting data over a network via IPSec, and Appx. P ("J. P. McGregor and R. B. Lee, "Performance impact of data compression on virtual private network transactions," *Proceedings 25th Annual IEEE Conference on Local Computer Networks. LCN 2000*, Tampa, FL, USA, 8-10 Nov. 2000, pp. 500-510), describing experiments where compression-before-encryption produced improved transmission efficiency). As shown below, McGregor expressly stated that packet payload compression must be completed before any security processing is performed.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

5.1. Protocol Description

IPComp reduces the size of IP packets in IPv4 and in IPv6 using data compression algorithms [21]. The protocol specification includes several restrictions and guidelines concerning these compression algorithms. To preserve the consistency of the packet payload, the compression algorithm must be lossless. Furthermore, the compression of a packet payload must be completed before any IP security processing is performed: a cryptographically secure encryption algorithm outputs ciphertext that cannot be compressed. It follows that the decompression and reassembly of IP packets must also occur after decryption. In addition, each IP packet must

Appx. P at Section 5.1.

116. Thus, in Keeney's disclosure of the spooling server transmitting a print job over the Internet, I find it obvious for the spooling server to apply an encoding operation on the print job to compress the print job for transmission over the Internet, especially because the print data were to be sent in encrypted format and applying the compression before encryption would have resulted in both reducing the size of the content and thus reduced transmission time, as well as saving computational cycles as I explained earlier. (¶¶34-50.)

117. In an alternate obviousness scenario, a POSITA would have found this encoding operation obvious as part of forming data packets that could be sent to another network node through Keeney's VPN. (¶28, 32.) Specifically, Keeney described an option for using a VPN, and explained: "A VPN is made up of two

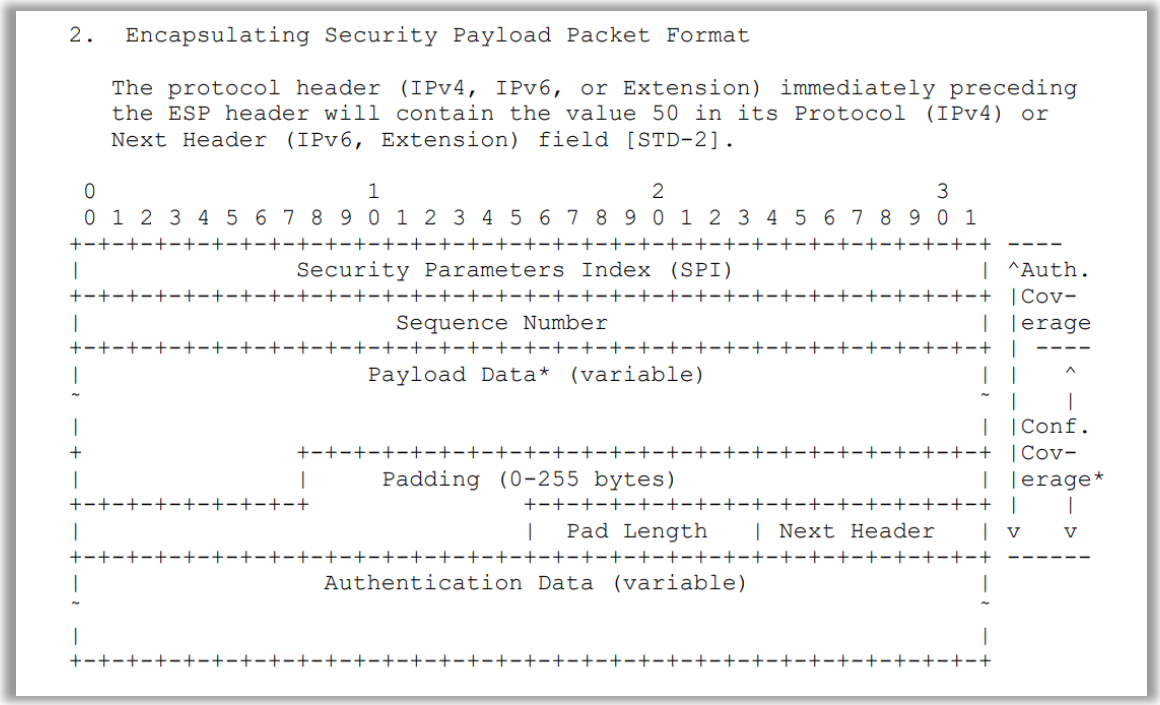
Lexmark, Exhibit 1002
Lexmark v. Flexiworld

components: L2TP (Layer 2 tunneling protocol) and IPSec (Internet Protocol Security).” (Keeney, 7:29-46).

118. The IPSec, a well-known and standardized protocol well before the filing date of the ’233 Patent, was a secure network protocol suite that authenticated and encrypted packeted data to provide secure communication between two computers over an IP network. The Request for Comments (RFC) 2406 (Ex. 1009), from 1998, was a document published by the Network Working Group, and provided explanations about the IP Encapsulating Security Payload (ESP) features of IPSec. (Ex. 1009, 1, 4, 7, 9, 13 (explaining the association with IPSec)).

119. As shown, for example, in the figure on page 3 of this document (below), the payload data was encoded into packets that included various headers and fields to ensure confidentiality, authentication and other features. (Ex. 1009, pp. 2-3, ¶¶ 111-116, *supra*).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



RFC 2406, Figure on p. 3

120. In my opinion, a POSITA, based on Keeney's explanation about deploying a VPN that used IPSec, would have understood and would have found it obvious that content data would be encoded to form the aforementioned packets for transport via IPSec. In sum, Keeney in combination with the general knowledge of a POSITA renders this limitation obvious in multiple ways.

h. [47.G]: apply, by the server system, an encryption scheme on at least part of the digital content for ensuring security of at least part of the digital content for rendering at the output system:

121. Again, Keeney discloses limitation 47.G in two ways. First, as with limitation 47.F, *supra*, when applying an encryption scheme to the digital content is

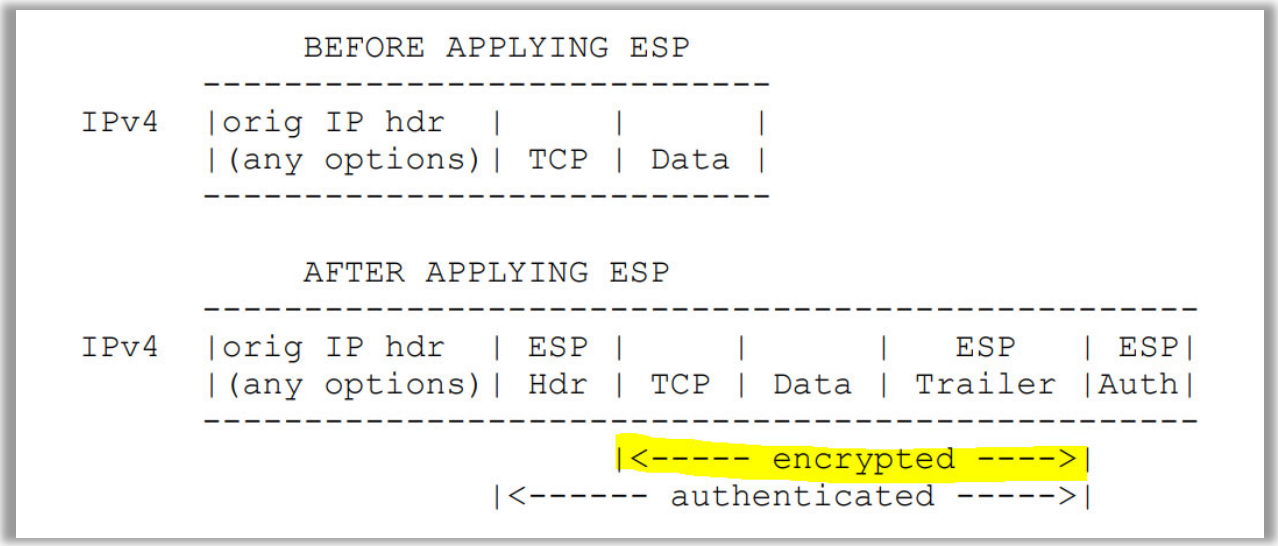
viewed as part of the traditional print process flow, Keeney clearly disclosed encrypting the digital content print job. Keeney described that a print job (digital content) would be encrypted by the print job source, decrypted by the spooling server to facilitate reformatting, and re-encrypted by the spooling server (an encryption scheme applied by the server system) before the encrypted print job was sent to the printer polling device. (Keeney, FIG. 2; 8:31-34, 8:51-55; 8:62-63). Encryption algorithms such as the International Data Encryption Algorithm (IDEA), the Data Encryption Standard (DES), or triple-DES could be employed in the encryption process. (*Id.*, 8:34-38). Keeney explained that this encryption scheme would help ensure security of the print job by providing limits on the person who could request the print job to be printed (i.e., rendered at the output system). (*Id.,* 8:22-24 (encryption of a print job provides that the print job "can only be printed by a person with the correct decryption key")).

122. In the alternate obviousness analysis, a POSITA would have known or found it obvious that by producing packets that conformed to the IPSec protocol, Keeney's system that used a VPN also performed an encryption operation. (See ¶¶ 111-120, *supra.*) This is shown in numerous sections of the RFC 2406 document, including the explanation on page 5, Section 2.3, where using an encryption algorithm for encrypting the payload data is explained, in Section 2.4, padding (for Encryption) is described, and on page 10, Section 3.2.1, which describes

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

"Encryption Algorithms." (Ex. 1009)

123. Additionally, the various figures in the same document show the encrypted payload, including the below figure that is reproduced from page 8 of Ex. 1009.



124. In sum, it is my opinion that limitation 47.G is rendered obvious under both these scenarios.

i. [47.H.i]: transmit, by the server system to the output system over the network that includes the Internet, at least part of the output data that is associated with at least part of the digital content, and the at least part of the output data includes the encryption scheme applied to at least part of the digital content:

125. Keeney described how the spooling server (e.g., server system) transmitted a print job formatted for a destination printer (output data associated with digital content) to the destination printer (output system). (Keeney, 8:53-57 (the

spooling server "reformatt[ed] [a print job] for the destination printer," re-encrypted the reformatted print job, and "transmitted [the encrypted, reformatted print job] from the spooling server 50 to the printer polling device 100")). The printer polling device received the print job from the spooling server via the "network 110" (*id.,* 14:40-42), which was described as including the Internet (*id.,* 7:11-12).

j. [47.H.ii]: wherein the transmitting of at least part of the output data is after the server system receives, from the output system, at least (i) the security or authentication information, (ii) the one or more content objects, and (iii) the one or more objects from the output system:

126. Keeney disclosed that the spooling server (server system) would transmit the encrypted, reformatted print job (output data) to the printer polling device after receiving the string of bits encrypted based on the user PIN ((i) security or authentication data; see [47.C]). (Keeney, 10:8-11 ("When a user desires to access their account or documents over a non-secure channel, ***first***, the spooling server needs to verify that it is indeed talking to the actual user;" emphasis added). Likewise, the output data was transmitted to the printer polling device after the spooling server received the polling query from the printer polling device containing a PIN, job number, or document information ((ii) content object; see [47.A]), as well as the identification of the destination printer ((iii) object; see [47.B]). (*Id.,* 9:45-48 ("***Once a print job has been selected*** to be printed to a destination printer 120, the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

job number for the print job may be entered into a queue associated with the destination printer;" emphasis added), 13:66-67 ("a target printer 120 ***must be identified before*** the spooling server 50 can forward the print job 11 to be printed;" emphasis added)).

k. [47.I]: wherein the one or more devices of the output system include at least an information apparatus, and at least part of the transmitted output data being device dependent for rendering at least part of the digital content at the output system that includes the one or more devices:

127. With respect to an "information apparatus," the '233 Patent states that this term refers generally to "computing devices, which include both stationary computers and mobile computing devices." ('233 Patent, 1:21-23). An output device, such as a printer, can then be "connected to [the] information apparatus via a wired connection such as a cable line." (*Id.,* 2:19-21). Similarly, Keeney explained that the printer polling device could be, *inter alia*, "a stand-alone device connected to one or more printers via a standard printer port." (Keeney, 3:59-61). The output system taught by Keeney – including the printer polling device and a printer, as described with respect to [47.PRE] – therefore included an information apparatus in the form of the printer polling device.

128. As described with respect to [47.E], the spooling server (server system) generated output data that was device dependent for rendering at the output system

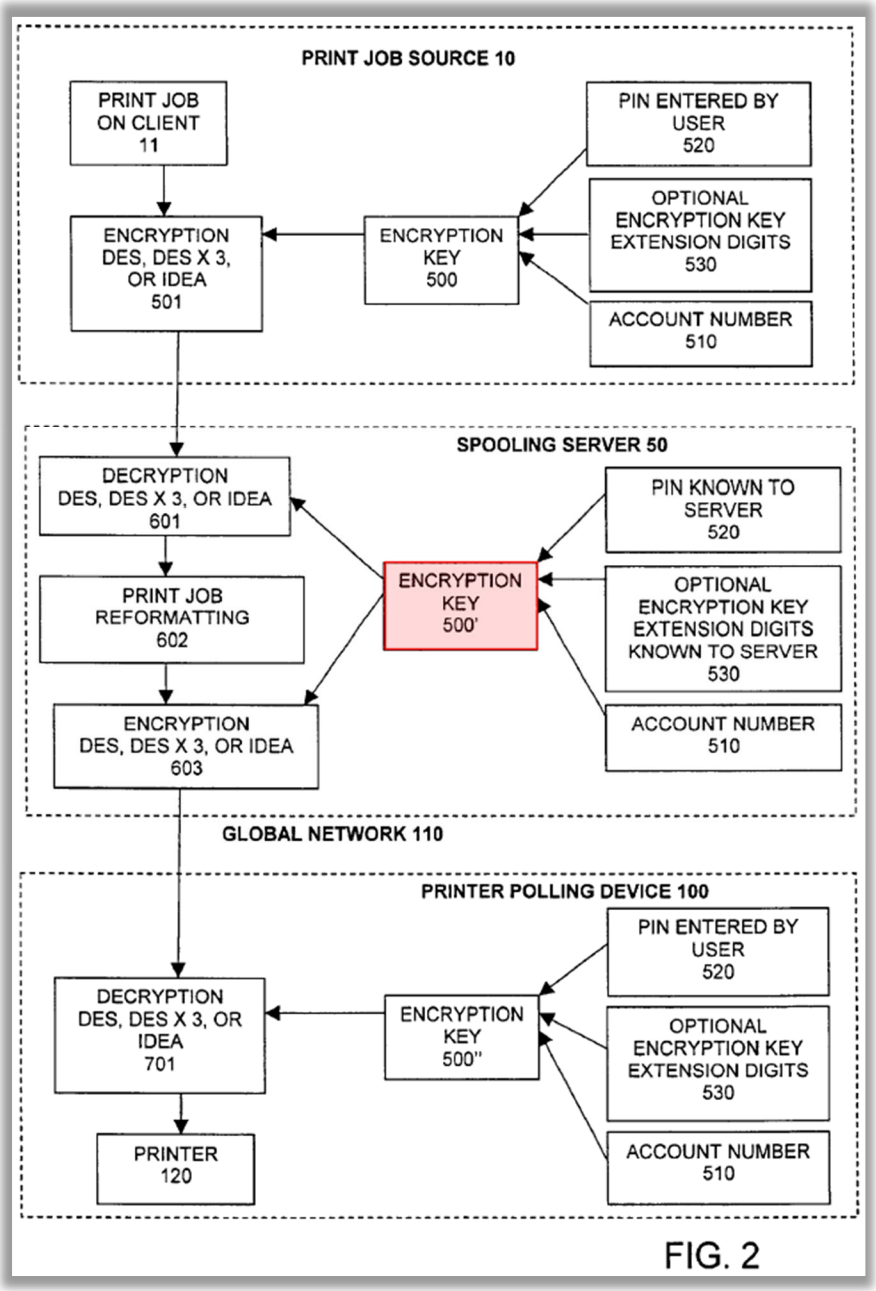
Lexmark, Exhibit 1002
Lexmark v. Flexiworld

(the printer) because the output data was generated according to the specific characteristics of the target printer. (*See e.g.,* Keeney, 13:62-64 (the print job would be converted "a form that [could] be understood by the printer")). In sum, it is my opinion that independent claim 47 is rendered obvious by Keeney in view of the general knowledge of a POSITA.

2. Dependent claim 48: …in which the encryption scheme includes security keys for ensuring security of the digital content.

129. As described with respect to [47.G], Keeney disclosed an encryption scheme. This scheme included a security key for ensuring security of the digital content. (Keeney, 8:41-44 ("The key 500 used for encryption 501 may be derived from an account number 510, a user's secret PIN (personal identification number) 520, and/or optionally some additional encryption key digits 530 supplied by the user")). Keeney illustrated this security key in Figure 2, annotated below.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



Keeney, Figure 2 (Annotated)

It is my opinion that claim 48 is rendered obvious by Keeney in view of the general knowledge of a POSITA.

3. Dependent Claim 61: …in which the one or more objects include device identification information, the device identification information including one or more of a device brand, a model, an identification number, a name, an address, a device capability, a path, a registration, or an IP address, individually or in any combination, and the device identification information is related to the output system or to the one or more devices of the output system.

130. As described with respect to 47[B], Keeney disclosed an object (e.g., a printer identification in the polling query) that included device identification information (e.g., "target printer identification 801"). (Keeney, 14:1-3). The spooling server taught by Keeney used the printer identification to "retrieve[] printer specific information 805 for the target printer 120 from a printer database 803…such as data type, page dimensions, color capabilities, margins, and other print characteristics." (*Id.,* 14:4-8). A POSITA would have understood that the "target printer identification" would have included a device brand, a model, an identification number, a name, an address, a device capability, a path, a registration, or an IP address, since such information would have enabled the spooling server to use the target printer identification to retrieve the printer information as disclosed.

131. Finally, since the printer was part of the output system as described with respect to [47.PRE], the identification of the printer related to the output system or to the one or more devices of the output system. It is my opinion that claim 61 is rendered obvious by Keeney in view of the general knowledge of a POSITA.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

1. Independent Claim 1:

a. [1.PRE]: A method of operating a digital content service for providing digital content over a network that includes the Internet to an information apparatus or a device for rendering, the digital content service including:

132. As described with respect to [47.PRE], Keeney disclosed a method of operating a digital content service (e.g., a service provided by the spooling server) for providing digital content over a network that includes the Internet. As described with respect to [47.I], the service disclosed by Keeney included providing digital content to an information apparatus or device (e.g., printer polling device) for rendering (e.g., printing).

b. [1.A]: one or more servers for providing the digital content to the information apparatus or to the device, the one or more servers being distinct and separate from the information apparatus or the device:

133. See [47.PRE].

c. [1.B]: one or more server applications executable at the one or more servers for managing at least part of the digital content service, the digital content service method comprising:

134. See [47.PRE].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

d. [1.C]: receiving, from the information apparatus or the device and by the one or more servers, security or authentication information associated with a user operating the information apparatus or the device for authentication of the user, and in dependence of successful authentication of the user, enabling the user access to the digital content service:

135. As described with respect to [47.C], Keeney disclosed security or authentication information associated with a user of the printer polling device (information apparatus or device) for authentication of the user. Keeney also disclosed that successful authorization enabled the user to access the digital content service provided by the spooling server by its disclosure that the spooling server used this security or authentication information to "verify that it is indeed talking to the actual user" before the user could access accounts or document for printing via the service provided by the spooling server. (Keeney, 10:9-12).

e. [1.D]: receiving, from the information apparatus or the device and by the one or more servers, payment or subscription information associated with the user for accessing one or more digital content that are available at the one or more servers operated, at least in part, by the digital content service:

136. Keeney suggested receiving payment information by its disclosure of a "fee…charged to access the spooling server." (Keeney, 7:60-61). Keeney explained that the fee was received from the information apparatus or device (printer polling device) by the one or more servers operated by the digital content service (spooling server). (*Id.,* 8:2-6 ("The fee can be paid via…the printer polling device 100, or any

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

other suitable device capable of communicating with the spooling server")). The fee enabled the user to access digital content. (*Id.,* 7:67-8:2 ("The fee may be charged for…retrieving a print job from the spooling server")). A POSITA would have found it obvious for payment information to be transmitted to the spooling server to enable this fee to be paid. Finally, Keeney also disclosed that the payment could itself be associated with subscription information of the user, such as a "lifetime membership or monthly membership." (*Id.,* 7:64).

f. [1.E]: receiving, at the one or more servers, a device object from the information apparatus or the device over the network that includes the Internet, the device object including device information related to one or more devices, and the device information corresponding to at least one of device identification information, device address information, device capability information, model information, brand information, or registration information, individually or in any combination, and the one or more devices include the information apparatus or the device:

137. See [47.B] for receiving a device object, [47.PRE] for communications between the printer polling device (information apparatus or device) and spooling server (one or more servers) over a network that included the Internet, and [61] for the device object including the enumerated types of device information.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

g. [1.F]: providing, to the information apparatus or the device and by the one or more servers, one or more pointers or references to at least part of the one or more digital content available at the one or more servers for selection by the user at the information apparatus or the device:

138. As explained with respect to [47.A], Keeney described the spooling server (the one or more servers) communicating a directory of documents to the printer polling device (information apparatus or device). (*See* Keeney, 12:18-20; 12:45). Communicating the directory caused the printer polling device to display a list of documents – that is, pointers or references to digital content – such that "[s]election of a print job 11 from the directory [could] be made (e.g., via a user interface)." (*Id.,* 12:45-55).

h. [1.G]: receiving, at the one or more servers and from the information apparatus or the device, at least a pointer or reference to selected digital content from among the one or more pointers or references provided, by the one or more servers, to the information apparatus or the device:

139. See [47.A].

i. [1.H]: retrieving, by the one or more server applications, at least part of the selected digital content from the one or more servers employing, at least in part, the pointer or reference to the selected digital content received from the information apparatus or the device:

140. See [47.D].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

j. [1.I]: generating output data, at the one or more servers using the one or more server applications, the output data being related to at least part of the selected digital content, and the generating of the output data including:

141. See [47.E].

k. [1.J]: applying, by the one or more server applications, an encoding operation on at least part of the selected digital content:

142. Keeney disclosed an encoding operation by its disclosure of generating a "printer specific data file" for a print job. (Keeney, 14:21). To generate this file, the spooling server (the one or more servers) would make "printer specific adjustments 810 to the print job 11 at the page description language level prior to rasterization." (*Id.,* 14:8-10). For some types of printers, the spooling server would rasterize "page description language…into a bitmap form…[and] format[] [the bitmap data] into the form understood by the printer." (*Id.,* 14:13-16). By these disclosures, Keeney described a transformation of a print job into a different format that was suitable for printing by a target printer – and thus taught the spooling server applied an encoding operation on the selected digital content (print job). Additionally, since as described with respect to [47.PRE] it would have been obvious for the spooling server to execute a server application to perform its disclosed functionality, it likewise would have been obvious for this server application to perform the encoding operation taught by Keeney.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

l. [1.K]: applying, by the one or more server applications, an encryption scheme on at least part of the selected digital content for ensuring security of at least part of the selected digital content, and wherein, subsequent to applying the encoding operation and the encryption scheme on at least part of the selected digital content, the selected digital content is, at least in part, device dependent for rendering at the one or more devices:

143. See [47.G] regarding an encryption scheme. With respect to "device dependent" data, Keeney disclosed that the spooling server encrypted a print job that had been "reformatt[ed] for the destination printer." (Keeney, 8:54-55). As described with respect to [1.J], Keeney taught that print jobs were reformatted into a printer-specific data file for the destination printer by applying an encoding operation. Thus, the encrypted printer-specific data file was device dependent for rendering at the target printer.

m. [1.L]: transmitting, by the one or more servers, at least part of the generated output data to the information apparatus or the device for rendering at least part of the selected digital content at the one or more devices:

144. See [47.H.i] and [47.I].

n. [1.M.i]: wherein the generated output data is, at least in part, device dependent to the one or more devices, and the generated output data having a relationship, at least in part, to the device information included in the device object and received by the one or more servers from the information apparatus or the device over the network that includes the Internet:

145. See [1.K] for generated output data that is device dependent, [47.E] for

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

output data having a relationship to the device information included in the device object, and [47.PRE] for the communications between the printer polling device (information apparatus or device) and spooling server (one or more servers) over a network that included the Internet.

o. [1.M.ii]: wherein the transmitting of at least part of the generated output data by the one or more servers to the information apparatus or the device is subsequent to having received the security or authentication information associated with the user operating the information apparatus or the device, and the transmitting of at least part of the generated output data is based, at least in part, on having received, from the information apparatus or the device, the payment or subscription information:

146. See [47.H.ii] for transmitting output data subsequent to having received security or authentication information and [1.D] for payment or subscription information. As described with respect to [1.D], Keeney described that a fee may be charged for "retrieving a print job from the spooling server." (Keeney, 7:67-8:2). Thus, it would have been obvious to a POSITA for the spooling server to transmit the output data on having received payment or subscription information in order to ensure that the fee for the print job was properly paid.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

4. Claim 2:

a. [2.A]: …providing, by a client application at the information apparatus or the device and over a graphical user interface at a display screen of the information apparatus or the device, one or more pointers or references to the one or more digital content for selection:

147. See [18.C] and [18.D] for an information apparatus that includes a graphical user interface and a display screen, [18.E] for a client application that displays information over a graphical user interface, and [1.F] for providing pointers or references for selection.

b. [2.B]: receiving, by the client application at the information apparatus or the device, the selected digital content over the graphical user interface from among the one or more pointers or references to the one or more digital content that are available at the one or more servers:

148. See [18.J].

c. [2.C]: obtaining, by the client application at the information apparatus or the device, a content object that includes a pointer or reference to the selected digital content from among the one or more pointers or references:

149. See [18.J].

d. Claim 4: …in which the encryption scheme includes use of security keys for generating the output data.

150. See claim 48.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

5. Claim 6:

a. [6.A]: …receiving, by the client application at the information apparatus or the device, at least part of the output data from the one or more servers:

151. See [18.E] for a client application and [1.L] for the information apparatus receiving output data from a server.

b. [6.B]: decoding, by the client application at the information apparatus or the device, at least part of the output data received from the one or more servers for rendering of at least part of the selected digital content:

152. As described with respect to [47.F], a POSITA would have known that encoding operations were commonly performed to transmit data over the Internet. Likewise, a POSITA would have known that a device receiving data transmitted over the Internet would have needed to decode the received data. For example, if a server encoded the data as part of an SSL protocol[8] used to ensure security of the data as it was transmitted from the server to a receiving device, a POSITA would have known that the receiving device would have performed a corresponding

[8] SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

decoding operation on the received data. Thus, in Keeney's disclosure of the spooling server transmitting a print job to the printer polling device over the Internet, a POSITA would have found it obvious for the printer polling device to use a client application to decode the received print job to prepare the print job for rendering (*i.e.*, printing).

6. Claim 7: …in which the information apparatus is a wireless information apparatus and includes at least one of an e-book, a smart phone, a screen phone, an Internet phone, an information pad, an Internet pad, an Internet appliance, and a digital camera, individually or in any combination.

153. According to Keeney's disclosure, the printer polling device (an information apparatus) "may be a portable device." (Keeney, 10:27). The printer polling device includes "a connection to the network 110," (*id.,* 9:8-9), such as a "polling transmitter 105…for transmitting polling requests to the spooling server 50 via the network 110." (*Id.,* 14:36-38). Thus, Keeney suggested that the printer polling device can be a wireless information apparatus including at least devices such as an information pad, an Internet pad, or an Internet appliance.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

7. Claim 9:

a. [9.A]: …in which the information apparatus or the device are part of an output system for receiving the output data for rendering at the one or more devices, the one or more devices include at least one of a television, a projector, a display screen, a printer, or an audio output device, individually or in any combination:

154. See [47.PRE].

b. [9.B]: wherein the one or more devices are either: (a) part of the output system for rendering or playing at least part of the selected digital content at the one or more devices; or (b) connectable, by one or more wire or wireless connections, to the information apparatus or the device for rendering or playing at least part of the selected digital content at the one or more devices:

155. See [47.PRE]; a printer in Keeney's disclosure is part of the output system for rendering (*i.e.*, printing) digital content. Additionally, a printer is connectable to the printer polling device for receiving print jobs from the printer polling device. (Keeney, 7:51-53 ("The printer polling device 100 may be: (i) a stand-alone device connected to the printer 120 via a standard printer port"), 14:44-45 ("A printer interface 112 enables the printer polling device 100 to forward the print job to a printer 120 for printing")).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

8. Claim 11:

a. [11.PRE]: …wherein the generated output data, corresponding to at least part of the selected digital content, is at least in part device dependent for rendering or output or playing at the one or more devices that include the information apparatus or the device, and at least part of the selected digital content includes an encryption scheme for ensuring security of the selected digital content:

156. See [1.K].

b. [11.A]: decoding, by the client application at the information apparatus or at the device, at least part of the selected digital content, using, at least in part, the output data received from the one or more servers:

157. See [6.B].

c. [11.B]: rendering, by the client application at the information apparatus or at the device, at least part of the selected digital content that has been decoded:

158. See [6.B].

9. Claim 12: …in which at least one device of the one or more devices is an output device, and the output device includes at least one of a display device, a television, a printer, an audio output device, a speaker, or a music output device, individually or in any combination, and wherein the output device is wirelessly connected or connectable to the information apparatus or the device.

159. See claim 9.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

10. Claim 13: …wherein the one or more devices include one or more output devices, and subsequent to having received the device object by the one or more servers from the information apparatus or the device, the method further comprises receiving, from the information apparatus and by the one or more servers, a selected output device from among the one or more output devices; and the generated output data is device dependent for rendering at the selected output device; and wherein the selected output device includes an audio output device, a display device, a television, or a printer.

160. See claim 9.

11. Claim 17: …in which generating at least part of the output data at the one or more servers includes one or more of invoking, launching, calling, utilizing, or involving one or more applications.

161. See [47.PRE].

12. Claim 18:

a. [18.PRE]: An information apparatus for obtaining digital content from a digital content service provided by one or more servers over a network, the digital content accessible, at least partly, by the information apparatus, the information apparatus including:

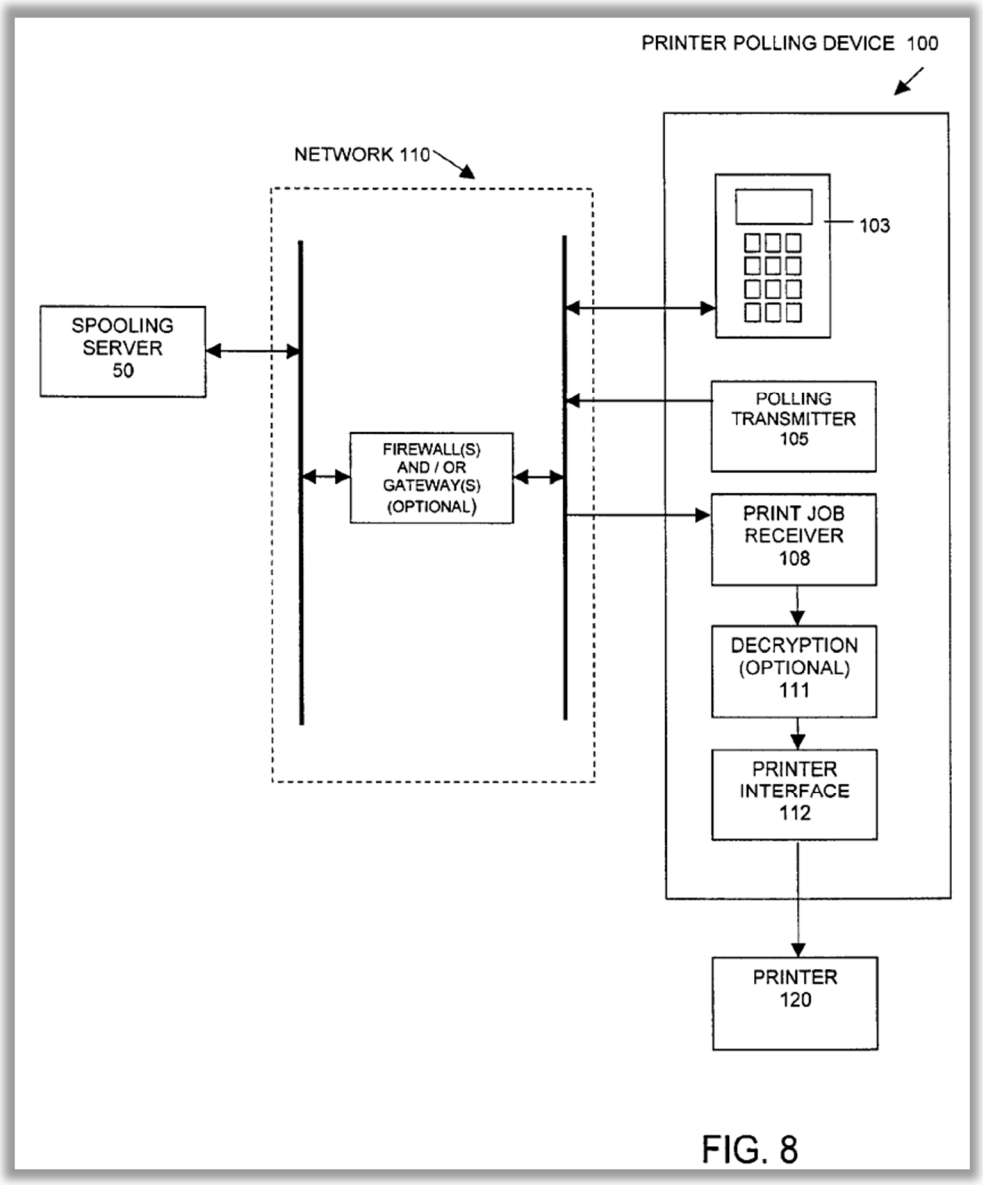
162. As described with respect to [47.I], Keeney disclosed an information apparatus in the form of a printer polling device. As described with respect to [47.PRE], the printer polling devices was used to obtain digital content from a digital content service provided by one or more servers (e.g., the spooling server) over a network.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

b. [18.A]: a processor:

163. It would have been obvious for the printer polling device to include a processor. For example, Keeney explained that the printer polling device had a connection to a network and a printer (Keeney, 9:8-10), and Figure 8 of Keeney depicted components of the printer polling device including a polling transmitter 105, a print job receiver 108, decryption 111, and a printer interface 112. (*Id.,* 14:31-45).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



Keeney, Figure 8

164. A POSITA would have found it obvious for these components to include or to be executed by a processor, since it was well known before the '233 Patent's priority date for computing devices to include a processor.

c. [18.B]: an operating system:

165. It was well known for computing devices to include operating systems,

and thus it would have been obvious for Keeney's printer polling device to have an operating system.

d. [18.C]: a touch sensitive screen:

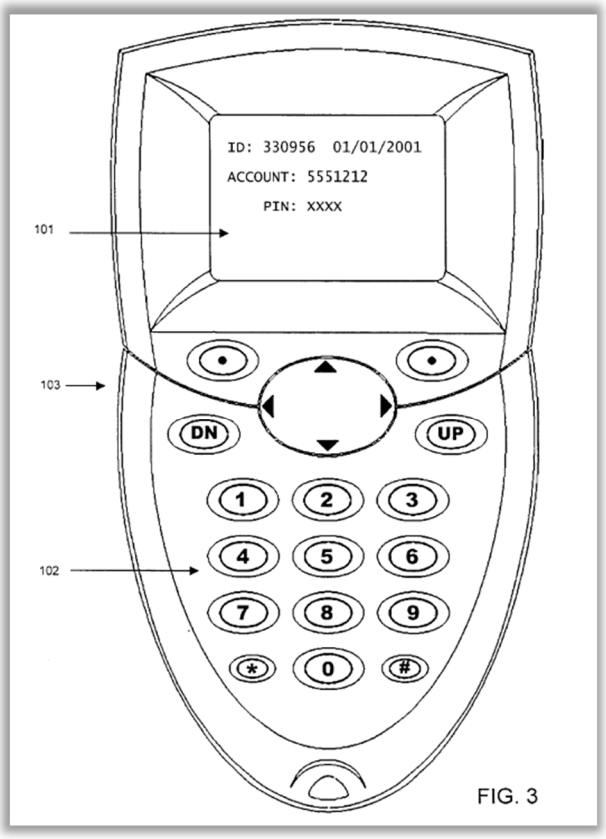
166. The printer polling device taught by Keeney had a display. (Keeney, 9:10-12). It would have been obvious for this display to take the form of a touch sensitive screen, since touch sensitive screens were well known before the '233 Patent's priority date.[9]

e. [18.D]: a graphical user interface over the touch sensitive screen for selecting an icon or function control, and for scrolling display:

167. Figure 3 of Keeney depicted a graphical user interface on the display 101:

[9] See https://en.wikipedia.org/wiki/Touchscreen

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



Keeney, Figure 3

168. The graphical user interface enabled icon selection or function control, such as a user "providing a PIN to the spooling server" (Keeney, 14:34-36) or the polling device "display[ing]…files to the user…and allow[ing] one of the files to be selected, or allow[ing] additional navigation of directory 204 to take place." (*Id.,* 13:19-24). It also would have been obvious for the user interface to scroll the display because scrolling would have enabled the user interface to display more information (e.g., identifiers of more files in an available directory) than would fit on a non-scrollable display.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

f. [18.E]: a client application for interacting with a user via the graphical user interface over the touch sensitive screen of the information apparatus, the client application facilitated, at least partly, by the operating system, wherein execution of at least part of the client application by the processor at the information apparatus configures the information apparatus for:

169. As described with respect to [18.D], a user interacted with Keeney's printer polling device via the graphical user interface to, e.g., input a PIN or select a file to print. It would have been obvious for these interactions to be enabled by a client application facilitated by an operating system (e.g., software on the printer polling device) because a POSITA would have known that software applications were commonly used to perform functions of computing devices.

g. [18.F]: sending, from the information apparatus to one or more servers over a network, security or authentication information associated with the user of the information apparatus for authentication of the user, the one or more servers being distinct and separate from the information apparatus, and in dependence of successful authentication of the user, enabling the user access to the digital content service provided by the one or more servers:

170. See [1.C]. With respect to "the one or more servers being distinct and separate from the information apparatus," see [47.PRE].

h. [18.G]: sending, from the information apparatus to the one or more servers, payment or subscription information for accessing the digital content service provided by the one or more servers:

171. See [1.D].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

i. [18.H]: delivering, from the information apparatus, one or more device objects to the one or more servers over a network, the one or more device objects related to one or more devices, the one or more devices for outputting or playing digital content that are received from the digital content service, and the one or more devices being distinct and separate devices from the one or more servers, and the one or more device objects including information related to at least one of device identification information, device address information, device capability information, model information, brand information, registration information, or IP address information, individually or in any combination, related to the one or more devices:

172. See [1.E].

j. [18.I]: receiving, by the information apparatus and from the one or more servers over the network, one or more pointers or references to one or more digital content that are stored in the one or more servers:

173. See [1.F].

k. [18.J]: receiving, via the graphical user interface over the touch sensitive screen of the information apparatus, information associated with selected digital content from among the one or more pointers or references to the one or more digital content stored in the one or more servers:

174. Keeney disclosed the printer polling device receiving selected digital content (e.g., a selected file) from among the pointers or references delivered to the printer polling device (a directory of documents). (Keeney, 12:18-20; 12:48-49 ("[s]election of a print job 11 from the directory [could] be made (e.g., via a user interface)").

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

l. [18.K]: delivering, from the information apparatus to the one or more servers, at least a pointer or reference to the selected digital content:

175. See [47.A] and [1.G].

m. [18.L]: receiving, at the information apparatus and from the one or more servers, output data corresponding to at least part of the selected digital content, the output data is related, at least in part, to the information included in the one or more device objects delivered from the information apparatus to the one or more servers, and the output data, corresponding to at least part of the selected digital content, is at least in part device dependent with respect to the one or more devices for output or playing at least part of the selected digital content:

176. See [47.H.i] and [47.I].

n. [18.M]: wherein the receiving of the output data is subsequent to having sent, from the information apparatus to the one or more servers, the security or authentication information associated with the user, and the receiving of the output data is further subsequent to having sent, from the information apparatus to the one or more servers, the payment or subscription information:

177. See [1.M.ii].

13. Claim 19:

a. [19.PRE]: …wherein the one or more devices include at least the information apparatus:

178. See [47.PRE].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

b. [19.A]: the information apparatus is further operable for: decoding, at the information apparatus, at least part of the output data received from the one or more servers for rendering or outputting or playing of at least part of the selected digital content:

179. See [6.B].

c. [19.B]: rendering or outputting or playing at least part of the selected digital content corresponding to the decoded output data that is device dependent for rendering at the one or more devices:

180. See [6.B].

14. Claim 20: …in which the network includes one or more of an Internet, a local area network, a cellular network, a wireless network, or a wide area network, individually or in any combination.

181. See [47.PRE]; the network disclosed by Keeney was the Internet.

15. Claim 21: …further comprising transmitting at least part of the received output data or the decoded output data from the information apparatus to an output device for rendering, the output device including one or more of a printing device, a display device, a projection device, a television device, an audio output device, a speaker, or an output controller wire connected to a television, individually or in any combination.

182. See claim 9.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

16. Claim 27:

a. [27.PRE]: A method for accessing digital content, by an apparatus over a network that includes the Internet, the apparatus including a communication unit, one or more processors, an operating system, and one or more client applications facilitated by the operating system and executable by the one or more processors for providing one or more functionalities of the apparatus, the method comprising:

183. As described with respect to [18.PRE], [18.A], [18.B], and [18.E], Keeney disclosed an apparatus (e.g., the printer polling device) with the components recited in [27.PRE] that performed a method for accessing digital content. Keeney further disclosed that the printer polling device included a communication unit, such as a polling transmitter or a printer interface. (Keeney, Fig. 8, 14:31-45). With respect to "a network that includes the Internet," see [47.PRE].

b. [27.A]: establishing, using the communication unit, a connection between the apparatus and one or more servers over a network that includes the Internet, the one or more servers being distinct and separate from the apparatus, and the one or more servers operated, at least partly, by a digital content service that is accessible, at least in part, by the apparatus over the Internet:

184. Keeney disclosed the printer polling device using the polling transmitter to establish a connection to a server (the spooling server) by its description of the printer polling device "polling" the spooling server. (Keeney, 6:16-18 ("The printer polling device uses 'pull' technology, which polls the spooling

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

server so that the spooling server does not have to initiate a connection into a printer"), 14:36-39 ("A polling transmitter 105 is provided for transmitting polling requests to the spooling server 50 via the network 110 to identify a print job stored at the spooling server 50")). Keeney also disclosed that the spooling server was operated by a digital content service accessible by the polling device over the Internet. (*Id.,* 6:42-44 ("Printing services are provided via a spooling server 50 which is capable of receiving and storing one or more print jobs from a print job source 10 via a network 110")). With respect to the spooling server being distinct and separate from the printer polling device, see [47.PRE].

c. [27.B,C]: subsequent to establishing, using the communication unit, the connection between the apparatus and the one or more servers, the method further comprises receiving, at the apparatus over the network from the one or more servers, one or more pointers or references to digital content that is accessible, at least partly, by the apparatus from the digital content service:

185. See [1.F].

d. [27.D]: delivering, from the apparatus to the one or more servers, at least a pointer or reference to selected digital content, the pointer or the reference to the selected digital content being related to a selection by the user, at the apparatus, from among the one or more pointers or references to the digital content accessible by the apparatus from the digital content service, the one or more pointers or references provided, at least in part, to the apparatus by the one or more servers:

186. See [47.A], [1.G], and [18.J].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

e. [27.E]: delivering, from the apparatus, one or more device objects to the one or more servers, the one or more device objects including information corresponding to one or more devices that are distinct and separate devices from the one or more servers, and the information including at least one of device identification information, device address information, device capability information, model information, brand information, or registration information, individually or in any combination, related to the one or more devices:

187. See [1.E].

f. [27.F]: sending, from the apparatus to the one or more servers over the network, security or authentication information associated with a user operating the apparatus, and, in dependence of a successful authentication of the user, enabling the user access to the digital content service:

188. See [1.C].

g. [27.G]: receiving, at the apparatus, from the one or more servers operated, at least partly, by the digital content service, output data, the output data corresponding to the selected digital content and having a relationship, at least in part, to the one or more device objects delivered from the apparatus to the one or more servers over the network that includes the Internet, and the output data corresponding to the selected digital content further including an encryption scheme for ensuring security of the selected digital content for rendering at the one or more devices, and at least part of the output data is device dependent for rendering at least part of the selected digital content at the one or more devices:

189. See [47.H.i] and [47.I].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

h. Claim 29: …in which the apparatus is embodied to be at least one of a smart phone, a screen phone, an Internet pad, an Internet appliance, a speaker, an audio output device, a display device, or a digital camera, individually or in any combination.

190. See claim 7.

i. Claim 30: …further comprising sending a payment or subscription information from the apparatus to the one or more servers over the network as payment or subscription for accessing the digital content service, and, subsequent to having sent the payment or subscription information to the one or more servers, the apparatus receives the output data corresponding, at least in part, to the selected digital content from the one or more servers operated, at least partly, by the digital content service.

191. See [1.D] for payment or subscription information, and [1.M.ii] for receiving output data subsequent to having sent payment or subscription information.

j. Claim 35: …in which the communication unit of the apparatus includes a communication interface and the method further comprises the apparatus establishing a wired or wireless connection, via the communication interface, with a device that includes at least one of a printer, a display device, a television, a controller wire connected to a television, or an audio output device, individually or in any combination, for transmitting at least part of the selected digital content over the wired or wireless connection for rendering at the device.

192. As described with respect to [27.PRE], Keeney's printer polling device had a communication unit including a printer interface (e.g., a communication

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

interface). The printer interface "enable[d] the printer polling device 100 to forward the print job to a printer 120 for printing." (Keeney, 14:44-45). It would have been obvious to a POSITA for the printer polling device to establish a wired or wireless connection with the printer, via the printer interface, since establishing this connection would have enabled the printer polling device to forward the print job to the printer.

k. Claim 36: …further comprising decoding or decompressing at least part of the output data corresponding to at least part of the selected digital content at the apparatus for rendering, and wherein the encryption scheme included with the output data corresponding to the selected digital content includes security keys for ensuring security of the selected digital content.

193. See [6.B] for decoding output data for rendering and claim 48 for an encryption scheme including security keys.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

17. Claim 50:

a. [50.PRE]: In a non-transitory computer memory or storage device at one or more servers or at an information apparatus, containing client software configured to be executable, at least partly, by one or more processors at the information apparatus for obtaining digital content from a digital content service operated, at least partly, over the Internet, the digital content obtained from the digital content service being encoded, at least partly, with an encryption scheme that is acceptable for rendering at one or more devices, the one or more devices includes the information apparatus, the information apparatus including:

194. As described with respect to [18.PRE], [18.A], and [18.E], Keeney disclosed an information apparatus (e.g., a printer polling device) containing client software executable by a processor for obtaining digital content from a digital content service (e.g., a service operated by the spooling server) over the Internet. It would have been obvious for the printer polling device to include a non-transitory computer memory or storage device, since it was well known for computing devices to include a memory or storage device. Finally, with respect to digital content being encoded with an encryption scheme that is acceptable for rendering at one or more devices (e.g., a printer), see [1.K]. Here, the claimed "one or more devices" are satisfied by Keeney's disclosure of the printer polling device (an information apparatus) and the printer.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

b. [50.A]: memory or storage device for storing at least part of the client software:

195. It was well known for computing devices to include a memory or storage device for storing software, and thus it would have been obvious for Keeney's printer polling device to have a memory or storage device.

c. [50.B]: a display screen or a speaker, individually or in any combination:

196. The printer polling device taught by Keeney had a display. (Keeney, 9:10-12).

d. [50.C]: a user interface for interacting with a user:

197. See [18.D].

e. [50.D,E]: the information apparatus further including an operating system for facilitating operations of the client software at the information apparatus, the digital content for rendering at the one or more devices wherein the client software, facilitated by the operating system, is executable, at least partly, by the one or more processors at the information apparatus to perform a method at the information apparatus, the method comprising:

198. See [18.B] for an operating system and [27.PRE] for client software facilitated by the operating system that is executable to perform a method. With respect to the digital content being configured for rendering at the one or more devices, Keeney disclosed a "printer specific data file 805 for delivery to the printer polling device 100 to be printed at the target printer 120." (Keeney, 14:19-22).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

f. [50.F]: establishing a communication channel between the information apparatus and the one or more servers over a network that includes the Internet, the one or more servers being distinct and separate devices from the information apparatus and the one or more devices, and the one or more servers is operated, at least partly, by the digital content service:

199. See [27.A].

g. [50.G]: sending, from the information apparatus to the one or more servers over the network that includes the Internet, security or authentication information associated with the user operating the information apparatus for authentication of the user to access the digital content service:

200. See [47.C].

h. [50.H]: obtaining, at the information apparatus, one or more device objects related to the one or more devices, the one or more device objects including device information related, at least in part, to at least one of device identification information, device address information, device capability information, model information, brand information, or registration information, individually or in any combination:

201. See [1.E].

i. [50.I]: delivering, from the information apparatus, at least part of the one or more device objects to the one or more servers:

202. See [1.E].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

j. [50.J]: providing, over the user interface at the information apparatus, one or more pointers or references to the digital content that are available from the digital content service for selection by the user of the information apparatus:

203. See [1.F].

k. [50.K]: receiving a selected digital content, via the user interface at the information apparatus, from among the one or more pointers or references to the digital content provided over the user interface for selection by the user:

204. See [18.J].

l. [50.L]: obtaining, at the information apparatus, a content object that includes a pointer or reference to the selected digital content:

205. See [18.J].

m. [50.M]: delivering, from the information apparatus, at least part of the content object to the one or more servers:

206. See [18.K].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

n. [50.N]: receiving, from the one or more servers and at the information apparatus, output data that is device dependent for rendering at the one or more devices, the output data corresponding, at least in part, to the selected digital content and having a relationship, at least in part, to the device information included in the one or more device objects delivered from the information apparatus to the one or more servers over the network that includes the Internet, and the received output data further including an encryption scheme applied on at least part of the selected digital content for ensuring security of the selected digital content:

207. See [47.H.i] and [47.I].

o. [50.O]: decoding, at the information apparatus, at least part of the received output data that is at least in part device dependent for rendering at the one or more devices, the output data includes an encryption scheme applied on at least part of the selected digital content:

208. As stated in [50.PRE], the information apparatus receives digital content that is "encoded, at least partly, with an encryption scheme." Likewise, [50.N] states that the received output data has "an encryption scheme applied on at least part of the selected digital content." Thus, Petitioner interprets "decoding, at the information apparatus, at least part of the received output data" to include an operation to decrypt the received, encrypted output data. To this end, Keeney disclosed the printer polling device (the information apparatus) "decrypt[ing] [an encrypted print job] using a key." (Keeney, 8:59-60). Decrypting the print job enabled it to be rendered (e.g., printed) at a printer, one of the claimed "one or more

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

devices." (*Id.,* 8:62-63 (The decrypted print job can then be forwarded to the printer 120 for printing")).

p. [50.P]: rendering at least part of the decoded output data related, at least in part, to the selected digital content at the one or more devices, the one or more devices includes the information apparatus:

209. As discussed in [50.O], Keeney's printer polling device (information apparatus) sent the decrypted print job (decoded output data) to a printer for printing (rendering).

18. Claim 52: …in which the encryption scheme of the output data for ensuring security of the selected digital content includes security keys.

210. See claim 48.

19. Claim 57: …further comprising software executable at the information apparatus for sending, from the information apparatus to the one or more servers over the network that includes the Internet, payment or subscription information for accessing the digital content service provided by the one or more servers, and the receiving of the output data for rendering at the one or more devices is subsequent to having sent the payment or subscription information.

211. See [18.E] for software executable at the information apparatus, [1.D] for payment or subscription information, and [1.M.ii] for receiving output data subsequent to having sent payment or subscription information.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B. Ground 2: Claims 47, 48, and 61 are obvious over Keeney and Vatland

1. Motivation to Combine

212. As discussed below, a POSITA would have used Vatland's disclosed compression and decompression of raster data to transmit the raster data disclosed by Keeney over a network. It is my opinion that a POSITA would have been motivated to combine Keeney and Vatland with reasonable expectation of success.

213. For example, Keeney and Vatland both described transmitting raster data over a network for printing. (Keeney, 14:13-15 and 14:40-42; Vatland, 11:8-10). Vatland taught that raster data can be compressed prior to transmission "to obtain additional network bandwidth" (*id.,* 11:25-26), thereby increasing the speed of the data transfer. (*Id.*, 11:8-10). Thus, a POSITA would have been motivated to compress raster data for transmission over a network, as taught by Vatland, to improve the speed of the data transfer taught by Keeney. (Rabbani, [xx]).

2. Claim 47

214. For the reasons set forth in Ground 1, claim 47 is obvious over Keeney in view of the general knowledge of a POSITA. In the alternative, claim 47 is obvious over Keeney in view of Vatland.

a. [47.PRE]:

215. The discussion of Keeney with respect to [47.PRE] above is incorporated herein.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

b. [47.A]:

216. The discussion of Keeney with respect to [47.A] above is incorporated herein.

c. [47.B]:

217. The discussion of Keeney with respect to [47.B] above is incorporated herein.

d. [47.C]:

218. The discussion of Keeney with respect to [47.C] above is incorporated herein.

e. [47.D]:

219. The discussion of Keeney with respect to [47.D] above is incorporated herein.

f. [47.E]:

220. The discussion of Keeney with respect to [47.E] above is incorporated herein.

g. [47.F]: apply, by the server system, an encoding operation on at least part of the digital content for transmitting the output data that is associated with at least part of the digital content:

221. The discussion of Keeney with respect to [47.F] above is incorporated herein. *See also* ¶¶212-213 and ¶¶111-120, *supra.*

222. To the extent Keeney in view of the general knowledge of a POSITA

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

does not render obvious the "encoding operation" recited in limitation [47.F], it is my opinion that Vatland taught it.

223. Keeney described transmitting rasterized data over a network for printing. (Keeney, 14:13-15 ("the page description language needs to be rasterized into a bitmap form"); 14:40-42 (describing transmission of the print job over the "network 110"); 7:11-12 (explaining that the network 110 included the Internet)).

224. Vatland was generally directed towards "facilitating printing a document over a network by increasing the speed of raster data transfers over the network." (Vatland, 11:8-10). To help achieve the goal of increasing data transfer speed, Vatland disclosed that "data may be compressed and decompressed to obtain additional network bandwidth." (*Id.,* 11:25-26). Compression is a form of encoding, as I explain above. ¶¶ 28-50. Thus, Vatland taught performing an encoding operation on data being transmitted over a network for printing.

225. A POSITA would have been motivated to combine Keeney and Vatland to render obvious the encoding operation recited in [47.F] because, as explained by Vatland, applying a compression operation to rasterized data prior to transmitting the data over a network would have increased the speed of the data transfer. A faster data transfer speed would have enabled faster printing of the transferred document, improving the experience for the user of Keeney's system. *See* ¶¶ 28-50, ¶¶ 212-213, ¶¶ 111-120, *supra.*

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

h. [47.G]:

226. The discussion of Keeney with respect to [47.G] above is incorporated herein. *See* ¶¶ 28-50, ¶¶ 212-213, ¶¶ 111-120, *supra.*

i. [47.H.i]:

227. The discussion of Keeney with respect to [47.H.i] above is incorporated herein.

j. [47.H.ii]:

228. The discussion of Keeney with respect to [47.H.ii] above is incorporated herein.

k. [47.I]:

229. The discussion of Keeney with respect to [47.I] above is incorporated herein.

230. It is my opinion that independent claim 47 is rendered obvious by Kenney in view of Vatland.

3. Dependent claim 48:

231. The discussion of Keeney with respect to claim 48 above is incorporated herein.

4. Dependent Claim 61:

232. The discussion of Keeney with respect to claim 61 above is incorporated herein.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

VIII. CONCLUSION

I hereby declare that all the statements made in this declaration are based on my own true knowledge, that all statements made on information and belief are believed to be true, and that all statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001. I declare under the penalty of perjury under the laws of the United States that all statements made in this declaration are true and correct.

Dated this 28th day of April, 2023

Majid Rabbani

Dr. Majid Rabbani

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Majid Rabbani – Curriculum Vitae

Majid Rabbani
36 Wexford Glen
Pittsford, NY 14534-4206
+1-585-586-7025 Home
+1-585-506-7369 Cell
rabbani6@yahoo.com

Current Employment (since August of 2016)

Visiting Full Professor from 2016-2021 and Professor of Practice from 2021
Electrical Engineering Department
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, NY 14623-5603

Education

Ph.D.	Electrical and Computer Engineering, 1983 University of Wisconsin-Madison Thesis Title: *Restoration of Nonlinearly Distorted Images* GPA - 4.0/4.0
M.S.	Electrical and Computer Engineering, 1979 University of Wisconsin-Madison GPA - 4.0/4.0
B.S.	Electrical Engineering, 1978 AryaMehr (now Sharif) Institute of Technology, Tehran, Iran GPA - 3.9/4.0 (Ranked 1st, five years in a row, in the entire school of engineering)
College Entrance	Ranked 2nd among 103,000 participants in the nationwide College Entrance Examination (*Concours*) in Iran - 1973.

Research Interests

Image and video compression research and standardization
Digital image & video processing and analysis
Computer vision, pattern recognition, and machine learning
Image security including data hiding, embedding, and encryption
Image and video enhancement, deblurring, and super-resolution

Honors and Awards (in chronological order)

- **SPIE Directors' Award** - presented to an individual who, in the opinion of the SPIE Board of Directors, has rendered a significant service of outstanding benefit to the Society (2016).
- **Electronic Imaging Scientist of the Year Award**, awarded by IS&T and SPIE *for fundamental contributions to image compression and for more than 25 years of sharing this knowledge through courses taught at EI* (2015).
- **Electronic Imaging Distinguished Educator Award**, awarded by IS&T and SPIE for *25 years of educational service to the electronic imaging community* (2012).
- IEEE **Region 1 Technical Award** for pioneering work in *image compression and digital watermarking* (2008).
- Co-recipient of the **Kodak C.E.K. Mees Award** (Kodak's highest honor for scientific research and reporting) for the Technical Report entitled "*Kodak Robust, Invisible Watermarking: Concepts, Performance, and Applications*," (2005).
- Eastman Innovation Award Nominee for contributions to the JPEG2000 standard (2002).
- Inducted in the Kodak Inventors Gallery as a **Kodak Distinguished Inventor** for achieving the milestone of 20 issued patents (2002) – currently 44 issued patents.
- **Kodak Fellow** (2000)
- **Fellow of IEEE** – Citation: *For the development of image compression technology and products* (1998). *Currently a Life fellow.*
- **Kodak Senior Research Associate** (A rank held at that time by ~3% of Kodak researchers) (1997).
- Lifetime member of the Kodak Research Scientific Council (1997).
- Co-recipient of the **Emmy Engineering Award** granted by the National Academy of Television Arts and Sciences for contributions to and co-editorship role in the JPEG and MPEG standardization committees (1997).
- Kodak *Mittelstadt* instructor of the year award in recognition of excellence in development and teaching of digital image processing courses for the Kodak Image Science Career Development Program (1995).
- Recognition Award for the development and teaching of a course at the request of Kodak CTO on "Introduction to Digital Imaging" for 24 Kodak Senior VP's (1994).
- **Fellow of SPIE** (1993). *Currently a Life Fellow.*
- Co-recipient of the **Emmy Engineering Award** (for providing image compression research) granted by the National Academy of Television Arts and Sciences to Kodak in recognition of the Kodak Still-Video Transceiver System SV9600 (1990).
- Co-recipient of the **Kodak C.E.K. Mees** award (Kodak's highest honor for scientific research and reporting) as the principle author of the Technical Report entitled "*Detective Quantum Efficiency of imaging systems with amplifying and scattering mechanisms*," (1988).
- Recognition Award for achieving the highest GPA in the entire school of engineering and graduating with honors from the EE Department at the AryaMehr University (currently Sharif University), Tehran, Iran (1978).
- Recognition Award (personally received from the Iranian Prime Minister) for achieving the 2nd rank among 103,000 participants in the nationwide College Entrance Examination (*Concours*) in Iran (1973).

Newspapers, Magazines and Television Interviews (in chronological order)

- **Information Today** Magazine interview on the status of digital watermarking technology entitled "*The Copyright Capers Continue*," Vol. 24, No. 9, October 2007.
- **El Pais** Spanish newspaper interview regarding leadership in JPEG2000 entitled "*Majid Rabbani: Es major desarrollar estandares abiertos que propietarious*," Madrid, Spain, October 2, 2003.
- **Democrat and Chronicle** newspaper interview article for contributions to digital watermarking used in digital cinema entitled "*Kodak builds digital trap for movie pirates*," Rochester, NY, April 2001.
- **Persian Heritage** Magazine interview for technical accomplishments in the field of digital imaging (Majid's picture appeared on the cover page), No 18, Washington D.C., July 2000.
- **Photo Electronic Imaging** Magazine interview regarding the Photo CD product entitled "*A Lossless Explanation of Photo CD*," Vol. 37, No 2, 1994.
- **CBS** (Rochester local Channel 8) interview broadcasted at the Rochester CBS nightly news for directing the efforts of Los Alamos scientists in the digital enhancement of George Holliday's video of the beating of Rodney King and Majid's subsequent testimony as the prosecution expert witness in the Federal Court in Los Angeles, April 1993.
- **Iran Times** newspaper article regarding Majid's expert witness testimony in Rodney King's trial, Washington D.C., April 9, 1993.
- **Democrat and Chronicle** newspaper interview for expert witness testimony in Rodney King's trial, Rochester, NY, March 8, 1993.

Professional Activities

- Judge for granting the ICIP Best Paper Award, October 2020.
- Chair of Search Committee for Editor in Chief of the Journal of Electronic Imaging (2020)
- Chair of the SPIE Commercial + Scientific Sensing and Imaging Symposium, Anaheim, CA, April 2017.
- Associate Editor, IEEE Transactions on Circuits and Systems for Video Technology (CSVT), (2014-2018).
- Chair of the SPIE Fellows Committee responsible for selection of Fellows (2013-2017)
- Vice President, IS&T, (2015-2017).
- Co-Chair of the SPIE Commercial + Scientific Sensing and Imaging Symposium, Baltimore, MD, April 2016.
- Short Course Chair and Co-Chair, SPIE/IS&T Electronic Imaging Symposium, (2014-2016).
- Editorial Member of SPIE eBooks (2014-2015).
- Member of the IEEE IVMSP TC (Image, Video, and Multidimensional Signal Processing Technical Committee, which oversees the ICASSP and ICIP conferences) (2010-2015)
- Member of Electronic Imaging Symposium Steering Committee (EISC), (2010-2016).
- Symposium Chair, SPIE/IS&T Electronic Imaging Symposium (Symposium consisted of 23 conferences in the field of electronic imaging), Burlingame, CA, (January 2012).
- Symposium Co- Chair, SPIE/IS&T Electronic Imaging Symposium, Burlingame, CA, (January 2011).
- Member of the Program Committee of VICP, 2011-2015.
- Member of the SPIE Fellows Committee responsible for selection of Fellows (2010-2012)
- Technical Area Chair of IEEE ICIP 2019, ICIP 2018, ICIP 2016, ICIP 2015, ICIP2011, ICIP 2010, ICASSP 2016, ICASSP 2014, ICASSP 2013, ICASSP 2011.
- Conference co-chair, Visual Communication and Image Processing (VCIP) 2009, San Jose, CA, January 2009, (SPIE Vol. **7257**).
- Board Member of IEEE Signal Processing Magazine (SPM), (2008-2010).
- United States Patent and Trademark Office (USPTO) invited *Chief Technical Expert*. residing on site for one week each summer (2007 and 2008) as a consultant for planning and execution of a comprehensive training program of patent examiners in the field of digital image and video processing, (2007 & 2008).
- Chair of the IEEE Rochester Signal Processing Chapter, Rochester, NY (2007).
- Chair of Western NY Image Processing Workshop, Rochester, NY, (September 2007).
- Awards Committee Chair of the VCIP 2005 and 2006 Conferences, San Jose, CA.
- Industrial Research Chair for the National Sciences and Engineering Research Council (NSERC) of Canada “Digital Cinema” Project, (2004-2009).
- Industry representative to the *Digital Cinema Compression Experts Panel* of DCI (Digital Cinema Initiatives), which selected the digital cinema compression algorithm, (2003-2004).

- Technical Chair of the VCIP Conference, Lugano, Switzerland, (July 2003).
- Tutorial Chair of ICIP, Rochester, NY, (October 2002).
- Book review: "*JPEG2000: Image Compression Fundamentals, Standards and Practice*," Journal of Electronic Imaging, p. 286, Vol. 11, No 2, (April 2002).
- Member of the Program Committee of the IEEE Packet Video, (2000).
- Member of the IEEE IMDSP TC (Image, and Multidimensional Signal Processing Technical Committee, which oversees the ICIP and ICASSP conferences), (2000-2006).
- Chair of the Coding Efficiency subgroup (50 members) of JPEG2000 (1997-2000).
- Guest co-editor, special issue of Journal of Electronic Imaging on *Image/Video Compression and Processing for Visual Communications*, (July 1998).
- Conference co-chair, Visual Communication and Image Processing (VCIP) '98, San Jose, CA, January 1998, (SPIE Vol. **3309**).
- Symposium co-chair of the SPIE/IS&T Electronic Imaging, San Jose, CA, (January 1996).
- Conference co-chair "Still Image Compression: Algorithms and Technologies," SPIE/IS&T EI Symposium, (SPIE Vol. **2418**), (January 1995).
- Organized and chaired a 2-day event at the request of the National Research Council entitled "*Maintaining and Certifying Image and Instrument Integrity.*" Co-authored the Proceedings used by the Computer Science and Telecommunications Board (CSTB) to identify research topics and allocate government funding, Washington D.C., June 15-16, 1995.
- NSF Panelist, *Signal Processing for the National Information Infrastructure (NII)*, 49-page proceedings published by the NSF, August 18-19, 1994.
- NSF NYI Panel reviewer, Washington D.C., May 3, 1994.
- Directed an image processing team in Los Alamos National Laboratory to digitally enhance the George Holliday videotape of the beating of Rodney King, Los Alamos (November 1992-March 1993). Subsequently testified as the prosecution expert witness in Federal court in LA.
- Conference co-chair "Image and Video Compression," SPIE/IS&T EI Symposium, (SPIE Vol. **2186**), January 1994.
- Conference co-chair "Image and Video Processing," SPIE/IS&T EI Symposium, (SPIE Vol. **1903**), January 1993.
- Conference co-chair "Image Processing Algorithms and Techniques III," SPIE/IS&T EI Symposium, (SPIE Vol. **1657**), January 1992.
- Associate Editor, Journal of Electronic Imaging, 1991 (since its foundation)-present.
- Member of the Program Committee of VCIP, 1997-2005.
- Member of the Program Committee of DCC, 1997-present.
- Reviewer for IEEE TIP, Signal Processing Letters, Optical Engineering, Journal of Electronic Imaging, Image Communications Journal, etc.
- Member of technical committee of conferences and workshops (too many to enumerate).
- Session Chair at IEEE, SPIE and IS&T conferences since 1985 (too many to enumerate).
- Active reviewer for ICIP, VCIP, ICASSP, DCC, PCS (too many to enumerate).

Educational and Mentoring Activities

Visiting professor: RIT, EE Department, Fall 2016 - Present

1. Fall 2016	EEEE779 – Digital Image Processing
2. Fall 2016	EEEE678 – Digital Signal Processing
3. Spring 2017	EEEE781 – Image and Video Compression
4. Spring 2017	EEEE794 – Information Theory
5. Spring 2017	EEEE281 – Circuits I
6. Fall 2017	EEEE281 – Circuits I
7. Fall 2017	EEEE779 – Digital Image Processing
8. Fall 2017	EEEE678 – Digital Signal Processing
9. Spring 2018	EEEE281 – Circuits I
10. Spring 2018	EEEE781 – Image and Video Compression
11. Spring 2018	EEEE670 – Pattern Recognition
12. Summer 2018	1-day seminar on Machine Learning
13. Fall 2018	EEEE281 – Circuits I
14. Fall 2018	EEEE779 – Information Theory
15. Fall 2018	EEEE678 – Digital Signal Processing
16. Fall 2018	EEEE781 – Image and Video Compression
17. Summer 2019	1-day seminar on Machine Learning
18. Fall 2019	EEEE281 – Circuits I
19. Fall 2019	EEEE779 – Information Theory
20. Fall 2019	EEEE678 – Digital Signal Processing
21. Spring 2019	EEEE281 – Circuits I
22. Spring 2019	EEEE781 – Digital Image Processing
23. Spring 2019	EEEE670 – Pattern Recognition
24. Fall 2020	EEEE281 – Circuits I - Online
25. Fall 2020	EEEE602 – Random Signals and Noise (Online)
26. Fall 2020	EEEE678 – Digital Signal Processing
27. Spring 2020	EEEE281 – Circuits I - Online
28. Spring 2020	EEEE602 – Random Signals and Noise (Online)
29. Spring 2020	EEEE670 – Pattern Recognition

Adjunct professor: RIT, EE Department Graduate Program – 1985-1992, 2011-2016

1. *Digital Image Processing*, (graduate level course) fall semester 2015, fall semester 2014.
2. *Image and Video Compression* (graduate level course - online version), Spring semester 2016, Spring semester 2015, spring quarter 2014, summer quarter 2013.
3. *Image and Video Compression* (developed online version) winter quarter 2012-2013.
4. *Image & Video Compression* (developed the graduate course) spring quarter 2012.
5. *Information Theory* (graduate level course) winter quarter 2011-2012.
6. *Error Control Coding*, (graduate level course) 1987, 1989 & 1992.
7. *Pattern Recognition* (First time offering; developed the graduate course), 1985 & 1988.
8. *Information Theory and Coding* (First time offering; developed the graduate course), 1986.

Adjunct professor: University of Rochester, EE Department, 2012

1. *Digital Signal Processing* (EE246/446, fall semester 2012).

National Technological University (NTU) Advanced Technology Programs – 1989-2000

1. *Digital Image Compression* (Course MC891016A1), October 16 & 18, 1989.
2. *Digital Image Enhancement* (Course MC921210B1), December 10, 1992.
3. *Image and Video Compression Fundamentals and international Standards* (Course MC930422B1), April 22, 1993.
4. *Introduction to Electronic Imaging* (Course MC95051802), May 18, 1995.
5. *Introduction to Digital Image Processing* (Course MC97111292), November 12 & 19, 1997.
6. *Introduction to Electronic Imaging Systems* (Course EN00082399), August 23, 2000.
7. *Fundamentals of Wavelet Image Compression and the Emerging JPEG-2000 Standard* (Course EN00083091), August 30, 2000.

RIT Center for Imaging Science (CIS) and Technical and Education Center (T&E)

1. *Digital Image Restoration* – (1-day seminar), RIT CIS summer school, July 27, 1988.
2. *Digital Image Enhancement, Restoration and Compression* – (2-day seminar), RIT CIS 1989-1992.
3. *Digital Image Processing*– (4-day seminar), RIT T&E Center, 1998-2000.
4. *Digital Image & Video processing and Compression* – (4-day seminar), in collaboration with RIT T&E Center, VIGC, Turnhout, Belgium, 2001, 2002.
5. *Digital Image Processing* – (4-hour seminar as part of *Printing Process Identification and Image Analysis for Forensic Document Examiners*), twice a year, 2004-2018.

Portland State University (PSU) – 1995-2004

1. *Digital Image & Multimedia Compression* (3-day seminar), 1995-2001.
2. *Digital Image Watermarking & Compression* (2-day IEEE Lecture Series), 2002.
3. *Digital Image & Video Compression* (1-day IEEE Lecture Series), 2003, 2004.

MIT - 1993

Digital Image and Video Compression – 2-day invited seminar (invited by Professor Jae Lim) presented to the Consortium on Advanced Television Systems (CATS), January 11-13, 1993.

U.C. Berkeley Extension - 2003

JPEG2000: Theory and Practice – 2-day course at U.C. Berkeley Extension, March 24-25, 2003.

SPIE Educational Video/CD-ROM (currently marketed by SPIE for corporate training)

1. *Digital Image Enhancement* (5-hour video, VT558), 1993.
2. *Image and Video Compression Fundamentals and International Standards* (5-hour video, VT589), 1993.
3. *Introduction to Electronic Imaging* (5-hour video, VT051895), 1995.
4. *Introduction to Digital Image Processing* (10-hour video, VT245), 1997.
5. *Introduction to Electronic Imaging Systems* (5-hour CD-ROM, CDV064), 2000.
6. *Fundamentals of Wavelet Image Compression and the Emerging JPEG-2000 Standard* (5-hour video, VT080), 2000.

Internal to Eastman Kodak Company

- At the request of Kodak CTO, developed a 2-day class on *Introduction to Digital Imaging* for 24 Kodak Senior VP's (1994), which was also videotaped and used for corporate training.

- *Digital Image Processing* (2-day) class presented once a year to the "Image Science Career Development" Program, **1992-2004**. Majid was awarded the *Mittelstadt* instructor of the year award in 1995 for excellence in development and teaching of these courses.
- Taught courses on various topics of digital imaging at the Kodak Interplant Conference each year (1998-present).

Other Educational Activities

Additionally, Majid has taught about 70 short courses and seminars at the various SPIE, IS&T and IEEE conferences in the past 30 years. In January of 2012 he was awarded the "Electronic Imaging Distinguished Educator Award," by IS&T and SPIE for teaching short courses in the various areas of electronic imaging at the SPIE/IS&T flagship conference, the EI Symposium, for 25 consecutive years in the period of 1988-2012. He has continued teaching at that venue since then for a record 33 consecutive years.

Patents

1. Andrew Gallagher, Jiebo Luo, and Majid Rabbani, "Determining a Stereo Image From Video," US Patent 9,113,153 (2015).
2. Andrew Kurtz, Kurt Schroeder, and Majid Rabbani, "Spectral Visible Edge Marking For Steganography or Watermarking," US Patent 8,947,744 (2015).
3. Andrew Kurtz, Kurt Schroeder, and Majid Rabbani, "Spectral Edge Marking For Steganography or Watermarking," US Patent 8,941,886 (2015).
4. Paul Kane, Sen Wang, and Majid Rabbani, "Range Measurement Using Symmetric-Coded Apertures," US Patent 8,330,852 (2012).
5. Z. Sun and Majid Rabbani, "Enhancing Image Resolution of Digital Images," US Patent 8,289,395 (2012).
6. Jiebo Luo, *et al*, "Analyzing Camera Captured Video for Key Frames," US Patent 8,031,775 (2011).
7. Aaron Deever, *et al*, "Digital Camera Using Multiple Image Sensors to Provide Improved Temporal Sampling," US Patent 7,978,239 (2011)
8. Majid Rabbani *et al*, "Image Sequence Stabilization Method and Camera Having Dual Path Image Sequence Stabilization," US Patent 7,755667 (2010).
9. Rajan Joshi and Majid Rabbani, "Transcoding a JPEG2000 Compressed Image," US Patent 7,382,926 (2008).
10. Zhaohui Sun, Gabriel Fielding and Majid Rabbani, "Method and System For Video Filtering with Joint Motion and Noise Estimation," US Patent 7,295,616 (2007).
11. Rajan Joshi and Majid Rabbani, "Method for Transcoding a JPEG2000 Image," US Patent 7,200,277 (2006).
12. Michael Miller, Paul Jones, Jian Yang and Majid Rabbani, "Method and System for Displaying an Image," US Patent 7,075,553 (2006).
13. Thomas Stephany, Majid Rabbani, John Squilla, Donald Olson, "Method for Authenticating Animation," US Patent 6,993,162 (2006).
14. Michael Miller, Paul Jones, Jian Yang and Majid Rabbani, "Method and System for Displaying an Image," US Patent 6,985,158 (2006).
15. Paul Jones, Majid Rabbani, John Hoover, "Method and System for Determining DCT Block Boundaries," US Patent 6,895,122 (2005).
16. Majid Rabbani, Rajan Joshi, Kenneth Parulski, "Selectively Adjusting the Resolution Levels or the quality Levels of Digital Images Stored in a Digital Camera," US Patent 6,885,395 (2005).
17. Paul Jones, Chris Honsinger, Majid Rabbani, "Method and System for Embedding Message Data in a Digital Image Sequence," US Patent 6,870,931 (2005).
18. Majid Rabbani and Aaron Deever, "Digital Image Compression Utilizing Shrinkage of Wavelet Coefficients," US Patent 6,853,318 (2005).
19. Andreas Savakis, Majid Rabbani, Stephen Etz, "System and Method for Controlling Image Compression Based on Image Emphasis," US Patent 6,832,006 (2004).

20. Paul Jones, Chris Honsinger and Majid Rabbani, “System and Method for Embedding a Watermark Signal that Contains a Message Data in a Digital Image,” US Patent 6,792,130 (2004).
21. Rajan Joshi, Bernard Brower, Kenneth Parulski and Majid Rabbani, “Producing a Compressed Digital Image Organized Into Layers Having Information Reating to Different Viewing Conditions and Resolution,” US Patent 6,785,423 (2004).
22. Andrew Gallagher, Majid Rabbani, Edward Gindele and Robert Gray, “Method for Compressing an Image from a Sparsely Sampled Extended Dynamic Range Image Sensing Device,” US Patent 6,765,611 (2004).
23. Thomas Henderson and Majid Rabbani, “Method for Compressing and Decompressing Digital having Text,” US Patent 6,625,323 (2003).
24. Chris Honsinger, Majid Rabbani and Paul Jones, “Method for Efficient Rate Control,” US Patent 6,356,668 (2002).
25. Paul Jones, Chris Honsinger and Majid Rabbani, “Method and Apparatus for Visually Optimized Compression Parameters,” US Patent US Patent 6,349,151 (2002).
26. Chris Honsinger, Paul Jones, Majid Rabbani and James Stoffel, “Lossless Recovery of an Original Image Containing Embedded Data,” US Patent 6,278,791 (2001).
27. Chris Honsinger, Majid Rabbani and Paul Jones, “Method and Apparatus for Visually Optimized Rate Control,” 6,212,302 (2001).
28. Chris Honsinger and Majid Rabbani, “Method for Generating an Improved Carrier For Use in an Image Data Embedding Application,” US Patent 6,044,156 (2000).
29. Jonathan Riek, Majid Rabbani and Arif Erdem, “Method and Apparatus for Encoding High Fidelity Still Images in MPEG Bitstreams,” US Patent 5,987,179 (1999).
30. Majid Rabbani and Chris Honsinger, “Encoding Color Image Data for Multipass Color Printers,” US Patent 5,956,467 (1999).
31. John Squilla, Omid Moghadam and Majid Rabbani, “Photographic System with Selected Area Authentication,” US Patent 5,898,779 (1999).
32. Omid Moghadam, Majid Rabbani and Kevin Townsend, “Secure Photographic Systems,” US Patent 5,801,856 (1998).
33. Bhavan Gandhi, Majid Rabbani and Rodney Miller, “Method and Apparatus for Suppressing Blocking Artifacts in Block-Transform Coded Images,” US Patent 5,737,451 (1998).
34. Majid Rabbani, “Method and Apparatus for Reducing Quantization Artifacts in a hierarchical Image Storage and Retrieval System,” US Patent 5,568,570 (1996).
35. Majid Rabbani and Paul Melnychuck, “Method and Associated Apparatus for Encoding Bitplanes for Improved Coding efficiency,” US Patent 5,442,458 (1995).
36. Majid Rabbani and Craig Smith, “Electronic Camera Using Image Compression Feedback for Improved Color Processing,” US Patent 5,412,427 (1995).
37. Yushing Tsai, Scott Daly and Majid Rabbani, “Image Compression with Color Interpolation for a Single Sensor Image System,” US Patent 5,172,227 (1992).
38. Yushing Tsai, Kenneth Parulski and Majid Rabbani, “Compression Method and Apparatus for Single-Sensor Color Imaging Systems,” US Patent 5,065,229 (1991).

39. Yushing Tsai, Kenneth Parulski and Majid Rabbani, “Compression Method and Apparatus for Single-Sensor Color Imaging Systems,” US Patent 5,053,861 (1991).
40. Paul Jones and Majid Rabbani, “Delta Modulation Encoder/Decoder Method and System,” US Patent 4,905,002 (1990).
41. Paul Jones and Majid Rabbani, “Differential Pulse Code Modulation scheme Incorporating a Reconstructed Value Constrainer,” US Patent 4,847,866 (1989).
42. Scott Daly, Majid Rabbani and Cheng-Tie Chen, “Digital Image Compression and Transmission System Visually Weighted Transform Coefficients,” US Patent 4,780,761 (1988).
43. Scott Daly, Majid Rabbani and Cheng-Tie Chen, “Adaptive Block Transform Image Coding Method and Apparatus,” US Patent 4,774,574 (1988).
44. Cheng-Tie Chen, Majid Rabbani and Scott Daly, “Image Transmission System with Preview Mode,” US Patent 4,774,562 (1988).

Books

1. M. Rabbani and P. W. Jones, *Digital Image Compression Techniques*, SPIE Tutorial Text Series, **TT7**, Bellingham, WA, Feb. 1991. (*This book has gone through seven printings and has sold about 7,000 copies, breaking the sales record of any SPIE publication*).
2. M. Rabbani (editor), *Selected Papers on Image Coding and Compression*, SPIE Milestone Series, **MS48**, Bellingham, WA, Apr. 1992.

Book Chapters

1. Majid Rabbani, Rajan Joshi and Paul W. Jones, “JPEG2000 Core Coding system (Part 1)” *The JPEG2000 Suite*, Schelkens *et al*, Editors, pp. 3-69. John Wiley & Sons, 2009.
2. R. Chandramouli, Nasir Memon, Majid Rabbani, “Digital Watermarking,” in *The Encyclopedia of Imaging Science and Technology*, Joseph P. Hornak, Editor, Vol 1, pp. 158-172, John Wiley & Sons, 2002.
3. P. W. Jones and M. Rabbani, "JPEG Compression in Medical Imaging," in *Handbook of Medical Imaging, Volume 3. Display and PACS*, Yongmin Kim and Steven C. Hori, Editors, pp. 221-275, SPIE Press, Bellingham, WA, 2000.
4. P. W. Jones and M. Rabbani, "Digital Image Compression," in *Digital Image Processing: Fundamentals and Applications*, E. Dougherty, Editor, Marcel Dekker, New York, 1995.

Invited Presentations, Plenary or Keynote Speeches

1. Majid Rabbani (invited seminar) “Super-resolution: Traditional and Deep Learning Based Approaches” RIT Center for Imaging Science Seminar Series, April 21, 2021.
2. .Majid Rabbani (RIT MSEE Graduate Seminar Presentation), “Image Understanding” Rochester Institute of Technology, April 6, 2018.
3. Majid Rabbani (invited seminar) “Importance of Motion in Digital Video Processing” IEEE Signal Processing Society Seminar, Rochester Institute of Technology, April 4, 2017.
4. Majid Rabbani (RIT MSEE Graduate Seminar Presentation), “Motion Processing,” Rochester Institute of Technology, March 24, 2017.
5. Majid Rabbani (invited speaker), “Advances and Challenges in Image Understanding,” Presentation at the SPIE Fellows Luncheon during SPIE DCS, Baltimore, MD, April 18, 2016.
6. Majid Rabbani (invited workshop), “Advanced Image Processing,” Image Sensors (IS) Americas Conference, Berkeley, CA, November 17, 2015 (http://www.image-sensors.com/image-sensors-americas/pre-conference-workshop).
7. Majid Rabbani (RIT MSEE Graduate Seminar Presentation), “Image Understanding,” Rochester Institute of Technology, November 6, 2015.
8. Majid Rabbani (invited talk), “Event-Based Frameworks for Multimedia Content Management,” Penn State University EE Colloquia, February 16, 2012.
9. Majid Rabbani (2-hour invited seminar), “Understanding and Interpreting Images and Video,” VCIP Conference, Huangshan, China, July 11, 2010.
10. Majid Rabbani (plenary talk), “Understanding and Interpreting Images – An Overview,” ICSCCW, Cypress, September 2, 2009.
11. Majid Rabbani (invited talk at the request of local IS&T Chapter), “Opportunities in Video,” RIT Center for Imaging Science, September 26, 2008.
12. Majid Rabbani (invited lecture), “The JPEG2000 Wavelet Based Compression Standard,” 2-hour lecture at the US Patent and Trademark Office (USPTO), July 24, 2008.
13. Majid Rabbani (invited lecture), “Video Processing Algorithms,” 2-hour lecture at the US Patent and Trademark Office (USPTO), July 23, 2008.
14. Majid Rabbani (invited lecture), “Contrast Enhancement,” 2-hour lecture at the US Patent and Trademark Office (USPTO), July 22, 2008.
15. Majid Rabbani (invited lecture), “Image Deblurring and Sharpening Algorithms,” 3-hour lecture at the US Patent and Trademark Office (USPTO), July 19, 2007.
16. Majid Rabbani (invited lecture), “Digital Image Watermarking,” 3-hour lecture at the US Patent and Trademark Office (USPTO), July 18, 2007.
17. Majid Rabbani (invited lecture), “The JPEG2000 Compression Standard,” 3-hour lecture at the US Patent and Trademark Office (USPTO), July 17, 2007.
18. Majid Rabbani (invited talk), “Robust Invisible Watermarking,” in Frontiers In Imaging, International Congress of Imaging Science (ICIS’06), Rochester, NY, May 11, 2006.
19. Majid Rabbani (invited talk), “The JPEG2000 wavelet-based still image compression standard”, RIT Department of Mathematics and Statistics Colloquia Series, February 3, 2006.

20. Majid Rabbani (distinguished lecture), “Digital Watermarking: Technology and Applications,” SPIE Workshop on Images, Buenos Aires, November15, 2005.
21. Majid Rabbani (plenary talk), “Robust Invisible Digital Watermarking of Images”, Western New York Image Processing Workshop, RIT, September 30, 2005.
22. Majid Rabbani and Bernie Brower (invited lecture), “The JPEG 2000 Image Compression Standard: A new generation of capabilities,” University of Rochester IEEE Student Branch, February 7, 2005.
23. Majid Rabbani (invited seminar), “The Emerging JPEG2000 Compression Standard: Technology and Practice,” 3-hour seminar sponsored by IEEE Spain, Madrid, Sep. 19, 2003.
24. Majid Rabbani (invited lecture) “Robust Invisible Digital Watermarking of Images,” 1-hour lecture at RIT Center for Imaging Science Seminar Series, March 14, 2003.
25. Majid Rabbani and Bernard Brower (invited lecture) “The JPEG2000 Wavelet-Based Still Image Compression Standard: Status Report and Implementation Demo,” 90-minute lecture at the Rochester Chapter of IEEE Signal Processing Society, CEIS, February 19, 2003.
26. Majid Rabbani (invited lecture) “Invisible Digital Watermarking of Images,” 1-hour lecture at the request of the Rochester Chapter of IS&T, RIT, November 13, 2002.
27. Majid Rabbani (distinguished lecture) “Invisible Digital Watermarking of Images,” Center for Advanced Computer Studies, Distinguished Lecture series, UL Lafayette, Nov. 1, 2002.
28. Majid Rabbani (invited seminar) “Digital Image Watermarking” ECE Graduate Seminar Series, University of Rochester, May 2001.
29. Majid Rabbani (distinguished lecture) “Invisible Digital Watermarking of Images,” IEEE Signal Processing Society Colloquia, UT-Dallas, October 26, 2001.
30. Majid Rabbani (invited talk), “A Technical Overview of the JPEG2000 Standard,” ISO Meeting, Singapore, March 5, 2001.
31. Majid Rabbani (invited plenary talk), “Review of Watermarking Technologies,” Photo Marketing Association (PMA), Las Vegas, Nevada, January 30, 2000.
32. Majid Rabbani (plenary talk), “Review of State-of-the-art Digital Image Watermarking Technologies”, Western New York Image Processing Workshop, September 17, 1999.
33. Majid Rabbani (invited lecture), “The Emerging JPEG2000 Compression Standard” 1-hour lecture at the IEEE Colloquium, Cornell University, December 1, 1998.
34. Majid Rabbani (invited lecture), “The JPEG2000 Emerging Still-Image Compression Standard,” IEEE Signal Processing Society Rochester Section, September 1997.
35. Majid Rabbani (invited panelist), “Digital Video” IEEE Rochester Section Joint Chapters Meeting, March 12, 1997.
36. Majid Rabbani (invited seminar), “Digital Image Processing,” 2-hour seminar for Society for Information Display (SID) Annual Meeting, May 14, 1995.
37. Majid Rabbani (invited lecture), “Image Compression Fundamentals” SMPTE Annual Forum – “The Compression Experience”, Rochester, NY, October 28, 1995.
38. Majid Rabbani (distinguished lecture), “Digital Image Compression,” 1-hour lecture at Portland State University (PSU), May 26, 1995.

39. Majid Rabbani (distinguished lecture), “Digital Image Compression: Research and Standardization,” 90-min lecture at University of Nevada Las Vegas (UNLV), July 28, 1994.
40. Majid Rabbani (invited seminar), “Digital Image Enhancement,” 75-minute seminar for Society for Information Display (SID) Annual Meeting, June 13, 1994.
41. Majid Rabbani (guest lecture), “World Standards in Image Data Compression” IEEE Rochester Section Joint Chapters Meeting, February 5, 1992.
42. Majid Rabbani (invited lecture), “Status of World Standardization in Image Data Compression”, RIT Center for Imaging Science, October 22, 1991.
43. Majid Rabbani (invited lecture), “Digital Image Compression: The New International Standards” Polytechnic University Imaging Colloquia, NYC, October 1, 1991.
44. Majid Rabbani (invited seminar), “Digital Image Compression,” 2-hour seminar for Society for Information Display (SID), Las Vegas, NV, May 14, 1990.
45. Majid Rabbani (invited lecture), “Digital Image Compression” IEEE Computer Society Rochester Chapter, May 9, 1990.
46. Majid Rabbani (invited seminar), “Digital Image Restoration” RIT Electrical Engineering Seminar Series on DSP, March 31, 1987.
47. Majid Rabbani (invited seminar), “Recent Advances in Image Compression Techniques” RIT Center for Imaging Science Seminar Series, November 12, 1986.

Refereed Journal Publications

1. M. Park, J. Luo, A. Gallagher, and Majid Rabbani, "Learning to Produce 3D Media from a Captured 2D Video," *IEEE Transactions on Multimedia*, **15**(7), pp. 1569-1578, Nov. 2013.
2. M. Rabbani and Rajan Joshi (invited paper), "An Overview of the JPEG2000 Still Image Compression Standard" *Signal Processing: Image Communications*, **17**(1), pp. 3-48, Jan. 2002.
3. J. R. Price and M. Rabbani, "Biased reconstruction for JPEG decoding," *Signal Processing Letters*, **6** (12), pp. 297-299, Dec. 1999.
4. M. Rabbani and P. Melnychuck, "Conditioning contexts for the arithmetic coding of bit planes," *IEEE Trans. Signal Proc.*, **40**, pp. 232-236, Jan. 1992.
5. M. Rabbani and P. W. Jones (Invited Paper), "Image compression techniques for medical diagnostic imaging systems," *Journal of Digital Imaging*, **4**, pp. 65-78, May 1991.
6. R. L. VanMetter and M. Rabbani, "An application of multivariate moment generating functions to the analysis of signal and noise propagation in radiographic screen-film systems," *Medical Physics*, **17**, pp. 65-71, Jan/Feb 1990.
7. M. Rabbani and R. L. VanMetter, "An analysis of signal and noise propagation for several imaging mechanisms," *J. Opt. Soc. of Am. A*, **6**, pp. 1156-1164, Aug. 1989.
8. M. I. Sezan, M. Rabbani, and P. W. Jones, "Progressive transmission of images using a prediction/residual encoding approach,"" *Optical Eng.*, **28**, pp. 556-564, May 1989.
9. M. Rabbani, "Bayesian filtering of Poisson noise using local statistics," *IEEE Trans. Acoust. Speech and Signal Proc.*, **36**, pp. 933-937, June 1988.
10. M. Rabbani, R. Shaw, and R. VanMetter, "Detective Quantum Efficiency of imaging systems with amplifying and scattering mechanisms,"" *J. Opt. Soc. Am. A*, **4** pp. 895-901, May 1987.
11. M. Rabbani, L. A. Ray, and J. R. Sullivan, "Adaptive predictive coding with applications to radiographs," *Medical Instrumentation*, **20**, pp. 182-191, Aug. 1986.
12. P. D. Burns, M. Rabbani, and L. A. Ray, "Analysis of image noise due to position errors in laser writers," *Appl. Opt.*, **25**, pp. 2158-2168, July 1986.
13. M. Rabbani and B.E.A. Saleh, "Bayesian filtering of speckled images," *Optics Communications*, **53**, pp. 147-152, Mar. 1985.
14. M. Rabbani and B.E.A. Saleh, "Bayesian restoration of partially coherent imagery in conventional and scanning microscopy," *Optica Acta*, **31**, pp. 63-79, 1984.
15. M. Rabbani and B.E.A. Saleh, "Restoration of bilinearly distorted images. 3: Constrained least-squares method," *Appl. Opt.*, **22**, pp. 3549-3554, Nov. 1983.
16. B.E.A. Saleh and M. Rabbani, "Restoration of bilinearly distorted images: II. Bayesian method," *J. Opt. Soc. Am.*, **73**, pp. 71-75, Jan. 1983.
17. B.E.A. Saleh and M. Rabbani, "Restoration of bilinearly distorted images: I. Finite impulse response linear digital filtering," *J. Opt. Soc. Am.*, **73,** pp. 66-70, Jan. 1983.
18. B.E.A. Saleh and M. Rabbani, "Simulation of partially coherent imagery in the space and frequency domains and by modal expansion," *Appl. Opt.*, **21**, pp. 2770-2777, Aug. 1982.
19. B.E.A. Saleh and M. Rabbani, "Linear filtering of speckled images," *Optics Communications*, **35**, pp. 327-331, Dec. 1980.

Conference Presentations/Proceedings Publications

1. S. Sah, R. Longman, A. Shringi, R. Loce, M. Rabbani, R. Ptucha, “Detection without Recognition for Redaction”, *CVPR Workshop The First International Workshop on The Bright and Dark Sides of Computer Vision: Challenges and Opportunities for Privacy and Security*, Honolulu, Hawaii, July 2017.
2. Minwoo Park, Jiebo Luo, Andrew Gallagher, Majid Rabbani, “Learning to Produce 3D Media from a Captured 2D Video,” *MM '11 Proceedings of the 19th ACM international conference on Multimedia*, pp 1557-1560, December 2011.
3. Majid Rabbani and Chris Honsinger (invited talk), “Robust Invisible Watermarking,” in Frontiers In Imaging, *International Congress of Imaging Science (ICIS’06)*, Rochester, NY, May 11, 2006.
4. Peter Burns, Scott Houchin, Ken Parulski, and Majid Rabbani, “Using JPEG2000 in Future Digital Cameras,” *Proceedings International Congress of Imaging Science 2002*, Tokyo, SPSTJ, pp. 371, May 2002.
5. Rajan L. Joshi, Majid Rabbani, and Margaret Lepley, “Comparison of multiple compression cycle performance for JPEG and JPEG2000,” *Proceedings of SPIE Applications of Digital Image Processing XXIII*, San Diego, **4115**, pp. 492-501, 31 July – 3 August, 2000.
6. K. Parulski and M. Rabbani, “The Continuing Evolution of Digital Cameras and Digital Photography Systems,” *IEEE International Symposium on Circuits and Systems - ISCAS 2000*, Geneva, Switzerland, pp. V-101 to V-104, May 2000.
7. Jeffery R. Price and Majid Rabbani (Speaker), "Dequantization bias for JPEG decompression," *IEEE International Conference on Information Technology: Coding and Computing (ITCC2000)*, Las Vegas, NV, pp. 30-35, Mar. 2000.
8. Chris Honsinger and Majid Rabbani (Speaker), “Data embedding using phase dispersion,” *International Conference on Information Technology: Coding and Computing (ITCC2000)*, Las Vegas, NV, Mar. 2000.
9. Majid Rabbani (invited presentation), "Overview of the state-of-the-art watermarking technologies," PMA 2000, Las Vegas, NV, February 2000
10. M. Rabbani, (Invited paper) “Review of the state-of-the-art watermarking technologies,” *1999 Western New York Image Processing Workshop*, Rochester, NY, Sept. 1999.
11. M. Rabbani and K. Parulski (Invited paper), “Digital cameras,” *IEEE Signal Processing Magazine*, **15**, pp. 28, Mar. 1998.
12. M. Rabbani (Invited paper), “Image compression: past, present, and future,” *Image Tech Conference*, Atlanta, GA, Mar. 1996.
13. P. Jones, S. Daly, R. Gaborski, and M. Rabbani, “Comparative study of wavelet and DCT decompositions with equivalent quantization and encoding strategies for medical images,” *Proceedings of SPIE Medical Imaging*, **2431**, pp. 571, Feb. 1995.
14. M. Rabbani and S. Dianat, “DPCM image compression using artificial neural networks,”' *Proceedings of SPIE, Image and Video Processing*, **1903**, Feb. 1993.
15. M. Rabbani and R. Shaw, “A parametric image simulation of a radiographic screen-film system,” *Proceedings of World Congress on Medical Physics and Biomedical Engineering*, Kyoto, Japan, July 1991.

16. M. Rabbani and P. Jones (Invited Seminar), "Digital image compression," *Seminar Lecture Notes, Society for Information Display,* pp. M-5.1 - M-5.42, Las Vegas, NV, May 1990.
17. R. L. VanMetter and M. Rabbani, "Analysis of the effects of depth dependence of x-ray interactions on the MTF and NPS of radiographic screens," *Proceedings of SPIE, Medical Imaging IV,* **1231**, pp. 271-275, Feb. 1990.
18. M. Rabbani (Invited Paper), "Restoration techniques for quantum limited images," *Proceedings of the Opt. Soc. of Am. Topical Meeting on Quantum-Limited Imaging and Information Processing*, **13**, pp. 30-33, June 1989.
19. M. Rabbani (Speaker) and R. L. VanMetter, "Some new applications of multivariate moment-generating functions to screen-film systems," *Proceedings of SPIE, Medical Imaging III*, **1090**, pp. 86-94, Feb. 1989.
20. M. Rabbani (Speaker) and S. Daly, "An optimized image data compression technique utilized in the KODAK SV9600 still video transceiver," *Proceedings of SPIE, Optical Sensors and Electronic Photography*, **1071,** pp. 246-256, Jan. 1989.
21. P. W. Melnychuck and M. Rabbani, "Survey of lossless image coding techniques," *Proceedings of SPIE, Digital Image Processing Applications,* **1075**, pp. 92-100, Jan. 1989.
22. M. Rabbani (Speaker) and R. L. VanMetter, "Analysis of signal and noise propagation for several imaging mechanisms," *Annual Meeting of the Opt. Soc. of Am.*, Santa Clara, CA, Nov. 1988.
23. M. Rabbani (Speaker) and R. Shaw, "The influence of grain threshold on quantum mottle in radiographic screen-film systems II," *Proceedings of SPIE, Medical Imaging II,* **914 A**, pp. 117-127, Feb. 1988.
24. M. Rabbani (Invited Paper), "Use of local statistics in adaptive filtering of Poisson noise," *Proceedings of The 26th IEEE Conference on Decision and Control,* pp. 1210-1212, Dec. 1987.
25. M. I. Sezan, C. L. Yeh, A. M. Tekalp, M. Rabbani and P. Jones, "Application of vector quantization to progressive encoding and transmission of images," *SPIE Cambridge Symp. on Optics in Med. and Visual Image Proc.*, Oct. 1987.
26. M. Rabbani (Speaker) and R. Shaw, "The influence of grain threshold on quantum mottle in radiographic screen-film systems," *Proceedings of SPIE, Medical Imaging,* **767**, pp. 226-235, Feb. 1987.
27. M. Rabbani (Speaker), R. Shaw, and R. VanMetter, "Detective quantum efficiency of imaging systems with amplifying and scattering mechanisms," *Annual meeting of the Opt. Soc. of Am.*, Seattle, WA, Oct. 1986.
28. M. Rabbani, "Restoration of images degraded by Poisson noise," *Proceedings of the International Electronic Image Week,* **2**, pp. 769-775, Nice, France, Apr. 1986.
29. R. Shaw and M. Rabbani, "Fundamental statistical limitations to x-ray quantum detection," *Proceedings of the Opt. Soc. of Am. Topical Meeting on Quantum Limited Imaging,* pp. 59-62, Apr. 1986.
30. R. Shaw and M. Rabbani, "A comparison of physical autocorrelation and digital Wiener spectrum techniques for screen-film evaluation," *Proceedings of SPIE, Medicine XIV*, **626**, pp. 76-81, Feb. 1986.

31. R. VanMetter, P. L. P. Dillon, K. E. Huff, and M. Rabbani, "Computer simulation of radiographic screen-film images," *Proceedings of SPIE, Medicine XIV*, **626**, pp. 82-99, Feb. 1986.
32. P. Jones and M. Rabbani, "Application of adaptive delta modulation to document and text image compression," *Proceedings of the SPSE 25th Fall Symposia Imaging*, pp. 139-142, Nov. 1985.
33. M. Rabbani (Speaker), L. Ray, and J. R. Sullivan, "Low-distortion compression of radiographs," *Proceedings of Image Science '85*, **1**, pp. 99-102, Helsinki, Finland, June 1985.
34. P. L. P. Dillon, J. F. Hamilton, M. Rabbani (Speaker), R. Shaw, and R. L. VanMetter, "`Principles governing the transfer of signal modulation and photon noise by amplifying and scattering mechanisms," *Proceedings of SPIE, Medicine XIII*, **535**, pp. 130-139, Feb. 1985.
35. M. Rabbani (Speaker) and B.E.A. Saleh, "Bayesian filtering of speckled images," *Annual Meeting of the Opt. Soc. of Am.*, San Diego, CA, Oct. 1984.
36. M. Rabbani (Speaker) and B.E.A. Saleh, "Bayesian restoration of partially coherent imagery in microscopy," *Annual Meeting of the Opt. Soc. of Am.*, New Orleans, LA, Oct. 1983.
37. B.E.A. Saleh and M. Rabbani (Speaker), "Bayesian restoration of partially coherent distorted images," *Annual Meeting of the Opt. Soc. of Am.*, Tucson, AZ, Oct. 1982.

Encoding

Encoding is the process of converting **data** from one form to another. While "encoding" can be used as a verb, it is often used as a noun, and refers to a specific type of encoded data. There are several types of encoding, including image encoding, audio and video encoding, and character encoding.

Media files are often encoded to save disk space. By encoding digital audio, video, and image files, they can be saved in a more efficient, compressed format. Encoded media files are typically similar in quality to their original uncompressed counterparts, but have much smaller file sizes. For example, a WAVE (**.WAV**) audio file that is converted to an MP3 (**.MP3**) file may be 1/10 the size of the original WAVE file. Similarly, an MPEG (**.MPG**) compressed video file may only require a fraction of the disk space as the original digital video (**.DV**) file.

Character encoding is another type of encoding that encodes **characters** as **bytes**. Since computers only recognize **binary** data, text must be represented in a binary form. This is accomplished by converting each character (which includes letters, numbers, symbols, and spaces) into a binary code. Common types of text encoding include **ASCII** and Unicode.

Whenever data is encoded, it can only be read by a program that supports the correct type of encoding. For audio and video files, this is often accomplished by a **codec**, which decodes the data in **real-time**. Most text editors support multiple types of text encoding, so it is rare to find a text file that will not open in a standard text editor. However, if a text editor does not support the encoding used in a text document, some or all of the characters may appear as strange symbols rather than the intended text.

Updated September 23, 2010 by **Per C.**

© 2023 Sharpened Productions

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

encoding

Dictionary Thesaurus

encode verb

en·code in-ˈkōd en-

encoded; encoding; encodes

Synonyms of *encode* >

transitive verb

1 a : to convert (something, such as a body of information) from one system of communication into another

especially **:** to convert (a message) into code

b : to convey symbolically

the capacity of poetry to *encode* ideology
– J. D. Niles

2 : to specify the genetic code for

noun

Credit cards are *encoded* with cardholder information.

a technology that *encodes* images

Recent Examples on the Web

An embedding module is used to *encode* both text tokens and other input modalities into vectors.

– Benj Edwards, *Ars Technica*, 1 Mar. 2023

Deleting neurotensin in those neurons in the thalamus made mice slower to learn about the sucrose reward and faster to *encode* the shock, as evidenced by behaviors such as running to a spout that delivers sugar water and freezing when predicting a shock.

– Ingrid Wickelgren, *Scientific American*, 12 Aug. 2022

The approach that Khesin and co have come up with is to *encode* the random lattice into the quantum properties of a unit of quantum money, perhaps as an atomic array.

– The Physics Arxiv Blog, *Discover Magazine*, 3 Aug. 2022

See More

Dictionary Thesaurus

First Known Use

circa 1919, in the meaning defined at sense 1a

Time Traveler

The first known use of *encode* was circa 1919

See more words from the same year

enclothe
encode
encodement

See More Nearby Entries >

Dictionary Thesaurus

"Encode." *Merriam-Webster.com Dictionary*, Merriam-Webster, https://www.merriam-webster.com/dictionary/encode. Accessed 28 Apr. 2023.

Copy Citation

Facebook Twitter

encode verb

en·code in-ˈkōd en-

: to change (as a body of information) from one system of communication into another

especially **:** to change (a message) into code

noun

Dictionary Thesaurus

encode transitive verb

en·code in-ˈkōd, en-

encoded; encoding

: to specify the genetic code for

each nucleotide triplet or codon on the mRNA chain *encodes* a specific amino acid
– J. E. Darnell, Jr.

Thesaurus: All synonyms and antonyms for *encode*
Nglish: Translation of *encode* for Spanish Speakers
Britannica English: Translation of *encode* for Arabic Speakers

Last Updated: 18 Apr 2023 - Updated example sentences

Love words? Need even more definitions?

Dictionary Thesaurus

Your email address

OTHER MERRIAM-WEBSTER DICTIONARIES

MERRIAM-WEBSTER'S UNABRIDGED DICTIONARY

SCRABBLE® WORD FINDER

MERRIAM-WEBSTER DICTIONARY API

NGLISH - SPANISH-ENGLISH TRANSLATION

BRITANNICA ENGLISH - ARABIC TRANSLATION

FOLLOW US

Browse the Dictionary: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0-9 BIO GEO

Home | Help | About Us | Shop | Advertising Info | Dictionary API | Contact Us | Join MWU | Videos | Word of the Year | Kid's Dictionary | Law Dictionary | Medical Dictionary | Privacy Policy | Terms of Use

Browse the Thesaurus | Browse the Medical Dictionary | Browse the Legal Dictionary | Browse the Kid's Dictionary

© 2023 Merriam-Webster, Incorporated

Information from your device can be used to personalize your ad experience.

Do not sell or share my personal information.

APPENDIX B

Techopedia

Don't miss an insight. Subscribe to Techopedia for free.

Subscribe

Advertisements

Dictionary

Software Development

Encoding



Margaret Rouse Editor

Last updated: 18 August, 2011

What Does Encoding Mean?

Encoding is the process of converting data into a format required for a number of information processing needs, including:

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Advertisements

- Program compiling and execution
- Data transmission, storage and compression/decompression
- Application data processing, such as file conversion

Encoding can have two meanings:

- In computer technology, encoding is the process of applying a specific code, such as letters, symbols and numbers, to data for conversion into an equivalent cipher.
- In electronics, encoding refers to analog to digital conversion.

Techopedia Explains Encoding

Encoding involves the use of a code to change original data into a form that can be used by an external process.

The type of code used for converting characters is known as American Standard Code for Information Interchange (ASCII), the most commonly used encoding scheme for files that contain text. ASCII contains printable and nonprintable characters that represent uppercase and lowercase letters, symbols, punctuation marks and numbers. A unique number is assigned to some characters.

The standard ASCII scheme has only zero to 127 character positions; 128 through 255 are undefined. The problem of undefined characters is solved by Unicode encoding, which assigns a number to every character used worldwide. Other types of codes include BinHex, Uuencode (UNIX to UNIX encoding) and Multipurpose Internet Mail Extensions (MIME).

Encoding is also used to reduce the size of audio and video files. Each audio and video file format has a corresponding coder-decoder (codec) program that is used to code it into the appropriate format and then decodes for playback.

Encoding should not be confused with encryption, which hides content. Both techniques are used extensively in the networking, software programming, wireless communication and storage fields.

Advertisements

Share This Article

Related Terms

Encoding

Decoding

Encryption

Decryption

American Standard for Information Interchange

Binary Digit

Data

Digital To Analog Conversion

Multipurpose Internet Mail Extensions

Binhex

Analog

Cipher

Related Reading

Coding: The Key to Getting More Women into Tech

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Pursuing a Job in DevOps: What Every Engineer Must Know in 2022

UAT Testing: How to Do it Right

UAT Testing: How to Do it Right

Tags

Cybersecurity

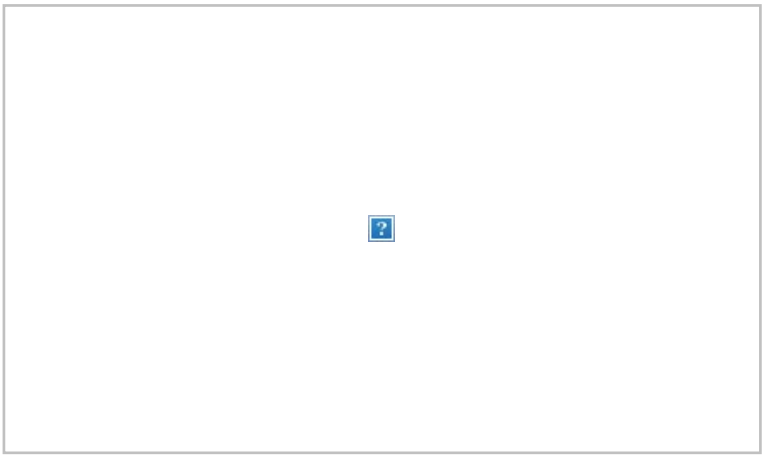
Encryption

Privacy and Compliance

Software Development

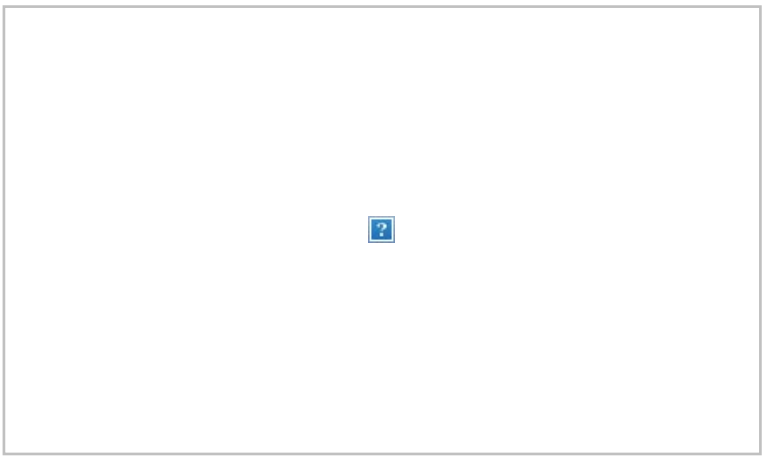
Storage

Trending Articles



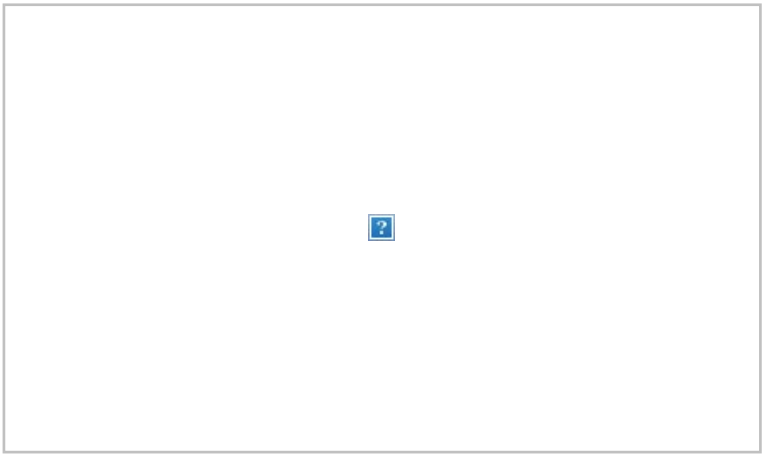
BLOCKCHAIN

An Introduction to Blockchain Technology



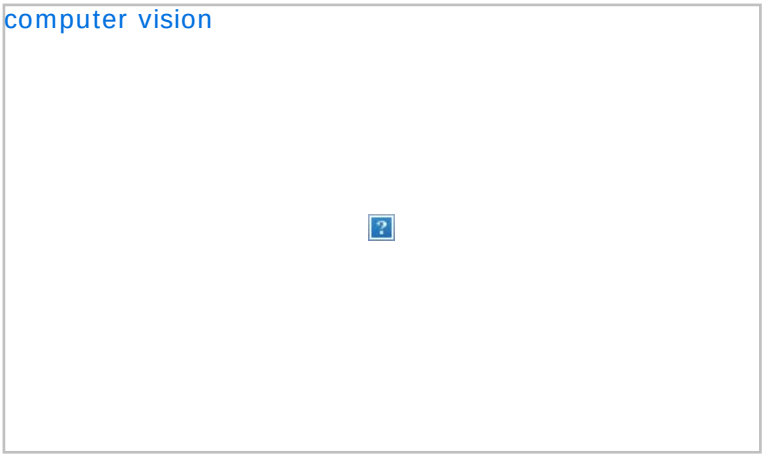
BLOCKCHAIN

How ChatGPT is Revolutionizing Smart Contract and Blockchain



BLOCKCHAIN

How Do Cryptocurrencies Work?



ARTIFICIAL INTELLIGENCE

The role of AI in computer vision

Tech moves fast! Stay ahead of the curve with Techopedia!

Join nearly 200,000 subscribers who receive actionable tech insights from Techopedia.

Email Address

Sign Up

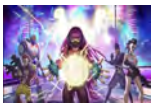
This site is protected by reCAPTCHA and the GooglePrivacy Policy andTerms of Service apply.

Latest Articles



9 Security Functions Your Organization Should Automate

By: **Linda Rosencrance** | Contributor



Meta Masters Guild Relaunches NFT Store with Upgradeable NFTs

By: **Alan Draper** | Editor-in-Chief



The Digital Revolution in Banking: Exploring the Future of Finance

By: **John Meah** | Cybersecurity Consultant



The Power of Edge Computing for Redefining Retail Customer Experience

By: **Assad Abbas** | Assistant Professor of Computer Science at COMSATS University



Exploring the Asilomar AI Principles: A Guide to Ensuring Safe and Beneficial AI Development

By: **Dr. Tehseen Zia** | Assistant Professor at Comsats University Islamabad

Advertisements

APPENDIX B

Advertisements

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Techopedia Logo

Techopedia™ is your go-to tech source for professional IT insight and inspiration.
We aim to be a site that isn't trying to be the first to break news stories,
but instead help you better understand technology and — we hope — make better decisions as a result.

Copyright © 2023 Techopedia Inc. - Terms of Use
-Privacy Policy - Editorial Review Policy

Cookies Settings

Dictionary

Topics

Tutorials

Webinars

Q&A

Downloads

Job Board

About Us

Contributors

Advertising Info

Write for Us

Contact Us

Sitemap

Follow Us

Subscribe To Our Newsletters

Email Address

Subscribe

- ☐ Term of the Day
- ☐ Best of Techopedia (weekly)
- ☐ News and Special Offers (occasional)
- ☐ Webinars (monthly)

By clicking sign up, you agree to receive emails from Techopedia and agree to our Terms of Use & Privacy Policy.
This site is protected by reCAPTCHA and the GooglePrivacy Policy andTerms of Service apply.

WIKIPEDIA
The Free Encyclopedia

JPEG

JPEG (/ˈdʒeɪpɛg/ *JAY-peg*, short for **Joint Photographic Experts Group**)[2] is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. JPEG typically achieves 10:1 compression with little perceptible loss in image quality.[3] Since its introduction in 1992, JPEG has been the most widely used image compression standard in the world,[4][5] and the most widely used digital image format, with several billion JPEG images produced every day as of 2015.[6]

The term "JPEG" is an acronym for the Joint Photographic Experts Group, which created the standard in 1992.[7] JPEG was largely responsible for the proliferation of digital images and digital photos across the Internet and later social media.[8]

JPEG compression is used in a number of image file formats. JPEG/Exif is the most common image format used by digital cameras and other photographic image capture devices; along with JPEG/JFIF, it is the most common format for storing and transmitting photographic images on the World Wide Web.[9] These format variations are often not distinguished and are simply called JPEG.

The MIME media type for JPEG is "image/jpeg," except in older Internet Explorer versions, which provide a MIME type of "image/pjpeg" when uploading JPEG images.[10] JPEG files usually have a filename extension of "jpg" or "jpeg." JPEG/JFIF supports a maximum image size of 65,535×65,535 pixels,[11] hence up to 4 gigapixels for an aspect ratio of 1:1. In 2000, the JPEG group introduced a format intended to be a successor, JPEG 2000, but it was unable to replace the original JPEG as the dominant image standard.[12]

JPEG



A photo of a European wildcat with the compression rate decreasing and hence quality increasing, from left to right

Filename extension	`.jpg, .jpeg, .jpe .jif, .jfif, .jfi`
Internet media type	`image/jpeg`
Type code	`JPEG`
Uniform Type Identifier (UTI)	public.jpeg
Magic number	`ff d8 ff`
Developed by	Joint Photographic Experts Group, IBM, Mitsubishi Electric, AT&T, Canon Inc.[1]
Initial release	September 18, 1992
Type of format	Lossy image compression format
Extended to	JPEG 2000

History

Background

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Standard	ISO/IEC 10918, ITU-T T.81, ITU-T T.83, ITU-T T.84, ITU-T T.86
Website	jpeg.org/jpeg/ (https://jpeg.org/jpeg/)

The original JPEG specification published in 1992 implements processes from various earlier research papers and patents cited by the CCITT (now ITU-T) and Joint Photographic Experts Group.[1]

The JPEG specification cites patents from several companies. The following patents provided the basis for its arithmetic coding algorithm.[1]

- IBM
  - U.S. Patent 4,652,856 (https://patents.google.com/patent/US4652856) – February 4, 1986 – Kottappuram M. A. Mohiuddin and Jorma J. Rissanen – Multiplication-free multi-alphabet arithmetic code
  - U.S. Patent 4,905,297 (https://patents.google.com/patent/US4905297) – February 27, 1990 – G. Langdon, J. L. Mitchell, W. B. Pennebaker, and Jorma J. Rissanen – Arithmetic coding encoder and decoder system
  - U.S. Patent 4,935,882 (https://patents.google.com/patent/US4935882) – June 19, 1990 – W. B. Pennebaker and J. L. Mitchell – Probability adaptation for arithmetic coders
- Mitsubishi Electric
  - JP H02202267 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=JPH02202267) (1021672 (https://patents.google.com/patent/JPH02202267A)) – January 21, 1989 – Toshihiro Kimura, Shigenori Kino, Fumitaka Ono, Masayuki Yoshida – Coding system
  - JP H03247123 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=JPH03247123) (2-46275 (https://patents.google.com/patent/JPH0834434B2/en)) – February 26, 1990 – Fumitaka Ono, Tomohiro Kimura, Masayuki Yoshida, and Shigenori Kino – Coding apparatus and coding method



Continuously varied JPEG compression (between Q=100 and Q=1) for an abdominal CT scan

The JPEG specification also cites three other patents from IBM. Other companies cited as patent holders include AT&T (two patents) and Canon Inc.[1] Absent from the list is U.S. Patent 4,698,672 (https://patents.google.com/patent/US4698672), filed by Compression Labs' Wen-Hsiung Chen and Daniel J. Klenke in October 1986. The patent describes a DCT-based image compression algorithm, and would later be a cause of controversy in 2002 (see *Patent controversy* below).[13] However, the JPEG specification did cite two earlier research papers by Wen-Hsiung Chen, published in 1977 and 1984.[1]

JPEG standard

"JPEG" stands for Joint Photographic Experts Group, the name of the committee that created the JPEG standard and also other still picture coding standards. The "Joint" stood for ISO TC97 WG8 and CCITT SGVIII. Founded in 1986, the group developed the JPEG standard during the late 1980s. The group published the JPEG standard in 1992.[4]

In 1987, ISO TC 97 became ISO/IEC JTC 1 and, in 1992, CCITT became ITU-T. Currently on the JTC1 side, JPEG is one of two sub-groups of ISO/IEC Joint Technical Committee 1, Subcommittee 29, Working Group 1 (ISO/IEC JTC 1/SC 29/WG 1) – titled as *Coding of still pictures*.[14][15][16] On the

ITU-T side, ITU-T SG16 is the respective body. The original JPEG Group was organized in 1986,[17] issuing the first JPEG standard in 1992, which was approved in September 1992 as **ITU-T Recommendation T.81**[18] and, in 1994, as **ISO/IEC 10918-1**.

The JPEG standard specifies the codec, which defines how an image is compressed into a stream of bytes and decompressed back into an image, but not the file format used to contain that stream.[19] The Exif and JFIF standards define the commonly used file formats for interchange of JPEG-compressed images.

JPEG standards are formally named as *Information technology – Digital compression and coding of continuous-tone still images*. ISO/IEC 10918 consists of the following parts:

APPENDIX C

Digital compression and coding of continuous-tone still images – Parts[15][17][20]

Part	ISO/IEC standard	ITU-T Rec.	First public release date	Latest amendment	Title	Description
Part 1	ISO/IEC 10918-1:1994 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18902)	T.81 (09/92) (http://www.itu.int/rec/T-REC-T.81)	Sep 18, 1992		Requirements and guidelines	
Part 2	ISO/IEC 10918-2:1995 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=20689)	T.83 (11/94) (http://www.itu.int/rec/T-REC-T.83)	Nov 11, 1994		Compliance testing	Rules and checks for software conformance (to Part 1).
Part 3	ISO/IEC 10918-3:1997 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=25037)	T.84 (07/96) (http://www.itu.int/rec/T-REC-T.84)	Jul 3, 1996	Apr 1, 1999	Extensions	Set of extensions to improve the Part 1, including the **Still Picture Interchange File Format** (SPIFF).[21]
Part 4	ISO/IEC 10918-4:1999 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=25431)	T.86 (06/98) (http://www.itu.int/rec/T-REC-T.86)	Jun 18, 1998	Jun 29, 2012	Registration of JPEG profiles, SPIFF profiles, SPIFF tags, SPIFF colour spaces, APPn markers, SPIFF compression types and Registration Authorities (REGAUT)	methods for registering some of the parameters used to extend JPEG
Part 5	ISO/IEC 10918-5:2013 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=54989)	T.871 (05/11) (http://www.itu.int/rec/T-REC-T.871)	May 14, 2011		JPEG File Interchange Format (JFIF)	A popular format which has been the de facto file format for images encoded by the JPEG standard. In 2009, the JPEG Committee formally established an Ad Hoc Group to standardize JFIF as JPEG Part 5.[22]
Part 6	ISO/IEC 10918-6:2013	T.872 (06/12)	Jun 2012		Application to printing systems	Specifies a subset of

Part	ISO/IEC standard	ITU-T Rec.	First public release date	Latest amendment	Title	Description
	(http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=59634)	(http://www.itu.int/rec/T-REC-T.872)				features and application tools for the interchange of images encoded according to the ISO/IEC 10918-1 for printing.
Part 7	ISO/IEC 10918-7:2021 (https://www.iso.org/standard/80896.html)	T.873 (06/21) (http://www.itu.int/rec/T-REC-T.873)	May 2019	June 2021	Reference Software	Provides reference implementations of the JPEG core coding system

Ecma International TR/98 specifies the JPEG File Interchange Format (JFIF); the first edition was published in June 2009.[23]

Patent controversy

In 2002, Forgent Networks asserted that it owned and would enforce patent rights on the JPEG technology, arising from a patent that had been filed on October 27, 1986, and granted on October 6, 1987: U.S. Patent 4,698,672 (https://patents.google.com/patent/US4698672) by Compression Labs' Wen-Hsiung Chen and Daniel J. Klenke.[13][24] While Forgent did not own Compression Labs at the time, Chen later sold Compression Labs to Forgent, before Chen went on to work for Cisco. This led to Forgent acquiring ownership over the patent.[13] Forgent's 2002 announcement created a furor reminiscent of Unisys' attempts to assert its rights over the GIF image compression standard.

The JPEG committee investigated the patent claims in 2002 and were of the opinion that they were invalidated by prior art,[25] a view shared by various experts.[13][26]

Between 2002 and 2004, Forgent was able to obtain about US$105 million by licensing their patent to some 30 companies. In April 2004, Forgent sued 31 other companies to enforce further license payments. In July of the same year, a consortium of 21 large computer companies filed a countersuit, with the goal of invalidating the patent. In addition, Microsoft launched a separate lawsuit against Forgent in April 2005.[27] In February 2006, the United States Patent and Trademark Office agreed to re-examine Forgent's JPEG patent at the request of the Public Patent Foundation.[28] On May 26, 2006, the USPTO found the patent invalid based on prior art. The USPTO also found that Forgent knew about the prior art, yet it intentionally avoided telling the Patent Office. This makes any appeal to reinstate the patent highly unlikely to succeed.[29]

Forgent also possesses a similar patent granted by the European Patent Office in 1994, though it is unclear how enforceable it is.[30]

As of October 27, 2006, the U.S. patent's 20-year term appears to have expired, and in November 2006, Forgent agreed to abandon enforcement of patent claims against use of the JPEG standard.[31]

The JPEG committee has as one of its explicit goals that their standards (in particular their baseline methods) be implementable without payment of license fees, and they have secured appropriate license rights for their JPEG 2000 standard from over 20 large organizations.

Beginning in August 2007, another company, Global Patent Holdings, LLC claimed that its patent (U.S. Patent 5,253,341 (https://patents.google.com/patent/US5253341)) issued in 1993, is infringed by the downloading of JPEG images on either a website or through e-mail. If not invalidated, this patent could apply to any website that displays JPEG images. The patent was under reexamination by the U.S. Patent and Trademark Office from 2000 to 2007; in July 2007, the Patent Office revoked all of the original claims of the patent but found that an additional claim proposed by Global Patent Holdings (claim 17) was valid.[32] Global Patent Holdings then filed a number of lawsuits based on claim 17 of its patent.

In its first two lawsuits following the reexamination, both filed in Chicago, Illinois, Global Patent Holdings sued the Green Bay Packers, CDW, Motorola, Apple, Orbitz, Officemax, Caterpillar, Kraft and Peapod as defendants. A third lawsuit was filed on December 5, 2007, in South Florida against ADT Security Services, AutoNation, Florida Crystals Corp., HearUSA, MovieTickets.com, Ocwen Financial Corp. and Tire Kingdom, and a fourth lawsuit on January 8, 2008, in South Florida against the Boca Raton Resort & Club. A fifth lawsuit was filed against Global Patent Holdings in Nevada. That lawsuit was filed by Zappos.com, Inc., which was allegedly threatened by Global Patent Holdings, and sought a judicial declaration that the '341 patent is invalid and not infringed.

Global Patent Holdings had also used the '341 patent to sue or threaten outspoken critics of broad software patents, including Gregory Aharonian[33] and the anonymous operator of a website blog known as the "Patent Troll Tracker."[34] On December 21, 2007, patent lawyer Vernon Francissen of Chicago asked the U.S. Patent and Trademark Office to reexamine the sole remaining claim of the '341 patent on the basis of new prior art.[35]

On March 5, 2008, the U.S. Patent and Trademark Office agreed to reexamine the '341 patent, finding that the new prior art raised substantial new questions regarding the patent's validity.[36] In light of the reexamination, the accused infringers in four of the five pending lawsuits have filed motions to suspend (stay) their cases until completion of the U.S. Patent and Trademark Office's review of the '341 patent. On April 23, 2008, a judge presiding over the two lawsuits in Chicago, Illinois granted the motions in those cases.[37] On July 22, 2008, the Patent Office issued the first "Office Action" of the second reexamination, finding the claim invalid based on nineteen separate grounds.[38] On Nov. 24, 2009, a Reexamination Certificate was issued cancelling all claims.

Beginning in 2011 and continuing as of early 2013, an entity known as Princeton Digital Image Corporation,[39] based in Eastern Texas, began suing large numbers of companies for alleged infringement of U.S. Patent 4,813,056 (https://patents.google.com/patent/US4813056). Princeton claims that the JPEG image compression standard infringes the '056 patent and has sued large numbers of websites, retailers, camera and device manufacturers and resellers. The patent was originally owned and assigned to General Electric. The patent expired in December 2007, but Princeton has sued large numbers of companies for "past infringement" of this patent. (Under U.S. patent laws, a patent owner can sue for "past infringement" up to six years before the filing of a lawsuit, so Princeton could theoretically have continued suing companies until December 2013.) As of

March 2013, Princeton had suits pending in New York and Delaware against more than 55 companies. General Electric's involvement in the suit is unknown, although court records indicate that it assigned the patent to Princeton in 2009 and retains certain rights in the patent.[40]

Typical use

The JPEG compression algorithm operates at its best on photographs and paintings of realistic scenes with smooth variations of tone and color. For web usage, where reducing the amount of data used for an image is important for responsive presentation, JPEG's compression benefits make JPEG popular. JPEG/Exif is also the most common format saved by digital cameras.

However, JPEG is not well suited for line drawings and other textual or iconic graphics, where the sharp contrasts between adjacent pixels can cause noticeable artifacts. Such images are better saved in a lossless graphics format such as TIFF, GIF, PNG, or a raw image format. The JPEG standard includes a lossless coding mode, but that mode is not supported in most products.

As the typical use of JPEG is a lossy compression method, which reduces the image fidelity, it is inappropriate for exact reproduction of imaging data (such as some scientific and medical imaging applications and certain technical image processing work).

JPEG is also not well suited to files that will undergo multiple edits, as some image quality is lost each time the image is recompressed, particularly if the image is cropped or shifted, or if encoding parameters are changed – see digital generation loss for details. To prevent image information loss during sequential and repetitive editing, the first edit can be saved in a lossless format, subsequently edited in that format, then finally published as JPEG for distribution.

JPEG compression

JPEG uses a lossy form of compression based on the discrete cosine transform (DCT). This mathematical operation converts each frame/field of the video source from the spatial (2D) domain into the frequency domain (a.k.a. transform domain). A perceptual model based loosely on the human psychovisual system discards high-frequency information, i.e. sharp transitions in intensity, and color hue. In the transform domain, the process of reducing information is called quantization. In simpler terms, quantization is a method for optimally reducing a large number scale (with different occurrences of each number) into a smaller one, and the transform-domain is a convenient representation of the image because the high-frequency coefficients, which contribute less to the overall picture than other coefficients, are characteristically small-values with high compressibility. The quantized coefficients are then sequenced and losslessly packed into the output bitstream. Nearly all software implementations of JPEG permit user control over the compression ratio (as well as other optional parameters), allowing the user to trade off picture-quality for smaller file size. In embedded applications (such as miniDV, which uses a similar DCT-compression scheme), the parameters are pre-selected and fixed for the application.

The compression method is usually lossy, meaning that some original image information is lost and cannot be restored, possibly affecting image quality. There is an optional lossless mode defined in the JPEG standard. However, this mode is not widely supported in products.

There is also an interlaced *progressive* JPEG format, in which data is compressed in multiple passes of progressively higher detail. This is ideal for large images that will be displayed while downloading over a slow connection, allowing a reasonable preview after receiving only a portion of the data.

However, support for progressive JPEGs is not universal. When progressive JPEGs are received by programs that do not support them (such as versions of Internet Explorer before Windows 7)[41] the software displays the image only after it has been completely downloaded.

There are also many medical imaging, traffic and camera applications that create and process 12-bit JPEG images both grayscale and color. 12-bit JPEG format is included in an Extended part of the JPEG specification. The libjpeg codec supports 12-bit JPEG and there even exists a high-performance version.[42]

Lossless editing

Several alterations to a JPEG image can be performed losslessly (that is, without recompression and the associated quality loss) as long as the image size is a multiple of 1 MCU block (Minimum Coded Unit) (usually 16 pixels in both directions, for 4:2:0 chroma subsampling). Utilities that implement this include:

- jpegtran and its GUI, Jpegcrop.
- IrfanView using "JPG Lossless Crop (PlugIn)" and "JPG Lossless Rotation (PlugIn)", which require installing the JPG_TRANSFORM plugin.
- FastStone Image Viewer using "Lossless Crop to File" and "JPEG Lossless Rotate".
- XnViewMP using "JPEG lossless transformations".
- ACDSee supports lossless rotation (but not lossless cropping) with its "Force lossless JPEG operations" option.

Blocks can be rotated in 90-degree increments, flipped in the horizontal, vertical and diagonal axes and moved about in the image. Not all blocks from the original image need to be used in the modified one.

The top and left edge of a JPEG image must lie on an 8 × 8 pixel block boundary, but the bottom and right edge need not do so. This limits the possible **lossless crop** operations, and also prevents flips and rotations of an image whose bottom or right edge does not lie on a block boundary for all channels (because the edge would end up on top or left, where – as aforementioned – a block boundary is obligatory).

Rotations where the image is not a multiple of 8 or 16, which value depends upon the chroma subsampling, are not lossless. Rotating such an image causes the blocks to be recomputed which results in loss of quality.[43]

When using lossless cropping, if the bottom or right side of the crop region is not on a block boundary, then the rest of the data from the partially used blocks will still be present in the cropped file and can be recovered. It is also possible to transform between baseline and progressive formats without any loss of quality, since the only difference is the order in which the coefficients are placed in the file.

Furthermore, several JPEG images can be losslessly joined, as long as they were saved with the same quality and the edges coincide with block boundaries.

JPEG files

The file format known as "JPEG Interchange Format" (JIF) is specified in Annex B of the standard. However, this "pure" file format is rarely used, primarily because of the difficulty of programming encoders and decoders that fully implement all aspects of the standard and because of certain shortcomings of the standard:

- Color space definition
- Component sub-sampling registration
- Pixel aspect ratio definition.

Several additional standards have evolved to address these issues. The first of these, released in 1992, was the JPEG File Interchange Format (or JFIF), followed in recent years by Exchangeable image file format (Exif) and ICC color profiles. Both of these formats use the actual JIF byte layout, consisting of different *markers*, but in addition, employ one of the JIF standard's extension points, namely the *application markers*: JFIF uses APP0, while Exif uses APP1. Within these segments of the file that were left for future use in the JIF standard and are not read by it, these standards add specific metadata.

Thus, in some ways, JFIF is a cut-down version of the JIF standard in that it specifies certain constraints (such as not allowing all the different encoding modes), while in other ways, it is an extension of JIF due to the added metadata. The documentation for the original JFIF standard states:[44]

> JPEG File Interchange Format is a minimal file format which enables JPEG bitstreams to be exchanged between a wide variety of platforms and applications. This minimal format does not include any of the advanced features found in the TIFF JPEG specification or any application specific file format. Nor should it, for the only purpose of this simplified format is to allow the exchange of JPEG compressed images.

Image files that employ JPEG compression are commonly called "JPEG files", and are stored in variants of the JIF image format. Most image capture devices (such as digital cameras) that output JPEG are actually creating files in the Exif format, the format that the camera industry has standardized on for metadata interchange. On the other hand, since the Exif standard does not allow color profiles, most image editing software stores JPEG in JFIF format, and also includes the APP1 segment from the Exif file to include the metadata in an almost-compliant way; the JFIF standard is interpreted somewhat flexibly.[45]

Strictly speaking, the JFIF and Exif standards are incompatible, because each specifies that its marker segment (APP0 or APP1, respectively) appear first. In practice, most JPEG files contain a JFIF marker segment that precedes the Exif header. This allows older readers to correctly handle the older format JFIF segment, while newer readers also decode the following Exif segment, being less strict about requiring it to appear first.

JPEG filename extensions

The most common filename extensions for files employing JPEG compression are **.jpg** and **.jpeg**, though .jpe, .jfif and .jif are also used. It is also possible for JPEG data to be embedded in other file types – TIFF encoded files often embed a JPEG image as a thumbnail of the main image; and MP3 files can contain a JPEG of cover art in the ID3v2 tag.

Color profile

Many JPEG files embed an ICC color profile (color space). Commonly used color profiles include sRGB and Adobe RGB. Because these color spaces use a non-linear transformation, the dynamic range of an 8-bit JPEG file is about 11 stops; see gamma curve.

If the image doesn't specify color profile information (*untagged*), the color space is assumed to be sRGB for the purposes of display on webpages.[46][47]

Syntax and structure

A JPEG image consists of a sequence of *segments*, each beginning with a *marker*, each of which begins with a 0xFF byte, followed by a byte indicating what kind of marker it is. Some markers consist of just those two bytes; others are followed by two bytes (high then low), indicating the length of marker-specific payload data that follows. (The length includes the two bytes for the length, but not the two bytes for the marker.) Some markers are followed by entropy-coded data; the length of such a marker does not include the entropy-coded data. Note that consecutive 0xFF bytes are used as fill bytes for padding purposes, although this fill byte padding should only ever take place for markers immediately following entropy-coded scan data (see JPEG specification section B.1.1.2 and E.1.2 for details; specifically "In all cases where markers are appended after the compressed data, optional 0xFF fill bytes may precede the marker").

Within the entropy-coded data, after any 0xFF byte, a 0x00 byte is inserted by the encoder before the next byte, so that there does not appear to be a marker where none is intended, preventing framing errors. Decoders must skip this 0x00 byte. This technique, called byte stuffing (see JPEG specification section F.1.2.3), is only applied to the entropy-coded data, not to marker payload data. Note however that entropy-coded data has a few markers of its own; specifically the Reset markers (0xD0 through 0xD7), which are used to isolate independent chunks of entropy-coded data to allow parallel decoding, and encoders are free to insert these Reset markers at regular intervals (although not all encoders do this).

Common JPEG markers[48]

Short name	Bytes	Payload	Name	Comments
SOI	0xFF, 0xD8	*none*	Start Of Image	
SOF0	0xFF, 0xC0	*variable size*	Start Of Frame (baseline DCT)	Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
SOF2	0xFF, 0xC2	*variable size*	Start Of Frame (progressive DCT)	Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
DHT	0xFF, 0xC4	*variable size*	Define Huffman Table(s)	Specifies one or more Huffman tables.
DQT	0xFF, 0xDB	*variable size*	Define Quantization Table(s)	Specifies one or more quantization tables.
DRI	0xFF, 0xDD	4 bytes	Define Restart Interval	Specifies the interval between RST*n* markers, in Minimum Coded Units (MCUs). This marker is followed by two bytes indicating the fixed size so it can be treated like any other variable size segment.
SOS	0xFF, 0xDA	*variable size*	Start Of Scan	Begins a top-to-bottom scan of the image. In baseline DCT JPEG images, there is generally a single scan. Progressive DCT JPEG images usually contain multiple scans. This marker specifies which slice of data it will contain, and is immediately followed by entropy-coded data.
RST*n*	0xFF, 0xD*n* (*n*=0..7)	*none*	Restart	Inserted every *r* macroblocks, where *r* is the restart interval set by a DRI marker. Not used if there was no DRI marker. The low three bits of the marker code cycle in value from 0 to 7.
APP*n*	0xFF, 0xE*n*	*variable size*	Application-specific	For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on TIFF.
COM	0xFF, 0xFE	*variable size*	Comment	Contains a text comment.
EOI	0xFF, 0xD9	*none*	End Of Image	

There are other *Start Of Frame* markers that introduce other kinds of JPEG encodings.

Since several vendors might use the same APP*n* marker type, application-specific markers often begin with a standard or vendor name (e.g., "Exif" or "Adobe") or some other identifying string.

At a restart marker, block-to-block predictor variables are reset, and the bitstream is synchronized to a byte boundary. Restart markers provide means for recovery after bitstream error, such as transmission over an unreliable network or file corruption. Since the runs of macroblocks between restart markers may be independently decoded, these runs may be decoded in parallel.

JPEG codec example

Although a JPEG file can be encoded in various ways, most commonly it is done with JFIF encoding. The encoding process consists of several steps:

1. The representation of the colors in the image is converted from RGB to $Y'C_BC_R$, consisting of one luma component (Y'), representing brightness, and two chroma components, (C_B and C_R), representing color. This step is sometimes skipped.
2. The resolution of the chroma data is reduced, usually by a factor of 2 or 3. This reflects the fact that the eye is less sensitive to fine color details than to fine brightness details.
3. The image is split into blocks of 8×8 pixels, and for each block, each of the Y, C_B, and C_R data undergoes the discrete cosine transform (DCT). A DCT is similar to a Fourier transform in the sense that it produces a kind of spatial frequency spectrum.
4. The amplitudes of the frequency components are quantized. Human vision is much more sensitive to small variations in color or brightness over large areas than to the strength of high-frequency brightness variations. Therefore, the magnitudes of the high-frequency components are stored with a lower accuracy than the low-frequency components. The quality setting of the encoder (for example 50 or 95 on a scale of 0–100 in the Independent JPEG Group's library[49]) affects to what extent the resolution of each frequency component is reduced. If an excessively low quality setting is used, the high-frequency components are discarded altogether.
5. The resulting data for all 8×8 blocks is further compressed with a lossless algorithm, a variant of Huffman encoding.

The decoding process reverses these steps, except the *quantization* because it is irreversible. In the remainder of this section, the encoding and decoding processes are described in more detail.

Encoding

Many of the options in the JPEG standard are not commonly used, and as mentioned above, most image software uses the simpler JFIF format when creating a JPEG file, which among other things specifies the encoding method. Here is a brief description of one of the more common methods of encoding when applied to an input that has 24 bits per pixel (eight each of red, green, and blue). This particular option is a lossy data compression method.

Color space transformation

First, the image should be converted from RGB (by default sRGB,[46][47] but other color spaces are possible) into a different color space called $Y'C_BC_R$ (or, informally, YCbCr). It has three components Y', C_B and C_R: the Y' component represents the brightness of a pixel, and the C_B and C_R components represent the chrominance (split into blue and red components). This is basically the same color space as used by digital color television as well as digital video including video DVDs. The $Y'C_BC_R$ color space conversion allows greater compression without a significant effect on perceptual image quality (or greater perceptual image quality for the same compression). The compression is more efficient because the brightness information, which is more important to the eventual perceptual quality of the image, is confined to a single channel. This more closely corresponds to the perception of color in the human visual system. The color transformation also improves compression by statistical decorrelation.

A particular conversion to $Y'C_BC_R$ is specified in the JFIF standard, and should be performed for the resulting JPEG file to have maximum compatibility. However, some JPEG implementations in "highest quality" mode do not apply this step and instead keep the color information in the RGB color

model,[50] where the image is stored in separate channels for red, green and blue brightness components. This results in less efficient compression, and would not likely be used when file size is especially important.

Downsampling

Due to the densities of color- and brightness-sensitive receptors in the human eye, humans can see considerably more fine detail in the brightness of an image (the Y' component) than in the hue and color saturation of an image (the Cb and Cr components). Using this knowledge, encoders can be designed to compress images more efficiently.

The transformation into the $Y'C_BC_R$ color model enables the next usual step, which is to reduce the spatial resolution of the Cb and Cr components (called "downsampling" or "chroma subsampling"). The ratios at which the downsampling is ordinarily done for JPEG images are 4:4:4 (no downsampling), 4:2:2 (reduction by a factor of 2 in the horizontal direction), or (most commonly) 4:2:0 (reduction by a factor of 2 in both the horizontal and vertical directions). For the rest of the compression process, Y', Cb and Cr are processed separately and in a very similar manner.

Block splitting

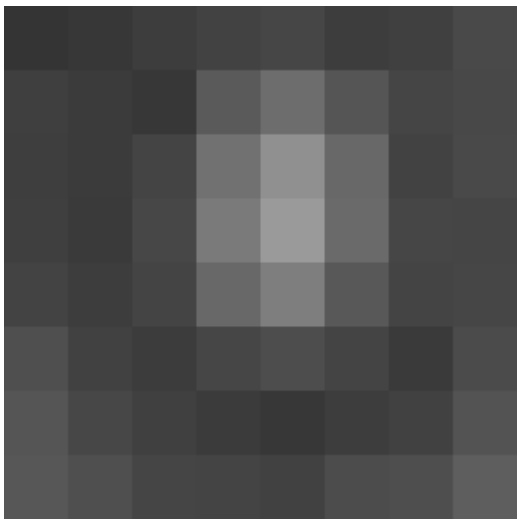
After subsampling, each channel must be split into 8×8 blocks. Depending on chroma subsampling, this yields Minimum Coded Unit (MCU) blocks of size 8×8 (4:4:4 – no subsampling), 16×8 (4:2:2), or most commonly 16×16 (4:2:0). In video compression MCUs are called macroblocks.

If the data for a channel does not represent an integer number of blocks then the encoder must fill the remaining area of the incomplete blocks with some form of dummy data. Filling the edges with a fixed color (for example, black) can create ringing artifacts along the visible part of the border; repeating the edge pixels is a common technique that reduces (but does not necessarily eliminate) such artifacts, and more sophisticated border filling techniques can also be applied.

Discrete cosine transform

Next, each 8×8 block of each component (Y, Cb, Cr) is converted to a frequency-domain representation, using a normalized, two-dimensional type-II discrete cosine transform (DCT), see Citation 1 in discrete cosine transform. The DCT is sometimes referred to as "type-II DCT" in the context of a family of transforms as in discrete cosine transform, and the corresponding inverse (IDCT) is denoted as "type-III DCT".

As an example, one such 8×8 8-bit subimage might be:



The 8×8 sub-image shown in 8-bit grayscale

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}.$$

Before computing the DCT of the 8×8 block, its values are shifted from a positive range to one centered on zero. For an 8-bit image, each entry in the original block falls in the range $[0, 255]$. The midpoint of the range (in this case, the value 128) is subtracted from each entry to produce a data range that is centered on zero, so that the modified range is $[-128, 127]$. This step reduces the dynamic range requirements in the DCT processing stage that follows.

This step results in the following values:

$$g = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \downarrow y. \end{matrix}$$

$$\begin{matrix} & & & & x & & & \\ & & & & \longrightarrow & & & \end{matrix}$$

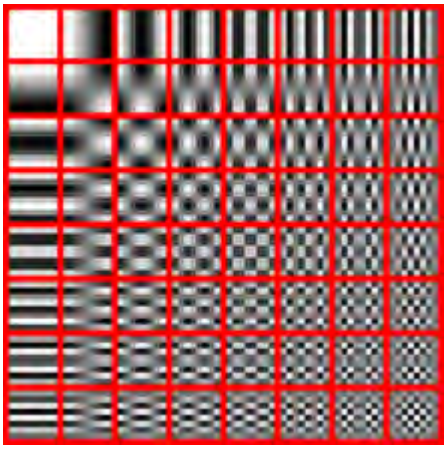
The next step is to take the two-dimensional DCT, which is given by:

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

where

- u is the horizontal spatial frequency, for the integers $0 \leq u < 8$.
- v is the vertical spatial frequency, for the integers $0 \leq v < 8$.
- $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } u = 0 \\ 1, & \text{otherwise} \end{cases}$ is a normalizing scale factor to make the transformation orthonormal
- $g_{x,y}$ is the pixel value at coordinates (x, y)
- $G_{u,v}$ is the DCT coefficient at coordinates (u, v).

If we perform this transformation on our matrix above, we get the following (rounded to the nearest two digits beyond the decimal point):



The DCT transforms an 8×8 block of input values to a linear combination of these 64 patterns. The patterns are referred to as the two-dimensional DCT *basis functions*, and the output values are referred to as *transform coefficients*. The horizontal index is $\boldsymbol{u}$ and the vertical index is $\boldsymbol{v}$.

$$
\begin{matrix}
 & & & & \boldsymbol{u} & & & \\
 & & & & \longrightarrow & & & \\
\end{matrix}
$$

$$
G = \begin{bmatrix}
-415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\
4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\
-46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\
-48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\
12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\
-7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\
-1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\
-0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68
\end{bmatrix}
\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \downarrow \boldsymbol{v}. \end{matrix}
$$

Note the top-left corner entry with the rather large magnitude. This is the DC coefficient (also called the constant component), which defines the basic hue for the entire block. The remaining 63 coefficients are the AC coefficients (also called the alternating components).[51] The advantage of the DCT is its tendency to aggregate most of the signal in one corner of the result, as may be seen above. The quantization step to follow accentuates this effect while simultaneously reducing the overall size of the DCT coefficients, resulting in a signal that is easy to compress efficiently in the entropy stage.

The DCT temporarily increases the bit-depth of the data, since the DCT coefficients of an 8-bit/component image take up to 11 or more bits (depending on fidelity of the DCT calculation) to store. This may force the codec to temporarily use 16-bit numbers to hold these coefficients, doubling the size of the image representation at this point; these values are typically reduced back to 8-bit

values by the quantization step. The temporary increase in size at this stage is not a performance concern for most JPEG implementations, since typically only a very small part of the image is stored in full DCT form at any given time during the image encoding or decoding process.

Quantization

The human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. This allows one to greatly reduce the amount of information in the high frequency components. This is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer. This rounding operation is the only lossy operation in the whole process (other than chroma subsampling) if the DCT computation is performed with sufficiently high precision. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers, which take many fewer bits to represent.

The elements in the quantization matrix control the compression ratio, with larger values producing greater compression. A typical quantization matrix (for a quality of 50% as specified in the original JPEG Standard), is as follows:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

The quantized DCT coefficients are computed with

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0, 1, 2, \ldots, 7; k = 0, 1, 2, \ldots, 7$$

where G is the unquantized DCT coefficients; Q is the quantization matrix above; and B is the quantized DCT coefficients.

Using this quantization matrix with the DCT coefficient matrix from above results in:

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$



Left: a final image is built up from a series of basis functions. Right: each of the DCT basis functions that comprise the image, and the corresponding weighting coefficient. Middle: the basis function, after multiplication by the coefficient: this component is added to the final image. For clarity, the 8×8 macroblock in this example is magnified by 10x using bilinear interpolation.

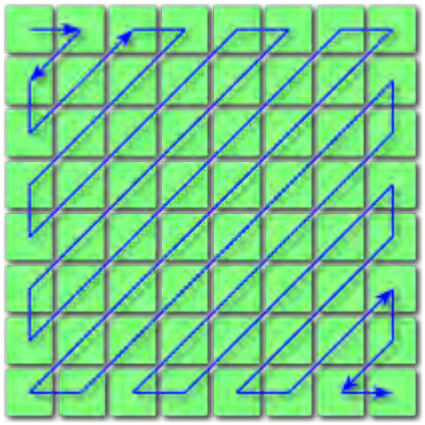
For example, using −415 (the DC coefficient) and rounding to the nearest integer

$$\text{round}\left(\frac{-415.37}{16}\right) = \text{round}\left(-25.96\right) = -26.$$

Notice that most of the higher-frequency elements of the sub-block (i.e., those with an x or y spatial frequency greater than 4) are quantized into zero values.

Entropy coding

Entropy coding is a special form of lossless data compression. It involves arranging the image components in a "zigzag" order employing run-length encoding (RLE) algorithm that groups similar frequencies together, inserting length coding zeros, and then using Huffman coding on what is left.



Zigzag ordering of JPEG image components

The JPEG standard also allows, but does not require, decoders to support the use of arithmetic coding, which is mathematically superior to Huffman coding. However, this feature has rarely been used, as it was historically covered by patents requiring royalty-bearing licenses, and because it is slower to encode and decode compared to Huffman coding. Arithmetic coding typically makes files about 5–7% smaller.

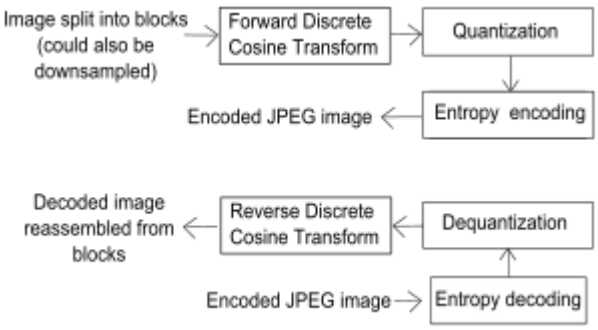
The previous quantized DC coefficient is used to predict the current quantized DC coefficient. The difference between the two is encoded rather than the actual value. The encoding of the 63 quantized AC coefficients does not use such prediction differencing.

The zigzag sequence for the above quantized coefficients are shown below. (The format shown is just for ease of understanding/viewing.)

```
−26
 −3   0
 −3  −2  −6
  2  −4   1  −3
  1   1   5   1   2
 −1   1  −1   2   0   0
```

```
0   0   0  −1  −1   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0
0   0   0   0   0   0
0   0   0   0   0
0   0   0   0
0   0   0
0   0
0
```

If the *i*-th block is represented by B_i and positions within each block are represented by (p, q) where $p = 0, 1, \ldots, 7$ and $q = 0, 1, \ldots, 7$, then any coefficient in the DCT image can be represented as $B_i(p, q)$. Thus, in the above scheme, the order of encoding pixels (for the *i*-th block) is $B_i(0, 0)$, $B_i(0, 1)$, $B_i(1, 0)$, $B_i(2, 0)$, $B_i(1, 1)$, $B_i(0, 2)$, $B_i(0, 3)$, $B_i(1, 2)$ and so on.



Baseline sequential JPEG encoding and decoding processes

This encoding mode is called baseline *sequential* encoding. Baseline JPEG also supports *progressive* encoding. While sequential encoding encodes coefficients of a single block at a time (in a zigzag manner), progressive encoding encodes similar-positioned batch of coefficients of all blocks in one go (called a *scan*), followed by the next batch of coefficients of all blocks, and so on. For example, if the image is divided into N 8×8 blocks $B_0, B_1, B_2, \ldots, B_{n-1}$, then a 3-scan progressive encoding encodes DC component, $B_i(0, 0)$ for all blocks, i.e., for all $i = 0, 1, 2, \ldots, N - 1$, in first scan. This is followed by the second scan which encoding a few more components (assuming four more components, they are $B_i(0, 1)$ to $B_i(1, 1)$, still in a zigzag manner) coefficients of all blocks (so the sequence is: $B_0(0, 1), B_0(1, 0), B_0(2, 0), B_0(1, 1), B_1(0, 1), B_1(1, 0), \ldots, B_N(2, 0), B_N(1, 1)$), followed by all the remained coefficients of all blocks in the last scan.

Once all similar-positioned coefficients have been encoded, the next position to be encoded is the one occurring next in the zigzag traversal as indicated in the figure above. It has been found that *baseline progressive* JPEG encoding usually gives better compression as compared to *baseline sequential* JPEG due to the ability to use different Huffman tables (see below) tailored for different frequencies on each "scan" or "pass" (which includes similar-positioned coefficients), though the difference is not too large.

In the rest of the article, it is assumed that the coefficient pattern generated is due to sequential mode.

In order to encode the above generated coefficient pattern, JPEG uses Huffman encoding. The JPEG standard provides general-purpose Huffman tables; encoders may also choose to generate Huffman tables optimized for the actual frequency distributions in images being encoded.

The process of encoding the zig-zag quantized data begins with a run-length encoding explained below, where:

- x is the non-zero, quantized AC coefficient.
- *RUNLENGTH* is the number of zeroes that came before this non-zero AC coefficient.
- *SIZE* is the number of bits required to represent x.
- *AMPLITUDE* is the bit-representation of x.

The run-length encoding works by examining each non-zero AC coefficient x and determining how many zeroes came before the previous AC coefficient. With this information, two symbols are created:

Symbol 1	Symbol 2
(RUNLENGTH, SIZE)	(AMPLITUDE)

Both *RUNLENGTH* and *SIZE* rest on the same byte, meaning that each only contains four bits of information. The higher bits deal with the number of zeroes, while the lower bits denote the number of bits necessary to encode the value of x.

This has the immediate implication of *Symbol 1* being only able store information regarding the first 15 zeroes preceding the non-zero AC coefficient. However, JPEG defines two special Huffman code words. One is for ending the sequence prematurely when the remaining coefficients are zero (called "End-of-Block" or "EOB"), and another when the run of zeroes goes beyond 15 before reaching a non-zero AC coefficient. In such a case where 16 zeroes are encountered before a given non-zero AC coefficient, *Symbol 1* is encoded "specially" as: (15, 0)(0).

The overall process continues until "EOB" – denoted by (0, 0) – is reached.

With this in mind, the sequence from earlier becomes:

(0, 2)(-3);(1, 2)(-3);(0, 1)(-2);(0, 2)(-6);(0, 1)(2);(0, 1)(-4);(0, 1)(1);(0, 2)(-3);(0, 1)(1);(0, 1)(1);
(0, 2)(5);(0, 1)(1);(0, 1)(2);(0, 1)(-1);(0, 1)(1);(0, 1)(-1);(0, 1)(2);(5, 1)(-1);(0, 1)(-1);(0, 0);

(The first value in the matrix, −26, is the DC coefficient; it is not encoded the same way. See above.)

From here, frequency calculations are made based on occurrences of the coefficients. In our example block, most of the quantized coefficients are small numbers that are not preceded immediately by a zero coefficient. These more-frequent cases will be represented by shorter code words.

Compression ratio and artifacts

The resulting compression ratio can be varied according to need by being more or less aggressive in the divisors used in the quantization phase. Ten to one compression usually results in an image that cannot be distinguished by eye from the original. A compression ratio of 100:1 is usually possible, but will look distinctly artifacted compared to the original. The appropriate level of compression depends on the use to which the image will be put.

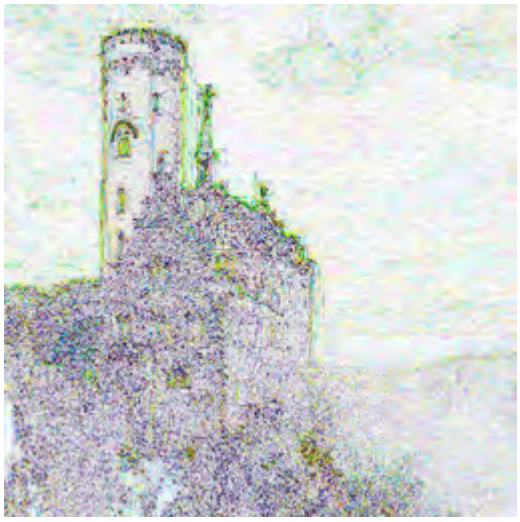
Those who use the World Wide Web may be familiar with the irregularities known as compression artifacts that appear in JPEG images, which may take the form of noise around contrasting edges (especially curves and corners), or "blocky" images. These are due to the quantization step of the JPEG algorithm. They are especially noticeable around sharp corners between contrasting colors (text is a good example, as it contains many such corners). The analogous artifacts in MPEG video are referred to as *mosquito noise*, as the resulting "edge busyness" and spurious dots, which change over time, resemble mosquitoes swarming around the object.[52][53]

These artifacts can be reduced by choosing a lower level of compression; they may be completely avoided by saving an image using a lossless file format, though this will result in a larger file size. The images created with ray-tracing programs have noticeable blocky shapes on the terrain. Certain low-intensity compression artifacts might be acceptable when simply viewing the images, but can be emphasized if the image is subsequently processed, usually resulting in unacceptable quality. Consider the example below, demonstrating the effect of lossy compression on an edge detection processing step.

Image	Lossless compression	Lossy compression
Original		
Processed by Canny edge detector		

Some programs allow the user to vary the amount by which individual blocks are compressed. Stronger compression is applied to areas of the image that show fewer artifacts. This way it is possible to manually reduce JPEG file size with less loss of quality.

Since the quantization stage *always* results in a loss of information, JPEG standard is always a lossy compression codec. (Information is lost both in quantizing and rounding of the floating-point numbers.) Even if the quantization matrix is a matrix of ones, information will still be lost in the rounding step.



This image shows the pixels that are different between a non-compressed image and the same image JPEG compressed with a quality setting of 50. Darker means a larger difference. Note especially the changes occurring near sharp edges and having a block-like shape.



The original image



The compressed 8×8 squares are visible in the scaled-up picture, together with other visual artifacts of the lossy compression.

External image
Illustration of edge busyness (http://i.cmpnet.com/videsignline/2006/02/algolith-fig2.jpg)[52]

Decoding

Decoding to display the image consists of doing all the above in reverse.

Taking the DCT coefficient matrix (after adding the difference of the DC coefficient back in)

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and taking the entry-for-entry product with the quantization matrix from above results in

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which closely resembles the original DCT coefficient matrix for the top-left portion.

The next step is to take the two-dimensional inverse DCT (a 2D type-III DCT), which is given by:

$$f_{x,y} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} \alpha(u)\alpha(v) F_{u,v} \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

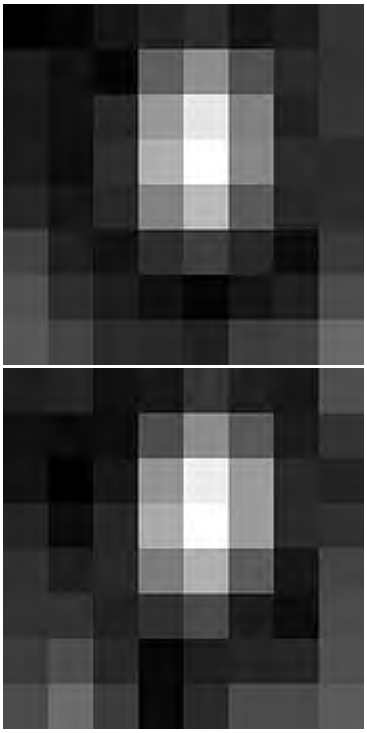
where

- x is the pixel row, for the integers $0 \leq x < 8$.
- y is the pixel column, for the integers $0 \leq y < 8$.
- $\alpha(u)$ is the normalizing scale factor defined earlier, for the integers $0 \leq u < 8$.
- $F_{u,v}$ is the approximated DCT coefficient at coordinates (u, v).
- $f_{x,y}$ is the reconstructed pixel value at coordinates (x, y)

Rounding the output to integer values (since the original had integer values) results in an image with values (still shifted down by 128)

$$\begin{bmatrix} -66 & -63 & -71 & -68 & -56 & -65 & -68 & -46 \\ -71 & -73 & -72 & -46 & -20 & -41 & -66 & -57 \\ -70 & -78 & -68 & -17 & 20 & -14 & -61 & -63 \\ -63 & -73 & -62 & -8 & 27 & -14 & -60 & -58 \\ -58 & -65 & -61 & -27 & -6 & -40 & -68 & -50 \\ -57 & -57 & -64 & -58 & -48 & -66 & -72 & -47 \\ -53 & -46 & -61 & -74 & -65 & -63 & -62 & -45 \\ -47 & -34 & -53 & -74 & -60 & -47 & -47 & -41 \end{bmatrix}$$

and adding 128 to each entry



Slight differences are noticeable between the original (top) and decompressed image (bottom), which is most readily seen in the bottom-left corner.

$$\begin{bmatrix} 62 & 65 & 57 & 60 & 72 & 63 & 60 & 82 \\ 57 & 55 & 56 & 82 & 108 & 87 & 62 & 71 \\ 58 & 50 & 60 & 111 & 148 & 114 & 67 & 65 \\ 65 & 55 & 66 & 120 & 155 & 114 & 68 & 70 \\ 70 & 63 & 67 & 101 & 122 & 88 & 60 & 78 \\ 71 & 71 & 64 & 70 & 80 & 62 & 56 & 81 \\ 75 & 82 & 67 & 54 & 63 & 65 & 66 & 83 \\ 81 & 94 & 75 & 54 & 68 & 81 & 81 & 87 \end{bmatrix}.$$

This is the decompressed subimage. In general, the decompression process may produce values outside the original input range of $[0, 255]$. If this occurs, the decoder needs to clip the output values so as to keep them within that range to prevent overflow when storing the decompressed image with the original bit depth.

The decompressed subimage can be compared to the original subimage (also see images to the right) by taking the difference (original – uncompressed) results in the following error values:

$$\begin{bmatrix} -10 & -10 & 4 & 6 & -2 & -2 & 4 & -9 \\ 6 & 4 & -1 & 8 & 1 & -2 & 7 & 1 \\ 4 & 9 & 8 & 2 & -4 & -10 & -1 & 8 \\ -2 & 3 & 5 & 2 & -1 & -8 & 2 & -1 \\ -3 & -2 & 1 & 3 & 4 & 0 & 8 & -8 \\ 8 & -6 & -4 & -0 & -3 & 6 & 2 & -6 \\ 10 & -11 & -3 & 5 & -8 & -4 & -1 & -0 \\ 6 & -15 & -6 & 14 & -3 & -5 & -3 & 7 \end{bmatrix}$$

with an average absolute error of about 5 values per pixels (i.e., $\frac{1}{64}\sum_{x=0}^{7}\sum_{y=0}^{7}|e(x,y)| = 4.8750$).

The error is most noticeable in the bottom-left corner where the bottom-left pixel becomes darker than the pixel to its immediate right.

Required precision

The required implementation precision of a JPEG codec is implicitly defined through the requirements formulated for compliance to the JPEG standard. These requirements are specified in ITU.T Recommendation T.83 | ISO/IEC 10918-2. Unlike MPEG standards and many later JPEG standards, the above document defines both required implementation precisions for the encoding and the decoding process of a JPEG codec by means of a maximal tolerable error of the forwards and inverse DCT in the DCT domain as determined by reference test streams. For example, the output of a decoder implementation must not exceed an error of one quantization unit in the DCT domain when applied to the reference testing codestreams provided as part of the above standard. While unusual, and unlike many other and more modern standards, ITU.T T.83 | ISO/IEC 10918-2 does not formulate error bounds in the image domain.

Effects of JPEG compression

JPEG compression artifacts blend well into photographs with detailed non-uniform textures, allowing higher compression ratios. Notice how a higher compression ratio first affects the high-frequency textures in the upper-left corner of the image, and how the contrasting lines become more fuzzy. The very high compression ratio severely affects the quality of the image, although the overall colors and image form are still recognizable. However, the precision of colors suffer less (for a human eye) than the precision of contours (based on luminance). This justifies the fact that images should be first transformed in a color model separating the luminance from the chromatic information, before subsampling the chromatic planes (which may also use lower quality quantization) in order to preserve the precision of the luminance plane with more information bits.

Sample photographs

For information, the uncompressed 24-bit RGB bitmap image below (73,242 pixels) would require 219,726 bytes (excluding all other information headers). The filesizes indicated below include the internal JPEG information headers and some metadata. For highest quality images (Q=100), about

8.25 bits per color pixel is required. On grayscale images, a minimum of 6.5 bits per pixel is enough (a comparable Q=100 quality color information requires about 25% more encoded bits). The highest quality image below (Q=100) is encoded at nine bits per color pixel, the medium quality image (Q=25) uses one bit per color pixel. For most applications, the quality factor should not go below 0.75 bit per pixel (Q=12.5), as demonstrated by the low quality image. The image at lowest quality uses only 0.13 bit per pixel, and displays very poor color. This is useful when the image will be displayed in a significantly scaled-down size. A method for creating better quantization matrices for a given image quality using PSNR instead of the Q factor is described in Minguillón & Pujol (2001).[54]



Visual impact of a jpeg compression on Photoshop on a picture of 4480x4480 pixels

Image	Quality	Size (bytes)	Compression ratio	Comment
	Highest quality (Q = 100)	81,447	2.7:1	Extremely minor artifacts
	High quality (Q = 50)	14,679	15:1	Initial signs of subimage artifacts
	Medium quality (Q = 25)	9,407	23:1	Stronger artifacts; loss of high frequency information
	Low quality (Q = 10)	4,787	46:1	Severe high frequency loss leads to obvious artifacts on subimage boundaries ("macroblocking")
	Lowest quality (Q = 1)	1,523	144:1	Extreme loss of color and detail; the leaves are nearly unrecognizable.

Note: The above images are not IEEE / CCIR / EBU test images, and the encoder settings are not specified or available.

The medium quality photo uses only 4.3% of the storage space required for the uncompressed image, but has little noticeable loss of detail or visible artifacts. However, once a certain threshold of compression is passed, compressed images show increasingly visible defects. See the article on rate–distortion theory for a mathematical explanation of this threshold effect. A particular limitation of JPEG in this regard is its non-overlapped 8×8 block transform structure. More modern designs such

as JPEG 2000 and JPEG XR exhibit a more graceful degradation of quality as the bit usage decreases – by using transforms with a larger spatial extent for the lower frequency coefficients and by using overlapping transform basis functions.

Lossless further compression

From 2004 to 2008, new research emerged on ways to further compress the data contained in JPEG images without modifying the represented image.[55][56][57][58] This has applications in scenarios where the original image is only available in JPEG format, and its size needs to be reduced for archiving or transmission. Standard general-purpose compression tools cannot significantly compress JPEG files.

Typically, such schemes take advantage of improvements to the naive scheme for coding DCT coefficients, which fails to take into account:

- Correlations between magnitudes of adjacent coefficients in the same block;
- Correlations between magnitudes of the same coefficient in adjacent blocks;
- Correlations between magnitudes of the same coefficient/block in different channels;
- The DC coefficients when taken together resemble a downscale version of the original image multiplied by a scaling factor. Well-known schemes for lossless coding of continuous-tone images can be applied, achieving somewhat better compression than the Huffman coded DPCM used in JPEG.

Some standard but rarely used options already exist in JPEG to improve the efficiency of coding DCT coefficients: the arithmetic coding option, and the progressive coding option (which produces lower bitrates because values for each coefficient are coded independently, and each coefficient has a significantly different distribution). Modern methods have improved on these techniques by reordering coefficients to group coefficients of larger magnitude together;[55] using adjacent coefficients and blocks to predict new coefficient values;[57] dividing blocks or coefficients up among a small number of independently coded models based on their statistics and adjacent values;[56][57] and most recently, by decoding blocks, predicting subsequent blocks in the spatial domain, and then encoding these to generate predictions for DCT coefficients.[58]

Typically, such methods can compress existing JPEG files between 15 and 25 percent, and for JPEGs compressed at low-quality settings, can produce improvements of up to 65%.[57][58]

A freely available tool called packJPG is based on the 2007 paper "Improved Redundancy Reduction for JPEG Files." As of version 2.5k of 2016, it reports a typical 20% reduction by transcoding.[59] JPEG XL (ISO/IEC 18181) of 2018 reports a similar reduction in its transcoding.

Derived formats for stereoscopic 3D

JPEG Stereoscopic

JPS is a stereoscopic JPEG image used for creating 3D effects from 2D images. It contains two static images, one for the left eye and one for the right eye; encoded as two side-by-side images in a single JPG file. JPEG Stereoscopic (JPS, extension .jps) is a JPEG-based format for stereoscopic images.[60][61] It has a range of configurations stored in the JPEG APP3 marker field, but usually

contains one image of double width, representing two images of identical size in cross-eyed (i.e. left frame on the right half of the image and vice versa) side-by-side arrangement. This file format can be viewed as a JPEG without any special software, or can be processed for rendering in other modes.



An example of a stereoscopic .JPS file

JPEG Multi-Picture Format

JPEG Multi-Picture Format (MPO, extension .mpo) is a JPEG-based format for storing multiple images in a single file. It contains two or more JPEG files concatenated together.[62][63] It also defines a JPEG APP2 marker segment for image description. Various devices use it to store 3D images, such as Fujifilm FinePix Real 3D W1, HTC Evo 3D, JVC GY-HMZ1U AVCHD/MVC extension camcorder, Nintendo 3DS, Panasonic Lumix DMC-TZ20, DMC-TZ30, DMC-TZ60, DMC-TS4 (FT4), and Sony DSC-HX7V. Other devices use it to store "preview images" that can be displayed on a TV.

In the last few years, due to the growing use of stereoscopic images, much effort has been spent by the scientific community to develop algorithms for stereoscopic image compression.[64][65]

Implementations

A very important implementation of a JPEG codec is the free programming library *libjpeg* of the Independent JPEG Group. It was first published in 1991 and was key for the success of the standard. This library was used in countless applications. [66] The development went quiet in 1998; when libjpeg resurfaced with the 2009 version 7, it broke ABI compatibility with previous versions. Version 8 of 2010 introduced non-standard extensions, a decision criticized by the original IJG leader Tom Lane.[67]

libjpeg-turbo, forked from the 1998 libjpeg 6b, improves on libjpeg with SIMD optimizations. Originally seen as a maintained fork of libjpeg, it has become more popular after the incompatible changes of 2009.[68][69] In 2019, it became the ITU|ISO/IEC reference implementation as ISO/IEC 10918-7 and ITU-T T.873.[70]

ISO/IEC Joint Photography Experts Group maintains the other reference software implementation under the JPEG XT heading. It can encode both base JPEG (ISO/IEC 10918-1 and 18477–1) and JPEG XT extensions (ISO/IEC 18477 Parts 2 and 6–9), as well as JPEG-LS (ISO/IEC 14495).[71]

There is persistent interest in encoding JPEG in unconventional ways that maximize image quality for a given file size. In 2014, Mozilla created mozjpeg from libjpeg-turbo, a slower but higher-quality encoder intended for web images.[72] In 2016, "JPEG on steroids" was introduced as an option for the ISO JPEG XT reference implementation.[73] In March 2017, Google released the open source project Guetzli, which trades off a much longer encoding time for smaller file size (similar to what Zopfli does for PNG and other lossless data formats).[74]

JPEG XT

JPEG XT (ISO/IEC 18477) was published in June 2015; it extends base JPEG format with support for higher integer bit depths (up to 16 bit), high dynamic range imaging and floating-point coding, lossless coding, and alpha channel coding. Extensions are backward compatible with the base

JPEG/JFIF file format and 8-bit lossy compressed image. JPEG XT uses an extensible file format based on JFIF. Extension layers are used to modify the JPEG 8-bit base layer and restore the high-resolution image. Existing software is forward compatible and can read the JPEG XT binary stream, though it would only decode the base 8-bit layer.[75]

JPEG XL

JPEG XL (ISO/IEC 18181) was published in 2021–2022. It replaces the JPEG format with a new DCT-based royalty-free format and allows efficient transcoding as a storage option for traditional JPEG images.[76] The new format is designed to exceed the still image compression performance shown by HEVC HM, Daala and WebP. It supports billion-by-billion pixel images, up to 32-bit-per-component high dynamic range with the appropriate transfer functions (PQ and HLG), patch encoding of synthetic images such as bitmap fonts and gradients, animated images, alpha channel coding, and a choice of RGB/YCbCr/ICtCp color encoding.[77][78][79][80]

See also

- Better Portable Graphics, a format based on intra-frame encoding of the HEVC
- C-Cube, an early implementer of JPEG in chip form
- Comparison of graphics file formats
- Deblocking filter (video), the similar deblocking methods could be applied to JPEG
- Design rule for Camera File system (DCF)
- FELICS, a lossless image codec
- File extensions
- Graphics editing program
- High Efficiency Image File Format, image container format for HEVC and other image coding formats
- Lenna (test image), the traditional standard image used to test image processing algorithms
- Motion JPEG
- WebP

References

1. "T.81 – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES" (https://www.w3.org/Graphics/JPEG/itu-t81.pdf) (PDF). CCITT. September 1992. Retrieved 12 July 2019.
2. "Definition of "JPEG" " (https://www.collinsdictionary.com/dictionary/english/jpeg). *Collins English Dictionary*. Retrieved 2013-05-23.
3. Haines, Richard F.; Chuang, Sherry L. (1 July 1992). *The effects of video compression on acceptability of images for monitoring life sciences experiments* (https://ntrs.nasa.gov/search.jsp?R=19920024689) (Technical report). NASA. NASA-TP-3239, A-92040, NAS 1.60:3239. Retrieved 2016-03-13. "The JPEG still-image-compression levels, even with the large range of 5:1 to 120:1 in this study, yielded equally high levels of acceptability"

4. Hudson, Graham; Léger, Alain; Niss, Birger; Sebestyén, István; Vaaben, Jørgen (31 August 2018). "JPEG-1 standard 25 years: past, present, and future reasons for a success" (https://doi.org/10.1117%2F1.JEI.27.4.040901). *Journal of Electronic Imaging*. **27** (4): 1. doi:10.1117/1.JEI.27.4.040901 (https://doi.org/10.1117%2F1.JEI.27.4.040901). S2CID 52164892 (https://api.semanticscholar.org/CorpusID:52164892).
5. "The JPEG Image Format Explained" (https://web.archive.org/web/20190805194553/https://home.bt.com/tech-gadgets/photography/what-is-a-jpeg-11364206889349). *BT.com*. BT Group. 31 May 2018. Archived from the original (https://home.bt.com/tech-gadgets/photography/what-is-a-jpeg-11364206889349) on 5 August 2019. Retrieved 5 August 2019.
6. Baraniuk, Chris (15 October 2015). "Copy Protections Could Come to JPEGs" (https://www.bbc.co.uk/news/technology-34538705). *BBC News*. BBC. Retrieved 13 September 2019.
7. "JPEG: 25 Jahre und kein bisschen alt" (https://www.heise.de/newsticker/meldung/JPEG-25-Jahre-und-kein-bisschen-alt-3342519.html). *Heise online* (in German). October 2016. Retrieved 5 September 2019.
8. "What Is a JPEG? The Invisible Object You See Every Day" (https://www.theatlantic.com/technology/archive/2013/09/what-is-a-jpeg-the-invisible-object-you-see-every-day/279954/). *The Atlantic*. 24 September 2013. Retrieved 13 September 2019.
9. "HTTP Archive - Interesting Stats" (http://httparchive.org/interesting.php#imageformats). *httparchive.org*. Retrieved 2016-04-06.
10. "MIME Type Detection in Internet Explorer" (https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775147(v=vs.85)). Microsoft. Retrieved 2 November 2022.
11. "JPEG File Interchange Format" (https://web.archive.org/web/20140903080533/http://www.jpeg.org/public/jfif.pdf) (PDF). 3 September 2014. Archived from the original on 3 September 2014. Retrieved 16 October 2017.
12. "Why JPEG 2000 Never Took Off" (https://blog.ansi.org/2018/07/why-jpeg-2000-never-used-standard-iso-iec/). *American National Standards Institute*. 10 July 2018. Retrieved 13 September 2019.
13. Lemos, Robert (23 July 2002). "Finding patent truth in JPEG claim" (https://www.cnet.com/news/finding-patent-truth-in-jpeg-claim/). *CNET*. Retrieved 13 July 2019.
14. ISO/IEC JTC 1/SC 29 (2009-05-07). "ISO/IEC JTC 1/SC 29/WG 1 – Coding of Still Pictures (SC 29/WG 1 Structure)" (https://web.archive.org/web/20131231055215/http://kikaku.itscj.ipsj.or.jp/sc29/29w12901.htm). Archived from the original (http://kikaku.itscj.ipsj.or.jp/sc29/29w12901.htm) on 2013-12-31. Retrieved 2009-11-11.
15. ISO/IEC JTC 1/SC 29. "Programme of Work, (Allocated to SC 29/WG 1)" (https://web.archive.org/web/20131231012300/http://kikaku.itscj.ipsj.or.jp/sc29/29w42901.htm). Archived from the original (http://kikaku.itscj.ipsj.or.jp/sc29/29w42901.htm) on 2013-12-31. Retrieved 2009-11-07.
16. ISO. "JTC 1/SC 29 – Coding of audio, picture, multimedia and hypermedia information" (http://www.iso.org/iso/standards_development/technical_committees/list_of_iso_technical_committees/iso_technical_committee.htm?commid=45316). Retrieved 2009-11-11.
17. JPEG. "Joint Photographic Experts Group, JPEG Homepage" (http://www.jpeg.org/jpeg/index.html). Retrieved 2009-11-08.
18. "T.81 : Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines" (http://www.itu.int/rec/T-REC-T.81). *Itu.int*. Retrieved 2009-11-07.
19. William B. Pennebaker; Joan L. Mitchell (1993). *JPEG still image data compression standard* (https://books.google.com/books?id=AepB_PZ_WMkC&dq=JPEG+%22did+not+specify+a+file+format%22&pg=PA291) (3rd ed.). Springer. p. 291. ISBN 978-0-442-01272-4.

20. ISO. "JTC 1/SC 29 – Coding of audio, picture, multimedia and hypermedia information" (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=45316). Retrieved 2009-11-07.
21. "SPIFF, Still Picture Interchange File Format" (https://www.loc.gov/preservation/digital/formats/fdd/fdd000019.shtml). Library of Congress. 30 January 2012. Archived (https://web.archive.org/web/20180731093543/https://www.loc.gov/preservation/digital/formats/fdd/fdd000019.shtml) from the original on 2018-07-31. Retrieved 2018-07-31.
22. JPEG (2009-04-24). "JPEG XR enters FDIS status: JPEG File Interchange Format (JFIF) to be standardized as JPEG Part 5" (https://web.archive.org/web/20091008041637/http://www.jpeg.org/newsrel25.html) (Press release). Archived from the original (http://www.jpeg.org/newsrel25.html) on 2009-10-08. Retrieved 2009-11-09.
23. "JPEG File Interchange Format (JFIF)" (http://www.ecma-international.org/publications/techreports/E-TR-098.htm). *ECMA TR/98 1st ed*. Ecma International. 2009. Retrieved 2011-08-01.
24. "Forgent's JPEG Patent" (https://pmt.sourceforge.io/SVG-patents/jpeg.html). *SourceForge*. 2002. Retrieved 13 July 2019.
25. "Concerning recent patent claims" (https://web.archive.org/web/20070714232941/http://www.jpeg.org/newsrel1.html). *Jpeg.org*. 2002-07-19. Archived from the original (http://www.jpeg.org/newsrel1.html) on 2007-07-14. Retrieved 2011-05-29.
26. "JPEG and JPEG2000 – Between Patent Quarrel and Change of Technology" (https://web.archive.org/web/20040817154508/http://www.algovision-luratech.com/company/news/patentquarrel.jsp?OnlineShopId=164241031081525276). Archived from the original on August 17, 2004. Retrieved 2017-04-16.
27. Kawamoto, Dawn (April 22, 2005). "Graphics patent suit fires back at Microsoft" (https://www.cnet.com/culture/graphics-patent-suit-fires-back-at-microsoft/). CNET News. Retrieved 2023-01-20.
28. "Trademark Office Re-examines Forgent JPEG Patent" (http://arquivo.pt/wayback/20160515113624/http://www.publish.com/c/a/Graphics-Tools/Trademark-Office-Reexamines-Forgent-JPEG-Patent/). *Publish.com*. February 3, 2006. Archived from the original (http://www.publish.com/c/a/Graphics-Tools/Trademark-Office-Reexamines-Forgent-JPEG-Patent/) on 2016-05-15. Retrieved 2009-01-28.
29. "USPTO: Broadest Claims Forgent Asserts Against JPEG Standard Invalid" (http://www.groklaw.net/article.php?story=20060526105754880). *Groklaw.net*. May 26, 2006. Retrieved 2007-07-21.
30. "Coding System for Reducing Redundancy" (https://web.archive.org/web/20110612114555/http://gauss.ffii.org/PatentView/EP266049). *Gauss.ffii.org*. Archived from the original (http://gauss.ffii.org/PatentView/EP266049) on 2011-06-12. Retrieved 2011-05-29.
31. "JPEG Patent Claim Surrendered" (http://www.pubpat.org/jpegsurrendered.htm). Public Patent Foundation. November 2, 2006. Retrieved 2006-11-03.
32. "Ex Parte Reexamination Certificate for U.S. Patent No. 5,253,341" (https://web.archive.org/web/20080602141045/http://www.uspto.gov/web/patents/patog/week30/OG/html/1320-4/US05253341-20070724.html). Archived from the original (http://www.uspto.gov/web/patents/patog/week30/OG/html/1320-4/US05253341-20070724.html) on June 2, 2008.
33. Workgroup. "Rozmanith: Using Software Patents to Silence Critics" (https://web.archive.org/web/20110716123228/http://eupat.ffii.org/pikta/xrani/rozmanith/index.en.html). *Eupat.ffii.org*. Archived from the original (http://eupat.ffii.org/pikta/xrani/rozmanith/index.en.html) on 2011-07-16. Retrieved 2011-05-29.
34. "A Bounty of $5,000 to Name Troll Tracker: Ray Niro Wants To Know Who Is saying All Those Nasty Things About Him" (http://www.law.com/jsp/article.jsp?id=1196762670106). *Law.com*. Retrieved 2011-05-29.

35. Reimer, Jeremy (2008-02-05). "Hunting trolls: USPTO asked to reexamine broad image patent" (https://arstechnica.com/news.ars/post/20080205-hunting-trolls-uspto-asked-to-reexamine-broad-image-patent.html). *Arstechnica.com*. Retrieved 2011-05-29.
36. U.S. Patent Office – Granting Reexamination on 5,253,341 C1
37. "Judge Puts JPEG Patent On Ice" (https://www.techdirt.com/articles/20080427/143205960.shtml). *Techdirt.com*. 2008-04-30. Retrieved 2011-05-29.
38. "JPEG Patent's Single Claim Rejected (And Smacked Down For Good Measure)" (https://techdirt.com/articles/20080731/0337491852.shtml). *Techdirt.com*. 2008-08-01. Retrieved 2011-05-29.
39. Workgroup. "Princeton Digital Image Corporation Home Page" (https://archive.is/20130411084058/http://www.princetondigitalimage.com/). Archived from the original (http://www.princetondigitalimage.com/) on 2013-04-11. Retrieved 2013-05-01.
40. Workgroup. "Article on Princeton Court Ruling Regarding GE License Agreement" (https://web.archive.org/web/20160309142416/http://patentlaw.jmbm.com/2013/04/hps-motion-to-dismiss-for-lack.html). Archived from the original (http://patentlaw.jmbm.com/2013/04/hps-motion-to-dismiss-for-lack.html/) on 2016-03-09. Retrieved 2013-05-01.
41. "Progressive Decoding Overview" (http://msdn.microsoft.com/en-us/library/ee720036(v=vs.85).aspx). *Microsoft Developer Network*. Microsoft. Retrieved 2012-03-23.
42. Fastvideo (May 2019). "12-bit JPEG encoder on GPU" (https://www.fastcompression.com/products/12-bit-jpeg-codec.htm). Retrieved 2019-05-06.
43. "Why You Should Always Rotate Original JPEG Photos Losslessly" (http://petapixel.com/2012/08/14/why-you-should-always-rotate-original-jpeg-photos-losslessly/). *Petapixel.com*. 14 August 2012. Retrieved 16 October 2017.
44. "JFIF File Format as PDF" (http://www.w3.org/Graphics/JPEG/jfif3.pdf) (PDF).
45. Tom Lane (1999-03-29). "JPEG image compression FAQ" (http://www.faqs.org/faqs/jpeg-faq/part1/). Retrieved 2007-09-11. (q. 14: "Why all the argument about file formats?")
46. "A Standard Default Color Space for the Internet - sRGB" (https://www.w3.org/Graphics/Color/sRGB.html). *www.w3.org*.
47. "IEC 61966-2-1:1999/AMD1:2003 | IEC Webstore" (https://webstore.iec.ch/publication/6168). *webstore.iec.ch*.
48. "ISO/IEC 10918-1 : 1993(E) p.36" (http://www.digicamsoft.com/itu/itu-t81-36.html).
49. Thomas G. Lane. "Advanced Features: Compression parameter selection" (http://apodeline.free.fr/DOC/libjpeg/libjpeg-3.html). *Using the IJG JPEG Library*.
50. Ryan, Dan (2012-06-20). *E - Learning Modules: Dlr Associates Series* (https://books.google.com/books?id=bYxMVVzdV80C&dq=A+particular+conversion+to+Y%E2%80%B2CBCR+is+specified+in+the+JFIF+standard%2C+and+should+be+performed+for+the+resulting+JPEG+file+to+have+maximum+compatibility.+A+particular+conversion+to+Y%E2%80%B2CBCR+is+specified+in+the+JFIF+standard%2C+and+should+be+performed+for+the+resulting+JPEG+file+to+have+maximum+compatibility.++A+particular+conversion+to+Y%E2%80%B2CBCR+is+specified+in+the+JFIF+standard%2C+and+should+be+performed+for+the+resulting+JPEG+file+to+have+maximum+compatibility.+&pg=PA53). AuthorHouse. ISBN 9781468575200.
51. "DC / AC Frequency Questions - Doom9's Forum" (http://forum.doom9.org/showthread.php?p=184647#post184647). *forum.doom9.org*. Retrieved 16 October 2017.
52. Phuc-Tue Le Dinh and Jacques Patry. Video compression artifacts and MPEG noise reduction (http://www.videsignline.com/howto/180207350) Archived (https://web.archive.org/web/20060314201405/http://www.videsignline.com/howto/180207350) 2006-03-14 at the Wayback Machine. Video Imaging DesignLine. February 24, 2006. Retrieved May 28, 2009.

53. "**3.9 mosquito noise:** Form of edge busyness distortion sometimes associated with movement, characterized by moving artifacts and/or blotchy noise patterns superimposed over the objects (resembling a mosquito flying around a person's head and shoulders)." ITU-T Rec. P.930 (08/96) Principles of a reference impairment system for video (http://eu.sabotage.org/www/ITU/P/P0930e.pdf) Archived (https://web.archive.org/web/20100216033245/http://eu.sabotage.org/www/ITU/P/P0930e.pdf) 2010-02-16 at the Wayback Machine
54. Julià Minguillón, Jaume Pujol (April 2001). "JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes" (http://openaccess.uoc.edu/webapps/o2/bitstream/10609/6263/6/jei-jpeg.pdf) (PDF). *Electronic Imaging*. **10** (2): 475–485. Bibcode:2001JEI....10..475M (https://ui.adsabs.harvard.edu/abs/2001JEI....10..475M). doi:10.1117/1.1344592 (https://doi.org/10.1117%2F1.1344592). hdl:10609/6263 (https://hdl.handle.net/10609%2F6263).
55. I. Bauermann and E. Steinbacj. Further Lossless Compression of JPEG Images. Proc. of Picture Coding Symposium (PCS 2004), San Francisco, US, December 15–17, 2004.
56. N. Ponomarenko, K. Egiazarian, V. Lukin and J. Astola. Additional Lossless Compression of JPEG Images, Proc. of the 4th Intl. Symposium on Image and Signal Processing and Analysis (ISPA 2005), Zagreb, Croatia, pp. 117–120, September 15–17, 2005.
57. M. Stirner and G. Seelmann. Improved Redundancy Reduction for JPEG Files. Proc. of Picture Coding Symposium (PCS 2007), Lisbon, Portugal, November 7–9, 2007
58. Ichiro Matsuda, Yukio Nomoto, Kei Wakabayashi and Susumu Itoh. Lossless Re-encoding of JPEG images using block-adaptive intra prediction. Proceedings of the 16th European Signal Processing Conference (EUSIPCO 2008).
59. Stirner, Matthias (19 February 2023). "packjpg/packJPG" (https://github.com/packjpg/packJPG). *GitHub*.
60. J. Siragusa, D. C. Swift (1997). "General Purpose Stereoscopic Data Descriptor" (https://web.archive.org/web/20111030182549/http://vrex.com/developer/sterdesc.pdf) (PDF). *VRex, Inc., Elmsford, New York, US*. Archived from the original (http://vrex.com/developer/sterdesc.pdf) (PDF) on 2011-10-30.
61. Tim Kemp, JPS files (http://ephehm.com/jps/)
62. "Multi-Picture Format" (https://web.archive.org/web/20160405200235/http://cipa.jp/std/documents/e/DC-007_E.pdf) (PDF). 2009. Archived from the original (http://www.cipa.jp/std/documents/e/DC-007_E.pdf) (PDF) on 2016-04-05. Retrieved 2015-12-30.
63. "MPO2Stereo: Convert Fujifilm MPO files to JPEG stereo pairs" (http://www.mtbs3d.com/phpbb/viewtopic.php?f=3&t=4124&start=15), *Mtbs3d.com*, retrieved 12 January 2010
64. Alessandro Ortis; Sebastiano Battiato (2015), Sitnik, Robert; Puech, William (eds.), "A new fast matching method for adaptive compression of stereoscopic images" (http://iplab.dmi.unict.it/publication/503), *Three-Dimensional Image Processing*, Three-Dimensional Image Processing, Measurement (3DIPM), and Applications 2015, SPIE - Three-Dimensional Image Processing, Measurement (3DIPM), and Applications 2015, **9393**: 93930K, Bibcode:2015SPIE.9393E..0KO (https://ui.adsabs.harvard.edu/abs/2015SPIE.9393E..0KO), doi:10.1117/12.2086372 (https://doi.org/10.1117%2F12.2086372), S2CID 18879942 (https://api.semanticscholar.org/CorpusID:18879942), retrieved 30 April 2015
65. Alessandro Ortis; Francesco Rundo; Giuseppe Di Giore; Sebastiano Battiato, *Adaptive Compression of Stereoscopic Images* (http://iplab.dmi.unict.it/publication/49), International Conference on Image Analysis and Processing (ICIAP) 2013, retrieved 30 April 2015
66. "Overview of JPEG" (http://jpeg.org/jpeg). *jpeg.org*. Retrieved 2017-10-16.
67. Tom Lane, January 16, 2013: jpeg-9, API/ABI compatibility, and the future role of this project (https://lists.fedoraproject.org/pipermail/devel/2013-January/176400.html)

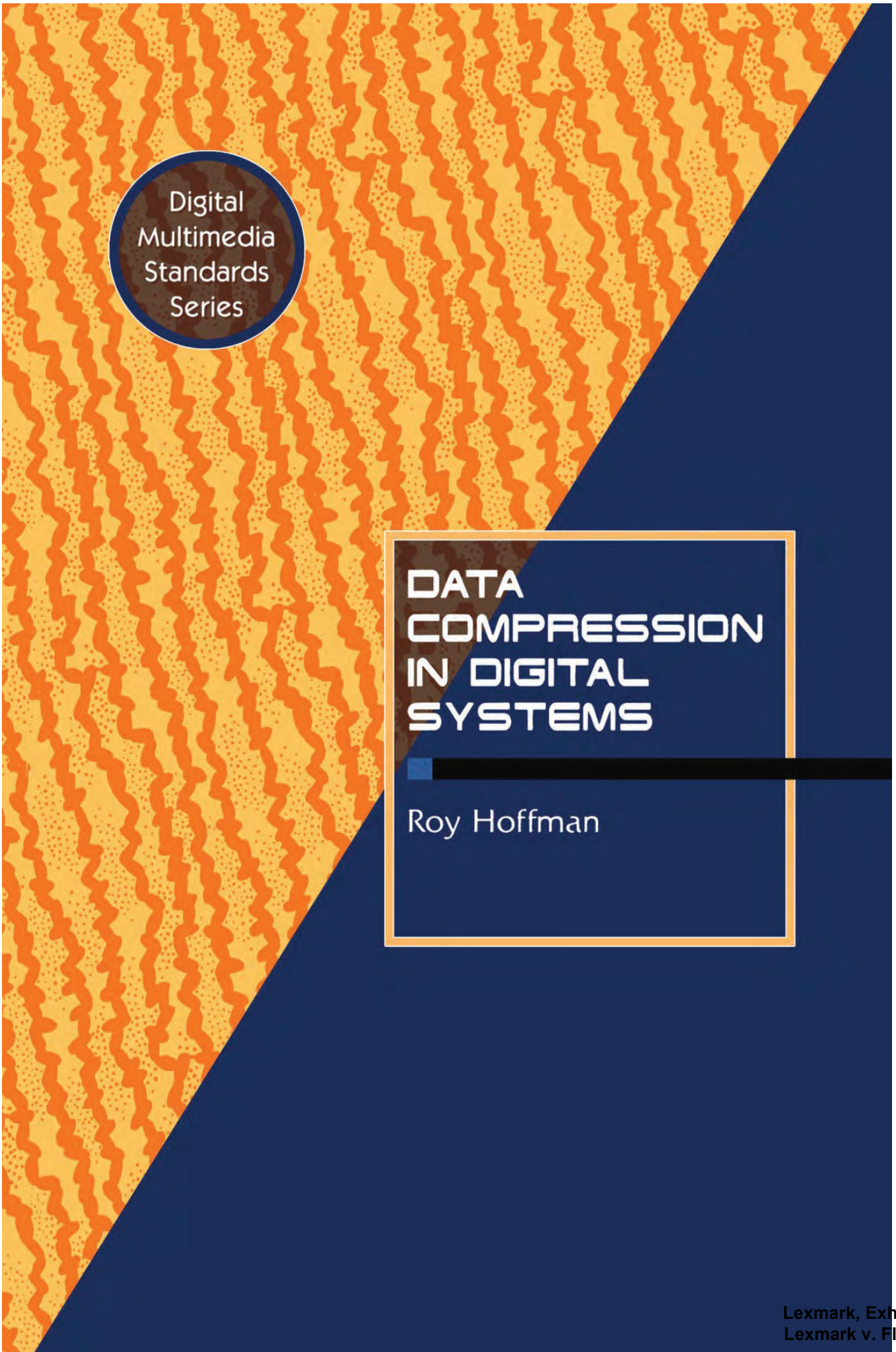
68. Software That Uses or Provides libjpeg-turbo (http://libjpeg-turbo.virtualgl.org/About/Software). February 9, 2012.
69. Issue 48789 – chromium – Use libjpeg-turbo instead of libjpeg (https://code.google.com/p/chromium/issues/detail?id=48789#c19). April 14, 2011.
70. "ISO/IEC 10918-7: 2019 Information technology — Digital compression and coding of continuous-tone still images — Part 7: Reference software" (https://www.iso.org/standard/75845.html). *ISO*."T.873 (05/19): Information technology - Digital compression and coding of continuous-tone still images: Reference software" (https://www.itu.int/rec/T-REC-T.873-201905-S/en). *www.itu.int*.
71. "JPEG - JPEG XT" (https://jpeg.org/jpegxt/software.html). *jpeg.org*.
72. "Introducing the 'mozjpeg' Project" (https://research.mozilla.org/2014/03/05/introducing-the-mozjpeg-project/). *Mozilla Research*.
73. Richter, Thomas (September 2016). "JPEG on STEROIDS: Common optimization techniques for JPEG image compression". *2016 IEEE International Conference on Image Processing (ICIP)*: 61–65. doi:10.1109/ICIP.2016.7532319 (https://doi.org/10.1109%2FICIP.2016.7532319). ISBN 978-1-4673-9961-6. S2CID 14922251 (https://api.semanticscholar.org/CorpusID:14922251).
74. "Announcing Guetzli: A New Open Source JPEG Encoder" (https://research.googleblog.com/2017/03/announcing-guetzli-new-open-source-jpeg.html). *Research.googleblog.com*. Retrieved 16 October 2017.
75. "JPEG - JPEG XT" (https://jpeg.org/jpegxt/). *jpeg.org*.
76. Alakuijala, Jyrki; van Asseldonk, Ruud; Boukortt, Sami; Bruse, Martin; Comșa, Iulia-Maria; Firsching, Moritz; Fischbacher, Thomas; Kliuchnikov, Evgenii; Gomez, Sebastian; Obryk, Robert; Potempa, Krzysztof; Rhatushnyak, Alexander; Sneyers, Jon; Szabadka, Zoltan; Vandervenne, Lode; Versari, Luca; Wassenberg, Jan (2019-09-06). "JPEG XL next-generation image compression architecture and coding tools" (https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11137/111370K/JPEG-XL-next-generation-image-compression-architecture-and-coding-tools/10.1117/12.2529237.full?SSO=1). In Tescher, Andrew G; Ebrahimi, Touradj (eds.). *Applications of Digital Image Processing XLII*. p. 20. doi:10.1117/12.2529237 (https://doi.org/10.1117%2F12.2529237). ISBN 9781510629677. S2CID 202785129 (https://api.semanticscholar.org/CorpusID:202785129).
77. Rhatushnyak, Alexander; Wassenberg, Jan; Sneyers, Jon; Alakuijala, Jyrki; Vandevenne, Lode; Versari, Luca; Obryk, Robert; Szabadka, Zoltan; Kliuchnikov, Evgenii; Comsa, Iulia-Maria; Potempa, Krzysztof; Bruse, Martin; Firsching, Moritz; Khasanova, Renata; Ruud van Asseldonk; Boukortt, Sami; Gomez, Sebastian; Fischbacher, Thomas (2019). "Committee Draft of JPEG XL Image Coding System". arXiv:1908.03565 (https://arxiv.org/abs/1908.03565) [eess.IV (https://arxiv.org/archive/eess.IV)].
78. "N79010 Final Call for Proposals for a Next-Generation Image Coding Standard (JPEG XL)" (https://jpeg.org/downloads/jpegxl/jpegxl-cfp.pdf) (PDF). *ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG16)*. Retrieved 29 May 2018.
79. *ISO/IEC 18181-1:2022 Information technology — JPEG XL image coding system — Part 1: Core coding system* (https://www.iso.org/standard/77977.html).
80. *ISO/IEC 18181-2:2021 Information technology — JPEG XL image coding system — Part 2: File format* (https://www.iso.org/standard/80617.html).

External links

- Official website (https://jpeg.org/jpeg/)
- JPEG Standard (JPEG ISO/IEC 10918-1 ITU-T Recommendation T.81) (https://www.w3.org/Graphics/JPEG/itu-t81.pdf) at W3.org

- JFIF File Format (https://www.w3.org/Graphics/JPEG/jfif3.pdf) at W3.org
- Example images over the full range of quantization levels from 1 to 100 (https://www.visengi.com/products/jpeg_hardware_encoder) at visengi.com
- JPEG decoder open source code, copyright (C) 1995–1997, Thomas G. Lane (https://opensource.apple.com/source/WebCore/WebCore-1C25/platform/image-decoders/jpeg/)

Retrieved from "https://en.wikipedia.org/w/index.php?title=JPEG&oldid=1150326002"



Lexmark, Exhibit 1002
Lexmark v. Flexiworld

DATA COMPRESSION IN DIGITAL SYSTEMS

Roy Hoffman
Former IBM Senior Technical Staff Member
and a member of the
IBM Academy of Technology

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Join Us on the Internet

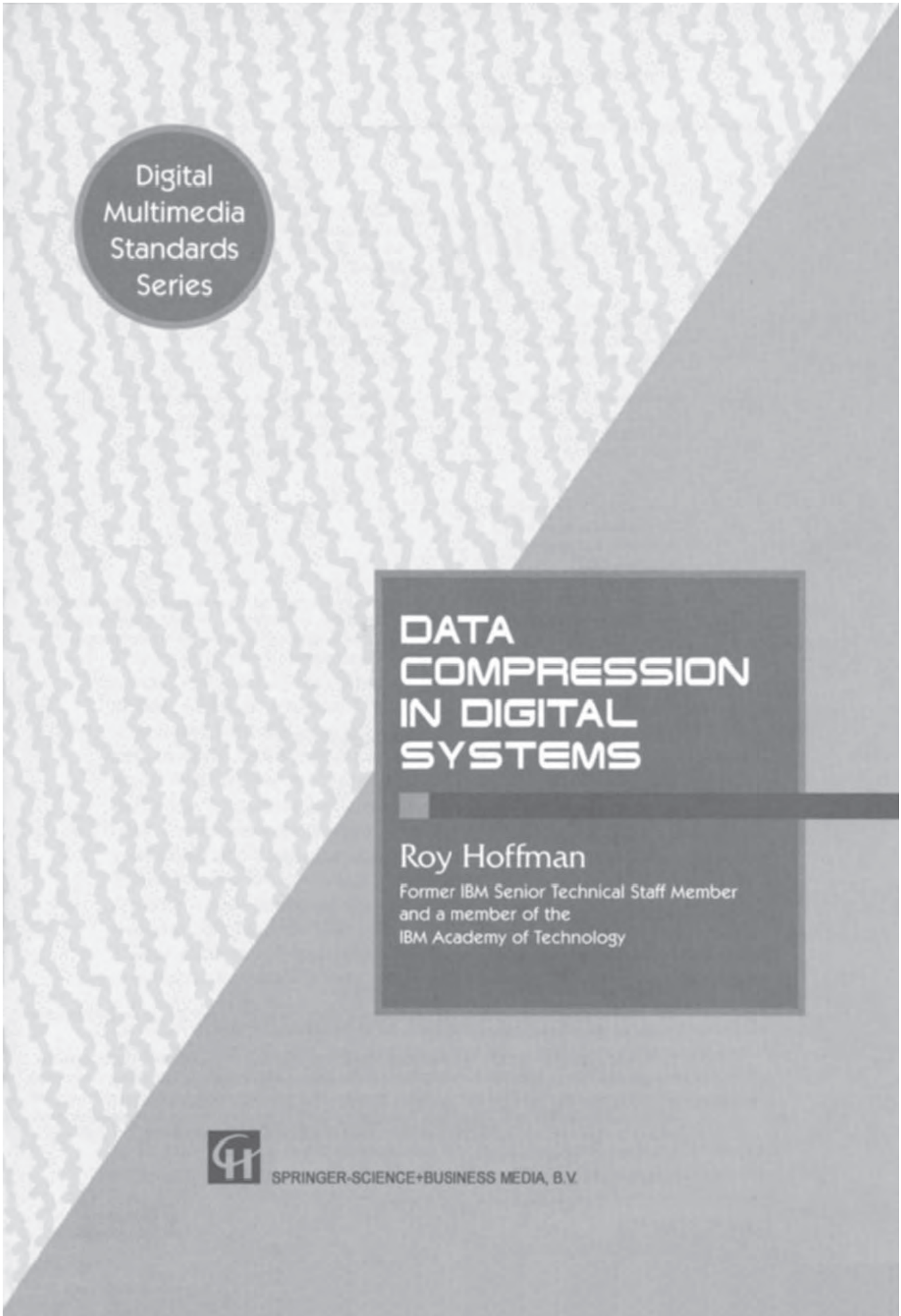
WWW: http://www.thomson.com
EMAIL: findit@kiosk.thomson.com

thomson.com is the on-line portal for the products, services and resources available from International Thomson Publishing (ITP). This Internet kiosk gives users immediate access to more than 34 ITP publishers and over 20,000 products. Through *thomson.com* Internet users can search catalogs, examine subject-specific resource centers and subscribe to electronic discussion lists. You can purchase ITP products from your local bookseller, or directly through *thomson.com.*

Visit Chapman & Hall's Internet Resource Center for information on our new publications, links to useful sites on the World Wide Web and an opportunity to join our e-mail mailing list. Point your browser to: **http://www.chaphall.com** or **http://www.thomson.com/chaphall/electeng.html** for Electrical Engineering

---

A service of I(T)P®



Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Cover design: Curtis Tow Graphics

Copyright © 1997 by Springer Science+Business Media Dordrecht
Originally published by Chapman & Hall in 1997
Softcover reprint of the hardcover 1st edition 1997

All rights reserved. No part of this book covered by the copyright hereon may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without the written permission of the publisher.

2 3 4 5 6 7 8 9 10 XXX 01 00 99 98

Library of Congress Cataloging-in-Publication Data

Hoffman, Roy (Roy L.)
Data compression in digital systems / by Roy Hoffman.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4613-7764-1 ISBN 978-1-4615-6031-9 (eBook)
DOI 10.1007/978-1-4615-6031-9
1. Data compression (Computer science). 2. Digital electronics.
I. Title.
QA76.9.D33H64 1996
005.74'6--dc20 96-9522
CIP

British Library Cataloguing in Publication Data available

Warning and Disclaimer

The views and opinions expressed in this book are solely the author's and should not be attributed to any person or organization.

This book is designed to provide information about applying data compression in digital systems. The information is provided on an *as is* basis, without warranty of any kind. Although every effort has been made by the author and publisher to make this book as complete and as accurate as possible, neither the author nor Chapman & Hall shall be liable to the purchaser or any person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly arising out of use of the information contained in this book.

Some of the clip art contained herein was obtained from CorelDraw™.

To order this or any other Chapman & Hall book, please contact **International Thomson Publishing, 7625 Empire Drive, Florence, KY 41042.** Phone: (606) 525-6600 or 1-800-842-3636. Fax: (606) 525-7778. e-mail: order@chaphall.com.

Preface

This book is about compressing data to make digital systems work more efficiently. According to the dictionary, when something is compressed, it is condensed, squeezed, constricted, or pressed together to fit into less space. Air is compressed for a variety of useful purposes. Businesses are downsized to make them more efficient. We pack our daily schedules tighter and tighter to accomplish more. Who has not crushed an empty soda can for recycling? Many different things can be compressed, including the data in computers, communications links, consumer-electronics gear, and all sizes and shapes of digital systems. Are you curious about how data compression squeezes the "air" out of digital bits? Would you like to know where it is used and, increasingly, why the marketplace demands it be used? Would you like to learn the right way to build data compression into your products? Then, this book is for you.

This exciting technology and its importance for current and future digital systems are explained in easy to understand terms. No previous knowledge of data compression is required because the necessary technical background is carefully developed. Neither is an extensive understanding of mathematics because there are few equations and important ideas are graphically illustrated. If you read any of the popular or professional monthly magazines that cover the latest advances in digital systems, your background is adequate. If you understand this preface, you are ready to tackle this book.

This book explains why and how data compression went from being an obscure technology, known by few, to one used every day by millions of people. Not long ago, only people with specialized skills in information theory, computer science, or communications knew of data compression, much less understood its workings or its importance. Today, almost everyone interested in the latest ad-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

vances in digital information technology knows something about data compression. Magazine and newspaper articles dealing with progress in consumer electronics and communications mention it frequently. Data compression is fast becoming, not just a household word, but a household servant. Everyone who views a direct broadcast satellite television program, uses a facsimile machine, or waits patiently while their personal computer downloads image-laden pages from the World Wide Web experiences the magic of data compression. It is all around us in computer-based products and services from many industries. Like computers and communications, the consumer-electronics, publishing, entertainment, and healthcare industries, to name a few, find data compression is now indispensable, not just to be competitive, but, often, to deliver practical products and services.

Anyone interested in the *digital revolution* now bringing sweeping changes in electronic information handling should buy and read this book. That includes purchasers, administrators, system managers, product and market analysts, consultants, manufacturers, sales personnel, planners, designers, engineers, programmers, and all who are involved in creating an ever-expanding universe of products and services that depend on data compression. Also included are technologists who know data compression well, students just learning about it, technically-oriented consumers experiencing its power, or anyone who simply wants to know more. All will discover how data compression is used in real-world applications and gain insight for its future application and development.

The material is organized in four parts: Marketplace, Algorithms, Applications, and Digital Systems. *Marketplace* explores the user requirements for data compression and the marketplace constraints and rules that effect its application. *Algorithms* provides the background needed to understand how the most important data compression algorithms operate. *Applications* contains a comprehensive industry-by-industry guide to modern data compression applications. *Digital Systems* describes important decisions and techniques for incorporating data compression in current and future digital information-handling systems.

The inspiration for this book came from a study sponsored by the IBM Academy of Technology. The Academy, whose members are key technical professionals drawn from all locations in the worldwide IBM family of companies, each year commissions studies on technical matters of importance to IBM. The reports that result from these studies are directed to two audiences, to the IBM technical community and to IBM management. Therefore, while focused on technology, Academy studies give equal treatment to the marketplace, application, system, and implementation issues influencing successful deployment of technology. It was my good fortune to be selected to lead an Academy study of data compression for digital computer systems and computer-based products. Going into that assignment, my qualifications were those of as an engineer who had applied data compression to midrange IBM business computers. I knew about data compression for business computers, but not of other applications, much less what data compression technology might be appropriate for those applications, and certainly

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

not of many factors relating to its deployment. Consequently, I needed to learn everything there is to know about data compression—fast!

In a frenzied effort to become a worthy study leader, I set out to gather information from within and outside the IBM Corporation. It was easy to find a wealth of sources that explain how data compression works. For those who are technically adept, there are books filled with the theory and equations for data compression algorithms and professional society journals that convey recent work. Within organizations devoted to digital system research and development, such as IBM, there are knowledgeable technologists who can explain the latest happenings. All are very helpful if you too are a technologist, at or near the forefront of research, or hope to become one. Similarly, there is a growing abundance of information for practitioners interested in using data compression products such as disk compression software for personal computers and video compression software for CD-ROM application development. Popular how-to-do-it books and magazines, along with conversations with fellow practitioners, are very helpful if your intent is to apply existing products. For technically-oriented consumers, or anyone interested in the societal impacts of data compression, the people and publications devoted to consumer electronics have a lot to offer. Unfortunately, it is but one area for data compression application and, while the discussions and publications are numerous, too often little is explained about data compression technology beyond the buzzwords. Perhaps the people who must invest the most time learning about it are those who design the products and services that use data compression. It was difficult to locate the kind of information needed to make design, development, and deployment decisions and even more difficult to unearth the marketplace factors that (as you will learn) have important influences on data compression.

After making a considerable investment of time, and with much help from many study participants, I did get the overview of data compression that I desperately needed. Out of this experience came my observation that there ought to be an easier way to get the whole picture. There should be a reference one could turn to for learning the vocabulary and the concepts of data compression. It should explain where modern-day data compression came from, what is happening today, and, equally important, what to expect in the future. This source should also provide a perspective on marketplace dynamics and how data compression influences what is taking place in computers, communications, or any of the other industries it touches. Conversely, it should explain how data compression technology is influenced by the marketplace. Finally, it should also speak to the issues system designers face when incorporating data compression in their products and services.

The search for a comprehensive introduction to data compression continued without much success while the Academy study was in progress. When the study was completed, it was time to write the final report, which we realized must include an introduction that would reach out to the report's diverse readers, many of whom would be learning about data compression for the first time. Having

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

no other choice, we wrote an introduction to data compression from scratch. Shortly after the Academy report was published, one warm summer day in 1994 I was mowing the lawn. Suddenly, it occurred to me: "Ah-ha, parts of the Academy report have the makings of an introductory book about data compression!" To be sure that it was more than an aberration of thought induced by heat stress, I consulted with Dr. Joan Mitchell, an IBM Researcher and coauthor of a recent book on data compression [Penn93], who agreed with my observation. To shorten a long story, as Academy reports are not distributed outside IBM, I asked for and received the Academy's permission to use material from its data compression report. As the saying goes, the rest is history, and you are now about to read the book that resulted.

And now a few words about what you will find in this book: Data compression is a very dynamic field. Many applications described in this book are still evolving, as is the data compression technology needed to support them. This is particularly true for compressing speech, audio, image, and video data where the development of new applications and new data compression technology goes hand-in-hand. Standards for applying data compression are still being created. New data compression methods are still being discovered that someday may be included in new standards. Consequently, what this book contains is a snapshot of important developments and events as they happened up to the time of writing (1996). By the time you read this, the picture for some applications, particularly in the fast-paced world of consumer electronics, surely will have changed. Thus, you should use this book as the beginning of your education on data compression. To keep up, you (and I) must continue to read and learn about this dynamic technology.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

1

Overview

Information, the computers to process it, and the communications infrastructure to transmit it are essential in today's global economy. Distance, time, even differences in spoken language are no longer barriers thanks to advances in computers and communications technology. Routinely, we accept as everyday events in our lives the acquisition and use of information from distant sources. Online credit card validation, international banking transactions, news reports from around the world, E-mail, or videoconferencing all have become commonplace. Some of us cannot remember a world without these services.

These advances in information handling are tangible results achieved by integrating computers and communications. In the 1970s and 1980s, computers learned to communicate, and communications networks learned to speak the language of digital computers. It was a time when the computer and communications industries converged on standard data representations, digital data representations. Today, combinations of data processing equipment (computers) and data communication (transmission and switching) equipment form integrated information systems that seamlessly handle data encoded in digital formats. Convergence continues in the 1990s as other industries discover the advantages of computer technology and digital data. Music on compact discs (CDs), books on CD-ROM, digitally enhanced motion pictures, products and services offered from video kiosks, and, soon, interactive television—these are but a few applications joining what might be called the digital revolution.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

1.1 Representation of Information

Information comes to us in many forms from many sources: text in books, numbers in reports, a person speaking, music, photographs, images, motion pictures, television, and computer-generated video are but a few examples. To *electronically* process, store, or communicate information, the facts, concepts, or instructions each information source conveys must be represented as data.

There are many ways to represent data. One choice is to use a representation that mimics the information itself. Some information is discrete: text and integer numbers take on discrete values (A, B, C, . . ., 0, 1, 2, . . .). For discrete information, a *digital* representation is the natural choice. Some information is continuous: speech, audio, video, and most information collected by data sensors come to us as continuously varying values. Speech and audio are ever-changing patterns of sound-pressure level amplitudes and frequencies. Video is a continuous sequence of image "snapshots" (frames) depicting lighting and color. For continuous information, where data take on a continuous range of values, an *analog* representation is the natural choice. Indeed, most early information-handling systems including telephone, radio, and television used analog.

Now, with the availability of low-cost VLSI (very large-scale integration) digital (computer) technology, it is becoming practical to represent both discrete and continuous information in digital form. Processing, storing, and transmitting data in digital format offers many advantages. Digital data is easily manipulated and digital allows the use of many signal processing and enhancement techniques not available in the analog world. Most all information processing operations—including data compression—are easier to do and more effective in digital. As for storage and transmission, unlike analog data, digital data can be stored and transmitted with essentially no degradation. Copies of digital audio or video can be identical to the original. A received signal can be identical to the original signal when digital transmission links are used. Not only are digital signals less susceptible to noise than analog signals, but errors that occur during storage or transmission are easily corrected.

Five common types of information—text, speech, audio, image, and video—are shown in Figure 1.1. Each is digitally represented by doing the following: For each character of text, a bit pattern code is selected from a set of characters. Common character sets for computer systems include the 8-bit ASCII and EBCDIC character sets or, less common, the 16-bit DBCS (double byte character set) character set.[1] For speech and audio, digitized samples are extracted from the analog signal at regular intervals of time. Each sample typically contains from 4 to 20 bits; using more bits provides higher-quality reproduction. For example, telephone-quality speech uses 8-bit samples, whereas music-oriented

[1] A *bit* (binary digit) that takes on values of 0 or 1 is the basic unit of information in digital systems. A *byte* is an 8-bit group used to represent a character or other information.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

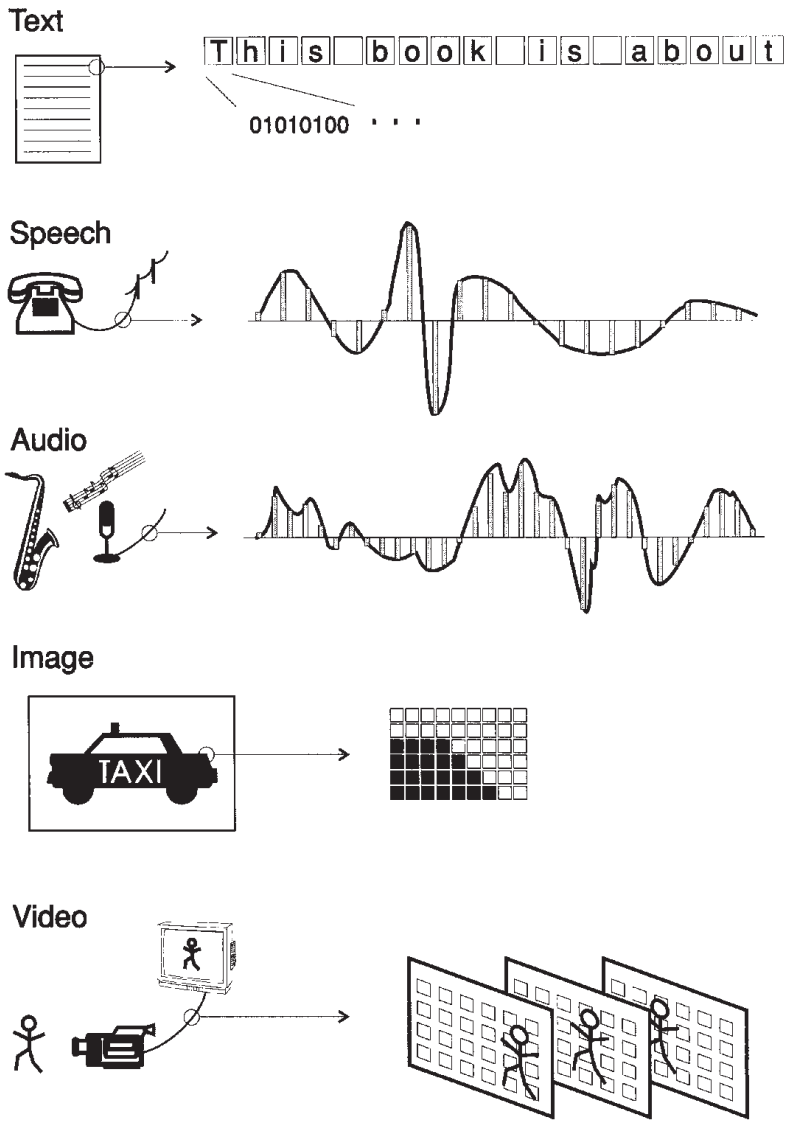


Figure 1.1 Digital representation.

compact discs use 16-bit samples. For images (and video too), the image is scanned in a predefined pattern, say left-to-right, top-to-bottom, and samples are collected at regular intervals. These samples are called *pixels* (picture elements) for obvious reasons.[2] They are organized as a two-dimensional array of points. Whereas a single pixel array is needed to represent a still image, for video a time-ordered sequence of pixel arrays must be used to capture motion within a

[2] Some authors use the term *pels*.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

televised scene. Here, each pixel array represents a digitized image captured from the video signal. For both images and video, the number of bits per pixel varies, typically, from 1 to 24; using more bits provides a higher-quality reproduction of complex images and scenes. Whereas FAX (facsimile) image transmissions of black-and-white documents require only 1-bit samples, high-resolution color images may require 24-bit samples. For video, color videoconferences are captured with 12-bit samples, but higher-quality transmissions may use 16-bit or 24-bit samples.

1.2 Data Compression—What Is It and Why Is It Needed

Data compression is nothing more—or less—than efficient coding designed to correct the overrepresentation that occurs in digital data-handling systems.[3] All of the representation systems described in Section 1.1 share two common characteristics. First, each imposes a regular, fixed structure on the data. Second, bits can be and usually are wasted because of this regularity. Each character is 8 or 16 bits. Each speech or audio sample is a fixed number of bits collected at regular intervals of time. Each video or image pixel is a fixed number of bits collected at regular points within a scene. Make no mistake about it, regularity simplifies the process of converting information to digital data, and it makes electronic systems easier to build; but most of the data collected does not contain much information, at least not as much as the digital bits would allow.

To make this clear, consider what studies of linguistics and information theory tell us: By accounting for the frequency of occurrence of individual characters, text usually can be represented by not more than four and often even fewer bits per character. Yet, 8 or 16 bits are used. In digital speech and audio systems, the number of bits per sample and the sampling rate are chosen to represent the most rapidly changing signals with maximum fidelity. However, as seen in Figure 1.1, often there are intervals of time when not much is happening and far fewer samples (or bits) would be sufficient. Yet, the maximum sampling and digital bit rates are always used. In image and video systems, similar phenomena occur. The image and video digitization process is designed to capture changes in the scene with some prespecified fidelity. However, some parts of the image may have the same light intensity or color and, consequently, all the pixels within a spatial region will have similar digital values. In video, a sequence of frames will not change much if there is little movement in the scene and pixels within a region will not change much from frame to frame. Yet, the maximum prespecified space and time pixel sampling rates are always used.

What data compression strives to do is find innovative ways to represent

[3] Other names for data compression are source coding (an information theory term), data compaction, and data reduction (when there is information loss).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

information using as few bits as possible for storage and transmission. The two main techniques data compression uses are redundancy reduction and, sometimes, intelligent removal of unusable information. When too many bits are used to represent data, those that are excessive can be discarded. When information is collected that cannot or will not be used, it can be discarded. Although these may seem simple tasks, they are not. In reality, data compression is complex; complex enough that people write books about the subject!

In digital systems, there are three reasons to use data compression:

- Storage efficiency
- Transmission bandwidth conservation
- Transmission time reduction

Both the storage capacity and the transmission bandwidth available for digitized data have grown and continue to grow at remarkable rates. But the amount of data to be stored and transmitted is growing even faster. Existing business applications require ever-larger, often geographically distributed databases. New applications—particularly those for home and office that use digitized speech, audio, image, and video—require exponentially increasing amounts of storage and bandwidth. Meeting the needs of end users, who are always looking to do more things in less time, is rarely easy. Digital system designers know there is never enough storage, bandwidth, or time, and none of them are cheap enough to waste. Data compression allows each of these precious commodities to be used more efficiently, creating more opportunity for innovative new products and services. This is accomplished by adding software and, sometimes, hardware for data compression processing. The result, from a system cost perspective, is a more balanced system. As Figure 1.2 indicates, modest increases in processing cost usually are offset by greatly reduced costs for storage and bandwidth.

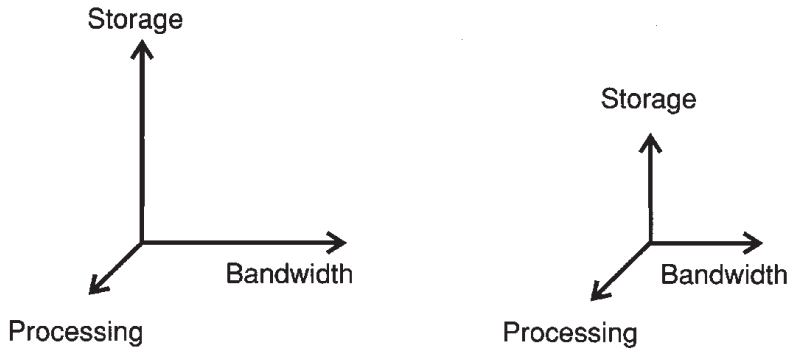


Figure 1.2 System costs.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

1.3 Historical Origins of Data Compression

The need to efficiently represent information, particularly text, has been with us since man learned to write. Stenography, the art of writing in shorthand can be traced back to the first century B.C. when a form of shorthand was used to record the speeches of the Roman orator Cicero. Shorthand is a system that uses simple strokes, abbreviations, or symbols to represent letters of the alphabet or even words and phrases. It is a form of data compression for writing. Over the centuries, various shorthand systems have been introduced. With the invention of electronic recording equipment, the need to write as fast as someone speaks has all but disappeared, except for court reporters.

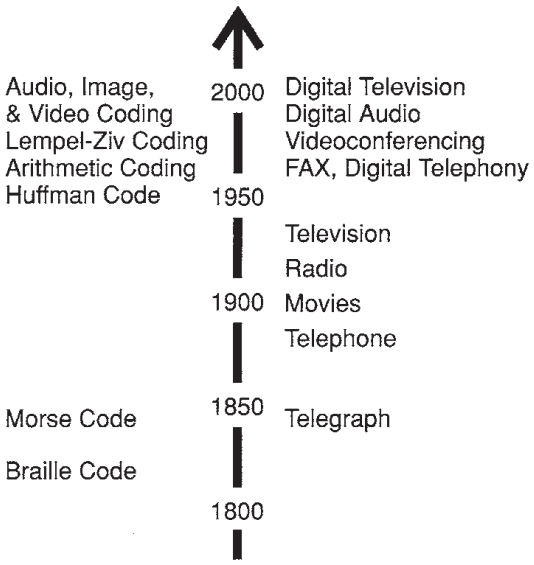


Figure 1.3 Data compression timeline.

More recent data compression developments are shown in Figure 1.3. Two important events occurred in the 1800s. In 1843, S.F.B. Morse developed an efficient code consisting of dots, dashes, and spaces to allow transmitting messages electrically by telegraph, first by wire and later by wireless radio. Morse code uses a basic form of data compression. It assigns short easy-to-key codes to E, I, and other frequently transmitted letters, and longer codes to Q, Z, and other infrequently occurring letters. A few years earlier, in 1829, the invention of Braille code by Louis Braille created a system of writing for blind persons. Braille code represents the letters of the alphabet by combinations of raised dots that are read by touch. Its extensions include music, mathematical, and scientific notation. Like Morse code, it uses shorter representations for common characters. It also includes a second form of data compression, "contractions," that represent common words with two or three Braille characters [Bell90, Witt94].

The ideas of Morse and Braille are the basis for many modern-day text compres-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

sion schemes. Beginning in the 1950s, information theory and probability concepts formed the backdrop for creating a class of statistical data compression algorithms. As with Morse code, these algorithms use variable-length bit patterns to encode individual symbols based on their frequency of occurrence. Two statistical codes, Huffman coding [Huff52] and arithmetic coding [Riss79], are now in widespread usage. In the 1970s, a second class of data compression algorithms, dictionary algorithms, was formalized by the work of Lempel and Ziv [Ziv77, Ziv78]. Dictionary algorithms encode, as does Braille code, sequences of symbols into shorter codes that are found in a "dictionary."

Data compression for other forms of information has a much shorter history. Before the 1950s, systems for transmitting and storing speech, audio, image, and video information primarily used analog technology. Analog signal (data) compression techniques were employed in some of these systems, particularly for speech transmission over telephone lines. However, analog signals and analog technology greatly limit what can be accomplished with compression.[4] It was not until the 1950s that technology advances allowed exploring digital systems for speech, audio, image, and video. Only then did the need for and power of data compression—digital data compression—became clear. In the 1970s, telephone networks began to go digital. It was at this time when, for transmission efficiency, the telecommunications industry began developing digital coding and compression techniques to allow packing more voice channels into digital telephone links.

Another early application was FAX transmission of images, an idea stretching back to the mid-1800s. In the early 1980s, FAX became a reality with the development of digital technology and standards for compressing and transmitting black-and-white, bi-level images over ordinary analog telephone lines. Data compression is essential for transmitting an FAX image in a reasonable amount of time. But image compression applications did not end with FAX; by the mid-1980s, business and consumer applications for high-quality, digitized photographs and all types of continuous-tone images began to appear. These applications were supported by the development of a new standard for compressing continuous-tone, still images, and an improved standard for compressing bi-level images.

Audio and video applications are the most recent to be invaded by data compression technology. In the 1980s, the digital revolution washed over the consumer audio marketplace and digital audio compression techniques began to appear. In this era, data compression also enabled the production of reasonably

[4] Analog signal compression, which is not what this book is about, takes several forms. One is to improve transmissions over noisy channels where, for instance, an analog electrical signal representing light or sound is compressed by applying gain control to increase small values and limit large values; this is somewhat akin to the way that humans handle wide variations in the light levels reaching their eyes or the sound levels impinging on their ears. Analog compression is also used to conserve bandwidth, particularly for speech, where the original analog signal is analyzed and processed into a few essential frequency components.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

priced digital videoconferencing gear for businesses. Now, in the 1990s, broadcast television and all forms of video are going digital and here, more than for any other type of information, data compression is essential because video generates immense amounts of data. Standards for compressing audio and video are, depending on the application, in various states of development.

1.4 Marketplace Trends Relating to Data Compression

Three important trends are enabling affordable data compression that will change substantially how digital systems handle text, speech, audio, image, and video data, and that will nurture the creation of many new products and services. These trends are as follows:

- The development of effective techniques for compressing all types of data
- The commercial availability of fast, low-cost, single-chip, microprocessors, DSPs (digital signal processors), and VLSI hardware data compressors
- The convergence of the computer, communications, consumer electronics, publishing, entertainment, and other industries on standards for digital data representation

These trends have long-term impacts that are not limited to just products and services and what they do or how they operate. The industries themselves are affected, for converging on digital data changes the ground rules for doing business. The common underpinning of digitized text, speech, audio, image, and video data opens new opportunities for each industry. It allows sharing of technology, tools, processes, and ideas across the industries. This, in some cases, blurs the boundaries between computer, communications, consumer electronics, publishing, entertainment, and other applications, but it allows each application and industry to leverage the strengths of the others. For the computer industry, as an example, this means a wider application of computing technology, access to better multimedia computing application technology, and access to low-cost, low-power VLSI technology (such as data compression chips) developed for mass-consumption, consumer-electronics products. For the communications industry, this means access to computing technology (RISC microprocessors for set-top boxes, disk[5] storage devices for servers, and more), new opportunities to transport data such as for the entertainment industry, and opportunities to enable new applications such as interactive video. A similar list exists for each of the other industries caught up in digital convergence.

All of the converging industries are moving rapidly to define the technology, applications, and products that will capture the promised riches of digital data.

[5] In this book, we will follow industry convention and use *disk* for magnetic disk storage and *disc* for any disc that is read using a light beam.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Although most of the focus is on video, the opportunities for innovative applications, products, and services based on text, speech, audio, or images are not to be overlooked.

1.5 Data Compression Algorithms

As students of this subject soon learn, the variety of data compression algorithms is almost endless, ranging from ad hoc to highly formalized, employing a diversity of theories and mathematical techniques from various branches of science. To promote understanding, most authors choose to group data compression algorithms into a few classes based on the techniques used to do compression[6] [Lync85, Netr95]. Unfortunately, there is no one standard classification scheme. Adding to the confusion, many algorithms combine techniques and do not fit neatly into any one class. This book takes a different approach: Data compression algorithms are organized according to the data they compress, and all the algorithms that apply to each type of data are examined as a group.

As to classifying the data itself, there are various views of the universe of data encountered in digital systems. The communications industry classifies data in the following ways: Services are provided for voice, data, image, and video (where "data" means the symbols found on a terminal keyboard). Communications deals with information sources that either produce digital data (which takes on discrete values) or that produce analog data (which takes on continuous values in some interval) [Stal94A, Stal94B]. Sometimes the terms discrete media (such as text and digital images) and continuous media (such as audio and video) are used when describing the timing requirements for the (data transport) services provided [Gemm95, Lieb95]. Within the computer industry, all data is considered to be digital (because that is how it is processed), and data is referred to by its type, such as text, speech, audio, image, video, and so on. The consumer-electronics, publishing, and entertainment industries also follow this convention, at least when dealing with digital data.

For this book, we have chosen the unifying view of data described in [Arps79]. As shown in Table 1.1, there is *symbolic data*, where a letter, a figure, or other character or mark, or a combination of letters or the like, represents something that a human would recognize. The alphanumeric and special characters in the EBCDIC, ASCII, and DBCS codes for computer systems are familiar examples of symbolic data. There is also *diffuse data*, where the meaning and structural properties have not yet been extracted (i.e., remain widely spread, scattered, or dispersed—diffused) and not yet converted to something a human would recog-

[6] The term *algorithm* refers to a collection of techniques used for any particular data compression application. In contrast, a data compression *technique* is a particular method of compression, usable by itself, but likely to be used in combination with other techniques [Luth91].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 1.1 Data types.

Symbolic	Diffuse
Character text	Speech
Numeric	Audio
Computer program code	Image
Graphics and Icons	• Binary (bi-level)
	• Gray scale
	• Color
	Video
	• Black-and-white
	• Color

nize. Speech, audio, image, and video information, when represented as digital samples from the original analog data, are examples of diffuse data.

Data compression algorithms for symbolic and diffuse data operate differently. To compress symbolic data, only *lossless* data compression applies, because an exact, bit-identical reproduction of the original data usually is required. For business, computer programming, database, electronic mail, and scientific applications, which deal with exact representations of information, the loss of even a single bit in character text, numeric data, or computer programs is unacceptable.

Compressing diffuse data may involve *lossy* data compression that throws away bits which are not needed for reproducing speech, audio, image, and video. True, some information may be lost, but humans usually are the recipients, and their auditory and visual limits make lossy data compression acceptable for many applications. Also, often in succeeding phases, special forms of lossless data compression are applied for more efficient storage or transmission of compressed diffuse data.

How much a data collection will compress is a property of the data. It also depends on the data compression algorithm employed and, moreover, on how much and what kind of information loss is acceptable. For symbolic data, where no information loss is acceptable, *compression ratios* of 2:1 or 3:1 are typical.[7] In contrast, diffuse data can compress up to 100:1 or more depending on the type of data, the effectiveness of the compression algorithms, and the information loss acceptable.

Data compression does not come free; there is a cost. It trades processing

[7] Compression ratio is a measure of the amount of compression achieved. It is computed by dividing the original number of bits or bytes by the number of bits or bytes remaining after data compression is applied.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

power and processing time for storage capacity, transmission bandwidth, or transmission time. How quickly data can be compressed is determined by the data, the data compression algorithms, and the processor speed. Data compression algorithms are compute-intensive, and performance (or more properly the lack of it) has sometimes restricted the opportunities for data compression. With the availability of high-performance microprocessors, data compression for many applications can be done in software. Sometimes more performance is required, and for those applications single-chip, high-performance DSPs and VLSI hardware data compressors are available. This range of implementations provides powerful data compression algorithms at a reasonable cost that execute fast enough to process the main dataflows of almost all applications in almost all digital systems.

1.6 Applications for Data Compression

Some applications for data compression from five key industries are show in Table 1.2.[8] Many text-based applications in the computer and communications industries are well established. In contrast, applications dealing with speech, audio, image, and video data are just emerging or are still being defined. This diversity reflects the state of the art for data compression technology. Compression for digitized text, as noted earlier, has decades of development history, and most computer-based applications that handle text data are fully committed to data compression. Compression for digitized speech is widely deployed, too, but mostly in the telecommunications industry and not in others. Compression for digitized audio, image, and video data is, by comparison, relatively new for everyone. In many instances, the compression technology is still being defined along with the applications that will use it.

1.7 Integrating Data Compression in Digital Systems

The advantages of using data compression for storage efficiency, transmission bandwidth conservation, and transmission time reduction are obvious. What sometimes is not so obvious is how to integrate data compression into new or existing digital systems. First, we need to define what a *digital system* is and does. A digital system is a collection of elements for processing digital information. In this book, we are considering digital systems that include data compression facilities. Figure 1.4 shows a simplified representation of a digital system that

[8] These industries and applications are examined in detail later in this text. In addition, some brief examples will be provided to show how data compression is applied in other industries.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Table 1.2 Data compression applications.

Computers	Communications	Consumer Electronics	Publishing	Entertainment
Tape	Voice • Telephones • Cellular telephony	Digital audio • CD • Tape • Home theater	CD-ROM multimedia publications	Media • Production • Distribution • Storage and archiving
Magnetic disc	Data communications networks • WAN • LAN	Digital video • HDTVs • Set-top boxes • Digital VCRs • Digital video discs	Online multimedia publications	
Optical disc	FAX	Digital photography • Digital cameras • Photo CD		
Solid-state storage	Teleconferencing • Audiographics • Videophones • Videoconferencing	Interactive multimedia set-top boxes • Education • Arcade games • Karaoke players		
Software distribution	Broadcasting • Terrestrial SDTV, HDTV • Satellite DBS • Cable TV • Telco video • Digital radio	Multimedia PCs		
Software execution • System software • Database • User data and programs	Telemetry	Multifunction office machines		
Chip and system interconnections		Digital speech products		
Servers • Text • Speech • Audio • Image • Video				

captures, encodes, and compresses data which it stores and/or transmits; it also decompresses, decodes, and displays or presents data to end users. A digital system can be as small as a hand-held audio/video product or as large as a worldwide computer network. Each digital system that uses compression performs the information processing steps shown in Figure 1.4. Digital systems obviously do more than manipulate compressed data; those information processing steps have not been shown.

It is important to know that the compression process, the compressed data it produces, and the decompression process interacts with the physical system and with all other logical functions performed by the system software. These interactions may not affect system operation, or how users perceive the system, but all too often they do, as examples throughout this book will show. Consequently, when data compression is driven into the main dataflows of digital systems, many technical, operational, and marketplace issues arise and must be

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

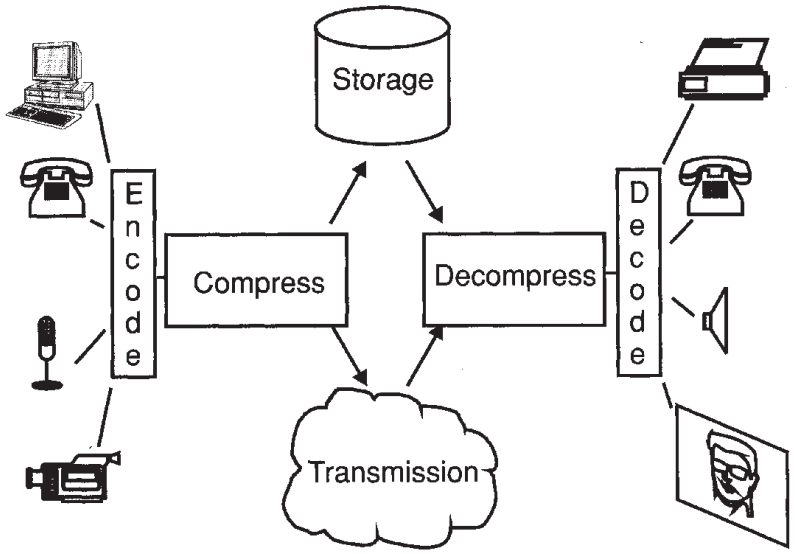


Figure 1.4 Integrating data compression in digital systems.

addressed. Often these issues override what many involved with data compression consider as the most important issue, selecting the data compression algorithm. Although assuredly an important step, algorithm selection is often the easiest decision facing digital system designers because the marketplace has made the choice for them. This is particularly true in a world of increasingly "open systems" (a computer industry term meaning that components and products of various origins can interoperate), where the choice of compression algorithm often is mandated by agreed-to standards.

A more difficult technical decision in digital system design is where to place the compression and decompression functions. As later chapters of this book will clarify, there are solutions that are either technically correct or market-driven but usually not both. Prudent design tells us to place data compression close to the producer or source of data, then compress the data and do not decompress it until it is delivered to the end user. This, the optimum system design for compression, provides the maximum storage and transmission efficiency. But the marketplace sometimes dictates other choices, especially in complex systems consisting of many components, from different manufacturers, perhaps even from different industries.

For example, consider the client-server computer system shown in Figure 1.5.[9]

[9] Client-server computing was developed for LAN-based computer systems in the 1980s. Application programs on client computers communicate with server computers that provide services including access to information stored in databases. The processing workload in the clients is reduced at the expense of increased network traffic and increased work load on the servers [Laro94].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

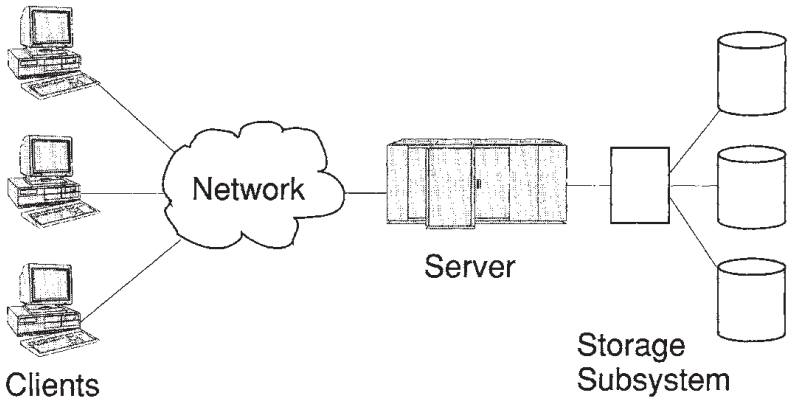


Figure 1.5 Client-server computer system.

End-user personal computer systems are connected to a telecommunications network that delivers data from a server—really another computer system—where the data is stored in a disk storage subsystem. Not only are each of these four components digital systems in their own right, but each has its own technical and marketing objectives, which, not surprisingly, may include compression optimized for the product or service. So, in complex digital systems, such as this example, it is not unusual that compression and decompression can take place several times between data capture and delivery to the end user.

Continuing to explore this example, we can appreciate that each component of the system needs to process the data in some way. Consider a simple data retrieval operation. The disk storage subsystem must package data for efficient storage and retrieval. The server needs to manage the data in a database. The telecommunications network needs to encode and possibly encrypt the data for transmission. Finally, the clients must prepare the data for presentation to end users. Complex systems, including this one, lead to a set of design decisions about handling compressed information. In removing redundancy from data (and sometimes information too), the compression process also destroys the fixed, regular storage structures that other digital data processing operations have come to expect. Fixed-length computer text records are no longer of fixed length. Depending on their content, images are a variable number of bits or bytes with no fixed storage structure corresponding to rows and columns of pixels. Fixed-rate video bit streams become variable rate, with each frame allocated a variable number of bits depending on the video content. How to process compressed data—short of decompressing, processing, and recompressing it in each component of a complex digital system—is a subject explored in later chapters of this book.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

1.8 Summary

In this chapter we learned the following:

- The computer, telecommunications, consumer-electronics, publishing, entertainment, and other industries are converging on digital data.
- Digital data-handling systems overrepresent data, using more bits than required.
- Data compression employs innovative ways to represent data, using as few bits as possible for storage and transmission.
- The history of data compression traces back to the 1800s and much earlier times.
- Advances in data compression algorithms, VLSI technology, and standards for digital data representation are enablers for new products and services that take advantage of data compression.
- Digital data can be classified as symbolic (text, etc.) or diffuse (speech, audio, image, video, etc.).
- Lossless compression is applicable to symbolic data; lossless and lossy compression are applicable to diffuse data.
- Lossless data compression removes redundancy; lossy compression removes unneeded information.
- Symbolic data may compress by 2:1 or 3:1 whereas some types of diffuse data may compress by 100:1 or more.
- Many applications already use data compression and many more will in the future.
- Important digital system design decisions include where to compress and how to handle compressed data.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Part I—Marketplace

This first part of the book explores what users of digital systems want and need from data compression. Also, it explores the marketplace constraints and rules that affect data compression.

Chapter 2 provides an overview of the requirements that users of digital systems have for data compression technology. The chapter begins with trends for digital systems and their applications. Next, the role of data compression for delivering multimedia information is described. Then user expectations for data compression are examined. The chapter concludes with a short history of successful applications of data compression.

Chapter 3 continues to expand the discussion of marketplace factors that are important influences on data compression technology. First, industry convergence on digital data representation and its effect on data compression technology is examined. Then standards for data compression and their increasing importance in a world of open systems are analyzed. This is followed by a discussion of data compression for closed processes, situations where data compression standards are not all-important. The chapter concludes with a look at the restrictions on using proprietary, patented, and shared data compression technology.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

2

Data Compression in the Marketplace

Table 1.2 in Chapter 1 presents an extensive list of applications that use data compression to store or transmit data more efficiently or to transmit it in less time. This chapter examines what the end users of those applications require, expect, and will get from data compression. It explores why data compression is *the* enabler for new, innovative products and services.

2.1 Digital Systems and Applications Trends

Processors, storage (electronic, magnetic, and optical), communications channels, and I/O devices—along with software—are the essential technologies for building computers and all systems that handle digital information. For several decades, each of these technologies have grown in capacity and performance while shrinking in size and/or cost. The good news is these trends are predicted to continue throughout the 1990s [Cuta90, Lewi94]. The bad news is technology advances alone are not sufficient to enable many new applications, particularly those that use video data. "Smart" processing techniques, including data compression, are needed too. This section looks at some important trends in digital systems and their applications, trends that accelerate the need for data compression.

Shrinking computers and expanding markets

The single-chip microprocessor has forever changed the computing industry. Computing tasks once confined to large, expensive computers now are done on

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

the desktop. Computing costs have dropped by orders of magnitude, and computers are now mass-marketed products. In 1994, some 16 million personal computers, about the same as the number of automobiles, were sold in the United States, and nearly 50 million were sold worldwide [Juli95]. In the United States, almost half were sold as home computers [Arms94]. At the other end of the scale, microprocessors are increasingly the building blocks for large-scale computing systems where they are organized into teams and operate in parallel [Kher95]. The microprocessor has revolutionized other industries too. Automobiles, industrial equipment, communications gear, microwave ovens, stereos, VCRs, and a variety of home electronics products—all are controlled by microprocessors.

Microprocessors are becoming more powerful too. A frequently cited statistic is that the speed of Intel™ microprocessors has improved by nearly 50% each year for the past two decades. As shown in Figure 2.1, the PentiumPro™ chip

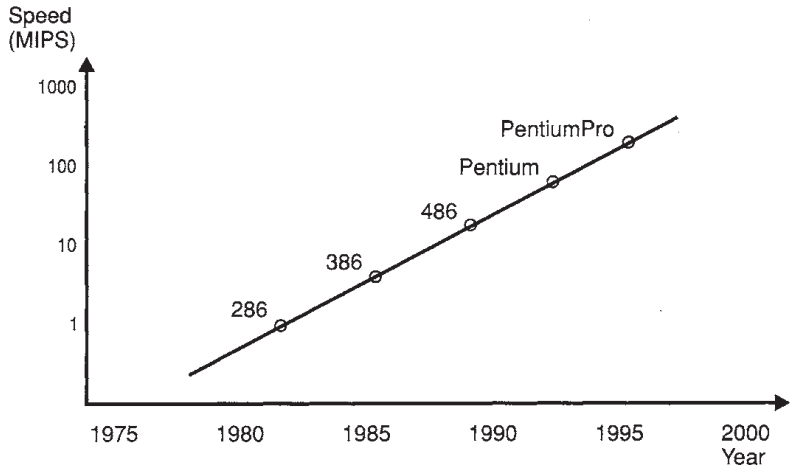


Figure 2.1 Intel microprocessors.

introduced in 1995 is some 250 times faster than its counterpart from the early 1980s [Hof95].[1] Every 18 months or so, the performance of silicon doubles at no increase in cost, thanks to improvements in chip-making technology. Smaller line widths make the circuitry more compact, which, in turn, makes the chips run faster (fewer nanoseconds per cycle) and allows denser chips (more logic circuits per chip). With each new chip generation, designers have given microprocessors more powerful dataflows by using wider data paths, more registers, parallel instruction execution, and other techniques to make them run faster. As a result, the instructions for manipulating text and numbers—the work of traditional processors—require less time, but there is some evidence that future improvements must take a different direction.

[1] MIPS (millions of instructions per second) is a measure of processor speed.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

New applications for microprocessors such as in set-top cable boxes, HDTV (high-definition television) receivers, a variety of digital audio/video consumer-electronics products, and especially multimedia computers have growing needs for processing diffuse data. All spend less time doing text and numbers and more time with graphics, images, and handling real-time speech, audio, and video data. Consequently, these products need a microprocessor that can quickly process both symbolic and diffuse data. However, even the fastest microprocessors are not yet fast enough to do all graphics operations or image and video processing—particularly video compression—in software.

One solution is to continue to use separate chips for programmable DSPs, specialized data compressors, graphics accelerators, and video processors (which do data compression too). For products that do not need microprocessor programmability, this is an effective solution. But for personal computers, it is an expensive solution, especially when, as in present-day personal computers, those chips are mounted on separate boards. A more cost-effective solution for personal computers, one that some believe will lead to the microprocessor of the future, is to build a specialized multimedia microprocessor. This new type of microprocessor, as shown in Figure 2.2, uses the increasing number of logic circuits on future-

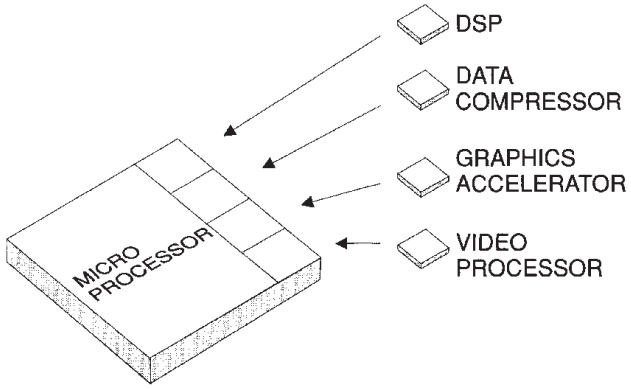


Figure 2.2 Multimedia microprocessor.

generation VLSI chips to integrate the function of DSPs, data compressors, graphics accelerators, and video processors. Not everyone agrees that a single microprocessor chip can or should do all these functions. However, following various tradepress reports of pending product announcements [Webb93, Wils94A, Wils95A, Wils95B, Slat95], a new generation of microprocessors with integrated multimedia functions has arrived. (Further discussion of this and many other multimedia processing options appears in Section 9.11.)

Communications for everyone

Communications are an integral part of modern society. The end-user demand for communications, which is fueled by widely available, low-cost personal

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

computers, telephones, and television sets, has spurred all branches of the communications industry. Thanks to intensive competition and technology advances like fiber optics and high-power satellites, the number of telecommunications networks, computer data networks, cable television systems, terrestrial broadcast channels, and satellite transponders continues to grow. All are providing increased bandwidth and more choice to end users. However, just as computing has become more personal, so too will communications become more personal. Much as everyone expects a private line for telephone conversations, users are looking forward to private communications channels that can deliver all types of data—including images and video—customized to meet their needs. All signs point to a future where communications bandwidth will remain a scarce (or expensive) commodity. In brief, the number of users continues to grow, the demand for private communications is increasing, and more data must be transmitted, which translates to more excess bits and more need for data compression.

As important as these communications activities and trends are, some would say they are just the warm-up acts for the headline event of the 1990s. The Information Superhighway, Infobahn, National Information Infrastructure (NII), and Global Information Infrastructure (GII): in the 1990s no one can avoid hearing these terms and everyone involved with communications is trying to define them. The idea is simple: There should be available an affordable information infrastructure that everyone can access, to communicate with whomever they choose, whenever they choose, to get whatever information they need, in whatever form it may be, no matter where that person or information may reside [Wein94, Nesd95]. Consider for a moment the telephone system: Anyone with a telephone can dial a few numbers, reach anyone else in the world who has a telephone, speak with them, send them a FAX, or exchange computer data. It is affordable, accessible, universal, and interactive. Now, take that idea and mix in a few new types of data, including images and video, and many established communications providers, politics, and government regulatory agencies. It is easy to appreciate why moving the NII from concept to definition to deployment is so complicated.

Figure 2.3 shows some of many users and activities proposed for the NII. It is not clear whether there will be a single infrastructure for the NII—a universal "information pipe" that homes and businesses can plug into—or whether there will be, as now, many different delivery channels [Wirb94]. Nor is it clear what services will be available. What is clear is that there are two loosely clustered groups of providers, each intent on defining a new information infrastructure [Pres93]. Each plans to use client-server computing technology to provide all types of information—text, voice, audio, images, video, and whatever else can be digitized and transported.

The first group believes the personal computer is the "information appliance" for today and the future [Arms94]. For this group, the Internet is the model for the NII. They may be correct. The Internet experienced tremendous growth in the early 1990s. Homes, businesses, universities, and government agencies; all rushed to be connected. Clients now number in the millions, and many host

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

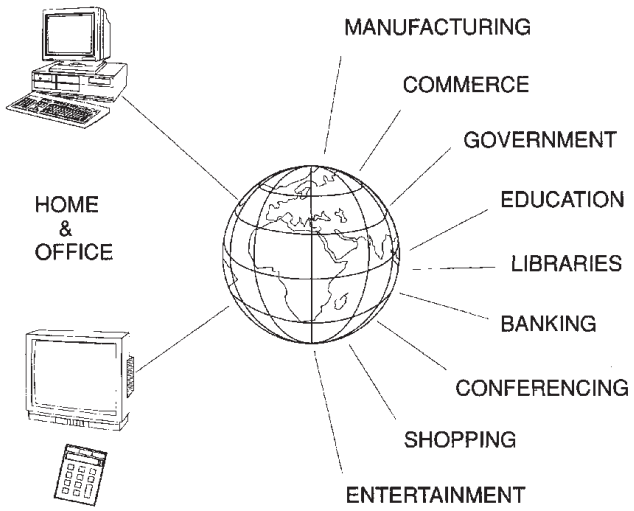


Figure 2.3 National Information Infrastructure (NII).

servers are online. What started as a computer network for research organizations in the United States has taken on a commercial flavor and has become global. Text-based communications have been augmented with graphics, voice, audio, images, and video. However, Internet bandwidth, although adequate for text and, perhaps, audio, severely limits transmissions of images and video—even compressed images and video.

The second group believes an *interactive* television receiver is the information appliance of the future. This device combines a standard-resolution television receiver or (in the future) an HDTV receiver and a set-top box. Actually a low-cost, microprocessor-driven computer, the set-top box is envisioned to be a product that provides functions ranging from channel selection and video decompression to controlling a camera for picking up and transmitting compressed video images back into the network. The knottiest question about interactive television is—what is the network? None of the existing communications systems—telephone networks, cable systems, nor broadcasting systems—meets all the requirements for fully-interactive television.

Increasing information bandwidth for the end user

Digital systems continue, as in the past, to undergo continuous change. Thanks to advances in image and video displays, graphics devices, audio gear, and all I/O devices, and the software that drives them, digital systems are providing more information bandwidth to end users. This trend in digital systems reflects the rising expectations end users have about electronic media. One constant in all that has gone before, and what is to come, is the growing demand for information. Once, end users were happy to speak via telephone and to be entertained

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

or informed by listening to radio or records; television and computers changed their expectations. End users expect to communicate and be entertained or informed by text, graphics, images, high-quality speech and audio, and video. Their demands have not gone unanswered. Table 2.1 provides some examples to show how the information bandwidth has increased between digital systems and the end users.

Table 2.1 More information bandwidth for end users.

Industry	Old Function	New Function
Computers	Black-and-white monitors	Color monitors
	Character text	Graphic user interfaces (GUIs) using text plus icons, graphs, and tables
		Speech and audio
		Images and video
Communications	Text-only	Computer networks carrying text data, audio, images, and video
	Voice-only	Telecommunications channels carrying voice, FAX, images, and video
Consumer electronics	Stereo audio	Surround sound 5.1 channel audio
	Television	HDTV
Publishing	Paper	Multimedia publications with text plus sound, images, and video
Entertainment	Movies on film	Digital special-effects movies
	Video games	Virtual reality

How does this relate to data compression? With each advance that increases the information bandwidth which a digital system can deliver to its end users, there are more bits to be stored and transmitted. Whatever form that information may take, whether video or something totally different, some of those bits will be excess bits that can be eliminated by data compression.

Interactive Multimedia

End users are intrigued by what interactive multimedia has to offer. It is more than just the bright colors and sound. End users can deal with information more efficiently when all their senses are involved, and they can use information more effectively by being interactive controllers of what they see and hear, not just passive observers and listeners. So goes the argument for why all digital systems

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

of the future will be interactive multimedia systems. Let us explore some reasons why interactive multimedia is attractive.

It began innocently enough—a few icons, a splash of color, a snippet of sound, a bouncing ball.[2] At first only computer wizards and, soon, kids were attracted. Then, gradually, ever so slowly, the rest of us learned what they already knew—computers are fun to use and educational too! Interactivity and multimedia (plus good programming!) makes them that way. Gone are the drab black-white or gray-green screens with only a few lines of text and a blinking cursor to tell us what is happening inside the box. In their place are colors, meaningful icons, and windows to help us organize and display information. In those windows, not only is there text, but graphics, animation, images, and video too. Computers talk and listen to us. Stereo speakers provide audio playback and a microphone allows entering commands or capturing whatever sounds we choose.

Multimedia computing mixes text, graphics, animation, voice, audio, images, and video in whatever way works best to capture and deliver information. Interactivity involves the end users in a two-way dialog with the computer. Although, in the future, multimedia information will be delivered by high-speed networks, such as those foreseen for the NII, now most multimedia information is stored locally on CD-ROMs. Multimedia CD-ROM software for reference, education, and entertainment became a billion-dollar industry in 1994 [Cort95, Juli95]. Judging from sales figures, 1994 can be marked as the year of the multimedia computer. In that year, at least in the home computer market, machines without multimedia capability piled up in warehouses while those with CD-ROM players, sound cards, and speakers moved briskly off dealers' shelves.

Interactive multimedia is slowly finding its way into business applications too. Visions of workers playing video games or multimedia-retrofitting accounting and other traditional applications do not stir much interest, but new applications do. Desktop videoconferencing, video E-mail, and information kiosks are some examples. Multimedia has already established itself as an effective, highly productive tool for selling complex products, whereas a multimedia-equipped laptop computer (see Figure 2.4) can produce a dazzling customer presentation [Howa95, Moha95].

In the home market, personal computers are not the only vehicles for delivering multimedia information. The consumer-electronics industry continues to pump out hand-held video games and an expanding collection of products based on CD-ROMs. However, some believe even bigger opportunities exist for television-based products. In the United States, some 98% of homes have television, whereas in 1994, only about one-third of those homes had computers. Home computers are more complex and costly that television, and, therefore, they are not projected to ever achieve the same level of penetration [Arms94]. The problem with television—plain old television—is that it is a broadcast medium without infrastructure

[2] Remember Pong,™ from Atari™, circa 1972?

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

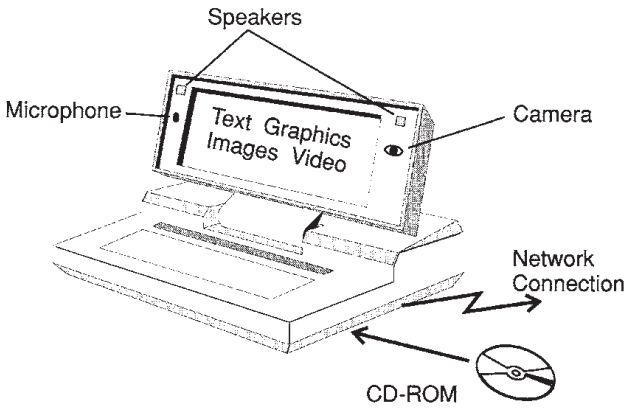


Figure 2.4 Multimedia laptop computer.

for two-way, interactive communications. Consequently, television cannot offer end users the full multimedia experience that computers provide. Chapters 8 and 9 describe the plans for making television interactive, but the real question is, once it is interactive, what multimedia-based services should it deliver? Suffice it to say that what consumers will want to do with interactive television probably are not the same things they already are doing with multimedia computers.

2.2 The Role of Data Compression

Let us now examine the role of data compression in delivering multimedia information. Even the most effective compression that it is practical to deploy today cannot overcome all the storage and bandwidth demands of text, graphics, speech, audio, image, and video data. This can be illustrated by a simple multimedia application. Our "application" presents the ideas on one page of this book using different presentation media ranging from text on paper to HDTV with 5.1-channel CD-quality audio. Figure 2.5 shows the storage required (assuming a single FAX page, image, or slide, and 5-minute audio or video presentations). Figure 2.6 shows the bandwidth required (for compressed data). For detailed information about channel bandwidth, and the storage and bandwidth required by various media used in this application, see Appendices 2A and 2B at the end of this chapter.

One may choose to change the presentation media, the presentation quality, or the length of presentation to best convey the ideas on one page of this book. But whatever your choice, the storage for uncompressed audio, images, and video exceeds by one or more orders of magnitude that needed for text. Also the transmission bandwidth required for transmitting uncompressed audio, images, and video would overload all available networks and broadcast channels.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

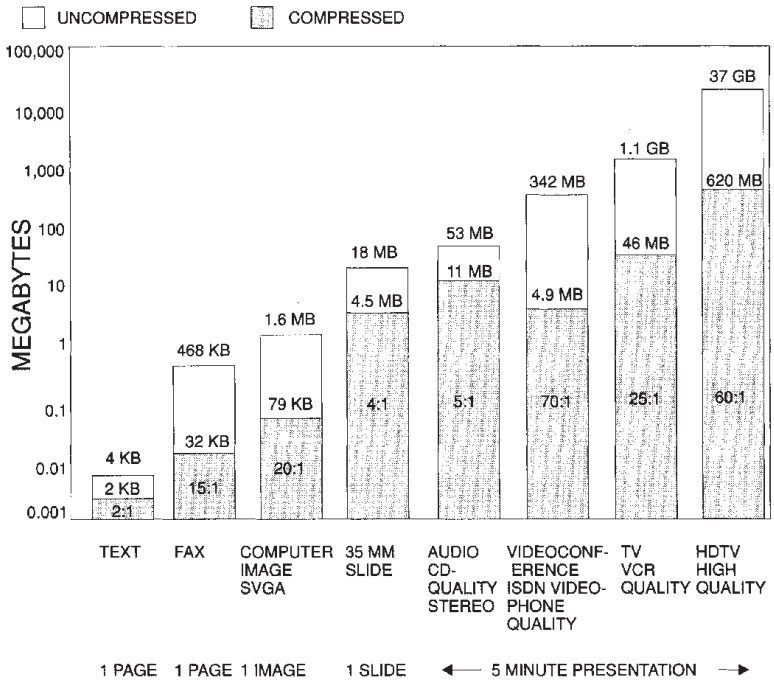


Figure 2.5 Storage required to present the ideas on a typical page of this book.

With data compression, the situation is more manageable—except for video. Multimedia applications that use video, even compressed video, stress the storage capacity of personal computers (the clients) and larger computers (the servers) and stress the bandwidth of all but the fastest available networks and broadcast channels. Consequently, the multimedia applications that have appeared do not deliver all that interactive multimedia can offer. If video storage is required, CD-ROM is the only economical storage media currently available for personal computers. Because it is read-only, authoring applications like the one above, which require writable storage, are problematical. If networking is required, high-bandwidth, two-way networks to connect clients and servers are still expensive and often unavailable. This has slowed the introduction of networked, interactive multimedia applications. To live within storage and bandwidth limitations, most multimedia applications for personal computers compromise video quality. Small video windows, low frame rates, and limited color depth are common. All are obvious and annoying limitations.

What does the future hold? Both home and business multimedia applications will benefit from advances in storage and networking technology. Storage density and capacity will continue to increase for both read-only and read-write storage with advances in optical and magnetic technology. High-density optical storage will likely continue to be the media of choice for video storage. Network band-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

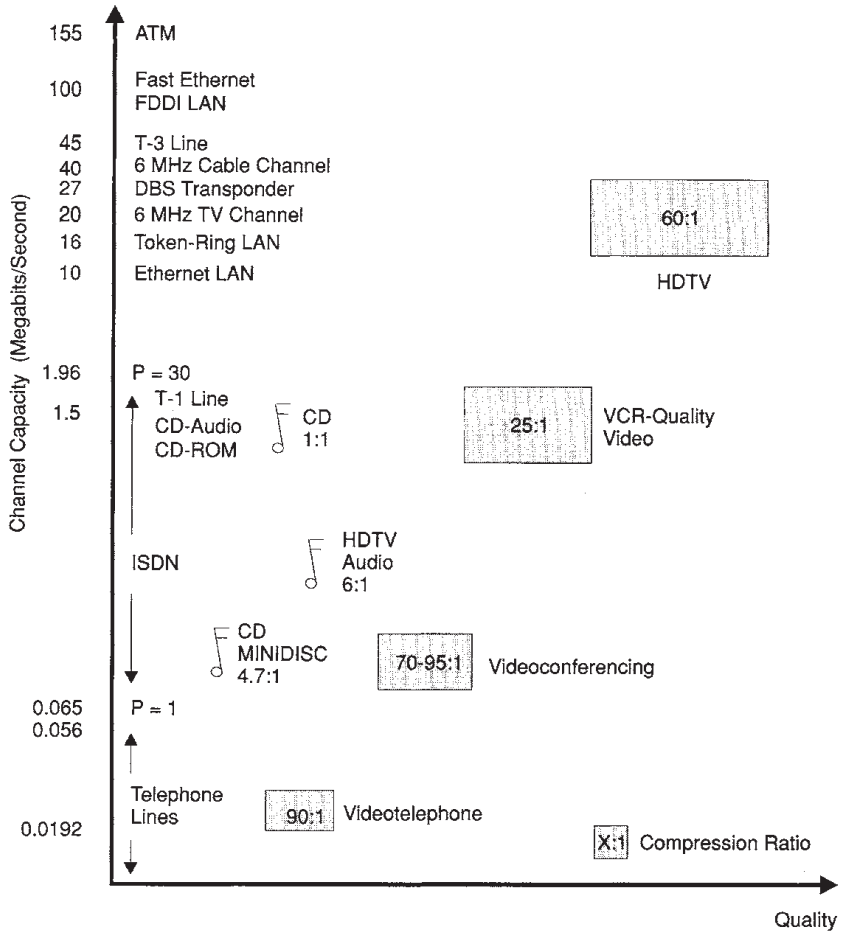


Figure 2.6 Quality and channel-capacity relationships for various media.

width, cost, and availability will continue to improve with increasing deployment of optical fiber. But even with these improvements, it will not be economical to store or transmit uncompressed data.

The consumer-electronics industry will exert an even stronger influence on the future of multimedia. Consumers, the end users, are accustomed to 30 frames-per-second video and ever-better speech, audio, and image quality. They expect multiple simultaneous image viewing options, special control and processing capabilities, and all the other capabilities provided by TVs and VCRs. It follows that consumers are likely to demand those same capabilities for the multimedia products in their homes and businesses too. If true, computer manufacturers, communications providers, and all other providers of multimedia products and services must meet the norms of the consumer-electronics industry for performance, function, and data quality.

It would be comforting to think data compression is the magic bullet that erases

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

all the storage capacity and transmission bandwidth problems that multimedia information creates—that with a little more invention, time, and dollars these will be past issues. But this is not true. Throughout the 1990s, data compression technology will improve for all types of data. Data compression algorithms that are considered leading edge today will give way to newer algorithms that provide more compression, more quality, and faster operation, causing old standards to be revised or give way to new standards. However, although it is always hazardous to predict the future, experts and observers do not foresee breakthroughs that will increase compression ratios by orders of magnitude. For symbolic data, such as text, decidedly modest improvements are forecast. For diffuse data, image and video in particular, expect perhaps an order of magnitude in general-purpose and somewhat more in special-purpose applications, but not orders of magnitude [Habi92]. Economical delivery of full-function solutions for multimedia applications can only be assured when data compression is teamed with advances in VLSI technology, storage capacity, and communications bandwidth—plus lower costs for all.

2.3 End-User Expectations for Data Compression

End-user expectations for data compression are simple but demanding. Consider the following analogy: Automobiles operate more efficiently with fuel injection, stop more reliably with antilock brakes, and are more crash worthy with air bags. Each of these technologies contributes, in the eyes of the consumer, to making a better automobile. However, none of these technologies could address the mass market for automobiles until it was proven effective, demonstrated to be reliable, unobtrusively integrated, and, when integrated, still yield an automobile that consumers could afford.

So it is also for data compression, a technology for making better digital systems. End users expect data compression to be nothing short of totally invisible and invincible. They do not want to understand how it works, manage it, or experience anything except its positive effects; specifically:

- Data compression must be completely integrated, not an add-on or afterthought visible to end users.
- Data compression must not change how end users perceive digital system operation; there can be no slowdowns, no performance glitches, no changes in function.
- Data compression must predictably compress data with no surprises, no exceptions.
- Data compression must not compromise data in any way observable to end users.
- Data compression must be reliable; data should not be lost, but if it is, end users should not be involved in its recovery.

End users also expect that data compression will make products and services more affordable. This does not mean that the extra processing power or added

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

hardware logic for data compression always must be provided at no additional cost. Rather, end users must recognize how the savings they realize from reduced storage or transmission costs (or time) reduces the total cost of the product or service.

Finally, end users expect that, like any new technology, data compression will enable new products and services that provide added value, other than just pure economics. This can mean products that are smaller, lighter, or faster. Also it can mean products and services that contain new functions. Sometimes, as the marketplace has demonstrated, end users view these as advances, and they are willing to pay for them.

2.4 Applications: From Dreams to Mass Markets

Every market has its (to use the vernacular of the computer software industry) "killer" applications; those applications, so attractive, so essential, customers will stand in line to buy. The trouble with killer applications, once you know what they are, is making them happen. This section presents some killer applications from the recent past that were impossible (or at least impractical) without data compression.

In the 1960s and 1970s, both the computer and communications industries were growing rapidly. The amount of digital data to be stored and transmitted (mostly text and voice) had grown to the point where computer disk storage and telecommunications network bandwidth were seen as technology constraints looming on the horizon. It was a time to experiment, using the then-fledgling data compression technology to relieve those constraints. Computer manufacturers, telecommunications providers, and large customers for computers and telecommunications were involved. The literature records many proposals and some products for compressing databases, computer files, and voice for storage or transmission [Snyd70, Regh81, Coop82, Hark82, Rait87, Held91]. But none of these efforts elevated data compression to the status of being an essential technology, in part, because solutions could still be found by throwing more money at the problem—buy more storage, contract for more bandwidth—and customers were willing and able to do so. By the 1980s, changes were afoot. Money became scarcer. Computer and telecommunications customers became more sophisticated. The applications that captured their attention grew increasingly more functional, more complex. Digital technology took on more importance for delivering the right products, but only if the price was right. In the 1990s, these trends continued.

That brings us to the applications shown in Table 2.2. The common thread is that data compression not only made (or is making) these applications possible, but it also revolutionized the marketplace. When these new products using data compression were introduced, they had readily discernible, easily measurable competitive advantages over older products, and demand quickly followed availability. Part III of this book (Applications) explores how data compression is used in each of these applications.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Table 2.2 Killer applications for data compression.

Application	Date	Constraint
FAX	1980	Telephone line bandwidth
Videoconferencing	1982-1985	Telecommunications bandwidth
Cartridge tapes for small computers	1989	Tape cartridge capacity and tape-drive data rate
Disk storage for personal computers	1990	Disk storage capacity
Direct broadcast satellite (DBS) TV	1994	Satellite transponder bandwidth
HDTV	2000	Broadcast channel bandwidth

2.5 Summary

This chapter explored trends in digital systems and their applications along with what end users expect and will get from data compression and how it is the enabler for many of those applications. We learned the following:

- Technology growth for processors, storage, and communications channels will continue but will be outpaced by application demand.
- Microprocessors are key to economical data compression implementation.
- A National Information Infrastructure (NII) is being defined that will provide wider access to all types of information—text, voice, audio, image, video, and whatever can be digitized and transported.
- Personal computers and interactive television receivers are contending to be the future information appliances for tapping into the NII.
- Users not only expect information to be more accessible but are using more bandwidth to access it.
- Interactive multimedia is becoming pervasive.
- Data compression alone cannot solve all storage and bandwidth limitations but must be used in combination with technology advances.
- End users expect data compression to be transparent, to make products and services more affordable, and to enable innovative new products and services.
- Data compression is essential to FAX, videoconferencing, computer tapes and disks, DBS television, and HDTV.
- Data compression will continue to become more pervasive.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Appendix 2A—Channel Bandwidth

Table 2A.1 summarizes the bandwidth of some important information storage and transmission channels encountered throughout this book.

Table 2A.1 Channel bandwidth.

Information Channel	Bandwidth	Comments
Standard telephone line	64 Kbps	
ISDN telephone line	P * 64 Kbps	P = 1 - 30
CD-ROM	1.248 Mbps	2 - 8X enhanced products
CD audio	1.4112 Mbps	
T-1 communications line	1.566 Mbps	
Digital video disc (DVD)	3.5 - 11 Mbps	Average - maximum rates
Ethernet LAN	10 Mbps	
Token-ring LAN	16 Mbps	
6-MHz terrestrial TV channel	20 Mbps	
DBS satellite transponder	27 Mbps	Transmits 4 - 8 channels
6-MHz cable TV channel	40 Mbps	
T-3 communications line	45 Mbps	
FDDI LAN	100 Mbps	
Fast ethernet LAN	100 Mbps	
ATM	155 Mbps	

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Appendix 2B—Media Characteristics

Tables 2B.1, 2B.2, and 2B.3 summarize the characteristics of various media encountered throughout this book.

Table 2B.1 Speech and audio.

Media	Resolution	Data Rate	Channel Bandwidth[1]	Compression Ratio[2]	Compression Algorithm
Telephone speech	8 KHz x 8 bits/sample	0.065 Mbps	< 0.065 Mbps	2 - 16:1	See Table 5.2(a)
Wideband speech	16 KHz x 8 bits/sample	0.131 Mbps	< 0.065 Mbps	2 - 4:1	See Table 5.2(b)
CD audio	2 x 44.1 KHz x 16 bits/sample	1.4112 Mbps	1.4112 Mbps	-	Uncompressed
CD MiniDisc™	2 x 44.1 KHz x 16 bits/sample	1.4112 Mbps	0.292 Mbps	5:1	ATRAC™, See Table 5.3
DCC™ audio tape	2 x 44.1 KHz x 16 bits/sample	1.4112 Mbps	0.384 Mbps[3]	4:1	PASC™, See Table 5.3
HDTV audio	5.1 (6) x 48 KHz x 16 bits/sample	4.608 Mbps	0.384 Mbps	12:1	MPEG-Audio, See Table 5.3

Notes:
[1] Bandwidth of existing channels and networks.
[2] Compression ratio objective to meet the limitations of existing channels and networks.
[3] Audio + ECC data = 0.768 Mbps.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 2B.2 Still image.

Media	Resolution[1]	Original Image	Typical Compressed Image	Typical Compression Ratio	Compression Algorithm
FAX	8.5 in. x 11 in. @ 200 x 200; 1 bit/pixel	3.74 Mb	0.250 Mb	15:1	ITU-T T.4/6
Computer VGA (black-and-white)	640 x 480; 8 bits/pixel	2.46 Mb	0.123 Mb	20:1	JPEG and others
Computer SVGA (color)	1024 x 768; 16 bits/pixel	12.58 Mb	0.629 Mb	20:1	JPEG and others
Kodak Photo CD™ 35-mm slide	3072 x 2048; 24 bits/pixel	151 Mb	48 Mb[2]	4:1	Proprietary; See Section 9.9

Notes:
[1] Pixels/line x lines/image.
[2] The original image and four smaller images are stored at various resolutions and compression ratios. Each 650-MB CD-ROM contains 100 - 110 images.

Table 2B.3 Video.

Media	Resolution[1]	Data Rate	Channel Bandwidth[2]	Compression Ratio[3]	Compression Algorithm
Videophone (AT&T VideoPhone 2500™)	128 x 112; 12 bits/pixel @ 2-10 fps	0.344 - 1.72 Mbps	0.0192 Mbps (Analog telephone line)	18:1 - 90:1	Proprietary
Video-conferencing (ISDN videophone)	QCIF 176 x 144; 12 bits/pixel @ 10-30 fps	3.04 - 9.12 Mbps	128 Kbps (ISDN; P = 2)	23:1 - 70:1	H.261
Video-conferencing (high quality)	CIF 352 x 288; 12 bits/pixel @ 10-30 fps	12.2 - 36.5 Mbps	384 Kbps (ISDN; P = 6)	32:1 - 95:1	H.261
VCR-Quality (VHS equivalent MPEG-1 video)	SIF 352 x 240; 12 bits/pixel @ 30 fps	30.4 Mbps	1.248 Mbps (CD)[4]	25:1	MPEG-1
SDTV	MAIN profile 720 x 480 16 bits/pixel @ 30 fps	166 Mbps	4 Mbps (DBS channel)	42:1	MPEG-2
HDTV	HIGH profile 1920 x 1080 16 bits/pixel @ 30 fps	995 Mbps	15-20 Mbps (6-MHz TV channel)	50:1 - 66:1	MPEG-2

Notes:
[1] Pixels/line x lines/frame.
[2] Bandwidth of existing channels and networks.
[3] Compression ratio objective to meet the limitations of existing channels and networks.
[4] 1.248 Mbps of 1.4112 Mbps bandwidth is available for video.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

5

Compression Algorithms for Diffuse Data

In this chapter, the introduction to data compression algorithms continues with discussions of those algorithms selected as the official standards (or, in some cases, de facto standards) for compressing diffuse data—speech, audio, image, and video. Marketplace forces tremendously influence which compression algorithms become standards, and this is most evident for diffuse data compression standards. The algorithms described in this chapter were selected as standards—first, because they can provide good quality compressed speech, audio, image, and video at reasonable data rates. Second, they can be economically implemented in VLSI hardware (or software, sometimes). Third, they can deliver data in real time—a requirement for speech, audio, and video applications. However, diffuse data compression—particularly video compression—is still new, with a plethora of compression techniques to pick from and still more—and likely better ones—waiting to be discovered. This, coupled with the continuing discovery of new applications for diffuse data compression, means the process of creating standards is hardly finished. In fact, it may have just begun. Throughout the chapter, we will be describing the evolution leading up to today's standards, pointing out those standards in decline and areas where new standards are likely to emerge. As with our discussion of symbolic data compression algorithms, the simplest, most general algorithms for each data type are described first, followed by more powerful or more specialized algorithms.

5.1 Diffuse Data Compression

Diffuse data is compressed by applying lossy compression, which throws away nonessential information, in combination with lossless compression for efficient

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

coding. Many advances in compressing diffuse data have been made by exploiting the limitations of human auditory and visual systems. Humans, who usually are the end receivers of diffuse data, do not need—or cannot use—all the information captured during digitization. Powerful and complex models for speech, audio, image, and video data have been created using what is described as *perceptual coding* techniques that exploit the limitations of human ears and eyes. These are also called psychoacoustic and psychovisual coding techniques because they are based on the principles of human hearing and vision. A perceptual coding model followed by an entropy coder, which uses one or more techniques of the type examined in Chapter 4, produces effective compression. However, a unique model (and entropy coder) is needed for each type of diffuse data because the requirements are so different. The state of the art in compressing and decompressing diffuse data using lossy perceptual coding and lossless entropy coding is described in forthcoming sections of this chapter. Examples are drawn from products and the official standards for speech, audio, image, and video data.

5.2 Speech Coding

The objective of speech coding[1] is to represent speech using as few bits as possible while preserving its perceptual quality. Speech coding began over five decades ago with research and development of coders for analog telephone transmissions, and it has continued to grow and expand with digital telephony. Beyond the realm of telephony, speech coding is used by other communications and storage applications that demand high-quality speech reproduction. There has been continual upgrading of speech quality for telephone systems and other voice communications channels and for storing voice messages. As Table 5.1 shows, wideband speech, a class of service beyond telephone-quality speech but stopping short of wideband audio, has emerged for applications that require higher quality.

There are two classes of speech coders. *Waveform coders* attempt to preserve the input signal waveform in the sense that the decoded speech signal they produce is a close approximation of the original speech signal. In contrast, *vocoders* (voice coders, which are also called parametric or source coders) attempt to produce a decoded signal that sounds like the input speech signal. Vocoders incorporate vocal-tract models specially tuned for speech signals, unlike waveform coders that rely solely on exploiting the perceptual limitations of the human ear for compression. As a result, vocoders achieve greater compression—but only on speech. Waveform coders may need up to 32 Kbps for telephone-quality

[1] The term *speech coding* is used rather than speech compression, which in speech research and development is sometimes reserved for time-scale modification of the speech signal. For example, as a learning aid, speech may be speeded up during playback.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 5.1 Speech and audio signals.

Audio Channel	Frequency Range (Hz)	Sampling Rate (KHz)	Bits per Sample	PCM[1] Bit Rate (Kbps)	Encoded Bit Rate (Kbps)
Telephone speech	200 - 3,200	8	8	64	4 - 32
Wideband speech	50 - 7,000	16	8 (or more)	128 (or more)	32 - 64
Wideband audio	10 - 20,000	32 / 44.1 / 48	16	512 -768	64 -192

Notes:
[1] See Figure 1.1 for an example of PCM (pulse code modulation).

speech, and still more for wideband speech, whereas vocoders can produce perceptually intelligible speech at bit rates reaching down to 2 Kbps. There also are modern hybrid speech coders that—by combining elements of both waveform coders and vocoders—exploit models of speech production and auditory perception. They offer telephone-quality speech at 4–16 Kbps and wideband speech at less than 64 Kbps.

Throughout its long, rich history, speech coding has made significant contributions, not only to speech but also to audio coding. For instance, subband and transform coding (to be described in Section 5.3) were first used for speech coding. However, the main thrust of speech coding research and development has been focused on techniques suitable for just one application—telephone systems. One consequence is that many speech coders operate at a constant bit rate, sampling at regular, fixed intervals like the speech digitization example shown in Figure 1.1. They take no advantage of the pauses and silent intervals that occur in conversational speech, nor do they attempt to code different speech segments at different rates while maintaining a given reproduction quality. Variable bit-rate coders may be more efficient, but, until recently, there was no demand for them because most analog telephone and telecommunications channels operated at fixed rates. With the availability of digital, packet-oriented broadband communications services, like ATM (asynchronous transfer mode), this has changed, and variable-rate speech coders are beginning to appear. Another constraint imposed on the design of most speech encoders (and decoders) is that they must operate in real-time, two-way communications systems. In this environment, encoding and decoding times must be nearly equal, and neither can introduce much delay because delaying the signal disrupts voice communications. A third constraint is that speech encoders and decoders must operate when nonspeech signals are present on the communications channel, whether those signals are channel control signals or just channel noise.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The point to be made is that other applications for speech coding such as broadcasting and voice messaging do not have these constraints. For some of these applications, more complex algorithms that use more processing power may provide higher-quality speech at lower bit rates (while incurring longer delays). For others, simpler algorithms having different design parameters and requiring less processing power may be more appropriate.

Another influence on speech coding has been the technology available for its implementation. Vocoders, which first appeared in 1939, were constrained by the limitations of analog (telephone) signals and analog signal processing. Later, when the telephone network went digital, new signal processing opportunities developed but speech coding was still constrained by the speed (and cost) of digital circuitry. Only applications such as military communications could afford to use the low-bit-rate speech coders that researchers knew how to build. Today, with advances in VLSI technology, along with advances in the theory of digital signal processing, the selection of affordable speech coding algorithms is wider than ever.

Although a tremendous variety of speech coders have been developed and standards created for them, some have quietly passed into history, having been replaced by improved techniques. Some currently popular, widely used, standardized techniques for telephone-quality and wideband speech are examined in this section. For a more thorough treatment of speech coding techniques, the reader is directed to surveys by [Gers94, Span94, Noll95] and many references cited by those surveys.

ADPCM coders

Adaptive differential pulse code modulation (ADPCM) was formalized as a standard in 1984 by the ITU-T G.721 recommendation (a standard since replaced by the ITU-T G.726 recommendation). ADPCM operates by encoding and transmitting the difference between the original speech signal and its predicted value. As shown in Figure 5.1, the encoding process begins by breaking digitized, time-continuous speech into a series of discrete frames, each containing the PCM (pulse code modulation) samples for a few milliseconds of speech, and processing each frame. Then, the difference between a frame of the (8-bit, 64 Kbps) PCM input speech signal and a prediction of that signal is fed to a quantizer. By exploiting the correlation between adjacent samples, the quantizer constructs a 16-level (4 bit, 32 Kbps) representation of the difference signal. This coded difference or prediction error signal is sent to the decoder. Meanwhile, within the encoder, the same coded difference signal is used by the inverse (quantizer) function to reconstruct the original difference signal. The inverse function supplies the reconstructed difference signal to the predictor, which attempts to predict the original speech signal. Both the quantizer and the inverse function are adaptive in that the quantization levels are not fixed, as in standard PCM, but change with the signal. The predictor is adaptive too. It operates in two modes as determined by the characteristics of the difference signal: In the fast mode, the original signal

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

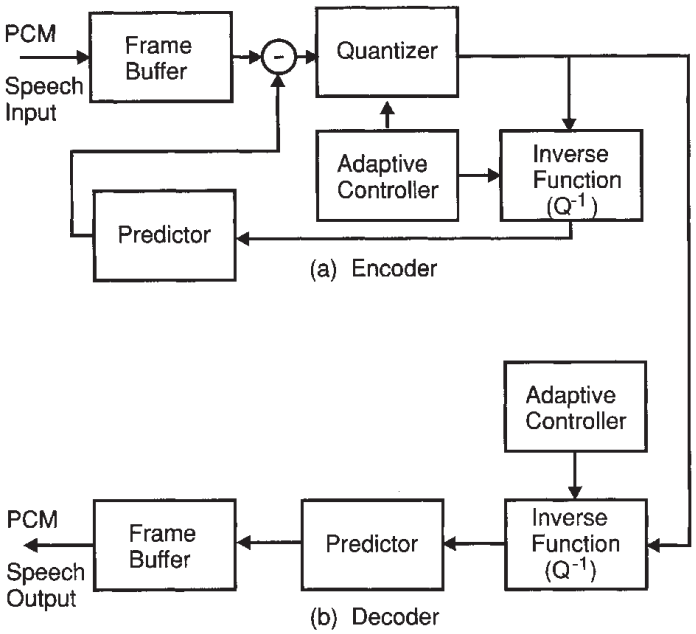


Figure 5.1 ADPCM coder.

is assumed to be speech, whereas in the slow mode, it assumed to be nonspeech, network signaling or voiceband data. Two different modes are required because speech and modem-generated signals have different statistical characteristics.

The decoder receives only the difference signal and uses it to reconstruct an approximation of the original speech signal. The decoder has a copy of the inverse function and the predictor, both of which operate and adapt identically to the corresponding encoder functions. Therefore, if the predicted signal really is a good estimate of the original speech signal, the encoder output will be too.

It is important to note that, like most speech encoders designed for communications applications, the ADPCM encoder operates in a "backward adaptive" mode. Backward adaptive means that the encoder uses only previously seen samples of the difference signal to generate its output. It does not buffer large blocks of the input speech signal (nor does the decoder). Thus, the encoder avoids delaying the signal while looking forward for information to adapt its output. This provides the low delay needed for real-time, two-way voice communications.

Beyond the ITU-T G.721 recommendation, which operated at 32 Kbps, the basic ADPCM algorithm has been extended to operate at higher and lower rates in systems with variable bit rates. The ITU-T G.726 and ITU-T G.727 recommendations operate at 16, 24, 32, and 40 Kbps.[2] However, at 16 Kbps (or

[2] ITU-T G.726 replaces ITU-T G.721 and another recommendation pertaining to ADPCM, ITU-T G.723 [Sayo95]. Later, ITU-T reused the G.723 designation for a new recommendation discussed later.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

lower), other algorithms can produce better quality speech. These algorithms are considered next.

LPAS coders

Today, the most widely studied and implemented approaches to speech coding use linear prediction-based analysis-by-synthesis (LPAS) coding techniques. LPAS coders use a linear predictive coding (LPC) synthesis filter (the linear predictive part of the acronym) to generate synthesized speech from stored excitation signals. Analyzing the speech signal by attempting to synthesize it (the analysis-by-synthesis part of the acronym) allows the most appropriate excitation signal to be selected. LPAS coders, like ADPCM coders, are waveform coders because they transmit a representation of the speech signal waveform to the decoder. But it is a highly compressed representation, and these coders can work effectively at 4–16 Kbps.

We will illustrate how this class of coders works by using one type, the LD-CELP (low-delay code-excited linear predictive coding) 16-Kbps coder shown in Figure 5.2. This coder is specified for the ITU-T G.728 recommendation. The encoding process begins by breaking the time-continuous input speech signal into a series of frames, each containing the PCM samples for a few milliseconds

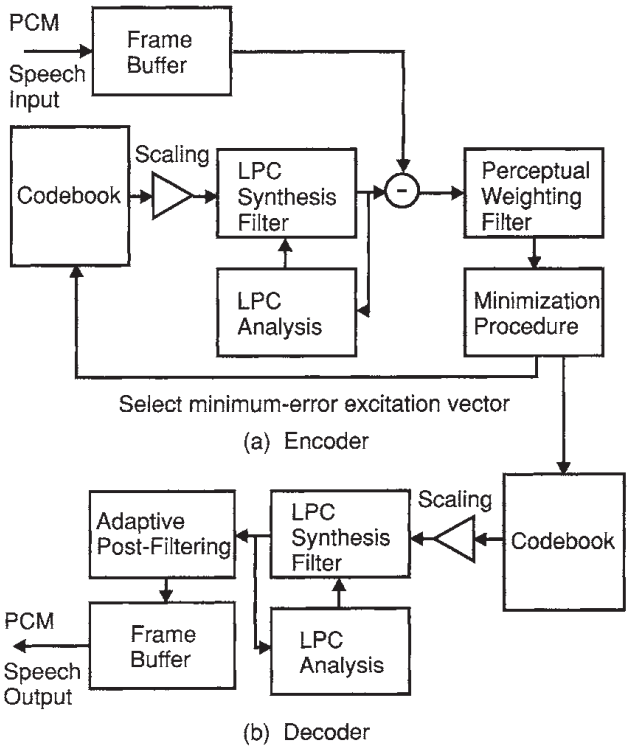


Figure 5.2 Low-delay CELP coder.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

of speech, and processing each frame. Here, we see the difference between a frame of the (8-bit, 64 Kbps) PCM input signal and a synthesized speech signal is fed to a perceptually weighting filter that computes a "score" for the frequency-weighted difference of the two signals. The result is provided to the minimization procedure whose job is to cycle through all the excitation signals stored in the codebook and find the one that will generate a synthesized speech signal that achieves the best score—the one most closely matching the input speech signal. When the match is found, the codebook address for that excitation signal is transmitted to the decoder. Because the codebook address is much smaller than the segment of speech it represents, compression results.

An identical codebook is maintained at the decoder. From it, a copy of the same excitation signal is selected and is used to regenerate the synthesized speech representation most closely matching the input signal. After the synthesized speech signal is filtering to reduce the perceptual effects of quantization noise (at the expense of signal distortion), it becomes the output speech signal.

Let us look more closely at the heart of LPAS coders—the excitation signals and the codebook that contains them. Excitation signals are vectors of numbers representing common phonetic sounds. These signals are generated according to voice-production models that imitate the human vocal tract. Excitation refers to how the human vocal tract is stimulated to generate sounds. There are numerous models for excitation signals. The one for LD-CELP uses a complete sequence of impulses to represent each excitation. The representation is stored as a code in a codebook (the CE part of the LD-CELP acronym). Incidentally, LD means low delay because backward adaption, a small frame buffer, and short (five-sample) excitation signals are used.

Creating and managing the codebook is a major challenge for LPAS coders. A typical CELP codebook may contain 1024 excitation vectors, each of which may be up to 40 samples in length. Because each sample is 8 bits in size, 40 KB of storage is required for the codebook. Advances in VLSI capacity no longer make storage capacity the important factor it was when CELP algorithms were first proposed, but searching the storage—efficiently—was and still is an issue. Indeed, many variations and improvements on CELP algorithms have been directed to finding ways to locate the one excitation vector which best matches the input speech signal while avoiding the time and computation overhead incurred by a complete search of the codebook. Another issue, beyond the scope of this discussion, relates to finding the best excitation signals to place in the codebook: If the codebook is not adaptive, then the initial excitation signals are critical. If the codebook is adaptive (which the LP-CELP codebook is not) and the excitation signals are frequently changed to track changes in the input speech signal, the overhead for codebook adaption may be great.

Vocoders

The basic idea of vocoders is that the encoder can analyze speech in terms of a model and the model parameters can be extracted and transmitted to the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

decoder. The available bit rate for some communications channels, particularly for military and cellular telephone applications, is lower than for regular telephone lines; yet, intelligible speech communications are required. As bit rates fall below 4 Kbps, the speech produced by the best available waveform coders exhibits noticeable noise and distortion causing both speech quality and intelligibility to markedly degrade. The reason this happens is that not enough bits are available to transmit even a highly compact representation of the original waveform. For very low bit-rate speech coding, vocoders are solutions because they do not try to reproduce a waveform similar to the original. Instead, vocoders rely on reproducing a signal that sounds perceptually similar to the original, using only parameters obtained from models of human speech. Consequently, they obtain lower bit rates; but, as noted earlier, they work only on speech signals.

Vocoders were the earliest type of speech coders but fell into disuse because the speech they produced was unnatural, machinelike, and not particularly intelligible. However, interest in vocoders has revived. Speech quality has improved thanks to advances in speech models and technology that allows more complex, more powerful algorithms to be implemented. Indeed, many of the adaptive techniques used in LPAS coders are also used in vocoders (or perhaps LPAS coders borrowed them from vocoders; the distinction between the two is not always clear).

While the search for still better vocoder algorithms continues, one known as LPC-10 has been standardized as (U.S.) Federal Standard FS-1015 for speech coding at 2.4 Kbps. The LPC-10 algorithm uses a 10th-order linear predictor to estimate the vocal tract parameters which are sent to the decoder without sending bits that describe the excitation waveform. The decoder uses a linear prediction filter to synthesize speech from an excitation signal that it generates using only the passed parameters. Further details can be found in [Gers94, Span94].

Another vocoder based on a multipulse maximum likelihood quantization (MP-MLQ) linear predictive coding (LPC) technique has recently been standardized by ITU-T as part of their G.723 recommendation.[3] The G.723 recommendation, which specifies operation at 6.3, 5.3, and 4.8 Kbps, is intended for use in visual telephony over the public telephone network [Dspg95]. This technology was first offered by the DSP Group, Inc. in their Truespeech™ family of speech compression algorithms and software. Truespeech has become a widely used de facto standard for digital speech in personal computer and computer telephony applications.

Wideband speech coding

Wideband speech is an important part of computer-generated speech, videoconferencing, and other new applications that use ISDN (Integrated Services Digital Network) telecommunications. We shall consider two of the many approaches

[3] This is a new ITU-T G.723 recommendation. An earlier ITU-T G.723 recommendation pertaining to ADPCM coding was replaced by ITU-T G.726 [Sayo95].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

for wideband speech coding. One is provided by the MPEG-2 audio standard which will be examined in Section 5.3: It uses subband and adaptive transform hybrid coding techniques developed for MPEG-1 audio coding. MPEG-2 extended the MPEG-1 audio standard to include lower sampling rates at 16, 22.05, and 24 KHz. This allows wideband speech signals with bandwidths of 7.5, 10.3, and 11.25 KHz, respectively, to be coded at bit rates of 32 Kbps or higher.

Another approach is provided by the ITU-T G.722 recommendation for wideband speech coding. This standard uses a sampling rate of 16 KHz and 14-bit PCM samples, allowing wideband speech signals with a 7 KHz bandwidth to be coded at bit rates of 48, 56, and 64 Kbps. This coder is a hybrid too; it uses two subband filters followed by ADPCM coders. Both subbands are subsampled to 8 KHz. The low (frequency) subband signal is roughly equivalent to telephone-quality speech; it is coded with a 6-bits-per-sample quantizer. The high subband contains less information and is coded with a 2-bit per sample quantizer. The 48- and 56-Kbps bit rates are achieved by reducing the low subband quantizer resolution to 4 and 5 bits per sample, respectively, with a reduction of speech quality.

State-of-the-art speech coding and beyond

Among the speech coders examined in this section, those intended for communications over telephone lines achieve compression ratios in the 2:1 to 4:1 range. For applications such as military communications, cellular telephones, and personal communications systems, where channel bandwidth is more restricted or more expensive, higher compression ratios (and lower bit rates) are required. These applications also demand improved coding techniques because their communications channels have higher error rates, and more background audio noise is present in mobile environments. In computer-based speech applications, the requirements for speech coding are different too. There, reducing the amount of storage consumed by speech files (really low-bit-rate coding in disguise) and ease of coding are equally important as speech intelligibility itself. Then, too, speech and audio (music) are mixed for many applications where lower bit-rate coding would be welcome.

Progress in speech coding continues in all these areas. One can expect that lower bit rates will be achieved for higher-quality speech by using ever-more-complex algorithms based on improved models of speech production and better understanding of the human auditory system. Continued advances in faster and more highly integrated VLSI signal processing devices will make real-time implementation of more powerful speech coding algorithms possible. Meanwhile, many standards have been set for speech coding. Some of today's most important speech coding standards are listed in Table 5.2. More information about the speech coding algorithms they use can be found in the references cited in Table 5.2.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 5.2(a) Telephone-quality speech coding standards.

Organization Standard[1]	Algorithm[2]	Encoded Bit Rate (Kbps)[3]	Application
ITU-T G.711	PCM	64	General telephony [Port94, Trow94]
ITU-T G.726	ADPCM	16 / 24 / 32 / 40	General telephony and digital circuit multiplication equipment, an extension of G.721 [Gers94, Span94, Noll95]
ITU-T G.727	Embedded-ADPCM	16 / 24 / 32 / 40	Speech transmission over packet-oriented networks [Noll95]
ITU-T G.728	LPAS (LD-CELP)	16	General telephony [Gers94, Port94, Span94, Trow94, Noll95]
ITU-T G.729	LPAS (CELP)	13	Cellular telephony [Noll95]
GSM	LPAS (RPE-LTP)	13	European cellular telephony [Gers94, Span94, Noll95]
JDC TDMA	LPAS (PSI-CELP)	11.2	Japanese cellular telephony [Gers94, Span94, Noll95]
BTI Skyphone Service	LPAS (MPLP)	9.6	Aeronautical telephone service [Span94]
TIA IS54	LPAS (VSELP)	7.95	TDMA cellular telephony [Gers94, Span94, Noll95]
TIA IS95	LPAS (QSELP)	< 8 (variable)	CDMA cellular telephony [Gers94]
ITU-T G.723	MP-MLQ LPC (Vocoder)	4.8 / 5.3 / 6.3	Visual telephony and computer-generated speech, based on an industry wide de facto standard —Truespeech™ [Dspg95]
FS 1016	LPAS (CELP)	4.8	Secure voice transmission [Gers94, Span94, Noll95]
ITU-T SG 15/2 (Future proposal)	na	4	Videotelephone [Port94]
FS 1015	LPC (Vocoder)	2.4	Secure voice communications [Gers94, Span94]

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Table 5.2(b) Wideband speech coding standards.

Organization Standard[1]	Algorithm[2]	Encoded Bit Rate (Kbps)[3]	Application
ITU-T G.722	Subband-ADPCM	48 / 56 / 64	Videoconferencing [Gers94, Noll95, Port94, Trow94]
MPEG-2	Subband-adaptive transform	32 / 48 / 64 or higher	General voice communications [Noll95]

Notes:

[1] Standards groups:

BTI	British Telecom International
FS	U.S. Government Federal Standards
GSM	Groupe Spéciale Mobile
ITU-T	International Telecommunication Union - Transmission
JDC	Japanese Digital Cellular
MPEG	Motion Picture Experts Group
TIA	(North American) Telephone Industry Association

[2] Algorithms:

Acronym names are provided in the Acronyms and Abbreviations section of this book. Descriptions are found in the preceding text and the references listed in this table.

[3] Input bit rate is 64 Kbps (8 bits per sample x 8 KHz sampling rate) except:

G.723	128 Kbps (16 bits per sample x 8 KHz sampling rate)
G.722	224 Kbps (14 bits per sample x 16 KHz sampling rate)
MPEG-2	256 / 353 / 384 Kbps (16 bits per sample x 16 / 22.05 / 24 KHz sampling rate)

5.3 Audio Coding

There is an ever-growing demand for high-quality audio—sometimes called wideband audio—for communications, consumer-electronics products, and entertainment applications. The marketplace trends for audio continue unabated—provide more fidelity and more channels, often together with video, over communications channels where bandwidth is limited, using ever-smaller storage devices where capacity is limited. All these factors provide the motivation for efficient audio coding.[4]

Audio perceptual coding takes advantage of the inability of humans to hear every sound that impinges upon their ears. Two psychoacoustic principles are

[4] The terms *audio coding*, rather than data compression, and *audio coder*, rather than data compression algorithm, are used, following the conventions of literature in this field.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

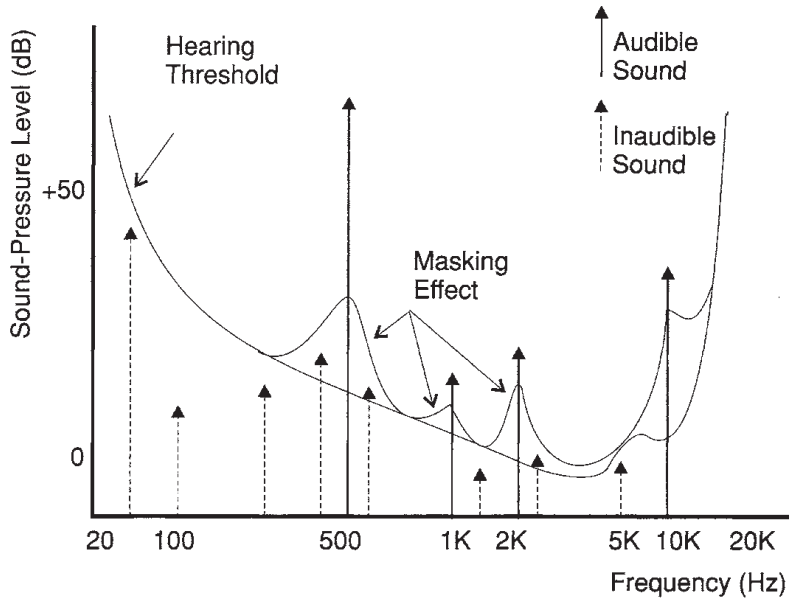


Figure 5.3 Audio thresholding and masking.

involved: thresholding and masking. These concepts are illustrated in Figure 5.3. Thresholding simply means there are sounds too soft for humans to hear because they fall below the threshold of audibility, the hearing threshold. Masking means there are sounds that cannot be heard because—temporarily—they are masked by nearby louder sounds that raise the threshold of audibility. When two sounds with a small difference in frequency are present simultaneously, *frequency-domain masking* (the effect shown in Figure 5.3) occurs and the stronger sound masks the weaker. This is one type of masking exhibited by the human ear. Another type is *temporal masking*, which occurs when a weaker sound of similar frequency appears within a small interval of time, before or after (or simultaneously with) a stronger sound. In either case, the stronger one masks the weaker one.

Thresholding and masking lay the groundwork for state-of-the-art audio coders. Some compression is achieved by simply ignoring sounds that are too soft—those that fall below the threshold of audibility. Further reduction is achieved by recording audible sounds with only the degree of digital resolution necessary for high-fidelity reproduction. The most effective audio coders continuously vary the number of bits used to record each component of a complex sound. Adaptively, they assign bits according to the audibility of that component, the potential audibility of the playback noise generated by the encoding (quantization) process itself, and the number of bits available for storing or transmitting the compressed data.

Another characteristic of the human ear is used in designing effective audio coders: The human auditory system can be thought of as bandpass filter bank covering the frequency range of 10–20,000 Hz. Within this range, the bandpass

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

filters strongly overlap each other, and their bandwidth, which may be as narrow as 100 Hz at low frequencies, becomes increasingly wider, growing to 5000 Hz for the highest frequencies. This characterization of the human ear, along with thresholding and masking, becomes the basis for designed effective *frequency-domain audio coders*. These coders achieve compression by splitting the audio spectrum into frequency bands that can be individually quantized and coded according to the rules for perceptual coding.

Audio coding basics

The encoding and decoding processes for audio data are shown in Figure 5.4. They begin with a source signal that has been time sampled and digitally coded

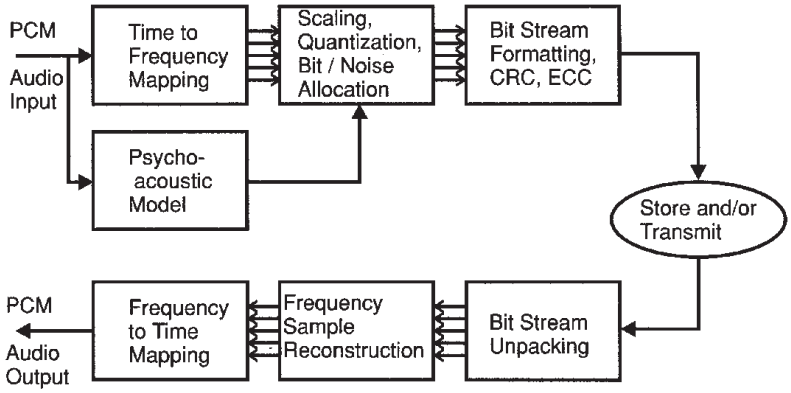


Figure 5.4 Audio encoding and decoding.

using PCM. For high-quality audio, 16-bit PCM samples are used and sampling rates range up to 48 KHz. The first step of encoding is a time-to-frequency transformation that breaks the audio frequency spectrum down into many narrow spectral components arranged according to the ear's sensitivity characteristics. The number of spectral components varies with different designs; current applications of *subband coders* use as few as 32 subbands whereas those for *adaptive transform coders* use up to 576 spectral components or transform coefficients. Generally, more spectral components allow greater compression at the expense of more complexity. To generate the spectral components, subband coders feed the source signal into a set of bandpass filters arranged contiguously in frequency. Adaptive transform coders process the input signal using a discrete transform—either a discrete Fourier transform (DFT) or a modified discrete cosine transform (MDCT). There also are *hybrid coders* that combine subband and adaptive transform coding techniques. This allows the characteristics of the human ear to be matched more closely, providing higher frequency resolution at the lower frequencies where it is needed, and lower resolution at higher frequencies.

The spectral components are fed to a scaler and quantizer along with the output from a psychoacoustic model which provides the information needed for deciding how many bits to allocate each spectral component. The complexity and details

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

of psychoacoustic models vary, but the general idea is to analyze each spectral component for the masking and thresholding characteristics of the signals it contains. Simple ideas are involved: If a spectral component is not audible, or contains no energy at all, then it should receive no bits. The closer the signal is to being audible, the more bits it should receive. There are no rules for setting up psychoacoustic models—other than the audio coder must provide a certain compression ratio for specific applications—and different coders take different approaches: A technique used by the MPEG-1 audio standard and Philips PASC™ coders is to measure the distance from the peak value of the signal to the masking level, the SMR (signal-to-mask ratio), to find the number of bits needed to record each component of a complex sound [Pan94, Hoog94]. An adaptive noise allocation scheme is used for the MPEG-2 audio standard where the encoder iteratively varies the quantizers to maximize the signal-to-noise ratio for the encoded data stream [Pan95]. The Dolby AC-3™ coder uses an adaptive psychoacoustic model to extract the spectral envelope, a frequency-sensitive representation of the overall audio signal used for both encoding and decoding [Todd94, Noll95]. The Sony ATRAC™ coder uses a nonuniform time slitting approach that provides information which allows the encoding rate to vary depending on whether the signals are changing rapidly or slowly [Yosh94C].

Next, in the scaler and quantizer, each spectral component is allocated bits from a pool of available bits where the size of the pool is defined by the recording or transmission channel data rate. The number of bits assigned to each spectral component is determined by the bit or noise allocation algorithm. Using information supplied by the psychoacoustic model, the bit allocation algorithm allocates bits so as to use the minimum number, whereas a noise allocation algorithm does the same thing but attempts to minimize the audible noise in the encoded output. Each spectral component is coded as a quantization value and a scale-factor value (called mantissa and exponent in floating-point number notation), allowing the signal to cover a wide dynamic range.

There are no standards for representing spectral components, and different algorithms have chosen different schemes. In MPEG-1 and PASC the quantization values range from 0 to 15 bits, whereas 6 bits are used for scaling, which allows an input signal range of 120 dB [Pan94]. In Dolby AC-3, each spectral component can be quantized to a maximum resolution of 16 bits. But rather than just encoding the most significant digits, AC-3 scales and offsets the mantissa to provide zero-centered, equal-width, symmetrical quantization levels that reduce distortion and noise [Davi93, Davi95]. The exponent of each AC-3 spectral component can range in value from 0 to 24 (where a delta of 1 represents a 6-dB sound-pressure level change). But rather than sending all the bits, exponents (except the lowest-frequency term) are encoded as differential values. Changes in value of +2 to −2 from the spectral component next lower in frequency are allowed, and each is encoded by AC-3 using only slightly more than two bits [Todd94, Davi95]. In the AC-3 decoder, the mantissas and exponents are restored to their actual values.

Lossless coding, which was mentioned earlier, may be applied after all the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

preceding steps have taken place, but only the more sophisticated coders—that need the extra compression it provides—choose to use it. For example, Huffman entropy coding is included in the more advanced forms of MPEG-1 audio [Pan94].

The last step of encoding is to pack the encoded spectral components into blocks of data for storage or transmission. In this step, depending on the needs of the storage or transmission channel, CRC (cyclic redundancy check) error-detecting code or ECC (error-correcting code) may be added.

The decoding process reverses what was done during encoding. Most audio decoders provide a straightforward inversion of the encoding process, but there is no provision for psychoacoustic modeling in the decoder—with the exception of Dolby AC-3 [Noll95].

Multichannel audio coding

The audio encoding and decoding processes shown in Figure 5.4 apply to one audio channel. Multiple audio channels can be coded by duplicating the encoder and decoder for each channel. However, for the predominant multichannel audio systems—two-channel stereo and 5.1-channel surround sound—this is wasteful for there is considerable correlation between individual channels. In stereophonic audio, the left and right channels are not totally independent. In addition, the human ear has yet another perceptual weakness when it comes to stereo imaging that allows interchannel masking effects to be exploited. The net result is that much of the data in stereophonic audio does not contribute to the localization of sound sources, allowing some spectral components of the individual channels to be combined and coded as single values.

There is even more potential for combining multiple channels with surround sound, an audio reproduction technique that provides realistic lifelike three-dimensional spatial acoustics. Surround sound was originally developed for motion pictures and soon will be applied to HDTV. It uses left, center, and right main channels, left and right surround channels, and a low-frequency enhancement subwoofer channel. The subwoofer channel is band limited (15–120 Hz), resulting in surround sound being designated as a 5.1-channel system. State-of-the-art MPEG-2 and AC-3 multichannel coders compress the 5.1 channels and provide satisfactory performance using a total bit rate of only 320 or 384 Kbps. This is about the same bit rate as for compressed two-channel audio and one-fourth that of ordinary (uncompressed) two-channel CD-quality audio.

State-of-the-art audio coding and beyond

The characteristics of today's perceptual audio coders are that they achieve 4:1 or 5:1 compression ratios on CD-quality audio, and up to 12:1 compression for multichannel applications. They are almost symmetrical (decoding is about 2 times faster than encoding), somewhat computational intense, and quite readily

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

implemented in DSPs (digital signal processors). Table 5.3 summaries key attributes of the audio coders discussed in this chapter. The simplest applications require audio coders for two channels of CD-quality audio. The more complex MPEG-2 and Dolby AC-3 audio coders provide multichannel coding for 5.1 audio channels. In addition, the MPEG-2 audio standard supports a wide range of data rates and bandwidths from studio quality to telephone toll-line quality.

Table 5.3 Perceptual audio coders.

	ATRAC	MPEG-1[1] & PASC	MPEG-2	Dolby AC-3[2]
Audio sampling rate (KHz)	44.1	32, 44.1, 48[3]	16, 22.05, 24, 32, 44.1, 48	32, 44.1, 48
Audio channels	2	2	5.1	5.1
Compression ratio	5:1	Varies (4:1 PASC)	Varies	Varies
Encoded bit rate (Kbps)	292 (stereo)	32-224 (mono) 384 (stereo)	384 (5.1)	32-640 (mono) 320/384 (5.1)
Coding	Hybrid	Subband	Hybrid	Transform
Subband filters	3	32	32	-
Spectral components	512	32	192 / 576 (dynamic)	64 / 128 / 256 (dynamic)
Psychoacoustic model	Adaptive transform block size using signal change rate	Signal-mask-ratio	Signal-mask ratio	Log-domain spectral envelope encoding and decoding
Bit or noise allocation scheme	Implementation dependent	Mask-to-noise ratio	Adaptive noise allocation	Mask-to-signal ratio
Quantization bits per component	na	0 -15	0 - 15	≤ 16
Scale factor bits per component	na	6	6	< 5[4]
Entropy coding	No	No	Yes	No

Notes:
[1] MPEG-1 has three layers of coders: Layer I is described. The Layer II coder is similar to Layer I. The Layer III coder is similar to MPEG-2.
[2] AC-3 is specified for HDTV and DVD as an alternative to the MPEG-2 audio standard coder.
[3] PASC operates only at 44.1 KHz.
[4] Differential encoding assigns about 2.3 bits per exponent. See text.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

A variety of applications are anticipated for MPEG-2 audio, not only in consumer audiophile products but also for cellular phones, digital radios, conferencing gear, and just about anywhere high-quality audio is needed.

Perceptual audio coding is a relatively new application of data compression and, as is apparent, there are several approaches. In the future, there may be new solutions for audio coding based on improved psychoacoustic models. Many applications, particularly multichannel audio coding, are also new, and there remains much to be learned. Perhaps the algorithms described in this section provide audio of sufficient quality, at acceptably low data rates, for costs that will spur mass-produced products—perhaps not; audiophile experts are still debating the merits of these psychoacoustic coding techniques [Mitc95, Taki95B]. This issue will be examined in more detail in Section 9.2. Interesting new applications for audio are emerging too. Some require coding at much lower bit rates, and others use audio to support low-bit-rate video coding where speech information can be used to enhance picture quality. These applications will require extensions of existing designs, or they may result in completely new approaches [Noll95].

5.4 Image Compression

In this section, compression techniques for still images are examined; in the next section, moving images (video) will be examined. The applications for digitized still images are many and varied—the now familiar FAX machine, commercial imaging systems for paperless offices, medical imaging, satellite imaging for remote sensing of earth and space, filmless electronic cameras, computer-generated images, and more. All digital image systems represent a two-dimensional picture as a string of bits, which, as learned in Chapter 1, is accomplished by scanning the image in a regular pattern (say left to right, top to bottom) and collecting samples at prespecified points. As shown in Figure 5.5, usually these points are arranged as an equally spaced rectangular grid on the image. Each sample is called a pixel (picture element). The digital value(s) of each pixel represents some measure of the brightness, color, or spectral attributes sensed at that point in the image. This data may be stored (typically as a row-column matrix that corresponds to the image grid), processed by computer, transmitted, and used to reproduce a likeness of the original image. How faithful the reproduction will be depends on many factors, one of the most important being the sampling resolution that determines how many pixels represent the image. Using more pixels allows more image detail to be captured; it also means larger capacities for storing images and larger bandwidths for transmitting them (and more expensive, complex imaging systems for collecting them too).

The adage "A picture is worth a thousand words" may be true for human communications, but digital pictures, especially large-format, high-resolution color images, are worth far more—in the realm of digital words (or bits). Digital imaging systems generate huge amounts of data. For example, a color image

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

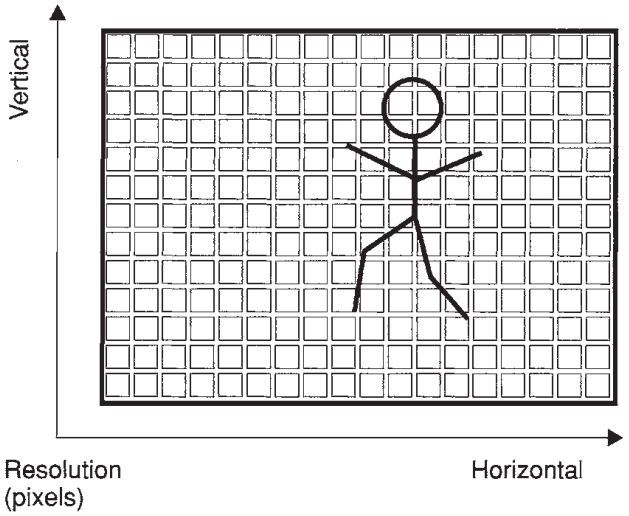


Figure 5.5 Digital image representation.

digitized at 24 bits per pixel and displayed at VGA resolution (a 640 × 480 pixel computer screen format) requires nearly a megabyte of storage; transmitting this image via telephone, even over the fastest lines, would take several minutes. Luckily, digitized images are highly compressible because (1) when an image is digitized, the pixels in any region of the image are similar and (2) many imaging systems capture information the human eye cannot see. Image compression algorithms take advantage of these redundancies and reduce the number of bits required to represent images. Compression ratios range from as little as 2:1 or 3:1 to 100:1 or more depending on the characteristics of the image and whether the application can tolerate the use of lossy compression. However, as learned in Chapter 2, even compressed images still use a lot of bits, several times more than the proverbial thousand words.

There are many different types of images, and each demands different compression techniques. This section examines the standardized compression algorithms for bi-level images and for continuous-tone, gray-scale and color, photographic-quality images. The pixel representations for bi-level, gray-scale, and color images are shown in Figure 5.6. These types of images arise in the following situations:

Bi-level or binary images are captured by scanners that use only two intensity levels, one for "information" and another for the "background." Therefore, each pixel is a single bit whose value is either 1 or 0. One-bit pixels are appropriate for human-created documents containing text, line drawings, or illustrations. The best known example of bi-level imaging is transmitting black-and-white documents via FAX. Photographs, too, are sometimes captured as bi-level images with a digital halftoning process. This is similar to the technique newspapers use to reproduce photographs where black dots of varying sizes, spaced closer or further apart, are used to simulate shading, creating the effect of continuous tones.

Gray-scale images are captured by scanners that use multiple intensity levels

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

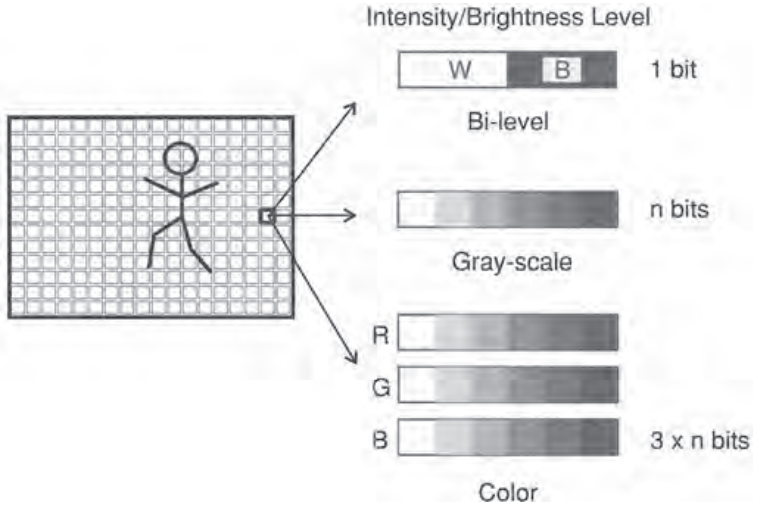


Figure 5.6 Pixel representation.

to record shadings between black and white. Typically, each pixel is represented by N bits—8–12 bits, or more. Gray-scale imaging techniques are appropriate for monochrome photographs and medical images where accurate representation of shading is important.

Color, photographic-quality images are captured by scanners that use multiple intensity levels and filtering to capture the brightness level for each of the three primary colors in visible light—red, green, and blue.[5] Here, each pixel is represented by three values, one for each of the primary color intensities, requiring up to 24 bits or more. This imaging technique is appropriate for images of natural scenes or even human-created images where it important to capture continuous tones.

There are also multispectral images that include information which lies outside the visible-light spectrum. For instance, satellite imaging systems for remote sensing of earth and space may capture infrared radiation. Multispectral images will not be specifically addressed in this book, except to remark that either the JBIG or JPEG algorithm can be used to compress them.

ITU-T T.4 and T.6 recommendations

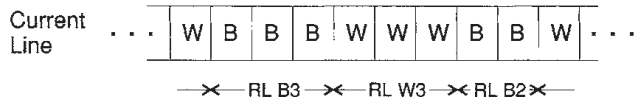
The ITU-T T.4 and T.6 recommendations (standards) for Group 3 and Group 4 facsimile machines contain algorithms that losslessly compress bi-level FAX images. They provide compression ratios that vary widely for different documents but can easily approach 15:1 or more. Their algorithms use spatially nearby neighbor pixels to compress successive image pixels. The MH and MR (modified

[5] Red, green, and blue are the *additive* primary colors for light-emitting systems such as computer displays and scanners. Cyan (a blue-green), magenta, and yellow are the *subtractive* primary colors that, in combination with black, are used in printing, a light-absorbing process [Penn93, Trow94].

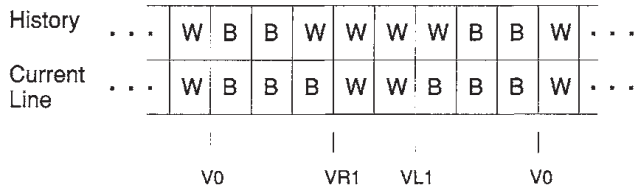
Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Huffman and modified READ[6]) algorithms of ITU-T T.4 use one- and two-dimensional information, respectively. Both use run-length and Huffman coding. These algorithms also include integrated error detection, reflecting their application to communications over noisy telephone lines. The MMR (modified modified READ) algorithm in ITU-T T.6 improves the performance of the MR algorithm for noise-free environments (such as in computers using separate error detection) by eliminating the integrated error detection.

Examples of ITU-T T.4 and T.6 coding are shown in Figure 5.7. The ITU-T committee defined one-dimensional compression first. The one-dimensional ITU-T T.4 MH algorithm uses a run-length coding model followed by static Huffman coding. Only the current line of pixels is used for one-dimensional coding. Figure 5.7(a) illustrates one-dimensional run-length coding with runs of three black pixels, three white pixels, and then two black pixels. Each line is terminated with an EOL (end of line) code to allow resynchronization after transmission errors. Additional details of modified Huffman coding are provided in Section 4.4.



(a) One-dimensional run-length codes



(b) Two-dimensional vertical reference codes

Figure 5.7 ITU-T T.4 and T.6 bi-level image coding.

One-dimensional coding of a page containing dense text did not give enough compression to achieve subminute page transmission over 4800 bits/second communication lines. So the ITU-T T.4 committee also defined an optional two-dimensional MR coding technique in which some lines are coded using one-dimensional MH coding while others are coded relative to the previous line using vertical relationships (to be explained momentarily). Each line is terminated with an EOL code followed by a bit to specify whether the next line is coded one or two dimensionally. Error recovery from transmission errors is possible at the one-dimensionally coded lines (which are forced to occur at least every four lines to minimize possible error propagation). The two-dimensional MR algorithm

[6] READ is an acronym standing for Relative Element Address Designate.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

begins by applying MH coding to a line of pixels. It then goes on to code the next several lines by finding matching pairs of black-white and white-black transitions in consecutive lines and coding the vertical relationships between them. Figure 5.7(b) shows a section of a current line in which the first white-black transition is vertically aligned with that in the previous (history) line; it is coded as "V0." The next black-white transition is vertically offset right one position from the history line; it is coded as "VR1." The next white-black transition is vertically offset left one position; it is coded as "VL1," and so on. This vertical mode of coding is used only when a transition occurs within ±3 pixels of an identical transition in the history scan line. When this condition is not met, one of two alternative coding modes applies: If the history line contains a run of black or white pixels that are not in the current line, the pass mode is used; the coder outputs a special code to indicate this run has been skipped and should not be used during decoding. If the current line contains a run that is not in the history line, the horizontal mode is used; the coder outputs yet another special code and codes the next two runs of the current line using one-dimensional MH coding. Additional details of both MH and MR coding can be found in [Mcco92, Witt94].

For digital transmission environments where transmissions can be guaranteed to be error free, the MR algorithm was further modified to create the ITU-T T.6 MMR coding technique. In MMR, every line is coded two dimensionally and no EOL codes are used. By eliminating the MR provision for error detection, compression is maximized. According to one source, if MR compresses a simple text document by 20:1, MMR will increase compression about 30% [Penn93]. However, compression results for individual documents may vary widely.

JBIG—ITU-T T.82 recommendation

More recent developments in bi-level image coding are incorporated in the ITU-T T.82 recommendation known as JBIG. In 1988, JBIG, the Joint Bi-level Image Group, was chartered to establish an improved standard for lossless coding of bi-level images. The JBIG algorithm they defined introduced adaptivity and new capabilities, including the following:

- Adaptive modeling using two-dimensional prediction to achieve higher compression in images with periodicity (i.e., halftone images)
- Adaptive arithmetic coding to replace static Huffman coding, allowing robust compression of more data types
- Hierarchical coding, allowing the same image to be decompressed at progressively higher spatial resolutions from a compressed data stream
- Scalability to amplitude precision greater than one bit per pixel for lossless compression of gray-scale or color images

JBIG uses neighboring pixels as a template to statistically predict the current pixel X. As shown in Figure 5.8, JBIG uses a two-dimensional, three-line template

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

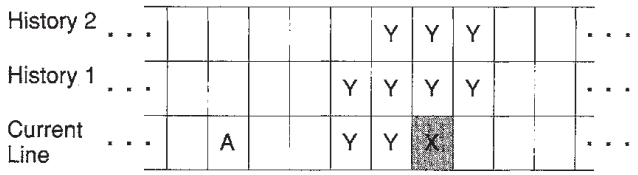


Figure 5.8 JBIG two-dimensional, three-line prediction template.

that includes neighboring *context* pixels (Y)—two from the current line, four from the history line, and three from the previous history line. It also uses an *adaptive* template pixel (A) whose position can vary, depending on the image being coded. As to function, the context pixels control the arithmetic coder decision probabilities and the adaptive template pixel provides more effective coding of halftoned images.

JBIG allows both sequential (nonprogressive) and progressive modes of operation. The sequential mode captures an image and codes it at full resolution. Besides providing slightly better compression than the progressive mode, the sequential mode can be somewhat faster because a simplified two-line template is a coding option. The progressive mode captures a low-resolution rendition image and, with a sequence of encodings, progressively doubles the vertical and horizontal resolution. Progressive coding can be advantageous for displaying images, such as situations where different displays have differing resolution capabilities. It is also useful for browsing a collection of images, where a low-resolution rendition is sufficient to screen out unwanted images, and for transmitting images over noisy channels, where part of the image may be corrupted.

Although the main objective of JBIG was to improve bi-level image coding, it can also be used to losslessly code (multilevel or multitonal) gray-scale and color images. This is accomplished by compressing the "bit planes" used to describe multitonal images coded with multiple bits per pixel. For instance, a gray-scale image composed of 8-bit pixels would be compressed by repetitively applying the JBIG algorithm, once to each of the 8 bits. For a color image, the JBIG algorithm would be applied to each bit of each color component.

JBIG compression performance, according to one source, improves bi-level image compression about 30% when compared to the MMR algorithm; on gray-scale images, JBIG is about equal to the JPEG algorithm (in lossless mode with arithmetic coding) [Penn93]. The results of extensive tests of the JBIG algorithm on both bi-level and multitonal images are provided by another source, where, on color images, very little difference was observed between JBIG and JPEG (again, in lossless mode with arithmetic coding) [Arps94]. However, compression results for individual documents may vary widely.

JPEG—ITU-T T.81 recommendation

JPEG is an international standard for compressing gray-scale and color, photographic-quality (continuous-tone) images. The JPEG acronym stands for the Joint

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Photographic Experts Group.[7] The intent of JPEG is to provide algorithms and tools for a variety of image compression applications. Implementations are flexible too; the JPEG algorithms are symmetric and they allow for software-hardware cost-performance trade-offs. Very quickly, JPEG has become the image coding standard by which all others are judged, and many of its compression techniques have found their way into standards for compressing video data (which will be examined in Section 5.5).

A motivation for developing JPEG was to promote the usage of color, photographic-quality images as commercially viable media. When uncompressed, they simply are too expensive to transmit or store. Consider the following: A typical digitized color image may use 24 bits per pixel—8 bits each for RGB (red, green, blue) or YC_BC_R (luminance, chrominance, chrominance) components.[8] The explosion in transmission times (and storage costs) for color, photographic-quality images is evident when compared to FAX images that use only 1 bit per pixel. Thus, the JPEG group set about developing algorithms that would offer quality image representation at 1.0 bits per pixel (a 24:1 compression ratio for the above example). JPEG accomplishes this (and more) by using both lossless and lossy compression, resulting in an image that is an approximation of the original. In fact, JPEG allows the user to select the compression ratio, with the understanding that higher compression ratios lead to poorer approximations and lower-quality images. JPEG allows compression of any continuous-tone image by as little as 2:1 with absolute visual image fidelity (using only lossless compression). It also allows these same images to be compressed by 10:1 with "visually lossless" image fidelity and by 20:1 or even 100:1 with lesser degrees of visual image fidelity (using lossless and lossy compression).

To reduce the number of bits used to represent a digitized image, JPEG takes advantage of redundancy and irrelevancy. We are familiar with *statistical redundancy*—the kind of redundancy that occurs in a sequence of symbols—that can be removed by lossless compression algorithms, such as Huffman and arithmetic entropy coding, which JPEG uses in its final stage of coding. But, in images there is another kind of rudundancy—*spatial redundancy*. Within images, neighboring elements for any one of the RGB or YC_BC_R matrices are highly correlated because all the pixels in a region of the image tend to represent the same color, luminance, or chrominance. Also, it should be noted that spatial redundancy increases with higher image resolutions (needed to capture detail)

[7] The JPEG committee was first chartered in 1986 by ISO and ITU-T. The JPEG standard is a collaborative effort of three major international standards organizations, ISO, ITU-T, and IEC. It is formally specified by the ISO Draft Information Standard 10918 | ITU-T Recommendation T.81 [Penn93].

[8] The Y luminance component provides a gray-scale version of the image. It is formed as a weighted sum of red, green, and blue color components. The C_B and C_R luminance components provide the additional information needed to convert the gray-scale image to a color image. They are formed by combinations of the luminance and color components [Penn93].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

because more pixels represent each region of the image. JPEG removes this type of redundancy, too, by using transform techniques to be discussed shortly.

To understand *irrelevancy* and how, through quantization techniques, JPEG uses it for image compression, we turn to psychovisual compression techniques that result from studies of the human visual system. Psychovisual compression takes advantage of several limitations of the human eye, including imperfect response to color. The human eye is less sensitive to some image components than others. Research has shown the human eye is less sensitive to chrominance than to luminance, meaning that fewer bits are needed for chrominance information, and it can be coded more coarsely than luminance. Another trick of psychovisual compression is to take advantage of the human eye being most sensitive to mid-spatial frequencies and not so sensitive to low- and high-spatial frequencies. This allows designers to make compromises in the fidelity of edge contours where rapid transitions in brightness occur. Also, the human eye is less sensitive to quantization distortions at high luminance levels, the so-called noise-masking property. This allows more coarse quantization to save bits at high luminance levels.

These data compression concepts are embodied in the JPEG standard for still, color, photographic-quality image compression. Figure 5.9 shows the baseline system, the most basic mode of JPEG encoding and decoding. To begin the process, the components of the digitized image are placed in a frame store. The JPEG algorithm operates on each component of the image. It produces a single compressed image for gray-scale images, one for each of the RGB color components or YC_BC_R luminance-chrominance components, or an appropriate set for whatever may be the chosen image representation. JPEG does not specify how the image should be represented, so any color transformation—say from RGB to YC_BC_R, where luminance-chrominance psychovisual differences can be best exploited—must precede the steps shown in Figure 5.9.

The JPEG encoding algorithm breaks each image component into blocks of 8×8 pixels and processes each block as follows:

- Spatial redundancy within the image is removed when waveform processing techniques are applied with DCT (discrete cosine transform) coding. The objective of transform coding is to decorrelate the image pixels; that is, statistically dependent image elements are converted into independent coefficients in a transformed space, where the energy of the image is concentrated onto as few coefficients as possible. This allows removing the irrelevant components within the image during the next step.
- Irrelevancy reduction occurs when lossy compression is applied and the DCT coefficients are quantized (encoded) according to whatever rules are loaded into the quantization table. Typically, larger step sizes are used to more-coarsely quantize any coefficient representing color data whose frequencies lie above some threshold (because the eye is not so sensitive); smaller step sizes are used to quantize those below the same threshold (because the eye is more sensitive). Often different values

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

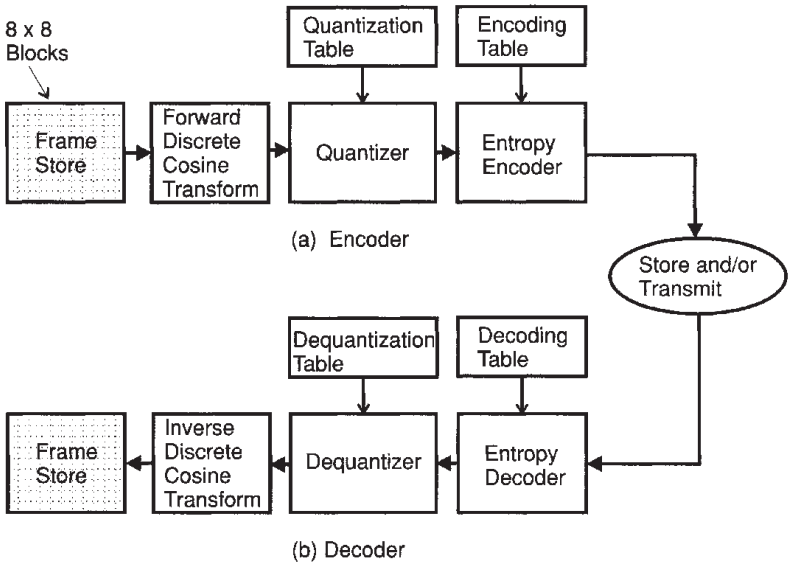


Figure 5.9 JPEG encoding and decoding.

are loaded in the quantization table for each of the RGB or YC_BC_R matrices, corresponding to the limitations of the human eye.

- Statistical redundancy reduction occurs when lossless compression is performed using run-length and Huffman entropy coding which is specified for the baseline system. Arithmetic coding may be selected for other modes of operation (to be described shortly), yielding approximately 10% more compression at the expense of execution time [Witt94].

The results from processing individual 8×8 blocks are combined to form the encoded image which may be stored or transmitted. By reversing the encoding operations, the JPEG decoder produces an approximate representation of the input image.

Figure 5.10 provides insight to how JPEG processes blocks of 8×8 pixels. The steps JPEG uses to process images reflect the state of the art in commercial image compression technology. Because these same steps appear in the video compression algorithms to be examined in Section 5.5, a closer look into their operation is warranted. For the example shown in Figure 5.10, we have chosen to transform the pixels for a small, shaded gray square set in a field of white.

The encoding process begins with the DCT step that transforms the two-dimension block of pixels from the spatial domain to a two-dimension array of frequency coefficients in the frequency domain.[9] The purpose of the DCT step

[9] Before the DCT step, the spatial domain pixel values are threshold shifted (by 128 for 8-bit pixels), becoming positive and negative values. Following the IDCT, this threshold shift is undone.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

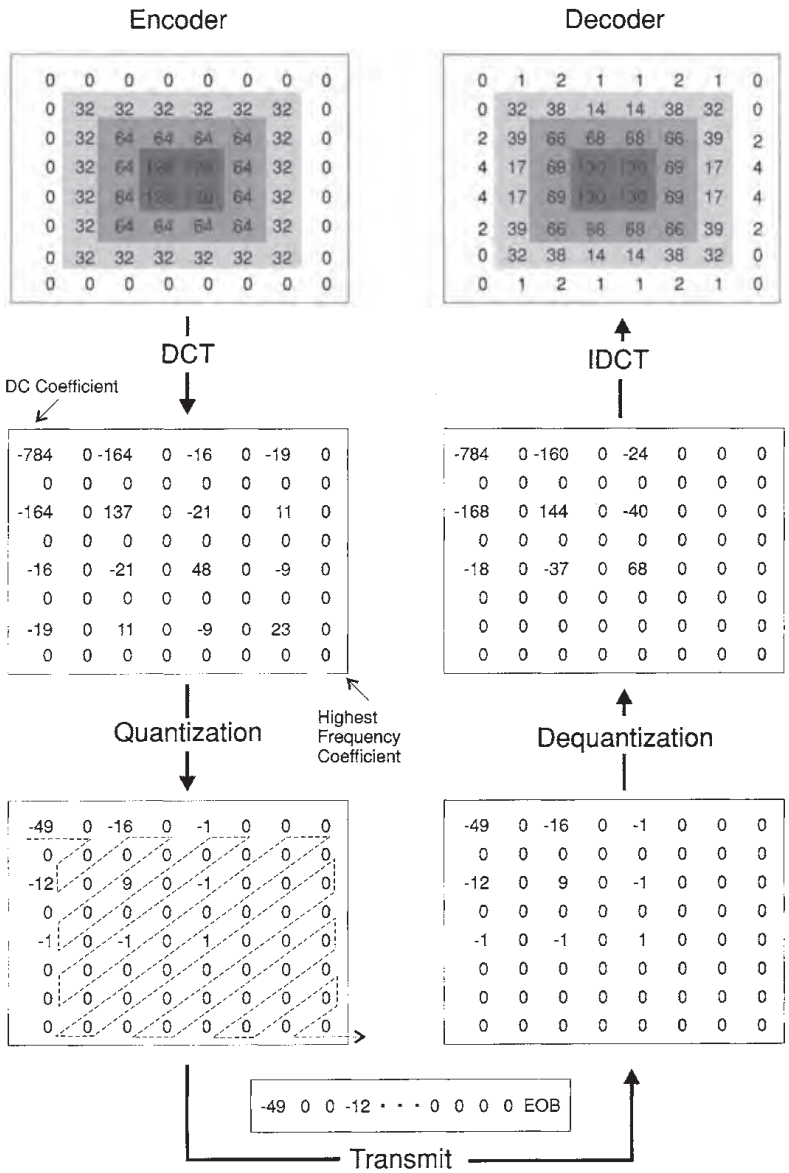


Figure 5.10 JPEG 8 × 8 pixel block coding.

is to decorrelate the 64 image pixels and concentrate most of their energy into coefficients in the top left corner of the frequency array, rendering the other coefficients to have near-zero values. Among the frequency coefficients, the "DC coefficient" represents the average intensity value (color, luminance, or chrominance) of the 64 pixels in the spatial domain. The other coefficients represent spatial features occurring at different frequencies in the 8 × 8 block of pixels. Within the frequency coefficient array, the lowest frequency coefficients

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

are located up and left; the highest are down and right. Moving down the array one finds coefficients representing higher vertical frequencies; moving right in the array, one finds coefficients representing higher horizontal frequencies. The terms "vertical frequency" and "horizontal frequency" refer to patterns found within the 8 × 8 block of pixels in the spatial domain—variations of intensity in the vertical and horizontal dimensions. It should be noted that DCT is a lossless operation because, if the coefficients are represented with sufficient accuracy, the 8 × 8 block of pixels can be recovered exactly.

The next step, quantization, is a lossy operation, after which only an approximation of the 8 × 8 block of pixels can be recovered. Quantization is designed to drive small, nonessential frequency components to zero by scaling each coefficient to the nearest multiple of a value found in a quantization table. To produce the quantized two-dimensional array of frequency components shown in Figure 5.10, a quantization table provided as an example by the ISO JPEG standard was used.[10] JPEG allows the user to decide what is nonessential by making the 64-element quantization table loadable. In general, quantization tables are constructed so that the quantization step size varies with frequency and image component. Low frequencies are more important than high, luminance is more important than chrominance, and nonessential elements are quantized with larger step sizes (i.e., quantized more coarsely). The zigzag sequence shown in the lower left corner of Figure 5.10 is used to read out the quantized frequency coefficients into a one-dimension array in preparation for transmission. This orders the frequency coefficients from lowest to highest, allowing run-length and Huffman or arithmetic entropy coding (not shown) to be applied for efficient transmission to the decoder.

In the decoder, using the same quantization table, the quantized coefficients are dequantized. Then, an inverse discrete cosine transform (IDCT) is applied, which allows an approximate representation of the 8 × 8 block of pixels to be reconstructed. The effects of quantizing the original small, shaded gray square are clearly visible in the reconstituted 8 × 8 block of pixels, shown in the upper right corner of Figure 5.10.

The baseline system, which all JPEG decoder implementations must support, is only one option that may be selected. The JPEG standard describes the following modes of operation:

- Sequential DCT based: This mode (which includes the baseline system) encodes each image component in a single left-to-right, top-to-bottom scan.
- Progressive DCT based: This mode encodes the image in multiple scans. It is appropriate for applications in which transmission time is long and the viewer prefers to see the image build up in multiple coarse-to-fine passes.
- Sequential lossless: This mode allows the decoder output to bit-identically match

[10] In the ISO JPEG standard, see "Table K.1 Luminance quantization table," as reproduced on page 503 of [Penn93].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

the encoder input, supporting applications where no loss can be tolerated. Predictive coding techniques are used.

- Hierarchical: This mode provides progressive coding with increased spatial resolution between successive stages. (Note: JBIG defines this to be its progressive coding mode of operation.) The image is encoded at multiple resolutions so that lower-resolution versions may be accessed without first having to decompress the image at its full resolution. The JPEG hierarchical mode allows DCT-based (lossy) coding, lossless coding, or both in combination.

The JPEG standard encompasses a diversity of techniques for compressing color, photographic-quality images and offers immense flexibility: Images may range in size from 1×1 to $65{,}535 \times 65{,}535$ active pixels. Each pixel may have from 1 to 255 color components or spectral bands (except the progressive mode where only 1 to 4 are allowed). Pixels can have 8 or 12 bits of precision for DCT-based coding and any value between 2 and 16 bits for the sequential lossless coding mode. The JPEG standard allows for interleaved data supporting, for example, still images captured from television. Besides defining the algorithms for image compression and decompression, the standard defines the compressed image data stream, including an interchange format for storage and transmission. More detailed descriptions of the JPEG standard and its many options can be found in several sources, including [Wall91], [Arav93], and [Witt94], and [Penn93], where the complete JPEG specification can be found.

State-of-the-art image coding and beyond

The image coding techniques examined in this section are in various stages of deployment. For bi-level images, the MH, MR, and MMR coding techniques specified by the ITU-T T.4 and T.6 recommendations are ubiquitous, being used in all FAX equipment. JBIG, which applies to bi-level images, too, and allows for coding multilevel, multitonal images, has not yet been widely deployed. Although these standards may not be end-all solutions for bi-level image coding, they are highly effective, and they achieve lossless coding results that are near optimum.

JPEG is *the* standard for coding still, photographic-quality (continuous-tone) images. Although there are *many* other image compression techniques, some of which offer superior compression ratios in some situations, none yet provide the balance of coding efficiency, implementability, and breadth of applications that JPEG provides. None has the widespread acceptance that JPEG does. Algorithms other than JPEG have been and will continue to be used for niche image compression applications. However, they do not appear destined to become new image compression standards, at least not until they are more than the equal of JPEG in all respects. Meanwhile, JPEG image coding is not stagnant; it is an extendable standard. Many advancements in continuous-tone, still-photographic image coding can be made through compatible extensions to the current JPEG standard.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Improved coding techniques for greater coding efficiency, wider ranges of image types, wider ranges of image quality, including handling noisy images, preprocessing and postprocessing—all are possible under the JPEG umbrella. A list of some proposed extensions can be found in [Penn93]. More recently, the JPEG committee is now defining new lossless algorithms to be included in future versions of the standard [Mitc96].

5.5 Video Compression

During the 1980s and early 1990s, compressed digital video made the transition from research topic to commercial products. Among all types of data, video is without question the most difficult to compress, and finding techniques that meet all the challenges of compressing video has not been quick or easy. Real-time operation, very high data rates, stringent transmission channel bandwidth limitations, compatibility with existing standards for television broadcasting and telecommunications transmission, a marketplace with an ever-increasing demand for higher picture resolution—all set high barriers for success. Today, thanks to advances in microelectronics and improved compression techniques, we have ITU-T H.261 and ISO MPEG, the first generation of video compression standards containing algorithms that are practical and effective for videoconferencing and television applications, respectively. We will examine these standards in this section, realizing that additional applications for compressed video are still being discovered, too, and some of these applications demand improved (or totally new) compression algorithms to support them. Without a doubt, new video compression standards will emerge in coming years, and this section will conclude with a glimpse of what to expect.

At first thought, video compression might not seem much more difficult than image compression. In one sense, the distinction between compression techniques for still and moving images—video—is thin: A sequence of video frames can be treated as a sequence of still images to be coded individually but displayed sequentially at a video rate. Actually, this is what some applications do—the ones that use the so-called M-JPEG (motion-JPEG) video compression algorithm. The only problem with M-JPEG is that although the video quality can be very high, compression is not. In contrast, H.261 and MPEG are specifically designed for low-bit-rate video coding. They exploit the characteristics of video to maximum advantage and do produce highly compressed video—but not without additional complexity. One of the things they do differently is to exploit the *temporal redundancy* (or temporal correlation) that all video sequences contain because persons or objects in the scene do not move very much from frame to frame. By extracting frame-to-frame temporal redundancy, H.261 and MPEG compress the video data stream an additional three times (three times more than M-JPEG)—at the same quality level.

The H.261 and MPEG standards, and the systems that use them, employ

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

temporal redundancy and many other techniques to compress video. Techniques to be discussed later in this section include the following:

- Color-space transformations, say from RGB to YC_BC_R where luminance-chrominance psychovisual differences can be exploited
- Picture resolution reduction (for applications where less than broadcast-quality pictures are acceptable)
- Frame rate reduction (for applications where less than full-motion video is acceptable)
- Intraframe coding using JPEG-like techniques to remove spatial redundancy (except the video algorithms allow for variable quantization)
- Interframe coding using temporal redundancy removal techniques
- Entropy coding techniques similar to those of JPEG

Analog video basics

To understand digital video, we need to look at how analog video systems operate. Consider television: A video camera captures a sequence of still pictures displayed one after the other as images on a television receiver. As shown in Figure 5.11, each picture is captured and converted to an electronic signal by a raster scanning process. Starting at the upper left corner and moving downward, picture information is captured in a sequence of scan lines that form a field. When the bottom of the picture is reached, scanning resumes at the top of the picture to form a second field, interleaved (interlaced) with the first. The two

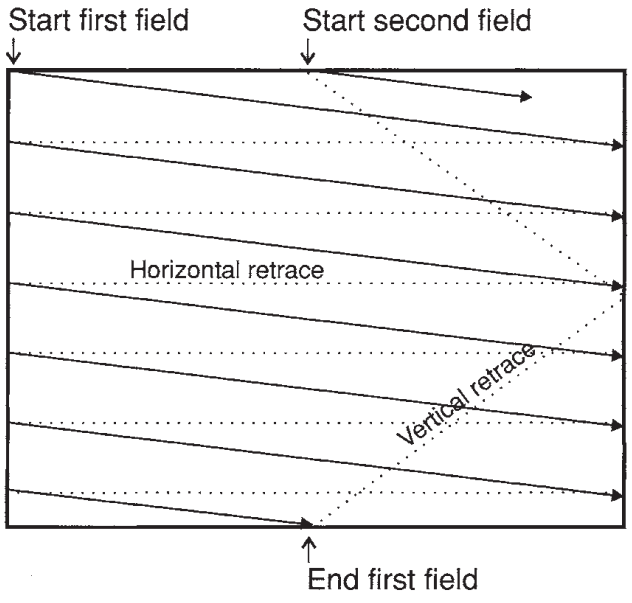


Figure 5.11 Interlaced TV raster scan.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

fields, together, make up the picture and are called a frame.[11] There also are video systems that progressively scan the picture, filling the frame sequentially left to right, top to bottom. Computer video monitors use this scanning method, as will some future HDTV (high-definition television) receivers. In either scanning method, using more scan lines increases the (vertical) picture resolution, whereas to increase the horizontal resolution (which in digital systems is measured by the number of pixels per scan line), more bandwidth must be used to transmit the analog video signal. The scanning process is repeated at regular intervals of time, establishing a frame rate for transmission, which is measured in frames per second (fps). The frame rate must be high enough to produce the effect of smooth motion for moving objects, but increasing the frame rate increases the transmission bandwidth too.

There are three analog television transmission standards in use throughout the world today: NTSC, PAL, and SECAM. NTSC is used in North America and Japan, whereas PAL and SECAM are used in European countries. The rest of the world shares these three standards on a country-by-country basis [Trow94]. Each standard specifies a different number of scan lines and frame rate. NTSC uses 525 scan lines and 30 fps (actually 29.97 for color), whereas PAL and SECAM use 625 scan lines and 25 fps. There are more differences, too, particularly in how the analog signals are encoded for transmission and the resulting horizontal resolution realized by these systems.[12] In the recent past, one of the challenges faced by applications such as videoconferencing—which must interoperate with these existing incompatible standards —has been to create a digital television transmission system that can bridge between them. The ITU-T H.320 suite of standards for videoconferencing, of which the ITU-T H.261 compression standard is a part, does this.

A fourth worldwide standard for television transmission—*digital* HDTV (high-definition television)—looms in the immediate future. NTSC, PAL, and SECAM provide television pictures at a resolution that in the 1990s world of digital television is called SDTV—standard-definition television. It is not clear whether there will be one standard for HDTV, another for SDTV, or even several standards for both, defined on a country-by-country basis. In any event, it (or they) will be digital, which simplifies matters somewhat.

Digital video systems

Figure 5.12 shows the elements of digital video transmission systems. Because digital systems represent picture information as a series of digital bits, the first

[11] Interleaving is a "trick" analog television systems use to conserve transmission bandwidth. It exploits psychovisual properties of the human eye, making the frame rate appear to be twice its actual value. This reduces the "flicker" that is visible when frames are displayed at too low a frame rate.

[12] The CCIR 601 recommendation (standard) for studio-quality, digitized NTSC television specifies 480 active lines per picture and 720 active pixels per line. Home television receivers can display

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

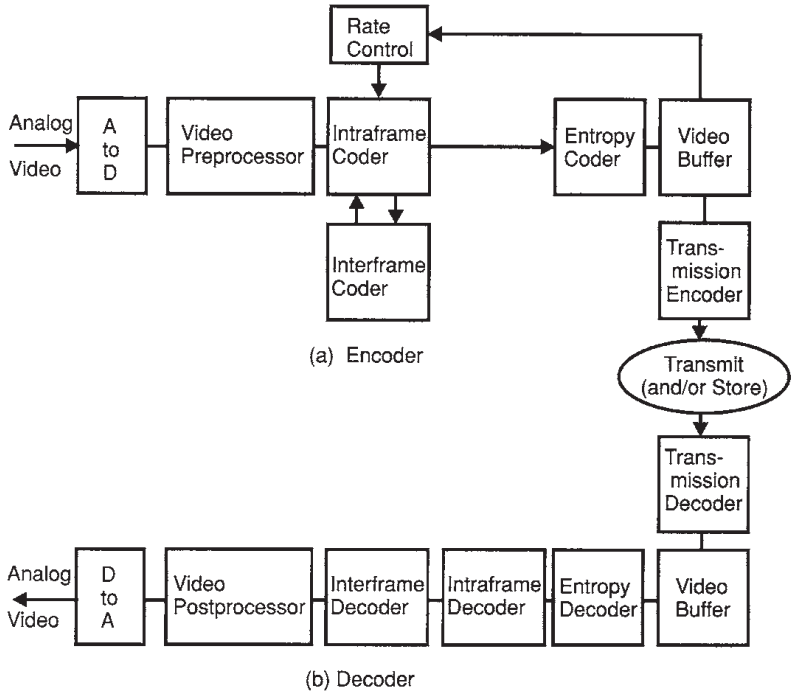


Figure 5.12 Digital video system dataflow.

step is to convert analog video signals to digital bit streams. Digital color television can be compressed most effectively when it is represented in RGB (red, green, blue) color component format or any of several luminance-chrominance component formats. These include YUV, YC_BC_R (a color coordinate system closely related to YUV), and YIQ (another luminance-chrominance format preferred for NTSC-standard video).[13] If the analog video signal arrives in one of these component formats, it can be directly converted to digital format. Or, if it arrives encoded into what is known as an analog composite format for transmission, it first must be decoded into analog component format and then converted to digital component format.

The next step is to preprocess the digital video in preparation for compression. The main purpose of preprocessing is to reduce the number of pixels to be coded and arrange the remaining pixels in a format most suitable for coding. The video preprocessor performs some or all of the following transformations:

all the lines but, thanks to the bandwidth-limited standard NTSC transmission channel, can reproduce only 300-350 pixels per line.

[13] The H.261 and MPEG standards specify YC_BC_R format. References to YUV format appear frequently in popular digital video compression literature, where the two formats are used interchangeably. For a comparison of the various color component systems, see [Penn93].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

- Conversion, if needed, to the color component format best suited for compression (usually YC_BC_R)
- Removal of digital bits collected during blanking intervals[14]
- Conversion from interleaved to progressive scan format (for compression algorithms such as H.261 and MPEG-1 that work only on progressively scanned pictures)
- Conversion from analog standard picture resolution to digital standard picture resolution (by downsampling or upsampling as appropriate)
- Subsampling the chrominance components (by downsampling in the horizontal direction, the vertical direction, or both to take advantage of the human eye's insensitivity to changes in color)
- Subsampling in the temporal direction to reduce the frame rate prior to coding[15]

The H.261 and MPEG video compression algorithms operate on individual pictures (frames) and share processing methods that are common to the JPEG still-image compression standard. The intraframe coding step in these algorithms processes each picture component with DCT transforms and lossy quantization techniques. This step is followed by an entropy encoder that does run-length and Huffman coding. Both steps are similar to those described for JPEG in Section 5.4.

The new element in video compression is the interframe coder. Its objective is to reduce the number of bits to be transmitted by finding the temporal redundancy that exists between frames in the video data stream. The most extensively used intraframe coding technique is motion-compensated prediction. The concept is this: The intensity (and color) of an object will change only slightly from frame to frame. Also, if the object moves, its motion is slight. Therefore, when coding a new frame, use frames that have already been encoded to predict what the object will be and where it will be. Instead of transmitting all the pixels representing the object, transmit pixel changes resulting from object change and motion—as a prediction error and a motion vector.

Figure 5.13 shows the simplest type of motion-compensated prediction, the type used by the H.261 algorithm, where the prediction is based only on the previous frame. In H.261, each frame to be interframe coded is broken into blocks of 16×16 pixels and only blocks from the luminance component are used for motion-compensated prediction. The previous frame is examined to locate the best-matching block, which may not be in the same position because of object motion. So, nearby blocks in the previous picture are also searched. When the best-matching block is found, it and the block in the current frame are differenced, DPCM (differential pulse code modulation) is applied to predict

[14] Blanking intervals are those times when the scanning process is moving between lines and fields within a frame (shown as horizontal and vertical retrace in Figure 5.11). Only the digital bits that represent active parts of the picture need be coded and transmitted.

[15] Frame rate subsampling is only used for applications where the resulting jerkiness in motion can be tolerated. Often, when frame rate subsampling is used, it is done dynamically by the encoder and only if there is no other option for meeting an immutable transmission bandwidth or storage constraint.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

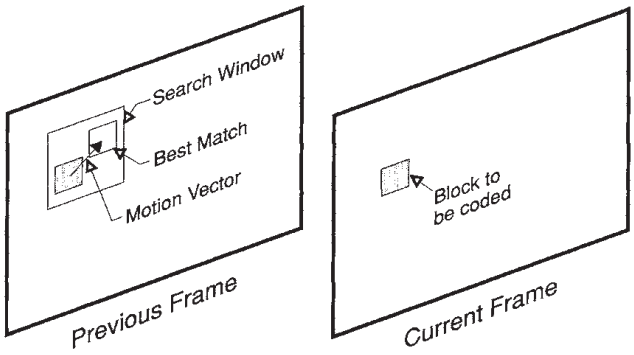


Figure 5.13 Motion-compensated prediction.

the block in the current frame, and a prediction error is generated. The prediction error along with a motion vector, which specifies the horizontal and vertical pixel-displacement of the block within the picture, are encoded for transmission.

The final elements for video coding (shown in Figure 5.12) are buffering, where the various picture components are stored while compression is in process, and a transmission encoding function, where the compressed pictures are assembled into a standard video data stream for transmission (or for storage). The decoding process, like other algorithms examined, performs inverse operations to restore an approximate representation of the original video data stream.

To date, three international standards have been defined for video compression, ITU-T H.261 and two MPEG standards by the Motion Picture Experts Group within ISO. All three are based on discrete cosine transforms, and all three support compression between frames as well as compression within each frame. The ITU-T H.261 standard is designed for videoconferencing over ISDN digital, switched circuits. ISO MPEG-1 was originally designed for playing multimedia from industry-standard CD-ROMs, with the fixed video data rate of 1.2 Mbps (1.5 Mbps total for audio and video). More recently, it has been used for direct broadcast satellite (DBS) transmissions and computer-generated video applications. ISO MPEG-2 is designed for higher-quality multimedia playback, for conferencing, and, most important, HDTV. Let us now look at these standards in more detail.

ITU-T H.261

The ITU-T H.261 recommendation is part of ITU-T H.320, a suite of recommendations for audiovisual terminals that provide videotelephone and videoconferencing services on IDSN (Integrated Services Digital Network).[16] H.320-com-

[16] H.320 actually is the third videoconferencing standard created by ITU-T. The first, H.120, was adopted by the ITU-T in 1984. It operated at 1.5 Mbps and could not compete with proprietary algorithms that offered better picture quality and lower data rates. The same fate afflicted N x 384, a mid-eighties effort that never became a formal standard [Port94, Halh91].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

pliant products provide low-latency, jitter-free, full-duplex communication of coordinated sound, video, and data over limited-bandwidth ISDN channels. H.261, which was approved in 1990, defines a standardized video bit stream and a decoding algorithm. It provides the low-delay, real-time video decoding (and encoding too) needed for face-to-face videotelephone and videoconferencing conversations.[17] H.261 is designed for transmission of color video at bit rates of $p \times 64$ Kbps, where p is an integer ranging from 1 to 30 (64 Kbps–1.92 Mbps). In practice, 64 or 128 Kbps ($p = 1, 2$) are appropriate for videotelephony, whereas rates of 128 Kbps and higher are needed for acceptable picture quality during videoconferencing.

Two digital image formats are defined for H.261, as shown in Table 5.4. When H.261 is used for videotelephony or videoconferencing, video signals from all equipment must be converted to one of these two formats. CIF (common intermediate format) is an optional higher-resolution format, whereas QCIF (quarter-CIF) is a standard lower-resolution format that must be supported by all H.261 implementations. The picture aspect ratio is 4:3 (horizontal:vertical). The frame rate can be 30 (actually 29.97), 15, 10, or 7.5 noninterlaced frames per second. CIF and higher frame rates apply to videoconferencing; QCIF and lower frame rates are appropriate for videotelephony.

Table 5.4 H.261 video formats.

Format	CIF		QCIF	
	Lines/Frame	Pixels/Line	Lines/Frame	Pixels/Line
Luminance (Y)	288	352	144	176
Chrominance (C_R)	144	176	72	88
Chrominance (C_B)	144	176	72	88

Selection of CIF or QCIF, and the frame rate, depends on the picture-quality objective and available channel bandwidth. When transmitted at the maximum frame rate, the uncompressed bit rates for CIF and QCIF are 36.45 and 9.115 Mbps, respectively. In practice, transporting these video signals on ISDN requires selecting the number of ISDN channels, the format, and the frame rate so that the compression ratio is 100:1 or less to retain reasonable picture quality—with available encoders. Table 5.5 shows various combinations of channels, formats, frame rates, and compression ratios.

[17] H.261 restricts encoding delay to be 150 milliseconds so as not to disrupt interactivity in two-way, face-to-face conversations. Longer delays produce effects similar to those experienced when telephone conversations are transmitted by satellite.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 5.5 H.261 compression ratios.

ISDN Bandwidth (Channels)	CIF				QCIF			
	30	15	10	7.5 fps	30	15	10	7.5 fps
64 Kbps (p = 1)	556:1	278:1	185:1	139:1	140:1	70:1	47:1	35:1
128 Kbps (p = 2)	278:1	139:1	93:1	70:1	70:1	35:1	23:1	18:1
384 Kbps (p = 6)	93:1	46:1	31:1	23:1	23:1	12:1	8:1	6:1

H.261 defines a data structure for encoding a sequence of pictures that allows a decoder to unambiguously decode the received bit stream. Each video frame is organized into a hierarchical block structure consisting of pictures, GOBs (groups of blocks), and macroblocks as shown in Figure 5.14. In turn, pictures are sequenced to form the compressed bit stream. The macroblock is the basic unit for encoding. It is composed of four 8×8 luminance blocks (Y) and two 8×8 chrominance blocks (C_B and C_R), where the chrominance blocks are formed by subsampling the chrominance pixels related to the corresponding luminance pixels.

Block diagrams for an H.261 encoder and decoder are shown in Figure 5.15. There are two modes of operation: intraframe coding and interframe coding. Intraframe coding is used for the first picture in a sequence; interframe coding is used for subsequent pictures in the sequence that are similar to the first picture. Intraframe coding removes redundancy within a picture. The steps for intraframe coding—DCT, quantization (Q), and entropy coding—are nearly identical to those for the JPEG algorithm, the primary difference being that the quantizer stepsize can be adjusted depending on the fullness of the encoder transmission buffer. H.261 does not specify how this should be done—it is an encoder implementation decision. In fact, only the decoding operation is completely specified by the H.261 recommendation. Encoder designers are free to make implementation decisions about when (or if) to use motion-compensated prediction, about how motion vectors are estimated, about controlling the encoder buffer rate, about controlling adaptive quantization to maintain uniform picture quality—about anything that does not corrupt the encoded bit stream or, thus, prevent the decoder from doing its job.

The second mode of operation, interframe coding, removes redundancy between similar pictures in a sequence. The technique used, motion-compensated prediction, was described in Figure 5.13 and the related discussion that appeared earlier in this section. In the H.261 implementation of motion-compensated prediction, only the previous frame is used for prediction, thus reducing the encoding delay. The previous frame is recovered (by Q^{-1} and DCT^{-1}) and stored in the motion-compensated prediction (MCP) memory. Also, during motion-compen-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

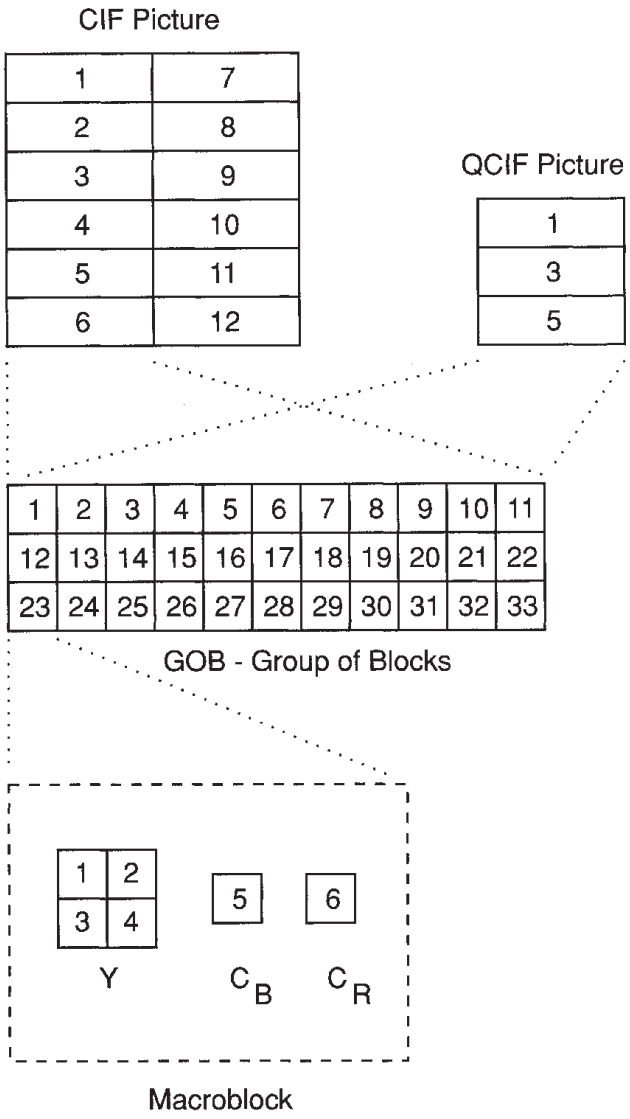


Figure 5.14 H.261 picture structure.

sated prediction, a loop filter may be used to improve picture quality by removing any high-frequency noise present in the picture. The encoder switches on the filter as needed. [To simply the encoder depicted in Figure 5.15(a), loop filter switching is not shown.]

The encoder chooses which mode of operation to use, intraframe or interframe coding, based on which will produce the fewest encoded bits. When the picture is changing rapidly, intraframe coding may give the best result, whereas for sequences of nearly similar pictures, interframe coding is preferred. Actually, there are several options the encoder may choose for interframe coding: If a macroblock has not changed or moved, no information need be transmitted. If

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

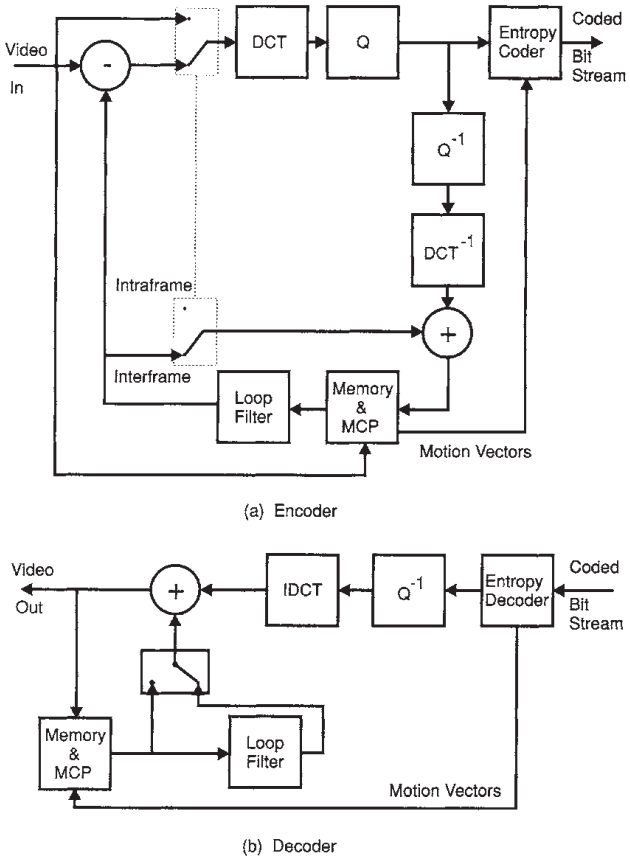


Figure 5.15 H.261 encoding and decoding.

it has moved but not changed, only the motion vector need be transmitted. If it has changed but not moved, only the pixel differences need be transmitted. If it has both changed and moved, then both pixel differences and a motion vector must be transmitted. Loop filtering is an option when pixel differences are transmitted and picture noise is present.

Although H.261 allows the encoder design to be flexible, there are requirements it must satisfy to assure the transmitted bit stream meets the standards set for decoding. To control error propagation, every macroblock must be intraframe coded at least once within every 132 transmitted picture frames. No more than 256 Kbps can be used to transmit each CIF frame (64 Kbps for QCIF). Thus, roughly 8 kilobits (2 kilobits) are the maximum allowed for each frame. As a result, the quantization step size may need to be increased during intraframe coding (or interframe coding be used). If neither can meet the transmission bandwidth constraint, then frame dropping is allowed.

H.261 is the most fully developed and widely deployed of today's digital video compression standards. Almost all room-size videoconferencing systems

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

provide H.261 as an option, allowing them to interoperate with other manufacturer's products. H.261 is also specified for some desktop videoconferencing products. The H.261 compression algorithm, as should be apparent from our discussion, is somewhat complex, and when first available, it was expensive to implement. For several years after its introduction, manufacturers interested in low-cost, one-on-one desktop videoconferencing sought to find lower-cost solutions. However, advances in VLSI have made the cost of H.261 less formidable and the lure of interoperation with an established market for room-sized systems has brought many desktop videoconferencing equipment manufacturers around to H.261.

Although H.261 has established itself as a standard for videoconferencing, it has known limitations: One is picture quality at low transmission bandwidth. Some proprietary algorithms from manufacturers of room-size videoconferencing equipment provide higher-quality video at 128 Kbps or less [Trow94]. Interestingly enough, as noted in Chapter 3, H.261 allows parties with the same proprietary algorithm to negotiate and use it, retaining the H.261 compression algorithm as a fallback mode while operating completely within the standard [Port94, Trow94]. Another limitation is picture quality at higher transmission bandwidth, where proprietary algorithms for large room-size videoconferencing systems offer more resolution [Trow94]. In response, a new ITU-T standard (H.262, which specifies MPEG-2 video) will allow high-resolution videoconferencing over ATM (asynchronous transfer mode) networks, where increased bandwidth is available [Schä95].

MPEG

In 1988 the Moving Picture Experts Group (MPEG) began working to develop international standards for digital audio and video transmission and storage. The outcome of this joint effort by ISO/IEC has produced MPEG-1 and (together with ITU-T) MPEG-2. Soon, these standards will be joined by a third, MPEG-4. Just as H.261 is designed for particular applications and transmission media, the MPEG standards, as shown in Table 5.6, are too. MPEG-1 was designed for playback of stored multimedia and, by performing significant between-frame compression on limited-resolution pictures, aimed to get visually acceptable picture quality at the 1.2 Mbps video data rate provided by CD-ROM. MPEG-1 has also proved useful in other applications, such as for early DBS (direct broadcast satellite) transmissions and for computer-generated video, where transmission bandwidth and storage capacity are limited or expensive. In contrast, MPEG-2 is intended for both playback and conferencing, providing considerable flexibility in the amount of between-frame processing and the use of much higher-resolution pictures—at higher data rates. MPEG-2 can produce the video quality needed for multimedia entertainment piped to the home and for more demanding business and scientific applications too. MPEG-2 also supports the picture resolution and quality needed for HDTV. MPEG-4, a standard still in the making, has

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 5.6 MPEG Parameters.

	MPEG-1	MPEG-2	MPEG-4
Final draft	1992	1995	1998
Data rate	≤ 1.86 Mbps[1]	≥ 4 Mbps	≤ 64 Kbps
Resolution - maximum	720 x 576[1]	1920 x 1152	na
- typical	352 x 240	Varies by application	na
Scanning method	Progressive	Progressive / interlaced	na
Frame rate - maximum	30 fps[1]	60 fps	na
Coding methods	DCT and BMCP[2]	DCT and BMCP	DCT and BMCP ? Object oriented ?
Applications	< SDTV-quality video storage on CD-ROM Computer-generated video < SDTV-quality video transmission	SDTV → HDTV-quality storage and transmission	Video on PSTN and mobile networks Video on low-capacity storage devices

Notes:
[1] MPEG-1 constrained parameter set maximum values [Chia95].
[2] Discrete cosine transform and bidirectional motion-compensated prediction.

a different set of objectives—to provide video on low-bandwidth transmission links or for low-capacity storage devices where no existing standardized video compression algorithm has proved satisfactory.

Each MPEG standard defines the syntax of an audio-video bit stream for storage or transmission and specifies both audio and video decoding algorithms.[18] Very much like H.261, but for different reasons, the video encoding approach is not completely specified. This allows trade-offs to be made between encoding time and compression efficiency or image quality. Asymmetric algorithms (complex, slow encoding and simple, fast decoding) are highly appropriate for most MPEG applications, which involve one-way communications—broadcasting or storage for delayed presentation. Here real-time encoding is not always needed, whereas improved compression ratios and improved image quality always are, and increasing the encoding time delay (and complexity too) may be a wise choice. From an economic viewpoint, asymmetric algorithms make sense because there are few encoders but many decoders for most one-way communications applications.

[18] MPEG audio was described in Section 5.3.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The MPEG standards (excluding MPEG-4 which is still in development and whose details are not finalized) can be considered the outgrowth of H.261, and JPEG too. The MPEG standards use variations of the same data compression techniques, and the details will not be repeated here. Instead, let us concentrate on the major innovations MPEG introduces beginning with the interframe compression technique.

Interframe coding and how compression is achieved through temporal redundancy removal are shown in the MPEG model for interframe coding depicted in Figure 5.16. MPEG-1 images are organized in groups of pictures (GOPs) that may be transmitted in an order different than the display order. GOP is an option for MPEG-2. Within a GOP, there are three types of pictures: intraframe pictures (I), predicted pictures (P), and bi-directional pictures (B). Intraframe pictures are complete pictures coded in isolation with DCT and quantization techniques similar to those used by H.261 and JPEG to remove spatial redundancy. Intraframe pictures provide random access points into the video stream, but they lead to only moderate compression. Predicted pictures are interframe coded with reference to the closest past I or P pictures. Like H.261, motion-compensated prediction is used for interframe coding. Predicted pictures provide more compression and serve as references for bi-directional pictures. MPEG departs from H.261 in using motion-compensated interpolation technique to perform bidirectional prediction for coding B pictures (those between I and P pictures). B pictures are coded using both past and future pictures as reference. They provide the most compression and, because they are never used as a reference, they do not propagate errors. B pictures allow MPEG to improve the quality of the compressed picture stream and to increase compression beyond that provided with H.261.

The MPEG standard allows the encoder to choose the frequency of occurrence

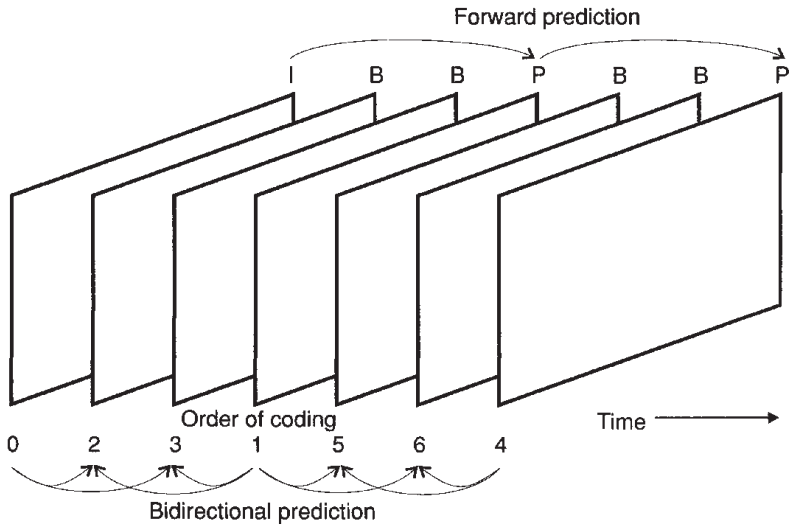


Figure 5.16 MPEG group of pictures.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

of I, P, and B pictures based on application-specific needs for random accessibility, coding delay, visual image fidelity, and—most important for video transmission and storage applications—compressed video stream bit rates [Schä95]. Coding the bit stream with only I pictures improves random accessibility and editibility but achieves low compression. Coding with only I and P pictures achieves moderate compression while still allowing random access and fast-forward/fast-reverse searches through the encoded bit stream. Using I, P, and B pictures provides the most compression, as B pictures can be coded in very few bits, leaving more bits for I or P pictures. It is advantageous to use more B pictures but, as can be deduced from Figure 5.16, there is a downside. When the number of B pictures in a GOP is increased, the random access points into the video stream grow further apart and the encoding (and decoding, too) delay time grows longer, characteristics that many applications cannot tolerate. An implementation detail is that an entire GOP must be buffered to allow I, B, and P pictures to be resequenced and played back in the correct order; increasing the number of B pictures in a GOP increases the buffer size, a characteristic that many applications cannot afford. Movement within the video scene also affects the number of B pictures that can be used, necessitating the use of adaptive techniques to vary how many B pictures can be coded between I and P pictures [Chal95].

The MPEG bit-stream syntax is more flexible than that of H.261, reflecting the wider variety of applications anticipated for MPEG. As shown in Table 5.7, it is constructed in several hierarchical layers. Further details can be found in [Arav93, Chen93].

MPEG-2 is an algorithms toolset and also a set of performance specifications for various operating environments. MPEG-2 is a superset of MPEG-1 with the

Table 5.7 MPEG bit-stream syntax.

Syntax	Function
Video sequence layer	Global context unit: video frame size, frame rate, bit rate, minimum decoder buffer size, constrained parameters information
Group-of-pictures layer	Video coding unit: random access, search, editing
Picture layer	Frame coding unit: type (I, B, P), display-order position
Slice layer	Resynchronization unit: a horizontal or vertical string of macroblocks used for resynchronization during decoding
Macroblock layer	Motion compensation unit: a 16 x 16 block
Block layer	DCT unit: an 8 x 8 block

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

requirement that every MPEG-2 compatible decoder must also decode valid MPEG-1 bit streams. MPEG-2 introduces new coding features not found in MPEG-1 to accommodate the functionality and quality of an expanded range of applications. Enhancements include [Chia95, Schä95] the following:

- A choice of picture scanning methods, interlace and progressive scan, to fit television-oriented and computer-oriented applications
- A choice of chrominance signal encoding methods
- Two bit streams are defined, transport and program, to support transmission and storage media applications
- Spatial and temporal scalability is supported

The macroblock is the basic unit for MPEG-2 encoding. Each frame is composed of nonoverlapping macroblocks. Each macroblock is made up of four 8 $\times$ 8 luminance blocks (Y), one, two, or four 8 $\times$ 8 C_B chrominance blocks, and an equal number of C_R chrominance blocks. The chrominance blocks are formed by subsampling the C_B and C_R pixels related to the corresponding Y pixels. This results in the three different MPEG-2 chrominance signal encoding methods shown in Figure 5.17. The 4:2:0 format provides MPEG-1 and H.261 compatibility. The 4:2:2 and 4:4:4 formats are provided for future higher-resolution applications such as studio video coding [Anas94, Schä95].

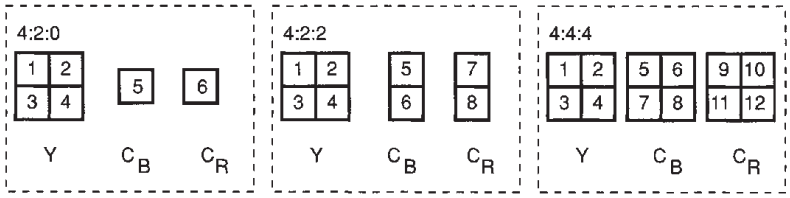


Figure 5.17 MPEG-2 macroblock structure.

The most expansive addition is support for spatial and temporal scalability through a set of "profiles" and "levels" which are shown in Table 5.8. In the spirit of "one size does not fit all," the intent is to allow equipment to be designed for interoperation at the profile and level that best matches an application. The profiles define the degree of functionality from no B frames (simple profile) to fully scalable video (high profile), with the main profile considered the most important profile having immediate application for cable, DBS, and SDTV/HDTV broadcasting. In general, moving left to right in Table 5.8, each successive profile includes new algorithms that build on those found in less functional profiles. Each profile is further divided into four levels which define various image resolutions. Within a profile, at any level, an MPEG-2 compatible decoder must decode bit streams from lower levels. As indicated in Table 5.8, only some profile-level combinations are expected to be used for known, practical applications.

If we take a closer look at the MPEG-2 profiles, we find that the simple profile

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 5.8 MPEG-2 profiles and levels.

Level[1]	Profiles				
	Simple (No B Pictures) (4:2:0)[2]	Main (Nonscalable) (4:2:0)	SNR-Scalable (4:2:0)	Spatially-Scalable (4:2:0)	High (Fully Scalable) (4:2:2)
High - HDTV[3] 1920 x 1080 60 fps 80 Mbps	—	✓	—	—	✓
High - HDTV[4] 1440 x 1080 60 fps 60 Mbps	—	✓	—	✓	✓
Main - SDTV[5] 720 x 480 30 fps 15 Mbps	✓	✓	✓	—	✓
Low - SIF[6] 352 x 240 30 fps 4 Mbps	—	✓	✓	—	—

Notes:
[1] Maximum values for active pixels, frame rate, and transmission bandwidth are shown [Schä95].
[2] YC_BC_R luminance-chrominance representation.
[3] The U.S. Grand Alliance HDTV system specifies 1920 x 1080 pixels interlaced scan at 60 fps and progressive scan at 30 or 24 fps using the main profile [Hopk94].
[4] The U.S. Grand Alliance HDTV system specifies 1280 x 720 pixels progressive scan at 60, 30, and 24 fps using the main profile [Hopk94].
[5] SDTV (standard definition television) CCIR 601 studio standard compatible - 4:3 (and 16:9) aspect ratio.
[6] SIF (source input format) MPEG-1 compatible - 4:3 aspect ratio.

is a subset of the main profile where there are no B pictures. The main profile is an extension of MPEG-1 to include fields and frame pictures for interlaced video. The remaining three profiles (SNR scalable, spatially scalable, and high) deal with interoperation between video services and products where different bandwidth, spatial resolution, or temporal resolution capabilities must be bridged. The SNR scalable profile provides tools for quality scalability allowing, for example, recognizable video to be transmitted over limited-bandwidth channels. The spatially scalable profile addresses supporting displays with various resolutions. For instance, there would be no point in displaying an HDTV signal at

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

full resolution on a small-screen receiver used for casual viewing. Here, SDTV resolution would be appropriate. The high profile supports full scalability, including temporal scalability where different frame rates can be bridged. Further details of MPEG-2 scalability can be found in [Chia94].

Together MPEG-1 and MPEG-2 are *the* video coding standards for television-based applications in the 1990's. The original application for MPEG-1 (storing movies for playback from CD-ROM) had but a brief moment in the limelight, and in its second application (DBS compression), it was quickly replaced by MPEG-2. But MPEG-1 has moved on, proving its merit for a variety of multimedia computer applications where quality video at moderate resolution is acceptable, low data rates are desirable, and inexpensive encoders and decoders are essential. And, thanks to both hardware and software implementations, MPEG-1 is available to all who choose to use it. For entertainment television, MPEG-2 has widespread endorsement, both from the standards-dominated communications industry and from the consumer-electronics industry that will build receivers for SDTV and HDTV applications. Mass production will assure a variety of low-cost MPEG-2 decoders, but it is unclear whether MPEG-2 encoders will be mass marketed or remain rare and expensive, as they will if MPEG-2 is used only for broadcast applications.

State-of-the-art video coding and beyond

Real-time video compression is leading-edge technology today. Although H.261 is a mature standard, widely deployed in commercial videoconferencing products, there is need for extending videoconferencing to networks other than ISDN. For general digital video applications, the MPEG algorithms and the MPEG standards are new and they are still being refined (or defined) to fit the products that use them, which are new too. Thus, there are opportunities and a need to advance video coding beyond the level set by H.261, MPEG-1, and MPEG-2. Here are some examples:

- Simplified encoding is needed for multimedia computing, digital VCRs, and other applications doing low-cost video source production. For these applications, symmetrical real-time compression is essential. M-JPEG and H.261 are more symmetrical (than MPEG) but do not always provide sufficient compression. Nonstandard, proprietary compression algorithms, some widely acclaimed as solutions for multimedia computing, may be more symmetric and may provide sufficient compression. But the problem is that they are not recognized as standards outside the multimedia-computing industry, particularly by industries that control the production of video source material.
- Economical frame-accurate random access is needed for digital VCRs. MPEG provides this function but at the cost of extensive buffering for groups of pictures.
- Preprocessing is needed to adjust the image before compression, not only to scale

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

the image resolution but also to remove noise and provide whatever data-filter operations are necessary to make the compression process operate more efficiently.

- Postprocessing is needed to reduce the undesirable aesthetic effects of artifacts introduced by compression. This can be as important as the compression process itself, but it has received little attention.
- Better performance is needed at low bit rates. When operated at their lower design limits, today's DCT-based algorithms provide a picture with noticeable "blockiness." When an inadequate number of bits are available for transmitting DCT coefficients, blockiness artifacts make the discontinuities in gray-scale values between 8×8 blocks highly visible.

Of these, the most pressing need is for improved picture quality at lower bit rates because many "hot" applications for video have bandwidth or storage limitations far more restrictive than what current standards can deal with. In the brief history of video coding, the quest for ever-lower bit rates has been never-ending, as Figure 5.18 illustrates for the oldest, best-developed digital video application, videoconferencing. The progress in video coding for moderate-quality videoconferencing (QCIF resolution—176×144 pixels) has been nothing short of remarkable. Initially, intraframe coding and, later, the combination of intra-frame and interframe coding reduced the coding bit rate by nearly 30 times. But videoconferencing products and videoconferencing standards have been stuck at 112/128 Kbps for several years. With renewed interest in the most personal of all videoconferencing applications, videotelephony over analog telephone lines, and new interest in transmitting all kinds of video over mobile radio communications networks, it is clear there exists a class of applications that H.261 (or

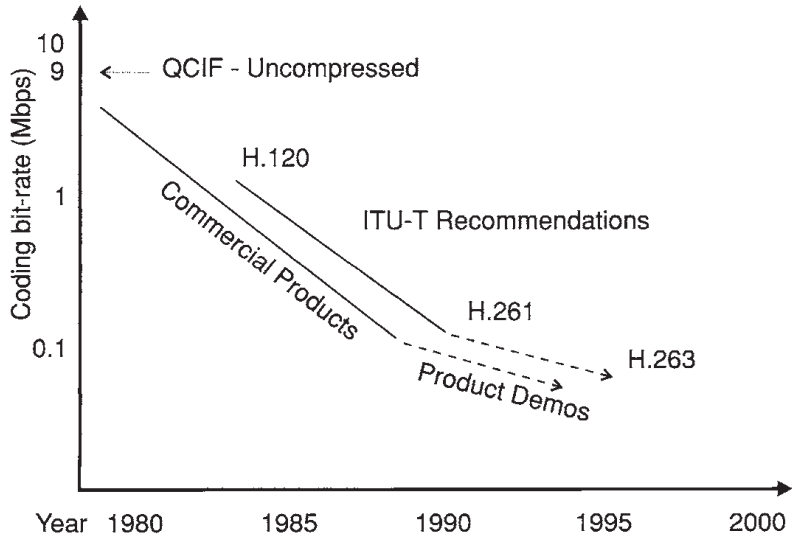


Figure 5.18 Videoconferencing bit rates.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

MPEG-1 or MPEG-2) cannot serve. H.261 simply is not applicable to analog PSTN (public switched telephone service, i.e., POTS—plain old telephone service), where even 64 Kbps is problematical, and mobile radio communications networks—they are digital but offer even less bandwidth.

Recognizing a demand for this service, ITU-T has initiated efforts to develop new standards for video coding at rates of 64 Kbps or less. The first result of their efforts is the H.324 suite of recommendations that includes H.263, a new video compression algorithm designed to outperform but interoperate with H.261 [Port94, Mess95B]. The purpose of H.324 is to provide an international standard for interoperation of videotelephones connected to the PSTN, where, today, several noncompatible products exist. It also applies to interoperation of desktop videoconferencing computers connected to the PSTN. ITU-T is also working on a yet-unnamed standard, to be introduced in the 1998 time frame, that will provide for video transmission via mobile radio networks as well as the PSTN and provide more advanced performance [Scha94]. ISO is also focusing on a new standard for the 1998 time frame. ISO MPEG-4 will operate in these same environments and serve a wide range of applications, including videotelephone via PSTN and mobile radio. It will also address telecommunications, computer multimedia, and remote sensing applications that require a high degree of interactivity, interoperability, and flexibility. Included are communications with mobile experts, emergency crew communications, educational networks for remote areas, interactive editing and playback of audiovisual material, networked games, security and surveillance systems monitoring, and more [Scha94, Schä95].

All these new standards will employ "low-bit-rate video coding" techniques. The definition of what constitutes a low bit rate is a moving target, usually meaning any value lower than what existing algorithms can provide. In 1980, 1.5 Mbps was low-bit-rate video; today, it is 64 Kbps with mention of 10 Kbps as a future target. To reach 64 Kbps, or even lower, new approaches are required. Picture resolution and frame rates can be decreased, giving fewer pixels to process, but the negative impact on applications is severe; therefore, this is not a general solution.[19] The direction that H.263 will take is to improve upon existing inter/intraframe techniques: Smaller motion vector search windows for more precise prediction and refined motion vector quantization will reduce the number of bits to be transmitted. Optimizing entropy coding will help too. But long-range solutions mean moving beyond coding techniques based on blocks of pixels and their motion. To improve coding rates by an order of magnitude, which is what must happen to go below 64 Kbps, a new generation of coding techniques is required. These coding techniques most likely will deal with video content as

[19] Experience with current video coding algorithms teaches that about 25K luminance pixels per frame and about one-fourth this value for the chrominance pixels (i.e., QCIF resolution) at frame rates that approach 30 progressively-scanned frames per second are needed to achieve acceptable quality video for general applications [Ebra95].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

something humans would recognize, rather than treating it as blocks of pixels. In Chapter 15, we will return to this subject and learn more about the coding techniques that future video compression algorithms will use.

5.6 Alternative Image and Video Compression Methods

Our introduction to compression algorithms for diffuse data has concentrated on the coding techniques used by the official standards. Many real-world applications do not use these coding techniques and standards, and no discussion would be complete without considering some of the alternatives. In today's marketplace, there are situations where nonstandard, proprietary compression algorithms are *the* de facto standards. This is most evident for image and video compression where the alternatives to official standard compression algorithms are numerous. There are many reasons this situation exists. The proprietary algorithms may have been there first. There may be marketing advantage in using them. They may be easier to implement, faster, and of lower cost. Most important, they genuinely may work better than the official standards. In this section, let us briefly consider three of the coding techniques frequently used by de facto standard compression algorithms. All are lossy compression methods that apply to images and video and, sometimes, other diffuse data types.

Vector quantization

Vector quantization compresses an image by replacing each block of pixels with an index into a codebook (or dictionary). Usually, the blocks do not overlap and all are the same size and shape, but they need not be. Within each block of pixels, each pixel value is treated as a component of a multidimension vector for reasons that will be discussed momentarily. To compress the image, as shown in Figure 5.19, the encoder cycles through its codebook and finds the stored

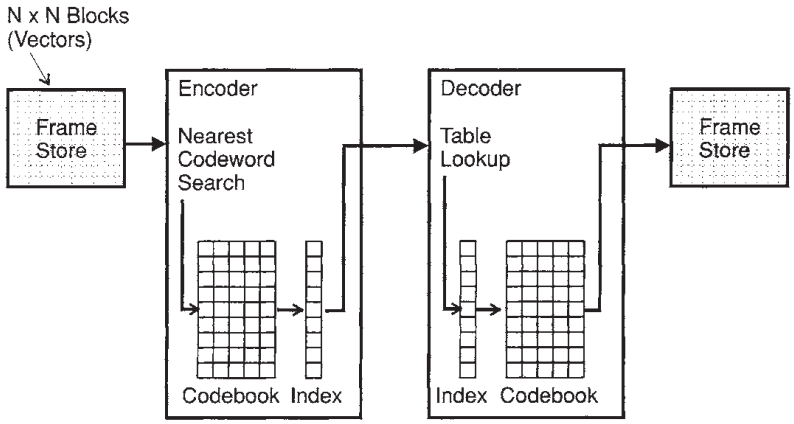


Figure 5.19 Vector quantization encoding and decoding.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

codeword (or code vector) that best approximates each input block. The elements of this code vector are the quantized values of the pixels. When the match is found, the encoder transmits the corresponding index (codebook address) to the decoder. To decompress the image, the decoder maintains an identical codebook. The decoder reconstructs the image by looking up each index in its codebook and outputting the corresponding codeword. Because the compressed image is represented by indices, the compressed representation requires fewer bits. For example, assume each input block contains 4×4, 8-bit pixels and there are 16 codewords in the codebook. Because each codeword requires a 4-bit index, the compression ratio is 32:1.[20]

Does this process seem familiar? It should, because LPAS speech codecs use the same technique. (See Figure 5.2 and related discussion in Section 5.2.) Also, just as with speech coding, the difficult part of using vector quantization for image (and video) coding is finding efficient ways to search the codebook and deciding what codewords should be stored in the codebook. These issues are beyond the scope of this discussion, but extensive treatments can be found in several sources, including [Nasr88, Cosm93, Sayo95].

Vector quantization offers several advantages: The obvious one is that decoding is extremely simple, requiring only a table lookup operation. This makes vector quantization a favorite technique for software decoding and all applications where limited resources and time is available for decoding. Another advantage of vector quantization is that the quantization operation is more efficient than scalar quantization. All the previously described algorithms use scalar quantization, working on one component at a time. Vector quantization groups several components and encodes them as a single block. To appreciate why this is more efficient, consider that lossless Huffman coding is less efficient than arithmetic coding which groups symbols and encodes them as a single number. The same principle applies to lossy quantization, where, for a given bit rate, the result will be higher-quality images or video.

The illustration of vector quantization in Figure 5.19 indicates that it operates directly on the image pixels. Although this is how vector quantization is often used—because it is simple and effective—vector quantization can be combined with other signal processing techniques [Nasr88]. For example, one could use DCT or another form of transform coding and then quantize the transformed coefficients more efficiently with vector quantization. The result is better picture quality at lower bit rates. Another example is applying vector quantization for efficient video compression, a technique employed by several PC software video compression algorithms described in Section 9.6. When interframe coding is required, it is possible to develop vector quantizers that work in three dimensions on both intraframe and interframe components.

[20] In most vector quantization applications, the compression ratio is further increased by applying entropy coding to the codewords before transmission.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Wavelet coding

Basically, wavelet coding is a variant on DCT-based transform coding that reduces or eliminates some of its limitations. The name "wavelet" comes from the special localized *basis functions* used to break down an image (or other data) into its essential details. By using bursts of short-duration waves, rather than the continuous sinusoids used by the DCT, wavelet coding can more efficiently represent small features in an image such as the rapid transitions that occur at object edges. Another advantage is that rather than working with 8×8 blocks of pixels, as do JPEG and other block-based DCT techniques, wavelet coding can simultaneously compress the entire image. Thus, a wavelet-compressed image never suffers from the blocking artifacts apparent when block-based DCT techniques are pushed to high compression ratios. Taken in combination, these factors allow wavelet coding to offer higher compression ratios for a given image quality level. In some image compression applications, compression ratios as high as 300:1 are possible, several times greater than for JPEG and other DCT-based algorithms [Edge95A].

A two-dimensional spatial discrete wavelet transform (DWT) is used for coding digital images. Most often, this transform is used for coding video (as a sequence of images), but there also are three-dimensional DWTs. Essentially, DWTs are subband coding systems [Sayo95]. Therefore, as shown in Figure 5.20(a), the first step for compression is to process the entire image with a bank of filters conceptually similar to those used by the audio subband coders described in Section 5.3. The wavelet filters separate the image into a series of subbands of logarithmically increasing spatial frequency. The filter outputs create a set of transformed images at different resolutions that can be efficiently coded. The lowest-frequency subband contains a reduced-resolution, blurred copy of the

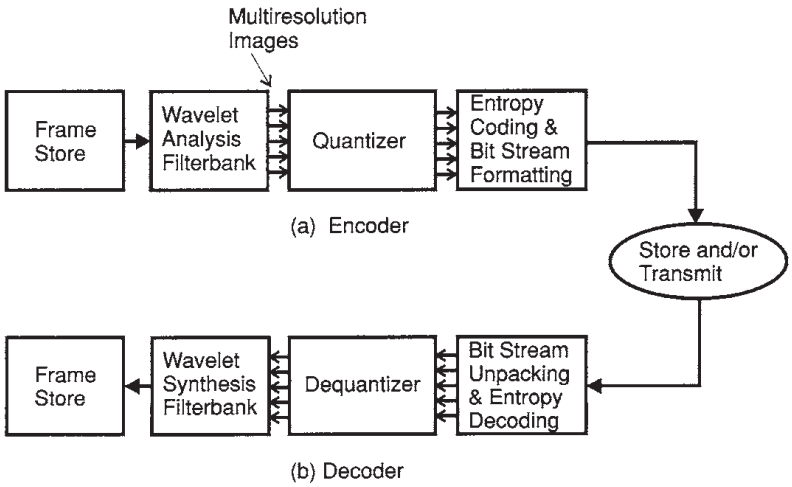


Figure 5.20 Wavelet encoding and decoding.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

original image. The higher-frequency subbands contain edge information needed to reproduce the fine detail of the original image at successively larger resolutions. Because of the way humans perceive spatial detail, the wavelet coefficients for consecutively higher-frequency subbands can be quantized more coarsely. Either scalar or vector quantization can be used [Anto92]. The final step is to combine the quantized coefficients from all subbands and then entropy code them with run-length and Huffman or arithmetic coding. By reversing the encoding operations, the wavelet decoder shown in Figure 5.20(b) produces an approximate representation of the input image.

One of the advantages of wavelet coding is that it quite naturally supports progressive transmission and decompression of images; that is, by simply ordering the compressed image file, an image can first be displayed at low resolution and fine detail can be added as more image subbands are transmitted. The multiresolution characteristics of wavelet-compressed images can also used to advantage for wavelet-compressed video. When storage or transmission limits are encountered, image resolution can be temporarily reduced to produce slightly "softer" images. This, a form of quality-level scaling, is easily accomplished by dropping one of more of the subbanded images. In contrast, under the same conditions, those DCT-based compression techniques which do not use progressive transmission may deliver images with blocky artifacts or must resort to more drastic measures such as frame dropping.

Another advantage is that wavelet coding is a symmetrical coding technique which can be implemented either in software or hardware. Although often judged superior to DCT-based techniques in this and many other ways, wavelet coding is not a standardized technique. As such, it has found its way into only a few applications. These include PC software video compression (described in Section 9.6) and law enforcement applications including a fingerprint identification system implemented by the FBI (U.S. Federal Bureau of Investigation) [Hunt93].

Fractal coding

Fractal coding provides yet another alternative to DCT-based transform coding, one that can provide resolution-independent decoding and higher compression ratios at the expense of far more complex encoding. The term "fractal" comes from a Latin word meaning broken or irregular fragments. Fractals are shapes or pieces of images that can be described by mathematical formulas. Historically, fractals were first used to generate natural-looking images having the interesting property that they can be shrunk or magnified without loss of detail. The same mathematics is used to analyze an image and create a compressed representation that can be decompressed into resolutions which are both lower and higher than the original image. Fractal coding is a block-based technique, and although fractal-coded images may display blocking artifacts, in practice the compression ratio can be pushed to higher values that for images coded with block-based DCT coding. Although, in some situations, fractal coding can achieve compression

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

ratios up to 10,000:1, results on real-world images suggest it is about three times more efficient than traditional block-based DCT (or vector quantization) algorithms at comparable quality levels [Jacq93, Boyd95].

The basic idea of fractal image compression is to exploit the self-similarity in an image by modeling a picture as several smaller pictures of itself. As shown in Figure 5.21, block-based fractal image coding accomplishes this in the following way: To compress an image, first, a pool of domain (D) blocks is created. These blocks will act as prototypes to represent the entire image. Then the image is broken into a set of nonoverlapping range (R) blocks. Typically, domain block sizes are 16×16 pixels and range blocks are 8×8 pixels. The block sizes are chosen to efficiently compress large, smoothly varying regions of the image. Also, each range block may be split into up to four nonoverlapping subblocks to capture fine detail in complex areas such as at object edges. For each range block, one by one, domain blocks are selected from the pool and through a series of transformations, known as affine transformations, are mapped to the range block. These transformations scale, translate, and transform pixels within the domain block (through luminance shifts, rotations, and so on). This process continues until a transformed domain block is found that closely approximates the range block. The list of block transformation parameters, known as the fractal code, is then transmitted to the decoder.

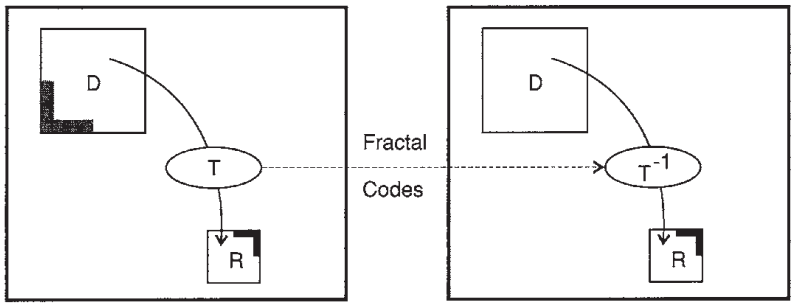


Figure 5.21 Fractal encoding and decoding.

Decoding is accomplished by iteratively applying inverse transformations. An interesting feature of fractal coding is that the domain blocks need not be transmitted because any arbitrary image can be used to start the reconstruction process. The pixels in this image can have any value—black, white, random, even pixels from another picture—for, iteratively, they will be changed to closely match those in the original image. The first application of the transformations generates an image containing rough approximations of the range blocks. Then, the transformations are applied to that image to better approximate the range blocks and so on. Usually, only a few iterations are required to create an image that closely resembles the original [Sayo95].

A limitation of fractal coding is that it is highly asymmetric because the process

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

of searching for the best-matching domain blocks is computation intensive. Smart encoding strategies can reduce the workload, as there are classes of blocks whose similarity can be recognized (do they contain an edge, for instance) before applying the transformations. Also, most often, the best-matching domain block is found nearby the range block being coded. Decoding is far less complex, even less complex that DCT-based techniques, and most current implementations for both image and video coding are in software. Block-based fractal coding can be extended to three dimensions for coding image sequences, but this technique has not yet proven practical [Jacq93]. Therefore, current implementations for video coding, including those described in Section 9.11, operate on one frame at a time.

5.7 Summary

This chapter continued our introduction to data compression algorithms, examining some important algorithms for compressing diffuse data—algorithms incorporated in standards for speech, audio, image, and video. We learned the following:

- The algorithms described in this chapter were selected as standards because they meet the needs of the marketplace: They provide good quality compressed speech, audio, image, and video at reasonable data rates. They can be economically implemented in VLSI hardware (or, sometimes, software). Also, they can deliver speech, audio, and video data in real time.
- Diffuse data compression, particularly video compression, is still new, and although there already are many compression techniques to pick from, still more and likely better ones are possible. This, coupled with the continuing discovery of new applications for diffuse data compression, means the process of creating standards is hardly finished. In fact, the next generation of standards is already being defined.
- Powerful and complex models have been created for speech, audio, image, and video data. These models, based on the principles of human hearing and vision, use psychoacoustic and psychovisual perceptual coding techniques to exploit the limitations of human ears and eyes.
- Diffuse data is compressed by applying lossy compression, which throws away nonessential information, in combination with lossless compression for efficient coding.
- Speech coders represent speech using as few bits as possible while preserving its perceptual quality. Historically, they were developed for telephone transmissions, and, more recently, speech coders have been developed for new communications and storage applications that demand high-quality speech reproduction.
- Speech coders either use perceptual, waveform coding techniques related to audio coders or incorporate a model of the human vocal tract that achieves greater compression but only on speech.
- Many speech coding standards exist; some were abandoned when techniques offering improved quality at lower data rates emerged. Thanks to research, more complex,

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

more powerful algorithms have been developed, and advances in VLSI technology (and software) support them.

- Audio coders exploit two psychoacoustic principles, thresholding and masking, to achieve compression. Some sounds are too soft for humans to hear; those that fall below the threshold of audibility can simply be ignored. Other sounds either are audible or are masked by louder sounds that temporarily raise the threshold of hearing; by adaptively assigning bits to audible and partially masked sounds, they may be recorded with only the degree of digital resolution necessary for high-fidelity reproduction.
- Multichannel audio coders achieve additional compression by exploiting the correlation between individual channels. Thus, state-of-the-art audio coders can deliver CD-quality audio using 64 Kbps per channel, reflecting a compression ratio of 12:1.
- Standards for audio compression exist as MPEG-1 and MPEG-2 audio, which are applicable to a variety of applications, and Dolby AC-3, which has been selected for transmitting HDTV audio and for a variety of other applications.
- Future applications will require that high-quality audio be coded at much lower bit rates, allowing it to be delivered along with low-bit-rate video.
- Image systems represent two-dimensional pictures as arrays of pixels, samples collected at regularly spaced points in the image. Compression is obtained by exploiting the correlation that exists between pixels and by exploiting psychovisual properties.
- There are many different types of images including bi-level, gray-scale, and color images, each demanding different compression techniques.
- Some image applications use only lossless compression, affording compression ratios as low as 2:1 or 3:1; others use combinations of lossless and lossy compression, affording compression ratios that can approach 100:1, or more.
- Lossless image compression exploits both spatial redundancy, the kind that exists between pixels in a region of the picture, and symbol redundancy, the kind found in the output bit stream produced by the coder. Lossy compression exploits irrelevancy to remove information from the picture that is unimportant to the human eye.
- Standards for image coding exist as ITU-T T.4 and T.6 for bi-level FAX images, JBIG also for but not limited to bi-level images, and JPEG for photographic-quality (continuous-tone) gray-scale and color still images.
- The coding techniques used by JPEG—DCT (discrete cosine transform), quantization, and entropy coding—also are basic to video coding.
- Among all types of diffuse data, image compression has the fewest, most-encompassing, extendable standards. Ignoring breakthroughs in image compression technology, which are always possible, future requirements are likely to be addressed by extensions to existing standards.
- Video systems represent video as a time-ordered sequence of pictures (frames).
- Video systems require compression that operates in real time at very high data rates, meets stringent transmission channel bandwidth limitations, provides compatibility with existing television broadcast and telecommunications transmission standards, and satisfies a marketplace with an ever-increasing demand for higher picture resolution.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

- Video coding algorithms use a variety of techniques to reduce the number of bits to be coded plus interframe coding to remove temporal redundancy between frames, intraframe coding to remove spatial redundancy within frames, and (statistical) entropy coding to remove symbol redundancy in the output bit stream.
- Video coding algorithms that use interframe coding obtain compression ratios which typically are three times greater than when the video sequence is compressed with JPEG-like intraframe coding—at the same quality level.
- ITU-T H.261 and ISO MPEG are the first generation of video compression standards containing algorithms that are practical and effective for videoconferencing and television applications, respectively.
- The most pressing need continues to be finding new video coding algorithms that can operate at lower bit rates. Standards that will operate at 64 Kbps or lower are emerging or are in development.
- Vector quantization, wavelet coding, and fractal coding are alternative image and video coding techniques used by de facto standard compression algorithms.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

have been used. Given the affordability of on-chip, dedicated VLSI circuits for data compression, it is conceivable that more complex Huffman coding and Lempel-Ziv algorithms will soon follow.

6.13 Multimedia Computers

In this chapter, we have encountered many instances where traditional computer systems use lossless compression to efficiently store symbolic data and programs. A new type of computer, the multimedia computer, has emerged in the 1990s. In this section, we will begin exploring how these computers use combinations of lossless and lossy compression to deal with diffuse speech, audio, image, and video data. We will look at document imaging systems, perhaps the first true multimedia computers, and multimedia servers, appropriate for image and real-time speech, audio, and video serving applications. Later in the book, we will return to this subject and, in Section 9.11, examine how multimedia PC client computers deal with all types of diffuse data.

Document imaging systems

Beginning in the 1980s, or earlier, long before the term *multimedia computer* was coined, computer-based document imaging systems were helping large businesses and government agencies to effectively automate business processes that handle large volumes of paper or film documents. The information on these documents is captured by scanning them, creating image files that are stored on computer disks. OCR (optical character recognition), bar-code reading, or manual keying allows indexing documents for later retrieval. For some applications, OCR is used to convert the characters on those documents into text. These steps allow the tools that computers provide for managing data and text to also store, organize, retrieve, and process documents. Incidentally, in business computers, until recently *multimedia* simply meant a combination of text and document images.

The original document imaging systems were mostly one-of-a-kind systems created for specific applications. More likely than not, they were stand-alone solutions unintegrated with other data processing activities. By the late-1980s, computer manufacturers began the process of integrating document imaging. This resulted in several thousand, multimillion-dollar document imaging systems being placed in production around the world. Mainframes and minicomputers or, at the least, high-powered UNIX workstations were combined with specialized components such as large-format, high-resolution displays, heavy-duty scanners and printers, and optical disc jukeboxes to build these niche products. Being high end in every way, and obviously expensive, only applications with huge volumes of images could afford them. Then came the PC revolution of the 1990s, totally revitalizing document imaging systems. Now, often document imaging is

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

nothing more than just another application where image and text data coexist in conventional databases on traditional computers. Not only can images be processed and displayed on more-or-less standard PCs, but image scanners and printers are now commonplace and low-cost. Today's document imaging systems are faster, cheaper, better integrated, and more standardized. They now cover the spectrum from the high-end, high-volume systems of old to low-end desktop systems intended for occasional use.

In the strict sense, document imaging involves the scanning of paper or film documents and converting them to digital format. However, in practice, the same system may include electronically received FAX images. It may also include images produced by computer programs such as drawings from computer-aided design programs. Photographs, both scanned and electronically generated, may be included too. Some examples of document imaging applications include check imaging for banks, claims processing for insurance companies, legal documents for law offices, and government records such as arrest records and driver licenses. Also, manuals and drawing for design and service organizations, electronic publishing repositories such as newspaper photo and text morgues (see Chapter 10), and patient records for medical institutions, including medical images (see Chapter 12).

To understand the role that data compression plays in document imaging systems, consider the simplified document imaging system shown in Figure 6.14. Documents are loaded into the system by scanning them into electronic format at an image scanning station. Data compression is applied (for reasons soon to be apparent), and OCR, bar-code reading, or manual keying may take place to capture text information and index the document. The image loading process also includes manual operations to assure the quality of the scanned image and the accuracy of the OCR information (although, software spell checkers and grammatical aids have greatly improved OCR operation). Index information must be created so that, later, the document can be retrieved. When image data leaves the scanning station, it is sent to the image server where it is stored. Later, as needed, image data is retrieved from the server and sent to an image display station for viewing or to a printer for hardcopy. Other types of images that may

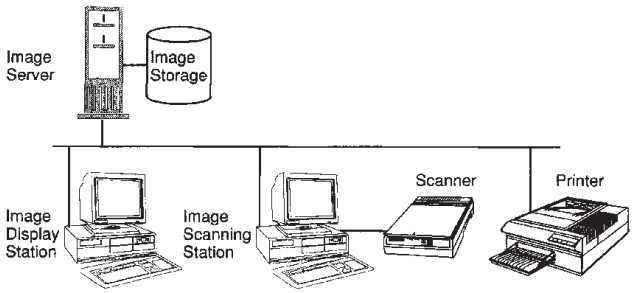


Figure 6.14 Document imaging system.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

be included in the system come from sources (not shown) such as FAX servers, computer-aided design workstations, or medical imaging modalities. All these system elements are connected by local-area or wide-area communications networks.

Although conceptually simple, real-world document imaging systems require careful design to enhance the productivity of the people using them while dealing with the realities of processing large blocks of data. For example, it is often acceptable to allow several seconds for retrieving an electronic folder of document images before presenting them to a customer service agent. However, when the folder is presented at the workstation, the agent will expect subsecond display of pages in order to perform the electronic equivalent of riffling through a manila folder in search of a particular document. Almost without exception, to make the system usable, sophisticated system management techniques must be applied to assure that user performance requirements are met. Here, the important role data compression plays in these systems becomes apparent. Even when compressed, an image may be 30–90 KB, many times larger than the database records needed for typical business transactions. Many applications contain millions of documents, requiring enormous amounts of storage, and a multilevel storage hierarchy consisting of at least main storage, online magnetic disk, and nearline optical disc is needed. Image files must be arranged within the storage hierarchy so that the most active are stored in the fastest storage. To assure users at display stations can retrieve and view images with acceptable response times, data staging and prefetching techniques must be used. Transmission times and network bandwidth are critical factors too. To reduce the traffic on the network, data compression is applied at the scanning station so that compressed images can be sent to the image server for storage. To further reduce network traffic and improve response time, compressed images are sent to the image display stations and to image printers where they are decompressed. As to implementing image compression and decompression, it can be accomplished entirely with PC software, whether for scanning, display, or printing. More commonly, image scanners contain compression function and image printers contain decompression function, either built into those products by their manufacturers or added as after-market plug-in cards [Durr94, Guen96].

A broad range of data compression and format options are used in document management systems. The storage required for a JPEG-compressed image of a driver on a driver's license, a compressed FAX image, and an uncompressed image of a work of art ranges from a few thousand to several million bytes. Not only are these different images with distinct requirements for compression, but they are typically found in systems with dissimilar designs. To understand what kinds of compression are used, designers must appreciate factors such as these:

- Diverse standards ranging from those that apply in high-end production systems to those in low-end, casual-use PC-based systems
- Many image types ranging from FAXes to photographs

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

- Many image representations including bi-level, gray-scale, and color photographic quality; For color images, RGB and many other color-space models; also, various pixel depths with both 8-bit and 24-bit color being common
- Various image resolutions ranging from 100 or 200 pixels/inch for FAX to 300, 600, 1200, or more pixels/inch for more complex images

This all comes together when the image is scanned and an electronic representation of the image is created. Each scanned image is stored in a bitmap data file. Every pixel in the image is represented by a bit or groups of bits in this file. Not only does the number of bits per pixel depend on the above factors, so too do the file structure and the compression technique. The simplest way to illustrate the diversity of compression techniques found in document imaging systems is to examine one widely used file format, TIFF (Tagged Image File Format).[10] TIFF supports various image resolutions and color-space representations. It supports uncompressed images and *six* compression methods, five lossless and one lossy [Andl96]. The lossless algorithms include packbit compression (a byte-oriented run-length coding scheme); also, MH, MR, and MMR from the ITU-T T.4 and T.6 recommendations for Group 3 and Group 4 bi-level FAX image compression (see Section 5.4) and a version of LWZ for color images (see Section 4.6). The lossy algorithm is JPEG (see Section 5.4). So, simply knowing that an image is stored as a TIFF file tells little; extensive decoding of the file header is required to learn what the image is and how it is compressed.

The diversity of file formats and compression schemes makes data interchange between today's document imaging systems difficult, requiring significant investments in software and skill. For organizations seeking to unify image handling, one solution is to convert all images to a common file format. However, today this may not assure seamless interoperation. Continuing with TIFF as an example, many imaging products support this file format; but the TIFF file specification is complex, with many variations and options, leaving room for errors and omissions in its interpretation. The result is that subtle differences have crept into vendor implementations of TIFF [Must94]. Perhaps with time, the strengthening or simplification of image file standards, and the migration to fewer, carefully crafted image compression standards such as JPEG, the interoperation of document imaging systems will be solved. Today, that goal has not been reached. The emergence of electronic document management architectures, the emergence of image data types as actual integral parts of relational databases, and standards for image objects—these are elements of general schemes for open interoperability now being worked on, but much remains to be done [Lidd96].

[10] A sampling of other commonly used image file formats includes BMP (a Microsoft bitmapped graphics file format), GIF (a bitmapped color graphics file format defined by CompuServe, as described in Sections 3.5 and 10.3), JPG (a color graphics file format using JPEG image compression, as described in Section 5.4), and PCX (a graphics file format originally developed by Softkey International Inc. for their PC Paintbrush™ program).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Multimedia server computers

Multimedia servers must store and manipulate huge quantities of speech, audio, image, and video diffuse data, making lossy compression an indispensable element needed to control the cost for storage capacity. They are distinguished from traditional computers in that when dealing with speech, audio, and video they must operate in real time. Multimedia servers target market segments beyond the reach of traditional computers. To be sure, multimedia servers are found on client-server computer networks where they are used for more-or-less traditional *business* applications. But they also are, or soon will be, used for not-so-traditional *entertainment* applications such as storing the digital data that communications networks and broadcast channels deliver to television set-top boxes and a variety of consumer-oriented devices. Indeed, entertainment applications are largely responsible for the development of multimedia servers. Table 6.3 lists some existing and potential applications for multimedia servers.

Table 6.3 Multimedia server applications.

Business Applications	Entertainment Applications
Employee training	Video on demand
Desktop multimedia databases	Interactive television
Videoconferencing switching	Home shopping
Security monitoring databases	Multiplayer video games
Interactive kiosk servers	Media production servers
Internet World Wide Web multimedia servers	Broadcast industry audio and video servers

Because multimedia servers primarily handle diffuse data, their operation differs from traditional computers to an extent that requires significant changes in design. The key difference is that they must store and retrieve speech, audio, and video data streams in real time, continuously, and without interruption. This entails transferring large quantities of data at high rates. Multimedia servers must connect to high-speed networks. They must manage storage hierarchies whose capacity, even for modest-sized multimedia servers, can easily rival or exceed that of all but the largest conventional computers. Building a true multimedia server requires new hardware and software. Some of the hardware changes include I/O subsystems than can handle high-speed networks, magnetic disk storage devices that can provide, not just high data transfer rates, but can sustain those rates over long periods of time, and high-bandwidth internal buses to tie it all together. Software changes include file systems that provide real-time scheduling

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

and performance optimization for large blocks of data, often for large numbers of users. Other software changes include bandwidth reservation techniques to assure consistent response times, techniques to synchronize delivery of speech, audio, image, and video data, and storage hierarchy management and buffering techniques that anticipate the demand for continuous delivery of speech, audio, and video with acceptable latency.

The aspects of multimedia server design reach far beyond the scope of this book; more thorough treatments can be found in several sources including [Crut94, Gemm95]. However, we must note that the computer industry is still struggling with the definition and design of multimedia servers, and the capabilities of current systems vary greatly. Multimedia servers come in sizes ranging from PCs to supercomputers, and not all products calling themselves multimedia servers have been totally redesigned to include all of the above changes. The primary criterion that determines how radically their design departs from conventional computers is how many simultaneous, independent multimedia data streams they can support and how efficiently they do this. With few changes, a low-end PC server may support single-stream playback, whereas highly modified designs are needed for high-end systems that can playback thousands of independent data streams.

Our interest in multimedia server design is limited to how data compression enters into the design. We will look at two classes of multimedia servers: store-and-forward servers and full-function servers. Store-and-forward servers essentially act as intelligent storage controllers and database servers for speech, audio, image, and video. They are used in applications where data intended for playback by clients arrives at the server already in compressed format. This data is loaded onto the server and delivered to clients who decompress it during playback. As an illustration of store-and-forward serving, which we will refer to throughout this section, a typical video server application is depicted in Figure 6.15.

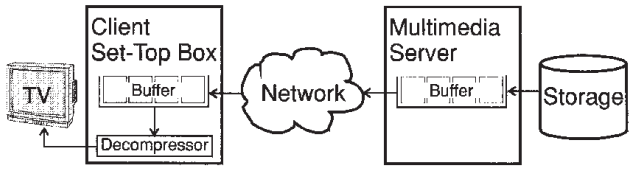


Figure 6.15 Store-and-forward video server application.

In store-and-forward applications, the server is challenged to supply each client's data stream buffers with enough data to ensure the playback process is not starved for data. To do this efficiently, just the right amount of data must be pulled from the server's storage hierarchy and shipped out on the network. The server mainly manages the database and traffic on the networks connecting the clients. It acts as accountant and traffic cop, providing cost-charging services and security, while routing data from its storage to the clients. Given this situation, we might conclude that the server is unaware of data compression, but that is

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

not true. Data compression affects store-and-forward multimedia servers in the following ways:

Assume the client is playing back data at a constant rate, say video at 30 fps (frames per second). When data is compressed with a variable-rate compression algorithm (MPEG for example), the number of data blocks that must be retrieved will vary according to how many video frames the compressor has packed into each block. Obviously, this causes a scheduling problem for the server in that it cannot decide how often to send a data block to the client.[11] There are several solutions. For instance, the server might pessimistically assume each block contains the fewest frames, but this may flood the client buffers with too much data. Perhaps the most effective option is for the server to interact with the compression algorithm and learn how many video frames each block contains. It can then tag each block with this information, using it to pace delivery during playback. The same information is also useful for loading data into the server's storage hierarchy, arranging data blocks on the storage devices in ways that facilitate retrieving a constant number of video frames from a varying number of data blocks.

Many store-and-forward multimedia servers also must support client-initiated interactive VCR-like control functions on video data streams. These functions include pause/resume and variable-speed forward/backward playback. This poses significant challenges to the server because the data rate may vary from nothing to several times the normal rate. A simple but often impractical solution is for the server to deliver data at whatever rate the client requests and let client logic select what information is presented to the viewer. Smarter and more practical solutions move the selection logic to the server. By examining the client requests, it can select the video frames that should be presented to the viewer. There are several approaches, none of which are yet fully developed, but all require that the server "get inside" the compressed video data stream, perhaps to select video frames, or even reconstruct the video data stream to support high-speed playback operations. Here, again, the algorithm used to compress data determines how this is accomplished and how effective it will be. To illustrate, when video is compressed with M-JPEG, which uses only interframe compression, the operations that the server must perform are relatively easy. On the other hand, MPEG and other algorithms that use intraframe compression make it much more difficult to assemble the appropriate data stream.

Turning now to full-function servers, they are distinguished by being actively involved in processing multimedia data, doing more than just storing, indexing,

[11] We assume a stream-oriented client-server relationship exists in which after the client initiates stream playback, the server periodically sends data to the client at the selected rate without further requests from the client. This relationship exists between multimedia servers and set-top boxes. There also exists a file-oriented relationship between client and server multimedia computers in which the client computer controls pacing by issuing a read request to the server for each block of data [Gemm95].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

and routing it. Some examples of the operations these servers may do include the following:

- Switching and routing audio and video to clients participating in multiparty videoconferencing. This may require compression and decompression transcoding functions to manage and deliver audio and video in different formats and resolutions for clients having differing capabilities.
- Delivering entertainment media to clients having equally diverse needs for transcoding.
- Performing the initial compression of diffuse data, as for music and movies, to allow direct capture by the servers used for media production and distribution. However, today this data is usually compressed by workstations that control the data capture devices, or by the workstations which are used to edit the media. This approach will likely continue to be popular because it is economical and it provides media developers with maximum control over the process.
- Doing decompression to allow data processing operations on objects in the database managed by the server. These operations might include semantic interpretation of speech, audio, image, or video message content. This allows for query retrieval and object recognition.
- Performing OCR (optical character recognition) of decompressed image documents, allowing the text portions of those documents to be stored and processed along with other symbolic data.
- Performing recognition on decompressed speech, allowing spoken words to be processed as text.
- Providing compression and decompression for graphics and image data along with the rescaling, resizing, and format conversions needed to drive shared print servers that generate compound documents, time-series arrays, geographic maps, binary halftone images, and so on.

The market for full-function multimedia servers is not yet well established, and neither are their requirements nor specifications. Therefore, how data compression and decompression will be used or implemented remains to be discovered. One might conclude that because diffuse data compression and decompression will be a widely used functions, they must be done in hardware. However, high-performance parallel server computers, such as the IBM SP1 and SP2, running software compression are used by the entertainment industry to prepare digitized motion pictures and video for distribution [PRNe93]. Thanks to the raw computing power of parallel systems, software-implemented data compression algorithms have proven acceptable for this application. One may speculate that high-performance diffuse data compression will be required in full-function servers of various sizes and costs, and hardware assists will be included in these products, but that scenario has not yet been established.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

6.14 Summary

In this chapter, we examined applications in digital computers, where, for the most part, lossless data compression is used for efficient storage of data and programs. We learned the following:

- Data compression reduces the cost, size, weight, and power consumption of mobile computers, and it squeezes the most from the limited available communications bandwidth.
- Software is widely used in desktop computers where abundant free CPU cycles are available for tape, disk, and communications compression. Add-on hardware for compressing diffuse data in multimedia computers is giving way to more integrated solutions.
- Hardware and software are prevalent in large computers for compressing symbolic data but not diffuse data.
- Data compression can be the right solution for interconnecting distributed computers when only low-bandwidth or low-cost links are available.
- Data compression allows more information to be contained in each level of the computer storage hierarchy, allowing it to be used more effectively.
- The running time of computer tape operations may be cut in half with data compression. Hardware and software, proprietary and standards-oriented solutions are offered for all computer tapes, large and small.
- The apparent storage capacity of computer disks can be doubled with data compression. High-performance, on-the-fly, unobtrusive solutions are available for file management software and disk subsystems.
- Data compression techniques similar to those used for tape and disk can increase the storage capacity of optical disks and solid-state files.
- The efficiency of main storage can be improved by compressing system programs, databases, user programs, and other data. Special data compression techniques are available for each.
- Compressed system software can be distributed, stored on disk, and stored in main memory; for most program modules; decompression can be done dynamically just before execution.
- Database compression accounts for the structure of the data within each page and how that data is used. Savings in disk and main storage result, and database access performance improves when data and indexes are compressed with semiadaptive algorithms.
- Compressed user programs can be executed by building an instruction-cache manager that decompresses cache blocks on demand.
- Compressed user data execution requires a data-cache manager that both decompresses and compresses cache blocks while managing the contents of main storage.
- On-chip lossless data compression can help eliminate interchip data transfer bottlenecks.
- Document imaging systems use lossless and lossy compression to efficiently store and transmit images ranging from FAX documents to color photographs. Unfortu-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

nately, a plethora of file formats and image compression techniques makes interoperability of existing systems problematical.

- Multimedia server computers use lossy data compression to control the cost for storage capacity. Variable-rate compression complicates scheduling and data selection during media playback. The requirements for lossy compression and decompression functions within multimedia servers are not yet fully established.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

7

Communications—Network Applications

Chapters 7 and 8 describe how data compression is used in communications. For many communications applications, bandwidth conservation is an important reason to use data compression. It can increase the effective throughput of existing transmission channels or, alternatively, reduce transmission costs by allowing lower-speed or shared transmission facilities to be used. Equally important in some applications, data compression can reduce transmission time. Various combinations of lossless and lossy compression are used in communications, with most algorithms being specially adapted for the transmission channels and the data being transmitted on those channels. For each application, we will examine the communications system and the compression algorithms that are appropriate. This chapter deals with applications where voice, data, images, and video are transmitted on communications networks. Chapter 8 describes broadcasting applications.

7.1 Industry Overview

Many businesses are involved with providing products and services for communicating information—between people, between businesses, and from industry to consumers. Everyone from booksellers to television broadcasters provides some form of communications. Our view of "the communications industry" will be limited to applications for digital electronic communications systems and the businesses that provide them. These include traditional telecommunications—telephone, telegraph, and other point-to-point communications services. Also, data communications services, the modern-day blend of computers and communications. And broadcasting services such as radio, television, cable, and satellite.

The communications industry provides rich and varied opportunities to apply

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

data compression. Most assuredly, communications applications provide a study in contrasts. Well-established applications, such as FAX, have clearly defined needs for data compression and mature technology is in place. Newer applications, such as HDTV, also have demonstrated needs, but their data compression technology is still evolving to meet changing definitions of the applications themselves. Sometimes, data compression simply solves a channel bandwidth or cost-of-transmission problem. But some applications, such as direct broadcast satellite television, use it to improve and expand existing services. Still others, like cable TV, plan to use data compression as the springboard for expanding the scope of the application, to do something new, to leverage other industries.

Standards are extremely important in the communications industry because of its global nature and so many diverse elements must work together. This is true of data compression standards too. However, only a few official standards for compressing symbolic data originated within the communications industry, and in some applications, there are no standards—when there should be. In contrast, all of the official standards for compressing diffuse data have come from, are closely connected to, or are strongly influenced by the communications industry. Unlike the computer industry, in the communications industry once a standard is set in place, data compression standards included, usage is widespread because standards compliance usually is compulsory, not voluntary.

In this and the following chapter, we will adopt the conventions of the communication industry and describe services for voice, data (computer data, the symbols found on a keyboard), image, and video transmissions. Moving symbolic digital data between computers is a well-established but relatively new business for the communications industry. However, it is only one component of the communications industry. Granted, the largest part of the communications industry is still involved in the traditional business of moving analog voice, image, and video information. But now all segments of the industry are on a fast path to completely replace the infrastructures of their traditional businesses with digital networks and transmission facilities. Digital data is expected to revitalize the industry and increase its profitability by providing new services that only digital technology will allow. The opportunities for digital data in the communications industry—and for data compression—are enormous, but one should be prepared for a slow roll-out. As we have noted, this industry is standards oriented and it is highly regulated. Expansions or new directions of any kind generate complex standards and regulatory issues, the sort of issues that easily take years to resolve. Communications is also an industry with huge investments in its infrastructure. For that reason (and many others), in the past, changes have come slowly.

Electronic communications systems

The communications industry is built on electronic communications systems that transmit signals containing encoded information over a distance using electrical energy to carry data between a transmitter and a receiver. As shown in Figure

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

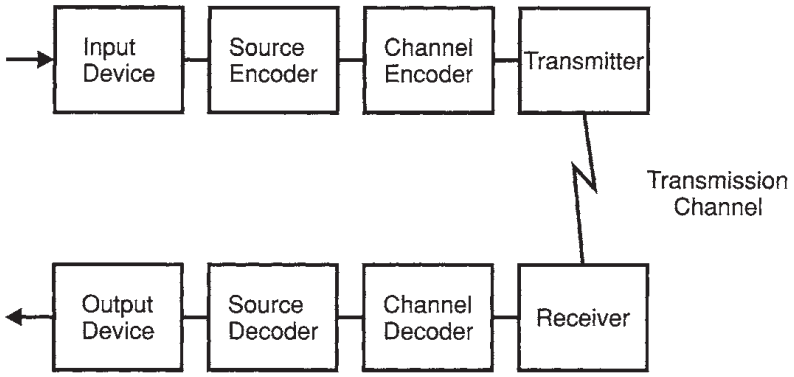


Figure 7.1 An electronic communications system.

7.1, information is conveyed by energy traveling along a transmission channel. That channel may be a wire, a wireless (radio wave) link, or a fiber-optic link.

The first task for any electronic communications system is to convert the information to be transmitted into electrical signals. This is accomplished by input devices such as microphones, computers, scanners, and cameras, where information in the form of voice (speech) or audio, computer data, images, and video are captured. Electrical signals from the input devices are processed in several steps: As previously stated, our interest is limited to *digital* electronic communications systems (for reasons that will soon become apparent). Digital systems transmit information as a sequence of encoded pulses (whereas analog systems transmit continuously varying signals). Therefore, the source encoder must perform analog-to-digital conversion and other transformations—including data compression—to efficiently encode the data. Then the data must be further processed by the channel encoder into a form suitable for transmission. This processing includes adding error-detection or error-correction data to compensate for transmission errors and conversion to an electrical signal that the transmitter can accept. In a wired system, the transmitted signal takes the form of an electrical current that travels along a conductor. In a wireless system, the signal is sent through space as an electromagnetic wave launched by the transmitter. In an optical system, the electrical signal is converted to light that is guided by the glass fiber.

At the receiver, the signal is captured and all the processing steps are undone, allowing the transmitted information to be recovered and reproduced. Output devices include computers, loudspeakers, printers, and image or video display devices. If all goes well, the recovered information will be an exact or acceptably faithful reproduction of the original.

Communications styles and compression

Electronic communications systems are designed to support the communications needs of people who use information in different ways to learn about the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

world around them. The written word, speech, music, pictures, and even touch and smell are all part of communications. As shown in Table 7.1, there are three styles of communications, each providing differing types and degrees of interactions between people.

Table 7.1 Communications styles.

Communications Style	Interaction Between People	Examples
One way	No direct interaction	Books, magazines, newspapers, radio, CDs, TV, VCR tapes
Bi-directional	Usually not in real time	Letters, FAXes, E-mail, voice messaging, images
Collaborative	Real time	Meetings, telephone calls, audio/video conferences

These different styles of communications call for different types of electronic communications systems and, not surprisingly, different types of compression and decompression algorithms. One-way communications often involve a one-to-many distribution of information—broadcasting and mass distribution—where (in computer terminology) there is one server and many clients. In this situation, getting the highest compression ratio is a good trade-off, for additional compression complexity can reduce transmission time, bandwidth, and implementation costs. Asymmetric compression algorithms (meaning compression can be complex and slow, but decompression is simple and fast) are encouraged. Simple, fast, and inexpensive decompression is highly desirable for all mass-distribution applications because it reduces the complexity and cost of the receivers. Many different situations can occur, but bandwidth and (to some extent) storage capacity—particularly for video information—are usually the major limiting factors. Therefore, compression algorithms may be architected to trade off quality (image resolution and video frame rate, for instance) in exchange for operating within available bandwidth or storage capacity.

Bi-directional communications are two-way and, therefore, each party must compress and decompress information. Here, symmetric compression algorithms are desirable. However, real-time interaction usually is not involved, and processing speed can be traded for higher compression ratios, higher quality, or lower implementation costs. For example, an inexpensive FAX machine that operates at less than the maximum data transmission rate may be perfectly adequate for FAXing orders to the local pizza parlor (but less than adequate if long-distance telephone line charges apply).

Collaborative communications are also two-way, and they demand real-time interaction. This means that compression and decompression algorithms must be symmetric, and they also must provide a low (and controlled) delay time for

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

both encoding and decoding because human interaction is involved. Long delays in processing data for client-server computing increases the response time for transactions, thus negating the advantage of compression. Long delays in processing either audio or video impedes interactions between individuals, and so do erratic processing times that disrupt the smooth flow of information. As with one-way communications, many different situations can occur, but audio and video compression algorithms often must be architected to sacrifice quality (speech intelligibility, audio bandwidth, and image resolution, for example) to operate within available time and bandwidth constraints.

7.2 Voice Telecommunications

For over 100 years, the telephone system has been the most widely used telecommunications service. Voice communications remain a huge revenue producer for the telecommunications industry, an industry whose origins trace back to the transmission of voice and telegraph traffic sent over metal wires and, later, microwave links. Starting in the 1970s, with the integration of computers and communications, the telephone network began carrying computer data traffic. Today, the telephone network has evolved to a network of interconnected networks that carry a mixture of voice, data, images, and video. It uses both analog and digital signals sent over wires and a variety of wireless and fiber-optic links. The major components of telephone networks are shown in Figure 7.2. The subscriber's equipment includes telephones for voice and modems for nonvoice traffic. The local loop connects subscribers to the system. The local exchange switches calls, and the long-distance network routes calls between local exchanges. Traffic within the local exchange and long-distance network complex is handled by a mixture of analog and digital technology. Traffic on the local loop between most subscribers' telephones and the local exchange switching center is still analog. However, with the deployment of ISDN (Integrated Services Digital Network), the local loop traffic is becoming digital too. ISDN provides users with end-to-end digital transmission and increased bandwidth, enabling voice, data, image, and video transmission services.



Figure 7.2 Telephone network.

With wireless communications having been added to the mix, wireline telephone networks are no longer the sole providers of telecommunications services. Table 7.2 lists a few of the major services, both operating and proposed [Kobb95].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

8

Communications—Broadcasting Applications

In this chapter, we continue our examination of communications and consider television and radio broadcasting applications where data compression is used to reduce the bandwidth needed for delivering digitized data. Various combinations of lossless and lossy compression are used, with most algorithms being specially adapted for the transmission channels and the data being transmitted on those channels.

8.1 Broadcasting Overview

Profound changes are taking place in television and radio broadcasting. Today, broadcasters are largely focused on new digital products and services. As pointed out in Chapter 3, no fewer than four approaches now compete to deliver analog video (and audio) to our homes. All of them—terrestrial broadcasting, cable, satellite, and in-home boxes, including VCRs and video disc players—are moving to digital. Telephone and computer networks also have plans for home delivery of digital audio and video, and, in so doing, they hope to capture some portion of the consumer dollars now going to broadcasters. In the past broadcasters engaged strictly in mass distribution of home entertainment and information services using one-way communications. Today, reflecting the changing needs of consumers, they are moving toward providing a more personal, more customized delivery of enhanced home entertainment and information services and to using two-way, interactive communications. Make no mistake—much remains unknown about what consumers really want and how the broadcasting industry will reshape itself to meet their needs. Major questions remain unanswered, particu-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

larly in television broadcasting where the demand for high-definition television and the requirement for interactivity is unknown. But, clearly, consumers do want—and will pay for—more freedom of choice and more quality. Examples include multichannel cable and satellite television versus a few terrestrial broadcast television channels; also, FM radio (and, soon, digital audio broadcasting offering CD-quality audio) versus AM radio. Without question, to move forward, the broadcasting industry needs digital technology—first for delivering more choice and more quality and later, perhaps, for providing computer-controlled interactivity. Existing analog broadcasting facilities provide neither the bandwidth nor the interactivity and, essentially, are incapable of providing the desired functions. Digital broadcasting—with data compression—does provide the needed capacity and function.

In this book, our exploration of the heady events taking place in broadcasting centers around data compression and how it is used to expand the effective bandwidth (carrying capacity) for broadcast channels. The immediately following sections of this chapter examine several different broadcasting methods, each of which must deal with the limited carrying capacity of the media. As in all communications, bandwidth is a limiting constraint for broadcasters, whether the transmission channels are wireless, wired, or optical fibers. For wireless and wired broadcast methods, there simply is no more bandwidth available; for optical-fiber-based methods more bandwidth means investing more dollars. In both situations, data compression is the enabling technology.

8.2 Terrestrial Broadcast Television

In the United States, for nearly 50 years commercial television has provided home entertainment and information using over-the-air terrestrial broadcasts in analog NTSC standard format. Elsewhere in the world, the PAL and SECAM analog standards have been used for nearly as long. Over the years, analog broadcast standards have been upgraded to include color, stereo audio, and a few specialized data services. Although few technology-based standards have lasted so long or served so well, the convergence of computers and communications—in the home—foretells their giving way to new digital-based broadcasting standards. Digitized television, when integrated with computers, telecommunications networks, and consumer products, holds promise for providing a new array of products and services never before available.

In 1987, the FCC formed ACATS [the Advisory Committee on Advanced Television (ATV) Service], a committee to assist in establishing a new television standard for the United States. Their mission was to develop a new standard for high-definition television that could deliver wide-screen, movie-theater-quality pictures and CD-quality sound to home television receivers. Various systems were considered including an analog HDTV system on the air in Japan since

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

1991 and four proposed digital HDTV systems[1] [Yosh93A, Hopk94]. In 1992, the FCC committee decided the future U.S. standard would be digital, and it would be broadcast on 6-MHz terrestrial channels, using the same bandwidth and the same channels as for regular-definition (NTSC) broadcast television. The problem was how could HDTV fit into a 6-MHz channel, as it displays more than twice the resolution of the existing NTSC standard, both vertically and horizontally. Also, it has a wide-screen format (16:9 rather than 4:3), and its high-quality video signal uses a component format that requires more bandwidth than the composite-format NTSC signal. It also provides 5.1-channel CD-quality audio. In total, HDTV would need as much as 60 times the data rate provided by the 6-MHz channel [Zou92]. Obviously, as any reader of this book knows by now, the solution is data compression. The MPEG-2 compression algorithm, which evolved along with the definition of HDTV, is now an integral part of the HDTV standard. The HDTV standard is now in the final stages of formalization.

While the HDTV standardization effort was in progress, broadcasters began to recognize the power of digital television could be used in ways other than for supporting HDTV. Although HDTV is technically intriguing, the downside is, initially at least, that it will be very expensive for both broadcasters and consumers. New production and distribution facilities are required, and HDTV receivers will cost several times more than standard NTSC (PAL or SECAM) receivers. According to many industry observers, this translates into a long period of waiting for HDTV to become widespread and profitable.[2] Meanwhile, terrestrial broadcasters petitioned the FCC—and received permission—to use the channels allocated for HDTV to broadcast lower-resolution SDTV (standard-definition television). The advantages of SDTV are that it provides studio-quality (NTSC CCIR-601 standard) video, which never before could be broadcast to home receivers. It is digital, which means it can be compressed and transmitted over standard broadcast channels, and it can be received and displayed by NTSC-standard receivers when equipped with set-top converter boxes. Also, SDTV, which uses MPEG-2 compression and conforms to the HDTV standard, requires not more than 5–6 Mbps of bandwidth, a fraction of that needed for HDTV. As shown in Figure 8.1, this allows some combination of several SDTV channels and other digital data services to be packed into one 6-MHz terrestrial broadcast channel. It is argued this will allow terrestrial broadcasters to compete with other home entertainment and information providers to deliver the variety and type of services consumers desire [Leop95D].

[1] A significant detail is that the Japanese HDTV system was transmitted via satellite because it required several times more bandwidth that standard television.

[2] The plan for phasing in HDTV in the United States allows each of the more than 1400 terrestrial broadcasters who choose to air HDTV programming to apply for a second (UHF) channel [Hopk94]. Conventional analog NTSC programming will air on the channel they now have been granted, simulcast along with digital HDTV programming on the second channel. At some point in time, analog broadcasts will be phased out.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

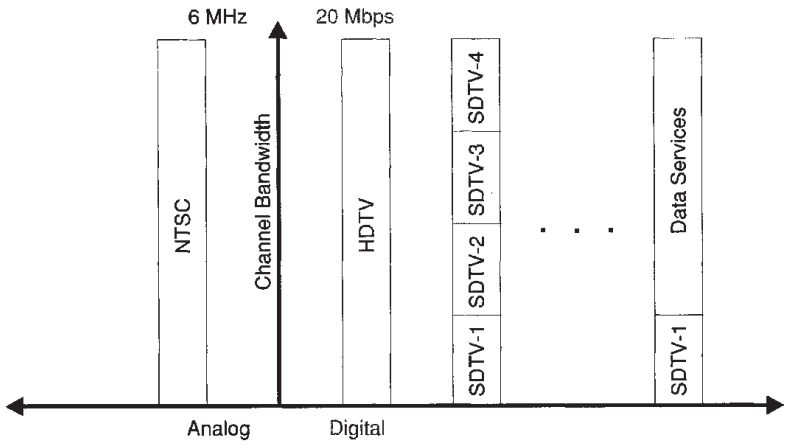


Figure 8.1 Terrestrial television channel bandwidth assignment options.

Plans for digital terrestrial broadcast television in the United States continue to move forward, with FCC approval on an ATV standard for HDTV and SDTV expected in 1997. In Europe and Asia, efforts leading toward digital television are moving forward too [Yosh95C, Yosh95D, Yosh95E]. In Europe, the Digital Video Broadcast (DVB) project is defining standards for terrestrial, cable, and satellite digital television broadcasts. The move from analog to digital television in Europe was delayed by a since-abandoned effort to create an analog HDTV standard [Fox95]. In Asia, the move to digital television lags behind both North America and Europe, in part because analog HDTV has been available in Japan for several years. Thanks to differing existing transmission standards and a host of other factors, a single worldwide standard for digital television is doubtful. However, all digital television proposals do share one common element: They all specify MPEG-2 compression. (See Section 5.5 and Table 5.8 for further discussion of MPEG-2 and HDTV formats.)

We will close our discussion of terrestrial television broadcasting, as we begun, with some thoughts on the durability of the new HDTV standard. The question is, Will it serve well for years and years, as the NTSC standard has done, or will advances in information handling technology cause the HDTV standard to self-destruct at a tender age? Throughout the process leading to its creation, lots of people, supporters and foes alike, have offered up strong opinions, and reasons to back up their position, on whether the HDTV standard will (or will not) endure. Reams have been written to explain, justify, or debate almost every conceivable subject relating to the HDTV standard, and several comprehensive overviews and introductions have been published, including [Hopk94, Basi95, Chal95]. Here, just one aspect of the HDTV standard will be addressed—the inclusion of MPEG-2 data compression and its effect on the durability of the standard.

Section 5.5 (Video Compression, State-of-the-art video coding and beyond) described some technical limitations of MPEG. Most notably, it does introduce

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

coding artifacts that can degrade picture quality, especially at low bit rates and in fast-action image sequences containing lots of motion with frequent scene-to-scene transitions. Neither is MPEG the most efficient video encoding algorithm when measured in terms of encoded bit rate or ease of encoding. Often these limitations are seized upon as reasons why another data compression technique—and there are many—would be a better choice and why MPEG will shorten the life span of the HDTV standard. This may seem faint praise for MPEG, but the truth is that *all* video data compression algorithms (at least those that use lossy compression) produce artifacts. Continued improvement in MPEG encoding techniques, which will assuredly reduce its production of noticeable artifacts during decoding, can be accomplished without changes to millions of HDTV receivers. (For proof, see the DBS discussion in Section 8.3.) As to whether MPEG coding efficiency will shorten the life span of the HDTV standard, one can only observe that the encoded bit rate is sufficiently low for broadcasting HDTV and, because it is a broadcast medium, the complexity (and cost) of encoding affects only the broadcaster, not the receiver.

There is one additional point: Should *any* data compression algorithm that compromises picture quality be part of the HDTV broadcasting (or any) transmission standard? Here, the assumption is that when (not if) some sufficiently better compression technique comes along, and everyone agrees the standard needs to be updated, millions of HDTV receivers will be obsoleted. Given that video compression is hardly a mature science, this could happen sooner than anyone expects. One proposal is to define HDTV without regard to bandwidth (and without compression) and transmit it over those broadband media able to provide sufficient bandwidth [Simo95]. Unfortunately, no existing transmission media reaching U.S. homes, not terrestrial, satellite, or cable broadcasting, nor telephone or computer networks, provides the bandwidth needed for uncompressed HDTV. So, if we want HDTV, it is going to be compressed, and that compression will be MPEG—for better or worse.

8.3 Direct Broadcast Satellite

The era of direct broadcast satellite (DBS) television service arrived in 1994 when Digital Satellite System (DSS™) broadcasts became an "overnight" sensation.[3] Simple in concept—digital signals are transmitted from satellite television broadcasters' ground stations to orbiting satellites, where the signals are then beamed to homes equipped with receiving dishes and decoders—DBS actually proved tremendously difficult to turn into a profitable commercial enterprise. This is

[3] Digital Satellite System is a joint effort of DirecTV-GM Hughes Electronics, RCA-Thompson Consumer Electronics, and United States Satellite Broadcasting-Hubbard Broadcasting. DSS began broadcasting in 1994 with 175 channels of video and music.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

what the backers of DSS learned after traveling a long and arduous path strewn with technical obstacles, financial roadblocks, and the failure of half a dozen other DBS ventures. Beginning in 1982, when the FCC authorized the orbital satellite slots needed for the service, many companies invested hundreds of millions of dollars to give consumers a low-cost, widely available, small-dish alternative to large-dish satellite, cable, and terrestrial broadcast television and videocassette rentals [Marr95]. To attract viewers, DSS backers knew they must offer consumers more than just low cost and wide availability. They concluded that DSS must provide near laser-videodisc-quality S-VHS video, CD-quality sound, and 100+ channels for video, music, and future broadband data services—all specifications in excess of what viewers could find on competing cable television systems. The backers of DSS, although they may not admit it, probably were surprised along with everyone else at its explosive growth. Within the first year, nearly one million DSS receivers were sold, exceeding by many times the first-year sales for VCRs, CD players, and big-screen televisions, all highly successful consumer products of their time. Among the many features of DSS, surely the enormous array of channels and the high-quality pictures and sound provided the push that consumers needed.

How could DSS afford to provide so many high-quality channels? Compressed digital video and audio provided the answer. Satellites are well known for being notoriously expensive to build and put in orbit, each costing perhaps 50-100 million dollars (or more) [Barr93], and satellite transponder bandwidth is limited. Data compression allows transponder bandwidth to be used far more efficiently. With compression, the 27 Mbps bandwidth of each DSS transponder can provide four to eight or more channels of high-quality, SDTV-resolution digital video. DSS also provides about 30 digital radio channels and, in the future, may offer digital data services. Data compression makes DSS economically feasible, and it provides the picture and sound quality consumers want.

DSS uses MPEG-2 compression, and herein lies an interesting tale. At the time DSS service was launched in 1994, the final MPEG-2 specification was still in process. However, the lack of a final specification had not stopped chip makers who, as soon as drafts of the MPEG-2 standard were available in 1993, rushed to put MPEG-2 decoder chips into production for use in DBS receivers and many other products. The problem DSS faced was that MPEG-2 encoder chips, which were needed for live broadcasts, would not be available for many months in the future.[4] The solution DSS chose was to use MPEG-2 decoder chips in its receivers but to begin operating with MPEG-1 encoded broadcasts. This allowed all DSS receivers, beginning with the very first unit shipped, to receive broadcasts encoded

[4] The complexity of MPEG-2 encoding delayed the arrival of encoder chips that could compress live broadcasts in real time. Non-real-time encoders, whether in hardware or in software running on powerful computers, which suffice for compressing programs stored on film or tape, could not handle sports events and other live DSS broadcasts.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

in MPEG-2 whenever they became available. Meanwhile, the receivers could decode MPEG-1 encoded broadcasts because MPEG-2 is backward compatible with MPEG-1. Surprisingly, the less-detailed MPEG-1 encoded pictures produced little reaction from consumers who were more concerned with the picture artifacts that appeared in fast-action image sequences containing lots of motion or in sequences containing frequent scene-to-scene transitions [Pohl95B].

Picture artifacts are the bane of all image and video compression algorithms, MPEG included. MPEG-2, however, by providing great latitude for how video is encoded, with the only rigid requirement being that the end result conform to the MPEG-2 bit-stream syntax, reduces the problem.[5] One of the flexibilities of MPEG-2 is that it allows variable bit-rate encoding. For instance, fewer bits per second can be used to encode low-resolution images, or image sequences that contain little motion, or image sequences captured at less than 30 fps (as with 24-fps movies). On the other hand, more bits per second can be used for encoding when these conditions do not prevail, the most important being to reduce visible artifacts for high-motion scenes and during scene-to-scene transitions.

DSS incorporates several innovations for reducing the visibility of picture artifacts. Initially, as shown in Figure 8.2(a), channels were assigned a bit rate

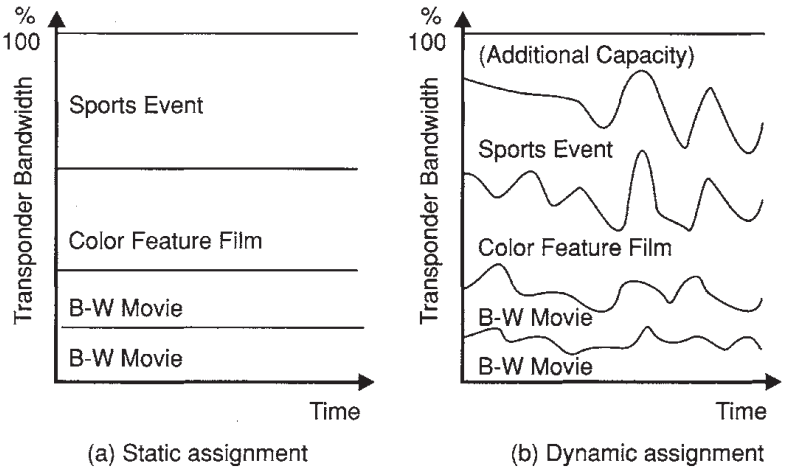


Figure 8.2 DSS transponder bandwidth sharing.

that did not vary with time but did vary according to program content and importance. For example, old black-and-white movies received fewer bits per second than first-run color, action-intensive, pay-per-view features, and all movies (running at 24 fps) received fewer bits per second than full-motion, action-intensive sports events. Over long periods of time the average results of this

[5] Although DSS uses MPEG-2 compression, its bit-stream packet protocol is different than that found in the MPEG-2 specification.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

approach may be satisfactory, but it cannot address the instantaneous demand for bits, which leads to consumer concerns about visible artifacts. DSS subsequently introduced the dynamic bandwidth sharing method shown in Figure 8.2(b), a statistical multiplexing process that continuously varies the bit rate according to program content. In case you were wondering—yes, its operation is similar to STDM, a method for multiplexing data packets on the shared data communications lines discussed in Section 7.3; only here, the packets contain MPEG encoded pictures. Like STDM, no channel can be absolutely guaranteed all the bandwidth it may need, but the overall effect of dynamic bit assignment has two positive outcomes: First, the total available satellite transponder bandwidth is used far more efficiently, which means each transponder can carry more channels. Second, when compared to static bit assignment, the number of times when not enough bits per second are available is greatly reduced for every channel. The consequence, of course, is that the encoding and multiplexing processes are far more sophisticated.

The unquestionable success of DSS has encouraged companies around the world to begin providing digital DBS television service. In 1996, similar services are in operation or are planned to go into operation for Canada, Latin America, South America, Europe, Japan, and other Asian countries. Also, four DBS television services are available in the United States (see Table 8.1), and others are being planned [Warr95, Arns96]. Two important groundrules for a successful DBS service are to provide a large number of channels and allow viewers to use a small receiving dish. Thus, all DBS providers are on or planning to move to high-power satellites (a requirement for small dish size), where space for DBS transmissions is now at a premium. The prospects for placing additional satellites in orbit are dimmed in that few orbital satellite slots remain available for DBS transmissions. This has increased the price for admission for other potential DBS providers, with MCI Communications Corporation paying nearly $700 million for a satellite slot and license to operate its 300-channel DBS system planned for the late 1990s [Shiv96].

8.4 Cable Television

Cable television is available to most U.S. households; by 1992 more than 60% of U.S. households were connected [Pres93]. Of the remainder, some choose not to be connected, and some 10–20 million of us live in mostly-rural areas not served by cable. Not only has the number of cable television subscribers grown, but so too has the number of cable television channels. With typical cable systems offering, perhaps, 40–60 channels, this still leaves content providers scrambling to find a place on crowded cable systems for at least 100 new special-interest cable channels. Technology exists for cable systems to provide no less than 150 channels, but cable system operators have moved slowly on expanding existing (analog) systems, as adding channels requires new electronics and laying new

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 8.1 United States DBS television services circa 1996.

	DSS	PrimeStar™	EchoStar™	AlphaStar™
Receiving dish size	18 in.	39 in.[1]	18 in.	30 in.
Channel capacity[2]				
1990	—	10[3]	—	—
1994	175	67	—	—
1996	200	95	95	100
1997	200+	140	200	300
Compression algorithm	MPEG-2[4]	MPEG-1[5]	MPEG-2	MPEG-2
References	[Pohl95D, Taki95A, Taki96]	[Edge95B, Taki95A, Nice96]	[ElNe95, Edge96, Nice96]	[Edge95D, Taki96, Nice96]

Notes:
[1] Dish size will decrease with move to high-power satellites.
[2] Total - video and music.
[3] Analog transmission. In 1994, PrimeStar upgraded to digital.
[4] Bit-stream packet protocol differs from the MPEG-2 specification.
[5] May move to MPEG-2 in 1997.

cable, or replacing cable with high-capacity optical-fiber links.[6] This, a hugely expensive undertaking, which could, at most, triple the channel capacity, would leave them with a cable system having more channels but not more function.

Instead, the cable industry sees a future where subscribers will be provided with individualized programming and real-time two-way interactive communications will be supported. To accomplish this, cable systems must remake themselves using digital television and computer technology. With digital television—and data compression—the channel capacity of existing systems can be multiplied tenfold.[7] Most important, the merger of digital television and digital computer technology allows cable systems to move from just providing one-way broadcast

[6] Cable systems operate within various bandwidths. Modern urban hybrid coaxial/fiber cable systems provide a forward (headend-to-user) bandwidth range of 54–750 MHz, allowing up to 110 (6 MHz analog) broadcast channels to be carried. All-fiber networks provide 1 GHz bandwidth, allowing 150 channels.

[7] Each 6-MHz cable channel can carry 40 Mbps of digitized video. With MPEG-2 compression, 10 high-quality SDTV video data streams can be carried by each channel.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

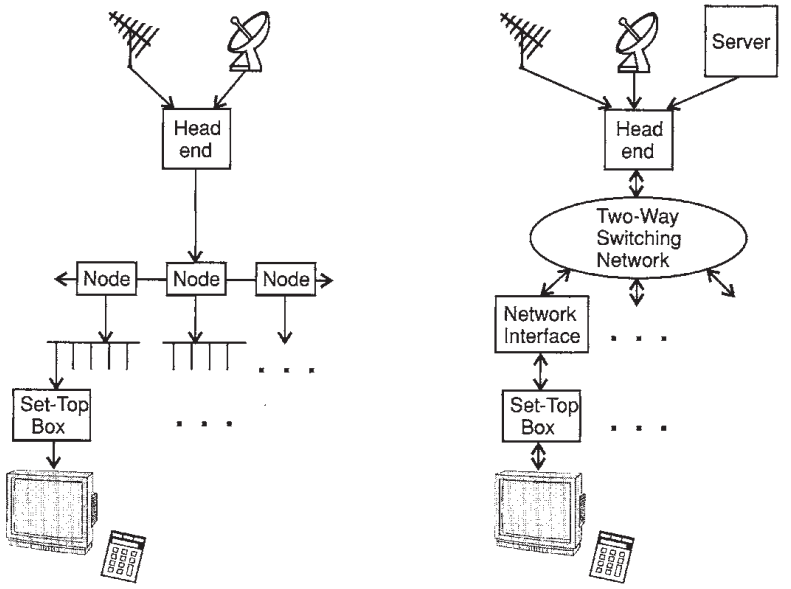
television to offering two-way interactive television and a host of data transmission services. A potpourri of services is envisioned for digital interactive cable systems including the following:

- On-demand movies and news
- VCR-like function for viewing movies
- User selection of camera angles for sporting events
- Interactive videogames with far-off competitors
- Home shopping and interactive advertising
- Electronic Yellow Pages browsing
- Internet access, E-mail messaging
- Subscriber-to-subscriber services including telephone, video telephone, videoconferencing, and more

In the early 1990s, the cable television industry, along with programming-content providers and a slew of computer and silicon vendors, engaged in what can only be described as frantic efforts to form strategic alliances and launch ambitious plans for full-service interactive networks. (Telephone companies, having the same goals in mind, engaged with these same groups in similar activities.) By the mid-1990s, many of the grand plans had fallen by the wayside and the early morning marathon dash to interactivity took on the characteristics of an evening stroll. Cable system operators point to the substantial costs for rebuilding their networks for two-way interactivity. Then, too, there are regulatory issues to resolve, and the uncertainty about what customers want on their digital interactive cable—and will pay for—remains, as it was in the beginning, a nagging concern.

To appreciate the challenges, consider the two cable systems shown in Figure 8.3. A conventional analog cable system, as shown in Figure 8.3(a), distributes broadcast channels through what is called a tree and branch network [Mill94]. Terrestrial and satellite broadcasts are distributed down through the network to all subscribers, whose set-top converter boxes determine which channels they may view. The first cable networks were built from coaxial cables, amplifiers, and distribution gear. In theory two-way communications are possible on all-coaxial cable systems, but, in practice, amplifier noise makes this impractical. The solution, which is well underway, is to eliminate the amplifiers by replacing coaxial cable with optical fiber. Hybrid fiber/cable networks not only increase the channel capacity, picture quality, and reliability of the cable system, they also make limited forms of two-way communications possible. Nevertheless, this still does not provide for the kind of fully two-way interactive communications that cable television visionaries dream of as a goal. For that, a network similar in concept to the one depicted in Figure 8.3(b) is needed. Here, by means of circuit switching, every subscriber is essentially provided a private path that links them to any one of thousands of video (or data) sources. In computer terms, one

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



(a) One-way broadcast (b) Two-way interactive

Figure 8.3 Cable television systems.

might think of this as a gigantic client-server network, where, instead of receiving a huge number of channels, each user has a single channel that carries personalized video or data.

To move from one-way broadcast to two-way, fully interactive cable systems, the subscriber's set-top box must grow more sophisticated, and therein lies one reason that interactive television for the masses may be a long time in coming—cost. Current set-top (analog converter) boxes, which essentially provide little more than channel selection, must be replaced with set-top boxes that contain a high-powered microprocessor and its operating system, peripheral chips for decompression, video and graphics processing, and copious amounts of memory. This is needed just to provide digital one-way television and limited two-way interactive services.[8] Advanced interactive services such as subscriber-to-subscriber videotelephony and videoconferencing require even more processing power and memory, real-time compression chips, camera interfaces, and more—endowing the set-top box with the characteristics (and cost) of a well-configured multimedia computer. Even with silicon integration and high-volume production, for sometime to come the cost of set-top boxes will delay the onset of interactive television services [Cole95]. Additional information about set-top boxes can be found in many sources, including [Cici95].

[8] DAVIC (the Digital Audio-Video Council) has set a cost target of $300 for the first standardized digital set-top box [Leop95C].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Details of how cable television systems will be reconstructed to provide interactive television reach far beyond the scope of this book. For those readers who wish to learn more, your attention is directed to several survey papers [Mill94, Blah95, Larg95]. Our interest is limited to the role data compression plays for cable to go digital and provide interactivity. The simplest scenario is one where data compression (and digital television) is used to increase the channel capacity of a conventional cable system. Compressors placed at the headend, or in the video sources themselves, will allow up to tenfold more channels to be squeezed into the space occupied by a single analog channel. The compressed digital video is then transmitted to set-top boxes where, under microprocessor control, a decompressor will reconstruct whatever channel the viewer selects. This might seem to be every channel surfer's dream, having 500 or more channels on tap, but not so.[9] Channel surfing is all but impossible with compressed digital video because (recall the MPEG group of pictures discussion in Section 5.5) when a new channel is selected, it can take several seconds to build up a viewable picture. Rather than offering viewers a bewildering array of channels to pick from, some channels will likely be reserved for program guides, similar to those found on DBS systems, that help viewers navigate through the sea of programming choices. Although new program content will be added, the first use for the increased channel capacity is predicted to be for enhancing pay-per-view offering [Cici93]. Similar to the approach taken by DBS, a movie will be offered on several channels with staggered start times, thus, allowing near-on-demand video.

A far more advanced scenario, one that only the most visionary of visionaries expect to be realized any time soon, will find set-top boxes that contain both compressors and decompressors. This, perhaps the full realization of the Information Superhighway, will allow subscribers to originate information, providing for true subscriber-to-subscriber communications.

Finally, note that throughout all the struggles and debates about what interactive cable television will be, and when it will happen, and everything seemingly up for grabs, there has been unwavering support for MPEG compression by all participants [John95A]. However, MPEG, a highly asymmetric algorithm, is best suited to one-way mass distribution applications. Consequently, as the story of two-way, fully interactive television unfolds, one might expect other algorithms such as H.261 for videoconferencing, H.263 for videotelephony, or advanced forms of MPEG will be needed at the set-top too. Stay tuned!

8.5 Telco Video

Under the 1992 FCC "video dial-tone service" ruling that permitted telcos to transport video services on their telephone networks, the portion of the network

[9] Channel surfing is the fine art of using the remote control to quickly scan through channels to determine what to watch.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

used for these services is regulated and must be made available to all companies seeking to distribute video programs [UPI92]. The FCC decision was followed by numerous court actions that cleared the way for telcos to someday provide video programming services nationwide [Leop95A]. With AT&T in the lead, long-distance companies and local exchange carriers (LECs) have begun efforts to define and demonstrate the types of video programming services they believe customers want[10] [Zieg93, EET95A]. This places them in direct competition with cable television systems for providing, not just video-on-demand but all the services described in Section 8.4. Telcos definitely have some technology advantages over cable systems, but they also have some liabilities: They have broadband switching networks in place (which must and are being upgraded for video distribution). They have interoperability standards that will allow nationwide operation (whereas cable systems tend to be localized and not conform to widespread standards). The piece of technology that telcos are missing is bandwidth on the local loop (see Figure 7.2), the connection between subscribers and the telephone network. When running standard telephone line protocols, the local loop provides data rates ranging from 28.8 Kbps or less (analog) to not more than 144 Kbps (ISDN)—bandwidth sufficient for videoconferencing, perhaps, but far less than sufficient for high-quality video entertainment. At least 1.5 Mbps are needed for even moderate-quality MPEG-compressed video.

To solve their bandwidth problem, telcos have several choices: One is to replace existing twisted-pair copper wire local-loop connections with coaxial or fiber optic cables. Although this is where telcos see themselves ultimately heading, the large installed base makes it very expensive, and most expert observers both inside and outside the telephone industry agree that the time for completing this transition is far in the future. AT&T has proved another choice in the form of ADSL (Asymmetric Digital Subscriber Loop) technology. ADSL is a new signaling method that allows downstream (network-to-subscriber) video transmissions to proceed at 1.5–6 Mbps rates over twisted-pair copper local loops that are not longer than 18,000 feet.[11] As shown in Figure 8.4, ADSL also provides a low-bit-rate bi-directional (digital) control channel and a conventional bi-directional analog voice channel [Lin95]. Although intended primarily for downstream video transmissions (which explains the asymmetric part of the ADSL acronym), in the future it may be used for two-way applications, such as videoconferencing, where upstream video transmissions proceed at much lower rates [Sant93].

In combination with MPEG compression, ADSL makes it possible to transmit

[10] LEC is the name given to the Regional Bell Operating Companies (RBOC) and independent telephone companies that provide subscribers with local-loop connections to the telephone network.

[11] Standard telephone line protocols are designed to allow low-cost, reliable communications over great distances and over lines of widely variable quality. ADSL takes advantage of advances in digital filtering, advances in VLSI that make more powerful coding techniques economical, and the fact that many LECs are using fiber optics to reduce the length of the copper portion of the local loop [Chen94].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

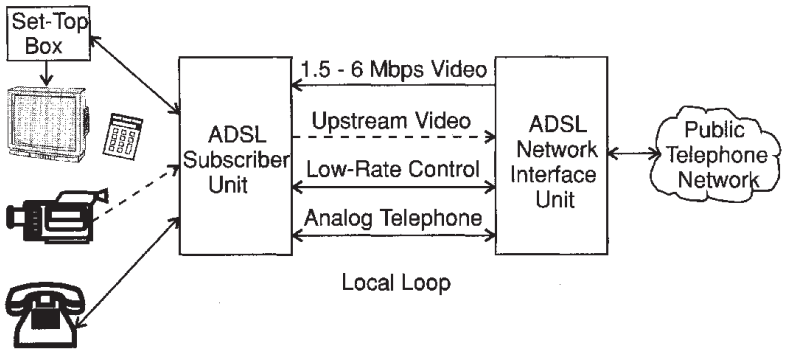


Figure 8.4 ADSL architecture.

a small number of high-quality video channels over existing copper telephone lines. The bandwidth and distance restrictions for ADSL are obvious, but, as it turns out, they are not debilitating. First, video switching is done within the telephone network. Because most subscribers are content to view one channel at a time, a local loop that carries a single video transmission is quite adequate for many video services, including video-on-demand. Second, in the United States, nearly 75% of the 100 million local loops that connect homes to local exchanges are less than 18,000 feet in length. Thus, ADSL, which is positioned as a low-cost gap-filler prior to the arrival of broadband, may be the technology that telcos need to quickly get into video distribution.

There are several versions of ADSL [Chen94]. The first, ADSL-1, operates at 1.5 Mbps for up to 18,000 feet. It allows a single VCR-quality, MPEG-1 compressed video transmission. But the competition (cable and DBS) has moved beyond MPEG-1 to MPEG-2. An updated version of ADSL, which is specified in the ANSI T1.413 ADSL standard, operates at a downstream rate of 6.1 Mbps and an upstream rate of not more than 640 Kbps for distances up to 12,000 feet [Bain95, Lieb96]. This will allow one very high-quality MPEG-2 encoded video transmission or several transmissions at lesser quality.

An important point must not be overlooked: Our discussion and the development of ADSL focuses on video transmission, but ADSL is a transmission technology that can transport any type of digital data over twisted-pair telephone lines at megabit rates. As a result, ADSL-based modems allow high-speed connections to digital data networks such as the Internet [Gree95B]. Also, complementary and competing symmetrical transmission technologies are under development that will increase the data rates in both the upstream and downstream directions [Gree95A, Bell96].

8.6 Wireless Cable

Wireless cable provides yet another means for delivering video, one that has proven popular where, for a variety of reasons, wired cable systems are difficult

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

to install. This includes large metropolitan areas, some rural communities, and developing countries where no infrastructure exists for wired systems. Wireless cable systems use microwaves to send signals from land-based transmitters to subscribers' rooftop-mounted receiving antennas. Much like DBS (and wired cable and telco video systems too), the received signals are sent to set-top boxes where they are descrambled and converted for viewing on standard television receivers. The technology is not new, as the first U.S. wireless cable systems went on the air in the mid-1970s. However, little growth occurred until the 1990s when the FCC allocated more channels to wireless cable systems and allowed them to offer the same programming as cable television; this includes rebroadcasts of local-broadcast television stations, an advantage not currently shared by DBS. Now, in the mid-1990s, U.S. telephone companies are heavily investing in wireless cable systems as a short-term solution for providing video services [Wyli95]. What makes it attractive is that, when compared with the technical and economic problems of upgrading their wired infrastructure to carry broadband traffic and considering that wireless cable systems do not obey the regulations and requirements restricting wired cable franchises, wireless cable offers telephone companies the prospect of lower investment costs and shorter deployment times to enter the video delivery business.

Existing wireless cable systems use a technology known as multipoint multichannel distribution service (MMDS) to broadcast analog video to subscribers. They operate at microwave (2-GHz band) frequencies that lie above the standard broadcast channels. This limits reception to, typically, a radius of 25 to 30 miles because the path between the transmitter and a receiver must be essentially free of any obstructions for what is sometimes called line-of-sight transmission. MMDS operates within a 200-MHz bandwidth and, therefore, existing systems can carry no more than 33 analog 6-MHz television channels, which, in today's marketplace, makes them noncompetitive with DBS and wired cable. The solution is to go digital and use MPEG-2 data compression to boost carrying capacity to more than 125 channels. With FCC authorization and digital technology comparable to that used by DBS and digital wired cable systems, the first digital MMDS system went on the air in 1996 [Piet96].

An emerging technology for wireless cable systems known as local multipoint distribution system (LMDS) has the potential to greatly increase their carrying capacity. Not only is it digital, but LDMS is allocated far more bandwidth than MMDS. There is also the potential for interactivity because LDMS allows two-way transmissions. LMDS uses ultrahigh-frequency (28-GHz band) microwaves. Because these signals are greatly affected by obstructions, they must be sent from lower-powered transmitters over shorter distances. Reception is limited to approximately a 3-mile radius in experimental systems now being tested and, therefore, these systems follow the model of cellular telephony, depending on multiple transmitters to cover a larger area. With more than one GHz of bandwidth allocated to LDMS, hundreds of compressed video channels can be transmitted. With its cellular structure and two-way transmission capability, in the future, it

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

may support data and telephone transmissions, too, although the technology for this does not yet exist. It is too early to tell what services LDMS will offer, how it will compete with other wireless services for digital-bit delivery (such as those listed in Table 7.2), or, more important, who will bring it to market. As of this writing, LMDS technology exists only in prototype one-way wireless cable systems; 200 MPEG-2 compressed video channels were provided by a system tested in Atlanta, Georgia [Anto96].

8.7 Digital Radio

Digital radio is a new technology that will enable broadcasters to transmit CD-quality digital audio without interference or perceptible loss of quality. Digitization not only eliminates the noise that accompanies AM broadcasts and the multipath fading of FM broadcasts, but with error detection and correction, it allows error-free reconstruction of the audio signal just as it was originally broadcast. No one yet believes digital radio will totally replace either AM or FM, but its backers expect digital will quickly become the premier radio broadcasting method. It is not just the superior sound quality that makes digital radio attractive, for digital radio also can transmit text-based information. This will allow broadcasters to offer a host of ancillary services such as audio-visual programs, value-added services, and other new program features. Receivers equipped with text displays will provide listeners with station call letters, song titles, information about the artist, and about the recording being aired. Other possibilities include program guides describing upcoming broadcasts, traffic reports for motorists, weather reports including maps, and (yes!) advertising.

Digital radio goes by several names: The FCC calls it digital audio radio services (DARS) which is sometimes shortened to DAR. In Europe and most of the world, it is known as digital audio broadcasting (DAB), the term we will use. Part of the confusion is that many different systems have been proposed for both satellite and terrestrial transmission. There are a variety of schemes for allocating broadcast spectrum to DAB, with little agreement on how to use it [Jurg92]. Satellite broadcasting proposals abound in the United States and in Europe and elsewhere in the world [Jurg93, Leop95B]. Terrestrial DAB systems are equally diverse. In Europe, the EUREKA 147 DAB system has been demonstrated by commandeering portions of the television broadcast spectrum [Pros94]. In Canada, EUREKA 147 DAB is being tested using newly allocated spectrum; it is planned to be a replacement service for existing AM and FM stations. In the United States, the favored terrestrial broadcasting scheme is IBOC (in-band, on-channel), which uses existing AM and FM radio broadcast and transmission facilities. With IBOC, DAB signals are placed within the broadcaster's existing frequency assignments, making it a somewhat compatible scheme. FM broadcasters, for instance, could simultaneously broadcast an analog and a digital signal within the 200 KHz frequency spectrum they are currently allocated.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Broadcasters recognize that CD-quality digital audio sounds great, but, due to spectrum scarcity, the data rate is far too high to transmit in any economical or feasible way—unless data compression is used. This most assuredly is true for terrestrial broadcasting, where radio channels are allocated only a fraction of the needed bandwidth; and satellite broadcasting, where more bandwidth is potentially available but is mostly given over to higher-paying video services. As observed in Section 8.3, DSS airs about 30 MPEG-compressed audio channels. All current and proposed DAB systems include audio compression, some variant of, or proposed extension to MPEG-2 audio coding [Jurg96]. ITU-R (International Telecommunication Union–Radio) recommendations for DAB specify MPEG audio coding at rates of 192 Kbps and less [Noll95]. This coding rate is sufficient for broadcasting two-channel stereo audio but not for 5.1-channel surround sound. For further discussion of MPEG audio compression, see Section 5.3.

8.8 Summary

This chapter examined communications applications where data compression is used for efficient transmission. We learned the following:

- Profound changes are taking place in broadcasting with broadcasters lining up to provide more personal, more customized delivery of home entertainment and information services, all of which depend on data compression for economical delivery of voice, data, images, and video.
- MPEG compression is an integral part of the standard for HDTV and SDTV digital television.
- Direct broadcast satellite television became an overnight sensation, thanks in part to MPEG compression that allows four to eight times more channels.
- Cable television is transforming itself to provide a host of new interactive services that depend on digital television and MPEG compression.
- Telephone companies have similar objectives and look to MPEG data compression as a key technology for enabling video dial-tone service over existing phone lines.
- Wireless cable systems are moving to digital, MPEG-compressed transmissions to become competitive with other video services.
- Digital radio, a new technology that will allow broadcasters to provide CD-quality audio (and text), depends on MPEG audio compression to squeeze digital audio into available radio broadcast channels.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

9

Consumer-Electronics Applications

In this chapter, we examine consumer-electronics applications where, primarily, data compression is used for efficient data storage. This comes as no surprise because many consumer-electronics products for entertainment and information use personal storage media, discs and tapes today, with solid-state storage a future possibility. Data compression will also be important for future, communications-oriented consumer-electronics products, like HDTVs, where it will be part of the transmission standards. Lossless compression is used for text and other symbolic data, whereas various combinations of lossless and lossy compression are used for speech, audio, image, and video data.

9.1 Industry Overview

Were they living today, Thomas Edison and Guglielmo Marconi, the 19th-century inventors of the phonograph and radio, would surely be amazed by the variety of products on the shelves in 1990s electronics superstores. Beginning with record players and radios, and then television, an ever increasing diversity of consumer-electronics products provide home entertainment and information. The definition of what comprises consumer electronics is a bit fuzzy. Once, consumer electronics meant audio gear, radios, and television receivers, but today the range of products is much broader. Many of today's consumer-electronics products were first introduced for industrial or office applications and then migrated into the mass consumer market only after technology advances reduced their cost and features were added that improved their ease of use. The personal computer is but the latest example of a product making the transition from office to home. For our

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

purposes, we will keep it simple: A consumer-electronics product is something sold by Best Buy, Circuit City, and other electronics retailers.

The rapid growth of consumer electronics can be traced to technology advances—low-cost, integrated-circuit semiconductor electronics, lasers, improved visual displays, smaller and more accurate audio reproducers, and advances in scanning and printing techniques to name a few. Low-cost mass production techniques for small electromechanical devices also play a role. Without a doubt, no factors are more important than the introduction of digital technology, the convergence of computers and communications, and the integration of audio, video, and computer-based equipment. Text, graphics, speech, audio, image, and video data all play starring roles in consumer-electronics products. The ability to store, process, and communicate all types of digital information is revolutionizing consumer electronics. This translates to improvements of existing products and to totally new products. Sometimes, it positions new, digital consumer-electronics products to eclipse older products based on analog technologies—film cameras, for example.

Every day it seems there are announcements of new, digital consumer electronics that will relegate today's analog products to join vinyl long-playing records and black-and-white television sets in the museum of old-fashioned, formerly dominant technologies. However, gee-whiz technology and amazing features do not always assure success. Consumers do not always assume new is better and, often, they are not easily motivated to pay more for it. Then, too, consumers may not be ready to give up on the products they already have, quashing sales opportunities for replacements, or they may not be interested in making revolutionary advances in how they are entertained or work. Simply pinning the digital label on a product does not influence consumers to buy it. Similarly, declaring that data compression is used in a product may not have much influence on consumers either, unless its advantages are obvious.

Earlier in the book data compression was said to be an essential ingredient in the digital revolution taking place in consumer-electronics products—and it is—for it is needed to tame the storage and transmission demands of digital data. However, in consumer electronics, data compression is neither totally transparent nor always a benign servant. Sometimes, but not always, data compression adds significant product cost when measured in additional logic chips and storage buffers. If so, its use may be constraining to high-end products, where cost sensitivities are not so great, but this is a double-edged sword because consumers expect higher quality when they pay more. The problem with today's data compression, at least for audio and video, is that it can compromise quality. Technologists may argue that the degradation can only be measured with sophisticated instruments, but consumers with keen ears and sharp eyes can (or think they can) spot the differences. This, along with high prices, has slowed sales of some compression-based digital products, particularly when those products are forced to compete with entrenched products that consumers consider to be satisfactory, if not superior.

Data compression standards for consumer-electronics products come from

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

sources within and outside the industry. Historically, the industry has created its own unique, de facto standards—and this now includes data compression standards—for its storage-based products and convinced other industries to produce media that conform to those standards. In contrast, HDTVs and other communications-based consumer-electronics products must conform to standards set by the communications industry.

This chapter describes consumer-electronics products for which, with varying degrees of success, data compression is used for text, graphics, speech, audio, image, and video to provide entertainment and information. Additional compression-based products found at electronics retailers such as telephones, FAX machines, and some computer equipment were described in Chapter 7.

9.2 Digital Audio—Stereo Gear and Home Theater

By the early 1990s, the digital audio compact disc (CD) had displaced analog playback media. Vinyl long-playing records had all but disappeared, and sales of prerecorded cassette tapes continued their long decline. For various reasons, be it superior sound quality, durably, or whatever, the CD became and remains the largest-selling consumer audio storage media. About the same time, a new concept—home theater—was beginning to emerge. Home theater, the combination of large-screen television receivers and multichannel audio, allows home viewing of movies and other entertainment with all the impact once reserved for commercial theaters. In this section, we will look at home audio products that, for storage and transmission efficiency, use psychoacoustic audio compression techniques of the type discussed in Section 5.3. See Figure 5.4 for a block diagram of the encoding and decoding processes and Table 5.3 for specifications of the perceptual coders mentioned in this section.

Stereo players and recorders

Audio CDs, excellent as they are, have some limitations: They are not recordable, at least not the versions in consumers' hands. They are not ideal for portable applications—the discs are larger and harder to handle than cassette tapes, and when bumped, CD players are prone to skipping tracks. Analog cassette tapes, long dominant for recording and portable listening applications, are limited, too, in that they cannot match the audio quality of CDs. Steadily declined sales led market analysts to conclude that a replacement was needed for the venerable analog cassette, a replacement that would combine the audio quality of CDs and the recordability and ruggedness of cassettes, all within a compact package. This section looks at three digital audio products that stake a claim on the portable CD and analog cassette marketplace. All use data compression. Their media are depicted in Figure 9.1.

The Sony™ MiniDisc™ (MD) system was introduced in 1992. It is recordable

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

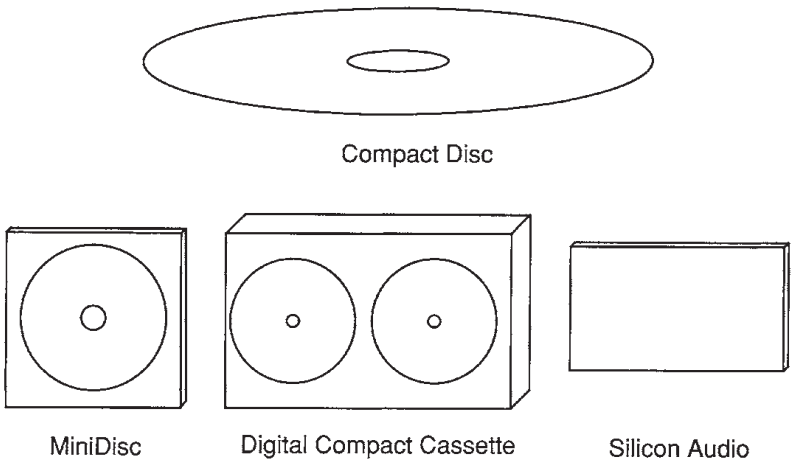


Figure 9.1 Audio media.

(allowing it to replace the analog cassette tape), smaller than a CD (allowing lightweight portable players), and, with the same portable applications in mind, it is shock resistant. The new technology in MD is extensive: Magneto-optical discs provide recordability, whereas conventional read-only optical discs are used for music and software distribution. A half-size disc (64-mm-diameter MDs versus 120-mm-diameter CDs) encased in a hard shell provides small size and durability. Buffer memory for the bit stream coming off the optical pickup allows MD players to mask bump-induced interruptions, a technique which, later, was added to portable CD players. MDs and CDs are alike in other ways too: Both use prerecorded discs made of polycarbonate, whereon tracks of information are recorded as microscopic pits in an aluminum reflective layer. Both the prerecorded and recordable versions of the MD use track widths, track spacings, and data-encoding methods that are similar but not identical to the CD. For a full discussion of MD technology, see [Rana93, Nish94, Yosh94C].

An important difference between MD and CD is that the smaller MD holds only 140 MB of data, about one-fifth as much as the standard CD. To provide the same 74-minute playing time that CDs offer, audio data compression is a necessity. Starting with the 16-bit quantization and 44.1-KHz sampled stereo audio signal used for CDs, Sony engineers developed ATRAC (Adaptive TRansform Acoustic Coding), a proprietary psychoacoustic audio compression method. It is a hybrid coder: The encoding process begins with bandpass filters that break the audio spectrum into three subbands. Then a modified discrete cosine transform is applied to generate 512 spectral components. The output bit stream is generated by quantizing the spectral values and assigning bits according to psychoacoustic principles [Yosh94C]. ATRAC was designed to provide a 5:1 compression ratio, just the amount needed for making up the capacity difference between the MD and CD disc formats.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Philips introduced the Digital Compact Cassette™ (DCC™) system to the marketplace in 1992, positioning it as an extension of the Compact Cassette™ (analog cassette). It improves upon analog cassettes with sound quality that approaches the CD, and it offers better track-search cueing, text information for song titles, and a cassette that is less vulnerable to damage. Playing time, up to 120 minutes, is identical to the analog cassette because both use the same width tape and move it at the same speed. DCC tape transports differ in minor ways from analog tape transports, but either cassette may be loaded in DCC machines because their external dimensions are nearly identical. This makes DCC machines backward-compatible, providing playback for the large inventory of analog cassettes now in the hands of consumers. Internally, DCC technology is quite different, beginning with the tape itself, which is made of high-performance, video-grade material. New, thin-film, magneto-resistive tape heads can play back two tracks from analog cassettes and lay down and play back nine much narrower tracks for DCC. These tape heads increase the data rate for digital recording, but, still, the raw data rate is only 768 Kbps. System information—error correction, synchronization, and addressing symbols—uses half the bits, leaving only 384 Kbps available for audio recording, far short of the 1.4112 Mbps needed for CD-quality stereo. For a full discussion of DCC technology, see [Rana92, Hoog94].

To squeeze CD-quality stereo onto DCC, Philips engineers developed Precision Audio Subband Coding (PASC), a proprietary psychoacoustic audio compression algorithm. As the name implies, it is a subband coder. The encoding process begins with bandpass filters that break the audio spectrum into 32 subbands. Then the compressed output bit stream is generated by quantizing the filter output values and assigning bits according to psychoacoustic principles based on the hearing threshold and signal masking[1] [Hoog94]. The compressed output bit stream is then combined with system data and distributed over eight of the nine magnetic tracks. (The ninth track is used for recording auxiliary information that provides time, text, and control signals which contribute to the improved cueing function.) PASC was designed to provide a 4:1 compression ratio, just the amount needed to make up the difference between DCC and CD data rates.

In 1995, NEC Corporation demonstrated Silicon Audio™ technology, an audio-data storage system that uses semiconductor flash memory cards. Although no products are yet available—and none will be until electronic memory costs decline dramatically—audio players (and recorders) no larger than a deck of playing cards are envisioned for future products. Not only will Silicon Audio players be a fraction of the size and weight of current cassette tape and CD players, they will also be rugged and reliable because there are no moving parts. For a full discussion of Silicon Audio technology, see [Sugi95].

The storage technology for Silicon Audio is shared with computer applications that use electronic memory storage devices. (See the description of solid-state

[1] Up to this point, PASC coding is identical to MPEG-1 audio coding.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

files in Section 6.6.) The problems with using this technology in consumer-electronics products are packaging density—stuffing sufficient memory into a small package—and cost. For a solution, NEC engineers turned to MPEG-1 layer II audio compression operating at 96 Kbps per channel (providing slightly more than 7:1 compression), a rate sufficient for near CD-quality sound. This allowed them to load 12 minutes of stereo music onto a 16-MB card. But for playing time equal to one CD, six of these cards are needed, and at mid-1990s prices, they would sell for about $6000! Thus, major advances must be made for solid-state storage to be a serious contender for home audio media.

Home theater

Multichannel sound for home theater recreates the conditions found in a movie theater by placing left, center, and right channel speakers across the front of the room near the television screen and surround channel speakers at the sides of the room, as shown in Figure 9.2. The sixth speaker, a subwoofer for low frequencies, is usually placed at the front too.

Initially, analog matrixing techniques were used to squeeze multichannel audio into the two stereo channels provided by laser videodiscs, VCRs, and stereo television. With matrixing, the center channel and a monophonic surround channel were encoded atop the left and right audio channels. During playback, they were extracted by decoding logic in the home audio/video receiver that amplified all four channels and directed low frequencies to the subwoofer. Various matrix-surround sound techniques were invented—Dolby Surround™, Dolby Pro Logic™, and others—but none could overcome the fundamental limitations of matrix encoding; namely channel separation between the left and right channels was high, as was the separation between the center and surround channels. But

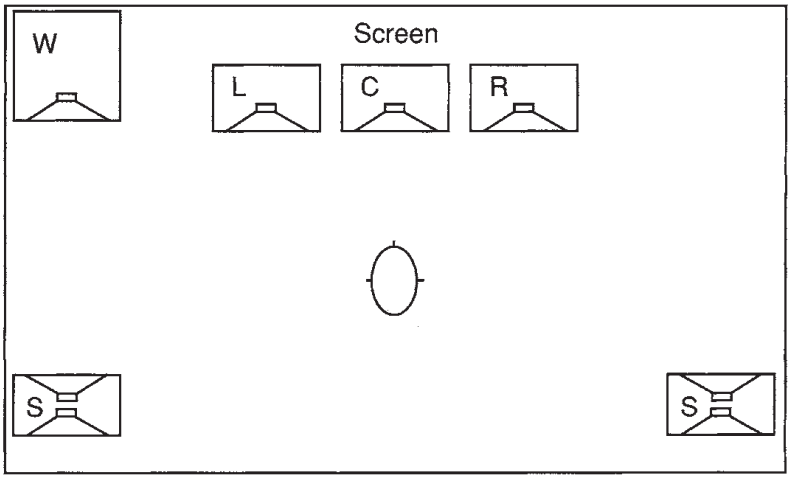


Figure 9.2 Home theater speaker placement.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

between the center and left or right channels and between the surround and left or right channels it was quite low. Even with careful encoding, there was channel bleed-through, the dynamic range was limited, and not all combinations of sound could be reproduced. All these limitations detracted from the illusion of 3-D (three-dimensional) sound.

The solution is, of course, to provide discrete channels, but the basic problem is fitting the extra channels into existing disc, tape, and broadcast audio/video media. Using perceptual audio coding, a discrete-channel system known as 5.1-channel digital surround sound provides that solution. Five full-range channels for the left, center, right, and two surround channels, and a sixth channel for the subwoofer, which is limited to frequencies below 120 Hz, are provided. This allows 5.1-channel digital surround systems to reproduce a much more accurate 3-D soundfield. One perceptual audio coder, Dolby AC-3 encoding, has been adopted for all media to provide multichannel audio transmissions that accompany video [Rigg95, Taki95B]. As learned in Chapter 5, AC-3 incorporates an advance transform coder that can encode and transmit 5.1 channels in just 384 Kbps, a compression ratio approaching 11:1. It does this by using psychoacoustic principles and exploiting the correlation between individual channels. All channels are compressed and combined into a single composite signal; this signal can be recorded on all existing stereo media and on new media such as digital video discs and can also be transmitted by conventional television and HDTV. Products enabled for AC-3 began appearing in 1995.

Compressed digital audio in the marketplace: is it a winner?

Since the introduction of MiniDisc and Digital Compact Cassette and, later, with digital surround sound, psychoacoustic audio compression has been a subject for microscopic examination and, at times, heated debate. Some see it as a leading-edge technology to be used in innovative new audio products of all kinds, whereas others believe it to be a colossal blunder. To oversimplify the controversy about psychoacoustic audio compression, videophiles seem willing to accept the minor aberrations it introduces, whereas audiophiles definitely remain unconvinced.

There are some marketplace factors that may help to explain why the reactions to psychoacoustic audio compression are so varied. To begin with, the criteria for sound quality are different for audio/video versus audio-only products. For the home theater market, which is young and continues to grow wildly, technology and consumer expectations are still evolving. Surround sound, whose primary goal is realistic 3-D soundfield imaging, is acknowledged to be vastly improved by Dolby AC-3's fully discrete channels, stereo surround channels, and increased dynamic range. The quality of home theater sound is judged by more than the fidelity of music. Both experts and consumers appear willing to accept the slight imperfections that perceptual coding may add to their favorite movie soundtracks

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

in favor of improvements to 3-D soundfield imaging. In contrast, the home audio market is saturated with products that offer extremely good stereo sound at very low prices. For several decades, stereo sound quality has improved, with audio professionals and consumers, alike, coming to expect the next generation of products to be even better. This is not an easy task, especially when the accuracy of stereo reproduction of music is the major or only factor (other than price) for judging goodness. But the search for higher levels of perfection continues; techniques such as higher sampling rates and 20- or 24-bit samples are being considered for improving on CD-quality sound [Pohl95C]. The introduction of perceptual coding to this environment violated the expectations for higher quality. Audiophile publications were quick to point this out, and they continue to warn that MD and DCC sound quality does not meet CD standards. In fairness, because MD and DCC aim to replace analog cassettes, perhaps neither should be judged by CD standards, but they are. However, both MD and DCC products currently are far more expensive than the analog cassette tape or CD products they intend to replace; so, applying higher standards may not be unreasonable.

There are some technical factors too. Not all perceptual coding algorithms are created equal. In praise of Dolby AC-3, experts often point out that it uses a highly efficient, sophisticated second-generation perceptual coder (an earlier version is used extensively in broadcasting) which can hide or reduce the sonic imperfections of perceptual coding [Taki95B]. What they are talking about is that AC-3 simultaneously encodes all channels and can exploit the correlation that exists between channels, an effect that grows with more channels. The encoder looks at what is happening in all channels and allocates to each of them bits from a shared pool. The channels with the greatest needs get larger proportions of the total bits available in the combined pool. This results in greater coding efficiency (higher compression ratios), and the overall sound quality can be much better than could be obtained with multiple, single-channel encoders. In contrast, ATRAC, in MD, and PASC, in DCC, encode each channel independently. Each channel receives a fixed number of bits, which may be more or less than needed for the highest sound quality. However, if simultaneous encoding were used in these applications, it would likely be less effective as there are only two stereo channels with which to work.

Another technical factor is that MD and DCC products can both decode and encode audio, unlike home theater equipment that only decodes. Audiophiles—and broadcasters who have devised demonstrations to prove the point—are concerned with the degradation that occurs when compressed media or compressed broadcasts are copied. Due to decoding and reencoding, minor sonic aberrations introduced by perceptual coding can be magnified in unpredictable ways[2]

[2] Both MD and DCC, like all digital audio recorders intended for U.S. consumer applications, incorporate circuitry that prevents making a direct digital copy of a source that is, itself, a direct digital copy of an original. When a compressed source is copied, it must first be decompressed before a copy can be made.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

[Rana94]. Thus, at its present stage of development, perceptual audio coding may be quite satisfactory for playback-only applications but far less viable for home recording.

9.3 Digital Television

Without thinking much about it, we have all used "digital" television. Starting somewhere in the 1980s, manufacturers began incorporating digital functions in millions of "analog" NTSC, PAL, and SECAM broadcast-standard television receivers. Digital technology is used for everything from channel selection, to picture processing, to feature enhancements both spectacular, as is picture-in-picture, and mundane, as is displaying the time of day. The new wrinkle is that television receivers must be prepared to receive digitized, compressed broadcasts—what in the nineties we choose to call digital television.

As learned in Chapter 8, by the mid-1990s standards for terrestrial broadcasts of digital television in standard-definition television and high-definition television formats were close to completion. Meanwhile, NTSC standard television receivers were already receiving digital pictures and sound from direct broadcast satellite systems and, soon, would be from wired and wireless cable systems and telco video systems. All digital television broadcasting methods use MPEG (or Dolby AC-3) audio compression (described in Section 5.3) and MPEG video compression (described in Section 5.5). This and immediately following sections look at how data compression applies to the consumer-electronics products for receiving, reproducing, and recording digital television.

Drop into any electronics store today and you will find a wide choice of television receivers and peripheral devices including set-top boxes, VCRs, video camcorders, videodisc players, and other audio/visual products. As illustrated in Figure 9.3, all can interoperate thanks to the NTSC (PAL or SECAM) transmission standard for analog television.

Now, fast forward to the year 2000—when digital television is in full bloom—and visit the same store. Will you find digital television receivers and digital peripheral devices that interoperate using the SDTV/HDTV transmission standard for digital television transmission? Silly question; of course you will, because this is *the* standard for digital television, right? Actually, you may find something quite different. One reason is that digital set-top boxes, VCRs, video camcorders, videodisc players, and other audio/visual products are reaching the market today without the benefit of a crucial industry standard—a standard for a digital interface between peripheral devices and television receivers. These products all work with analog television receivers, but how they, or their successors, will interoperate with digital television receivers is unclear. The fundamental problem is that the consumer-electronics industry has not come to grips with how to handle data compression in peripheral devices. The problem is exacerbated when those devices must interoperate with digital television receivers that in themselves incorporate data compression, receivers which conform to the SDTV/HDTV transmission

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

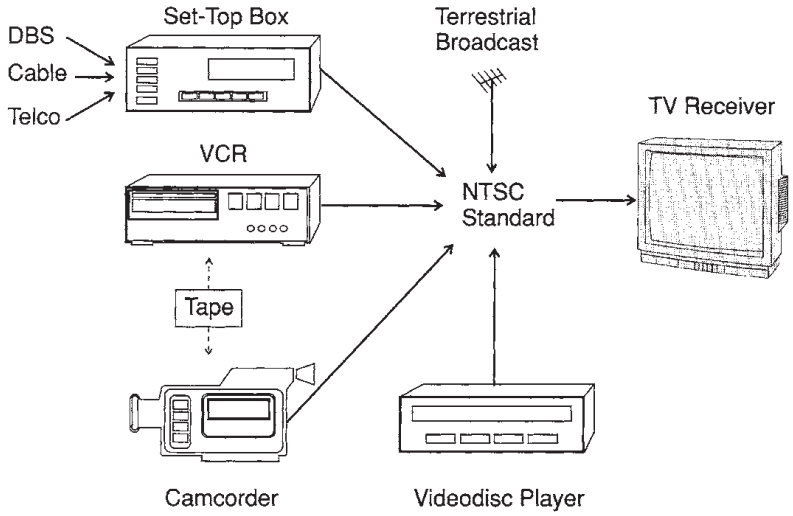


Figure 9.3 Television reception.

standard for terrestrial television broadcasts. To understand the difficulties that compressed digital television creates, let us explore what is happening in peripheral devices.

9.4 Digital Set-Top Boxes

A set-top box similar to that shown in Figure 9.4 is needed to receive digital television broadcasts on a standard analog television receiver. Digital set-top boxes are widely deployed for DBS and soon will be for cable television and telco video systems. While still in the formative stages, a digital set-top box may provide the solution for receiving digital SDTV on analog television receivers.

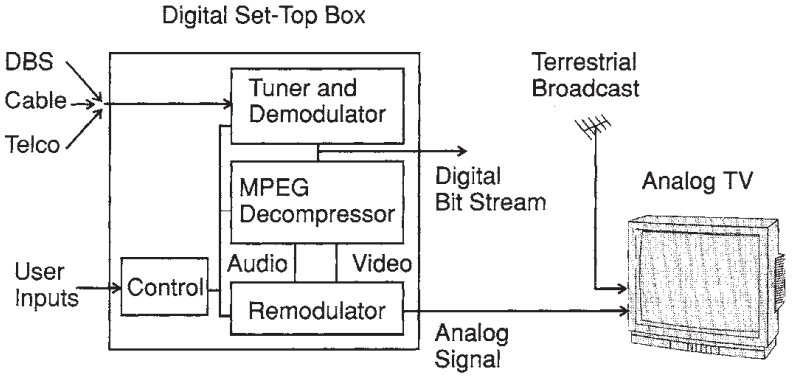


Figure 9.4 Digital television reception on analog television receivers.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Today's digital set-top boxes build on the capabilities found in convertor boxes used by (analog) cable television systems. To oversimplify the process, a channel is selected and the signal is demodulated. MPEG decompression is applied. Then the audio and video signals are remodulated to analog NTSC (PAL or SECAM) format for output on an unused television channel. Picture and sound appear when the television receiver is tuned to that channel. (The DSS set-top box for DBS television also outputs a compressed digital bit stream; more about this is coming up shortly.) A remote control for the set-top box allows users to select the digital channel for viewing and to control whatever interactive features are available.

A point often overlooked is that the tuning function in the analog television receiver is used only for receiving terrestrial broadcasts. A set-top box takes over the tuning function, nullifying or essentially bypassing the receiver's tuner, but that is not of much consequence with analog television.[3]

Now, consider how digital television sets of the future will receive digital television broadcasts. To *directly* receive SDTV/HDTV transmission-standard terrestrial broadcasts, without the aid of an external set-top box, digital television receivers will need a built-in tuner and an MPEG decompressor. But will they also receive digital broadcasts from DBS, cable, and telco networks in a way similar to that shown in Figure 9.3? The answer probably is "no," and here is why: When digital television sets are used with an external set-top box, which is needed to adapt to the differing broadcast standards of DBS, cable, and telco networks, the (tuning and) decompression function built into the television set must be nullified or bypassed—if the set-top box delivers a decompressed signal. Therein lies a problem that the consumer-electronics industry has yet to solve. Here are some possible solutions:

- A simple approach is for the set-top box to deliver a MPEG-compressed digital bit stream that the digital television receiver can decompress. To make this happen, a standard for compressed digital bit-stream inputs for television receivers must be developed. However, digital television service providers may be uninterested in this approach, for it reduces the set-top box to a simple digital version of the old analog cable television convertor box, potentially robbing the set-top box of interactive features.
- Digital set-top boxes of the future could include MPEG compressors, allowing the television signal to be remodulated to the SDTV/HDTV transmission standard for terrestrial broadcasts. Although fully supporting interactivity, it is not very cost-effective because MPEG compressors are complex and expensive. In addition, stacking lossy MPEG decoders and encoders can degrade sound and picture quality.
- Another possibility is to integrate the set-top box function into the television receiver.

[3] The only time anyone notices that the tuning function is done in the set-top box is when a VCR is used to record programs and channel-selection logic in the VCR cannot control tuning in the set-top box. Fortunately, ingenious but kludgy solutions have been devised.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

This raises a host of marketing and technical issues: Whose set-top should be built-in, or should all of them? How will the set-top box dataflow be integrated? The industry has struggled with these issues for years and has not yet found a solution [Cici95].

- Probably the cleanest choice is that digital television receivers have inputs to accept decompressed audio and video signals, effectively bypassing the tuning and decompression functions. To make this happen, a standard for interoperating with decompressed signals must be developed.

9.5 Digital VCRs and Camcorders

As shown in Table 9.1, over the years an unending series of videotape recording formats has been introduced for professional and consumer applications. Some

Table 9.1 Digital videotape recording formats.

Recorders[1]	Format[2]	Year	Tape Width	Playing Time
Professional:				
Analog	Various	1961	2 in.	60 min
	Various	1967	1 in.	60 min
	U-matic	1969	¾ in.	60 min
Digital	D1[3]	1986	¾ in.	94 min
	D2	1988	¾ in.	208 min
	D3	1991	½ in.	245 min
	D5	1993	½ in.	123 min
	Digital Betacam™	1993	½ in.	125 min
	Ampex DCT	1993	½ in.	208 min
	D6	1994	¾ in.	60 min
Consumer:				
Analog	Betamax™	1975	½ in.	2 - 5 hr
	VHS	1976	½ in.	2 - 8 hr
	8 mm	1985	8 mm	2 - 4 hr
	S-VHS	1987	½ in.	2 - 8 hr
Digital	D-VHS	1995	½ in.	7 hr[4]
	DVC	1995	½ in.	4½ hr[4]

Notes:
[1] Sources: [Hama95, Roth95, Warn96].
[2] All are compliant with NTSC, PAL, or SECAM standards except D6 which is intended for HDTV.
[3] D1 - D6 are SMPTE digital formats.
[4] Playing time will be cut by one-half for a future HDTV mode.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

formats appeared and quickly disappeared, having been supplanted by better technology or having met the fate of superior marketing at the hands of their competitors. Today, digital recording systems have largely replaced analog in professional videotape recorders, and newly introduced digital formats are looking to doing the same in consumer VCRs and camcorders.

In the broadcasting industry, the advantages of digital recording, namely faultless reproduction of the video signal even after multiple copying and unerring timing stability for video and audio, are widely recognized. Starting in 1986, these advantages were provided to professional videotape recorders by the SMPTE (Society of Motion Picture and Television Engineers) D1 format and others soon to follow [Hama95]. Digital video demands recording at data rates far higher than analog, and, initially, high-quality pictures were achieved by purely electromechanical means—wider tape, multiple stacked heads, and parallel recording channels, for instance. More recently, advances in head and tape technology and data compression has been exploited. Not only do the latest products offer the same or higher-quality pictures, but their size and complexity have been reduced, and portable professional digital camcorders are now available. Compression ratios in today's professional videotape recorders are only 2:1 or slightly more. This modest reduction is achieved by using frame-by-frame, spatial discrete-cosine transform (DCT) intraframe coding similar to that found in JPEG (described in Section 5.4). Recent announcements suggest that more powerful MPEG-based compression will soon appear in professional gear [Hara95A].

With this background, let us turn our attention to data compression in consumer video recording. At the time of this writing, the consumer-electronics industry is deeply divided on how digital technology should be deployed in consumer VCRs and camcorders. Data compression is the central issue—should digital VCRs and camcorders use data compression/decompression and, if so, what kind? Unfortunately for consumers, incompatible products are reaching the marketplace. Here is a brief overview of what is available:

In 1995, JVC (Victor Company of Japan) introduced the D-VHS™ (Data-Video Home System) format. D-VHS VCRs record a compressed digital bit stream obtained from a peripheral device, such as the set-top box shown in Figure 9.4, and play it back through the same device. Because the bit stream is compressed, the data rate is sufficiently low for existing analog recording technology to be used. D-VHS recorders essentially are standard S-VHS (Super-VHS) machines to which error correction and digital signal processing logic has been added. Currently, digital bit streams from the DSS DBS set-top boxes are accepted, but, in the future, interfaces to other set-top boxes, digital television receivers, and computers are possible [Krau95A]. The first generation of D-VHS machines record SDTV-quality signals at 14.1 Mbps, providing up to 7 hours of recording on a standard S-VHS cassette. When HDTV signals become available, the D-VHS specification includes a high-definition mode that will allow future machines to record at 28.2 Mbps, providing up to 3½ hours of recording on one tape. The chief advantage of recording compressed bit streams is that digital VCRs can be

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

much simpler and cheaper, for they contain no compression or decompression hardware. The downside, beyond requiring multiple interfaces because there is no standard yet for digital bit streams, is that digital camcorders cannot follow the same path. D-VHS compatible camcorders, should they become available, will need compression hardware to generate a compressed digital bit stream.

Also entering the marketplace in 1995 were products that conform to the Digital Video Cassette (DVC) standard. In 1994, a consortium of over 50 companies reached agreement on specifications for standard-definition and high-definition DVCRs (Digital VCRs) and camcorders that *do* incorporate compression and decompression. The DVC standard allows uncompressed video to be recorded digitally and, initially, decoded to analog for playback through NTSC-standard television receivers. Digital outputs also are available for making digital-to-digital copies with DVC-compliant VCRs and camcorders. In the future, digital outputs to personal computers will allow for DVC products to be part of computer-controlled video editing and production facilities [Hara95B]. First-generation DVC products provide over 500 lines of horizontal resolution, far better than the best analog VCRs which are limited to about 400 lines. SDTV-quality signals are recorded at 25 Mbps on a new ¼-inch format tape cassette that provides up to 4½ hours of recording time. When HDTV signals become available, the DVC specification includes a high-definition mode that will allow future machines to record at 50 Mbps, providing up to 2¼ hours of recording on one tape. The data compression found in DVC products is not MPEG but, rather, a frame-by-frame, DCT coding technique similar to that found in professional video gear. In the first-generation DVC products, the compression ratio is about 5:1.

Which solution will consumers embrace, D-VHS, DVC, or yet another alternative? Some industry observers assert that the D-VHS VCR will be just an interim device and all-digital VCRs that have the technology to decode compressed video signals will prevail. By the time you read this, the industry or the marketplace may have settled on a single solution. If not, welcome to a digital version of the Beta-VHS format war of the 1980s! Whatever the outcome, data compression will be part of the solution. Now, let us explore in greater depth the need for and the kind of data compression appropriate for digital VCRs and camcorders.

Video recording places different requirements on data compression than any of the video transmission applications that we have examined. An ideal data compression algorithm for video recording would:

- Provide symmetrical encoding and decoding time since video recorders must both record and playback
- Cause no (or very little) loss of image quality as video recorders are intermediate storage devices and tapes are often copied
- Allow each frame to be easily reconstructed for frame-by-frame access and allow moving forward and backward in the video stream during editing
- Allow decoding at varying speeds to provide a viewable picture during multispeed playback (slow-motion and high-speed picture search modes)

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

- Operate with widely-fluctuating bit error rates (caused by tape dropouts) encountered in magnetic recording channels
- Compress video sufficiently to overcome the limited data rates in the recording channel and the limited capacity and recording time of the recording media.

Clearly, none of the video compression algorithms examined so far meet all these requirements. The leader in transmission applications, MPEG, is not an ideal coding scheme for video recording: It is an asymmetric algorithm that requires sophisticated encoding hardware to achieve high compression ratios, which translates into expensive, power-hungry camcorders. It also makes extensive use of interframe coding, which makes frame-by-frame operations in VCRs difficult and expensive. The DCT techniques used in professional videotape records and DVC products, essentially the same as found in M-JPEG (motion-JPEG), come closest. (M-JPEG, as described Section 5.5, compresses video frame by frame as a sequence of still images.) But, can M-JPEG or any simple intraframe coding technique provide sufficient compression? To address this question, consider the following:

The recording mechanisms in consumer VCRs and camcorders must be simple to produce low-cost, small-sized devices for portable applications, and consumers expect ever-smaller tape cassettes to provide several hours of recording time. These two factors limit the recording rate and, essentially, determine how much compression is needed [With92]. Without going into great detail, the recording rate for today's consumer gear is limited to about 25 Mbps, with 50 Mbps projected for the future, and the maximum storage capacity of cassettes is about 50 GB for VCRs and 11 GB for camcorders [Hama95, Roth95]. This translates into the playing times and compression ratios shown in Table 9.2. Note that in computing compression ratios, the overhead for recording channel error correction is assumed to be 20% [With92].

Table 9.2 Digital VCR and camcorder playing times and compression ratios.

Recording Rate	Playing Time		Compression Ratio	
	50 GB cassette	11 GB cassette	SDTV @ 100 Mbps	HDTV @ 1.2 Gbps
25 Mbps	4.44 hr	1.0 hr	5:1	60:1
50 Mbps	2.22 hr	0.5 hr	2.5:1	30:1

Recording SDTV obviously poses few problems; the moderate compression ratios needed can be obtained with DCT intraframe coding with little if any compromise in image quality. Recording HDTV is more difficult. Higher compression ratios are needed, and whatever method is chosen—DCT intraframe, MPEG interframe, or something else—image quality degradation becomes a concern. Higher recording rates, which depend on future advances in head and tape technology, may provide an alternative.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

9.6 Digital Video Disc

If predictions come true, the digital video disc (DVD) will soon be the medium of choice for in-home playback of rented movies and direct-to-video productions. DVD promises a revolution in home video. Consumers will enjoy far-higher picture quality than provided by tapes played on their analog VCRs, with the ease and convenience of a CD-sized disc. DVDs will play on multimedia computers, sparking production of media for both entertainment and education. Media producers will also benefit with lower production costs and better control over unauthorized duplication. With immense digital storage and future potential for recording, DVDs may replace VCRs and become the mass storage media of choice in multimedia computers.

A bit of history: The DVD is not the first attempt to sell consumers on optical storage for video. The analog LaserDisc™, the lone survivor of videodisc format wars fought in the 1970s, has never won much public support, with LaserDisc players in fewer than 1% of U.S. households. Long the choice of serious videophiles seeking the highest-quality pictures, LaserDiscs continue to be relatively expensive, and the availability of new and rental titles remains limited. The plight of LaserDisc can be attributed partly to marketing and partly to its ungainly 12-inch size, acceptable in the era of vinyl records but not today.

The Video CD, the first generation of digital videodiscs, came to market in 1993. Using CD-ROM disc technology, MPEG-1 data compression, and modified CD players, it allows up to 74 minutes of full-motion video and stereo sound to be recorded on a standard 120-mm CD. The Video CD was originally created for video karaoke machines but much broader uses were envisioned [Yosh93B]. Its backers strongly promoted it as the new media for distributing movies, but the movie deals never got off the ground. Hollywood turned thumbs down, being duly unimpressed by a media that provided marginally acceptable video quality for big-screen productions and could not contain a feature film on a single disc. Today, Video CD lives on as the storage media for video karaoke, video games, and limited production of movies for the Philips CD-I™ (Compact Disc-Interactive) multimedia player. Also, there has been a recent resurgence of interest in using Video CD for computer multimedia.

The influence of Hollywood media producers (and the computer industry) helped to define today's DVD. Hollywood was looking for a media that not only provided high-quality video for full-length feature films but included videophile features such as 16:9 wide-screen display formats, multilanguage audio tracks, and surround sound, and included other features such as copy protect and parental lockout. The challenge to manufactures was to come up with a high-density CD that had the storage capacity to make this possible. And that they did—in duplicate! In 1994, teams led by Sony/Philips and by Toshiba/Time-Warner developed competing high-density CD formats that used different encoding and recording techniques for placing data on 120-mm discs. With the Sony/Philips format perhaps having some advantages for the computer industry and its need for high-density CD-ROMs, both proposed formats were equally adept for video, provided

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

the storage capacity, image quality, and features for which Hollywood was looking. For a time, it appeared that both formats would be launched in the marketplace. Sony/Philips lined up CD-ROM drive makers and computer industry support for its offering; not to be outdone, Toshiba/Time Warner garnered the support of some major Hollywood film studios. What followed was a marketing blitz by both groups, each showing their format could handle both computer and entertainment applications. However, there soon were strong signals from the marketplace urging a merger. Key players in the computer industry realized it is impossible to separate PC-based multimedia applications from television-based video applications, and they would be stuck with supporting both formats [Yosh95B]. Important members of the film production and distribution industry, eyeing prospects for launching a lucrative DVD movie-rental business, realized that having two groups of consumer-electronics companies slugging it out in the marketplace was not in their interest, and reportedly threatened to embrace neither format [Yosh95F].

In 1995, a merger deal was struck, creating today's DVD. Data encoding and recording technologies were merged and a jumble of patent cross-licensing issues was resolved to produce a single standard. Actually, the standard specifies several disc formats, a multimedia CD for computers, the DVD for use in consumer video players, and future higher-density discs and recordable formats.

DVD specifies MPEG-2 video and Dolby AC-3 audio compression. Both video and audio are decompressed by VLSI chips incorporated in consumer DVD players. For now, their output conforms to either NTSC or PAL standards, with HDTV to be addressed sometime in the future. An interesting aspect of DVD is how more than 2 hours of high-quality video is squeezed onto the disc and the role MPEG-2 plays. Part of the solution is the increased storage capacity realized by reducing the physical dimensions for storing each bit and using a shorter-wavelength laser for readout. Another part relates to using highly efficient encoding and (in the future) recording on dual-layer and dual-sided discs. MPEG-2 data compression contributes too; for unlike MPEG-1, which operates at a fixed rate, it supports variable-rate encoding. This allows the bit-stream rate to be increased during high-action sequences and decreased at other times. Somewhat akin to the variable transmission rate for channels in the DSS direct broadcast satellite system, this feature of MPEG-2 is exploited by variable-bit-rate technology in DVD. By adding a variable-bit-rate-controller function to the MPEG-2 encoder and varying the DVD playback bit rate, data can be efficiently encoded and placed on the disc at a lower average transfer rate, thus extending the playing time.

A summary of past, present, and future videodiscs discussed in this section is provided in Table 9.3.

9.7 Digital Photography

If you are looking to polish your image, try digital photography! It is still not inexpensive, and the picture quality is not yet the equal of film. But digital

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table 9.3 Videodisc specifications.

	Capacity (GB)	Data Rate (Mbps)	Playing Time (minutes)	Picture Resolution (pixels)	Audio/Video Compression
LaserDisc	32 / 64[1]	71.7[1]	60 / 120	415 x 480	None
Video CD	0.65	1.2	74	352 x 240	MPEG/MPEG-1
DVD	4.7	3.5-11 (avg-max)	133 (typical)	720 x 480	AC-3/MPEG-2
HD-DVD (future)	18	3-11 (avg-max)	Video format dependent	HDTV format	AC-3/MPEG-2

Notes:
[1] LaserDisc is an analog storage media. Digital capacity and data rate are estimated equivalents.

photography allows images to be processed in ways that film can never hope to do, and do it with greater speed and ease. There are consumer products and tools for digitally capturing, processing, printing, and displaying photographs, whether just for making snapshots or for more serious entertainment, photojournalism, and desktop publishing activities. Let us begin by exploring the selection of equipment available. Figure 9.5 depicts how photographs can be digitally captured, processed, and viewed. For those still using conventional cameras, or having a backlog of film-based images, scanners can digitize negatives, slides, and photographs for storage and processing. Digital cameras directly digitize images (how they do this will be examined in Section 9.8) and provide the

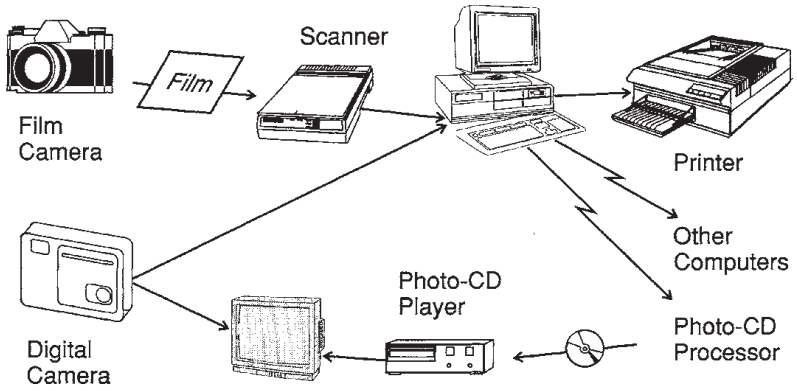


Figure 9.5 Tools for digital photography.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

quickest means for getting them into a computer. The computer is a vital element of digital photography. Image processing software allows digital photographs to be manipulated easily and edited as needed, included in documents, or simply to be printed. Digital photography increases the number of options for viewing photographs. Printers attached to personal computers can produce color prints that range in quality from good to excellent, roughly in proportion to the cost of the printer. But there are other options, one being that in the future, local photofinishers can easily go online, allowing them to receive digital pictures from customers and make high-quality prints to be picked up at the customer's convenience. Another choice is to display, not print, pictures. Photographers can display digital pictures on their computer monitor or transmit them on a network to other personal computer users. Some digital cameras have built-in LCD displays and, besides sending images to computers, where they can be displayed and manipulated, provide connections for displaying images on television receivers or recording them on VCRs. Yet another option is to send digital images to a photofinisher and have them recorded on a Photo CD™, a technique that has been available for some time. Photo CD technology will be examined in Section 9.9.

Despite all the advantages of digital imagery, which photographic and electronics companies are eager to exploit, technical problems have kept digital photography stalled in low gear for many years. The problem with digital images—we have said it before—is they require immense numbers of bits to represent them. Today, with advances in image sensors, storage technology, digital electronics, and data compression, digital photography is becoming affordable for professional photographers, small-office entrepreneurs, and well-heeled consumers.

9.8 Digital Cameras

Electronic still-video cameras were first introduced to the consumer marketplace in 1981 [With92]. These cameras operated like video camcorders but were designed to capture one frame at a time. They were expensive and, because they borrowed analog technology from video camcorders, their images were no better than television quality and could not be digitally processed. Today's totally digital electronic cameras are still expensive, but the image quality has improved greatly. All models, high-end and simple digital cameras, alike, offer much more function. The latest versions of digital cameras fall into three categories: There are professional digital cameras that seek to duplicate the image quality, look, feel, and operation of high-end, single-lens-reflex 35-mm film cameras. They are intended and priced for serious commercial photographers. The middle ground is held by far less costly semiprofessional digital cameras that offer many professional camera functions but with less image resolution. These cameras have proven attractive to small businesses such as car dealers, real-estate agents, insurance agents, and others who need small photos for advertising and record-keeping purposes. They are also attractive to large organizations, who can sometimes offset

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

the cost of digital cameras by film savings alone, such as insurance companies that have millions of digital photos as part of their underwriting and claims systems. At the low end, the "snapshot" digital cameras intended for amateurs, betraying their technical heritage, look, feel, and function like video camcorders, from which much is borrowed. Although the image resolution is (still too) limited, they include features that address the consumer market such as built-in liquid-crystal display (LCD) screens for image composition and review, NTSC outputs to television receivers, VCRs, and video color printers, and digital outputs to home computers.

There is still a ways to go before digital cameras can hope to match the image quality produced by film cameras. Table 9.4 provides a comparison of the resolution limits for 35-mm film and typical digital cameras marketed in 1996. Although inexpensive 35-mm cameras assuredly do not provide the full resolution of 35-mm film, they do set a high standard that only the most expensive digital cameras have yet to approach. However, ultrahigh resolution is not very important for amateur snapshots or even commercial applications where only small-size prints are needed. For these applications, digital camera manufacturers can point to features such as the ability to instantly view pictures, to digitally manipulate images, whether for making exposure adjustments, cropping, or some-

Table 9.4 Digital imaging specifications.

Still-Image Products[1] (Typical)	Image Resolution (megapixels)	Image Color Representation (3 x bits / pixel)	Image Size (Uncompressed)	Storage for 100 Images (Compressed 15:1)
35-mm film - max. resolution	18M = 5320 x 3550	36 bit	85 MB	567 MB
35-mm film - hi-resol. scanning	6M = 3072 x 2048	36 bit	28 MB	189 MB
Professional digital camera	1.5M = 1536 x 1024	36 bit	7 MB	47 MB
Semiprofessional digital camera	0.375M = 756 x 512	24 bit	1.2 MB	7.7 MB
"Snapshot" digital camera	0.125M = 512 x 256	24 bit	0.4 MB	2.6 MB

Notes:
[1] For descriptions of 1995-1996 products see [Anto95, Busc95, Scha95, Port96]. Prices range from less than $700 for snapshot digital cameras to $30,000 for professional models.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

thing more sophisticated, and to electronically transmit pictures and incorporate them in printed documents.

The block diagram for a digital camera is shown in Figure 9.6. Operating under microprocessor control, images are captured by a CCD (charge-coupled device) image sensor, processed, and converted to digital format. The digital image is temporarily stored in the frame buffer while it is being processed. The digital image processing block performs several functions: It first compresses a newly captured image and sends it to the on-board image storage. Later, under user control, it retrieves compressed images from image storage and decompresses them. The reconstructed images can be routed to computers via the digital output function and (depending on the camera design) to the camera's LCD screen, a television receiver, or a VCR via the analog output function.

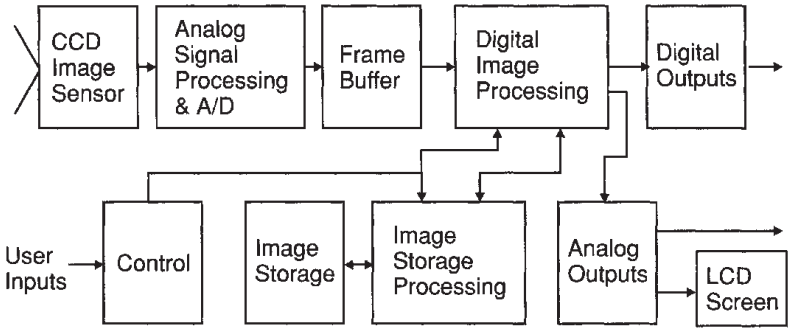


Figure 9.6 Digital camera block diagram.

Data compression provided the breakthrough needed for making totally digital electronic cameras feasible because, for portability, the camera must store up to 100 images. Reducing the amount of on-board storage is crucial, in that storage capacity largely determines how bulky and power hungry—and expensive—the camera will be. (The amount of on-board storage also contributes to setting limits on the allowable resolution, but this is largely controlled by the size and expense of the CCD array.) Compressing huge image files into smaller and smaller storage space allows using compact magnetic or solid-state storage media, either removable or fixed within the camera body. Today's high-end digital cameras use removable, large-capacity PCMCIA-format (credit-card-size) hard disk cards or flash memory cards. (See Section 6.6.) Less pricy models must make do with smaller amounts of nonremovable flash memory with removable, compact-format flash memory cards being an extra-cost option.

JPEG is now an accepted compression method for digital cameras. Simpler compression algorithms were used in early digital cameras—not because JPEG was unknown or unavailable—but for a quite different reason [With92]. One constraint in digital cameras is the size and power consumption of the electronics, which must be battery powered. Therefore, the adoption of JPEG was delayed

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

until small-size, power-efficient integrated circuits for its implementation became available. Compression ratios are typically 15:1 (or less) as indicated in Table 9.4.

Digital cameras are still-evolving products, sitting on the leading edges of several technology curves. Their success is most assuredly linked to higher-resolution and lower-cost CCD arrays, more compact and higher capacity storage, increased VLSI processing speed with lower power consumption, and better data compression. Marketplace demands for increased resolution at lower cost will continue to drive digital camera design. Whereas advances in CCD array technology define the front-end (image capture) function and cost, data compression essentially defines the back-end (image storage) function and cost. JPEG may now be the best possible image compression method, but one should not rule out other techniques. Being a closed process (see Section 3.4), there is no real requirement that standardized data compression be used in digital cameras. If a method can be found that will squeeze high-quality images into less storage space with affordable amounts of signal processing, it likely will be considered for digital cameras. (However, the adoption of a standard for sending compressed pictures between various digital photographic devices would change the situation.)

9.9 Photo CD

Although digital cameras may be priced out of the reach of amateur photographers, with the Photo CD system everyone can experience the advantages of electronic imaging using their existing 35-mm cameras. The Photo CD system was jointly developed by Kodak™ and Philips. It allows photographers to simply take an undeveloped or developed roll of 35-mm film to a photofinisher who, by using a special imaging workstation, can transfer their photos in digital form onto a write-once CD-ROM. Each disc can hold over 100 images and may be updated, so new photos can be added to a partially filled disc. In 1992, Kodak began marketing consumer and professional Photo CD products that allow stored digital images to be displayed on television receivers or computer monitors or to be printed by photofinishers. Consumers can load the discs into a specially modified CD player and view high-quality images on conventional television receivers. The consumer Photo CD player might be thought of as a digital replacement for the slide projector—one that allows viewers to sequence through the pictures in any order, to do fades, dissolves, and wipes, and to rotate, enlarge, and crop images during viewing. Whether a technology ahead of its time or whether consumers (who like the author) have relegated their 35-mm slides and slide projectors to the storage closet in favor of prints, the Photo CD player has not yet captured much attention in the consumer market.

Other uses for Photo CD are doing much better. Many consumers and professionals are into desktop publishing. Using a personal computer equipped with a CD-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

ROM XA[4] compatible drive, and Photo CD software, which is now a part of even modestly priced desktop publishing packages, they can view their photographs and include them in documents. Another option, called Photo CD Portfolio II, which has a more professional orientation, allows 35-mm photographs to be combined with audio, text, and computer-generated graphics on a single disc [ElTi92, Seyb95A]. Whether to liven up a slide presentation, develop a multilanguage tutorial, or produce photo-filled publications, authors can create high-quality audio/visual discs, either by sending their material to a professional lab or by using a disk writer attached to their computer. The resulting disc may be played on personal computers, the Kodak Photo CD player, or CD-based multimedia gear such as the Philips CD-I™ player.[5] Prints may also be made from the images stored on disc. In addition, professional versions of the Photo CD support applications such as storing images captured on 4 × 5-inch film and storing X-ray images captured by medical imaging modalities.

Let us explore the role data compression plays in the consumer Photo CD system. To capture 35-mm images, the Photo CD process begins by scanning each image at 3072 × 2048 pixels with 24-bit color. This provides the high-quality digital images needed for printing; it also produces a lot of data—nearly 19 MB for each image! If uncompressed, only 30-some images could be stored on a CD, far short of what a typical slide projector tray contains. Kodak and Philips engineers may or may not have had this comparison in mind when they designed the Photo CD system, but write-once optical discs are quite expensive, and packing more images on each disc makes the concept more attractive. Also, reading an uncompressed image from a CD would be a slow process—requiring about 30 seconds even with a 4X-speed CD-ROM player. Consumers would surely grow impatient while waiting for their favorite vacation slide to be painted on the television screen, and desktop computer users trying to scan through picture libraries could not tolerate this rate. Then, too, the resolution of conventional television receivers and computer displays is far lower than the Photo CD scanning resolution, so much of this data is not needed for display purposes. Obviously (we say with confidence), data compression is the answer, but what type?

The solution chosen for Photo CD is a multiresolution technology that loads the disc with an Image Pac™ file containing compressed images stored at *five* different resolutions.[6] This range of image sizes is intended to make the Photo CD system more useful to consumers and professional users alike. As Table

[4] CD-ROM XA is an extended CD-ROM format that provides the ability to interleave sound, images, and text on the same disc.

[5] The Philips CD-I player and other interactive multimedia set-top boxes are discussed in Section 9.10.

[6] A sixth format, scanned at 6144 × 4096 pixels with 24-bit color, and not included on the consumer Photo CD disc, is available for professional applications using 4 × 5-inch or other large-format film [ElTi92, Koda94].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

9.5 shows, each image resolution is intended for a different application and, interestingly, differing compression techniques are used to support these applications. Photo CD image coding and compression operate as follows: To code an image, the Photo CD process always begins with a positive image (i.e., negatives are converted to positive images). Through a series of transformations, the digital image is represented in a special PhotoYCC™ (luminance, chrominance, chrominance) color component format. This format differs from standard digital television YC_BC_R coding in that it uses primary RGB color values which preserve the extended dynamic range of film images, supporting both printing and video display [Rich91, Koda94].

Table 9.5 Photo CD image resolutions.

Image Component	Image Resolution	Application
Base/16	192 x 128	Thumbnail index images
Base/4	384 x 256	¼ SDTV
Base	768 x 512	SDTV
4Base	1536 x 1024	HDTV, Small prints
16Base	3072 x 2048	Hardcopy prints (≤ 16-in. x 20-in.)

Compression for the three lowest-resolution images consists of no more than subsampling the chrominance components by a factor of 2 in each direction. This technique, which is commonly used in image and video compression, provides a 2:1 compression with little, if any, visible loss of image quality. This method was chosen because the three lower-resolution images represent only a small fraction of the total data and because it allows these images to be quickly made available for viewing. For example, consumers who view photographs on television receivers or desktop publishers who use tools such as small thumbnail images for scanning image libraries can do so with simple hardware or software and without waiting for images to be decompressed.

To fit over 100 images on a single disc, all five different resolutions must fit into about 6 MB of storage. By itself, chrominance subsampling cannot sufficiently compress the two higher-resolution images to achieve this goal.[7] This means these images must be further compressed, and, to do so, a pyramid coding technique is used. Because this coding method has not been encountered previously, let us take a few moments to understand what it does.

[7] A detail of the Photo CD process is that the 16Base image is chrominance subsampled, but the 4Base image is not [Koda94].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

When a set of successively smaller images is created by "downsampling" (low-pass filtering and subsampling) the preceding larger image in the set and these images are conceptually stacked one atop the other, they form a pyramid like the one shown in Figure 9.7. This method is also known as hierarchical coding (the stack forms a hierarchy of images) because the image is coded in a way that allows it to be used at different quality levels or resolutions. It is also a form of progressive coding, yet another term used to describe this technique [Penn93]. A progressive encoder is one that successively scans an image with each scan producing finer and finer detail. This allows the image to be progressively transmitted to a receiver in stages; at first, a low-resolution image is sent, followed by more detailed information that allows higher and higher resolution images to be reconstructed [Rabb91]. This approach is useful for operating across a slow network, where users can work online with a lower-resolution version of an image, downloading a higher-resolution version only when necessary. The same is true for Photo CD users who must retrieve images from relatively slow CD-ROM drives.

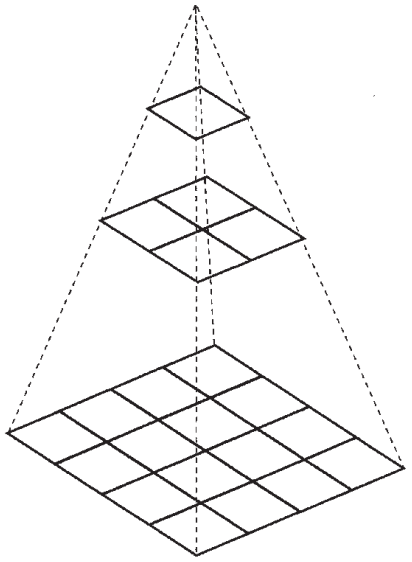


Figure 9.7 Pyramidal multiresolution coding.

The pyramid coding process begins with the original image that is low-pass filtered and subsampled, usually by a factor of 2 in each direction. This creates the first reduced-resolution image. Then that image is downsampled to produce the next image. This continues until the lowest-resolution image in the pyramid is formed. The actual compression process begins with the lowest resolution image that is upsampled (scaled up) to the next higher resolution and subtracted from the next higher resolution image to get a difference image. The difference image is then encoded (compressed) using an appropriate image compression method (such as JPEG's DCT, quantization, and entropy coding techniques).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

This operation continues until all higher-resolution images in the pyramid have been coded.

As previously stated, pyramid coding is used for storing only the two highest-resolution Photo CD images. These images contain the bulk of the total data, but, because (today) they are not used for television display, they do not have the same "real-time" decoding requirements as do lower-resolution images. Therefore, more complexity and longer decoding times are acceptable. Reconstructing the 1536 × 1024-pixel 4Base image involves the following: The compressed 4Base difference image is retrieved from storage and decompressed. The 768 × 512-pixel Base image is retrieved and upsampled to 1536 × 1024 pixels. These two images are added, pixel by pixel, to obtain the full 1536 × 1024-pixel image. Essentially the same steps are followed to reconstruct the 3072 × 2048-pixel image, using both the 4Base and 16Base difference images.

As to the exact compression method used for storing the difference images, it is not JPEG. At the time the Photo CD system was being developed, the JPEG specification was still being drafted. Kodak engineers chose, instead, to develop an algorithm based on quantization and variable length Huffman coding. Being proprietary, few details are publicly available [Rich91, Koda94].[8]

9.10 Interactive Multimedia Systems

Beginning in the mid-1980s and continuing into the 1990s, a multitude of intelligent set-top boxes have been introduced, or announced, all aimed at building interactive multimedia systems around television sets. These products are primarily intended for home entertainment and education but are found in schools and businesses too. Video games (from Nintendo™, Sega™, and others) are the leading consumer applications, making up some 90% of the market for interactive, multimedia set-top boxes. The remainder of the market encompasses Photo CD players, karaoke machines, and high-end multifunction boxes. In-home uses include interactive entertainment such as movies and music videos, reference material such as encyclopedias, consumer data and transactions, educational software, and more. School and business uses include education, training, interact-

[8] However, JPEG is the compression method chosen for the FlashPix™ digital photo file format recently developed by Kodak [Cost96, Koda96]. Whereas Photo CD is intended for storing and archiving digital images, FlashPix is designed for sharing them on computers and across networks. It allows images to be stored at multiple resolutions, similar to Photo CD, and in either RGB or Photo YCC color component format. A unique feature of FlashPix is that the original image is divided into tiles, each containing 64 × 64 pixels. For lower resolution images, adjacent tiles are combined, with the smallest image consisting of only one 64 × 64 pixel tile. JPEG compression is an option. Each tile is compressed independently, which does not result in the most efficient compression, but has the advantage that individual tiles may be decompressed to access, display, or print part of the image.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

ive kiosks, and other business-oriented multimedia-based applications. Multimedia data types include text, speech, audio, graphics, animation, still images, and motion-video.

Today's consumer multimedia systems are stand-alone players that range in price from less than $100 to somewhat more than $500. These price points—for products that are essentially computers without keyboards, displays, and hard files—are far lower than any multimedia home computer.[9] Low prices are achieved by using a television set for video and still-image display and for audio playback, and by using low-cost storage media. Some consumer multimedia systems store information in ROM cartridges, but, increasingly, the chosen storage media is CD-ROM, and future products will likely use DVD. These systems are interactive, allowing the viewer to control what is seen and heard with simple hand-held controllers for games or TV-like remote controls for higher-function players. Future interactive multimedia set-top boxes are likely to go online; at least that is what cable television, telco networks, and others (see Chapter 8) are planning as they eagerly eye the opportunities for interactive multimedia.

The marketplace reception for these products is decidedly mixed. Millions of videogame machines have been sold, whereas other products that can do more than play games face stiff competition and have not done as well. Today, the territory staked out by high-end stand-alone multimedia players has largely been overrun by multimedia PCs which, despite the difference in price, both consumers and businesses are turning to for multimedia entertainment and education applications. Part of the reason is many advanced interactive multimedia applications really require a keyboard, not a TV remote, to control them. Then, too, PCs are increasingly becoming capable multimedia platforms. Although no one will mistake a multimedia PC for an arcade-quality videogame machine, PCs are getting much better at running games, and they run almost all the applications for which high-end stand-alone multimedia players were created. Other advantages are that PCs allow more creative uses of multimedia and PCs can be connected to online information services to find more of it. Another factor favoring PCs is that a far broader range of multimedia applications is available for either IBM-compatible or Apple computers than for any one high-end stand-alone multimedia player. Largely incompatible products, these stand-alone multimedia players observe no interchange standards (except audio CDs, and perhaps Photo CDs, but definitely not the contents of CD-ROMs). This places stand-alone multimedia players at a disadvantage because software developers prefer "open" architecture PCs, of which there are only two targets, to proprietary architecture stand-alone multimedia players, of which there are many. Software developers may be easily enticed to videogame machines, which have huge distributions, but not to limited-distribution, high-end stand-alone multimedia players.

[9] The price of PCs, although still much greater than $500, is dropping. The $500 PC continues to be a target of the computer industry, but that is another (long) story.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Historically, interactive multimedia set-top boxes have been trendsetters, being the first to introduce new, innovative multimedia technologies for consumer and business applications. These products use high-performance 16-, 32-, and 64-bit VLSI including RISC processors, DSPs, 3-D graphics accelerators, and data decompressors. They were first with leading-edge features such as video on CD-ROMs and 3-D graphics—technologies that later migrated to multimedia PCs. Figure 9.8 shows the components to be found in an interactive multimedia set-top box. This hypothetical high-end system is organized around a shared bus and an RISC processor executing a real-time operating system that controls the operation of several specialized coprocessors and dedicated functional units. Lesser set-top products will not include all these components, but a challenge for all interactive multimedia set-top boxes, beginning with the very first and continuing to the present generation, is to include enough high-performance audiovisual elements to attract consumers and, yet, maintain a reasonable price.[10] An essential part of this effort is the processing and decompression of audio, image, and, particularly, video data.

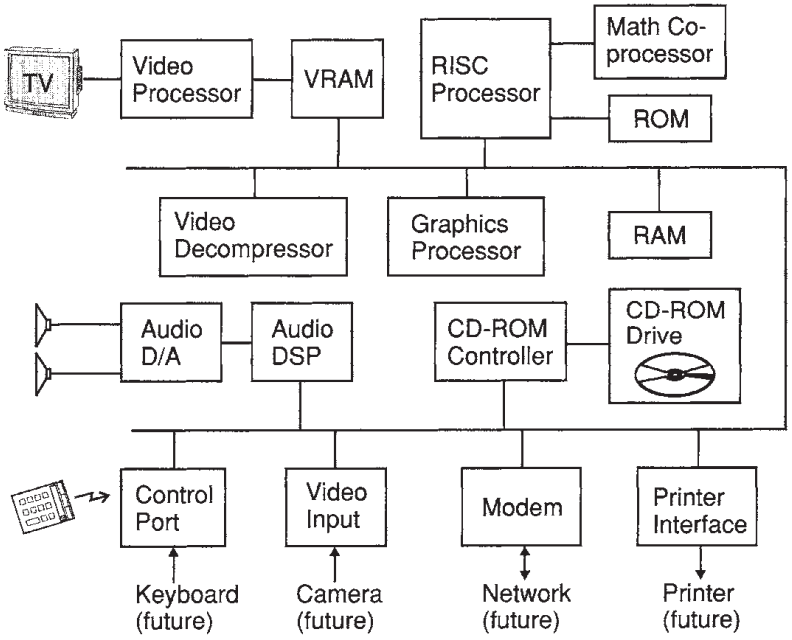


Figure 9.8 Interactive multimedia system.

For audio, the major challenge existing interactive multimedia players face is providing different levels of quality for various applications. For instance, the Philips CD-I (Compact Disc -Interactive) player, an early high-end entry in this

[10] A recent trend is to add keyboards, modems, and printer interfaces to interactive multimedia set-top boxes, equipping them to be Internet browsers [Gros95].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

field, offers four audio quality levels ranging from 16-bit PCM-coded CD-quality stereo to 4-bit ADPCM-coded speech-quality audio [Luth91, Mcco94A]. Future interactive multimedia products will be called on to handle Dolby AC-3 format compressed audio and, for game-oriented applications, 3-D interactive audio [Yosh95A]. A DSP (digital signal processor) is an economical solution for processing audio because it can be programmed for differing coding formats and decompression techniques.

Still images are relatively easy to deal with—that is, in comparison to video. Nowadays, most high-end set-top boxes have the bandwidth and processing power needed for video—and, therefore, for images too. The major challenge that images present, beyond handling differing image-encoding formats, is fetching them from the CD-ROM and displaying them quickly. The still images that interactive multimedia players must handle do not contain all that much data, even when stored in uncompressed format. Images containing about 720×480 pixels at 12 bits per pixel, or less, are all that is needed for television display. Therefore, they may be retrieved and moved through the system relatively quickly. Often the performance issue is not the CD-ROM speed, or the system bandwidth, but, rather, the processing time for decompressing compressed images. There are several solutions: One is to just avoid the problem and store uncompressed images, as does the Photo CD system described in Section 9.9. For approaches that begin with compressed images, the question is what format to use and how to quickly decompress an image. Early interactive multimedia players, such as the Philips CD-I announced in 1985, chose to store still images in (proprietary) formats that could be decompressed by the on-board video decompression hardware running in single-frame mode [Meer92]. Now, there are other choices. One is to use JPEG compression and a dedicated JPEG coprocessor, although, JPEG has yet to establish itself in this market. A more economical solution is to do decompression (JPEG or otherwise) in software running on the RISC processor; today's RISC processors are powerful enough to make software decompression acceptable for many image-based applications.

Video was and still is the most demanding of all digital multimedia data types. The first generation of interactive multimedia set-top boxes, designed in the mid-1980s when digital video for consumer electronics was definitely leading-edge (some wags would say bleeding-edge) technology, had limited video capability. Designers were constrained by CD-ROM storage capacity limitations, by the cost (and availability) of decompression hardware, and by not having effective decompression algorithms that would run on this hardware. Increasing CD-ROM storage capacity was not an option. So, instead, they concentrated on data compression techniques that would squeeze video into ever-smaller amounts of storage and that would allow it to be decompressed with the available hardware. What resulted was a bevy of proprietary compression algorithms that, to put it charitably, produced video of less than broadcast quality. Partial-screen video, low-to-medium resolution, limited color depth, and substandard frame rates were the norms. Consumers were not impressed.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The second generation of interactive multimedia products, at least second generation in video capability, began to take shape in the in the 1991-1993 time frame. An effort led by Philips and JVC produced the Video CD standard, initially for karaoke systems that show moving pictures and lyrics, and since adopted for a variety of set-top boxes and multimedia PCs. (See the Video CD discussion in Section 9.6.) The Video CD uses MPEG-1 compression, and, thanks to advances in VLSI, single-chip decoders are available and affordable to interactive multimedia set-top boxes. The result is these products deliver full-screen, medium-resolution (352 × 240 pixels) video with full color depth at 30 fps.

The next generation of interactive multimedia set-top boxes, still on the drawing boards but likely to reach the marketplace by 1997, can be expected to offer even higher-quality video. Industry observers expect these products to be based on the DVD[11] [Krau95B]. This means increased storage capacity will be available for all types of multimedia, video included. Also, by leveraging the low-cost MPEG-2 decompression chips that DVD players will use, interactive multimedia set-top boxes will deliver full-screen, high-resolution video (up to 720 × 480 pixels) with full color depth at 30 fps.

9.11 Multimedia PCs

Nearly all home personal computers and many intended for businesses carry the Multimedia PC (MPC) trademark. This means they include CD-ROM drives, sound cards, speakers, microphones, data compression features, and more.[12] Multimedia PCs are all-purpose tools for entertainment, education, and business. Not surprisingly, in 1995 they accounted for nearly 40% of sales by large electronics retailers, and by 1997 nearly 50% of U.S. homes are expected to have at least one PC [Mcwi95]. In Chapter 2, we began to learn of the problems that all computers face in delivering interactive multimedia, and in Chapter 6, we learned of the progress that mobile and desktop computer designers have made in addressing this challenge; that is, by surrounding the basic PC with the trappings of multimedia, and by improving the internals—adding storage capacity, bandwidth, CPU power, VLSI for graphics and compression, and better software for real-time processing—and by improving the externals—higher-resolution monitors and connections to higher bandwidth networks—PCs are becoming creditable multimedia machines.

Turning the humble PC into a bold multimedia warrior has not been easy. Finding the right strategy for handling compressed multimedia data has proven

[11] One observation is that essentially the same VLSI components are needed to build advanced game-playing systems and DVD players.

[12] The MPC trademark is administered by the Multimedia PC Marketing Council, an industry group that maintains an ever-changing standard which defines minimum requirements for multimedia PCs [MBR95].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

a most difficult task, a struggle that is still in progress. When multimedia PCs handle speech, audio, image, and video data, they face all the same limitations encountered by interactive multimedia set-top boxes (storage capacity, processing speed, and bandwidth)—and more! The reason is PCs must deal with a far wider range of multimedia applications including the following:

- Interactive multimedia playback from CD-ROM
- Multimedia serving and interactive applications from hard disk storage
- Networked multimedia applications
- Desktop multimedia production and authoring applications
- Desktop videoconferencing
- Analog television audio and video capture, digitization, and display

In fact, some people both outside the computer industry and insiders (who should be more realistic) expect PCs to do no less than flawlessly run every multimedia application that anyone could program. This blissful state of affairs has not been reached yet, in part, because effective, universal data compression technology still eludes PC system designers.

Our study of data compression in multimedia computers will focus, as has the PC industry, on video compression. Handling speech and audio is straightforward, with the electronics reduced to a single chip on the motherboard of most PCs [Wils95E]. However, as will soon be evident, coordinating audio and video is a challenge. Also, compressed multichannel audio has yet to appear in PCs, so there still may be some lessons to be learned. As to images, handling images has scarcely raised a blip on any PC designer's radar screen; software has proven adequate. PCs have sufficient processing power to manipulate images stored in JPEG, Photo CD, or any other miscellaneous image format, and images have no real-time processing requirements. Although, today's image processing activities mostly involve decompression, display, and simple editing tasks (cropping, scaling, and rotation), so the requirements may change if future applications demand more complex image processing. Handling video (with synchronized audio) is not so easy.

The problem with video on PCs, other than the high data rates, is no one representation of digital video or no one compression method is right for all applications. Over the years, the PC industry has produced a gaggle of video compression algorithms, mostly executed in software, mostly proprietary, and primarily intended for video playback. In 1995, with the adoption of the MPC Level 3 specification, MPEG-1 decompression became a standard feature for multimedia computers [MBR95]. Most observers concede that for full-screen, full-rate video playback MPEG-1 produces unquestionably better results than any of the proprietary algorithms.[13] However, when MPEG-1 encodes data in

[13] The accepted standard for full-screen, full-rate PC video is 640 × 480 pixels at 30 fps.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

its most compact format, it is essentially unusable for multimedia authoring applications where frame-by-frame access is required. For videoconferencing, even with full hardware implementation, none of the video playback algorithms—the proprietary algorithms or MPEG-1—are ideal. Therefore, multiple different PC video compression (and decompression) algorithms are needed to cover the range of PC video applications.

Video requirements for multimedia PCs

Which video compression algorithms are right for multimedia PCs and what makes them right? Video coders and decoders (codecs) are traditionally judged by the image quality they produce and how effectively they compress video to meet storage and bandwidth limitations. Video codecs for multimedia PCs must also be measured by the resources needed for compressing and decompressing video data, by their suitability for the intended applications, and by their support for PC graphics displays. All these criteria interact, and, often, the term "scalability" is used to describe the merits of a video codec. Codecs that can cover a wide range in any one or more of these criteria are said to be scalable. Let us examine more closely what scalability really means for multimedia PC video compression.

The first element is computation load. Any compression or decompression algorithm can be run in software—in theory. Practically speaking, some algorithms do require hardware, but discount that for the moment. Successful software video compression and decompression algorithms are those that produce quality images and leave resources available to run the application(s). They do not consume too many CPU cycles, nor do they consume too much memory, and they produce compact data representations that PC displays can handle. Software-only algorithms that rely exclusively on the PC's CPU to do video decompression must be designed to avoid its limitations: To accommodate base-level PCs, integer arithmetic is favored over floating-point arithmetic. To simplify and speed up the decoding task, because CPUs are better at manipulating data in a multiplicity of bytes, a byte-oriented video stream syntax is used by many software-only algorithms. Unfortunately, it results in loss of storage efficiency when compared to the typical compressed bit stream definitions employed for communications-oriented video applications, because it prohibits applying lossless variable-length coding at the tail end.

To date, software-only algorithms have proven nearly useless for broadcast-quality work. They do not offer real-time encoding, thus excluding video production applications. They do offer low-complexity, scalable video playback, but often at the expense of image quality. Reduced image spatial resolution, aggressive color subsampling, reduced pixel depth, and reduced frame rates or dropped frames are compromises made by software-only PC video (de)compression algorithms in the name of reduced computation work load. When these trade-offs are not acceptable, compression hardware, hardware assists for software-only algorithms, or better algorithms are required.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

That brings us to the second element, the applications. For multimedia playback, whether done in software or hardware, ease of decoding is the most important issue, as a multimedia CD-ROM that can be played on low-cost PCs without expensive video features has the greatest sales potential. For other applications, those where both encoding and decoding must be done on the multimedia PC, different codec criteria apply. In multimedia production and authoring, where editing is required, it is important that the coder produce a data stream that supports frame-by-frame access. This means interframe coding, which can produce the highest compression ratios, is not a wise choice. For videoconferencing, as learned in Chapter 5, it is important that the encoding and decoding time be symmetrical and the time delays for either be low and well controlled. Techniques that can be executed in real time, such as those used by the H.261 algorithm described in Chapter 5, are needed. Table 9.6 summarizes application requirements for PC video codecs.

The third element is support for PC graphics displays. PCs are blessed or cursed with tremendous display flexibility, perhaps similar to future HDTV

Table 9.6 Application requirements for multimedia PC video codecs.

Video Codec Characteristics	Multimedia Playback Applications	Video Production and Authoring Applications	Videoconferencing Applications
Scalable image quality	Trade-off for software-only playback	Trade-off for frame-by-frame access (I-frames only)	Trade-off for communications bandwidth
Scalable compression ratio	Trade-off for minimum storage size or data rate Trade-off for software-only playback	Trade-off for frame-by-frame access	Higher is better
Low memory consumption	Important for low-cost, software-only playback	Important, but high-end PCs are used for these applications	Important for low-cost, hardware-assisted codecs
Simple decode (playback speed)	Software-only decode for low-cost applications	Desirable	Hardware
Simple encode (asymmetry)	—	Desirable but not required	Hardware
Editability (frame by frame)	Some applications require frame-by-frame playback	Required	—
Low delay (real-time video)	—	—	Required

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

receivers. In conventional television receivers, a single color representation scheme, a single spatial resolution, and a fixed frame rate apply, but all these parameters can vary for PC displays and all interact with PC video compression. PC displays use an RGB color system, different from YC_BC_R or any broadcast video representation. Furthermore, there are various RGB formats; the highest-quality representations use 24 bits for each pixel, but 16-bit, 8-bit, and even 4-bit color is available to PC applications. It should come as no surprise that the software-only algorithms created by the PC industry tap this capability as a simple form of video compression [Luth91]. (Some would say data reduction, but we will call it compression.)

PC displays are driven with 24-bit RGB color. So, as Figure 9.9 shows, color format conversion is necessary when the video compression algorithm uses another encoding. The simplest conversion is upscaling 16-bit RGB color, restoring the low-order bits truncated during the compression process. (Typical 16-bit RGB color formats encode each color component in 5 bits, leaving 1

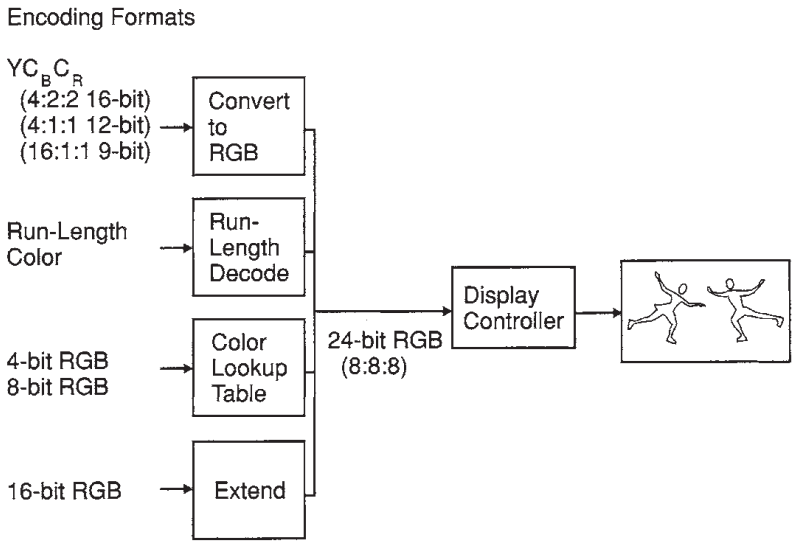


Figure 9.9 PC color format conversion.

bit unused.) Although 16-bit color produces realistic images, it affords little compression; consequently, some algorithms use 8-bit or even 4-bit color. These encodings are converted by accessing a color-lookup table that contains the 24-bit equivalent for each 8-bit or 4-bit pixel value. The color-lookup table is created during encoding, a process which can grow quite complex if the color-lookup table is dynamically changed to track color changes. Neither 8-bit nor 4-bit color can deal with color shading in complex scenes; so, these formats are best reserved for simple computer-generated images and animations where only a few colors are needed. The same is true for run-length color, the third encoding format shown in Figure 9.9. The idea here is to compactly represent runs of like-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

colored pixels, again, an idea that works well for computer-generated images and animations where large areas contain the same color. For complex images, YC_BC_R or some other form of luminance-chrominance representation provides greater compression because the chrominance components can be subsampled. The advantage of a luminance-chrominance format is, besides producing better color quality in full-motion images, that it uses less bandwidth and memory than any of the RGB schemes. Some software-only PC compression algorithms aggressively subsample the chrominance components to drive down bandwidth and memory to the lowest possible values. Using 4×4 blocks of pixels, they create, in effect, a 9-bits-per-pixel representation. Other PC video compression algorithms conform to less drastic subsampling strategies, the ones found in communications applications. In any event, video encoded in a luminance-chrominance format usually is not converted into RGB format until needed for display. The conversion process proves to be straightforward, using well-known color-space transformations [Penn93] and is implemented in modern PC display controllers [Cole94].

Another aspect of displaying video on PCs is scaling the image size. One might argue this has nothing to do with video compression, but there is a close connection, and some compression algorithms are better able to support image-size scaling than others. Algorithms that provide data at multiple resolutions, somewhat like the multiresolution technique used by the Photo CD system (although its TV-displayable images are not compressed), make image-size scaling easier. However, practically speaking, no compression algorithm could provide all the image sizes needed for PCs, so display scaling techniques still must be applied.

Here is what must be done to scale image size: Figure 9.10 shows a decoded but unscaled MPEG-1 video image as it might appear on a VGA-resolution or

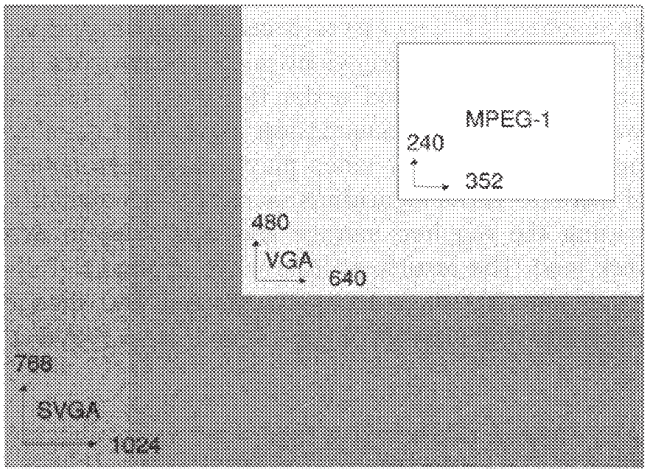


Figure 9.10 PC display screen.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

SVGA-resolution PC screen.[14] The MPEG-1 video window may be positioned anywhere on the screen. The unscaled window occupies about one-fourth of the VGA-resolution screen but would occupy a much smaller fraction of the SVGA-resolution screen. To control the window size, making it either larger or smaller, each pixel in the MPEG-1 image must be scaled. Making the window smaller is the least difficult. To do so, rows and columns of pixels within the MPEG-1 image may simply be dropped (decimated); or, for better image quality, each pixel in the smaller image may be created by averaging neighboring pixels in the original image. Making the window larger requires more care because the results will be more visible. One might simply replicate pixels in two dimensions, but the scaled-up image is likely to be blocky, exhibiting a type of artifact known as pixellation. Pixel interpolation provides a better solution in that each pixel in the scaled-up image is created by applying a color smoothing or dithering algorithm to pixels in the original MPEG-1 image [Netr95].

PC video software compression

For many years, highly influential hardware and software forces within the PC industry have promoted the idea that video compression should be run in software, just as are other processing tasks. Advocates for software video compression cite it as the only sensible choice for an era when the algorithms and the applications themselves continue to evolve, and no standard algorithm(s) exists that can be safely committed to hardware. Software video compression, they argue, allows PC costs to be kept low and a variety of algorithms to be run in support of all video-oriented applications.

Historically, PC video compression software owes much to DVI™ (Digital Video Interactive), a pioneering technology effort that in 1986 demonstrated full-motion video running on PCs[15] [Luth91]. DVI technology included video software compression algorithms that ran on a programmable coprocessor. (Recall that the 1980s PC processors were way underpowered for video, or anything but text processing.) More robust PC processors became available in the early 1990s, which offered the prospect of running video without coprocessor assistance. The DVI-inspired algorithms reappeared under the Indeo™ name and were soon joined by a slew of new video compression algorithms developed by the PC industry, all intended primarily for video playback at reasonable speeds on the low-cost PCs of that era. These algorithms sacrificed video quality and compression (i.e., increasing the required storage and data rate) in favor of reduced computation work load. The results, at least when applied to full-motion video, were disappointing—small postage-stamp-size images, and often jerky, questionable quality color video with unsynchronized sound, or no sound at all. Typically,

[14] MPEG-1 is used to illustrate display scaling. Other algorithms have similar requirements.

[15] DVI was developed by RCA Laboratories, then transferred to General Electric, and subsequently sold to Intel.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

the best of this generation of software video algorithms delivered 320 × 240 resolution pictures at 15 fps on a dedicated Intel i486™, values that scale with processor speed [Wolf93A, Rodr94]. However, most applications using these algorithms were forced to make do with much lower resolutions and frame rates, being constrained by storage size and data rate and by being unable to dedicate all the processor resources to video decompression. Nevertheless, the novelty of seeing video on a PC screen had a tremendous impact on everyone. A whole generation of CD-ROM multimedia playback applications for PCs and game machines was built on these algorithms, applications where less than broadcast-quality video met with some degree of consumer acceptance.

Table 9.7 lists some of many PC software video algorithms created during this era. Although primarily applied to video playback, all these software algorithms also can compress video but do so very slowly on standard PCs. However,

Table 9.7 Examples of proprietary PC software video algorithms.

Codec Manufacturer	Compression Technique	Coding Methods	Data Rate Control Method
Captain Crunch™ Media Vision Inc.	Wavelet transform subband	Intraframe	—
Cell™ Sun Microsystems Inc.	Vector quantization	Intraframe Interframe	—
Cinepak™ SuperMac Technologies (acquired by Radius Inc.)	Vector quantization	Intraframe Interframe	Quality-level scaling
Fractal Video Pro™ Iterated Systems Inc.	Fractal transform	—	—
Indeo™ 4.0 Intel Corp.	Vector quantization	Intraframe Interframe	Frame dropping
Indeo 4.1 Video Interactive Intel Corp.	Wavelet transform subband	Intraframe Interframe	Quality-level scaling & frame dropping
TrueMotion-S™ Horizon Technologies and Duck Corp.	Non-DCT transform subband Arithmetic coding	Intraframe	Quality-level scaling
Video 1™ Microsoft Corp. (developed by Media Vision)	Vector quantization (motive)	Intraframe Interframe	—

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

essentially all manufacturers back them with hardware cards that provide real-time compression (and decompression) for applications such as authoring and videoconferencing. All these algorithms are considered proprietary by their manufacturers, so public information about their inner workings is limited (as blank entries in Table 9.7 suggest). Their external operating characteristics are well known, nevertheless, having been subjected to extensive field testing. As a class, many observers judge them to provide about the same overall picture quality, but there are measurable differences [Ozer94, Brya95, Ozer95A]. For example, some are better at handling complexities such as scene-to-scene transitions, and some achieve higher compression ratios (i.e., smaller storage sizes and lower data rates) than others. The picture quality of these software algorithms is often said to be much lower than what hardware-assisted codecs like MPEG can provide. But, in fairness, the observed results depend greatly on the PCs on which they run, and the best of these algorithms can deliver full-screen, 30-fps performance on modern PCs [Ozer95B].

The software environment for desktop multimedia application development is built upon tools such as Apple QuickTime™ and Microsoft Video For Windows™ [Hoff92, Beer93, Pere95]. These operating system extensions allow developers to perform operations such as video capture, editing, and compression and to create data files that contain integrated audio and video. They support many of the software algorithms listed in Table 9.7 and various adaptions of M-JPEG, MPEG, and other standard video compression algorithms available in software and hardware. This dizzying array of algorithms presents multimedia application developers with a problem; namely, which algorithm should they choose? Circa 1995, Indeo 4.0, Cinepak, and TrueMotion-S are ascribed to be among the most widely used [Somo95, Turl95], but other algorithms dominate application niches. For instance, fractal compression is the leader for CD-ROM encyclopedias. Some of these algorithms also have been used for videoconferencing (with hardware assistance). For a time, Intel promoted Indeo 4.0 as a standard for PC videoconferencing but, bowing to compatibility, in 1995 endorsed H.320, an international standard for videoconferencing. Any algorithm that uses only intraframe coding is a clear-cut choice for video editing. Interframe coding *may* provide more compression (and this depends on many factors), but intraframe-only coding allows frame-by-frame access. Also, sometimes intraframe-only video algorithms are applied to still-image applications.

By 1993, it was painfully evident that PC video compression was in disarray. With multimedia software vendor loyalties split among many choices, not one of the proprietary algorithms listed in Table 9.7 (or any other algorithm) could claim to be a clear industry leader. Then, too, the compromises in picture quality and data rate that all the software-only algorithms made to reduce the computation burden on base-level PCs were regrettably apparent. Application developers, and the PC industry too, knew that PC video needed to be improved—quickly! PC users were soon to be (or were being) exposed to other home entertainment

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

products that offered higher-quality digital video. It was simply a matter of time before they would walk away from multimedia PCs that delivered less.

What was needed was a "standard" PC video compression algorithm, one that would provide all PC video applications with high-quality video, one that would operate in software on low-cost PCs, not just in hardware on high-end systems. But what should that algorithm be; should it be a new algorithm, or if an existing video algorithm, which one? In the 1993 time frame, various PC industry groups, committees, and consortiums were linked with efforts (or rumors of efforts) to select a standard PC video compression algorithm [Coll93, Wolf93A]. Alas, all those efforts led nowhere, which is not unexpected because they faced an impossible task: No one vendor had the marketplace presence to convince others that its proprietary algorithm should be the standard; and, more telling, no one known algorithm, proprietary or standard, could handle all PC video applications.

Now move the clock forward to 1995 when, as stated earlier, MPEG-1 decompression was included in the MPC Level 3 specification for multimedia computers—in effect, creating the new "standard" PC video algorithm. How did that come about? First, MPEG-1 technology can play back full-screen, full-motion video with much clearer pictures and synchronized sound at low data rates and at a quality level that the proprietary algorithms are hard-pressed to match. The catch is that the DCT-based intraframe coding and sophisticated interframe coding needed to accomplish this are compute intensive, and encoding is even more difficult. Conventional wisdom says MPEG-1 is a hardware-only algorithm, not suitable for low-cost PCs, and is intended for broadcasting (or multimedia playback), not video production or authoring. Between 1993 and 1995, three events occurred that changed conventional wisdom: A new generation of more powerful Pentium™ and PowerPC™ processors hit the marketplace. Becoming the engines for essentially all PCs, they allow MPEG-1 encoded video to be played back at reasonable frame rates in software (but on 100% dedicated processors). Hardware MPEG-1 decoders became affordable for PCs, having been reduced to a single, low-cost chip for widespread deployment in DBS and other consumer applications. Software for MPEG-1 *encoding* (and decoding) was developed that, using only I-frames, allows frame-by-frame access for video editing. The turning point for MPEG-1 that pushed it over the top came in 1995 when Microsoft included MPEG-1 decompression software in Windows 95 and several PC manufacturers introduced MPEG-1 decompression hardware as a standard feature [Turl95].

So, MPEG-1 won out in the multimedia PC video compression standards battle and, finally, there is *a* standard algorithm, right? Not quite. As discussed in Section 7.8, desktop videoconferencing is still a technology in transition, and it remains to be decided what compression algorithm it will use. Reasons were suggested why H.320-compliant H.261 or H.324-compliant H.263 may be winners—namely, they are standards designed specifically for videoconferencing and do have market presence. One would not expect MPEG-1 to be used for videoconferencing (because it is an asymmetric algorithm not designed for low

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

delay time, and so on). Nevertheless, if most future PCs have built-in MPEG-1 decoding hardware, and with the cost for MPEG-1 encoding hardware dropping rapidly, someone will surely try using MPEG-1 for videconferencing. Stranger things have happened!

Then, too, we must look forward and consider that multimedia PCs will continue to be influenced by other consumer-electronics hardware and applications. One event almost sure to occur is that players for DVDs (see Section 9.6) will be attached to PCs. At some future time, the allure of vastly increased storage capacity and higher data rates will prove irresistible to PC multimedia developers. If they choose to follow entertainment-industry standards, MPEG-2 encoded video and Dolby AC-3 encoded audio must be handled by multimedia PCs. MPEG-1 is hardly the final solution for PC video compression.

PC video hardware compression

PC manufacturers have found many ways to integrate multimedia processing, video compression included. Software-only video compression is one solution, but it is not a problem-free solution; namely a lot of processing muscle is required to decompress video and even more to compress it and dedicating the CPU to video processing may leave insufficient resources for running video together with anything else. More to the point, video processing demands continue to run ahead of CPU growth as visually better but computational more complex video compression algorithms invade the marketplace. Another factor that argues against a software-only solution is that most multimedia video applications demand both audio and video be compressed or decompressed together. Not only must audio and video be synchronized, but audio processing is CPU intensive. Also, audio and video data streams must be continually processed, which is difficult to do in the CPU, given the interrupt-driven nature of the PC. When interrupted, say by an I/O device, the CPU must suspend video processing, or whatever task it is doing, and go off to handle the interrupt. The result is pauses and gaps in audio and video delivery. Finally, a large amount of bandwidth is needed to move uncompressed video around inside the PC. What this means will be examined momentarily, but the end result is that the CPU may be starved for data, leaving it with nothing to compress or decompress, and, thus, interrupting the video (and audio) data stream.

When software-only processing is not enough, specialized hardware can provide the resources needed for processing audio and video. It can assure continuous processing of the data streams and, when properly placed, assure uncompressed video will not overwhelm the PC dataflow. With the aid of Figure 9.11, we will explore several options for placing hardware in the PC dataflow to accelerate audio and video processing. Ideally, to get the most out of data compression—to aggressively reduce storage and bandwidth—video should be compressed in the video subsystem and decompressed in the graphics controller. In this way, video is compressed when captured and remains compressed until needed for

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

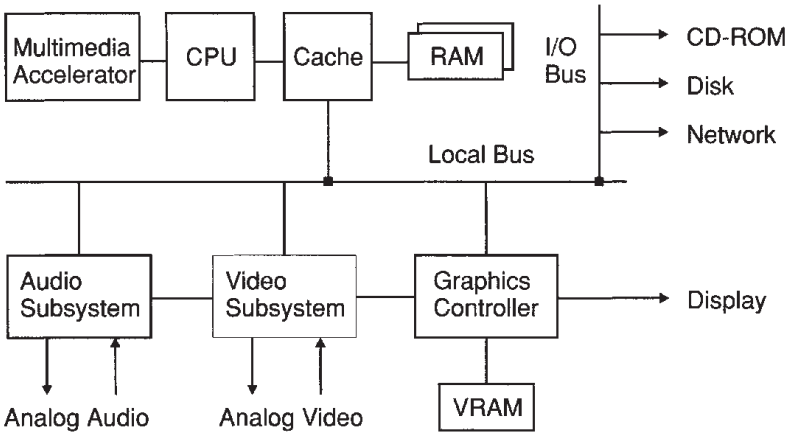


Figure 9.11 Multimedia PC organization.

display. It is always stored in compressed format, and no internal PC data path ever carries uncompressed video. For audio, ideally, both compression and decompression should be placed in the audio subsystem. Audio is compressed as it is captured and remains compressed until sent to the output devices. So much for what is ideal; here is reality:

First, there are multimedia-accelerated CPUs. PC manufacturers and silicon designers have devised several schemes for turning the CPU into a more efficient software engine for graphics, image, audio, and video processing. Included are the following:

- CPU chips that incorporate specialized multimedia instructions. Companies including Intel, Hewlett-Packard, and Sun Microsystems have announced multimedia CPU chips [Wils94A, Lee95, Ocon95, Wolf95B]. The new instructions in these processors comprise core operations needed to more efficiently manipulate pixels and other complex diffuse data. The performance gained by this approach is limited. For any given level of silicon integration, only a certain number of new instructions can be accommodated on-chip, and the CPU dataflow remains attuned to symbolic (text, numbers) data processing.
- Multimedia coprocessor chips. Several semiconductor companies have announced chips which act as companions to the host CPU chip [Clar95, Wils95C, Wils95G, Yosh95G]. These chips handle only multimedia data and are appropriate when the highest levels of performance are needed. They forego conventional dataflows in favor of parallel architectures that promise up to 50 times the performance of conventional CPUs.
- Other approaches that use a combination of chips. The intent is to allow the CPU to take on some multimedia processing tasks while offloading others to achieve lower costs or greater integration for cost-size-power-sensitive products such as notebook computers. One example is MPEG-1 decompression as implemented by IBM Aptiva™ PCs. Video decompression is run in software on the CPU and audio decompression runs in its programmable Mwave™ DSP chip [Ozer95C].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Closely aligned with multimedia-accelerated CPUs is the Native Signal Processing™ architecture developed by Intel [Wils95B]. The idea is to surround the conventional CPU with simple, low-cost peripheral hardware, allowing the CPU to take on signal processing tasks formerly relegated to DSP chips. However, neither the hardware nor software of host CPUs is ideal for real-time processing operations such as audio compression or decompression. For example, consider the Truespeech voice compression algorithm described in Section 5.2. In a benchmark reportedly devised by Intel, decompression consumed 3% and compression 32% of the computing power of a 100-MHz Pentium processor [Frit95]. It is also important to recognize that this is a simple algorithm, at least when compared to algorithms such as Dolby AC-3. Therefore, this approach will likely need to employ a CPU chip enhanced with multimedia instructions and seems best suited to low-cost, highly integrated applications [Wils95D].

Multimedia-accelerated CPUs can compress and decompress data faster but do nothing to address the bus bandwidth needed for moving uncompressed video data around inside the PC. Consider MPEG-1, by today's standards a moderate-resolution video format. When uncompressed, MPEG-1 video consumes a significant portion of local bus bandwidth and exceeds the bandwidth of typical I/O buses.[16] Thus, moving compression and decompression out of the CPU makes sense for better performance when PCs must do more than just playback video, and for higher-resolution video applications, or for compressing video of any resolution. Referring again to Figure 9.11, there are two locations for dedicated video compression and decompression hardware—in the video subsystem and in the graphics controller.

Video subsystems provide the most flexible, highest-quality video processing and they offload all uncompressed video traffic from the PC buses. Originally, PC video subsystems were implemented as add-on boards (because the internal dataflows of 1980s PCs were totally incapable of handling video). Today, there are two choices: add-on and integrated video subsystems. Whether add-on or integrated, a video subsystem dedicates silicon to doing nothing other than video processing. In this way, it can be assured that 30 fps of uncompressed video will be delivered to the graphics controller (and synchronized, uncompressed audio to the audio subsystem). For compression, it is assured that analog video can be captured and compressed to whatever resolution and frame rate required.

Add-on boards still are the only real solution for compressing video, and they are popular options for high-quality video decompression. The original add-on video subsystem boards were programmable, allowing a variety of compression algorithms. Some still are, but the trend is to offer just one standardized compres-

[16] Uncompressed MPEG-1 video in 24-bit RGB-displayable format requires 7.6 MBS. The data rate for the ISA (Industry Standard Architecture) bus, a widely used I/O bus, is 5–8 MBS. The typical data rate for the PCI™ (Peripheral Component Interface) bus, a widely used PC local bus, is about 35–40 MBS.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

sion algorithm—some variation of MPEG. The second choice, an integrated video subsystem, is an affirmation of the power of standardization too. Decompression-only (MPEG-1) video subsystems are beginning to be integrated on the PC motherboard [Frit95]. Not only is the cost for dedicated MPEG-1 decompression hardware comparable to that for a faster CPU needed to handle decompression in software, but, given the state of the art, the video quality is better.

The graphics controller is the ideal site for video decompression (but only for decompression, not compression) because for playback, no other PC component needs to handle uncompressed video. (Uncompressed audio must be routed to the audio subsystem via a direct path not shown in Figure 9.11.) A cost advantage of doing decompression in the graphics controller is that the VRAM (video random access memory) may be shared by the decompressor and the graphics accelerator chips. The ultimate low-cost solution for MPEG-1 video playback (or any other video format) is an integrated video-compression-and-graphics-accelerator chip. Circa 1995, these functions were on separate chips, and industry observers thought they would remain that way [Turl95]. The reasons they cited were that MPEG-1 decompressors and graphics accelerators each require a large number of logic circuits that cannot be shared, making the chip very large in 1995 silicon technology, and no one could be certain how long MPEG-1 would be the "standard" for PC video. However, the PC industry does not stand still. In 1996, some MPEG-2 decompression functions began appearing in graphics controller chips, this in anticipation of DVD playback becoming a standard PC function [Wils96].

PC video summary

PC video can roughly be divided into three eras: the formative years, when nonstandard video compression and playback in software on standard PC hardware dominated; the mid-1990s, when MPEG-1 video playback in software or hardware on multimedia-enabled PCs became a standard and videoconferencing was still a future application in search of a standard; and the future, when (if predictions come true) MPEG-2 video playback and H.320/H.324 videoconferencing will be standards. Table 9.8 summarizes characteristics for the variety of software and hardware implementations discussed in this section.

9.12 Multifunction Office Machines

The digital Swiss Army knife for home and office seems a fitting description of multifunction office machines. Major manufacturers are producing all-in-one multifunction office machines priced within reach of small or home offices. These products combine the function of a FAX machine, a scanner, a printer, a copier, and a telephone. They can operate as a stand-alone copy machine, and when connected to a telephone line, they can be used as a FAX machine or simply as

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

13

Digital System Design

In Part III of this book, we discovered that many applications and digital systems take advantage of data compression, be it for storage savings, bandwidth reduction, or transmission time savings. Incorporating data compression in a digital system requires careful design to assure it will provide the intended benefits and operate harmoniously with other system functions. In this chapter, we introduce decisions that designers face when choosing to integrate data compression in their digital systems.

Digital systems of varying complexity have been encountered throughout this book. From a physical perspective, there are simple self-contained systems, such as consumer products. There are more complex systems composed of several components, such as computer systems, and there are distributed systems, really a system of many interconnected systems. Table 13.1 provides some examples. Whatever the case, the design process starts (or should start) with the marketplace. Inputs from the marketplace define the need for data compression, if there is one. Marketplace requirements are the foundation for generating system requirements and specifications. They essentially establish the constraints and framework within which system designers must work, including the type of data compression that is acceptable and its impact on and visibility to end users.

Designing a digital system involves making trade-offs. System designers must wrestle with many factors. These include architectural considerations and compatibility with prior implementations for similar products. Also, software and hardware design, size, and packaging considerations. Then there are system performance requirements, application functional requirements, product cost considerations, and the list goes on. When data compression is part of the solution, system designers must first understand the problems it addresses, followed by a

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Table 13.1 Digital systems.

Type	Structure	Examples
Simple	Self-containing product	Audio tape player Set-top box Digital camera Multifunction office machine
Complex	Multiple components	Desktop computer Multimedia server
Distributed	Interconnected systems	Radio or television broadcast chain Data communications network Telephone network

close inspection of the wake of interactions created by its introduction. These include additional hardware and software complexities, conflicts with other system functions, requirements for data interchange, and data quality compromises to name a few.

13.1 Data Compression—Do You Need It?

Essentially, the most important question that system designers can ask about data compression is whether it is needed. Are there alternatives that will address the storage, bandwidth, or transmission time reduction it provides—more simply, more economically, or with less disruption? That is, are there, in some sense, better alternatives? Surprisingly (at least in a book dedicated to promoting data compression), sometimes there are. Let us begin with some of the best reasons for using data compression:

- Standards call for data compression
- Competitive products use data compression
- The system cannot operate without data compression
- System specifications are improved
- System cost is reduced

Sometimes the marketplace dictates that data compression must be used; sometimes it does not. Interoperability with a standard that includes data compression

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

leaves system designers no alternative. Going head-to-head in the marketplace with products that use data compression—and advertise it as an advantage—is almost as compelling a reason. Also, sometimes storage and transmission technology limits are unyielding, and data compression is the only practical solution. This is most evident among current applications where communications bandwidth is insufficient, or outrageously expensive. Then there are the gray areas—the cases where data compression is an option, one that may lead to competitive advantages. For instance, it may provide greater capacity, higher data rates, or more throughput; or it may simply reduce cost, providing yet another form of competitive advantage.

When data compression is an option, not a requirement, system designers must weigh its advantages, balancing them against the effect it has on other system functions and the complexity it adds to system design. In working through this process, a list of reasons for *not* using data compression may emerge. That list might include the following:

- More difficult-to-explain system operations and more complex user interfaces
- More complex hardware and software designs due to variable storage sizes, variable data rates, and the need for buffering
- More processing time and/or more VLSI circuitry
- Longer delay times for real-time data operations
- Increased sensitivity to storage and transmission errors
- More complex data recovery with or without errors

With lossy compression algorithms the list grows longer. Two important reasons not to use lossy compression are as follows:

- Loss of audio, image, and video quality
- Loss of VCR-like functions (random accessibility) for video data

Clearly, many very successful products have dealt with all these issues, but they paid a price, for data compression rarely is free. When data compression is optional, system designers sometimes may find it more cost-effective or satisfactory not to use compression, but to add the extra storage or buy the additional bandwidth it would save. Each situation must be carefully analyzed.

13.2 Selecting an Algorithm

Once a genuine need for data compression is established, system designers face the task of selecting a compression algorithm and adapting it to their application. Increasingly, the particular algorithm for an application is predetermined by prevailing official or de facto standards. In any event, system designers must

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

still make implementation trade-offs, trade-offs that can affect the compression ratio and many other factors which determine how the algorithm will operate. Data compression algorithms frequently are compared just on compression ratio. Compression ratio is important, but a more thorough characterization includes the factors listed in Table 13.2, all of which are considered in this section.

Table 13.2 Compression algorithm characteristics.

Factor	Importance
Compression ratio	Determines storage and bandwidth required
Adherence to standards	Key to widespread adoption of an algorithm
Algorithm complexity, speed, and processing delay	Determines implementation, cost-performance, and application acceptability
Adaptive versus nonadaptive	Key to supporting applications characteristics and needs
Adaptivity and robustness	Defines allowable input data variations
Error tolerance	Key to success in error-prone environments
Encoder/decoder asymmetry	Key to supporting application communications style
Quality	Defines the acceptability of lossy compression
Multiple-coding limitations	Key to repeated usage of lossy compression
Scalability	Required for differing video display resolutions
Progressive resolution	Required for progressive image transmission at various resolutions
Operating bit rate	Defines acceptable range of data transmission speeds
Added function	Key to more application-friendly compression

Compression ratio

Since compressing data is what this is all about, one would think higher compression ratios are always better, but that is not so. As a rule, getting more compression means making bigger investments in algorithm complexity, process-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

ing power, and time. However, these trade-offs have limits: If the compression is lossless, the gains in compression and the added investment costs become marginal as the point is reached where all redundancy has been removed. When the compression is lossy, more compression means more information loss, which nearly always leads to lower-quality results. The most practical approach is to shoot for the minimum compression ratio needed, balancing system storage and bandwidth limits and making adjustments when it is not feasible or acceptable to do it all with compression.

Adherence to standards

Standards for data compression algorithms and their importance are focal points of this book. They exist and many applications demand they be used. Nevertheless, the existence of a standard does not rule out innovative implementations. Many standards for data compression algorithms really are nothing more than frameworks that describe the encoded bit stream and how it is to be transmitted between the encoder and decoder. Many examples throughout the book show that system designers are free to follow different paths in creating and sometimes in decoding those bit streams. They may choose to simplify the encoding process in the name of cost savings, to add complexity in the name of speed or transmission bandwidth reduction, and so on. The only real constraint is that the encoded bit stream conforms to the standard. Some may also choose to develop implementations that "enhance" the standard, but this user-beware strategy is fraught with danger because, in effect, it creates a new standard that defies interoperability.

If compressed information must be communicated between systems and between products from different manufacturers, then a standard for compression and decompression that everyone agrees to is the only choice. Many applications, as we have seen, have already settled on an official standard. But standard algorithms are not always the best choice for applications that have control over their environment, such as a self-contained application (described as a closed process in Section 3.4) that can, if it chooses to do so, implement built-in proprietary compression technology. In addition, there are many other reasons why selecting a proprietary algorithm makes good sense, including taking advantage of technology advances the standard algorithms do not offer to gain more compression, or higher-quality results, or finding a simpler implementation, or having a market advantage by being proprietary. There are also hazards in selecting a proprietary algorithm. What happens to your product if the algorithm vendor goes out of business and, therefore, can no longer provide regular updates and improvements or, equally important, fixes when problems arise? A solution gaining in popularity, at least for software, is to require that vendors place a copy of their code in escrow with an agreement that, if the unthinkable should arise, this information be made legally available to their customers [Seby96, WinW96].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Algorithm complexity, speed, and processing delay time

Algorithm complexity, speed, and processing delay relate to the computational workload and memory requirements that an algorithm imposes, to the need for parallel processing or other efficient processing techniques, and to how fast the algorithm will operate in whatever implementation chosen. These factors decide whether software, DSP, or VLSI hardware logic implementations are suitable for an algorithm and which one(s) will be needed to meet the performance requirements of an application. Two-way communications applications are most sensitive to processing delay time for encoding and decoding, because human interaction places limits on what is acceptable. One-way communications applications and storage applications, where processing delay times are largely irrelevant, are sensitive to processing speed, as throughput determines what level of performance is acceptable.

Adaptive versus nonadaptive

The questions about adaptive algorithms relate to how to design them and when to apply them. Adaptive algorithms continually adjust their parameters to match the data being encoded or decoded. If done properly, more compression will be obtained, as compared to nonadaptive algorithms that operate with static, fixed rules for encoding and decoding. There are some trade-offs to consider when selecting an adaptive algorithm: All adaptive algorithms depend on past history. They have a start-up time, requiring some number of bytes be processed before the algorithm can adjust to the data and the best compression is achieved. The start-up time is a fundamental limitation on the kinds of applications that can use adaptive compression algorithms. Some adaptive algorithms are designed to have shorter start-up times than others. When small, randomly accessed blocks of data are to be compressed, a short start-up time is an advantage, but nonadaptive or semiadaptive algorithms may be still better choices. Examples in Chapters 4 (see Table 4.2) and 6 (see Section 6.9) suggest that a few hundred bytes of data are needed to adapt lossless Lempel-Ziv compression algorithms. Lossy algorithms have start-up times too. For instance, MPEG (see Section 5.5) must encode at least one frame (an I-frame) before motion compensation and interframe prediction can work.

Adaptivity and robustness

Adaptivity and robustness refer to the ability of an algorithm to continually adjust to changes in the data and to produce quality results under adverse conditions. Adaptive algorithms have proven to be superior performers on symbolic data. Many of these algorithms, which were initially designed for text, are robust in the sense that they can handle (or can be extended to handle) graphics and other forms of information. For example, see the discussion of the IBM ADLC

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

algorithm in Section 9.12. Most algorithms for diffuse data make limited use of this type of adaptivity. For example, the final stage of the JPEG algorithm allows adaptive Huffman and arithmetic entropy coding as options [Penn93]. However, the real opportunity for adaptivity is to deal with the variability of the input characteristics of audio, image, and video data. Diffuse data algorithms can be made more robust by applying adaptivity to varying characteristics such as data statistics, dynamic range, noise, frequency content, pixel-to-pixel correlation, and image resolution. Adaptivity can be applied at many steps of the algorithmic process, including transformation (to adjust the transform block size in transform coding, for example), coefficient selection, quantization step size, motion estimation and compensation, and so on.

Error tolerance

Error tolerance relates to the ability to encode and decode data successfully in the presence of errors that may occur during processing, storage, or transmission. Without question, data compression (coding) schemes are far more sensitive to transmission channel errors than all others. This stems from the fact that, totally apart from the needs of data compression, error control coding and recovery procedures are standard in the processing logic and storage devices used for most digital (computing) systems. These elements handle compressed data with the same aplomb as uncompressed data, with transparent error correction that assures near absolute accuracy, and with little or no increase in running time. Consequently, data compression algorithms intended for computing and storage applications include few if any special error control or recovery provisions.

This is not so for transmission channels. Here, the error rates often are much higher, meaning error recovery must be invoked with greater regularity. Another difference is the user (in this case the compression algorithm), not the transmission channel itself, may be responsible for system error control and recovery. Particularly for real-time transmissions, standard error control and recovery schemes are either far more intrusive or far less effective. For example, a packetized data network typically accomplishes transmission channel error recovery by resending corrupted packets. This form of system error recovery may not be practical for compressed data streams of audio and video data, where the latency of retransmission interferes with real-time playback. In this situation, a much more effective strategy is to limit the effects of a channel error to a small part of the data stream—a few seconds of audio, a small area of an image, or a few frames in a video stream. Section 14.2 will discuss techniques for doing this, but today, as a general rule, each compression algorithm for each application is forced to make additional specific provisions for error handling for each transmission channel. Numerous examples of what has been done are provided in Chapters 5 and 7. The price to be paid, unfortunately, is increased complexity and added bit rate (overhead) for error control and recovery.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Encoder/decoder asymmetry

Encoder/decoder asymmetry describes the expenditure of silicon and time for encoding in relation to the same factors for decoding. The basic idea of asymmetric coding is that a larger investment in encoding will result in fewer bits to be stored or transmitted. Asymmetric algorithms are frequently found in storage or broadcast applications, where encoding is done once and decoding is done many times. Most lossless algorithms are inherently symmetric and there is little than can or should be done to change them. Most lossy algorithms, by their very nature, are asymmetric because deciding what information to remove is frequently the most complex and time-consuming part of the encoding-decoding process. However, some of today's leading lossy algorithms allow trade-offs that will make them more or less asymmetric. Several applications illustrate that intelligently expending more encoding logic and time can result in both higher-quality audio, image, or video data and fewer bits to store or transmit. For example, see the discussion of DBS in Section 8.3 and CD-ROM publishing in Section 10.3.

Quality

Quality refers to the distortion that lossy coding introduces. Because lossy coding involves removing some data, compressed diffuse (speech, audio, image, and video) data may not retain all the quality of the original. As noted in some of the applications described earlier in this book, sometimes lossless compression of diffuse data is required when no loss of quality can be accepted, such as for nonhuman data processing or to meet legal requirements. But what is quality, particularly, what is quality when lossy psychoacoustical and psychovisual models are used for compression? For instance, image quality is a complex psychovisual combination of spatial resolution (detail, sharpness), temporal resolution (frame rate), and the effect of artifacts introduced by the compression process. Unfortunately, there are no good, easily applied quality measures for images or other data subjected to lossy compression.

Standard engineering measures of signal distortion, such as mean-squared error, frequency response, or anything that can be measured by linear instruments, frequently do not correlate well with how humans perceive quality, although, these measures are what engineers often resort to because little else is available. When humans are asked to examine a set of images and make decisions about their quality, they use subjective criteria. Subjective criteria such as whether an image is "visually lossless" are useful, but the downside is they have imprecise definitions. Subjective evaluations (for instance, see Section 12.5), despite their unreliability and time-consuming cost, are the best available tools for judging how objectionable the artifacts created by lossy coding systems are. The types of artifacts produced by these systems include the following:

- Audio distortion due to quantization error (bits/sample)
- Image softness due to low resolution

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

- Image fringing, blocking, or edge jaggedness
- Loss of image sharpness at edge transitions
- Random noise in images
- Video blurring due to limits in motion estimation

What is known today is that different algorithms create different artifacts, depending on their mode operation. Even a single given algorithm may exhibit different artifacts, depending on the bit rate at which it is operated. The presence of artifacts usually is more apparent at high compression ratios. Also, the objectionability of artifacts is highly dependent on the particular sound or image and the conditions under which it is presented to the listener or viewer.

Quality measures, perhaps, will be easier to develop for audio because we understand (or think we understand) quite well how the human ear perceives sound. But good quality measures for images and video are more difficult to develop. Although our understanding of the human visual system has increased significantly over the past century, it has also revealed many complexities of the visual system. This includes how shape, intensity, color, and motion are perceived. It also includes how we are affected by various media and different techniques to display images (for example, subtractive colors for printers and additive colors for displays), and so on. A simple and yet powerful model for visual processing, one that can be used for optimizing coding systems, remains a goal that still eludes the experts [Netr95].

Multiple-coding limitations

Multiple-coding limitations relate to the loss of quality that may result when data is repeatedly compressed and decompressed with lossy algorithms. Some applications may require that compression-decompression be repeatedly applied during transmission, or for storage on different media, or simply for processing such as when an image is rescaled, clipped, or cropped during document processing. Clearly, successive applications of lossless algorithms do not affect quality, but successive applications of most lossy algorithms cause additional degradation of audio, image, or video quality. Unfortunately, there are no general rules or guidelines that explain when multiple compression-decompression cycles can be safely used. Worse yet, as discussed in Chapter 14, when different algorithms are cascaded, the results can be unpredictable.

Scalability

Scalability, a term often restricted to video coding, may be broadly defined as the generation of bit streams that produce useful results even when only a subset has been decoded. Partial decoding of a video bit stream may be useful if the viewer wants an image of smaller size than the source, or the decoder lacks processing resources to decode the full bit stream, or insufficient channel capacity exists (perhaps momentarily) to carry the full signal. Existing television

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

standards (NTSC, PAL, and SECAM), like early proposals for HDTV standards, make no provisions to decouple the parameters of encoding from decoding. This means that all receivers must support the same bandwidth, resolution, aspect ratio, and frame rate. With the coming of HDTV, scalability begins to make sense because not every viewer or receiver needs its full resolution and features. A small-screen receiver to be used for casual viewing would be more economical to build if it could decode a subset of the standard. Similar conditions exist for displaying video on computer workstations, where small screens and smaller-yet windows are the norms. Provisions for scalability are incorporated in the emerging U.S. HDTV standard [Chal95]. Essentially, what is required to achieve scalability within the context of the standard are spatial conversion, frame-rate change, and conversion between interlaced and noninterlaced formats [Basi95].

Progressive resolution

Progressive resolution, a form of scalability usually restricted to images, refers to coding and transmitting images at different resolutions or precisions. The idea is to code an image in several passes and sequentially provide these encodings to viewers. The coding may take the form of spectral selection, where image detail is grouped into bands of visual frequency coefficients. It may also be accomplished by successive approximation, where the precision of all image components is limited in early passes, and later passes extend the precision. Either way, transmission begins with an image that can be quickly sent at a low bit rate, allowing the viewer to recognize the image contents as soon as possible. The viewer, perhaps, may choose to halt the transmission of unnecessary detail or may choose to initiate actions for transmission of additional detail. Progressive presentation, besides providing an aesthetically pleasing sequence of images that holds the viewer's attention, allows more efficient browsing through a picture database. This is also known as hierarchical or pyramidal multiresolution coding as described in the Photo CD example that appears in Section 9.9; see also [Rabb91].

Progressive resolution techniques are being actively investigated; many different compression techniques are inherently progressive or can be adapted to operate progressively. However, today there are no strong solutions for managing image compression fidelity, either to satisfy end-user requirements for spatial or amplitude resolution or to meet end-user response time requirements, or to operate within available decompression time limits, or to match available transmission media bandwidth.

Operating bit rate

Operating bit rate refers to the number of bits per second transmitted from the encoder to the decoder. It is usually associated with real-time transmission of audio or video and, sometimes, with image transmission. An ideal compression algorithm would be able to operate over a wide channel-bandwidth range, provid-

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

ing the highest-quality audio, image, or video sustainable within available bandwidth. The algorithm would be tunable by adjusting a small set of compression parameters to trade off quality for bit rate. Some of today's algorithms, including JPEG, H.261, and MPEG, allow some amount of bit-rate tunability. Others, including nearly all speech coding algorithms, are designed to work only at a given bit rate.

Most compression algorithms are designed to work efficiently within a certain bit-rate range. They may become inefficient or inoperable for transmission rates that fall outside that range—inefficient in the sense that increasing the bit rate will not measurably improve quality, and inoperable in that there is a lower limit below which insufficient transmission bandwidth is available for sending the minimal required information. For example, the H.261 videoconferencing algorithm is designed to operate at not more than 1.92 Mbps and not less than 64 Kbps. At higher rates, there is simply little if any additional information that can be sent to improve picture quality, and at lower rates there is insufficient bandwidth for sending the minimum number of transformed video blocks and motion vectors.

Among those algorithms that can operate at varying bit rates, some require the bit rate be selected and remain fixed for the life of an application, whereas others support dynamically variable bit rates. Examples of the latter include MPEG-2 as implemented for DBS transmission (described in Section 8.3) and for DVD storage (described in Section 9.6). The JPEG hierarchical coding mode also essentially transmits data in this manner.

Added function

This catchall category includes functions that compression algorithms may include to make compression more tolerable for an application. Examples include the sort-order preserving and semiadaptive algorithms developed for database applications (described in Section 6.9).

13.3 Selecting an Implementation Technique

Implementations ranging from software to dedicated special-purpose hardware allow data compression to be adapted to diverse market segments and products, each having differing economics and differing compression needs. Historically, in the computer industry, early compression techniques were executed in software as programs of conventional instructions. In the 1970s, IBM researchers were among the first to define special microcoded CPU instructions and dedicated hardware to speed up data compression techniques such as Huffman coding [Momm74]. By the 1980s, the complexity of compression algorithms had grown, and with more applications demanding faster operation, several VLSI codec chips came to market [Arps88]. In the communications industry, dedicated codec

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

hardware always has been the preferred implementation, essentially because real-time operations such as speech coding demand it and because computers and software that can operate in real time (or near real time) are relative newcomers in communications applications. In consumer-electronics products, where programmable computing technology is not always a given, hardware-implemented compression is often the method of choice.

The options

The optimal choice for implementation depends on the system requirements. Table 13.3 introduces four cost-performance options available to digital system designers for implementing data compression.

Table 13.3 Data compression implementations.

Option	Performance	Comments
CPU software	Low	Programmable
CPU accelerator	Medium - high	Programmable
DSP chip	Medium	May be programmable
ASIC codec chip	High	Fixed algorithm

CPU software compression is an easy way for computer-based products to introduce new function to the marketplace. For instance, when multiple data formats must be handled, a single hardware solution may be impractical. CPU software offers the cheapest and most flexible solution for many compression applications, as it can handle multiple formats and evolve most quickly to new formats. CPU software also is the right choice in situations calling for occasional usage. For example, when a data file is imported from another system in a heterogeneous computing environment, a software format translator is all that is needed. CPU software is not satisfactory in some situations because of computational complexity, resolution, or rate-of-delivery requirements; there one of the following solutions is often necessary.

Special-purpose CPU accelerators allow the flexibility of software programmability to be retained while providing vastly improved performance. These accelerators may take the form of specialized instructions added to the regular CPU chip, but specially designed coprocessor chips can be far more powerful. As learned in Section 9.11, both are available for multimedia computing applications where data compression and other audio, image, and video data operations require many times the performance of a regular CPU.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The DSP (digital signal processor) chip is a 1990s phenomenon finding its way into more and more applications. The primary advantage of a DSP chip is the specialized dataflow which allows high performance execution of real-time (signal) processing algorithms with far less expense and energy consumption than any CPU chip intended for general-purpose computing. Today's DSPs cost as little as $10, or less, and they are embedded in many portable products. They also offer the right cost-performance for high-volume, low-end home and small office environment applications such as audio products, modems, telephones, and FAX machines. They are used for implementing many diffuse data compression algorithms discussed in this book. (DSPs are not promoted as engines for implementing lossless symbolic data compression, although it is technically feasible.) The market for DSPs is about equally split between programmable chips that may be downloaded with various algorithms and fixed-function, algorithm-specific chips used for embedded applications where programmability is not required.

ASIC (application-specific integrated circuit) codec chips are appropriate when high-performance data compression solutions are required. In particular, ASIC codec chips may provide the only viable implementation of highly complex, computationally intense algorithms intended for real-time data compression applications. They also may be needed for simpler algorithms in applications where general-purpose computing logic is not available. When compared to the alternatives, ASIC codec chips can also be the lowest-cost solution for these applications. Data compression codec chips are available for almost all speech and audio algorithms, for JPEG and other image compression algorithms, and for video algorithms, including H.261 and MPEG. Chips also are available for many lossless data compression algorithms including Lempel-Ziv and binary arithmetic coding.

Choosing an option

Choosing an implementation for data compression is easy when the digital system design or the application itself dictates the correct choice. When a product does not include a general-purpose CPU, DSPs and ASIC codecs are the only choices. The same is true when software data compression performance is woefully inadequate. But there are gray areas, especially when a product does feature a CPU, one seemingly having enough performance to do software compression. In these situations, specialized data compression hardware could still be a wise choice. The issue is having a balanced design. One possible decision-making approach is to try to quantify the ratio between the operations-per-second devoted to compression versus those available for noncompression-related work. The larger this ratio, the more justified is hardware support for compression. Consider how this classification scheme might apply a hypothetical product built around a modern microprocessor: It would likely reveal that speech could be processed without special hardware. The same could be true for still-image processing if no hard real-time requirements are imposed. However, it would likely point to the need for compression hardware for high-quality video processing.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

Future directions

To appreciate what future data compression implementations may be, consider what has happened in the past. Computer CPUs have become microprocessors—smaller, cheaper, and faster—and far more pervasive in products from all industries, even low-cost, mass-produced products. Microprocessors are now able to execute nearly all symbolic data compression algorithms at speeds that satisfy most applications and at costs that are affordable for scores of products. Simultaneously, compression algorithms have grown more complex—handling speech, audio, and video data types with real-time requirements. Where only a few short years ago it was inconceivable that *any* of these real-time algorithms could be executed in software, as a rule today's microprocessors have advanced to execute nearly *all* these algorithms with at least marginally acceptable performance.

Some type of compression hardware assist is necessary for many of today's algorithms, but time and technology march on. One can assert, with visionary clarity, that someday soon, all of *today's* data compression algorithms will operate quite adequately and cost-effectively in software on microprocessors of the future. Those microprocessors will somehow include the processing logic needed for executing the basic data manipulations on which today's algorithms are based. But it would be shortsighted to assume that all future implementations of data compression will be in software. They will not. Data compression algorithms, particularly those for video (and probably speech, audio, and maybe still images too) are going to grow still more complex. Continuing demand for higher-quality video images, transmitted at ever-lower bandwidth and stored evermore compactly, assure that algorithm development must continue. Thus, leading-edge algorithms will undoubtedly continue to require specialized hardware implementations.

13.4 Where To Do Data Compression

Location, location, location: With data compression, as in real estate, location is everything. As asserted throughout this book, from a technology perspective, the optimum location for data compression is close to the source or producer of data, and for decompression, close to the end user or consumer of data. As illustrated in Figure 13.1, this produces the maximum benefit by reducing bandwidth and minimizing storage (and processing time) for all hardware and software elements throughout the system. Clearly, some systems are so simple that no other choice is possible; the digital speech products described in Section 9.13 are a classic example. For almost all systems that handle video data, short of a hugely expensive investment in networks and storage, this is the only practical way to support the data rates needed for real-time display.

Deciding where to place compression and decompression can be far more complex for most digital systems, particularly, digital computer systems. Figure 13.2 shows the work flow for a database computer system. A human interacts

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

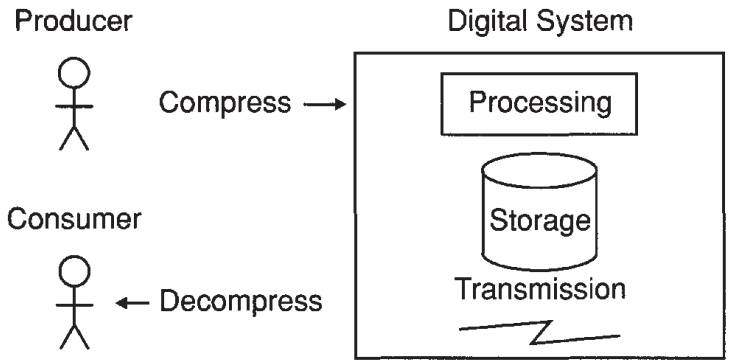


Figure 13.1 Optimal compression for a simple digital system.

with a workstation (WS) connected to a network, which, in turn, is connected to a server computer. Data enters the server through an I/O channel interface (IFAC). It passes through several software components, beginning with communications (COMM), through an application program (APPC), and then is passed to a database management system (DBMS), all of which run in the main storage (MSTG). Data is then routed through I/O channel interfaces to a disk subsystem where it enters a buffer (BUFF) and, subsequently, is passed through yet another interface to disk storage. Data is returned through these same elements in the reverse direction. Along this path there are many sites where it is possible to do compression/decompression functions (indicated by arrows), in hardware or in software.

This situation is quite different from earlier examples for several reasons. To begin with, the data rates and the amount of data for commercial database management applications are much lower; all components are quite able to operate without compression, but there are system cost and performance advantages when it is used. Another difference is that somewhere in this path, data must be processed—not simply passed along. At minimum, the application program, the DBMS code, and, usually, both must examine the data and make logical decisions based on its values. So, the strategy of compressing data when the producer generates it, and leaving it compressed until the consumer needs it, simply does

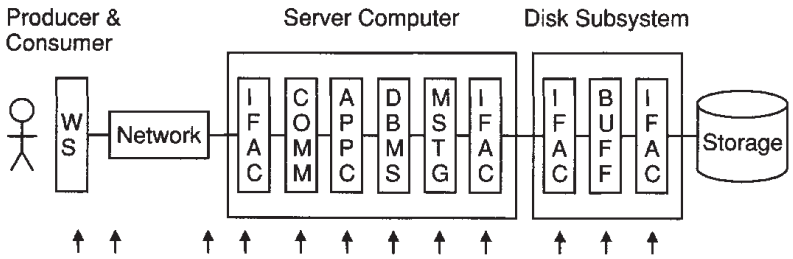


Figure 13.2 Work flow into and out of a database computer.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

APPENDIX D

not work. At least this is true for data processing as we know it today because, generally, the technology to execute program logic on compressed data does not exist. (There are exceptions, such as the compressed DBMS index operations discussed in Section 6.9.) Finally, there are marketplace factors that come into play. A system such as the one shown in Figure 13.2 typically is assembled from components from several sources—a workstation from a manufacturer of computer peripherals, a network from the telecommunications industry, a server from a computer manufacturer, software from several suppliers, and so on. Each manufacturer may be attracted to data compression, using it in their product for the marketing advantages it can offer and choosing to compress and decompress at the product interfaces. Thus, data may be compressed and decompressed several times within this system, a situation that could be interpreted as wasted motion (or processing cycles), but there are valid reasons. It must be noted, from a technology viewpoint, that this is acceptable if the compression is lossless (which it is in business-oriented database computer systems).

In summary, the "rule" that data compression should be placed close to the source of data and decompression placed close to where data is used does not always apply. Sometimes marketplace decisions outweigh technical decisions, and in these situations data compression/decompression can be placed anywhere that adequate computing and storage resources are available and wherever the marketplace makes it profitable!

13.5 Summary

This chapter examined decisions that designers face when choosing to integrate data compression in their digital systems. We learned the following:

- Marketplace requirements define the need for data compression, the type of compression that is acceptable, and limits within which system designers must work.
- To assure data compression can be effectively integrated, system designers must understand not only the problems data compression addresses but also its interactions with other system functions.
- Essentially, the most important question that system designers can ask about data compression is whether it is needed or whether other alternatives may provide a better choice.
- Sometimes the marketplace or the limitations of storage and transmission technology demand that data compression be used; in other situations it is an option, providing competitive advantage through improved system cost-performance.
- System designers must deal with the costs for using data compression including its effect on other system functions and the complexity it adds to system design.
- All too frequently, the compression ratio is the only parameter used to evaluate compression algorithms; actually, there are many other characteristics that may be far more important in many system designs.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

- Four cost-performance options are available for doing data compression, ranging from software to dedicated special-purpose hardware. The optimal choice for implementation depends on the system requirements.
- Software compression is becoming more powerful, but most leading-edge algorithms require some type of hardware assist and will continue to do so.
- The optimum location for data compression is close to the source or producer of data, and for decompression, close to the end user or consumer of data. But this cannot always be achieved for a variety of technical and marketing reasons.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

WIKIPEDIA
The Free Encyclopedia

Data compression

In information theory, **data compression**, **source coding**,[1] or **bit-rate reduction** is the process of encoding information using fewer bits than the original representation.[2] Any particular compression is either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. Lossy compression reduces bits by removing unnecessary or less important information.[3] Typically, a device that performs data compression is referred to as an encoder, and one that performs the reversal of the process (decompression) as a decoder.

The process of reducing the size of a data file is often referred to as data compression. In the context of data transmission, it is called source coding: encoding done at the source of the data before it is stored or transmitted.[4] Source coding should not be confused with channel coding, for error detection and correction or line coding, the means for mapping data onto a signal.

Compression is useful because it reduces the resources required to store and transmit data. Computational resources are consumed in the compression and decompression processes. Data compression is subject to a space–time complexity trade-off. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed, and the option to decompress the video in full before watching it may be inconvenient or require additional storage. The design of data compression schemes involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (when using lossy data compression), and the computational resources required to compress and decompress the data.[5]

Lossless

Lossless data compression algorithms usually exploit statistical redundancy to represent data without losing any information, so that the process is reversible. Lossless compression is possible because most real-world data exhibits statistical redundancy. For example, an image may have areas of color that do not change over several pixels; instead of coding "red pixel, red pixel, ..." the data may be encoded as "279 red pixels". This is a basic example of run-length encoding; there are many schemes to reduce file size by eliminating redundancy.

The Lempel–Ziv (LZ) compression methods are among the most popular algorithms for lossless storage.[6] DEFLATE is a variation on LZ optimized for decompression speed and compression ratio, but compression can be slow. In the mid-1980s, following work by Terry Welch, the Lempel–Ziv–Welch (LZW) algorithm rapidly became the method of choice for most general-purpose compression systems. LZW is used in GIF images, programs such as PKZIP, and hardware devices such as modems.[7] LZ methods use a table-based compression model where table entries are substituted for repeated strings of data. For most LZ methods, this table is generated dynamically from earlier data in the input. The table itself is often Huffman encoded. Grammar-based codes like this can compress highly repetitive input extremely effectively, for instance, a biological data collection of the same or

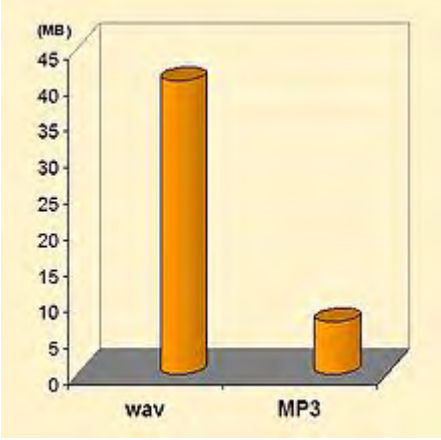
closely related species, a huge versioned document collection, internet archival, etc. The basic task of grammar-based codes is constructing a context-free grammar deriving a single string. Other practical grammar compression algorithms include Sequitur and Re-Pair.

The strongest modern lossless compressors use probabilistic models, such as prediction by partial matching. The Burrows–Wheeler transform can also be viewed as an indirect form of statistical modelling.[8] In a further refinement of the direct use of probabilistic modelling, statistical estimates can be coupled to an algorithm called arithmetic coding. Arithmetic coding is a more modern coding technique that uses the mathematical calculations of a finite-state machine to produce a string of encoded bits from a series of input data symbols. It can achieve superior compression compared to other techniques such as the better-known Huffman algorithm. It uses an internal memory state to avoid the need to perform a one-to-one mapping of individual input symbols to distinct representations that use an integer number of bits, and it clears out the internal memory only after encoding the entire string of data symbols. Arithmetic coding applies especially well to adaptive data compression tasks where the statistics vary and are context-dependent, as it can be easily coupled with an adaptive model of the probability distribution of the input data. An early example of the use of arithmetic coding was in an optional (but not widely used) feature of the JPEG image coding standard.[9] It has since been applied in various other designs including H.263, H.264/MPEG-4 AVC and HEVC for video coding.[10]

Archive software typically has the ability to adjust the "dictionary size", where a larger size demands more random access memory during compression and decompression, but compresses stronger, especially on repeating patterns in files' content.[11][12]

Lossy

In the late 1980s, digital images became more common, and standards for lossless image compression emerged. In the early 1990s, lossy compression methods began to be widely used.[13] In these schemes, some loss of information is accepted as dropping nonessential detail can save storage space. There is a corresponding trade-off between preserving information and reducing size. Lossy data compression schemes are designed by research on how people perceive the data in question. For example, the human eye is more sensitive to subtle variations in luminance than it is to the variations in color. JPEG image compression works in part by rounding off nonessential bits of information.[14] A number of popular compression formats exploit these perceptual differences, including psychoacoustics for sound, and psychovisuals for images and video.



MP3, an example of a lossy file format compared to WAV.

Most forms of lossy compression are based on transform coding, especially the discrete cosine transform (DCT). It was first proposed in 1972 by Nasir Ahmed, who then developed a working algorithm with T. Natarajan and K. R. Rao in 1973, before introducing it in January 1974.[15][16] DCT is the most widely used lossy compression method, and is used in multimedia formats for images (such as JPEG and HEIF),[17] video (such as MPEG, AVC and HEVC) and audio (such as MP3, AAC and Vorbis).

Lossy image compression is used in digital cameras, to increase storage capacities. Similarly, DVDs, Blu-ray and streaming video use lossy video coding formats. Lossy compression is extensively used in video.

In lossy audio compression, methods of psychoacoustics are used to remove non-audible (or less audible) components of the audio signal. Compression of human speech is often performed with even more specialized techniques; speech coding is distinguished as a separate discipline from general-purpose audio compression. Speech coding is used in internet telephony, for example, audio compression is used for CD ripping and is decoded by the audio players.[8]

Lossy compression can cause generation loss.

Theory

The theoretical basis for compression is provided by information theory and, more specifically, Shannon's source coding theorem; domain-specific theories include algorithmic information theory for lossless compression and rate–distortion theory for lossy compression. These areas of study were essentially created by Claude Shannon, who published fundamental papers on the topic in the late 1940s and early 1950s. Other topics associated with compression include coding theory and statistical inference.[18]

Machine learning

There is a close connection between machine learning and compression. A system that predicts the posterior probabilities of a sequence given its entire history can be used for optimal data compression (by using arithmetic coding on the output distribution). Conversely, an optimal compressor can be used for prediction (by finding the symbol that compresses best, given the previous history). This equivalence has been used as a justification for using data compression as a benchmark for "general intelligence".[19][20][21]

An alternative view can show compression algorithms implicitly map strings into implicit feature space vectors, and compression-based similarity measures compute similarity within these feature spaces. For each compressor C(.) we define an associated vector space ℵ, such that C(.) maps an input string x, corresponding to the vector norm ||~x||. An exhaustive examination of the feature spaces underlying all compression algorithms is precluded by space; instead, feature vectors chooses to examine three representative lossless compression methods, LZW, LZ77, and PPM.[22]

According to AIXI theory, a connection more directly explained in Hutter Prize, the best possible compression of x is the smallest possible software that generates x. For example, in that model, a zip file's compressed size includes both the zip file and the unzipping software, since you can't unzip it without both, but there may be an even smaller combined form.

Examples of AI-powered audio/video compression software include VP9, NVIDIA Maxine, AIVC, AccMPEG.[23] Examples of software that can perform AI-powered image compression include OpenCV, TensorFlow, MATLAB's Image Processing Toolbox (IPT) and High-Fidelity Generative Image Compression.[24]

Data differencing

Data compression can be viewed as a special case of data differencing.[25][26] Data differencing consists of producing a *difference* given a *source* and a *target,* with patching reproducing the *target* given a *source* and a *difference*. Since there is no separate source and target in data compression, one can consider data compression as data differencing with empty source data, the compressed file corresponding to a difference from nothing. This is the same as considering absolute entropy (corresponding to data compression) as a special case of relative entropy (corresponding to data differencing) with no initial data.

The term *differential compression* is used to emphasize the data differencing connection.

Uses

Image

Entropy coding originated in the 1940s with the introduction of Shannon–Fano coding,[27] the basis for Huffman coding which was developed in 1950.[28] Transform coding dates back to the late 1960s, with the introduction of fast Fourier transform (FFT) coding in 1968 and the Hadamard transform in 1969.[29]

An important image compression technique is the discrete cosine transform (DCT), a technique developed in the early 1970s.[15] DCT is the basis for JPEG, a lossy compression format which was introduced by the Joint Photographic Experts Group (JPEG) in 1992.[30] JPEG greatly reduces the amount of data required to represent an image at the cost of a relatively small reduction in image quality and has become the most widely used image file format.[31][32] Its highly efficient DCT-based compression algorithm was largely responsible for the wide proliferation of digital images and digital photos.[33]

Lempel–Ziv–Welch (LZW) is a lossless compression algorithm developed in 1984. It is used in the GIF format, introduced in 1987.[34] DEFLATE, a lossless compression algorithm specified in 1996, is used in the Portable Network Graphics (PNG) format.[35]

Wavelet compression, the use of wavelets in image compression, began after the development of DCT coding.[36] The JPEG 2000 standard was introduced in 2000.[37] In contrast to the DCT algorithm used by the original JPEG format, JPEG 2000 instead uses discrete wavelet transform (DWT) algorithms.[38][39][40] JPEG 2000 technology, which includes the Motion JPEG 2000 extension, was selected as the video coding standard for digital cinema in 2004.[41]

Audio

Audio data compression, not to be confused with dynamic range compression, has the potential to reduce the transmission bandwidth and storage requirements of audio data. Audio compression algorithms are implemented in software as audio codecs. In both lossy and lossless compression, information redundancy is reduced, using methods such as coding, quantization, DCT and linear prediction to reduce the amount of information used to represent the uncompressed data.

Lossy audio compression algorithms provide higher compression and are used in numerous audio applications including Vorbis and MP3. These algorithms almost all rely on psychoacoustics to eliminate or reduce fidelity of less audible sounds, thereby reducing the space required to store or

transmit them.[2][42]

The acceptable trade-off between loss of audio quality and transmission or storage size depends upon the application. For example, one 640 MB compact disc (CD) holds approximately one hour of uncompressed high fidelity music, less than 2 hours of music compressed losslessly, or 7 hours of music compressed in the MP3 format at a medium bit rate. A digital sound recorder can typically store around 200 hours of clearly intelligible speech in 640 MB.[43]

Lossless audio compression produces a representation of digital data that can be decoded to an exact digital duplicate of the original. Compression ratios are around 50–60% of the original size,[44] which is similar to those for generic lossless data compression. Lossless codecs use curve fitting or linear prediction as a basis for estimating the signal. Parameters describing the estimation and the difference between the estimation and the actual signal are coded separately.[45]

A number of lossless audio compression formats exist. See list of lossless codecs for a listing. Some formats are associated with a distinct system, such as Direct Stream Transfer, used in Super Audio CD and Meridian Lossless Packing, used in DVD-Audio, Dolby TrueHD, Blu-ray and HD DVD.

Some audio file formats feature a combination of a lossy format and a lossless correction; this allows stripping the correction to easily obtain a lossy file. Such formats include MPEG-4 SLS (Scalable to Lossless), WavPack, and OptimFROG DualStream.

When audio files are to be processed, either by further compression or for editing, it is desirable to work from an unchanged original (uncompressed or losslessly compressed). Processing of a lossily compressed file for some purpose usually produces a final result inferior to the creation of the same compressed file from an uncompressed original. In addition to sound editing or mixing, lossless audio compression is often used for archival storage, or as master copies.

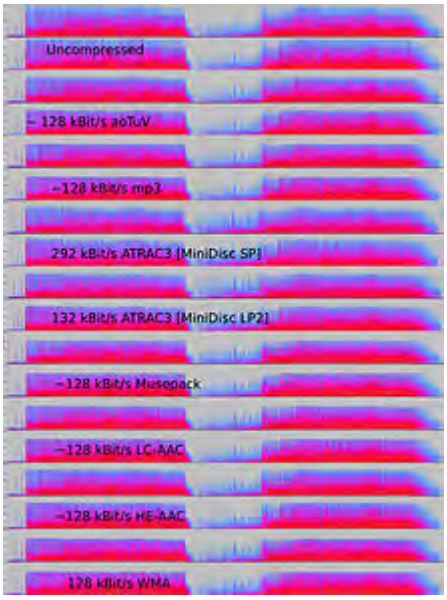
Lossy audio compression

Lossy audio compression is used in a wide range of applications. In addition to standalone audio-only applications of file playback in MP3 players or computers, digitally compressed audio streams are used in most video DVDs, digital television, streaming media on the Internet, satellite and cable radio, and increasingly in terrestrial radio broadcasts. Lossy compression typically achieves far greater compression than lossless compression, by discarding less-critical data based on psychoacoustic optimizations.[46]

Psychoacoustics recognizes that not all data in an audio stream can be perceived by the human auditory system. Most lossy compression reduces redundancy by first identifying perceptually irrelevant sounds, that is, sounds that are very hard to hear. Typical examples include high frequencies or sounds that occur at the same time as louder sounds. Those irrelevant sounds are coded with decreased accuracy or not at all.

Due to the nature of lossy algorithms, audio quality suffers a digital generation loss when a file is decompressed and recompressed. This makes lossy compression unsuitable for storing the intermediate results in professional audio engineering applications, such as sound editing and

multitrack recording. However, lossy formats such as MP3 are very popular with end-users as the file size is reduced to 5-20% of the original size and a megabyte can store about a minute's worth of music at adequate quality.



Comparison of spectrograms of audio in an uncompressed format and several lossy formats. The lossy spectrograms show bandlimiting of higher frequencies, a common technique associated with lossy audio compression.

Coding methods

To determine what information in an audio signal is perceptually irrelevant, most lossy compression algorithms use transforms such as the modified discrete cosine transform (MDCT) to convert time domain sampled waveforms into a transform domain, typically the frequency domain. Once transformed, component frequencies can be prioritized according to how audible they are. Audibility of spectral components is assessed using the absolute threshold of hearing and the principles of simultaneous masking—the phenomenon wherein a signal is masked by another signal separated by frequency—and, in some cases, temporal masking—where a signal is masked by another signal separated by time. Equal-loudness contours may also be used to weigh the perceptual importance of components. Models of the human ear-brain combination incorporating such effects are often called psychoacoustic models.[47]

Other types of lossy compressors, such as the linear predictive coding (LPC) used with speech, are source-based coders. LPC uses a model of the human vocal tract to analyze speech sounds and infer the parameters used by the model to produce them moment to moment. These changing parameters are transmitted or stored and used to drive another model in the decoder which reproduces the sound.

Lossy formats are often used for the distribution of streaming audio or interactive communication (such as in cell phone networks). In such applications, the data must be decompressed as the data flows, rather than after the entire data stream has been transmitted. Not all audio codecs can be used for streaming applications.[46]

Latency is introduced by the methods used to encode and decode the data. Some codecs will analyze a longer segment, called a *frame*, of the data to optimize efficiency, and then code it in a manner that requires a larger segment of data at one time to decode. The inherent latency of the coding algorithm can be critical; for example, when there is a two-way transmission of data, such as with a telephone conversation, significant delays may seriously degrade the perceived quality.

In contrast to the speed of compression, which is proportional to the number of operations required by the algorithm, here latency refers to the number of samples that must be analyzed before a block of audio is processed. In the minimum case, latency is zero samples (e.g., if the coder/decoder simply reduces the number of bits used to quantize the signal). Time domain algorithms such as LPC also often have low latencies, hence their popularity in speech coding for telephony. In algorithms such as MP3, however, a large number of samples have to be analyzed to implement a psychoacoustic model in the frequency domain, and latency is on the order of 23 ms.

Speech encoding

Speech encoding is an important category of audio data compression. The perceptual models used to estimate what aspects of speech a human ear can hear are generally somewhat different from those used for music. The range of frequencies needed to convey the sounds of a human voice is normally far narrower than that needed for music, and the sound is normally less complex. As a result, speech can be encoded at high quality using a relatively low bit rate.

This is accomplished, in general, by some combination of two approaches:

- Only encoding sounds that could be made by a single human voice.
- Throwing away more of the data in the signal—keeping just enough to reconstruct an "intelligible" voice rather than the full frequency range of human hearing.

The earliest algorithms used in speech encoding (and audio data compression in general) were the A-law algorithm and the μ-law algorithm.

History



Solidyne 922: The world's first commercial audio bit compression sound card for PC, 1990

Early audio research was conducted at Bell Labs. There, in 1950, C. Chapin Cutler filed the patent on differential pulse-code modulation (DPCM).[48] In 1973, Adaptive DPCM (ADPCM) was introduced by P. Cummiskey, Nikil S. Jayant and James L. Flanagan.[49][50]

Perceptual coding was first used for speech coding compression, with linear predictive coding (LPC).[51] Initial concepts for LPC date back to the work of Fumitada Itakura (Nagoya University) and Shuzo Saito (Nippon Telegraph and Telephone) in 1966.[52] During the 1970s, Bishnu S. Atal and Manfred R. Schroeder at Bell Labs developed a form of LPC called adaptive predictive coding (APC), a perceptual coding algorithm that exploited the masking properties of the human ear, followed in the early 1980s with the code-excited linear prediction (CELP) algorithm which achieved a significant compression ratio for its time.[51] Perceptual coding is used by modern audio compression formats such as MP3[51] and AAC.

Discrete cosine transform (DCT), developed by Nasir Ahmed, T. Natarajan and K. R. Rao in 1974,[16] provided the basis for the modified discrete cosine transform (MDCT) used by modern audio compression formats such as MP3,[53] Dolby Digital,[54][55] and AAC.[56] MDCT was proposed by J. P. Princen, A. W. Johnson and A. B. Bradley in 1987,[57] following earlier work by Princen and Bradley in 1986.[58]

The world's first commercial broadcast automation audio compression system was developed by Oscar Bonello, an engineering professor at the University of Buenos Aires.[59] In 1983, using the psychoacoustic principle of the masking of critical bands first published in 1967,[60] he started developing a practical application based on the recently developed IBM PC computer, and the broadcast automation system was launched in 1987 under the name Audicom. Twenty years later, almost all the radio stations in the world were using similar technology manufactured by a number of companies.

A literature compendium for a large variety of audio coding systems was published in the IEEE's *Journal on Selected Areas in Communications* (*JSAC*), in February 1988. While there were some papers from before that time, this collection documented an entire variety of finished, working audio coders, nearly all of them using perceptual techniques and some kind of frequency analysis and back-end noiseless coding.[61]

Video

Uncompressed video requires a very high data rate. Although lossless video compression codecs perform at a compression factor of 5 to 12, a typical H.264 lossy compression video has a compression factor between 20 and 200.[62]

The two key video compression techniques used in video coding standards are the DCT and motion compensation (MC). Most video coding standards, such as the H.26x and MPEG formats, typically use motion-compensated DCT video coding (block motion compensation).[63][64]

Most video codecs are used alongside audio compression techniques to store the separate but complementary data streams as one combined package using so-called *container formats*.[65]

Encoding theory

Video data may be represented as a series of still image frames. Such data usually contains abundant amounts of spatial and temporal redundancy. Video compression algorithms attempt to reduce redundancy and store information more compactly.

Most video compression formats and codecs exploit both spatial and temporal redundancy (e.g. through difference coding with motion compensation). Similarities can be encoded by only storing differences between e.g. temporally adjacent frames (inter-frame coding) or spatially adjacent pixels (intra-frame coding). Inter-frame compression (a temporal delta encoding) (re)uses data from one or more earlier or later frames in a sequence to describe the current frame. Intra-frame coding, on the other hand, uses only data from within the current frame, effectively being still-image compression.[47]

The intra-frame video coding formats used in camcorders and video editing employ simpler compression that uses only intra-frame prediction. This simplifies video editing software, as it prevents a situation in which a compressed frame refers to data that the editor has deleted.

Usually, video compression additionally employs lossy compression techniques like quantization that reduce aspects of the source data that are (more or less) irrelevant to the human visual perception by exploiting perceptual features of human vision. For example, small differences in color are more difficult to perceive than are changes in brightness. Compression algorithms can average a color across these similar areas in a manner similar to those used in JPEG image compression.[9] As in all lossy compression, there is a trade-off between video quality and bit rate, cost of processing the compression and decompression, and system requirements. Highly compressed video may present visible or distracting artifacts.

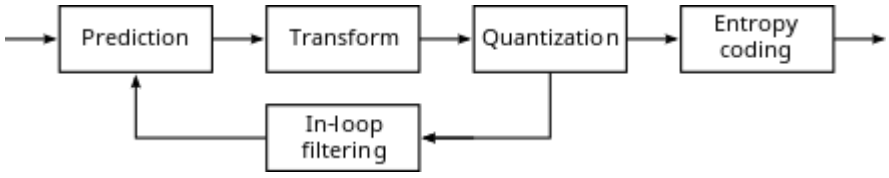
Other methods other than the prevalent DCT-based transform formats, such as fractal compression, matching pursuit and the use of a discrete wavelet transform (DWT), have been the subject of some research, but are typically not used in practical products. Wavelet compression is used in still-image

coders and video coders without motion compensation. Interest in fractal compression seems to be waning, due to recent theoretical analysis showing a comparative lack of effectiveness of such methods.[47]

Inter-frame coding

In inter-frame coding, individual frames of a video sequence are compared from one frame to the next, and the video compression codec records the differences to the reference frame. If the frame contains areas where nothing has moved, the system can simply issue a short command that copies that part of the previous frame into the next one. If sections of the frame move in a simple manner, the compressor can emit a (slightly longer) command that tells the decompressor to shift, rotate, lighten, or darken the copy. This longer command still remains much shorter than data generated by intra-frame compression. Usually, the encoder will also transmit a residue signal which describes the remaining more subtle differences to the reference imagery. Using entropy coding, these residue signals have a more compact representation than the full signal. In areas of video with more motion, the compression must encode more data to keep up with the larger number of pixels that are changing. Commonly during explosions, flames, flocks of animals, and in some panning shots, the high-frequency detail leads to quality decreases or to increases in the variable bitrate.

Hybrid block-based transform formats



Processing stages of a typical video encoder

Today, nearly all commonly used video compression methods (e.g., those in standards approved by the ITU-T or ISO) share the same basic architecture that dates back to H.261 which was standardized in 1988 by the ITU-T. They mostly rely on the DCT, applied to rectangular blocks of neighboring pixels, and temporal prediction using motion vectors, as well as nowadays also an in-loop filtering step.

In the prediction stage, various deduplication and difference-coding techniques are applied that help decorrelate data and describe new data based on already transmitted data.

Then rectangular blocks of remaining pixel data are transformed to the frequency domain. In the main lossy processing stage, frequency domain data gets quantized in order to reduce information that is irrelevant to human visual perception.

In the last stage statistical redundancy gets largely eliminated by an entropy coder which often applies some form of arithmetic coding.

In an additional in-loop filtering stage various filters can be applied to the reconstructed image signal. By computing these filters also inside the encoding loop they can help compression because they can be applied to reference material before it gets used in the prediction process and they can be guided using the original signal. The most popular example are deblocking filters that blur out blocking artifacts from quantization discontinuities at transform block boundaries.

History

In 1967, A.H. Robinson and C. Cherry proposed a run-length encoding bandwidth compression scheme for the transmission of analog television signals.[66] The DCT, which is fundamental to modern video compression,[67] was introduced by Nasir Ahmed, T. Natarajan and K. R. Rao in 1974.[16][68]

H.261, which debuted in 1988, commercially introduced the prevalent basic architecture of video compression technology.[69] It was the first video coding format based on DCT compression.[67] H.261 was developed by a number of companies, including Hitachi, PictureTel, NTT, BT and Toshiba.[70]

The most popular video coding standards used for codecs have been the MPEG standards. MPEG-1 was developed by the Motion Picture Experts Group (MPEG) in 1991, and it was designed to compress VHS-quality video. It was succeeded in 1994 by MPEG-2/H.262,[69] which was developed by a number of companies, primarily Sony, Thomson and Mitsubishi Electric.[71] MPEG-2 became the standard video format for DVD and SD digital television.[69] In 1999, it was followed by MPEG-4/H.263.[69] It was also developed by a number of companies, primarily Mitsubishi Electric, Hitachi and Panasonic.[72]

H.264/MPEG-4 AVC was developed in 2003 by a number of organizations, primarily Panasonic, Godo Kaisha IP Bridge and LG Electronics.[73] AVC commercially introduced the modern context-adaptive binary arithmetic coding (CABAC) and context-adaptive variable-length coding (CAVLC) algorithms. AVC is the main video encoding standard for Blu-ray Discs, and is widely used by video sharing websites and streaming internet services such as YouTube, Netflix, Vimeo, and iTunes Store, web software such as Adobe Flash Player and Microsoft Silverlight, and various HDTV broadcasts over terrestrial and satellite television.

Genetics

Genetics compression algorithms are the latest generation of lossless algorithms that compress data (typically sequences of nucleotides) using both conventional compression algorithms and genetic algorithms adapted to the specific datatype. In 2012, a team of scientists from Johns Hopkins University published a genetic compression algorithm that does not use a reference genome for compression. HAPZIPPER was tailored for HapMap data and achieves over 20-fold compression (95% reduction in file size), providing 2- to 4-fold better compression and is less computationally intensive than the leading general-purpose compression utilities. For this, Chanda, Elhaik, and Bader introduced MAF-based encoding (MAFE), which reduces the heterogeneity of the dataset by sorting SNPs by their minor allele frequency, thus homogenizing the dataset.[74] Other algorithms developed in 2009 and 2013 (DNAZip and GenomeZip) have compression ratios of up to 1200-fold—allowing 6 billion basepair diploid human genomes to be stored in 2.5 megabytes (relative to a reference genome or averaged over many genomes).[75][76] For a benchmark in genetics/genomics data compressors, see [77]

Outlook and currently unused potential

It is estimated that the total amount of data that is stored on the world's storage devices could be further compressed with existing compression algorithms by a remaining average factor of 4.5:1.[78] It is estimated that the combined technological capacity of the world to store information provides 1,300 exabytes of hardware digits in 2007, but when the corresponding content is optimally compressed, this only represents 295 exabytes of Shannon information.[79]

See also

- HTTP compression
- Kolmogorov complexity
- Lhasa (computing)
- Minimum description length
- Modulo-N code
- Motion coding
- Range coding
- Set redundancy compression
- Sub-band coding
- Universal code (data compression)
- Vector quantization

References

1. Wade, Graham (1994). *Signal coding and processing* (https://books.google.com/books?id=CJswCy7_W8YC) (2 ed.). Cambridge University Press. p. 34. ISBN 978-0-521-42336-6. Retrieved 2011-12-22. "The broad objective of source coding is to exploit or remove 'inefficient' redundancy in the PCM source and thereby achieve a reduction in the overall source rate R."
2. Mahdi, O.A.; Mohammed, M.A.; Mohamed, A.J. (November 2012). "Implementing a Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique" (http://ijcsi.org/papers/IJCSI-9-6-3-53-59.pdf) (PDF). *International Journal of Computer Science Issues*. **9** (6, No. 3): 53–59. Archived (https://web.archive.org/web/20130320103924/http://ijcsi.org/papers/IJCSI-9-6-3-53-59.pdf) (PDF) from the original on 2013-03-20. Retrieved 6 March 2013.
3. Pujar, J.H.; Kadlaskar, L.M. (May 2010). "A New Lossless Method of Image Compression and Decompression Using Huffman Coding Techniques" (http://www.jatit.org/volumes/research-papers/Vol15No1/3Vol15No1.pdf) (PDF). *Journal of Theoretical and Applied Information Technology*. **15** (1): 18–23. Archived (https://web.archive.org/web/20100524105217/http://www.jatit.org/volumes/research-papers/Vol15No1/3Vol15No1.pdf) (PDF) from the original on 2010-05-24.
4. Salomon, David (2008). *A Concise Introduction to Data Compression*. Berlin: Springer. ISBN 9781848000728.
5. Tank, M.K. (2011). "Implementation of Lempel-ZIV algorithm for lossless compression using VHDL". *Implementation of Limpel-Ziv algorithm for lossless compression using VHDL*. *Thinkquest 2010: Proceedings of the First International Conference on Contours of Computing Technology*. Berlin: Springer. pp. 275–283. doi:10.1007/978-81-8489-989-4_51 (https://doi.org/10.1007%2F978-81-8489-989-4_51). ISBN 978-81-8489-988-7.
6. Navqi, Saud; Naqvi, R.; Riaz, R.A.; Siddiqui, F. (April 2011). "Optimized RTL design and implementation of LZW algorithm for high bandwidth applications" (http://pe.org.pl/articles/2011/4/68.pdf) (PDF). *Electrical Review*. **2011** (4): 279–285. Archived (https://web.archive.org/web/20130520105146/http://pe.org.pl/articles/2011/4/68.pdf) (PDF) from the original on 2013-05-20.
7. Stephen, Wolfram (2002). *New Kind of Science* (https://www.wolframscience.com/nks/p1069--data-compression/). Champaign, IL. p. 1069. ISBN 1-57955-008-8.

8. Mahmud, Salauddin (March 2012). "An Improved Data Compression Method for General Data" (http://www.ijser.org/researchpaper%5CAn-Improved-Data-Compression-Method-for-General-Data.pdf) (PDF). *International Journal of Scientific & Engineering Research*. **3** (3): 2. Archived (https://web.archive.org/web/20131102022116/http://www.ijser.org/researchpaper%5CAn-Improved-Data-Compression-Method-for-General-Data.pdf) (PDF) from the original on 2013-11-02. Retrieved 6 March 2013.
9. Lane, Tom. "JPEG Image Compression FAQ, Part 1" (http://www.faqs.org/faqs/jpeg-faq/part1/). *Internet FAQ Archives*. Independent JPEG Group. Retrieved 6 March 2013.
10. G. J. Sullivan; J.-R. Ohm; W.-J. Han; T. Wiegand (December 2012). "Overview of the High Efficiency Video Coding (HEVC) Standard" (https://doi.org/10.1109%2FTCSVT.2012.2221191). *IEEE Transactions on Circuits and Systems for Video Technology*. IEEE. **22** (12): 1649–1668. doi:10.1109/TCSVT.2012.2221191 (https://doi.org/10.1109%2FTCSVT.2012.2221191).
11. "How to choose optimal archiving settings – WinRAR" (https://www.winrar-france.fr/winrar_instructions_for_use/source/html/HELPArcOptimal.htm).
12. "(Set compression Method) switch – 7zip" (https://sevenzip.osdn.jp/chm/cmdline/switches/method.htm).
13. Wolfram, Stephen (2002). *A New Kind of Science* (https://archive.org/details/newkindofscience00wolf/page/1069). Wolfram Media, Inc. p. 1069 (https://archive.org/details/newkindofscience00wolf/page/1069). ISBN 978-1-57955-008-0.
14. Arcangel, Cory. "On Compression" (http://www.coryarcangel.com/downloads/Cory-Arcangel-OnC.pdf) (PDF). Archived (https://web.archive.org/web/20130728082920/http://www.coryarcangel.com/downloads/Cory-Arcangel-OnC.pdf) (PDF) from the original on 2013-07-28. Retrieved 6 March 2013.
15. Ahmed, Nasir (January 1991). "How I Came Up With the Discrete Cosine Transform" (https://www.scribd.com/doc/52879771/DCT-History-How-I-Came-Up-with-the-Discrete-Cosine-Transform). *Digital Signal Processing*. **1** (1): 4–5. doi:10.1016/1051-2004(91)90086-Z (https://doi.org/10.1016%2F1051-2004%2891%2990086-Z).
16. Nasir Ahmed; T. Natarajan; Kamisetty Ramamohan Rao (January 1974). "Discrete Cosine Transform" (https://www.ic.tu-berlin.de/fileadmin/fg121/Source-Coding_WS12/selected-readings/Ahmed_et_al.__1974.pdf) (PDF). *IEEE Transactions on Computers*. **C-23** (1): 90–93. doi:10.1109/T-C.1974.223784 (https://doi.org/10.1109%2FT-C.1974.223784). S2CID 149806273 (https://api.semanticscholar.org/CorpusID:149806273). Archived (https://web.archive.org/web/20161208075733/https://www.ic.tu-berlin.de/fileadmin/fg121/Source-Coding_WS12/selected-readings/Ahmed_et_al.__1974.pdf) (PDF) from the original on 2016-12-08.
17. CCITT Study Group VIII und die Joint Photographic Experts Group (JPEG) von ISO/IEC Joint Technical Committee 1/Subcommittee 29/Working Group 10 (1993), "Annex D – Arithmetic coding", *Recommendation T.81: Digital Compression and Coding of Continuous-tone Still images – Requirements and guidelines* (https://www.w3.org/Graphics/JPEG/itu-t81.pdf) (PDF), pp. 54 ff, retrieved 2009-11-07
18. Marak, Laszlo. "On image compression" (https://web.archive.org/web/20150528012028/http://www.ujoimro.com/resources/Laszlo_Marak_image_compression.pdf) (PDF). University of Marne la Vallee. Archived from the original (http://www.ujoimro.com/resources/Laszlo_Marak_image_compression.pdf) (PDF) on 28 May 2015. Retrieved 6 March 2013.
19. Mahoney, Matt. "Rationale for a Large Text Compression Benchmark" (http://cs.fit.edu/~mmahoney/compression/rationale.html). Florida Institute of Technology. Retrieved 5 March 2013.

20. Shmilovici A.; Kahiri Y.; Ben-Gal I.; Hauser S. (2009). "Measuring the Efficiency of the Intraday Forex Market with a Universal Data Compression Algorithm" (http://www.eng.tau.ac.il/~bengal/28.pdf) (PDF). *Computational Economics*. **33** (2): 131–154. CiteSeerX 10.1.1.627.3751 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.627.3751). doi:10.1007/s10614-008-9153-3 (https://doi.org/10.1007%2Fs10614-008-9153-3). S2CID 17234503 (https://api.semanticscholar.org/CorpusID:17234503). Archived (https://web.archive.org/web/20090709143601/http://www.eng.tau.ac.il/~bengal/28.pdf) (PDF) from the original on 2009-07-09.
21. I. Ben-Gal (2008). "On the Use of Data Compression Measures to Analyze Robust Designs" (http://www.eng.tau.ac.il/~bengal/Journal%20Paper.pdf) (PDF). *IEEE Transactions on Reliability*. **54** (3): 381–388. doi:10.1109/TR.2005.853280 (https://doi.org/10.1109%2FTR.2005.853280). S2CID 9376086 (https://api.semanticscholar.org/CorpusID:9376086).
22. D. Scully; Carla E. Brodley (2006). "Compression and machine learning: A new perspective on feature space vectors". *Data Compression Conference, 2006*: 332. doi:10.1109/DCC.2006.13 (https://doi.org/10.1109%2FDCC.2006.13). ISBN 0-7695-2545-8. S2CID 12311412 (https://api.semanticscholar.org/CorpusID:12311412).
23. Gary Adcock (January 5, 2023). "What Is AI Video Compression?" (https://massive.io/file-transfer/what-is-ai-video-compression/). *massive.io*. Retrieved 6 April 2023.
24. Gilad David Maayan (Nov 24, 2021). "AI-Based Image Compression: The State of the Art" (https://towardsdatascience.com/ai-based-image-compression-the-state-of-the-art-fb5aa6042bfa). *Towards Data Science*. Retrieved 6 April 2023.
25. Korn, D.; et al. "RFC 3284: The VCDIFF Generic Differencing and Compression Data Format" (http://tools.ietf.org/html/rfc3284). Internet Engineering Task Force. Retrieved 5 March 2013.
26. Korn, D.G.; Vo, K.P. (1995). B. Krishnamurthy (ed.). *Vdelta: Differencing and Compression*. Practical Reusable Unix Software. New York: John Wiley & Sons, Inc.
27. Claude Elwood Shannon (1948). Alcatel-Lucent (ed.). "A Mathematical Theory of Communication" (http://www.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf) (PDF). *Bell System Technical Journal*. **27** (3–4): 379–423, 623–656. doi:10.1002/j.1538-7305.1948.tb01338.x (https://doi.org/10.1002%2Fj.1538-7305.1948.tb01338.x). hdl:11858/00-001M-0000-002C-4314-2 (https://hdl.handle.net/11858%2F00-001M-0000-002C-4314-2). Archived (https://web.archive.org/web/20110524064232/http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf) (PDF) from the original on 2011-05-24. Retrieved 2019-04-21.
28. David Albert Huffman (September 1952), "A method for the construction of minimum-redundancy codes" (http://compression.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf) (PDF), *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, doi:10.1109/JRPROC.1952.273898 (https://doi.org/10.1109%2FJRPROC.1952.273898), archived (https://web.archive.org/web/20051008115257/http://compression.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf) (PDF) from the original on 2005-10-08
29. Pratt, W.K.; Kane, J.; Andrews, H.C. (1969). "Hadamard transform image coding". *Proceedings of the IEEE*. **57**: 58–68. doi:10.1109/PROC.1969.6869 (https://doi.org/10.1109%2FPROC.1969.6869).
30. "T.81 – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES" (https://www.w3.org/Graphics/JPEG/itu-t81.pdf) (PDF). CCITT. September 1992. Retrieved 12 July 2019.
31. "The JPEG image format explained" (https://web.archive.org/web/20190805194553/https://home.bt.com/tech-gadgets/photography/what-is-a-jpeg-11364206889349). *BT.com*. BT Group. 31 May 2018. Archived from the original (https://home.bt.com/tech-gadgets/photography/what-is-a-jpeg-11364206889349) on 5 August 2019. Retrieved 5 August 2019.
32. Baraniuk, Chris (15 October 2015). "Copy protections could come to JPEGs" (https://www.bbc.co.uk/news/technology-34538705). *BBC News*. BBC. Retrieved 13 September 2019.

33. "What Is a JPEG? The Invisible Object You See Every Day" (https://www.theatlantic.com/technology/archive/2013/09/what-is-a-jpeg-the-invisible-object-you-see-every-day/279954/). *The Atlantic*. 24 September 2013. Retrieved 13 September 2019.
34. "The GIF Controversy: A Software Developer's Perspective" (https://mike.pub/19950127-gif-lzw). Retrieved 26 May 2015.
35. L. Peter Deutsch (May 1996). *DEFLATE Compressed Data Format Specification version 1.3* (https://datatracker.ietf.org/doc/html/rfc1951#section-Abstract). IETF. p. 1. sec. Abstract. doi:10.17487/RFC1951 (https://doi.org/10.17487%2FRFC1951). RFC 1951 (https://datatracker.ietf.org/doc/html/rfc1951). Retrieved 2014-04-23.
36. Hoffman, Roy (2012). *Data Compression in Digital Systems* (https://books.google.com/books?id=FOfTBwAAQBAJ). Springer Science & Business Media. p. 124. ISBN 9781461560319. "Basically, wavelet coding is a variant on DCT-based transform coding that reduces or eliminates some of its limitations. (...) Another advantage is that rather than working with 8 × 8 blocks of pixels, as do JPEG and other block-based DCT techniques, wavelet coding can simultaneously compress the entire image."
37. Taubman, David; Marcellin, Michael (2012). *JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards and Practice* (https://books.google.com/books?id=y7HeBwAAQBAJ&pg=PA402). Springer Science & Business Media. ISBN 9781461507994.
38. Unser, M.; Blu, T. (2003). "Mathematical properties of the JPEG2000 wavelet filters" (http://infoscience.epfl.ch/record/63104). *IEEE Transactions on Image Processing*. **12** (9): 1080–1090. Bibcode:2003ITIP...12.1080U (https://ui.adsabs.harvard.edu/abs/2003ITIP...12.1080U). doi:10.1109/TIP.2003.812329 (https://doi.org/10.1109%2FTIP.2003.812329). PMID 18237979 (https://pubmed.ncbi.nlm.nih.gov/18237979). S2CID 2765169 (https://api.semanticscholar.org/CorpusID:2765169).
39. Sullivan, Gary (8–12 December 2003). "General characteristics and design considerations for temporal subband video coding" (https://www.itu.int/wftp3/av-arch/video-site/0312_Wai/VCEG-U06.doc). *ITU-T*. Video Coding Experts Group. Retrieved 13 September 2019.
40. Bovik, Alan C. (2009). *The Essential Guide to Video Processing* (https://books.google.com/books?id=wXmSPPB_c_0C&pg=PA355). Academic Press. p. 355. ISBN 9780080922508.
41. Swartz, Charles S. (2005). *Understanding Digital Cinema: A Professional Handbook* (https://books.google.com/books?id=tYw3ehoBnjkC&pg=PA147). Taylor & Francis. p. 147. ISBN 9780240806174.
42. Cunningham, Stuart; McGregor, Iain (2019). "Subjective Evaluation of Music Compressed with the ACER Codec Compared to AAC, MP3, and Uncompressed PCM" (https://doi.org/10.1155%2F2019%2F8265301). *International Journal of Digital Multimedia Broadcasting*. **2019**: 1–16. doi:10.1155/2019/8265301 (https://doi.org/10.1155%2F2019%2F8265301).
43. The Olympus WS-120 digital speech recorder, according to its manual, can store about 178 hours of speech-quality audio in .WMA format in 500 MB of flash memory.
44. Coalson, Josh. "FLAC Comparison" (https://xiph.org/flac/comparison.html). Retrieved 2020-08-23.
45. "Format overview" (https://xiph.org/flac/documentation_format_overview.html). Retrieved 2020-08-23.
46. Jaiswal, R.C. (2009). *Audio-Video Engineering*. Pune, Maharashtra: Nirali Prakashan. p. 3.41. ISBN 9788190639675.
47. Faxin Yu; Hao Luo; Zheming Lu (2010). *Three-Dimensional Model Analysis and Processing* (https://archive.org/details/threedimensional00yufa). Berlin: Springer. p. 47 (https://archive.org/details/threedimensional00yufa/page/n62). ISBN 9783642126512.
48. US patent 2605361 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=US2605361), C. Chapin Cutler, "Differential Quantization of Communication Signals", issued 1952-07-29

49. Cummiskey, P.; Jayant, N. S.; Flanagan, J. L. (1973). "Adaptive Quantization in Differential PCM Coding of Speech". *Bell System Technical Journal*. **52** (7): 1105–1118. doi:10.1002/j.1538-7305.1973.tb02007.x (https://doi.org/10.1002%2Fj.1538-7305.1973.tb02007.x).
50. Cummiskey, P.; Jayant, Nikil S.; Flanagan, J. L. (1973). "Adaptive quantization in differential PCM coding of speech". *The Bell System Technical Journal*. **52** (7): 1105–1118. doi:10.1002/j.1538-7305.1973.tb02007.x (https://doi.org/10.1002%2Fj.1538-7305.1973.tb02007.x). ISSN 0005-8580 (https://www.worldcat.org/issn/0005-8580).
51. Schroeder, Manfred R. (2014). "Bell Laboratories" (https://books.google.com/books?id=d9IkBAAAQBAJ&pg=PA388). *Acoustics, Information, and Communication: Memorial Volume in Honor of Manfred R. Schroeder*. Springer. p. 388. ISBN 9783319056609.
52. Gray, Robert M. (2010). "A History of Realtime Digital Speech on Packet Networks: Part II of Linear Predictive Coding and the Internet Protocol" (https://ee.stanford.edu/~gray/lpcip.pdf) (PDF). *Found. Trends Signal Process*. **3** (4): 203–303. doi:10.1561/2000000036 (https://doi.org/10.1561%2F2000000036). ISSN 1932-8346 (https://www.worldcat.org/issn/1932-8346). Archived (https://web.archive.org/web/20100704113551/http://ee.stanford.edu/~gray/lpcip.pdf) (PDF) from the original on 2010-07-04.
53. Guckert, John (Spring 2012). "The Use of FFT and MDCT in MP3 Audio Compression" (http://www.math.utah.edu/~gustafso/s2012/2270/web-projects/Guckert-audio-compression-svd-mdct-MP3.pdf) (PDF). *University of Utah*. Archived (https://web.archive.org/web/20140124152337/http://www.math.utah.edu/~gustafso/s2012/2270/web-projects/Guckert-audio-compression-svd-mdct-MP3.pdf) (PDF) from the original on 2014-01-24. Retrieved 14 July 2019.
54. Luo, Fa-Long (2008). *Mobile Multimedia Broadcasting Standards: Technology and Practice* (https://books.google.com/books?id=l6PovWat8SMC&pg=PA590). Springer Science & Business Media. p. 590. ISBN 9780387782638.
55. Britanak, V. (2011). "On Properties, Relations, and Simplified Implementation of Filter Banks in the Dolby Digital (Plus) AC-3 Audio Coding Standards". *IEEE Transactions on Audio, Speech, and Language Processing*. **19** (5): 1231–1241. doi:10.1109/TASL.2010.2087755 (https://doi.org/10.1109%2FTASL.2010.2087755). S2CID 897622 (https://api.semanticscholar.org/CorpusID:897622).
56. Brandenburg, Karlheinz (1999). "MP3 and AAC Explained" (http://graphics.ethz.ch/teaching/mmcom12/slides/mp3_and_aac_brandenburg.pdf) (PDF). Archived (https://web.archive.org/web/20170213191747/https://graphics.ethz.ch/teaching/mmcom12/slides/mp3_and_aac_brandenburg.pdf) (PDF) from the original on 2017-02-13.
57. Princen, J.; Johnson, A.; Bradley, A. (1987). "Subband/Transform coding using filter bank designs based on time domain aliasing cancellation". *ICASSP '87. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 12. pp. 2161–2164. doi:10.1109/ICASSP.1987.1169405 (https://doi.org/10.1109%2FICASSP.1987.1169405). S2CID 58446992 (https://api.semanticscholar.org/CorpusID:58446992).
58. Princen, J.; Bradley, A. (1986). "Analysis/Synthesis filter bank design based on time domain aliasing cancellation". *IEEE Transactions on Acoustics, Speech, and Signal Processing*. **34** (5): 1153–1161. doi:10.1109/TASSP.1986.1164954 (https://doi.org/10.1109%2FTASSP.1986.1164954).
59. "Summary of some of Solidyne's contributions to Broadcast Engineering" (https://web.archive.org/web/20130308063719/http://www.solidynepro.com/indexahtmlp_Hist-ENG%2Ct.htm). *Brief History of Solidyne*. Buenos Aires: Solidyne. Archived from the original (http://www.solidynepro.com/indexahtmlp_Hist-ENG,t.htm) on 8 March 2013. Retrieved 6 March 2013.
60. Zwicker, Eberhard; et al. (1967). *The Ear As A Communication Receiver* (https://web.archive.org/web/20000914080525/http://asa.aip.org/books/ear.html). Melville, NY: Acoustical Society of America. Archived from the original (http://asa.aip.org/books/ear.html) on 2000-09-14. Retrieved 2011-11-11.

61. "File Compression Possibilities" (https://www.gadgetcouncil.com/compress-1GB-files-into-10-mb/). *A Brief guide to compress a file in 4 different ways*. 17 February 2017.
62. Dmitriy Vatolin; et al. (Graphics & Media Lab Video Group) (March 2007). Lossless Video Codecs Comparison '2007 (http://compression.ru/video/codec_comparison/pdf/msu_lossless_codecs_comparison_2007_eng.pdf) (PDF) (Report). Moscow State University. Archived (https://web.archive.org/web/20080515091507/http://www.compression.ru/video/codec_comparison/pdf/msu_lossless_codecs_comparison_2007_eng.pdf) (PDF) from the original on 2008-05-15.
63. Chen, Jie; Koc, Ut-Va; Liu, KJ Ray (2001). *Design of Digital Video Coding Systems: A Complete Compressed Domain Approach* (https://books.google.com/books?id=LUzFKU3HeegC&pg=PA71). CRC Press. p. 71. ISBN 9780203904183.
64. Li, Jian Ping (2006). *Proceedings of the International Computer Conference 2006 on Wavelet Active Media Technology and Information Processing: Chongqing, China, 29-31 August 2006* (https://books.google.com/books?id=FZiK3zXdK7sC&pg=PA847). World Scientific. p. 847. ISBN 9789812709998.
65. "Video Coding" (https://web.archive.org/web/20130523194345/http://csip.ece.gatech.edu/drupal7/?q=technical-area%2Fvideo-coding). *CSIP website*. Center for Signal and Information Processing, Georgia Institute of Technology. Archived from the original (http://csip.ece.gatech.edu/drupal7/?q=technical-area/video-coding) on 23 May 2013. Retrieved 6 March 2013.
66. Robinson, A. H.; Cherry, C. (1967). "Results of a prototype television bandwidth compression scheme". *Proceedings of the IEEE*. IEEE. **55** (3): 356–364. doi:10.1109/PROC.1967.5493 (https://doi.org/10.1109%2FPROC.1967.5493).
67. Ghanbari, Mohammed (2003). *Standard Codecs: Image Compression to Advanced Video Coding* (https://books.google.com/books?id=7XuU8T3ooOAC&pg=PA1). Institution of Engineering and Technology. pp. 1–2. ISBN 9780852967102.
68. Reader, Cliff (2016-08-31). "Patent landscape for royalty-free video coding" (https://web.archive.org/web/20161208075738/https://www.youtube.com/watch?v=wi1BefrfTos). In Tescher, Andrew G (ed.). *Applications of Digital Image Processing XXXIX*. Applications of Digital Image Processing XXXIX. Vol. 9971. San Diego, California: Society of Photo-Optical Instrumentation Engineers. pp. 99711B. Bibcode:2016SPIE.9971E..1BR (https://ui.adsabs.harvard.edu/abs/2016SPIE.9971E..1BR). doi:10.1117/12.2239493 (https://doi.org/10.1117%2F12.2239493). Archived from the original (https://www.youtube.com/watch?v=wi1BefrfTos) on 2016-12-08. Lecture recording, from 3:05:10.
69. "The History of Video File Formats Infographic — RealPlayer" (http://www.real.com/resources/digital-video-file-formats/). 22 April 2012.
70. "Patent statement declaration registered as H261-07" (https://www.itu.int/net4/ipr/details_ps.aspx?sector=ITU-T&id=H261-07). *ITU*. Retrieved 11 July 2019.
71. "MPEG-2 Patent List" (https://www.mpegla.com/wp-content/uploads/m2-att1.pdf) (PDF). *MPEG LA*. Archived (https://web.archive.org/web/20190529164140/https://www.mpegla.com/wp-content/uploads/m2-att1.pdf) (PDF) from the original on 2019-05-29. Retrieved 7 July 2019.
72. "MPEG-4 Visual - Patent List" (https://www.mpegla.com/wp-content/uploads/m4v-att1.pdf) (PDF). *MPEG LA*. Archived (https://web.archive.org/web/20190706184528/https://www.mpegla.com/wp-content/uploads/m4v-att1.pdf) (PDF) from the original on 2019-07-06. Retrieved 6 July 2019.
73. "AVC/H.264 – Patent List" (https://www.mpegla.com/wp-content/uploads/avc-att1.pdf) (PDF). *MPEG LA*. Retrieved 6 July 2019.
74. Chanda P, Bader JS, Elhaik E (27 Jul 2012). "HapZipper: sharing HapMap populations just got easier" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3488212). *Nucleic Acids Research*. **40** (20): e159. doi:10.1093/nar/gks709 (https://doi.org/10.1093%2Fnar%2Fgks709). PMC 3488212 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3488212). PMID 22844100 (https://pubmed.ncbi.nlm.nih.gov/22844100).

75. Christley S, Lu Y, Li C, Xie X (Jan 15, 2009). "Human genomes as email attachments" (https://doi.org/10.1093%2Fbioinformatics%2Fbtn582). *Bioinformatics*. **25** (2): 274–5. doi:10.1093/bioinformatics/btn582 (https://doi.org/10.1093%2Fbioinformatics%2Fbtn582). PMID 18996942 (https://pubmed.ncbi.nlm.nih.gov/18996942).
76. Pavlichin DS, Weissman T, Yona G (September 2013). "The human genome contracts again" (https://doi.org/10.1093%2Fbioinformatics%2Fbtt362). *Bioinformatics*. **29** (17): 2199–202. doi:10.1093/bioinformatics/btt362 (https://doi.org/10.1093%2Fbioinformatics%2Fbtt362). PMID 23793748 (https://pubmed.ncbi.nlm.nih.gov/23793748).
77. Hosseini, Morteza; Pratas, Diogo; Pinho, Armando (2016). "A Survey on Data Compression Methods for Biological Sequences" (https://doi.org/10.3390%2Finfo7040056). *Information*. **7** (4): 56. doi:10.3390/info7040056 (https://doi.org/10.3390%2Finfo7040056).
78. "Data Compression via Logic Synthesis" (https://infoscience.epfl.ch/record/188645/files/06742961.pdf) (PDF).
79. Hilbert, Martin; López, Priscila (1 April 2011). "The World's Technological Capacity to Store, Communicate, and Compute Information". *Science*. **332** (6025): 60–65. Bibcode:2011Sci...332...60H (https://ui.adsabs.harvard.edu/abs/2011Sci...332...60H). doi:10.1126/science.1200970 (https://doi.org/10.1126%2Fscience.1200970). PMID 21310967 (https://pubmed.ncbi.nlm.nih.gov/21310967). S2CID 206531385 (https://api.semanticscholar.org/CorpusID:206531385).

External links

- "Part 3: Video compression" (http://dvd-hq.info/data_compression_3.php), *Data Compression Basics*
- Pierre Larbier, *Using 10-bit AVC/H.264 Encoding with 4:2:2 for Broadcast Contribution* (https://web.archive.org/web/20090905092232/http://extranet.ateme.com/download.php?file=1114), Ateme, archived from the original (http://extranet.ateme.com/download.php?file=1114) on 2009-09-05
- Why does 10-bit save bandwidth (even when content is 8-bit)? (https://web.archive.org/web/20170830224011/http://extranet.ateme.com/download.php?file=1194) at the Wayback Machine (archived 2017-08-30)
- Which compression technology should be used? (https://web.archive.org/web/20170830224021/http://extranet.ateme.com/download.php?file=1196) at the Wayback Machine (archived 2017-08-30)
- *Introduction to Compression Theory* (http://media.wiley.com/product_data/excerpt/99/04705184/0470518499.pdf) (PDF), Wiley, archived (https://web.archive.org/web/20070928023157/http://media.wiley.com/product_data/excerpt/99/04705184/0470518499.pdf) (PDF) from the original on 2007-09-28
- EBU subjective listening tests on low-bitrate audio codecs (http://tech.ebu.ch/docs/tech/tech3296.pdf)
- Audio Archiving Guide: Music Formats (http://techgage.com/article/audio_archiving_guide_part_1_-_music_formats/) (Guide for helping a user pick out the right codec)
- MPEG 1&2 video compression intro (pdf format) (https://web.archive.org/web/20070928023157/http://mia.ece.uic.edu/~papers/WWW/MultimediaStandards/chapter7.pdf) at the Wayback Machine (archived September 28, 2007)
- hydrogenaudio wiki comparison (http://wiki.hydrogenaud.io/index.php?title=Lossless_comparison)
- Introduction to Data Compression (https://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf) by Guy E Blelloch from CMU
- Explanation of lossless signal compression method used by most codecs (http://www.monkeysaudio.com/theory.html)

- Videsignline – Intro to Video Compression (https://web.archive.org/web/20100315021124/http://www.videsignline.com/howto/showArticle.jhtml?articleID=185301351) at the Wayback Machine (archived 2010-03-15)
- Data Footprint Reduction Technology (https://web.archive.org/web/20130527124650/http://public.dhe.ibm.com/common/ssi/ecm/en/tsu12345usen/TSU12345USEN.PDF) at the Wayback Machine (archived 2013-05-27)
- What is Run length Coding in video compression (http://siliconmentor.blogspot.in/2014/12/what-is-run-length-coding-in-video.html)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Data_compression&oldid=1151130787"



INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

T.81

(09/92)

TERMINAL EQUIPMENT AND PROTOCOLS FOR TELEMATIC SERVICES

INFORMATION TECHNOLOGY – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES



Recommendation T.81

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Foreword

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The CCITT (the International Telegraph and Telephone Consultative Committee) is a permanent organ of the ITU. Some 166 member countries, 68 telecom operating entities, 163 scientific and industrial organizations and 39 international organizations participate in CCITT which is the body which sets world telecommunications standards (Recommendations).

The approval of Recommendations by the members of CCITT is covered by the procedure laid down in CCITT Resolution No. 2 (Melbourne, 1988). In addition, the Plenary Assembly of CCITT, which meets every four years, approves Recommendations submitted to it and establishes the study programme for the following period.

In some areas of information technology, which fall within CCITT's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC. The text of CCITT Recommendation T.81 was approved on 18th September 1992. The identical text is also published as ISO/IEC International Standard 10918-1.

CCITT NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized private operating agency.

© ITU 1993

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

Contents

		Page
Introduction		iii
1	Scope	1
2	Normative references	1
3	Definitions, abbreviations and symbols	1
4	General	12
5	Interchange format requirements	23
6	Encoder requirements	23
7	Decoder requirements	23
Annex A – Mathematical definitions		24
Annex B – Compressed data formats		31
Annex C – Huffman table specification		50
Annex D – Arithmetic coding		54
Annex E – Encoder and decoder control procedures		77
Annex F – Sequential DCT-based mode of operation		87
Annex G – Progressive DCT-based mode of operation		119
Annex H – Lossless mode of operation		132
Annex J – Hierarchical mode of operation		137
Annex K – Examples and guidelines		143
Annex L – Patents		179
Annex M – Bibliography		181

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Introduction

This CCITT Recommendation | ISO/IEC International Standard was prepared by CCITT Study Group VIII and the Joint Photographic Experts Group (JPEG) of ISO/IEC JTC 1/SC 29/WG 10. This Experts Group was formed in 1986 to establish a standard for the sequential progressive encoding of continuous tone grayscale and colour images.

Digital Compression and Coding of Continuous-tone Still images, is published in two parts:

- Requirements and guidelines;
- Compliance testing.

This part, Part 1, sets out requirements and implementation guidelines for continuous-tone still image encoding and decoding processes, and for the coded representation of compressed image data for interchange between applications. These processes and representations are intended to be generic, that is, to be applicable to a broad range of applications for colour and grayscale still images within communications and computer systems. Part 2, sets out tests for determining whether implementations comply with the requirments for the various encoding and decoding processes specified in Part 1.

The user's attention is called to the possibility that – for some of the coding processes specified herein – compliance with this Recommendation | International Standard may require use of an invention covered by patent rights. See Annex L for further information.

The requirements which these processes must satisfy to be useful for specific image communications applications such as facsimile, Videotex and audiographic conferencing are defined in CCITT Recommendation T.80. The intent is that the generic processes of Recommendation T.80 will be incorporated into the various CCITT Recommendations for terminal equipment for these applications.

In addition to the applications addressed by the CCITT and ISO/IEC, the JPEG committee has developped a compression standard to meet the needs of other applications as well, including desktop publishing, graphic arts, medical imaging and scientific imaging.

Annexes A, B, C, D, E, F, G, H and J are normative, and thus form an integral part of this Specification. Annexes K, L and M are informative and thus do not form an integral part of this Specification.

This Specification aims to follow the guidelines of CCITT and ISO/IEC JTC 1 on *Rules for presentation of CCITT | ISO/IEC common text*.

INTERNATIONAL STANDARD

CCITT RECOMMENDATION

INFORMATION TECHNOLOGY – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES

1 Scope

This CCITT Recommendation | International Standard is applicable to continuous-tone – grayscale or colour – digital still image data. It is applicable to a wide range of applications which require use of compressed images. It is not applicable to bi-level image data.

This Specification

- specifies processes for converting source image data to compressed image data;
- specifies processes for converting compressed image data to reconstructed image data;
- gives guidance on how to implement these processes in practice;
- specifies coded representations for compressed image data.

NOTE – This Specification does not specify a complete coded image representation. Such representations may include certain parameters, such as aspect ratio, component sample registration, and colour space designation, which are application-dependent.

2 Normative references

The following CCITT Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this CCITT Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this CCITT Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The CCITT Secretariat maintains a list of currently valid CCITT Recommendations.

- *CCITT Recommendation T.80 (1992), Common components for image compression and communication – Basic principles.*

3 Definitions, abbreviations and symbols

3.1 Definitions and abbreviations

For the purposes of this Specification, the following definitions apply.

3.1.1 abbreviated format: A representation of compressed image data which is missing some or all of the table specifications required for decoding, or a representation of table-specification data without frame headers, scan headers, and entropy-coded segments.

3.1.2 AC coefficient: Any DCT coefficient for which the frequency is not zero in at least one dimension.

3.1.3 (adaptive) (binary) arithmetic decoding: An entropy decoding procedure which recovers the sequence of symbols from the sequence of bits produced by the arithmetic encoder.

3.1.4 (adaptive) (binary) arithmetic encoding: An entropy encoding procedure which codes by means of a recursive subdivision of the probability of the sequence of symbols coded up to that point.

3.1.5 application environment: The standards for data representation, communication, or storage which have been established for a particular application.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

3.1.6 arithmetic decoder: An embodiment of arithmetic decoding procedure.

3.1.7 arithmetic encoder: An embodiment of arithmetic encoding procedure.

3.1.8 baseline (sequential): A particular sequential DCT-based encoding and decoding process specified in this Specification, and which is required for all DCT-based decoding processes.

3.1.9 binary decision: Choice between two alternatives.

3.1.10 bit stream: Partially encoded or decoded sequence of bits comprising an entropy-coded segment.

3.1.11 block: An 8×8 array of samples or an 8×8 array of DCT coefficient values of one component.

3.1.12 block-row: A sequence of eight contiguous component lines which are partitioned into 8×8 blocks.

3.1.13 byte: A group of 8 bits.

3.1.14 byte stuffing: A procedure in which either the Huffman coder or the arithmetic coder inserts a zero byte into the entropy-coded segment following the generation of an encoded hexadecimal X'FF' byte.

3.1.15 carry bit: A bit in the arithmetic encoder code register which is set if a carry-over in the code register overflows the eight bits reserved for the output byte.

3.1.16 ceiling function: The mathematical procedure in which the greatest integer value of a real number is obtained by selecting the smallest integer value which is greater than or equal to the real number.

3.1.17 class (of coding process): Lossy or lossless coding processes.

3.1.18 code register: The arithmetic encoder register containing the least significant bits of the partially completed entropy-coded segment. Alternatively, the arithmetic decoder register containing the most significant bits of a partially decoded entropy-coded segment.

3.1.19 coder: An embodiment of a coding process.

3.1.20 coding: Encoding or decoding.

3.1.21 coding model: A procedure used to convert input data into symbols to be coded.

3.1.22 (coding) process: A general term for referring to an encoding process, a decoding process, or both.

3.1.23 colour image: A continuous-tone image that has more than one component.

3.1.24 columns: Samples per line in a component.

3.1.25 component: One of the two-dimensional arrays which comprise an image.

3.1.26 compressed data: Either compressed image data or table specification data or both.

3.1.27 compressed image data: A coded representation of an image, as specified in this Specification.

3.1.28 compression: Reduction in the number of bits used to represent source image data.

3.1.29 conditional exchange: The interchange of MPS and LPS probability intervals whenever the size of the LPS interval is greater than the size of the MPS interval (in arithmetic coding).

3.1.30 (conditional) probability estimate: The probability value assigned to the LPS by the probability estimation state machine (in arithmetic coding).

3.1.31 conditioning table: The set of parameters which select one of the defined relationships between prior coding decisions and the conditional probability estimates used in arithmetic coding.

3.1.32 context: The set of previously coded binary decisions which is used to create the index to the probability estimation state machine (in arithmetic coding).

3.1.33 continuous-tone image: An image whose components have more than one bit per sample.

3.1.34 data unit: An 8×8 block of samples of one component in DCT-based processes; a sample in lossless processes.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

3.1.35 DC coefficient: The DCT coefficient for which the frequency is zero in both dimensions.

3.1.36 DC prediction: The procedure used by DCT-based encoders whereby the quantized DC coefficient from the previously encoded 8×8 block of the same component is subtracted from the current quantized DC coefficient.

3.1.37 (DCT) coefficient: The amplitude of a specific cosine basis function – may refer to an original DCT coefficient, to a quantized DCT coefficient, or to a dequantized DCT coefficient.

3.1.38 decoder: An embodiment of a decoding process.

3.1.39 decoding process: A process which takes as its input compressed image data and outputs a continuous-tone image.

3.1.40 default conditioning: The values defined for the arithmetic coding conditioning tables at the beginning of coding of an image.

3.1.41 dequantization: The inverse procedure to quantization by which the decoder recovers a representation of the DCT coefficients.

3.1.42 differential component: The difference between an input component derived from the source image and the corresponding reference component derived from the preceding frame for that component (in hierarchical mode coding).

3.1.43 differential frame: A frame in a hierarchical process in which differential components are either encoded or decoded.

3.1.44 (digital) reconstructed image (data): A continuous-tone image which is the output of any decoder defined in this Specification.

3.1.45 (digital) source image (data): A continuous-tone image used as input to any encoder defined in this Specification.

3.1.46 (digital) (still) image: A set of two-dimensional arrays of integer data.

3.1.47 discrete cosine transform; DCT: Either the forward discrete cosine transform or the inverse discrete cosine transform.

3.1.48 downsampling (filter): A procedure by which the spatial resolution of an image is reduced (in hierarchical mode coding).

3.1.49 encoder: An embodiment of an encoding process.

3.1.50 encoding process: A process which takes as its input a continuous-tone image and outputs compressed image data.

3.1.51 entropy-coded (data) segment: An independently decodable sequence of entropy encoded bytes of compressed image data.

3.1.52 (entropy-coded segment) pointer: The variable which points to the most recently placed (or fetched) byte in the entropy encoded segment.

3.1.53 entropy decoder: An embodiment of an entropy decoding procedure.

3.1.54 entropy decoding: A lossless procedure which recovers the sequence of symbols from the sequence of bits produced by the entropy encoder.

3.1.55 entropy encoder: An embodiment of an entropy encoding procedure.

3.1.56 entropy encoding: A lossless procedure which converts a sequence of input symbols into a sequence of bits such that the average number of bits per symbol approaches the entropy of the input symbols.

3.1.57 extended (DCT-based) process: A descriptive term for DCT-based encoding and decoding processes in which additional capabilities are added to the baseline sequential process.

3.1.58 forward discrete cosine transform; FDCT: A mathematical transformation using cosine basis functions which converts a block of samples into a corresponding block of original DCT coefficients.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

3.1.59 frame: A group of one or more scans (all using the same DCT-based or lossless process) through the data of one or more of the components in an image.

3.1.60 frame header: A marker segment that contains a start-of-frame marker and associated frame parameters that are coded at the beginning of a frame.

3.1.61 frequency: A two-dimensional index into the two-dimensional array of DCT coefficients.

3.1.62 (frequency) band: A contiguous group of coefficients from the zig-zag sequence (in progressive mode coding).

3.1.63 full progression: A process which uses both spectral selection and successive approximation (in progressive mode coding).

3.1.64 grayscale image: A continuous-tone image that has only one component.

3.1.65 hierarchical: A mode of operation for coding an image in which the first frame for a given component is followed by frames which code the differences between the source data and the reconstructed data from the previous frame for that component. Resolution changes are allowed between frames.

3.1.66 hierarchical decoder: A sequence of decoder processes in which the first frame for each component is followed by frames which decode an array of differences for each component and adds it to the reconstructed data from the preceding frame for that component.

3.1.67 hierarchical encoder: The mode of operation in which the first frame for each component is followed by frames which encode the array of differences between the source data and the reconstructed data from the preceding frame for that component.

3.1.68 horizontal sampling factor: The relative number of horizontal data units of a particular component with respect to the number of horizontal data units in the other components.

3.1.69 Huffman decoder: An embodiment of a Huffman decoding procedure.

3.1.70 Huffman decoding: An entropy decoding procedure which recovers the symbol from each variable length code produced by the Huffman encoder.

3.1.71 Huffman encoder: An embodiment of a Huffman encoding procedure.

3.1.72 Huffman encoding: An entropy encoding procedure which assigns a variable length code to each input symbol.

3.1.73 Huffman table: The set of variable length codes required in a Huffman encoder and Huffman decoder.

3.1.74 image data: Either source image data or reconstructed image data.

3.1.75 interchange format: The representation of compressed image data for exchange between application environments.

3.1.76 interleaved: The descriptive term applied to the repetitive multiplexing of small groups of data units from each component in a scan in a specific order.

3.1.77 inverse discrete cosine transform; IDCT: A mathematical transformation using cosine basis functions which converts a block of dequantized DCT coefficients into a corresponding block of samples.

3.1.78 Joint Photographic Experts Group; JPEG: The informal name of the committee which created this Specification. The “joint” comes from the CCITT and ISO/IEC collaboration.

3.1.79 latent output: Output of the arithmetic encoder which is held, pending resolution of carry-over (in arithmetic coding).

3.1.80 less probable symbol; LPS: For a binary decision, the decision value which has the smaller probability.

3.1.81 level shift: A procedure used by DCT-based encoders and decoders whereby each input sample is either converted from an unsigned representation to a two’s complement representation or from a two’s complement representation to an unsigned representation.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

3.1.82 lossless: A descriptive term for encoding and decoding processes and procedures in which the output of the decoding procedure(s) is identical to the input to the encoding procedure(s).

3.1.83 lossless coding: The mode of operation which refers to any one of the coding processes defined in this Specification in which all of the procedures are lossless (see Annex H).

3.1.84 lossy: A descriptive term for encoding and decoding processes which are not lossless.

3.1.85 marker: A two-byte code in which the first byte is hexadecimal FF (X'FF') and the second byte is a value between 1 and hexadecimal FE (X'FE').

3.1.86 marker segment: A marker and associated set of parameters.

3.1.87 MCU-row: The smallest sequence of MCU which contains at least one line of samples or one block-row from every component in the scan.

3.1.88 minimum coded unit; MCU: The smallest group of data units that is coded.

3.1.89 modes (of operation): The four main categories of image coding processes defined in this Specification.

3.1.90 more probable symbol; MPS: For a binary decision, the decision value which has the larger probability.

3.1.91 non-differential frame: The first frame for any components in a hierarchical encoder or decoder. The components are encoded or decoded without subtraction from reference components. The term refers also to any frame in modes other than the hierarchical mode.

3.1.92 non-interleaved: The descriptive term applied to the data unit processing sequence when the scan has only one component.

3.1.93 parameters: Fixed length integers 4, 8 or 16 bits in length, used in the compressed data formats.

3.1.94 point transform: Scaling of a sample or DCT coefficient.

3.1.95 precision: Number of bits allocated to a particular sample or DCT coefficient.

3.1.96 predictor: A linear combination of previously reconstructed values (in lossless mode coding).

3.1.97 probability estimation state machine: An interlinked table of probability values and indices which is used to estimate the probability of the LPS (in arithmetic coding).

3.1.98 probability interval: The probability of a particular sequence of binary decisions within the ordered set of all possible sequences (in arithmetic coding).

3.1.99 (probability) sub-interval: A portion of a probability interval allocated to either of the two possible binary decision values (in arithmetic coding).

3.1.100 procedure: A set of steps which accomplishes one of the tasks which comprise an encoding or decoding process.

3.1.101 process: See coding process.

3.1.102 progressive (coding): One of the DCT-based processes defined in this Specification in which each scan typically improves the quality of the reconstructed image.

3.1.103 progressive DCT-based: The mode of operation which refers to any one of the processes defined in Annex G.

3.1.104 quantization table: The set of 64 quantization values used to quantize the DCT coefficients.

3.1.105 quantization value: An integer value used in the quantization procedure.

3.1.106 quantize: The act of performing the quantization procedure for a DCT coefficient.

3.1.107 reference (reconstructed) component: Reconstructed component data which is used in a subsequent frame of a hierarchical encoder or decoder process (in hierarchical mode coding).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

3.1.108 renormalization: The doubling of the probability interval and the code register value until the probability interval exceeds a fixed minimum value (in arithmetic coding).

3.1.109 restart interval: The integer number of MCUs processed as an independent sequence within a scan.

3.1.110 restart marker: The marker that separates two restart intervals in a scan.

3.1.111 run (length): Number of consecutive symbols of the same value.

3.1.112 sample: One element in the two-dimensional array which comprises a component.

3.1.113 sample-interleaved: The descriptive term applied to the repetitive multiplexing of small groups of samples from each component in a scan in a specific order.

3.1.114 scan: A single pass through the data for one or more of the components in an image.

3.1.115 scan header: A marker segment that contains a start-of-scan marker and associated scan parameters that are coded at the beginning of a scan.

3.1.116 sequential (coding): One of the lossless or DCT-based coding processes defined in this Specification in which each component of the image is encoded within a single scan.

3.1.117 sequential DCT-based: The mode of operation which refers to any one of the processes defined in Annex F.

3.1.118 spectral selection: A progressive coding process in which the zig-zag sequence is divided into bands of one or more contiguous coefficients, and each band is coded in one scan.

3.1.119 stack counter: The count of X'FF' bytes which are held, pending resolution of carry-over in the arithmetic encoder.

3.1.120 statistical conditioning: The selection, based on prior coding decisions, of one estimate out of a set of conditional probability estimates (in arithmetic coding).

3.1.121 statistical model: The assignment of a particular conditional probability estimate to each of the binary arithmetic coding decisions.

3.1.122 statistics area: The array of statistics bins required for a coding process which uses arithmetic coding.

3.1.123 statistics bin: The storage location where an index is stored which identifies the value of the conditional probability estimate used for a particular arithmetic coding binary decision.

3.1.124 successive approximation: A progressive coding process in which the coefficients are coded with reduced precision in the first scan, and precision is increased by one bit with each succeeding scan.

3.1.125 table specification data: The coded representation from which the tables used in the encoder and decoder are generated and their destinations specified.

3.1.126 transcoder: A procedure for converting compressed image data of one encoder process to compressed image data of another encoder process.

3.1.127 (uniform) quantization: The procedure by which DCT coefficients are linearly scaled in order to achieve compression.

3.1.128 upsampling (filter): A procedure by which the spatial resolution of an image is increased (in hierarchical mode coding).

3.1.129 vertical sampling factor: The relative number of vertical data units of a particular component with respect to the number of vertical data units in the other components in the frame.

3.1.130 zero byte: The X'00' byte.

3.1.131 zig-zag sequence: A specific sequential ordering of the DCT coefficients from (approximately) lowest spatial frequency to highest.

3.1.132 3-sample predictor: A linear combination of the three nearest neighbor reconstructed samples to the left and above (in lossless mode coding).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

3.2 Symbols

The symbols used in this Specification are listed below.

A	probability interval
AC	AC DCT coefficient
AC_{ji}	AC coefficient predicted from DC values
Ah	successive approximation bit position, high
Al	successive approximation bit position, low
Ap_i	*i*th 8-bit parameter in APP_n segment
APP_n	marker reserved for application segments
B	current byte in compressed data
B2	next byte in compressed data when B = X'FF'
BE	counter for buffered correction bits for Huffman coding in the successive approximation process
BITS	16-byte list containing number of Huffman codes of each length
BP	pointer to compressed data
BPST	pointer to byte before start of entropy-coded segment
BR	counter for buffered correction bits for Huffman coding in the successive approximation process
Bx	byte modified by a carry-over
C	value of bit stream in code register
C_i	component identifier for frame
C_u	horizontal frequency dependent scaling factor in DCT
C_v	vertical frequency dependent scaling factor in DCT
CE	conditional exchange
C-low	low order 16 bits of the arithmetic decoder code register
Cm_i	*i*th 8-bit parameter in COM segment
CNT	bit counter in NEXTBYTE procedure
CODE	Huffman code value
CODESIZE(V)	code size for symbol V
COM	comment marker
Cs	conditioning table value
Cs_i	component identifier for scan
CT	renormalization shift counter
Cx	high order 16 bits of arithmetic decoder code register
CX	conditional exchange
d_{ji}	data unit from horizontal position i, vertical position j
d_{ji}^k	d_{ji} for component k
D	decision decoded

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Da	in DC coding, the DC difference coded for the previous block from the same component; in lossless coding, the difference coded for the sample immediately to the left
DAC	define-arithmetic-coding-conditioning marker
Db	the difference coded for the sample immediately above
DC	DC DCT coefficient
DC_i	DC coefficient for *i*th block in component
DC_k	*k*th DC value used in prediction of AC coefficients
DHP	define hierarchical progression marker
DHT	define-Huffman-tables marker
DIFF	difference between quantized DC and prediction
DNL	define-number-of-lines marker
DQT	define-quantization-tables marker
DRI	define restart interval marker
E	exponent in magnitude category upper bound
EC	event counter
ECS	entropy-coded segment
ECS_i	*i*th entropy-coded segment
Eh	horizontal expansion parameter in EXP segment
EHUFCO	Huffman code table for encoder
EHUFSI	encoder table of Huffman code sizes
EOB	end-of-block for sequential; end-of-band for progressive
EOBn	run length category for EOB runs
EOBx	position of EOB in previous successive approximation scan
EOB0, EOB1, ..., EOB14	run length categories for EOB runs
EOI	end-of-image marker
Ev	vertical expansion parameter in EXP segment
EXP	expand reference components marker
FREQ(V)	frequency of occurrence of symbol V
H_i	horizontal sampling factor for *i*th component
H_{max}	largest horizontal sampling factor
HUFFCODE	list of Huffman codes corresponding to lengths in HUFFSIZE
HUFFSIZE	list of code lengths
HUFFVAL	list of values assigned to each Huffman code
i	subscript index
I	integer variable
Index(S)	index to probability estimation state machine table for context index S
j	subscript index
J	integer variable

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

JPG	marker reserved for JPEG extensions
JPG_n	marker reserved for JPEG extensions
k	subscript index
K	integer variable
Kmin	index of 1st AC coefficient in band (1 for sequential DCT)
Kx	conditioning parameter for AC arithmetic coding model
L	DC and lossless coding conditioning lower bound parameter
L_i	element in BITS list in DHT segment
$L_i(t)$	element in BITS list in the DHT segment for Huffman table t
La	length of parameters in APP_n segment
LASTK	largest value of K
Lc	length of parameters in COM segment
Ld	length of parameters in DNL segment
Le	length of parameters in EXP segment
Lf	length of frame header parameters
Lh	length of parameters in DHT segment
Lp	length of parameters in DAC segment
LPS	less probable symbol (in arithmetic coding)
Lq	length of parameters in DQT segment
Lr	length of parameters in DRI segment
Ls	length of scan header parameters
LSB	least significant bit
m	modulo 8 counter for RST_m marker
m_t	number of $V_{i,j}$ parameters for Huffman table t
M	bit mask used in coding magnitude of V
Mn	*n*th statistics bin for coding magnitude bit pattern category
MAXCODE	table with maximum value of Huffman code for each code length
MCU	minimum coded unit
MCU_i	*i*th MCU
MCUR	number of MCU required to make up one MCU-row
MINCODE	table with minimum value of Huffman code for each code length
MPS	more probable symbol (in arithmetic coding)
MPS(S)	more probable symbol for context-index S
MSB	most significant bit
M2, M3, M4, ... , M15	designation of context-indices for coding of magnitude bits in the arithmetic coding models
n	integer variable
N	data unit counter for MCU coding
N/A	not applicable

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Nb	number of data units in MCU
Next_Index_LPS	new value of Index(S) after a LPS renormalization
Next_Index_MPS	new value of Index(S) after a MPS renormalization
Nf	number of components in frame
NL	number of lines defined in DNL segment
Ns	number of components in scan
OTHERS(V)	index to next symbol in chain
P	sample precision
Pq	quantizer precision parameter in DQT segment
Pq(t)	quantizer precision parameter in DQT segment for quantization table t
PRED	quantized DC coefficient from the most recently coded block of the component
Pt	point transform parameter
Px	calculated value of sample
Q_{ji}	quantizer value for coefficient AC_{ji}
Q_{vu}	quantization value for DCT coefficient S_{vu}
Q_{00}	quantizer value for DC coefficient
QAC_{ji}	quantized AC coefficient predicted from DC values
QDC_k	*k*th quantized DC value used in prediction of AC coefficients
Qe	LPS probability estimate
Qe(S)	LPS probability estimate for context index S
Qk	*k*th element of 64 quantization elements in DQT segment
r_{vu}	reconstructed image sample
R	length of run of zero amplitude AC coefficients
R_{vu}	dequantized DCT coefficient
Ra	reconstructed sample value
Rb	reconstructed sample value
Rc	reconstructed sample value
Rd	rounding in prediction calculation
RES	reserved markers
Ri	restart interval in DRI segment
RRRR	4-bit value of run length of zero AC coefficients
RS	composite value used in Huffman coding of AC coefficients
RST_m	restart marker number m
s_{yx}	reconstructed value from IDCT
S	context index
S_{vu}	DCT coefficient at horizontal frequency u, vertical frequency v

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

SC	context-index for coding of correction bit in successive approximation coding
Se	end of spectral selection band in zig-zag sequence
SE	context-index for coding of end-of-block or end-of-band
SI	Huffman code size
SIGN	1 if decoded sense of sign is negative and 0 if decoded sense of sign is positive
SIZE	length of a Huffman code
SLL	shift left logical operation
SLL α β	logical shift left of α by β bits
SN	context-index for coding of first magnitude category when V is negative
SOF_0	baseline DCT process frame marker
SOF_1	extended sequential DCT frame marker, Huffman coding
SOF_2	progressive DCT frame marker, Huffman coding
SOF_3	lossless process frame marker, Huffman coding
SOF_5	differential sequential DCT frame marker, Huffman coding
SOF_6	differential progressive DCT frame marker, Huffman coding
SOF_7	differential lossless process frame marker, Huffman coding
SOF_9	sequential DCT frame marker, arithmetic coding
SOF_{10}	progressive DCT frame marker, arithmetic coding
SOF_{11}	lossless process frame marker, arithmetic coding
SOF_{13}	differential sequential DCT frame marker, arithmetic coding
SOF_{14}	differential progressive DCT frame marker, arithmetic coding
SOF_{15}	differential lossless process frame marker, arithmetic coding
SOI	start-of-image marker
SOS	start-of-scan marker
SP	context-index for coding of first magnitude category when V is positive
Sq_{vu}	quantized DCT coefficient
SRL	shift right logical operation
SRL α β	logical shift right of α by β bits
Ss	start of spectral selection band in zig-zag sequence
SS	context-index for coding of sign decision
SSSS	4-bit size category of DC difference or AC coefficient amplitude
ST	stack counter
Switch_MPS	parameter controlling inversion of sense of MPS
Sz	parameter used in coding magnitude of V
S0	context-index for coding of V = 0 decision
t	summation index for parameter limits computation
T	temporary variable

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Ta_j	AC entropy table destination selector for *j*th component in scan
Tb	arithmetic conditioning table destination identifier
Tc	Huffman coding or arithmetic coding table class
Td_j	DC entropy table destination selector for *j*th component in scan
TEM	temporary marker
Th	Huffman table destination identifier in DHT segment
Tq	quantization table destination identifier in DQT segment
Tq_i	quantization table destination selector for *i*th component in frame
U	DC and lossless coding conditioning upper bound parameter
V	symbol or value being either encoded or decoded
V_i	vertical sampling factor for *i*th component
$V_{i,j}$	*j*th value for length *i* in HUFFVAL
V_{max}	largest vertical sampling factor
V_t	temporary variable
VALPTR	list of indices for first value in HUFFVAL for each code length
V1	symbol value
V2	symbol value
x_i	number of columns in *i*th component
X	number of samples per line in component with largest horizontal dimension
X_i	*i*th statistics bin for coding magnitude category decision
X1, X2, X3, ... , X15	designation of context-indices for coding of magnitude categories in the arithmetic coding models
XHUFCO	extended Huffman code table
XHUFSI	table of sizes of extended Huffman codes
X'values'	values within the quotes are hexadecimal
y_i	number of lines in *i*th component
Y	number of lines in component with largest vertical dimension
ZRL	value in HUFFVAL assigned to run of 16 zero coefficients
ZZ(K)	*K*th element in zig-zag sequence of quantized DCT coefficients
ZZ(0)	quantized DC coefficient in zig-zag sequence order

4 General

The purpose of this clause is to give an informative overview of the elements specified in this Specification. Another purpose is to introduce many of the terms which are defined in clause 3. These terms are printed in *italics* upon first usage in this clause.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

4.1 Elements specified in this Specification

There are three elements specified in this Specification:

a) An *encoder* is an embodiment of an *encoding process*. As shown in Figure 1, an encoder takes as input *digital source image data* and *table specifications*, and by means of a specified set of *procedures* generates as output *compressed image data*.

b) A *decoder* is an embodiment of a *decoding process*. As shown in Figure 2, a decoder takes as input compressed image data and table specifications, and by means of a specified set of procedures generates as output *digital reconstructed image data*.

c) The *interchange format*, shown in Figure 3, is a compressed image data representation which includes all table specifications used in the encoding process. The interchange format is for exchange between *application environments*.

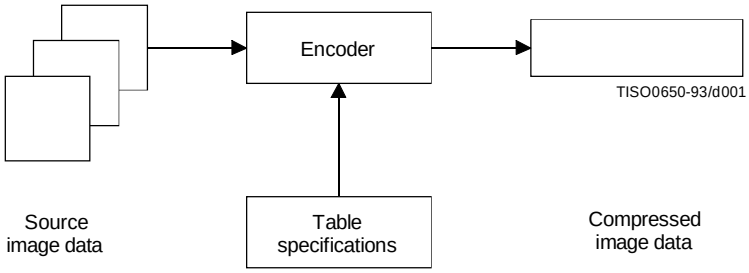


Figure 1 – Encoder

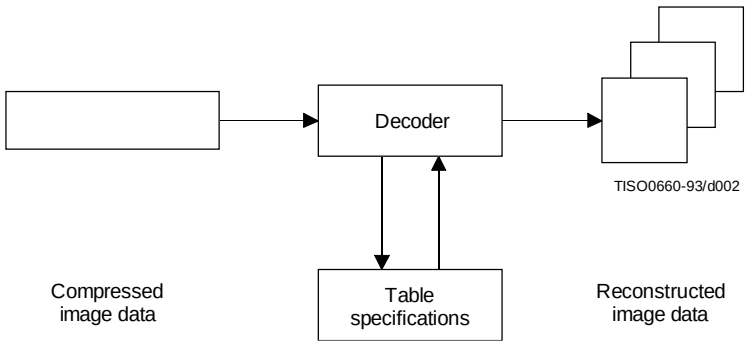


Figure 2 – Decoder

Figures 1 and 2 illustrate the general case for which the *continuous-tone* source and reconstructed image data consist of multiple *components*. (A *colour* image consists of multiple components; a *grayscale* image consists only of a single component.) A significant portion of this Specification is concerned with how to handle multiple-component images in a flexible, application-independent way.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

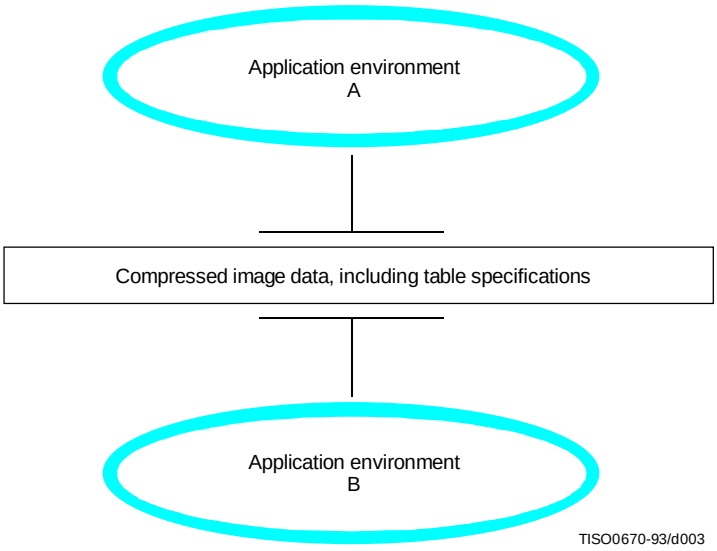


Figure 3 – Interchange format for compressed image data

These figures are also meant to show that the same tables specified for an encoder to use to compress a particular image must be provided to a decoder to reconstruct that image. However, this Specification does not specify how applications should associate tables with compressed image data, nor how they should represent source image data generally within their specific environments.

Consequently, this Specification also specifies the interchange format shown in Figure 3, in which table specifications are included within compressed image data. An image compressed with a specified encoding process within one application environment, A, is passed to a different environment, B, by means of the interchange format. The interchange format does not specify a complete coded image representation. Application-dependent information, e.g. colour space, is outside the scope of this Specification.

4.2 Lossy and lossless compression

This Specification specifies two *classes* of encoding and decoding processes, *lossy* and *lossless* processes. Those based on the *discrete cosine transform* (DCT) are lossy, thereby allowing substantial *compression* to be achieved while producing a reconstructed image with high visual fidelity to the encoder's source image.

The simplest DCT-based *coding process* is referred to as the *baseline sequential* process. It provides a capability which is sufficient for many applications. There are additional DCT-based processes which extend the baseline sequential process to a broader range of applications. In any decoder using *extended DCT-based decoding processes*, the baseline decoding process is required to be present in order to provide a default decoding capability.

The second class of coding processes is not based upon the DCT and is provided to meet the needs of applications requiring lossless compression. These lossless encoding and decoding processes are used independently of any of the DCT-based processes.

A table summarizing the relationship among these lossy and lossless coding processes is included in 4.11.

The amount of compression provided by any of the various processes is dependent on the characteristics of the particular image being compressed, as well as on the picture quality desired by the application and the desired speed of compression and decompression.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

4.3 DCT-based coding

Figure 4 shows the main procedures for all encoding processes based on the DCT. It illustrates the special case of a single-component image; this is an appropriate simplification for overview purposes, because all processes specified in this Specification operate on each image component independently.

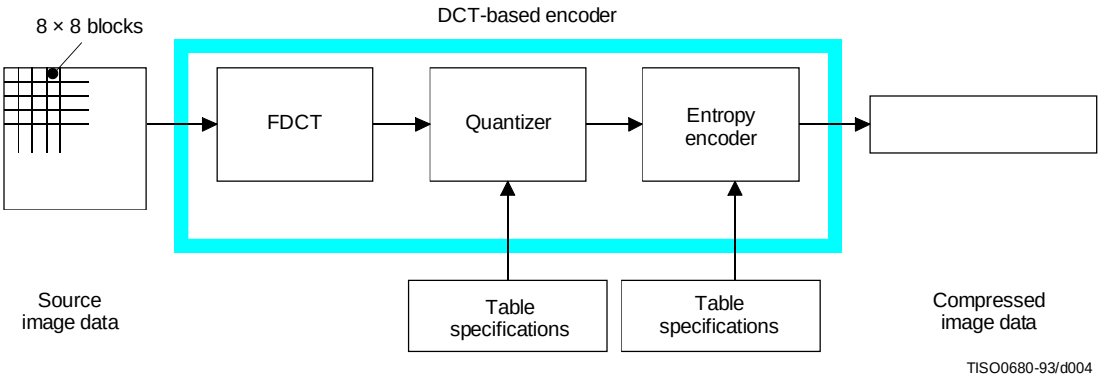


Figure 4 – DCT-based encoder simplified diagram

In the encoding process the input component's *samples* are grouped into 8 × 8 *blocks*, and each block is transformed by the *forward DCT* (FDCT) into a set of 64 values referred to as *DCT coefficients*. One of these values is referred to as the *DC coefficient* and the other 63 as the *AC coefficients*.

Each of the 64 coefficients is then *quantized* using one of 64 corresponding values from a *quantization table* (determined by one of the table specifications shown in Figure 4). No default values for quantization tables are specified in this Specification; applications may specify values which customize picture quality for their particular image characteristics, display devices, and viewing conditions.

After quantization, the DC coefficient and the 63 AC coefficients are prepared for *entropy encoding*, as shown in Figure 5. The previous quantized DC coefficient is used to predict the current quantized DC coefficient, and the difference is encoded. The 63 quantized AC coefficients undergo no such differential encoding, but are converted into a one-dimensional *zig-zag sequence*, as shown in Figure 5.

The quantized coefficients are then passed to an entropy encoding procedure which compresses the data further. One of two entropy coding procedures can be used, as described in 4.6. If *Huffman encoding* is used, *Huffman table* specifications must be provided to the encoder. If *arithmetic encoding* is used, arithmetic coding *conditioning table* specifications may be provided, otherwise the default conditioning table specifications shall be used.

Figure 6 shows the main procedures for all DCT-based decoding processes. Each step shown performs essentially the inverse of its corresponding main procedure within the encoder. The entropy decoder decodes the zig-zag sequence of quantized DCT coefficients. After *dequantization* the DCT coefficients are transformed to an 8 × 8 block of samples by the *inverse DCT* (IDCT).

4.4 Lossless coding

Figure 7 shows the main procedures for the lossless encoding processes. A *predictor* combines the reconstructed values of up to three neighbourhood samples at positions a, b, and c to form a prediction of the sample at position x as shown in Figure 8. This prediction is then subtracted from the actual value of the sample at position x, and the difference is losslessly entropy-coded by either Huffman or arithmetic coding.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

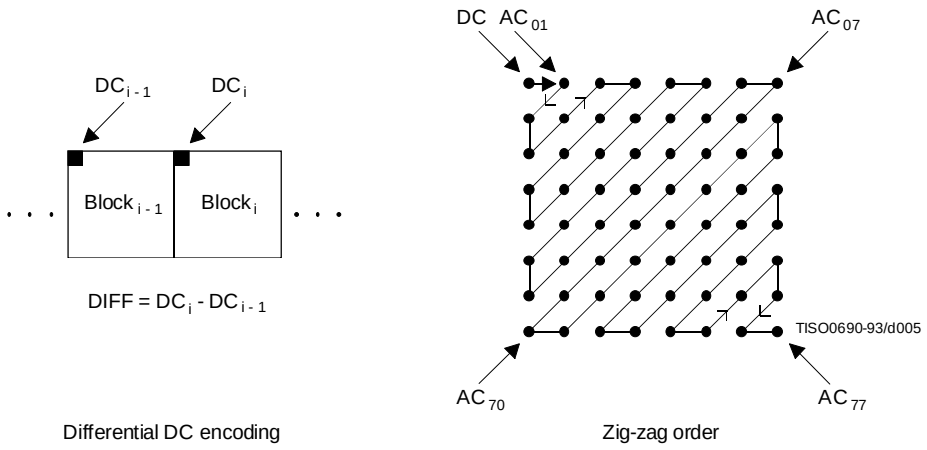


Figure 5 – Preparation of quantized coefficients for entropy encoding

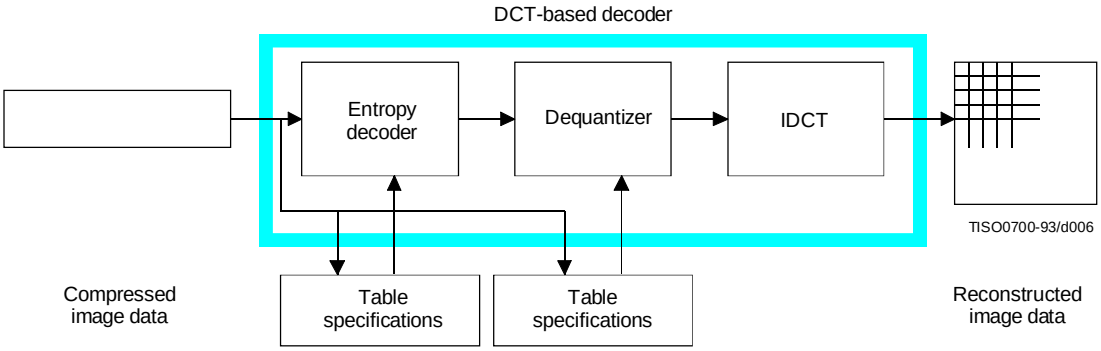


Figure 6 – DCT-based decoder simplified diagram

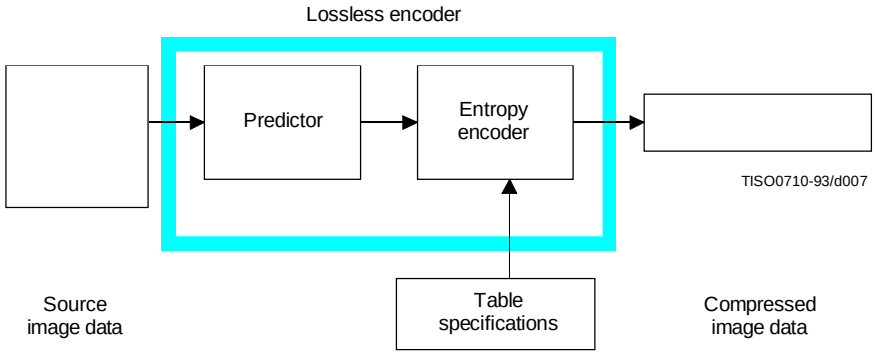


Figure 7 – Lossless encoder simplified diagram

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

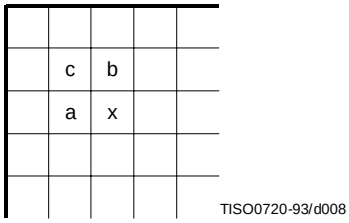


Figure 8 – 3-sample prediction neighbourhood

This encoding process may also be used in a slightly modified way, whereby the *precision* of the input samples is reduced by one or more bits prior to the lossless coding. This achieves higher compression than the lossless process (but lower compression than the DCT-based processes for equivalent visual fidelity), and limits the reconstructed image's worst-case sample error to the amount of input precision reduction.

4.5 Modes of operation

There are four distinct *modes of operation* under which the various coding processes are defined: *sequential DCT-based*, *progressive DCT-based*, lossless, and *hierarchical*. (Implementations are not required to provide all of these.) The lossless mode of operation was described in 4.4. The other modes of operation are compared as follows.

For the sequential DCT-based mode, 8×8 sample blocks are typically input block by block from left to right, and block-row by block-row from top to bottom. After a block has been transformed by the forward DCT, quantized and prepared for entropy encoding, all 64 of its quantized DCT coefficients can be immediately entropy encoded and output as part of the compressed image data (as was described in 4.3), thereby minimizing coefficient storage requirements.

For the progressive DCT-based mode, 8×8 blocks are also typically encoded in the same order, but in multiple *scans* through the image. This is accomplished by adding an image-sized coefficient memory buffer (not shown in Figure 4) between the quantizer and the entropy encoder. As each block is transformed by the forward DCT and quantized, its coefficients are stored in the buffer. The DCT coefficients in the buffer are then partially encoded in each of multiple scans. The typical sequence of image presentation at the output of the decoder for sequential versus progressive modes of operation is shown in Figure 9.

There are two procedures by which the quantized coefficients in the buffer may be partially encoded within a scan. First, only a specified *band* of coefficients from the zig-zag sequence need be encoded. This procedure is called *spectral selection*, because each band typically contains coefficients which occupy a lower or higher part of the *frequency* spectrum for that 8×8 block. Secondly, the coefficients within the current band need not be encoded to their full (quantized) accuracy within each scan. Upon a coefficient's first encoding, a specified number of most significant bits is encoded first. In subsequent scans, the less significant bits are then encoded. This procedure is called *successive approximation*. Either procedure may be used separately, or they may be mixed in flexible combinations.

In hierarchical mode, an image is encoded as a sequence of *frames*. These frames provide *reference reconstructed components* which are usually needed for prediction in subsequent frames. Except for the first frame for a given component, *differential frames* encode the difference between source components and reference reconstructed components. The coding of the differences may be done using only DCT-based processes, only lossless processes, or DCT-based processes with a final lossless process for each component. *Downsampling* and *upsampling filters* may be used to provide a pyramid of spatial resolutions as shown in Figure 10. Alternatively, the hierarchical mode can be used to improve the quality of the reconstructed components at a given spatial resolution.

Hierarchical mode offers a progressive presentation similar to the progressive DCT-based mode but is useful in environments which have multi-resolution requirements. Hierarchical mode also offers the capability of progressive coding to a final lossless stage.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

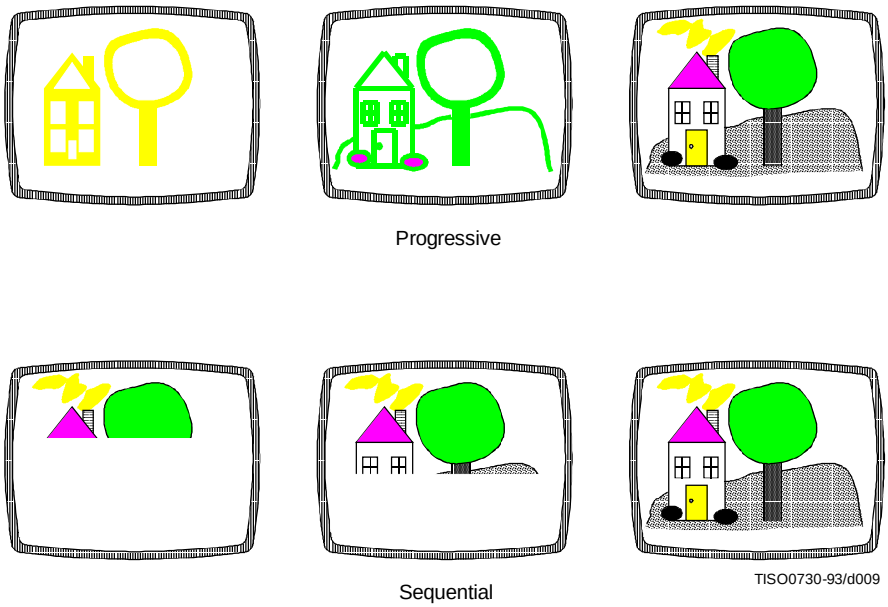


Figure 9 – Progressive versus sequential presentation

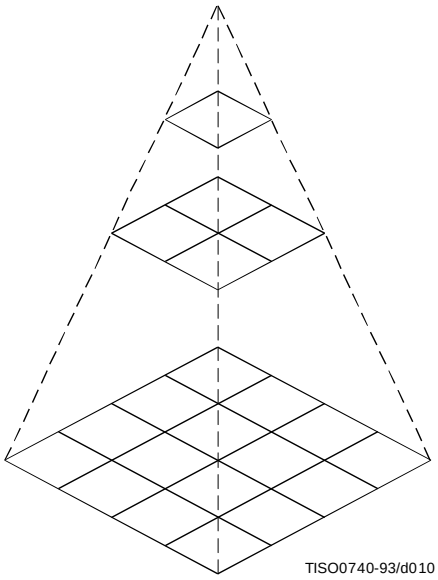


Figure 10 – Hierarchical multi-resolution encoding

4.6 Entropy coding alternatives

Two alternative entropy coding procedures are specified: Huffman coding and arithmetic coding. Huffman coding procedures use Huffman tables, determined by one of the table specifications shown in Figures 1 and 2. Arithmetic coding procedures use arithmetic coding conditioning tables, which may also be determined by a table specification. No default values for Huffman tables are specified, so that applications may choose tables appropriate for their own environments. Default tables are defined for the arithmetic coding conditioning.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The baseline sequential process uses Huffman coding, while the extended DCT-based and lossless processes may use either Huffman or arithmetic coding.

4.7 Sample precision

For DCT-based processes, two alternative sample precisions are specified: either 8 bits or 12 bits per sample. Applications which use samples with other precisions can use either 8-bit or 12-bit precision by shifting their source image samples appropriately. The baseline process uses only 8-bit precision. DCT-based implementations which handle 12-bit source image samples are likely to need greater computational resources than those which handle only 8-bit source images. Consequently in this Specification separate normative requirements are defined for 8-bit and 12-bit DCT-based processes.

For lossless processes the sample precision is specified to be from 2 to 16 bits.

4.8 Multiple-component control

Subclauses 4.3 and 4.4 give an overview of one major part of the encoding and decoding processes – those which operate on the sample values in order to achieve compression. There is another major part as well – the procedures which control the order in which the image data from multiple components are processed to create the compressed data, and which ensure that the proper set of table data is applied to the proper *data units* in the image. (A data unit is a sample for lossless processes and an 8×8 block of samples for DCT-based processes.)

4.8.1 Interleaving multiple components

Figure 11 shows an example of how an encoding process selects between multiple source image components as well as multiple sets of table data, when performing its encoding procedures. The source image in this example consists of the three components A, B and C, and there are two sets of table specifications. (This simplified view does not distinguish between the quantization tables and entropy coding tables.)

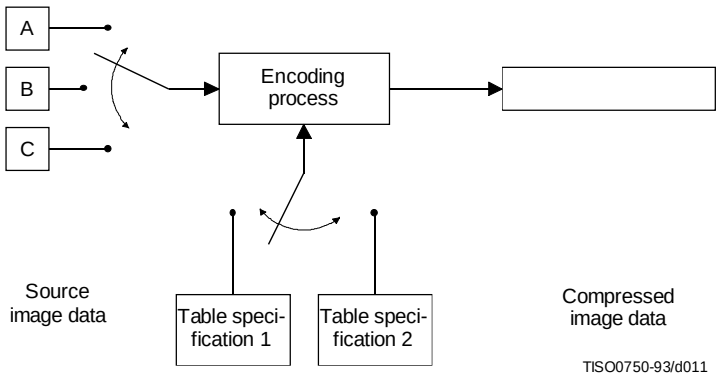


Figure 11 – Component-interleave and table-switching control

In sequential mode, encoding is *non-interleaved* if the encoder compresses all image data units in component A before beginning component B, and then in turn all of B before C. Encoding is *interleaved* if the encoder compresses a data unit from A, a data unit from B, a data unit from C, then back to A, etc. These alternatives are illustrated in Figure 12, which shows a case in which all three image components have identical dimensions: X *columns* by Y lines, for a total of n data units each.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

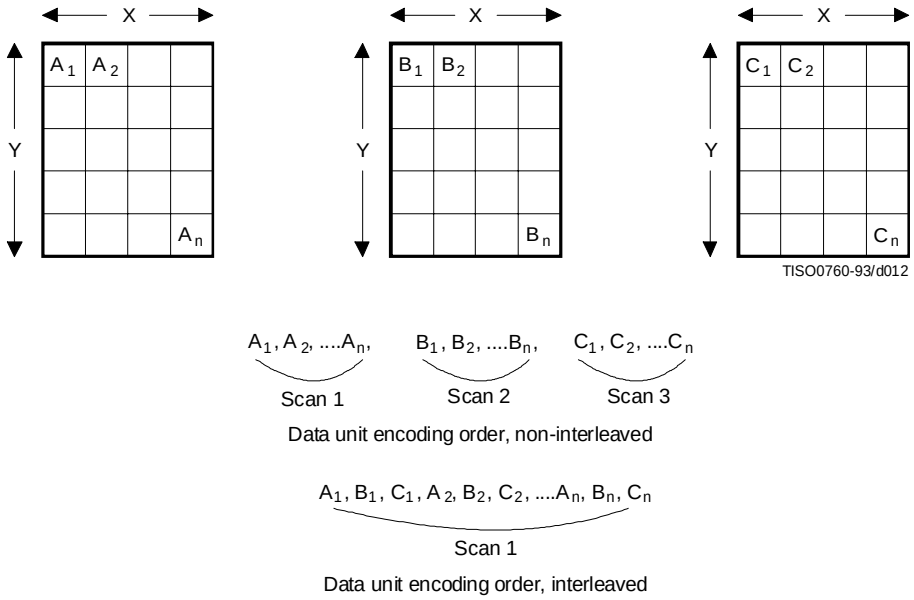


Figure 12 – Interleaved versus non-interleaved encoding order

These control procedures are also able to handle cases in which the source image components have different dimensions. Figure 13 shows a case in which two of the components, B and C, have half the number of horizontal samples relative to component A. In this case, two data units from A are interleaved with one each from B and C. Cases in which components of an image have more complex relationships, such as different horizontal and vertical dimensions, can be handled as well. (See Annex A.)

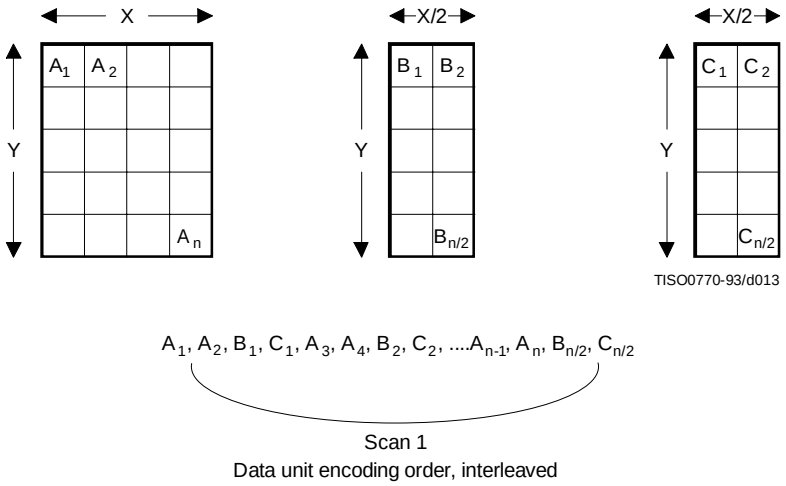


Figure 13 – Interleaved order for components with different dimensions

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

4.8.2 Minimum coded unit

Related to the concepts of multiple-component interleave is the *minimum coded unit* (MCU). If the compressed image data is non-interleaved, the MCU is defined to be one data unit. For example, in Figure 12 the MCU for the non-interleaved case is a single data unit. If the compressed data is interleaved, the MCU contains one or more data units from each component. For the interleaved case in Figure 12, the (first) MCU consists of the three interleaved data units A_1, B_1, C_1. In the example of Figure 13, the (first) MCU consists of the four data units A_1, A_2 , B_1, C_1.

4.9 Structure of compressed data

Figures 1, 2, and 3 all illustrate slightly different views of compressed image data. Figure 1 shows this data as the output of an encoding process, Figure 2 shows it as the input to a decoding process, and Figure 3 shows compressed image data in the interchange format, at the interface between applications.

Compressed image data are described by a uniform structure and set of *parameters* for both classes of encoding processes (lossy or lossless), and for all modes of operation (sequential, progressive, lossless, and hierarchical). The various parts of the compressed image data are identified by special two-byte codes called *markers*. Some markers are followed by particular sequences of parameters, as in the case of table specifications, *frame header*, or *scan header*. Others are used without parameters for functions such as marking the start-of-image and end-of-image. When a marker is associated with a particular sequence of parameters, the marker and its parameters comprise a *marker segment*.

The data created by the entropy encoder are also segmented, and one particular marker – *the restart marker* – is used to isolate *entropy-coded data segments*. The encoder outputs the restart markers, intermixed with the entropy-coded data, at regular *restart intervals* of the source image data. Restart markers can be identified without having to decode the compressed data to find them. Because they can be independently decoded, they have application-specific uses, such as parallel encoding or decoding, isolation of data corruptions, and semi-random access of entropy-coded segments.

There are three compressed data formats:

a) the interchange format;

b) the *abbreviated format* for compressed image data;

c) the abbreviated format for table-specification data.

4.9.1 Interchange format

In addition to certain required marker segments and the entropy-coded segments, the interchange format shall include the marker segments for all quantization and entropy-coding table specifications needed by the decoding process. This guarantees that a compressed image can cross the boundary between application environments, regardless of how each environment internally associates tables with compressed image data.

4.9.2 Abbreviated format for compressed image data

The abbreviated format for compressed image data is identical to the interchange format, except that it does not include all tables required for decoding. (It may include some of them.) This format is intended for use within applications where alternative mechanisms are available for supplying some or all of the table-specification data needed for decoding.

4.9.3 Abbreviated format for table-specification data

This format contains only table-specification data. It is a means by which the application may install in the decoder the tables required to subsequently reconstruct one or more images.

4.10 Image, frame, and scan

Compressed image data consists of only one image. An image contains only one frame in the cases of sequential and progressive coding processes; an image contains multiple frames for the hierarchical mode.

A frame contains one or more scans. For sequential processes, a scan contains a complete encoding of one or more image components. In Figures 12 and 13, the frame consists of three scans when non-interleaved, and one scan if all three components are interleaved together. The frame could also consist of two scans: one with a non-interleaved component, the other with two components interleaved.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

For progressive processes, a scan contains a partial encoding of all data units from one or more image components. Components shall not be interleaved in progressive mode, except for the DC coefficients in the first scan for each component of a progressive frame.

4.11 Summary of coding processes

Table 1 provides a summary of the essential characteristics of the various coding processes specified in this Specification. The full specification of these processes is contained in Annexes F, G, H, and J.

Table 1 – Summary: Essential characteristics of coding processes

Baseline process (required for all DCT-based decoders)
• DCT-based process • Source image: 8-bit samples within each component • Sequential • Huffman coding: 2 AC and 2 DC tables • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans

Extended DCT-based processes
• DCT-based process • Source image: 8-bit or 12-bit samples • Sequential or progressive • Huffman or arithmetic coding: 4 AC and 4 DC tables • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans

Lossless processes
• Predictive process (not DCT-based) • Source image: P-bit samples ($2 \leq P \leq 16$) • Sequential • Huffman or arithmetic coding: 4 DC tables • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans

Hierarchical processes
• Multiple frames (non-differential and differential) • Uses extended DCT-based or lossless processes • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

5 Interchange format requirements

The interchange format is the coded representation of compressed image data for exchange between application environments.

The interchange format requirements are that any compressed image data represented in interchange format shall comply with the syntax and code assignments appropriate for the decoding process selected, as specified in Annex B.

Tests for whether compressed image data comply with these requirements are specified in Part 2 of this Specification.

6 Encoder requirements

An encoding process converts source image data to compressed image data. Each of Annexes F, G, H, and J specifies a number of distinct encoding processes for its particular mode of operation.

An encoder is an embodiment of one (or more) of the encoding processes specified in Annexes F, G, H, or J. In order to comply with this Specification, an encoder shall satisfy at least one of the following two requirements.

An encoder shall

a) with appropriate accuracy, convert source image data to compressed image data which comply with the interchange format syntax specified in Annex B for the encoding process(es) embodied by the encoder;

b) with appropriate accuracy, convert source image data to compressed image data which comply with the abbreviated format for compressed image data syntax specified in Annex B for the encoding process(es) embodied by the encoder.

For each of the encoding processes specified in Annexes F, G, H, and J, the compliance tests for the above requirements are specified in Part 2 of this Specification.

NOTE – There is **no requirement** in this Specification that any encoder which embodies one of the encoding processes specified in Annexes F, G, H, or J shall be able to operate for all ranges of the parameters which are allowed for that process. An encoder is only required to meet the compliance tests specified in Part 2, and to generate the compressed data format according to Annex B for those parameter values which it does use.

7 Decoder requirements

A decoding process converts compressed image data to reconstructed image data. Each of Annexes F, G, H, and J specifies a number of distinct decoding processes for its particular mode of operation.

A decoder is an embodiment of one (or more) of the decoding processes specified in Annexes F, G, H, or J. In order to comply with this Specification, a decoder shall satisfy all three of the following requirements.

A decoder shall

a) with appropriate accuracy, convert to reconstructed image data any compressed image data with parameters within the range supported by the application, and which comply with the interchange format syntax specified in Annex B for the decoding process(es) embodied by the decoder;

b) accept and properly store any table-specification data which comply with the abbreviated format for table-specification data syntax specified in Annex B for the decoding process(es) embodied by the decoder;

c) with appropriate accuracy, convert to reconstructed image data any compressed image data which comply with the abbreviated format for compressed image data syntax specified in Annex B for the decoding process(es) embodied by the decoder, provided that the table-specification data required for decoding the compressed image data has previously been installed into the decoder.

Additionally, any DCT-based decoder, if it embodies any DCT-based decoding process other than baseline sequential, shall also embody the baseline sequential decoding process.

For each of the decoding processes specified in Annexes F, G, H, and J, the compliance tests for the above requirements are specified in Part 2 of this Specification.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex A

Mathematical definitions

(This annex forms an integral part of this Recommendation | International Standard)

A.1 Source image

Source images to which the encoding processes specified in this Specification can be applied are defined in this annex.

A.1.1 Dimensions and sampling factors

As shown in Figure A.1, a source image is defined to consist of Nf components. Each component, with unique identifier C_i, is defined to consist of a rectangular array of samples of x_i columns by y_i lines. The component dimensions are derived from two parameters, X and Y, where X is the maximum of the x_i values and Y is the maximum of the y_i values for all components in the frame. For each component, sampling factors H_i and V_i are defined relating component dimensions x_i and y_i to maximum dimensions X and Y, according to the following expressions:

$$x_i = \left\lceil X \times \frac{H_i}{H_{max}} \right\rceil \text{ and } y_i \left\lceil Y \times \frac{V_i}{V_{max}} \right\rceil,$$

where H_{max} and V_{max} are the maximum sampling factors for all components in the frame, and $\lceil\ \rceil$ is the ceiling function.

As an example, consider an image having 3 components with maximum dimensions of 512 lines and 512 samples per line, and with the following sampling factors:

Component 0	$H_0 = 4, \quad V_0 = 1$
Component 1	$H_1 = 2, \quad V_1 = 2$
Component 2	$H_2 = 1, \quad V_2 = 1$

Then $X = 512$, $Y = 512$, $H_{max} = 4$, $V_{max} = 2$, and x_i and y_i for each component are

Component 0	$x_0 = 512, \quad y_0 = 256$
Component 1	$x_1 = 256, \quad y_1 = 512$
Component 2	$x_2 = 128, \quad y_2 = 256$

NOTE – The X, Y, H_i, and V_i parameters are contained in the frame header of the compressed image data (see B.2.2), whereas the individual component dimensions x_i and y_i are derived by the decoder. Source images with x_i and y_i dimensions which do not satisfy the expressions above cannot be properly reconstructed.

A.1.2 Sample precision

A sample is an integer with precision P bits, with any value in the range 0 through 2^{P-1}. All samples of all components within an image shall have the same precision P. Restrictions on the value of P depend on the mode of operation, as specified in B.2 to B.7.

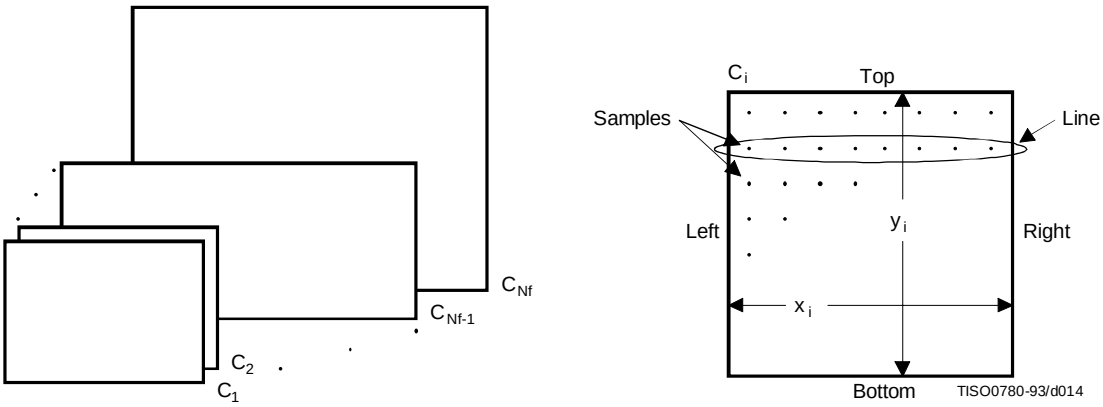
A.1.3 Data unit

A data unit is a sample in lossless processes and an 8×8 block of contiguous samples in DCT-based processes. The left-most 8 samples of each of the top-most 8 rows in the component shall always be the top-left-most block. With this top-left-most block as the reference, the component is partitioned into contiguous data units to the right and to the bottom (as shown in Figure A.4).

A.1.4 Orientation

Figure A.1 indicates the orientation of an image component by the terms top, bottom, left, and right. The order by which the data units of an image component are input to the compression encoding procedures is defined to be left-to-right and top-to-bottom within the component. (This ordering is precisely defined in A.2.) Applications determine which edges of a source image are defined as top, bottom, left, and right.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



a) Source image with multiple components

b) Characteristics of an image component

Figure A.1 – Source image characteristics

A.2 Order of source image data encoding

The scan header (see B.2.3) specifies the order by which source image data units shall be encoded and placed within the compressed image data. For a given scan, if the scan header parameter Ns = 1, then data from only one source component – the component specified by parameter Cs_1 – shall be present within the scan. This data is non-interleaved by definition. If Ns > 1, then data from the Ns components Cs_1 through Cs_{Ns} shall be present within the scan. This data shall always be interleaved. The order of components in a scan shall be according to the order specified in the frame header.

The ordering of data units and the construction of minimum coded units (MCU) is defined as follows.

A.2.1 Minimum coded unit (MCU)

For non-interleaved data the MCU is one data unit. For interleaved data the MCU is the sequence of data units defined by the sampling factors of the components in the scan.

A.2.2 Non-interleaved order (Ns = 1)

When Ns = 1 (where Ns is the number of components in a scan), the order of data units within a scan shall be left-to-right and top-to-bottom, as shown in Figure A.2. This ordering applies whenever Ns = 1, regardless of the values of H_1 and V_1.

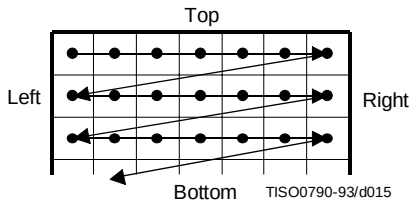


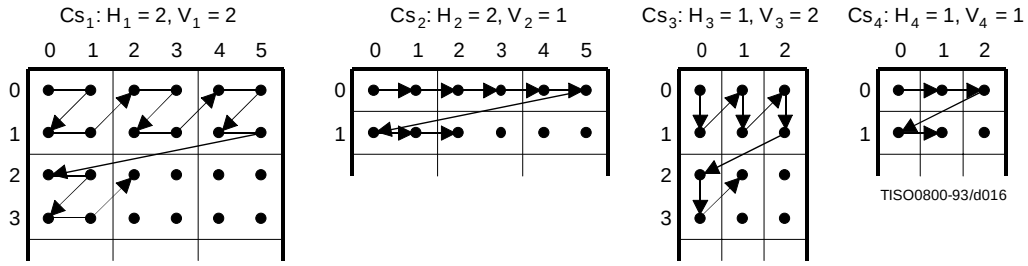
Figure A.2 – Non-interleaved data ordering

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

A.2.3 Interleaved order (Ns > 1)

When Ns > 1, each scan component Cs_i is partitioned into small rectangular arrays of H_k horizontal data units by V_k vertical data units. The subscripts k indicate that H_k and V_k are from the position in the frame header component-specification for which $C_k = Cs_i$. Within each H_k by V_k array, data units are ordered from left-to-right and top-to-bottom. The arrays in turn are ordered from left-to-right and top-to-bottom within each component.

As shown in the example of Figure A.3, Ns = 4, and MCU_1 consists of data units taken first from the top-left-most region of Cs_1, followed by data units from the corresponding region of Cs_2, then from Cs_3 and then from Cs_4. MCU_2 follows the same ordering for data taken from the next region to the right for the four components.



$$\begin{aligned}
MCU_1 &= d^1_{00}\ d^1_{01}\ d^1_{10}\ d^1_{11}\ d^2_{00}\ d^2_{01}\ d^3_{00}\ d^3_{10}\ d^4_{00},\\
MCU_2 &= d^1_{02}\ d^1_{03}\ d^1_{12}\ d^1_{13}\ d^2_{02}\ d^2_{03}\ d^3_{01}\ d^3_{11}\ d^4_{01},\\
MCU_3 &= d^1_{04}\ d^1_{05}\ d^1_{14}\ d^1_{15}\ d^2_{04}\ d^2_{05}\ d^3_{02}\ d^3_{12}\ d^4_{02},\\
MCU_4 &= d^1_{20}\ d^1_{21}\ d^1_{30}\ d^1_{31}\ d^2_{10}\ d^2_{11}\ d^3_{20}\ d^3_{30}\ d^4_{10},
\end{aligned}$$

Cs_1 data units Cs_2 Cs_3 Cs_4

Figure A.3 – Interleaved data ordering example

A.2.4 Completion of partial MCU

For DCT-based processes the data unit is a block. If x_i is not a multiple of 8, the encoding process shall extend the number of columns to complete the right-most sample blocks. If the component is to be interleaved, the encoding process shall also extend the number of samples by one or more additional blocks, if necessary, so that the number of blocks is an integer multiple of H_i. Similarly, if y_i is not a multiple of 8, the encoding process shall extend the number of lines to complete the bottom-most block-row. If the component is to be interleaved, the encoding process shall also extend the number of lines by one or more additional block-rows, if necessary, so that the number of block-rows is an integer multiple of V_i.

NOTE – It is recommended that any incomplete MCUs be completed by replication of the right-most column and the bottom line of each component.

For lossless processes the data unit is a sample. If the component is to be interleaved, the encoding process shall extend the number of samples, if necessary, so that the number is a multiple of H_i. Similarly, the encoding process shall extend the number of lines, if necessary, so that the number of lines is a multiple of V_i.

Any sample added by an encoding process to complete partial MCUs shall be removed by the decoding process.

A.3 DCT compression

A.3.1 Level shift

Before a non-differential frame encoding process computes the FDCT for a block of source image samples, the samples shall be level shifted to a signed representation by subtracting 2^{P-1}, where P is the precision parameter specified in B.2.2. Thus, when P = 8, the level shift is by 128; when P = 12, the level shift is by 2048.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

After a non-differential frame decoding process computes the IDCT and produces a block of reconstructed image samples, an inverse level shift shall restore the samples to the unsigned representation by adding 2^{P-1} and clamping the results to the range 0 to 2^{P-1}.

A.3.2 Orientation of samples for FDCT computation

Figure A.4 shows an image component which has been partitioned into 8×8 blocks for the FDCT computations. Figure A.4 also defines the orientation of the samples within a block by showing the indices used in the FDCT equation of A.3.3.

The definitions of block partitioning and sample orientation also apply to any DCT decoding process and the output reconstructed image. Any sample added by an encoding process to complete partial MCUs shall be removed by the decoding process.

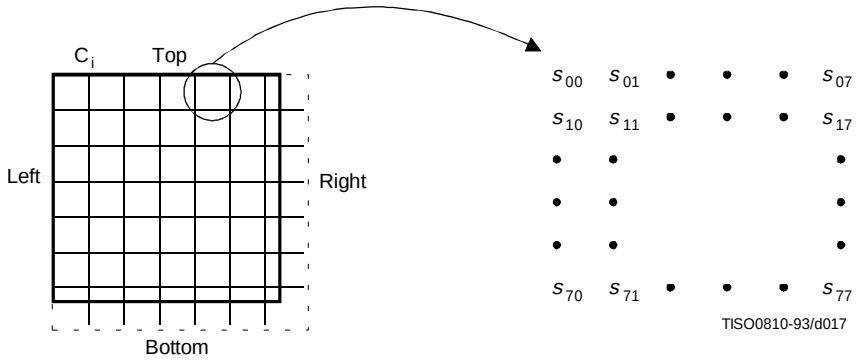


Figure A.4 – Partition and orientation of 8 x 8 sample blocks

A.3.3 FDCT and IDCT (informative)

The following equations specify the ideal functional definition of the FDCT and the IDCT.

NOTE – These equations contain terms which cannot be represented with perfect accuracy by any real implementation. The accuracy requirements for the combined FDCT and quantization procedures are specified in Part 2 of this Specification. The accuracy requirements for the combined dequantization and IDCT procedures are also specified in Part 2 of this Specification.

FDCT:
$$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

IDCT:
$$s_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{vu} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

where

$C_u, C_v = 1/\sqrt{2}$ for $u, v = 0$

$C_u, C_v = 1$ otherwise

otherwise.

A.3.4 DCT coefficient quantization (informative) and dequantization (normative)

After the FDCT is computed for a block, each of the 64 resulting DCT coefficients is quantized by a uniform quantizer. The quantizer step size for each coefficient S_{vu} is the value of the corresponding element Q_{vu} from the quantization table specified by the frame parameter Tq_i (see B.2.2).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The uniform quantizer is defined by the following equation. Rounding is to the nearest integer:

$$Sq_{vu} = round\left(\frac{S_{vu}}{Q_{vu}}\right)$$

Sq_{vu} is the quantized DCT coefficient, normalized by the quantizer step size.

NOTE – This equation contains a term which may not be represented with perfect accuracy by any real implementation. The accuracy requirements for the combined FDCT and quantization procedures are specified in Part 2 of this Specification.

At the decoder, this normalization is removed by the following equation, which defines dequantization:

$$R_{vu} = Sq_{vu} \times Q_{vu}$$

NOTE – Depending on the rounding used in quantization, it is possible that the dequantized coefficient may be outside the expected range.

The relationship among samples, DCT coefficients, and quantization is illustrated in Figure A.5.

A.3.5 Differential DC encoding

After quantization, and in preparation for entropy encoding, the quantized DC coefficient Sq_{00} is treated separately from the 63 quantized AC coefficients. The value that shall be encoded is the difference (DIFF) between the quantized DC coefficient of the current block (DC_i which is also designated as Sq_{00}) and that of the previous block of the same component (PRED):

$$DIFF = DC_i - PRED$$

A.3.6 Zig-zag sequence

After quantization, and in preparation for entropy encoding, the quantized AC coefficients are converted to the zig-zag sequence. The quantized DC coefficient (coefficient zero in the array) is treated separately, as defined in A.3.5. The zig-zag sequence is specified in Figure A.6.

A.4 Point transform

For various procedures data may be optionally divided by a power of 2 by a point transform prior to coding. There are three processes which require a point transform: lossless coding, lossless differential frame coding in the hierarchical mode, and successive approximation coding in the progressive DCT mode.

In the lossless mode of operation the point transform is applied to the input samples. In the difference coding of the hierarchical mode of operation the point transform is applied to the difference between the input component samples and the reference component samples. In both cases the point transform is an integer divide by 2^{Pt}, where Pt is the value of the point transform parameter (see B.2.3).

In successive approximation coding the point transform for the AC coefficients is an integer divide by 2^{Al}, where Al is the successive approximation bit position, low (see B.2.3). The point transform for the DC coefficients is an arithmetic-shift-right by Al bits. This is equivalent to dividing by 2^{Pt} before the level shift (see A.3.1).

The output of the decoder is rescaled by multiplying by 2^{Pt}. An example of the point transform is given in K.10.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

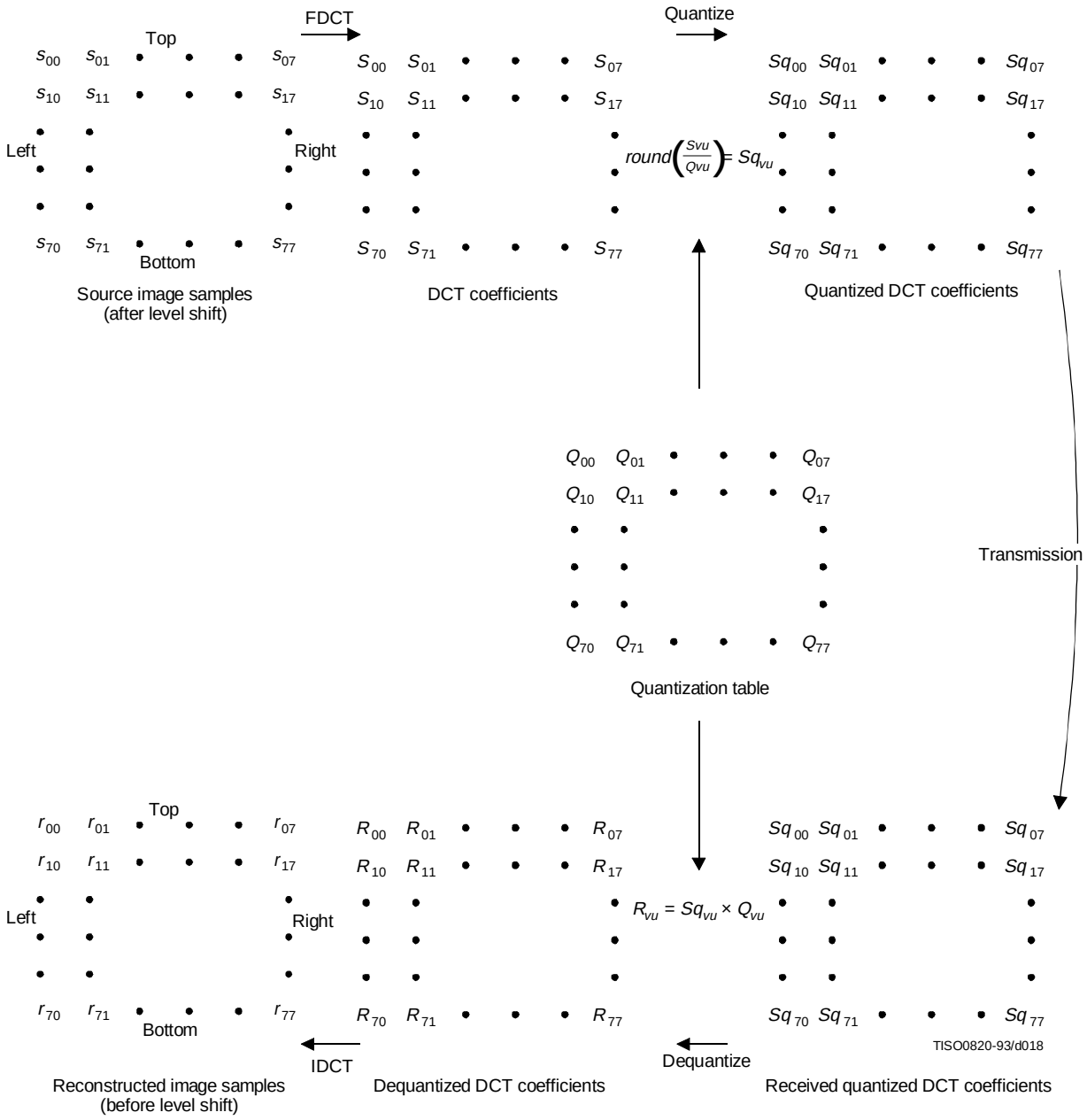


Figure A.5 – Relationship between 8 × 8-block samples and DCT coefficients

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure A.6 – Zig-zag sequence of quantized DCT coefficients

A.5 Arithmetic procedures in lossless and hierarchical modes of operation

In the lossless mode of operation predictions are calculated with full precision and without clamping of either overflow or underflow beyond the range of values allowed by the precision of the input. However, the division by two which is part of some of the prediction calculations shall be approximated by an arithmetic-shift-right by one bit.

The two's complement differences which are coded in either the lossless mode of operation or the differential frame coding in the hierarchical mode of operation are calculated modulo 65 536, thereby restricting the precision of these differences to a maximum of 16 bits. The modulo values are calculated by performing the logical AND operation of the two's complement difference with X'FFFF'. For purposes of coding, the result is still interpreted as a 16 bit two's complement difference. Modulo 65 536 arithmetic is also used in the decoder in calculating the output from the sum of the prediction and this two's complement difference.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex B

Compressed data formats

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies three compressed data formats:

a) the interchange format, specified in B.2 and B.3;
b) the abbreviated format for compressed image data, specified in B.4;
c) the abbreviated format for table-specification data, specified in B.5.

B.1 describes the constituent parts of these formats. B.1.3 and B.1.4 give the conventions for symbols and figures used in the format specifications.

B.1 General aspects of the compressed data format specifications

Structurally, the compressed data formats consist of an ordered collection of parameters, markers, and entropy-coded data segments. Parameters and markers in turn are often organized into marker segments. Because all of these constituent parts are represented with byte-aligned codes, each compressed data format consists of an ordered sequence of 8-bit bytes. For each byte, a most significant bit (MSB) and a least significant bit (LSB) are defined.

B.1.1 Constituent parts

This subclause gives a general description of each of the constituent parts of the compressed data format.

B.1.1.1 Parameters

Parameters are integers, with values specific to the encoding process, source image characteristics, and other features selectable by the application. Parameters are assigned either 4-bit, 1-byte, or 2-byte codes. Except for certain optional groups of parameters, parameters encode critical information without which the decoding process cannot properly reconstruct the image.

The code assignment for a parameter shall be an unsigned integer of the specified length in bits with the particular value of the parameter.

For parameters which are 2 bytes (16 bits) in length, the most significant byte shall come first in the compressed data's ordered sequence of bytes. Parameters which are 4 bits in length always come in pairs, and the pair shall always be encoded in a single byte. The first 4-bit parameter of the pair shall occupy the most significant 4 bits of the byte. Within any 16-, 8-, or 4-bit parameter, the MSB shall come first and LSB shall come last.

B.1.1.2 Markers

Markers serve to identify the various structural parts of the compressed data formats. Most markers start marker segments containing a related group of parameters; some markers stand alone. All markers are assigned two-byte codes: an X'FF' byte followed by a byte which is not equal to 0 or X'FF' (see Table B.1). Any marker may optionally be preceded by any number of fill bytes, which are bytes assigned code X'FF'.

NOTE – Because of this special code-assignment structure, markers make it possible for a decoder to parse the compressed data and locate its various parts without having to decode other segments of image data.

B.1.1.3 Marker assignments

All markers shall be assigned two-byte codes: a X'FF' byte followed by a second byte which is not equal to 0 or X'FF'. The second byte is specified in Table B.1 for each defined marker. An asterisk (*) indicates a marker which stands alone, that is, which is not the start of a marker segment.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table B.1 – Marker code assignments

Code Assignment	Symbol	Description
Start Of Frame markers, non-differential, Huffman coding		
X'FFC0' X'FFC1' X'FFC2' X'FFC3'	SOF_0 SOF_1 SOF_2 SOF_3	Baseline DCT Extended sequential DCT Progressive DCT Lossless (sequential)
Start Of Frame markers, differential, Huffman coding		
X'FFC5' X'FFC6' X'FFC7'	SOF_5 SOF_6 SOF_7	Differential sequential DCT Differential progressive DCT Differential lossless (sequential)
Start Of Frame markers, non-differential, arithmetic coding		
X'FFC8' X'FFC9' X'FFCA' X'FFCB'	JPG SOF_9 SOF_{10} SOF_{11}	Reserved for JPEG extensions Extended sequential DCT Progressive DCT Lossless (sequential)
Start Of Frame markers, differential, arithmetic coding		
X'FFCD' X'FFCE' X'FFCF'	SOF_{13} SOF_{14} SOF_{15}	Differential sequential DCT Differential progressive DCT Differential lossless (sequential)
Huffman table specification		
X'FFC4'	DHT	Define Huffman table(s)
Arithmetic coding conditioning specification		
X'FFCC'	DAC	Define arithmetic coding conditioning(s)
Restart interval termination		
X'FFD0' through X'FFD7'	RST_m*	Restart with modulo 8 count "m"
Other markers		
X'FFD8' X'FFD9' X'FFDA' X'FFDB' X'FFDC' X'FFDD' X'FFDE' X'FFDF' X'FFE0' through X'FFEF' X'FFF0' through X'FFFD' X'FFFE'	SOI* EOI* SOS DQT DNL DRI DHP EXP APP_n JPG_n COM	Start of image End of image Start of scan Define quantization table(s) Define number of lines Define restart interval Define hierarchical progression Expand reference component(s) Reserved for application segments Reserved for JPEG extensions Comment
Reserved markers		
X'FF01' X'FF02' through X'FFBF'	TEM* RES	For temporary private use in arithmetic coding Reserved

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B.1.1.4 Marker segments

A marker segment consists of a marker followed by a sequence of related parameters. The first parameter in a marker segment is the two-byte length parameter. This length parameter encodes the number of bytes in the marker segment, including the length parameter and excluding the two-byte marker. The marker segments identified by the SOF and SOS marker codes are referred to as headers: the frame header and the scan header respectively.

B.1.1.5 Entropy-coded data segments

An entropy-coded data segment contains the output of an entropy-coding procedure. It consists of an integer number of bytes, whether the entropy-coding procedure used is Huffman or arithmetic.

NOTES

1 Making entropy-coded segments an integer number of bytes is performed as follows: for Huffman coding, 1-bits are used, if necessary, to pad the end of the compressed data to complete the final byte of a segment. For arithmetic coding, byte alignment is performed in the procedure which terminates the entropy-coded segment (see D.1.8).

2 In order to ensure that a marker does not occur within an entropy-coded segment, any X'FF' byte generated by either a Huffman or arithmetic encoder, or an X'FF' byte that was generated by the padding of 1-bits described in NOTE 1 above, is followed by a "stuffed" zero byte (see D.1.6 and F.1.2.3).

B.1.2 Syntax

In B.2 and B.3 the interchange format syntax is specified. For the purposes of this Specification, the syntax specification consists of:

- the required ordering of markers, parameters, and entropy-coded segments;
- identification of optional or conditional constituent parts;
- the name, symbol, and definition of each marker and parameter;
- the allowed values of each parameter;
- any restrictions on the above which are specific to the various coding processes.

The ordering of constituent parts and the identification of which are optional or conditional is specified by the syntax figures in B.2 and B.3. Names, symbols, definitions, allowed values, conditions, and restrictions are specified immediately below each syntax figure.

B.1.3 Conventions for syntax figures

The syntax figures in B.2 and B.3 are a part of the interchange format specification. The following conventions, illustrated in Figure B.1, apply to these figures:

- **parameter/marker indicator:** A thin-lined box encloses either a marker or a single parameter;
- **segment indicator:** A thick-lined box encloses either a marker segment, an entropy-coded data segment, or combinations of these;
- **parameter length indicator:** The width of a thin-lined box is proportional to the parameter length (4, 8, or 16 bits, shown as E, B, and D respectively in Figure B.1) of the marker or parameter it encloses; the width of thick-lined boxes is not meaningful;
- **optional/conditional indicator:** Square brackets indicate that a marker or marker segment is only optionally or conditionally present in the compressed image data;
- **ordering:** In the interchange format a parameter or marker shown in a figure precedes all of those shown to its right, and follows all of those shown to its left;
- **entropy-coded data indicator:** Angled brackets indicate that the entity enclosed has been entropy encoded.

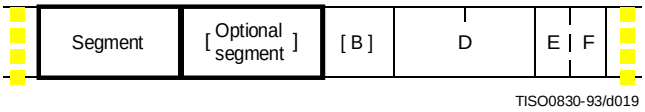


Figure B.1 – Syntax notation conventions

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B.1.4 Conventions for symbols, code lengths, and values

Following each syntax figure in B.2 and B.3, the symbol, name, and definition for each marker and parameter shown in the figure are specified. For each parameter, the length and allowed values are also specified in tabular form.

The following conventions apply to symbols for markers and parameters:

- all marker symbols have three upper-case letters, and some also have a subscript. Examples: SOI, SOF_n;
- all parameter symbols have one upper-case letter; some also have one lower-case letter and some have subscripts. Examples: Y, Nf, H_i, Tq_i.

B.2 General sequential and progressive syntax

This clause specifies the interchange format syntax which applies to all coding processes for sequential DCT-based, progressive DCT-based, and lossless modes of operation.

B.2.1 High-level syntax

Figure B.2 specifies the order of the high-level constituent parts of the interchange format for all non-hierarchical encoding processes specified in this Specification.

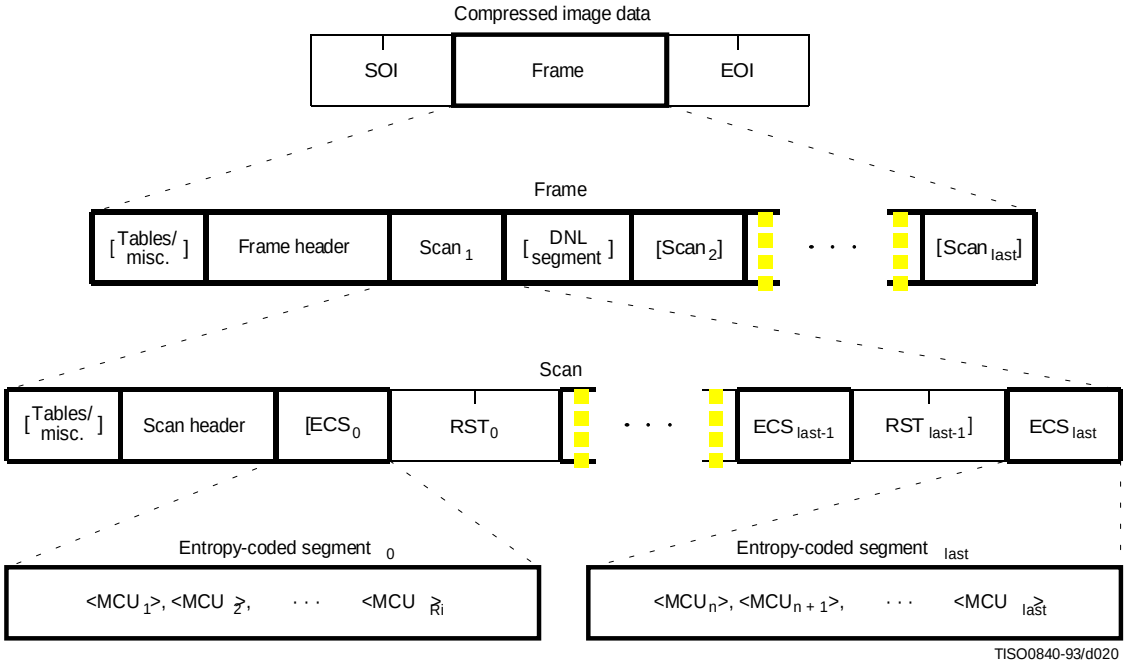


Figure B.2 – Syntax for sequential DCT-based, progressive DCT-based, and lossless modes of operation

The three markers shown in Figure B.2 are defined as follows:

SOI: Start of image marker – Marks the start of a compressed image represented in the interchange format or abbreviated format.

EOI: End of image marker – Marks the end of a compressed image represented in the interchange format or abbreviated format.

RST_m: Restart marker – A conditional marker which is placed between entropy-coded segments only if restart is enabled. There are 8 unique restart markers (m = 0 - 7) which repeat in sequence from 0 to 7, starting with zero for each scan, to provide a modulo 8 restart interval count.

The top level of Figure B.2 specifies that the non-hierarchical interchange format shall begin with an SOI marker, shall contain one frame, and shall end with an EOI marker.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The second level of Figure B.2 specifies that a frame shall begin with a frame header and shall contain one or more scans. A frame header may be preceded by one or more table-specification or miscellaneous marker segments as specified in B.2.4. If a DNL segment (see B.2.5) is present, it shall immediately follow the first scan.

For sequential DCT-based and lossless processes each scan shall contain from one to four image components. If two to four components are contained within a scan, they shall be interleaved within the scan. For progressive DCT-based processes each image component is only partially contained within any one scan. Only the first scan(s) for the components (which contain only DC coefficient data) may be interleaved.

The third level of Figure B.2 specifies that a scan shall begin with a scan header and shall contain one or more entropy-coded data segments. Each scan header may be preceded by one or more table-specification or miscellaneous marker segments. If restart is not enabled, there shall be only one entropy-coded segment (the one labeled “last”), and no restart markers shall be present. If restart is enabled, the number of entropy-coded segments is defined by the size of the image and the defined restart interval. In this case, a restart marker shall follow each entropy-coded segment except the last one.

The fourth level of Figure B.2 specifies that each entropy-coded segment is comprised of a sequence of entropy-coded MCUs. If restart is enabled and the restart interval is defined to be Ri, each entropy-coded segment except the last one shall contain Ri MCUs. The last one shall contain whatever number of MCUs completes the scan.

Figure B.2 specifies the locations where table-specification segments **may** be present. However, this Specification hereby specifies that the interchange format **shall** contain all table-specification data necessary for decoding the compressed image. Consequently, the required table-specification data **shall** be present at one or more of the allowed locations.

B.2.2 Frame header syntax

Figure B.3 specifies the frame header which shall be present at the start of a frame. This header specifies the source image characteristics (see A.1), the components in the frame, and the sampling factors for each component, and specifies the destinations from which the quantized tables to be used with each component are retrieved.

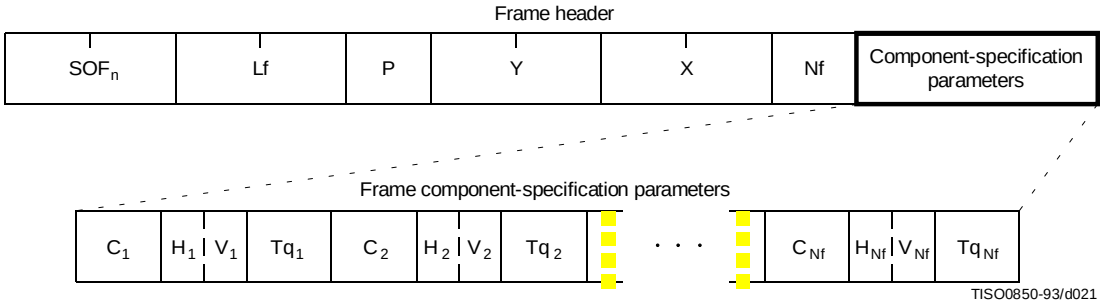


Figure B.3 – Frame header syntax

The markers and parameters shown in Figure B.3 are defined below. The size and allowed values of each parameter are given in Table B.2. In Table B.2 (and similar tables which follow), value choices are separated by commas (e.g. 8, 12) and inclusive bounds are separated by dashes (e.g. 0 - 3).

SOF_n: Start of frame marker – Marks the beginning of the frame parameters. The subscript n identifies whether the encoding process is baseline sequential, extended sequential, progressive, or lossless, as well as which entropy encoding procedure is used.

SOF_0: Baseline DCT

SOF_1: Extended sequential DCT, Huffman coding

SOF_2: Progressive DCT, Huffman coding

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

SOF_3: Lossless (sequential), Huffman coding

SOF_9: Extended sequential DCT, arithmetic coding

SOF_{10}: Progressive DCT, arithmetic coding

SOF_{11}: Lossless (sequential), arithmetic coding

Lf: Frame header length – Specifies the length of the frame header shown in Figure B.3 (see B.1.1.4).

P: Sample precision – Specifies the precision in bits for the samples of the components in the frame.

Y: Number of lines – Specifies the maximum number of lines in the source image. This shall be equal to the number of lines in the component with the maximum number of vertical samples (see A.1.1). Value 0 indicates that the number of lines shall be defined by the DNL marker and parameters at the end of the first scan (see B.2.5).

X: Number of samples per line – Specifies the maximum number of samples per line in the source image. This shall be equal to the number of samples per line in the component with the maximum number of horizontal samples (see A.1.1).

Nf: Number of image components in frame – Specifies the number of source image components in the frame. The value of Nf shall be equal to the number of sets of frame component specification parameters (C_i, H_i, V_i, and Tq_i) present in the frame header.

C_i: Component identifier – Assigns a unique label to the *i*th component in the sequence of frame component specification parameters. These values shall be used in the scan headers to identify the components in the scan. The value of C_i shall be different from the values of C_1 through C_{i-1}.

H_i: Horizontal sampling factor – Specifies the relationship between the component horizontal dimension and maximum image dimension X (see A.1.1); also specifies the number of horizontal data units of component C_i in each MCU, when more than one component is encoded in a scan.

V_i: Vertical sampling factor – Specifies the relationship between the component vertical dimension and maximum image dimension Y (see A.1.1); also specifies the number of vertical data units of component C_i in each MCU, when more than one component is encoded in a scan.

Tq_i: Quantization table destination selector – Specifies one of four possible quantization table destinations from which the quantization table to use for dequantization of DCT coefficients of component C_i is retrieved. If the decoding process uses the dequantization procedure, this table shall have been installed in this destination by the time the decoder is ready to decode the scan(s) containing component C_i. The destination shall not be re-specified, or its contents changed, until all scans containing C_i have been completed.

Table B.2 – Frame header parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lf	16	$8 + 3 \times Nf$			
P	8	8	8, 12	8, 12	2-16
Y	16	0-65 535			
X	16	1-65 535			
Nf	8	1-255	1-255	1-4	1-255
C_i	8	0-255			
H_i	4	1-4			
V_i	4	1-4			
Tq_i	8	0-3	0-3	0-3	0

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B.2.3 Scan header syntax

Figure B.4 specifies the scan header which shall be present at the start of a scan. This header specifies which component(s) are contained in the scan, specifies the destinations from which the entropy tables to be used with each component are retrieved, and (for the progressive DCT) which part of the DCT quantized coefficient data is contained in the scan. For lossless processes the scan parameters specify the predictor and the point transform.

NOTE – If there is only one image component present in a scan, that component is, by definition, non-interleaved. If there is more than one image component present in a scan, the components present are, by definition, interleaved.

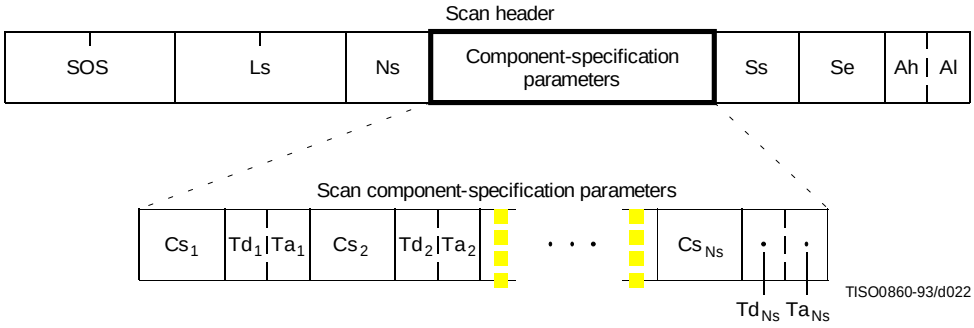


Figure B.4 – Scan header syntax

The marker and parameters shown in Figure B.4 are defined below. The size and allowed values of each parameter are given in Table B.3.

SOS: Start of scan marker – Marks the beginning of the scan parameters.

Ls: Scan header length – Specifies the length of the scan header shown in Figure B.4 (see B.1.1.4).

Ns: Number of image components in scan – Specifies the number of source image components in the scan. The value of Ns shall be equal to the number of sets of scan component specification parameters (Cs_j, Td_j, and Ta_j) present in the scan header.

Cs_j: Scan component selector – Selects which of the Nf image components specified in the frame parameters shall be the *j*th component in the scan. Each Cs_j shall match one of the C_i values specified in the frame header, and the ordering in the scan header shall follow the ordering in the frame header. If Ns > 1, the order of interleaved components in the MCU is Cs_1 first, Cs_2 second, etc. If Ns > 1, the following restriction shall be placed on the image components contained in the scan:

$$\sum_{j=1}^{N_s} H_j \times V_j \leq 10,$$

where H_j and V_j are the horizontal and vertical sampling factors for scan component j. These sampling factors are specified in the frame header for component i, where i is the frame component specification index for which frame component identifier C_i matches scan component selector Cs_j.

As an example, consider an image having 3 components with maximum dimensions of 512 lines and 512 samples per line, and with the following sampling factors:

Component 0	$H_0 = 4,$	$V_0 = 1$
Component 1	$H_1 = 1,$	$V_1 = 2$
Component 2	$H_2 = 2$	$V_2 = 2$

Then the summation of $H_j \times V_j$ is $(4 \times 1) + (1 \times 2) + (2 \times 2) = 10$.

The value of Cs_j shall be different from the values of Cs_1 to Cs_{j-1}.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Td_j: DC entropy coding table destination selector – Specifies one of four possible DC entropy coding table destinations from which the entropy table needed for decoding of the DC coefficients of component Cs_j is retrieved. The DC entropy table shall have been installed in this destination (see B.2.4.2 and B.2.4.3) by the time the decoder is ready to decode the current scan. This parameter specifies the entropy coding table destination for the lossless processes.

Ta_j: AC entropy coding table destination selector – Specifies one of four possible AC entropy coding table destinations from which the entropy table needed for decoding of the AC coefficients of component Cs_j is retrieved. The AC entropy table selected shall have been installed in this destination (see B.2.4.2 and B.2.4.3) by the time the decoder is ready to decode the current scan. This parameter is zero for the lossless processes.

Ss: Start of spectral or predictor selection – In the DCT modes of operation, this parameter specifies the first DCT coefficient in each block in zig-zag order which shall be coded in the scan. This parameter shall be set to zero for the sequential DCT processes. In the lossless mode of operations this parameter is used to select the predictor.

Se: End of spectral selection – Specifies the last DCT coefficient in each block in zig-zag order which shall be coded in the scan. This parameter shall be set to 63 for the sequential DCT processes. In the lossless mode of operations this parameter has no meaning. It shall be set to zero.

Ah: Successive approximation bit position high – This parameter specifies the point transform used in the preceding scan (i.e. successive approximation bit position low in the preceding scan) for the band of coefficients specified by Ss and Se. This parameter shall be set to zero for the first scan of each band of coefficients. In the lossless mode of operations this parameter has no meaning. It shall be set to zero.

Al: Successive approximation bit position low or point transform – In the DCT modes of operation this parameter specifies the point transform, i.e. bit position low, used before coding the band of coefficients specified by Ss and Se. This parameter shall be set to zero for the sequential DCT processes. In the lossless mode of operations, this parameter specifies the point transform, Pt.

The entropy coding table destination selectors, Td_j and Ta_j, specify either Huffman tables (in frames using Huffman coding) or arithmetic coding tables (in frames using arithmetic coding). In the latter case the entropy coding table destination selector specifies both an arithmetic coding conditioning table destination and an associated statistics area.

Table B.3 – Scan header parameter size and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Ls	16	$6 + 2 \times Ns$			
Ns	8	1-4			
Cs_j	8	0-255[a)]			
Td_j	4	0-1	0-3	0-3	0-3
Ta_j	4	0-1	0-3	0-3	0
Ss	8	0	0	0-63	1-7[b)]
Se	8	63	63	Ss-63[c)]	0
Ah	4	0	0	0-13	0
Al	4	0	0	0-13	0-15

a) Cs_j shall be a member of the set of C_i specified in the frame header.

b) 0 for lossless differential frames in the hierarchical mode (see B.3).

c) 0 if Ss equals zero.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B.2.4 Table-specification and miscellaneous marker segment syntax

Figure B.5 specifies that, at the places indicated in Figure B.2, any of the table-specification segments or miscellaneous marker segments specified in B.2.4.1 through B.2.4.6 may be present in any order and with no limit on the number of segments.

If any table specification for a particular destination occurs in the compressed image data, it shall replace any previous table specified for this destination, and shall be used whenever this destination is specified in the remaining scans in the frame or subsequent images represented in the abbreviated format for compressed image data. If a table specification for a given destination occurs more than once in the compressed image data, each specification shall replace the previous specification. The quantization table specification shall not be altered between progressive DCT scans of a given component.

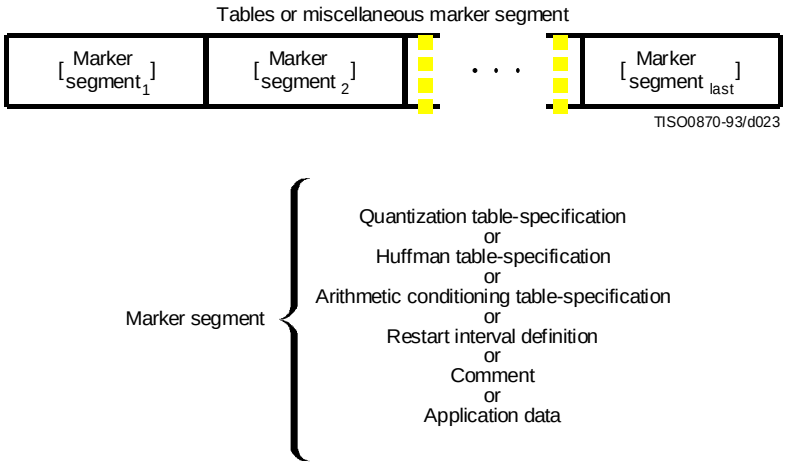


Figure B.5 – Tables/miscellaneous marker segment syntax

B.2.4.1 Quantization table-specification syntax

Figure B.6 specifies the marker segment which defines one or more quantization tables.

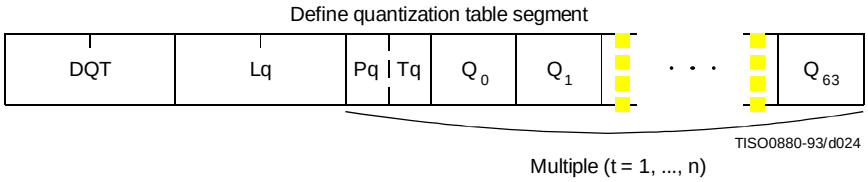


Figure B.6 – Quantization table syntax

The marker and parameters shown in Figure B.6 are defined below. The size and allowed values of each parameter are given in Table B.4.

DQT: Define quantization table marker – Marks the beginning of quantization table-specification parameters.

Lq: Quantization table definition length – Specifies the length of all quantization table parameters shown in Figure B.6 (see B.1.1.4).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Pq: Quantization table element precision – Specifies the precision of the Q_k values. Value 0 indicates 8-bit Q_k values; value 1 indicates 16-bit Q_k values. Pq shall be zero for 8 bit sample precision P (see B.2.2).

Tq: Quantization table destination identifier – Specifies one of four possible destinations at the decoder into which the quantization table shall be installed.

Q_k: Quantization table element – Specifies the *k*th element out of 64 elements, where *k* is the index in the zig-zag ordering of the DCT coefficients. The quantization elements shall be specified in zig-zag scan order.

Table B.4 – Quantization table-specification parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lq	16	$2 + \sum_{t=1}^{n} (65 + 64 \times Pq(t))$			Undefined
Pq	4	0	0, 1	0, 1	Undefined
Tq	4	0-3			Undefined
Q_k	8, 16	1-255, 1-65 535			Undefined

The value n in Table B.4 is the number of quantization tables specified in the DQT marker segment.

Once a quantization table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used, when referenced, in the remaining scans of the current image and in subsequent images represented in the abbreviated format for compressed image data. If a table has never been defined for a particular destination, then when this destination is specified in a frame header, the results are unpredictable.

An 8-bit DCT-based process shall not use a 16-bit precision quantization table.

B.2.4.2 Huffman table-specification syntax

Figure B.7 specifies the marker segment which defines one or more Huffman table specifications.

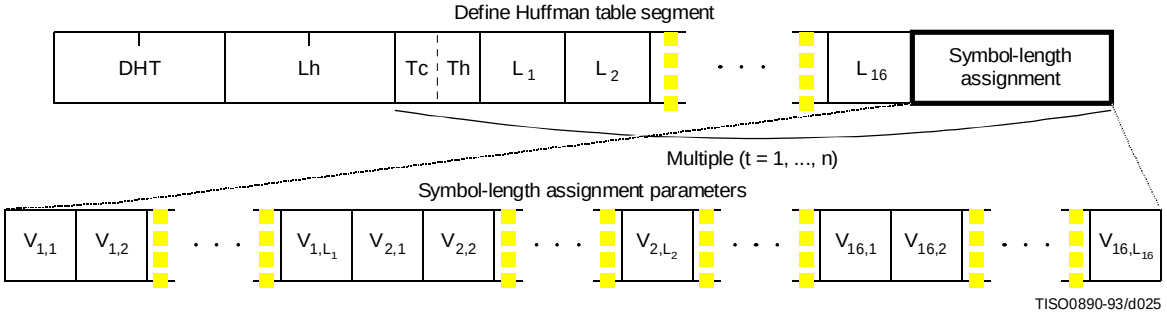


Figure B.7 – Huffman table syntax

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The marker and parameters shown in Figure B.7 are defined below. The size and allowed values of each parameter are given in Table B.5.

DHT: Define Huffman table marker – Marks the beginning of Huffman table definition parameters.

Lh: Huffman table definition length – Specifies the length of all Huffman table parameters shown in Figure B.7 (see B.1.1.4).

Tc: Table class – 0 = DC table or lossless table, 1 = AC table.

Th: Huffman table destination identifier – Specifies one of four possible destinations at the decoder into which the Huffman table shall be installed.

L_i: Number of Huffman codes of length i – Specifies the number of Huffman codes for each of the 16 possible lengths allowed by this Specification. L_i's are the elements of the list BITS.

$V_{i,j}$: Value associated with each Huffman code – Specifies, for each i, the value associated with each Huffman code of length i. The meaning of each value is determined by the Huffman coding model. The $V_{i,j}$'s are the elements of the list HUFFVAL.

Table B.5 – Huffman table specification parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lh	16	$2 + \sum_{t=1}^{n} (17 + m_t)$			
Tc	4	0, 1			0
Th	4	0, 1	0-3		
L_i	8	0-255			
$V_{i,j}$	8	0-255			

The value n in Table B.5 is the number of Huffman tables specified in the DHT marker segment. The value m_t is the number of parameters which follow the 16 $L_i(t)$ parameters for Huffman table t, and is given by:

$$m_t = \sum_{i=1}^{16} L_i$$

In general, m_t is different for each table.

Once a Huffman table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used when referenced, in the remaining scans of the current image and in subsequent images represented in the abbreviated format for compressed image data. If a table has never been defined for a particular destination, then when this destination is specified in a scan header, the results are unpredictable.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B.2.4.3 Arithmetic conditioning table-specification syntax

Figure B.8 specifies the marker segment which defines one or more arithmetic coding conditioning table specifications. These replace the default arithmetic coding conditioning tables established by the SOI marker for arithmetic coding processes. (See F.1.4.4.1.4 and F.1.4.4.2.1.)

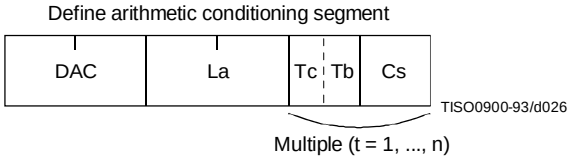


Figure B.8 – Arithmetic conditioning table-specification syntax

The marker and parameters shown in Figure B.8 are defined below. The size and allowed values of each parameter are given in Table B.6.

DAC: Define arithmetic coding conditioning marker – Marks the beginning of the definition of arithmetic coding conditioning parameters.

La: Arithmetic coding conditioning definition length – Specifies the length of all arithmetic coding conditioning parameters shown in Figure B.8 (see B.1.1.4).

Tc: Table class – 0 = DC table or lossless table, 1 = AC table.

Tb: Arithmetic coding conditioning table destination identifier – Specifies one of four possible destinations at the decoder into which the arithmetic coding conditioning table shall be installed.

Cs: Conditioning table value – Value in either the AC or the DC (and lossless) conditioning table. A single value of Cs shall follow each value of Tb. For AC conditioning tables Tc shall be one and Cs shall contain a value of Kx in the range $1 \le Kx \le 63$. For DC (and lossless) conditioning tables Tc shall be zero and Cs shall contain two 4-bit parameters, U and L. U and L shall be in the range $0 \le L \le U \le 15$ and the value of Cs shall be $L + 16 \times U$.

The value n in Table B.6 is the number of arithmetic coding conditioning tables specified in the DAC marker segment. The parameters L and U are the lower and upper conditioning bounds used in the arithmetic coding procedures defined for DC coefficient coding and lossless coding. The separate value range 1-63 listed for DCT coding is the Kx conditioning used in AC coefficient coding.

Table B.6 – Arithmetic coding conditioning table-specification parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
La	16	Undefined	$2 + 2 \times n$		
Tc	4	Undefined	0, 1		0
Tb	4	Undefined	0-3		
Cs	8	Undefined	0-255 (Tc = 0), 1-63 (Tc = 1)		0-255

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B.2.4.4 Restart interval definition syntax

Figure B.9 specifies the marker segment which defines the restart interval.

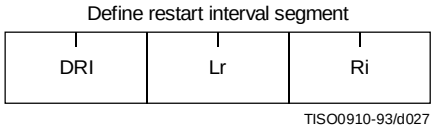


Figure B.9 – Restart interval definition syntax

The marker and parameters shown in Figure B.9 are defined below. The size and allowed values of each parameter are given in Table B.7.

DRI: Define restart interval marker – Marks the beginning of the parameters which define the restart interval.

Lr: Define restart interval segment length – Specifies the length of the parameters in the DRI segment shown in Figure B.9 (see B.1.1.4).

Ri: Restart interval – Specifies the number of MCU in the restart interval.

In Table B.7 the value n is the number of rows of MCU in the restart interval. The value MCUR is the number of MCU required to make up one line of samples of each component in the scan. The SOI marker disables the restart intervals. A DRI marker segment with Ri nonzero shall be present to enable restart interval processing for the following scans. A DRI marker segment with Ri equal to zero shall disable restart intervals for the following scans.

Table B.7 – Define restart interval segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lr	16	4			
Ri	16	0-65 535			n × MCUR

B.2.4.5 Comment syntax

Figure B.10 specifies the marker segment structure for a comment segment.

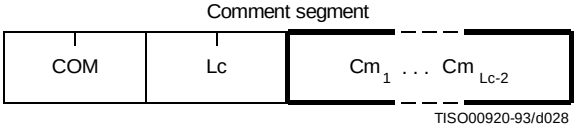


Figure B.10 – Comment segment syntax

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The marker and parameters shown in Figure B.10 are defined below. The size and allowed values of each parameter are given in Table B.8.

COM: Comment marker – Marks the beginning of a comment.

Lc: Comment segment length – Specifies the length of the comment segment shown in Figure B.10 (see B.1.1.4).

Cm_i: Comment byte – The interpretation is left to the application.

Table B.8 – Comment segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lc	16	2-65 535			
Cm_i	8	0-255			

B.2.4.6 Application data syntax

Figure B.11 specifies the marker segment structure for an application data segment.

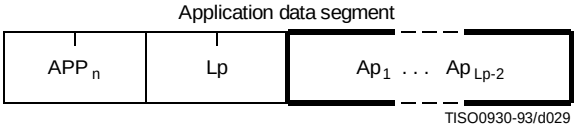


Figure B.11 – Application data syntax

The marker and parameters shown in Figure B.11 are defined below. The size and allowed values of each parameter are given in Table B.9.

APP_n: Application data marker – Marks the beginning of an application data segment.

Lp: Application data segment length – Specifies the length of the application data segment shown in Figure B.11 (see B.1.1.4).

Ap_i: Application data byte – The interpretation is left to the application.

The APP_n (Application) segments are reserved for application use. Since these segments may be defined differently for different applications, they should be removed when the data are exchanged between application environments.

Table B.9 – Application data segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lp	16	2-65 535			
Ap_i	8	0-255			

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

B.2.5 Define number of lines syntax

Figure B.12 specifies the marker segment for defining the number of lines. The DNL (Define Number of Lines) segment provides a mechanism for defining or redefining the number of lines in the frame (the Y parameter in the frame header) at the end of the first scan. The value specified shall be consistent with the number of MCU-rows encoded in the first scan. This segment, if used, shall only occur at the end of the first scan, and only after coding of an integer number of MCU-rows. This marker segment is mandatory if the number of lines (Y) specified in the frame header has the value zero.

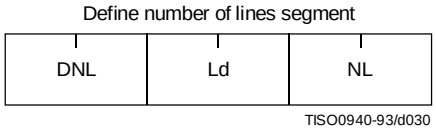


Figure B.12 – Define number of lines syntax

The marker and parameters shown in Figure B.12 are defined below. The size and allowed values of each parameter are given in Table B.10.

DNL: Define number of lines marker – Marks the beginning of the define number of lines segment.

Ld: Define number of lines segment length – Specifies the length of the define number of lines segment shown in Figure B.12 (see B.1.1.4).

NL: Number of lines – Specifies the number of lines in the frame (see definition of Y in B.2.2).

Table B.10 – Define number of lines segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Ld	16	4			
NL	16	1-65 535[a)]			

a) The value specified shall be consistent with the number of lines coded at the point where the DNL segment terminates the compressed data segment.

B.3 Hierarchical syntax

B.3.1 High level hierarchical mode syntax

Figure B.13 specifies the order of the high level constituent parts of the interchange format for hierarchical encoding processes.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

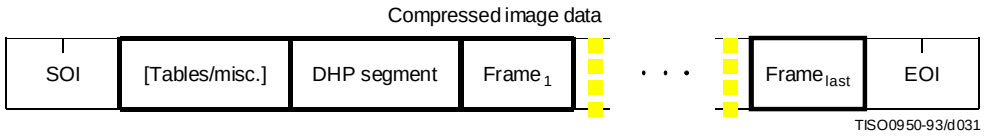


Figure B.13 – Syntax for the hierarchical mode of operation

Hierarchical mode syntax requires a DHP marker segment that appears before the non-differential frame or frames. The hierarchical mode compressed image data may include EXP marker segments and differential frames which shall follow the initial non-differential frame. The frame structure in hierarchical mode is identical to the frame structure in non-hierarchical mode.

The non-differential frames in the hierarchical sequence shall use one of the coding processes specified for SOF_n markers: SOF_0, SOF_1, SOF_2, SOF_3, SOF_9, SOF_{10} and SOF_{11}. The differential frames shall use one of the processes specified for SOF_5, SOF_6, SOF_7, SOF_{13}, SOF_{14} and SOF_{15}. The allowed combinations of SOF markers within one hierarchical sequence are specified in Annex J.

The sample precision (P) shall be constant for all frames and have the identical value as that coded in the DHP marker segment. The number of samples per line (X) for all frames shall not exceed the value coded in the DHP marker segment. If the number of lines (Y) is non-zero in the DHP marker segment, then the number of lines for all frames shall not exceed the value in the DHP marker segment.

B.3.2 DHP segment syntax

The DHP segment defines the image components, size, and sampling factors for the completed hierarchical sequence of frames. The DHP segment shall precede the first frame; a single DHP segment shall occur in the compressed image data.

The DHP segment structure is identical to the frame header syntax, except that the DHP marker is used instead of the SOF_n marker. The figures and description of B.2.2 then apply, except that the quantization table destination selector parameter shall be set to zero in the DHP segment.

B.3.3 EXP segment syntax

Figure B.14 specifies the marker segment structure for the EXP segment. The EXP segment shall be present if (and only if) expansion of the reference components is required either horizontally or vertically. The EXP segment parameters apply only to the next frame (which shall be a differential frame) in the image. If required, the EXP segment shall be one of the table-specification segments or miscellaneous marker segments preceding the frame header; the EXP segment shall not be one of the table-specification segments or miscellaneous marker segments preceding a scan header or a DHP marker segment.

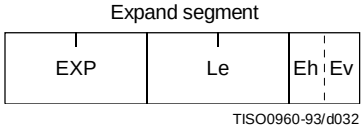


Figure B.14 – Syntax of the expand segment

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The marker and parameters shown in Figure B.14 are defined below. The size and allowed values of each parameter are given in Table B.11.

EXP: Expand reference components marker – Marks the beginning of the expand reference components segment.

Le: Expand reference components segment length – Specifies the length of the expand reference components segment (see B.1.1.4).

Eh: Expand horizontally – If one, the reference components shall be expanded horizontally by a factor of two. If horizontal expansion is not required, the value shall be zero.

Ev: Expand vertically – If one, the reference components shall be expanded vertically by a factor of two. If vertical expansion is not required, the value shall be zero.

Both Eh and Ev shall be one if expansion is required both horizontally and vertically.

Table B.11 – Expand segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Le	16	3			
Eh	4	0, 1			
Ev	4	0, 1			

B.4 Abbreviated format for compressed image data

Figure B.2 shows the high-level constituent parts of the interchange format. This format includes all table specifications required for decoding. If an application environment provides methods for table specification other than by means of the compressed image data, some or all of the table specifications may be omitted. Compressed image data which is missing any table specification data required for decoding has the abbreviated format.

B.5 Abbreviated format for table-specification data

Figure B.2 shows the high-level constituent parts of the interchange format. If no frames are present in the compressed image data, the only purpose of the compressed image data is to convey table specifications or miscellaneous marker segments defined in B.2.4.1, B.2.4.2, B.2.4.5, and B.2.4.6. In this case the compressed image data has the abbreviated format for table specification data (see Figure B.15).

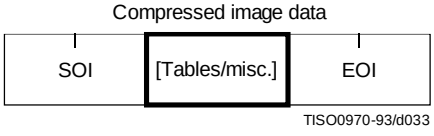


Figure B.15 – Abbreviated format for table-specification data syntax

B.6 Summary

The order of the constituent parts of interchange format and all marker segment structures is summarized in Figures B.16 and B.17. Note that in Figure B.16 double-lined boxes enclose marker segments. In Figures B.16 and B.17 thick-lined boxes enclose only markers.

The EXP segment can be mixed with the other tables/miscellaneous marker segments preceding the frame header but not with the tables/miscellaneous marker segments preceding the DHP segment or the scan header.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

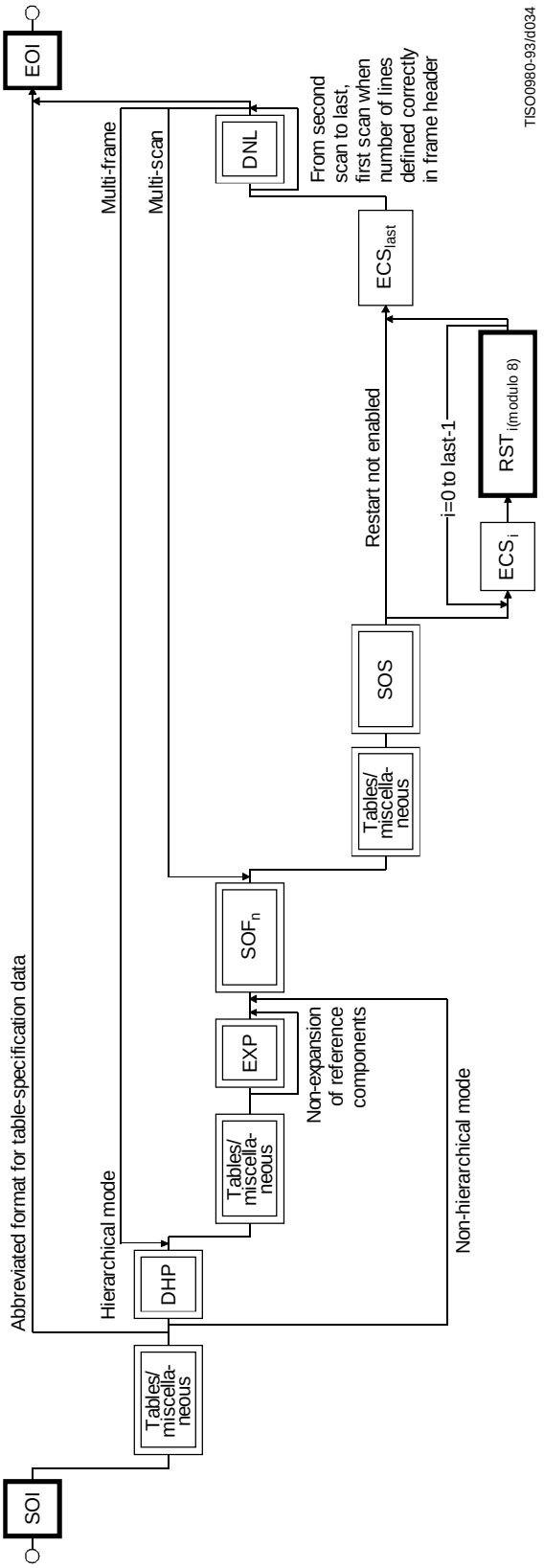


Figure B.16 – Flow of compressed data syntax

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

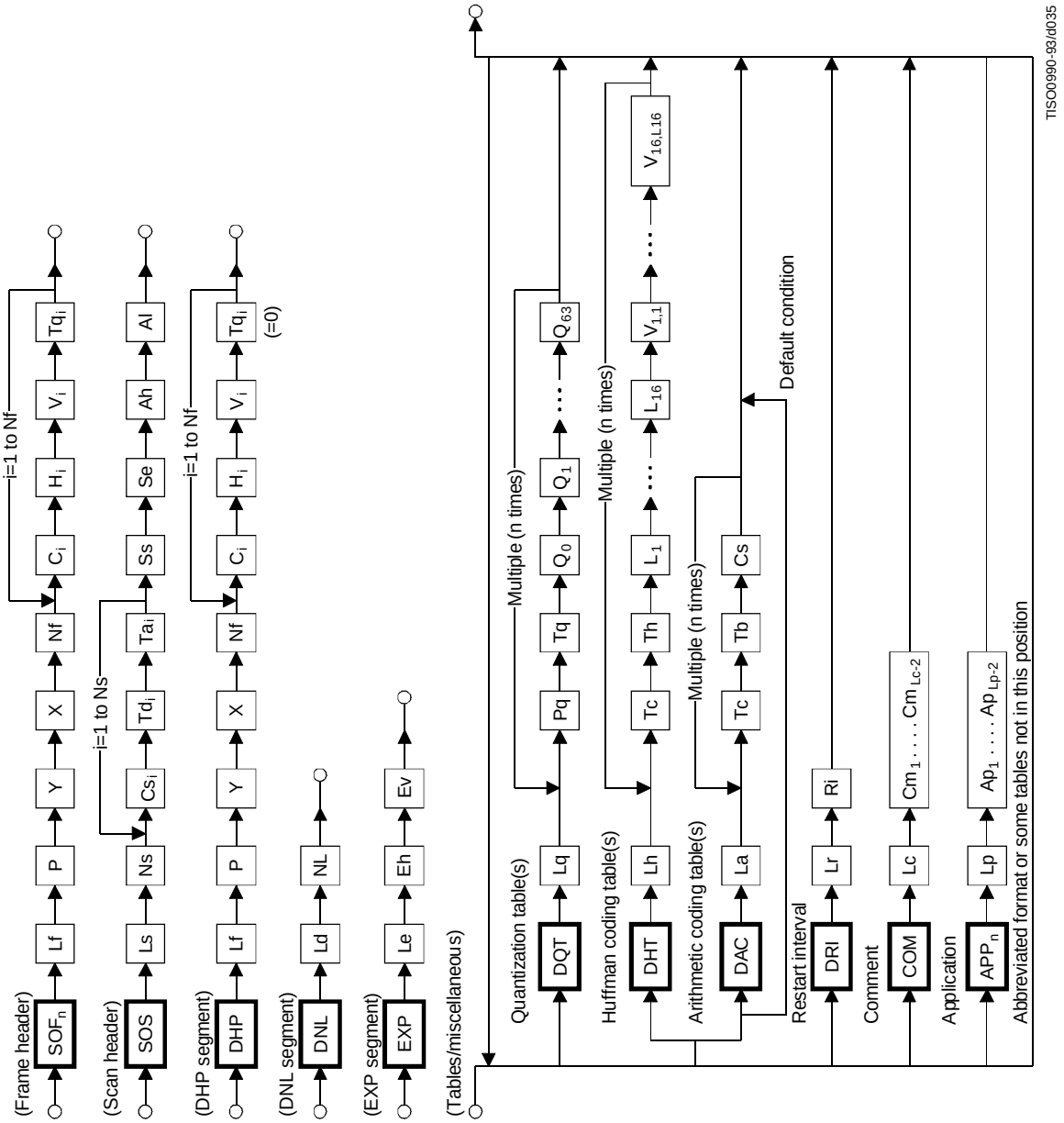


Figure B.17 – Flow of marker segment

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex C

Huffman table specification

(This annex forms an integral part of this Recommendation | International Standard)

A Huffman coding procedure may be used for entropy coding in any of the coding processes. Coding models for Huffman encoding are defined in Annexes F, G, and H. In this Annex, the Huffman table specification is defined.

Huffman tables are specified in terms of a 16-byte list (BITS) giving the number of codes for each code length from 1 to 16. This is followed by a list of the 8-bit symbol values (HUFFVAL), each of which is assigned a Huffman code. The symbol values are placed in the list in order of increasing code length. Code lengths greater than 16 bits are not allowed. In addition, the codes shall be generated such that the all-1-bits code word of any length is reserved as a prefix for longer code words.

NOTE – The order of the symbol values within HUFFVAL is determined only by code length. Within a given code length the ordering of the symbol values is arbitrary.

This annex specifies the procedure by which the Huffman tables (of Huffman code words and their corresponding 8-bit symbol values) are derived from the two lists (BITS and HUFFVAL) in the interchange format. However, the way in which these lists are generated is not specified. The lists should be generated in a manner which is consistent with the rules for Huffman coding, and it shall observe the constraints discussed in the previous paragraph. Annex K contains an example of a procedure for generating lists of Huffman code lengths and values which are in accord with these rules.

NOTE – There is **no requirement** in this Specification that any encoder or decoder shall implement the procedures in precisely the manner specified by the flow charts in this annex. It is necessary only that an encoder or decoder implement the **function** specified in this annex. The sole criterion for an encoder or decoder to be considered in compliance with this Specification is that it satisfy the requirements given in clause 6 (for encoders) or clause 7 (for decoders), as determined by the compliance tests specified in Part 2.

C.1 Marker segments for Huffman table specification

The DHT marker identifies the start of Huffman table definitions within the compressed image data. B.2.4.2 specifies the syntax for Huffman table specification.

C.2 Conversion of Huffman table specifications to tables of codes and code lengths

Conversion of Huffman table specifications to tables of codes and code lengths uses three procedures. The first procedure (Figure C.1) generates a table of Huffman code sizes. The second procedure (Figure C.2) generates the Huffman codes from the table built in Figure C.1. The third procedure (Figure C.3) generates the Huffman codes in symbol value order.

Given a list BITS (1 to 16) containing the number of codes of each size, and a list HUFFVAL containing the symbol values to be associated with those codes as described above, two tables are generated. The HUFFSIZE table contains a list of code lengths; the HUFFCODE table contains the Huffman codes corresponding to those lengths.

Note that the variable LASTK is set to the index of the last entry in the table.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

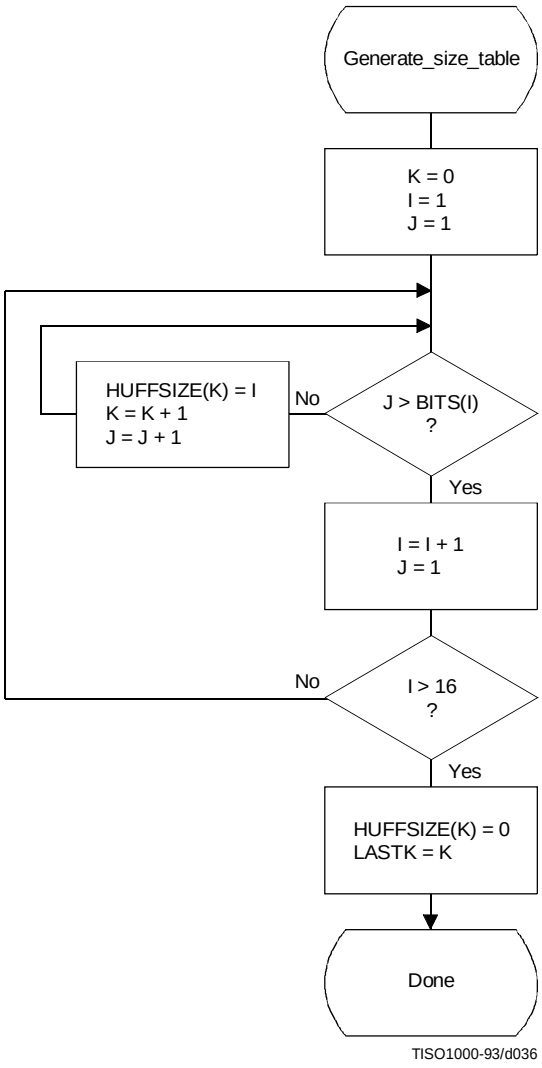


Figure C.1 – Generation of table of Huffman code sizes

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

A Huffman code table, HUFFCODE, containing a code for each size in HUFFSIZE is generated by the procedure in Figure C.2. The notation “SLL CODE 1” in Figure C.2 indicates a shift-left-logical of CODE by one bit position.

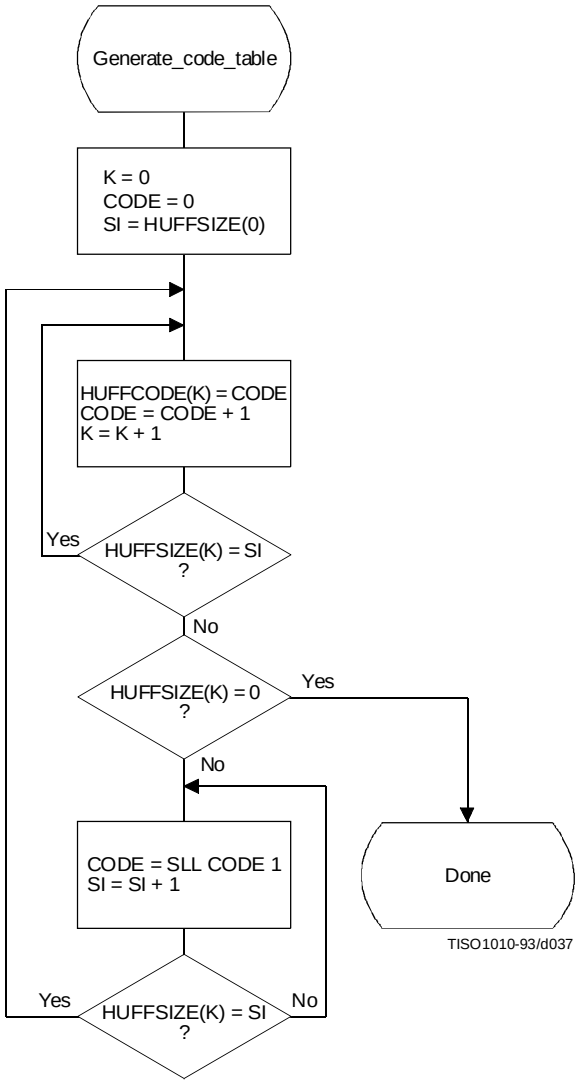


Figure C.2 – Generation of table of Huffman codes

Two tables, HUFFCODE and HUFFSIZE, have now been generated. The entries in the tables are ordered according to increasing Huffman code numeric value and length.

The encoding procedure code tables, EHUFCO and EHUFSI, are created by reordering the codes specified by HUFFCODE and HUFFSIZE according to the symbol values assigned to each code in HUFFVAL.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Figure C.3 illustrates this ordering procedure.

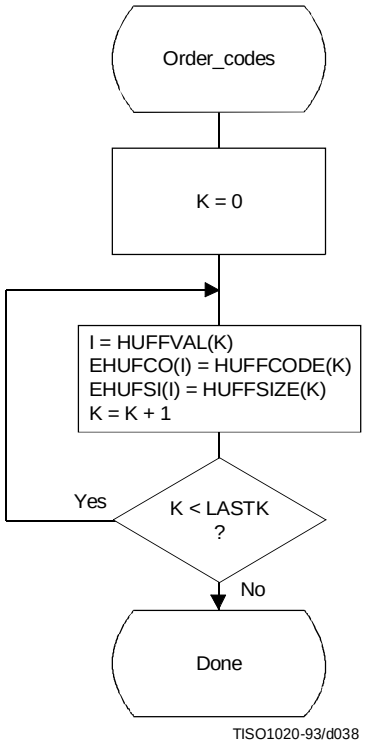


Figure C.3 – Ordering procedure for encoding procedure code tables

C.3 Bit ordering within bytes

The root of a Huffman code is placed toward the MSB (most-significant-bit) of the byte, and successive bits are placed in the direction MSB to LSB (least-significant-bit) of the byte. Remaining bits, if any, go into the next byte following the same rules.

Integers associated with Huffman codes are appended with the MSB adjacent to the LSB of the preceding Huffman code.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex D

Arithmetic coding

(This annex forms an integral part of this Recommendation | International Standard)

An adaptive binary arithmetic coding procedure may be used for entropy coding in any of the coding processes except the baseline sequential process. Coding models for adaptive binary arithmetic coding are defined in Annexes F, G, and H. In this annex the arithmetic encoding and decoding procedures used in those models are defined.

In K.4 a simple test example is given which should be helpful in determining if a given implementation is correct.

NOTE – There is **no requirement** in this Specification that any encoder or decoder shall implement the procedures in precisely the manner specified by the flow charts in this annex. It is necessary only that an encoder or decoder implement the **function** specified in this annex. The sole criterion for an encoder or decoder to be considered in compliance with this Specification is that it satisfy the requirements given in clause 6 (for encoders) or clause 7 (for decoders), as determined by the compliance tests specified in Part 2.

D.1 Arithmetic encoding procedures

Four arithmetic encoding procedures are required in a system with arithmetic coding (see Table D.1).

Table D.1 – Procedures for binary arithmetic encoding

Procedure	Purpose
Code_0(S)	Code a “0” binary decision with context-index S
Code_1(S)	Code a “1” binary decision with context-index S
Initenc	Initialize the encoder
Flush	Terminate entropy-coded segment

The “Code_0(S)”and “Code_1(S)” procedures code the 0-decision and 1-decision respectively; S is a context-index which identifies a particular conditional probability estimate used in coding the binary decision. The “Initenc” procedure initializes the arithmetic coding entropy encoder. The “Flush” procedure terminates the entropy-coded segment in preparation for the marker which follows.

D.1.1 Binary arithmetic encoding principles

The arithmetic coder encodes a series of binary symbols, zeros and ones, each symbol representing one possible result of a binary decision.

Each “binary decision” provides a choice between two alternatives. The binary decision might be between positive and negative signs, a magnitude being zero or nonzero, or a particular bit in a sequence of binary digits being zero or one.

The output bit stream (entropy-coded data segment) represents a binary fraction which increases in precision as bytes are appended by the encoding process.

D.1.1.1 Recursive interval subdivision

Recursive probability interval subdivision is the basis for the binary arithmetic encoding procedures. With each binary decision the current probability interval is subdivided into two sub-intervals, and the bit stream is modified (if necessary) so that it points to the base (the lower bound) of the probability sub-interval assigned to the symbol which occurred.

In the partitioning of the current probability interval into two sub-intervals, the sub-interval for the less probable symbol (LPS) and the sub-interval for the more probable symbol (MPS) are ordered such that usually the MPS sub-interval is closer to zero. Therefore, when the LPS is coded, the MPS sub-interval size is added to the bit stream. This coding convention requires that symbols be recognized as either MPS or LPS rather than 0 or 1. Consequently, the size of the LPS sub-interval and the sense of the MPS for each decision must be known in order to encode that decision.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The subdivision of the current probability interval would ideally require a multiplication of the interval by the probability estimate for the LPS. Because this subdivision is done approximately, it is possible for the LPS sub-interval to be larger than the MPS sub-interval. When that happens a "conditional exchange" interchanges the assignment of the sub-intervals such that the MPS is given the larger sub-interval.

Since the encoding procedure involves addition of binary fractions rather than concatenation of integer code words, the more probable binary decisions can sometimes be coded at a cost of much less than one bit per decision.

D.1.1.2 Conditioning of probability estimates

An adaptive binary arithmetic coder requires a statistical model – a model for selecting conditional probability estimates to be used in the coding of each binary decision. When a given binary decision probability estimate is dependent on a particular feature or features (the context) already coded, it is "conditioned" on that feature. The conditioning of probability estimates on previously coded decisions must be identical in encoder and decoder, and therefore can use only information known to both.

Each conditional probability estimate required by the statistical model is kept in a separate storage location or "bin" identified by a unique context-index S. The arithmetic coder is adaptive, which means that the probability estimates at each context-index are developed and maintained by the arithmetic coding system on the basis of prior coding decisions for that context-index.

D.1.2 Encoding conventions and approximations

The encoding procedures use fixed precision integer arithmetic and an integer representation of fractional values in which X'8000' can be regarded as the decimal value 0.75. The probability interval, A, is kept in the integer range X'8000' ≤ A < X'10000' by doubling it whenever its integer value falls below X'8000'. This is equivalent to keeping A in the decimal range $0.75 \le A < 1.5$. This doubling procedure is called renormalization.

The code register, C, contains the trailing bits of the bit stream. C is also doubled each time A is doubled. Periodically – to keep C from overflowing – a byte of data is removed from the high order bits of the C-register and placed in the entropy-coded segment.

Carry-over into the entropy-coded segment is limited by delaying X'FF' output bytes until the carry-over is resolved. Zero bytes are stuffed after each X'FF' byte in the entropy-coded segment in order to avoid the accidental generation of markers in the entropy-coded segment.

Keeping A in the range $0.75 \le A < 1.5$ allows a simple arithmetic approximation to be used in the probability interval subdivision. Normally, if the current estimate of the LPS probability for context-index S is Qe(S), precise calculation of the sub-intervals would require:

$Qe(S) \times A$	Probability sub-interval for the LPS;
$A - (Qe(S) \times A)$	Probability sub-interval for the MPS.

Because the decimal value of A is of order unity, these can be approximated by

Qe(S)	Probability sub-interval for the LPS;
A – Qe(S)	Probability sub-interval for the MPS.

Whenever the LPS is coded, the value of A – Qe(S) is added to the code register and the probability interval is reduced to Qe(S). Whenever the MPS is coded, the code register is left unchanged and the interval is reduced to A – Qe(S). The precision range required for A is then restored, if necessary, by renormalization of both A and C.

With the procedure described above, the approximations in the probability interval subdivision process can sometimes make the LPS sub-interval larger than the MPS sub-interval. If, for example, the value of Qe(S) is 0.5 and A is at the minimum allowed value of 0.75, the approximate scaling gives one-third of the probability interval to the MPS and two-thirds to the LPS. To avoid this size inversion, conditional exchange is used. The probability interval is subdivided using the simple approximation, but the MPS and LPS sub-interval assignments are exchanged whenever the LPS sub-interval is larger than the MPS sub-interval. This MPS/LPS conditional exchange can only occur when a renormalization will be needed.

Each binary decision uses a context. A context is the set of prior coding decisions which determine the context-index, S, identifying the probability estimate used in coding the decision.

Whenever a renormalization occurs, a probability estimation procedure is invoked which determines a new probability estimate for the context currently being coded. No explicit symbol counts are needed for the estimation. The relative probabilities of renormalization after coding of LPS and MPS provide, by means of a table-based probability estimation state machine, a direct estimate of the probabilities.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

D.1.3 Encoder code register conventions

The flow charts in this annex assume the register structures for the encoder as shown in Table D.2.

Table D.2 – Encoder register connections

	MSB			LSB
C-register	0000cbbb,	bbbbbsss,	xxxxxxxx,	xxxxxxxx
A-register	00000000,	00000000,	aaaaaaaa,	aaaaaaaa

The “a” bits are the fractional bits in the A-register (the current probability interval value) and the “x” bits are the fractional bits in the code register. The “s” bits are optional spacer bits which provide useful constraints on carry-over, and the “b” bits indicate the bit positions from which the completed bytes of data are removed from the C-register. The “c” bit is a carry bit. Except at the time of initialization, bit 15 of the A-register is always set and bit 16 is always clear (the LSB is bit 0).

These register conventions illustrate one possible implementation. However, any register conventions which allow resolution of carry-over in the encoder and which produce the same entropy-coded segment may be used. The handling of carry-over and the byte stuffing following X’FF’ will be described in a later part of this annex.

D.1.4 Code_1(S) and Code_0(S) procedures

When a given binary decision is coded, one of two possibilities occurs – either a 1-decision or a 0-decision is coded. Code_1(S) and Code_0(S) are shown in Figures D.1 and D.2. The Code_1(S) and Code_0(S) procedures use probability estimates with a context-index S. The context-index S is determined by the statistical model and is, in general, a function of the previous coding decisions; each value of S identifies a particular conditional probability estimate which is used in encoding the binary decision.

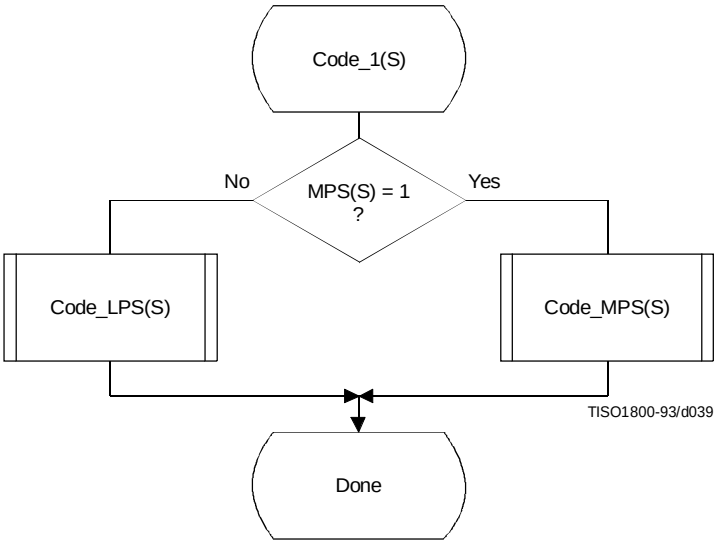


Figure D.1 – Code_1(S) procedure

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

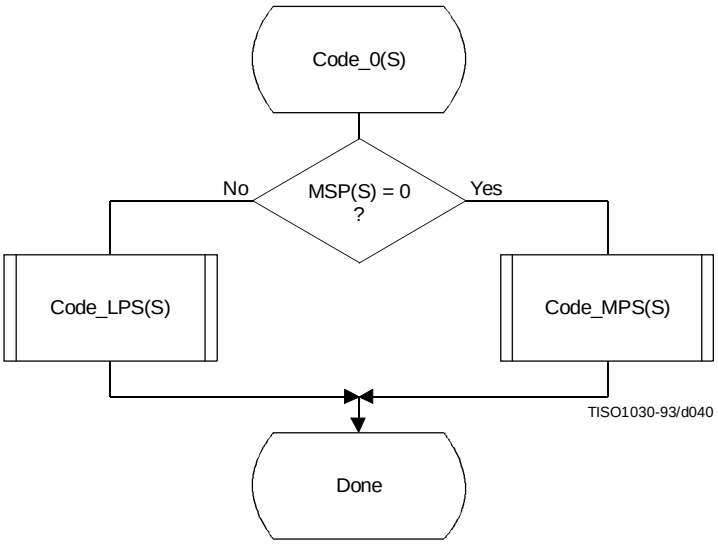


Figure D.2 – Code_0(S) procedure

The context-index S selects a storage location which contains Index(S), an index to the tables which make up the probability estimation state machine. When coding a binary decision, the symbol being coded is either the more probable symbol or the less probable symbol. Therefore, additional information is stored at each context-index identifying the sense of the more probable symbol, MPS(S).

For simplicity, the flow charts in this subclause assume that the context storage for each context-index S has an additional storage field for Qe(S) containing the value of Qe(Index(S)). If only the value of Index(S) and MPS(S) are stored, all references to Qe(S) should be replaced by Qe(Index(S)).

The Code_LPS(S) procedure normally consists of the addition of the MPS sub-interval A – Qe(S) to the bit stream and a scaling of the interval to the sub-interval, Qe(S). It is always followed by the procedures for obtaining a new LPS probability estimate (Estimate_Qe(S)_after_LPS) and renormalization (Renorm_e) (see Figure D.3).

However, in the event that the LPS sub-interval is larger than the MPS sub-interval, the conditional MPS/LPS exchange occurs and the MPS sub-interval is coded.

The Code_MPS(S) procedure normally reduces the size of the probability interval to the MPS sub-interval. However, if the LPS sub-interval is larger than the MPS sub-interval, the conditional exchange occurs and the LPS sub-interval is coded instead. Note that conditional exchange cannot occur unless the procedures for obtaining a new LPS probability estimate (Estimate_Qe(S)_after_MPS) and renormalization (Renorm_e) are required after the coding of the symbol (see Figure D.4).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

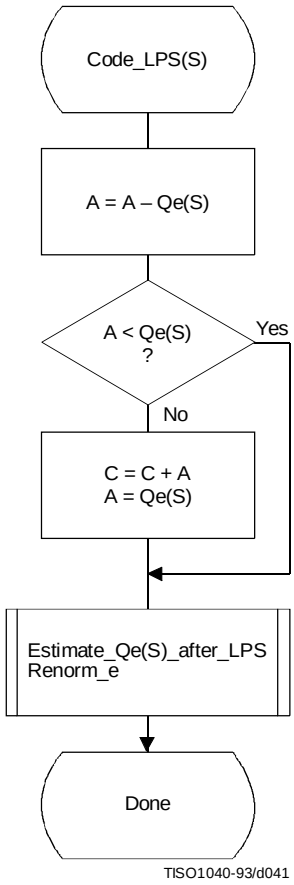


Figure D.3 – Code_LPS(S) procedure with conditional MPS/LPS exchange

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

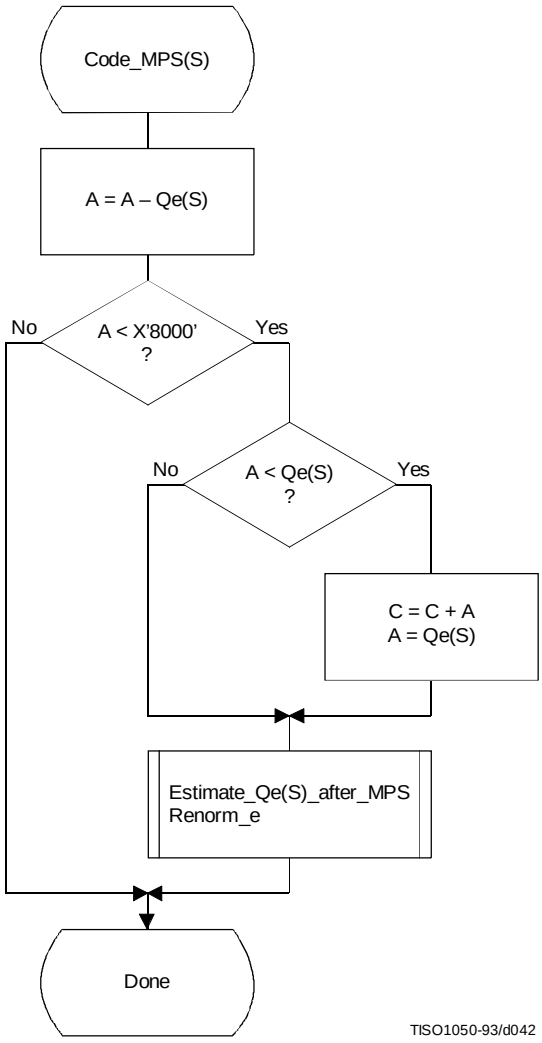


Figure D.4 – Code_MPS(S) procedure with conditional MPS/LPS exchange

D.1.5 Probability estimation in the encoder

D.1.5.1 Probability estimation state machine

The probability estimation state machine consists of a number of sequences of probability estimates. These sequences are interlinked in a manner which provides probability estimates based on approximate symbol counts derived from the arithmetic coder renormalization. Some of these sequences are used during the initial "learning" stages of probability estimation; the rest are used for "steady state" estimation.

Each entry in the probability estimation state machine is assigned an index, and each index has associated with it a Qe value and two Next_Index values. The Next_Index_MPS gives the index to the new probability estimate after an MPS renormalization; the Next_Index_LPS gives the index to the new probability estimate after an LPS renormalization. Note that both the index to the estimation state machine and the sense of the MPS are kept for each context-index S. The sense of the MPS is changed whenever the entry in the Switch_MPS is one.

The probability estimation state machine is given in Table D.3. Initialization of the arithmetic coder is always with an MPS sense of zero and a Qe index of zero in Table D.3.

The Qe values listed in Table D.3 are expressed as hexadecimal integers. To approximately convert the 15-bit integer representation of Qe to a decimal probability, divide the Qe values by $(4/3) \times (X'8000')$.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table D.3 – Qe values and probability estimation state machine

Index	Qe _Value	Next_ Index _LPS	Next_ Index _MPS	Switch _MPS	Index	Qe _Value	Next_ Index _LPS	Next_ Index _MPS	Switch _MPS
0	X'5A1D'	1	1	1	57	X'01A4'	55	58	0
1	X'2586'	14	2	0	58	X'0160'	56	59	0
2	X'1114'	16	3	0	59	X'0125'	57	60	0
3	X'080B'	18	4	0	60	X'00F6'	58	61	0
4	X'03D8'	20	5	0	61	X'00CB'	59	62	0
5	X'01DA'	23	6	0	62	X'00AB'	61	63	0
6	X'00E5'	25	7	0	63	X'008F'	61	32	0
7	X'006F'	28	8	0	64	X'5B12'	65	65	1
8	X'0036'	30	9	0	65	X'4D04'	80	66	0
9	X'001A'	33	10	0	66	X'412C'	81	67	0
10	X'000D'	35	11	0	67	X'37D8'	82	68	0
11	X'0006'	9	12	0	68	X'2FE8'	83	69	0
12	X'0003'	10	13	0	69	X'293C'	84	70	0
13	X'0001'	12	13	0	70	X'2379'	86	71	0
14	X'5A7F'	15	15	1	71	X'1EDF'	87	72	0
15	X'3F25'	36	16	0	72	X'1AA9'	87	73	0
16	X'2CF2'	38	17	0	73	X'174E'	72	74	0
17	X'207C'	39	18	0	74	X'1424'	72	75	0
18	X'17B9'	40	19	0	75	X'119C'	74	76	0
19	X'1182'	42	20	0	76	X'0F6B'	74	77	0
20	X'0CEF'	43	21	0	77	X'0D51'	75	78	0
21	X'09A1'	45	22	0	78	X'0BB6'	77	79	0
22	X'072F'	46	23	0	79	X'0A40'	77	48	0
23	X'055C'	48	24	0	80	X'5832'	80	81	1
24	X'0406'	49	25	0	81	X'4D1C'	88	82	0
25	X'0303'	51	26	0	82	X'438E'	89	83	0
26	X'0240'	52	27	0	83	X'3BDD'	90	84	0
27	X'01B1'	54	28	0	84	X'34EE'	91	85	0
28	X'0144'	56	29	0	85	X'2EAE'	92	86	0
29	X'00F5'	57	30	0	86	X'299A'	93	87	0
30	X'00B7'	59	31	0	87	X'2516'	86	71	0
31	X'008A'	60	32	0	88	X'5570'	88	89	1
32	X'0068'	62	33	0	89	X'4CA9'	95	90	0
33	X'004E'	63	34	0	90	X'44D9'	96	91	0
34	X'003B'	32	35	0	91	X'3E22'	97	92	0
35	X'002C'	33	9	0	92	X'3824'	99	93	0
36	X'5AE1'	37	37	1	93	X'32B4'	99	94	0
37	X'484C'	64	38	0	94	X'2E17'	93	86	0
38	X'3A0D'	65	39	0	95	X'56A8'	95	96	1
39	X'2EF1'	67	40	0	96	X'4F46'	101	97	0
40	X'261F'	68	41	0	97	X'47E5'	102	98	0
41	X'1F33'	69	42	0	98	X'41CF'	103	99	0
42	X'19A8'	70	43	0	99	X'3C3D'	104	100	0
43	X'1518'	72	44	0	100	X'375E'	99	93	0
44	X'1177'	73	45	0	101	X'5231'	105	102	0
45	X'0E74'	74	46	0	102	X'4C0F'	106	103	0
46	X'0BFB'	75	47	0	103	X'4639'	107	104	0
47	X'09F8'	77	48	0	104	X'415E'	103	99	0
48	X'0861'	78	49	0	105	X'5627'	105	106	1
49	X'0706'	79	50	0	106	X'50E7'	108	107	0
50	X'05CD'	48	51	0	107	X'4B85'	109	103	0
51	X'04DE'	50	52	0	108	X'5597'	110	109	0
52	X'040F'	50	53	0	109	X'504F'	111	107	0
53	X'0363'	51	54	0	110	X'5A10'	110	111	1
54	X'02D4'	52	55	0	111	X'5522'	112	109	0
55	X'025C'	53	56	0	112	X'59EB'	112	111	1
56	X'01F8'	54	57	0					

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

D.1.5.2 Renormalization driven estimation

The change in state in Table D.3 occurs only when the arithmetic coder interval register is renormalized. This must always be done after coding an LPS, and whenever the probability interval register is less than X'8000' (0.75 in decimal notation) after coding an MPS.

When the LPS renormalization is required, Next_Index_LPS gives the new index for the LPS probability estimate. When the MPS renormalization is required, Next_Index_MPS gives the new index for the LPS probability estimate. If Switch_MPS is 1 for the old index, the MPS symbol sense must be inverted after an LPS.

D.1.5.3 Estimation following renormalization after MPS

The procedure for estimating the probability on the MPS renormalization path is given in Figure D.5. Index(S) is part of the information stored for context-index S. The new value of Index(S) is obtained from Table D.3 from the column labeled Next_Index_MPS, as that is the next index after an MPS renormalization. This next index is stored as the new value of Index(S) in the context storage at context-index S, and the value of Qe at this new Index(S) becomes the new Qe(S). MPS(S) does not change.

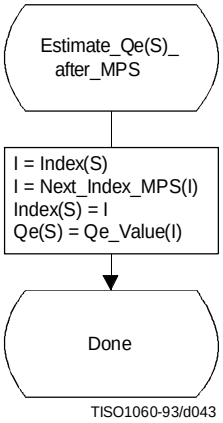


Figure D.5 – Probability estimation on MPS renormalization path

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

D.1.5.4 Estimation following renormalization after LPS

The procedure for estimating the probability on the LPS renormalization path is shown in Figure D.6. The procedure is similar to that of Figure D.5 except that when Switch_MPS(I) is 1, the sense of MPS(S) must be inverted.

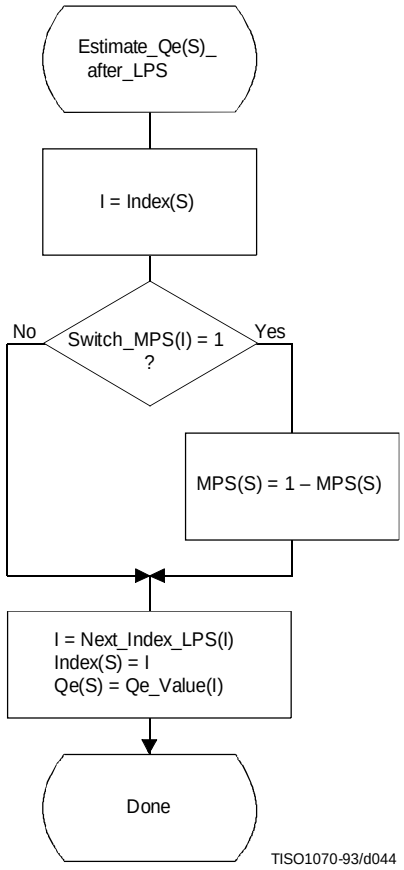


Figure D.6 – Probability estimation on LPS renormalization path

D.1.6 Renormalization in the encoder

The Renorm_e procedure for the encoder renormalization is shown in Figure D.7. Both the probability interval register A and the code register C are shifted, one bit at a time. The number of shifts is counted in the counter CT; when CT is zero, a byte of compressed data is removed from C by the procedure Byte_out and CT is reset to 8. Renormalization continues until A is no longer less than X'8000'.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

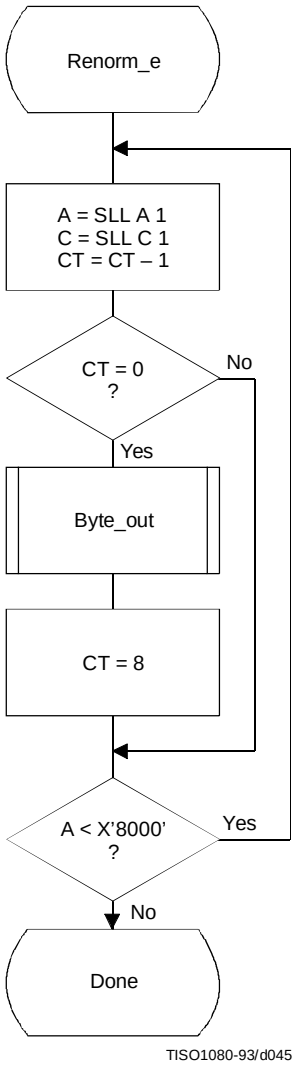


Figure D.7 – Encoder renormalization procedure

The Byte_out procedure used in Renorm_e is shown in Figure D.8. This procedure uses byte-stuffing procedures which prevent accidental generation of markers by the arithmetic encoding procedures. It also includes an example of a procedure for resolving carry-over. For simplicity of exposition, the buffer holding the entropy-coded segment is assumed to be large enough to contain the entire segment.

In Figure D.8 BP is the entropy-coded segment pointer and B is the compressed data byte pointed to by BP. T in Byte_out is a temporary variable which is used to hold the output byte and carry bit. ST is the stack counter which is used to count X'FF' output bytes until any carry-over through the X'FF' sequence has been resolved. The value of ST rarely exceeds 3. However, since the upper limit for the value of ST is bounded only by the total entropy-coded segment size, a precision of 32 bits is recommended for ST.

Since large values of ST represent a latent output of compressed data, the following procedure may be needed in high speed synchronous encoding systems for handling the burst of output data which occurs when the carry is resolved.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

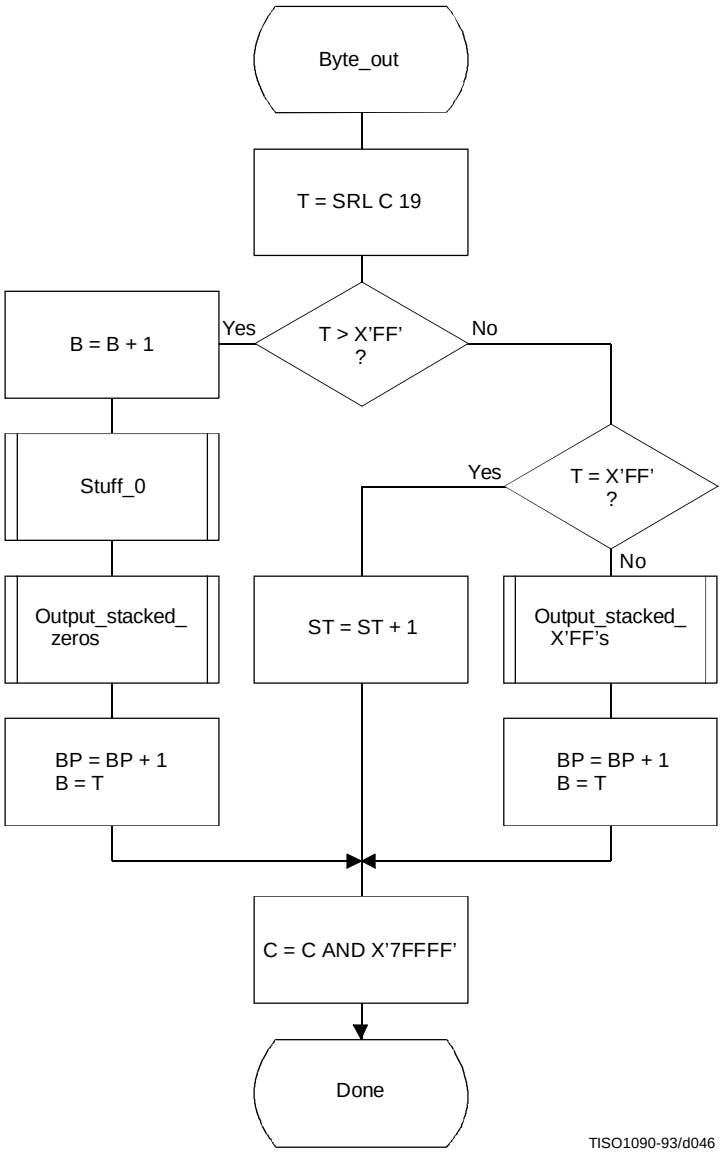


Figure D.8 – Byte_out procedure for encoder

When the stack count reaches an upper bound determined by output channel capacity, the stack is emptied and the stacked X'FF' bytes (and stuffed zero bytes) are added to the compressed data before the carry-over is resolved. If a carry-over then occurs, the carry is added to the final stuffed zero, thereby converting the final X'FF00' sequence to the X'FF01' temporary private marker. The entropy-coded segment must then be post-processed to resolve the carry-over and remove the temporary marker code. For any reasonable bound on ST this post processing is very unlikely.

Referring to Figure D.8, the shift of the code register by 19 bits aligns the output bits with the low order bits of T. The first test then determines if a carry-over has occurred. If so, the carry must be added to the previous output byte before advancing the segment pointer BP. The Stuff_0 procedure stuffs a zero byte whenever the addition of the carry to the data already in the entropy-coded segments creates a X'FF' byte. Any stacked output bytes – converted to zeros by the carry-over – are then placed in the entropy-coded segment. Note that when the output byte is later transferred from T to the entropy-coded segment (to byte B), the carry bit is ignored if it is set.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

If a carry has not occurred, the output byte is tested to see if it is X'FF'. If so, the stack count ST is incremented, as the output must be delayed until the carry-over is resolved. If not, the carry-over has been resolved, and any stacked X'FF' bytes must then be placed in the entropy-coded segment. Note that a zero byte is stuffed following each X'FF'.

The procedures used by Byte_out are defined in Figures D.9 through D.11.

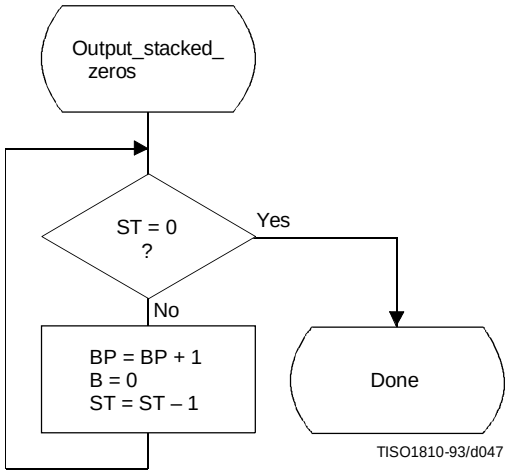


Figure D.9 – Output_stacked_zeros procedure for encoder

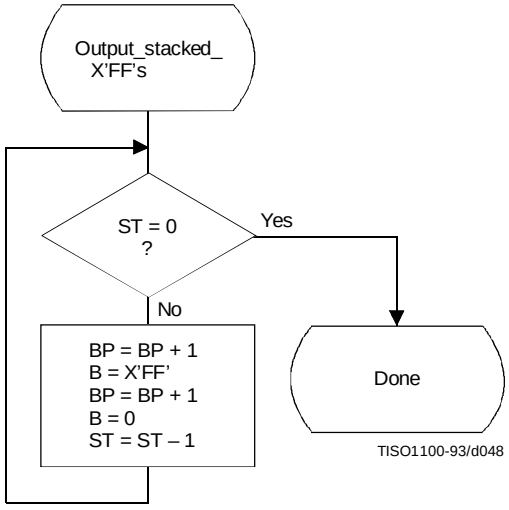


Figure D.10 – Output_stacked_X'FF's procedure for encoder

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

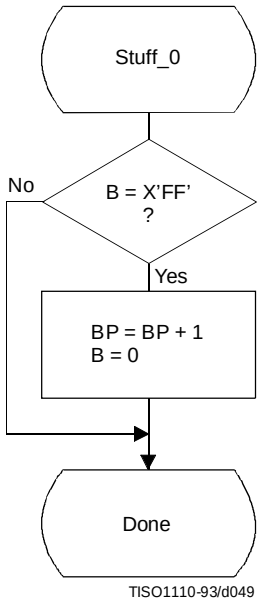


Figure D.11 – Stuff_0 procedure for encoder

D.1.7 Initialization of the encoder

The Initenc procedure is used to start the arithmetic coder. The basic steps are shown in Figure D.12.

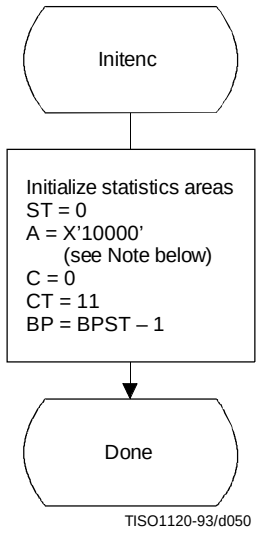


Figure D.12 – Initialization of the encoder

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The probability estimation tables are defined by Table D.3. The statistics areas are initialized to an MPS sense of 0 and a Qe index of zero as defined by Table D.3. The stack count (ST) is cleared, the code register (C) is cleared, and the interval register is set to X'10000'. The counter (CT) is set to 11, reflecting the fact that when A is initialized to X'10000' three spacer bits plus eight output bits in C must be filled before the first byte is removed. Note that BP is initialized to point to the byte before the start of the entropy-coded segment (which is at BPST). Note also that the statistics areas are initialized for all values of context-index S to MPS(S) = 0 and Index(S) = 0.

NOTE – Although the probability interval is initialized to X'10000' in both Initenc and Initdec, the precision of the probability interval register can still be limited to 16 bits. When the precision of the interval register is 16 bits, it is initialized to zero.

D.1.8 Termination of encoding

The Flush procedure is used to terminate the arithmetic encoding procedures and prepare the entropy-coded segment for the addition of the X'FF' prefix of the marker which follows the arithmetically coded data. Figure D.13 shows this flush procedure. The first step in the procedure is to set as many low order bits of the code register to zero as possible without pointing outside of the final interval. Then, the output byte is aligned by shifting it left by CT bits; Byte_out then removes it from C. C is then shifted left by 8 bits to align the second output byte and Byte_out is used a second time. The remaining low order bits in C are guaranteed to be zero, and these trailing zero bits shall not be written to the entropy-coded segment.

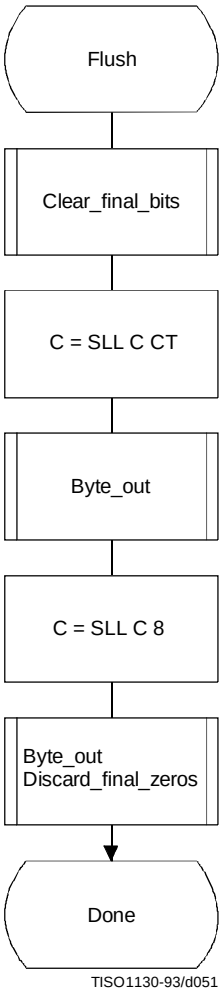


Figure D.13 – Flush procedure

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Any trailing zero bytes already written to the entropy-coded segment and not preceded by a X'FF' may, optionally, be discarded. This is done in the Discard_final_zeros procedure. Stuffed zero bytes shall not be discarded.

Entropy coded segments are always followed by a marker. For this reason, the final zero bits needed to complete decoding shall not be included in the entropy coded segment. Instead, when the decoder encounters a marker, zero bits shall be supplied to the decoding procedure until decoding is complete. This convention guarantees that when a DNL marker is used, the decoder will intercept it in time to correctly terminate the decoding procedure.

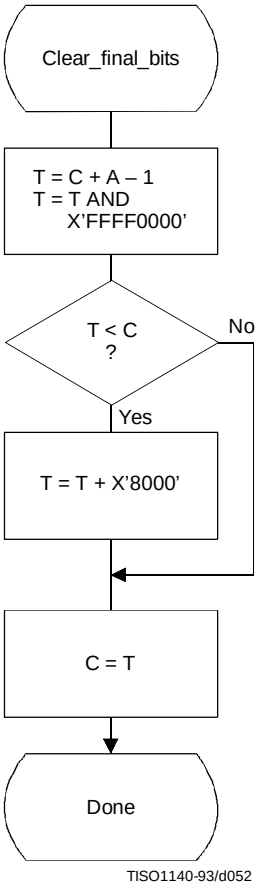


Figure D.14 – Clear_final_bits procedure in Flush

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

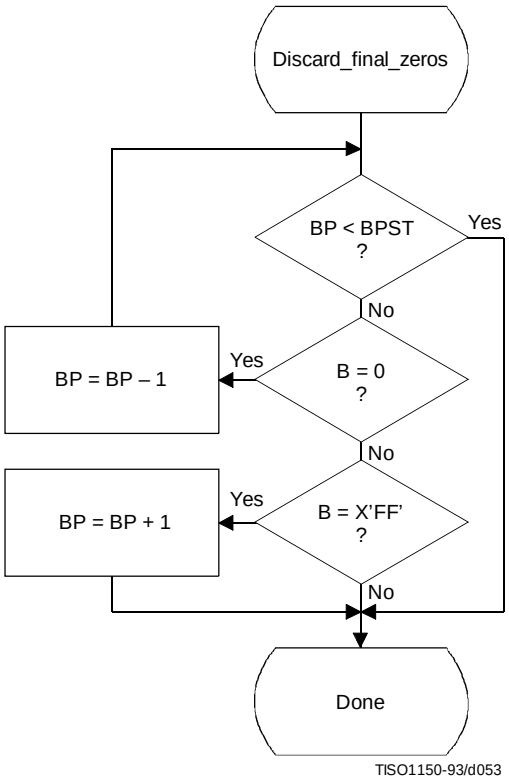


Figure D.15 – Discard_final_zeros procedure in Flush

D.2 Arithmetic decoding procedures

Two arithmetic decoding procedures are used for arithmetic decoding (see Table D.4).

The "Decode(S)" procedure decodes the binary decision for a given context-index S and returns a value of either 0 or 1. It is the inverse of the "Code_0(S)" and "Code_1(S)" procedures described in D.1. "Initdec" initializes the arithmetic coding entropy decoder.

Table D.4 – Procedures for binary arithmetic decoding

Procedure	Purpose
Decode(S)	Decode a binary decision with context-index S
Initdec	Initialize the decoder

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

D.2.1 Binary arithmetic decoding principles

The probability interval subdivision and sub-interval ordering defined for the arithmetic encoding procedures also apply to the arithmetic decoding procedures.

Since the bit stream always points within the current probability interval, the decoding process is a matter of determining, for each decision, which sub-interval is pointed to by the bit stream. This is done recursively, using the same probability interval sub-division process as in the encoder. Each time a decision is decoded, the decoder subtracts from the bit stream any interval the encoder added to the bit stream. Therefore, the code register in the decoder is a pointer into the current probability interval relative to the base of the interval.

If the size of the sub-interval allocated to the LPS is larger than the sub-interval allocated to the MPS, the encoder invokes the conditional exchange procedure. When the interval sizes are inverted in the decoder, the sense of the symbol decoded must be inverted.

D.2.2 Decoding conventions and approximations

The approximations and integer arithmetic defined for the probability interval subdivision in the encoder must also be used in the decoder. However, where the encoder would have added to the code register, the decoder subtracts from the code register.

D.2.3 Decoder code register conventions

The flow charts given in this section assume the register structures for the decoder as shown in Table D.5:

Table D.5 – Decoder register conventions

	MSB	LSB
Cx register	xxxxxxxx,	xxxxxxxx
C-low	bbbbbbbb,	00000000
A-register	aaaaaaaa,	aaaaaaaa

Cx and C-low can be regarded as one 32-bit C-register, in that renormalization of C shifts a bit of new data from bit 15 of C-low to bit 0 of Cx. However, the decoding comparisons use Cx alone. New data are inserted into the “b” bits of C-low one byte at a time.

NOTE – The comparisons shown in the various procedures use arithmetic comparisons, and therefore assume precisions greater than 16 bits for the variables. Unsigned (logical) comparisons should be used in 16-bit precision implementations.

D.2.4 The decode procedure

The decoder decodes one binary decision at a time. After decoding the decision, the decoder subtracts any amount from the code register that the encoder added. The amount left in the code register is the offset from the base of the current probability interval to the sub-interval allocated to the binary decisions not yet decoded. In the first test in the decode procedure shown in Figure D.16 the code register is compared to the size of the MPS sub-interval. Unless a conditional exchange is needed, this test determines whether the MPS or LPS for context-index S is decoded. Note that the LPS for context-index S is given by 1 – MPS(S).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

When a renormalization is needed, the MPS/LPS conditional exchange may also be needed. For the LPS path, the conditional exchange procedure is shown in Figure D.17. Note that the probability estimation in the decoder is identical to the probability estimation in the encoder (Figures D.5 and D.6).

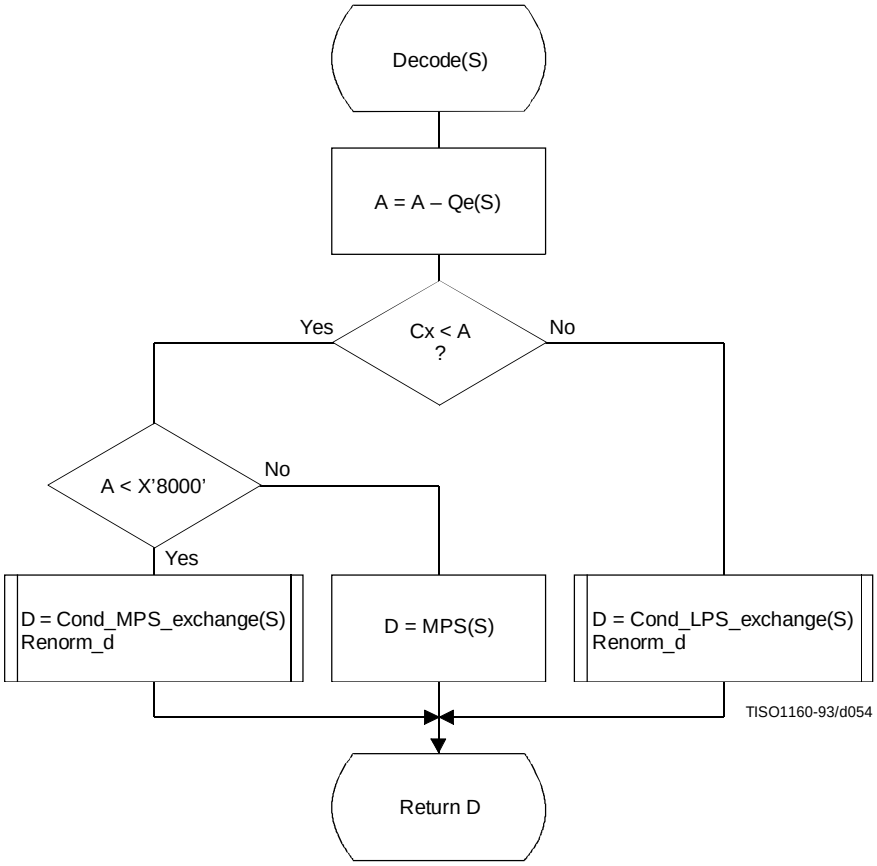


Figure D.16 – Decode(S) procedure

For the MPS path of the decoder the conditional exchange procedure is given in Figure D.18.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

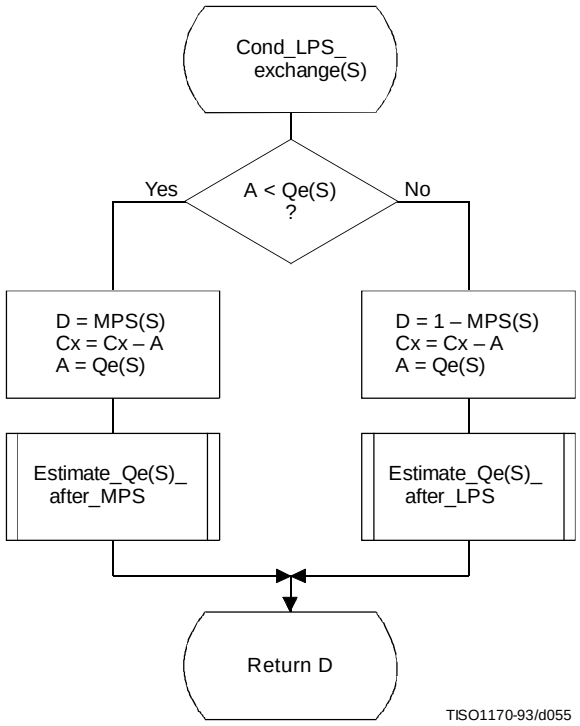


Figure D.17 – Decoder LPS path conditional exchange procedure

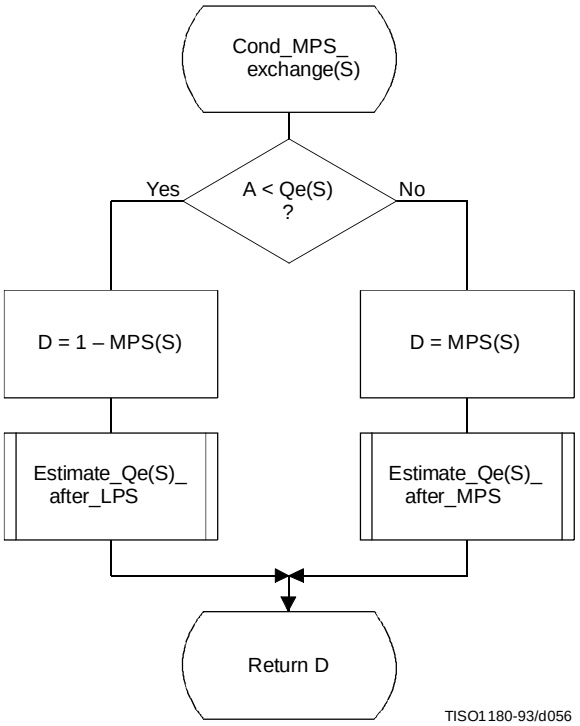


Figure D.18 – Decoder MPS path conditional exchange procedure

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

D.2.5 Probability estimation in the decoder

The procedures defined for obtaining a new LPS probability estimate in the encoder are also used in the decoder.

D.2.6 Renormalization in the decoder

The Renorm_d procedure for the decoder renormalization is shown in Figure D.19. CT is a counter which keeps track of the number of compressed bits in the C-low section of the C-register. When CT is zero, a new byte is inserted into C-low by the procedure Byte_in and CT is reset to 8.

Both the probability interval register A and the code register C are shifted, one bit at a time, until A is no longer less than X'8000'.

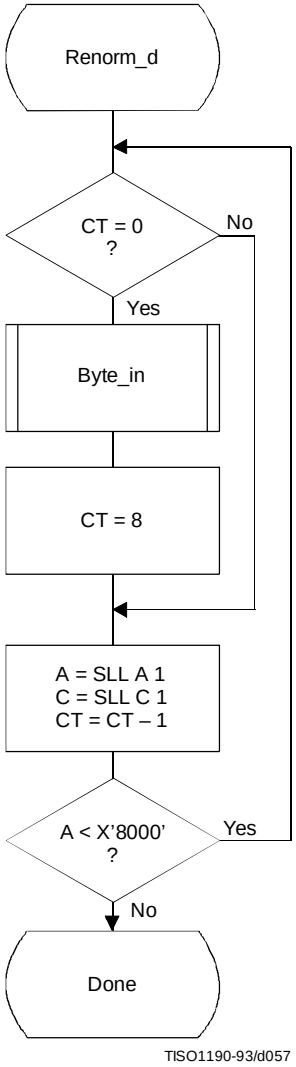


Figure D.19 – Decoder renormalization procedure

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The Byte_in procedure used in Renorm_d is shown in Figure D.20. This procedure fetches one byte of data, compensating for the stuffed zero byte which follows any X'FF' byte. It also detects the marker which must follow the entropy-coded segment. The C-register in this procedure is the concatenation of the Cx and C-low registers. For simplicity of exposition, the buffer holding the entropy-coded segment is assumed to be large enough to contain the entire segment.

B is the byte pointed to by the entropy-coded segment pointer BP. BP is first incremented. If the new value of B is not a X'FF', it is inserted into the high order 8 bits of C-low.

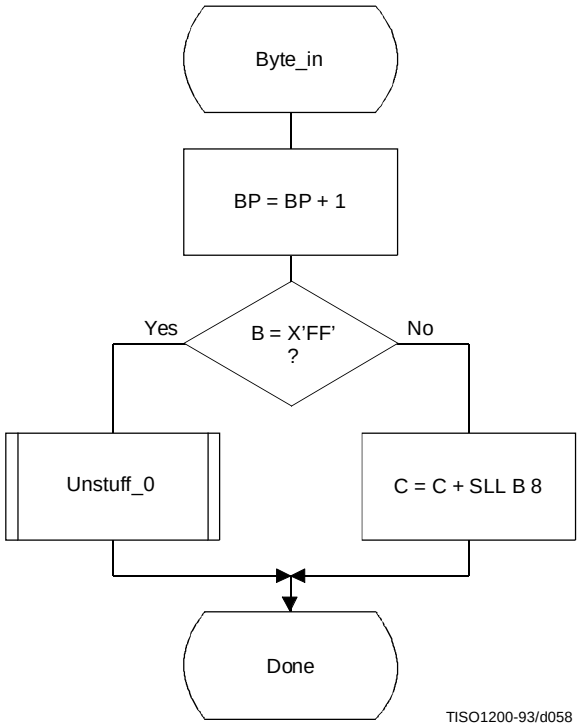


Figure D.20 – Byte_in procedure for decoder

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The Unstuff_0 procedure is shown in Figure D.21. If the new value of B is X'FF', BP is incremented to point to the next byte and this next B is tested to see if it is zero. If so, B contains a stuffed byte which must be skipped. The zero B is ignored, and the X'FF' B value which preceded it is inserted in the C-register.

If the value of B after a X'FF' byte is not zero, then a marker has been detected. The marker is interpreted as required and the entropy-coded segment pointer is adjusted ("Adjust BP" in Figure D.21) so that 0-bytes will be fed to the decoder until decoding is complete. One way of accomplishing this is to point BP to the byte preceding the marker which follows the entropy-coded segment.

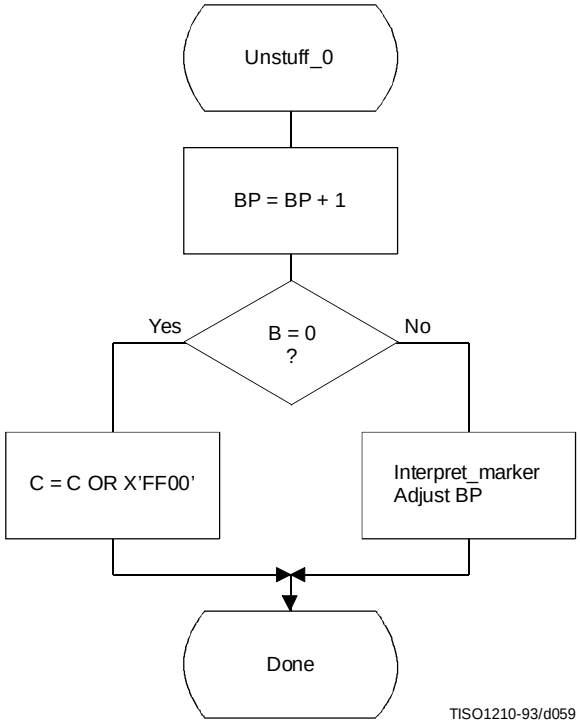


Figure D.21 – Unstuff_0 procedure for decoder

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

D.2.7 Initialization of the decoder

The Initdec procedure is used to start the arithmetic decoder. The basic steps are shown in Figure D.22.

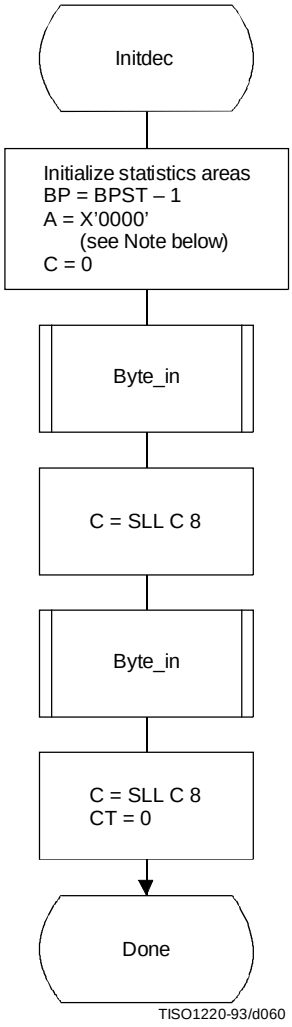


Figure D.22 – Initialization of the decoder

The estimation tables are defined by Table D.3. The statistics areas are initialized to an MPS sense of 0 and a Qe index of zero as defined by Table D.3. BP, the pointer to the entropy-coded segment, is then initialized to point to the byte before the start of the entropy-coded segment at BPST, and the interval register is set to the same starting value as in the encoder. The first byte of compressed data is fetched and shifted into Cx. The second byte is then fetched and shifted into Cx. The count is set to zero, so that a new byte of data will be fetched by Renorm_d.

NOTE – Although the probability interval is initialized to X'10000' in both Initenc and Initdec, the precision of the probability interval register can still be limited to 16 bits. When the precision of the interval register is 16 bits, it is initialized to zero.

D.3 Bit ordering within bytes

The arithmetically encoded entropy-coded segment is an integer of variable length. Therefore, the ordering of bytes and the bit ordering within bytes is the same as for parameters (see B.1.1.1).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex E

Encoder and decoder control procedures

(This annex forms an integral part of this Recommendation | International Standard)

This annex describes the encoder and decoder control procedures for the sequential, progressive, and lossless modes of operation.

The encoding and decoding control procedures for the hierarchical processes are specified in Annex J.

NOTES

1 There is **no requirement** in this Specification that any encoder or decoder shall implement the procedures in precisely the manner specified by the flow charts in this annex. It is necessary only that an encoder or decoder implement the **function** specified in this annex. The sole criterion for an encoder or decoder to be considered in compliance with this Specification is that it satisfy the requirements given in clause 6 (for encoders) or clause 7 (for decoders), as determined by the compliance tests specified in Part 2.

2 Implementation-specific setup steps are not indicated in this annex and may be necessary.

E.1 Encoder control procedures

E.1.1 Control procedure for encoding an image

The encoder control procedure for encoding an image is shown in Figure E.1.

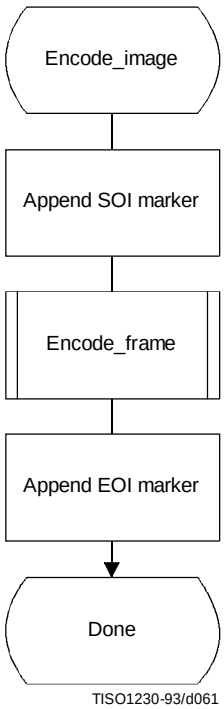


Figure E.1 – Control procedure for encoding an image

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

E.1.2 Control procedure for encoding a frame

In all cases where markers are appended to the compressed data, optional X'FF' fill bytes may precede the marker.

The control procedure for encoding a frame is oriented around the scans in the frame. The frame header is first appended, and then the scans are coded. Table specifications and other marker segments may precede the SOF_n marker, as indicated by [tables/miscellaneous] in Figure E.2.

Figure E.2 shows the encoding process frame control procedure.

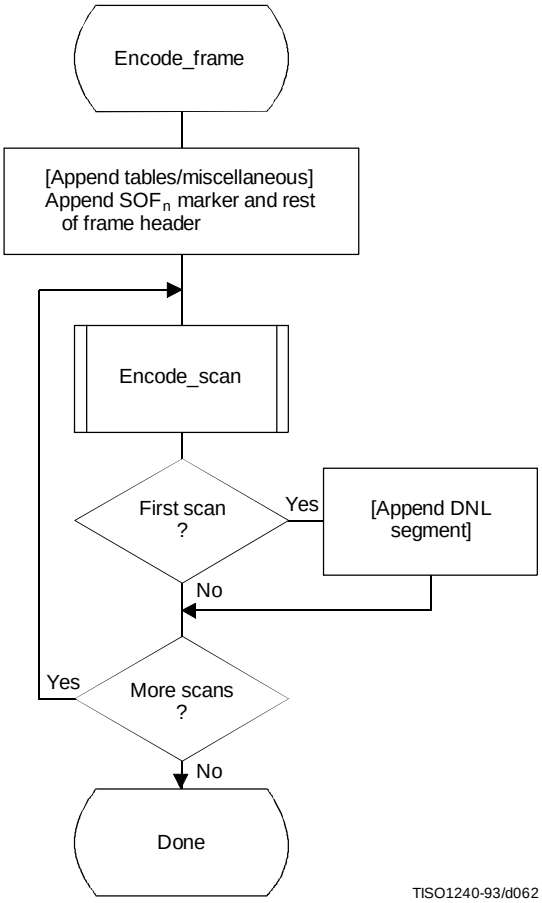


Figure E.2 – Control procedure for encoding a frame

E.1.3 Control procedure for encoding a scan

A scan consists of a single pass through the data of each component in the scan. Table specifications and other marker segments may precede the SOS marker. If more than one component is coded in the scan, the data are interleaved. If restart is enabled, the data are segmented into restart intervals. If restart is enabled, a RST_m marker is placed in the coded data between restart intervals. If restart is disabled, the control procedure is the same, except that the entire scan contains a single restart interval. The compressed image data generated by a scan is always followed by a marker, either the EOI marker or the marker of the next marker segment.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Figure E.3 shows the encoding process scan control procedure. The loop is terminated when the encoding process has coded the number of restart intervals which make up the scan. "m" is the restart interval modulo counter needed for the RST_m marker. The modulo arithmetic for this counter is shown after the "Append RST_m marker" procedure.

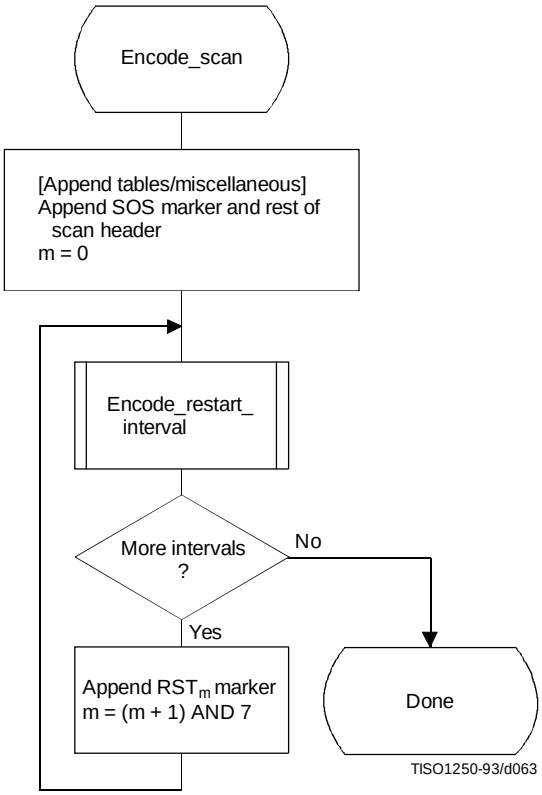


Figure E.3 – Control procedure for encoding a scan

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

E.1.4 Control procedure for encoding a restart interval

Figure E.4 shows the encoding process control procedure for a restart interval. The loop is terminated either when the encoding process has coded the number of minimum coded units (MCU) in the restart interval or when it has completed the image scan.

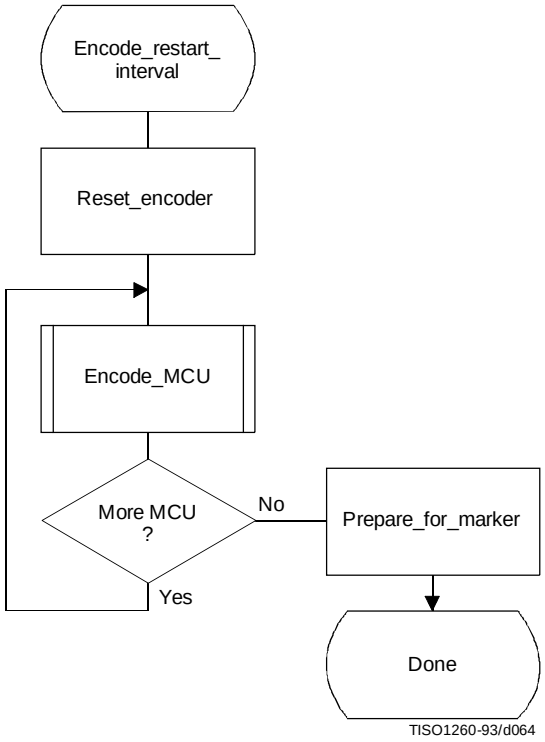


Figure E.4 – Control procedure for encoding a restart interval

The "Reset_encoder" procedure consists at least of the following:

a) if arithmetic coding is used, initialize the arithmetic encoder using the "Initenc" procedure described in D.1.7;

b) for DCT-based processes, set the DC prediction (PRED) to zero for all components in the scan (see F.1.1.5.1);

c) for lossless processes, reset the prediction to a default value for all components in the scan (see H.1.1);

d) do all other implementation-dependent setups that may be necessary.

The procedure "Prepare_for_marker" terminates the entropy-coded segment by:

a) padding a Huffman entropy-coded segment with 1-bits to complete the final byte (and if needed stuffing a zero byte) (see F.1.2.3); or

b) invoking the procedure "Flush" (see D.1.8) to terminate an arithmetic entropy-coded segment.

NOTE – The number of minimum coded units (MCU) in the final restart interval must be adjusted to match the number of MCU in the scan. The number of MCU is calculated from the frame and scan parameters. (See Annex B.)

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

E.1.5 Control procedure for encoding a minimum coded unit (MCU)

The minimum coded unit is defined in A.2. Within a given MCU the data units are coded in the order in which they occur in the MCU. The control procedure for encoding a MCU is shown in Figure E.5.

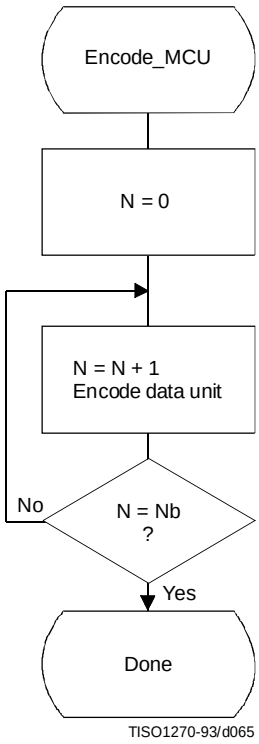


Figure E.5 – Control procedure for encoding a minimum coded unit (MCU)

In Figure E.5, Nb refers to the number of data units in the MCU. The order in which data units occur in the MCU is defined in A.2. The data unit is an 8×8 block for DCT-based processes, and a single sample for lossless processes.

The procedures for encoding a data unit are specified in Annexes F, G, and H.

E.2 Decoder control procedures

E.2.1 Control procedure for decoding compressed image data

Figure E.6 shows the decoding process control for compressed image data.

Decoding control centers around identification of various markers. The first marker must be the SOI (Start Of Image) marker. The "Decoder_setup" procedure resets the restart interval (Ri = 0) and, if the decoder has arithmetic decoding capabilities, sets the conditioning tables for the arithmetic coding to their default values. (See F.1.4.4.1.4 and F.1.4.4.2.1.) The next marker is normally a SOF_n (Start Of Frame) marker; if this is not found, one of the marker segments listed in Table E.1 has been received.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

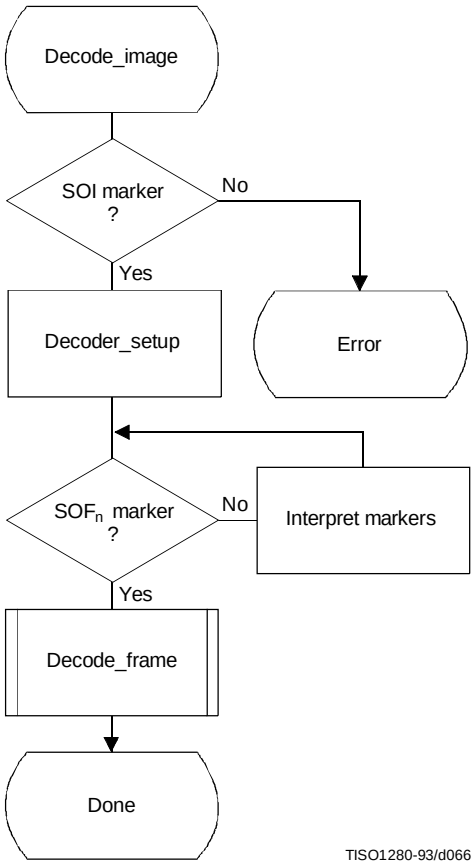


Figure E.6 – Control procedure for decoding compressed image data

Table E.1 – Markers recognized by "Interpret markers"

Marker	Purpose
DHT	Define Huffman Tables
DAC	Define Arithmetic Conditioning
DQT	Define Quantization Tables
DRI	Define Restart Interval
APP_n	Application defined marker
COM	Comment

Note that optional X'FF' fill bytes which may precede any marker shall be discarded before determining which marker is present.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The additional logic to interpret these various markers is contained in the box labeled “Interpret markers”. DHT markers shall be interpreted by processes using Huffman coding. DAC markers shall be interpreted by processes using arithmetic coding. DQT markers shall be interpreted by DCT-based decoders. DRI markers shall be interpreted by all decoders. APPn and COM markers shall be interpreted only to the extent that they do not interfere with the decoding.

By definition, the procedures in “Interpret markers” leave the system at the next marker. Note that if the expected SOI marker is missing at the start of the compressed image data, an error condition has occurred. The techniques for detecting and managing error conditions can be as elaborate or as simple as desired.

E.2.2 Control procedure for decoding a frame

Figure E.7 shows the control procedure for the decoding of a frame.

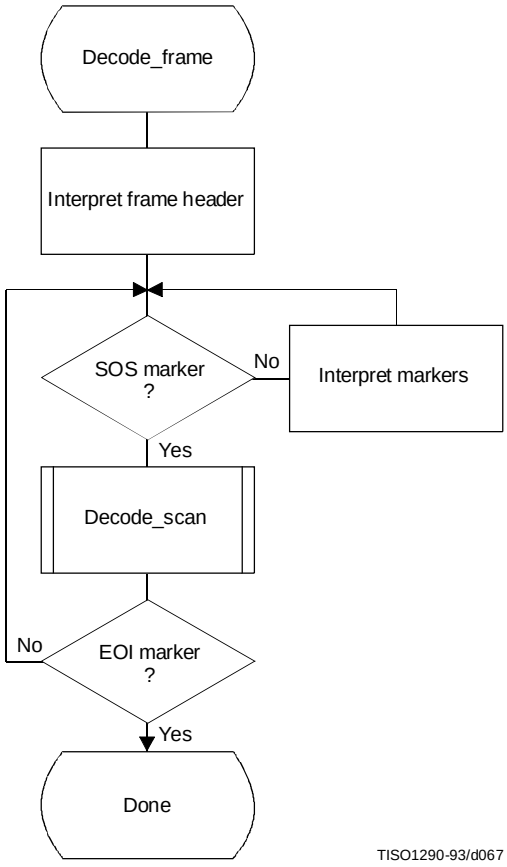


Figure E.7 – Control procedure for decoding a frame

The loop is terminated if the EOI marker is found at the end of the scan.

The markers recognized by “Interpret markers” are listed in Table E.1. Subclause E.2.1 describes the extent to which the various markers shall be interpreted.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

E.2.3 Control procedure for decoding a scan

Figure E.8 shows the decoding of a scan.

The loop is terminated when the expected number of restart intervals has been decoded.

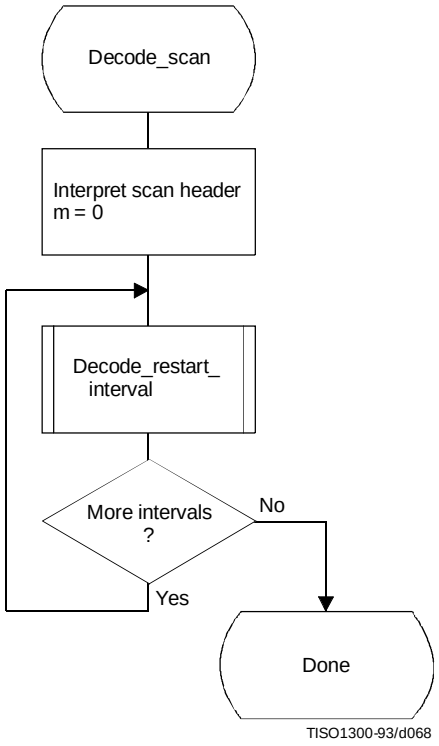


Figure E.8 – Control procedure for decoding a scan

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

E.2.4 Control procedure for decoding a restart interval

The procedure for decoding a restart interval is shown in Figure E.9. The "Reset_decoder" procedure consists at least of the following:

a) if arithmetic coding is used, initialize the arithmetic decoder using the "Initdec" procedure described in D.2.7;

b) for DCT-based processes, set the DC prediction (PRED) to zero for all components in the scan (see F.2.1.3.1);

c) for lossless process, reset the prediction to a default value for all components in the scan (see H.2.1);

d) do all other implementation-dependent setups that may be necessary.

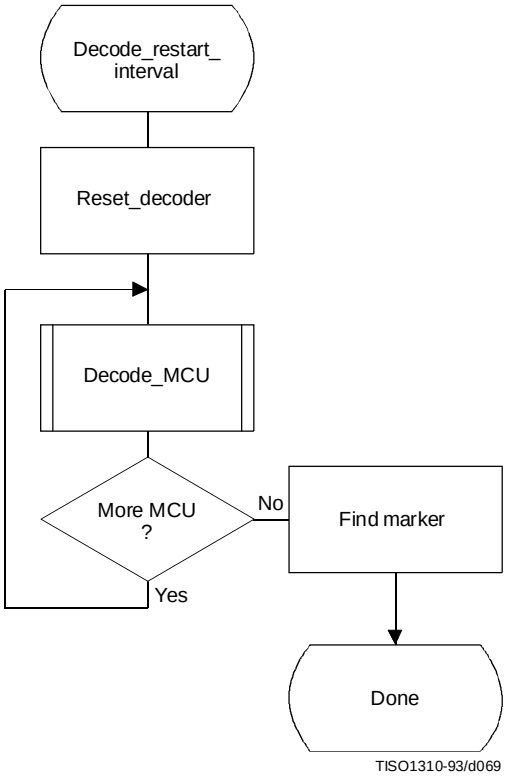


Figure E.9 – Control procedure for decoding a restart interval

At the end of the restart interval, the next marker is located. If a problem is detected in locating this marker, error handling procedures may be invoked. While such procedures are optional, the decoder shall be able to correctly recognize restart markers in the compressed data and reset the decoder when they are encountered. The decoder shall also be able to recognize the DNL marker, set the number of lines defined in the DNL segment, and end the "Decode_restart_interval" procedure.

NOTE – The final restart interval may be smaller than the size specified by the DRI marker segment, as it includes only the number of MCUs remaining in the scan.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

E.2.5 Control procedure for decoding a minimum coded unit (MCU)

The procedure for decoding a minimum coded unit (MCU) is shown in Figure E.10.

In Figure E.10 Nb is the number of data units in a MCU.

The procedures for decoding a data unit are specified in Annexes F, G, and H.

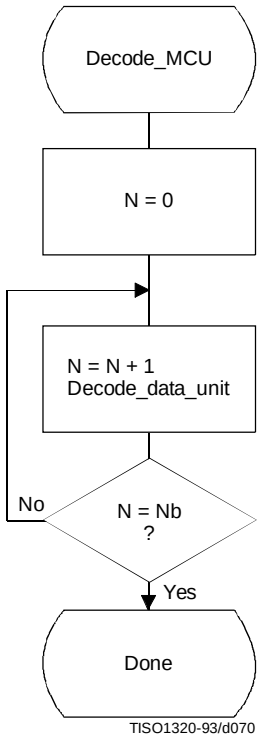


Figure E.10 – Control procedure for decoding a minimum coded unit (MCU)

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex F

Sequential DCT-based mode of operation

(This annex forms an integral part of this Recommendation | International Standard)

This annex provides a **functional specification** of the following coding processes for the sequential DCT-based mode of operation:

1) baseline sequential;
2) extended sequential, Huffman coding, 8-bit sample precision;
3) extended sequential, arithmetic coding, 8-bit sample precision;
4) extended sequential, Huffman coding, 12-bit sample precision;
5) extended sequential, arithmetic coding, 12-bit sample precision.

For each of these, the encoding process is specified in F.1, and the decoding process is specified in F.2. The functional specification is presented by means of specific flow charts for the various procedures which comprise these coding processes.

NOTE – There is **no requirement** in this Specification that any encoder or decoder which embodies one of the above-named processes shall implement the procedures in precisely the manner specified by the flow charts in this annex. It is necessary only that an encoder or decoder implement the **function** specified in this annex. The sole criterion for an encoder or decoder to be considered in compliance with this Specification is that it satisfy the requirements given in clause 6 (for encoders) or clause 7 (for decoders), as determined by the compliance tests specified in Part 2.

F.1 Sequential DCT-based encoding processes

F.1.1 Sequential DCT-based control procedures and coding models

F.1.1.1 Control procedures for sequential DCT-based encoders

The control procedures for encoding an image and its constituent parts – the frame, scan, restart interval and MCU – are given in Figures E.1 to E.5. The procedure for encoding a MCU (see Figure E.5) repetitively calls the procedure for encoding a data unit. For DCT-based encoders the data unit is an 8×8 block of samples.

F.1.1.2 Procedure for encoding an 8×8 block data unit

For the sequential DCT-based processes encoding an 8×8 block data unit consists of the following procedures:

a) level shift, calculate forward 8×8 DCT and quantize the resulting coefficients using table destination specified in frame header;

b) encode DC coefficient for 8×8 block using DC table destination specified in scan header;

c) encode AC coefficients for 8×8 block using AC table destination specified in scan header.

F.1.1.3 Level shift and forward DCT (FDCT)

The mathematical definition of the FDCT is given in A.3.3.

Prior to computing the FDCT the input data are level shifted to a signed two's complement representation as described in A.3.1. For 8-bit input precision the level shift is achieved by subtracting 128. For 12-bit input precision the level shift is achieved by subtracting 2048.

F.1.1.4 Quantization of the FDCT

The uniform quantization procedure described in Annex A is used to quantize the DCT coefficients. One of four quantization tables may be used by the encoder. No default quantization tables are specified in this Specification. However, some typical quantization tables are given in Annex K.

The quantized DCT coefficient values are signed, two's complement integers with 11-bit precision for 8-bit input precision and 15-bit precision for 12-bit input precision.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

F.1.1.5 Encoding models for the sequential DCT procedures

The two dimensional array of quantized DCT coefficients is rearranged in a zig-zag sequence order defined in A.3.6. The zig-zag order coefficients are denoted ZZ (0) through ZZ(63) with:

$$ZZ(0) = Sq_{00}, ZZ(1) = Sq_{01}, ZZ(2) = Sq_{10}, \bullet, \bullet, \bullet, ZZ(63) = Sq_{77}$$

Sq_{vu} are defined in Figure A.6.

Two coding procedures are used, one for the DC coefficient ZZ(0) and the other for the AC coefficients ZZ(1)..ZZ(63). The coefficients are encoded in the order in which they occur in zig-zag sequence order, starting with the DC coefficient. The coefficients are represented as two's complement integers.

F.1.1.5.1 Encoding model for DC coefficients

The DC coefficients are coded differentially, using a one-dimensional predictor, PRED, which is the quantized DC value from the most recently coded 8×8 block from the same component. The difference, DIFF, is obtained from

$$DIFF = ZZ(0) - PRED$$

At the beginning of the scan and at the beginning of each restart interval, the prediction for the DC coefficient prediction is initialized to 0. (Recall that the input data have been level shifted to two's complement representation.)

F.1.1.5.2 Encoding model for AC coefficients

Since many coefficients are zero, runs of zeros are identified and coded efficiently. In addition, if the remaining coefficients in the zig-zag sequence order are all zero, this is coded explicitly as an end-of-block (EOB).

F.1.2 Baseline Huffman encoding procedures

The baseline encoding procedure is for 8-bit sample precision. The encoder may employ up to two DC and two AC Huffman tables within one scan.

F.1.2.1 Huffman encoding of DC coefficients

F.1.2.1.1 Structure of DC code table

The DC code table consists of a set of Huffman codes (maximum length 16 bits) and appended additional bits (in most cases) which can code any possible value of DIFF, the difference between the current DC coefficient and the prediction. The Huffman codes for the difference categories are generated in such a way that no code consists entirely of 1-bits (X'FF' prefix marker code avoided).

The two's complement difference magnitudes are grouped into 12 categories, SSSS, and a Huffman code is created for each of the 12 difference magnitude categories (see Table F.1).

For each category, except SSSS = 0, an additional bits field is appended to the code word to uniquely identify which difference in that category actually occurred. The number of extra bits is given by SSSS; the extra bits are appended to the LSB of the preceding Huffman code, most significant bit first. When DIFF is positive, the SSSS low order bits of DIFF are appended. When DIFF is negative, the SSSS low order bits of (DIFF – 1) are appended. Note that the most significant bit of the appended bit sequence is 0 for negative differences and 1 for positive differences.

F.1.2.1.2 Defining Huffman tables for the DC coefficients

The syntax for specifying the Huffman tables is given in Annex B. The procedure for creating a code table from this information is described in Annex C. No more than two Huffman tables may be defined for coding of DC coefficients. Two examples of Huffman tables for coding of DC coefficients are provided in Annex K.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table F.1 – Difference magnitude categories for DC coding

SSSS	DIFF values
0	0
1	–1,1
2	–3,–2,2,3
3	–7..–4,4..7
4	–15..–8,8..15
5	–31..–16,16..31
6	–63..–32,32..63
7	–127..–64,64..127
8	–255..–128,128..255
9	–511..–256,256..511
10	–1 023..–512,512..1 023
11	–2 047..–1 024,1 024..2 047

F.1.2.1.3 Huffman encoding procedures for DC coefficients

The encoding procedure is defined in terms of a set of extended tables, XHUFCO and XHUFSI, which contain the complete set of Huffman codes and sizes for all possible difference values. For full 12-bit precision the tables are relatively large. For the baseline system, however, the precision of the differences may be small enough to make this description practical.

XHUFCO and XHUFSI are generated from the encoder tables EHUFCO and EHUFSI (see Annex C) by appending to the Huffman codes for each difference category the additional bits that completely define the difference. By definition, XHUFCO and XHUFSI have entries for each possible difference value. XHUFCO contains the concatenated bit pattern of the Huffman code and the additional bits field; XHUFSI contains the total length in bits of this concatenated bit pattern. Both are indexed by DIFF, the difference between the DC coefficient and the prediction.

The Huffman encoding procedure for the DC difference, DIFF, is:

SIZE = XHUFSI(DIFF)

CODE = XHUFCO(DIFF)

code SIZE bits of CODE

where DC is the quantized DC coefficient value and PRED is the predicted quantized DC value. The Huffman code (CODE) (including any additional bits) is obtained from XHUFCO and SIZE (length of the code including additional bits) is obtained from XHUFSI, using DIFF as the index to the two tables.

F.1.2.2 Huffman encoding of AC coefficients

F.1.2.2.1 Structure of AC code table

Each non-zero AC coefficient in ZZ is described by a composite 8-bit value, RS, of the form

RS = binary 'RRRRSSSS'

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The 4 least significant bits, 'SSSS', define a category for the amplitude of the next non-zero coefficient in ZZ, and the 4 most significant bits, 'RRRR', give the position of the coefficient in ZZ relative to the previous non-zero coefficient (i.e. the run-length of zero coefficients between non-zero coefficients). Since the run length of zero coefficients may exceed 15, the value 'RRRRSSSS' = X'F0' is defined to represent a run length of 15 zero coefficients followed by a coefficient of zero amplitude. (This can be interpreted as a run length of 16 zero coefficients.) In addition, a special value 'RRRRSSSS' = '00000000' is used to code the end-of-block (EOB), when all remaining coefficients in the block are zero.

The general structure of the code table is illustrated in Figure F.1. The entries marked "N/A" are undefined for the baseline procedure.

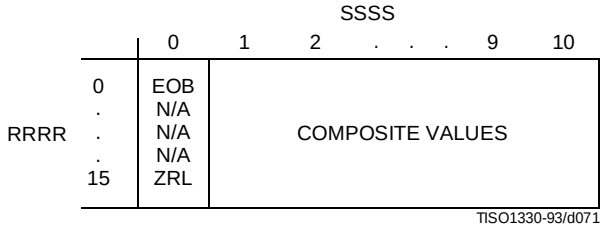


Figure F.1 – Two-dimensional value array for Huffman coding

The magnitude ranges assigned to each value of SSSS are defined in Table F.2.

Table F.2 – Categories assigned to coefficient values

SSSS	AC coefficients
1	–1,1
2	–3,–2,2,3
3	–7..–4,4..7
4	–15..–8,8..15
5	–31..–16,16..31
6	–63..–32,32..63
7	–127..–64,64..127
8	–255..–128,128..255
9	–511..–256,256..511
10	–1 023..–512,512..1 023

The composite value, RRRRSSSS, is Huffman coded and each Huffman code is followed by additional bits which specify the sign and exact amplitude of the coefficient.

The AC code table consists of one Huffman code (maximum length 16 bits, not including additional bits) for each possible composite value. The Huffman codes for the 8-bit composite values are generated in such a way that no code consists entirely of 1-bits.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The format for the additional bits is the same as in the coding of the DC coefficients. The value of SSSS gives the number of additional bits required to specify the sign and precise amplitude of the coefficient. The additional bits are either the low-order SSSS bits of ZZ(K) when ZZ(K) is positive or the low-order SSSS bits of ZZ(K) – 1 when ZZ(K) is negative. ZZ(K) is the *K*th coefficient in the zig-zag sequence of coefficients being coded.

F.1.2.2.2 Defining Huffman tables for the AC coefficients

The syntax for specifying the Huffman tables is given in Annex B. The procedure for creating a code table from this information is described in Annex C.

In the baseline system no more than two Huffman tables may be defined for coding of AC coefficients. Two examples of Huffman tables for coding of AC coefficients are provided in Annex K.

F.1.2.2.3 Huffman encoding procedures for AC coefficients

As defined in Annex C, the Huffman code table is assumed to be available as a pair of tables, EHUFCO (containing the code bits) and EHUFSI (containing the length of each code in bits), both indexed by the composite value defined above.

The procedure for encoding the AC coefficients in a block is shown in Figures F.2 and F.3. In Figure F.2, K is the index to the zig-zag scan position and R is the run length of zero coefficients.

The procedure "Append EHUFSI(X'F0') bits of EHUFCO(X'F0')" codes a run of 16 zero coefficients (ZRL code of Figure F.1). The procedure "Code EHUFSI(0) bits of EHUFCO(0)" codes the end-of-block (EOB code). If the last coefficient (K = 63) is not zero, the EOB code is bypassed.

CSIZE is a procedure which maps an AC coefficient to the SSSS value as defined in Table F.2.

F.1.2.3 Byte stuffing

In order to provide code space for marker codes which can be located in the compressed image data without decoding, byte stuffing is used.

Whenever, in the course of normal encoding, the byte value X'FF' is created in the code string, a X'00' byte is stuffed into the code string.

If a X'00' byte is detected after a X'FF' byte, the decoder must discard it. If the byte is not zero, a marker has been detected, and shall be interpreted to the extent needed to complete the decoding of the scan.

Byte alignment of markers is achieved by padding incomplete bytes with 1-bits. If padding with 1-bits creates a X'FF' value, a zero byte is stuffed before adding the marker.

F.1.3 Extended sequential DCT-based Huffman encoding process for 8-bit sample precision

This process is identical to the Baseline encoding process described in F.1.2, with the exception that the number of sets of Huffman table destinations which may be used within the same scan is increased to four. Four DC and four AC Huffman table destinations is the maximum allowed by this Specification.

F.1.4 Extended sequential DCT-based arithmetic encoding process for 8-bit sample precision

This subclause describes the use of arithmetic coding procedures in the sequential DCT-based encoding process.

NOTE – The arithmetic coding procedures in this Specification are defined for the maximum precision to encourage interchangeability.

The arithmetic coding extensions have the same DCT model as the Baseline DCT encoder. Therefore, Annex F.1.1 also applies to arithmetic coding. As with the Huffman coding technique, the binary arithmetic coding technique is lossless. It is possible to transcode between the two systems without either FDCT or IDCT computations, and without modification of the reconstructed image.

The basic principles of adaptive binary arithmetic coding are described in Annex D. Up to four DC and four AC conditioning table destinations and associated statistics areas may be used within one scan.

The arithmetic encoding procedures for encoding binary decisions, initializing the statistics area, initializing the encoder, terminating the code string, and adding restart markers are listed in Table D.1 of Annex D.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

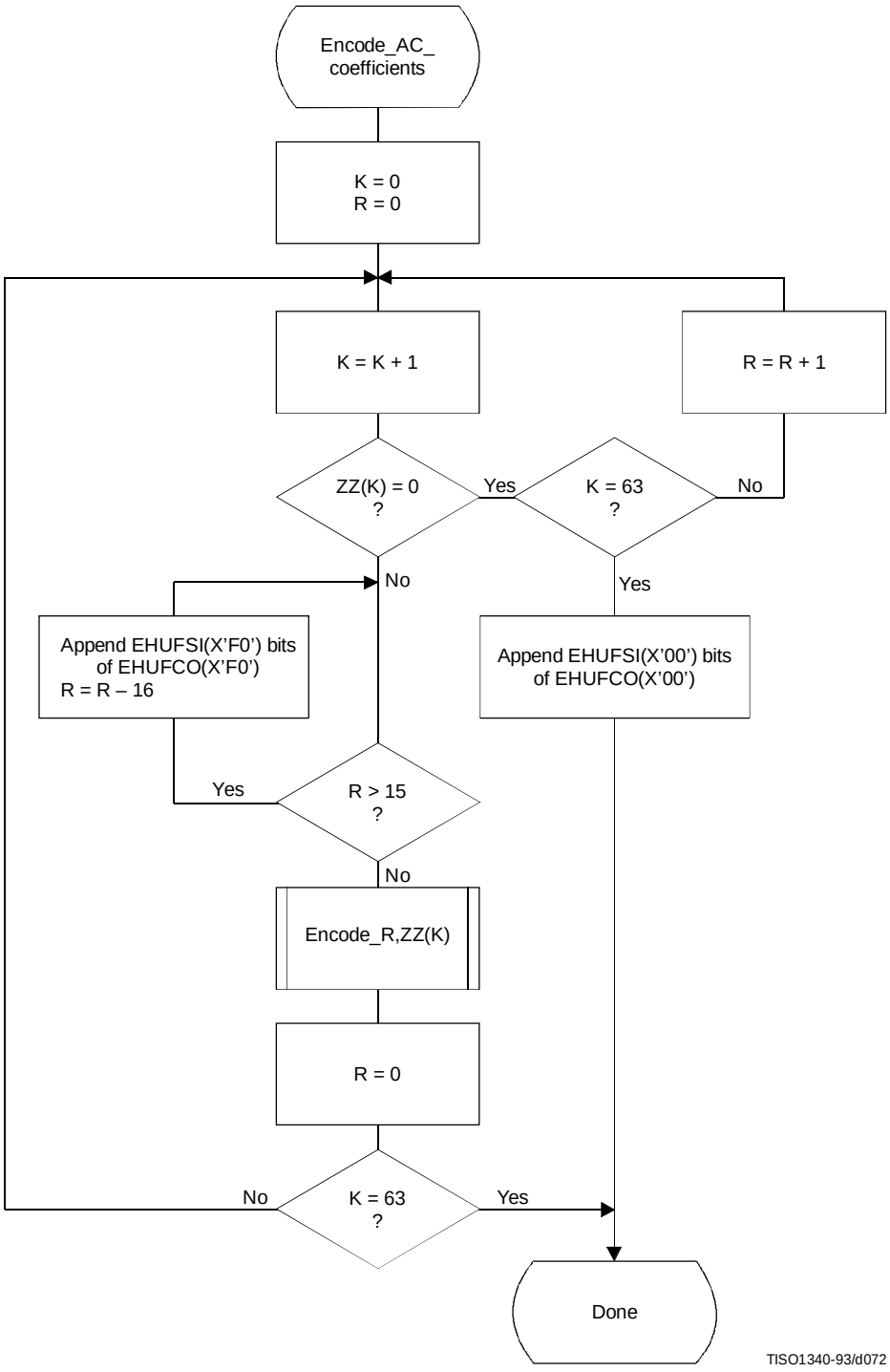


Figure F.2 – Procedure for sequential encoding of AC coefficients with Huffman coding

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

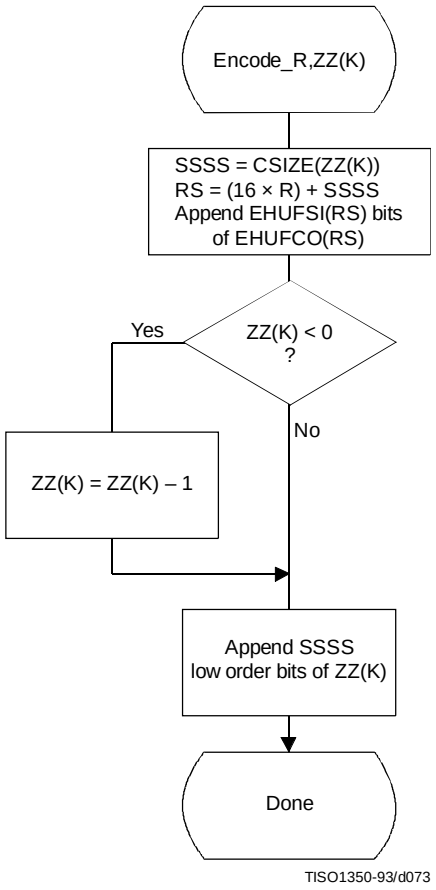


Figure F.3 – Sequential encoding of a non-zero AC coefficient

Some of the procedures in Table D.1 are used in the higher level control structure for scans and restart intervals described in Annex E. At the beginning of scans and restart intervals, the probability estimates used in the arithmetic coder are reset to the standard initial value as part of the Initenc procedure which restarts the arithmetic coder. At the end of scans and restart intervals, the Flush procedure is invoked to empty the code register before the next marker is appended.

F.1.4.1 Arithmetic encoding of DC coefficients

The basic structure of the decision sequence for encoding a DC difference value, DIFF, is shown in Figure F.4.

The context-index S0 and other context-indices used in the DC coding procedures are defined in Table F.4 (see F.1.4.4.1.3). A 0-decision is coded if the difference value is zero and a 1-decision is coded if the difference is not zero. If the difference is not zero, the sign and magnitude are coded using the procedure Encode_V(S0), which is described in F.1.4.3.1.

F.1.4.2 Arithmetic encoding of AC coefficients

The AC coefficients are coded in the order in which they occur in the zig-zag sequence ZZ(1,...,63). An end-of-block (EOB) binary decision is coded before coding the first AC coefficient in ZZ, and after each non-zero coefficient. If the EOB occurs, all remaining coefficients in ZZ are zero. Figure F.5 illustrates the decision sequence. The equivalent procedure for the Huffman coder is found in Figure F.2.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

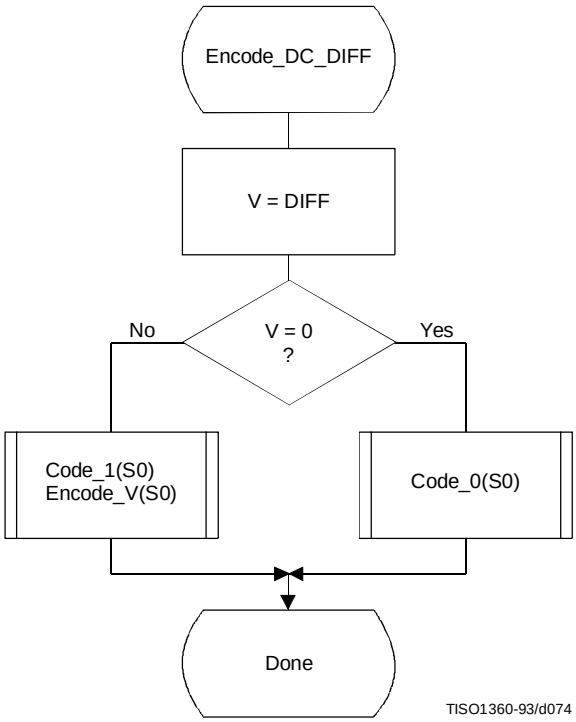


Figure F.4 – Coding model for arithmetic coding of DC difference

The context-indices SE and S0 used in the AC coding procedures are defined in Table F.5 (see F.1.4.4.2). In Figure F.5, K is the index to the zig-zag sequence position. For the sequential scan, Kmin is 1 and Se is 63. The V = 0 decision is part of a loop which codes runs of zero coefficients. Whenever the coefficient is non-zero, "Encode_V(S0)" codes the sign and magnitude of the coefficient. Each time a non-zero coefficient is coded, it is followed by an EOB decision. If the EOB occurs, a 1-decision is coded to indicate that the coding of the block is complete. If the coefficient for K = Se is not zero, the EOB decision is skipped.

F.1.4.3 Encoding the binary decision sequence for non-zero DC differences and AC coefficients

Both the DC difference and the AC coefficients are represented as signed two's complement integer values. The decomposition of these signed integer values into a binary decision tree is done in the same way for both the DC and AC coding models.

Although the binary decision trees for this section of the DC and AC coding models are the same, the statistical models for assigning statistics bins to the binary decisions in the tree are quite different.

F.1.4.3.1 Structure of the encoding decision sequence

The encoding sequence can be separated into three procedures, a procedure which encodes the sign, a second procedure which identifies the magnitude category, and a third procedure which identifies precisely which magnitude occurred within the category identified in the second procedure.

At the point where the binary decision sequence in Encode_V(S0) starts, the coefficient or difference has already been determined to be non-zero. That determination was made in the procedures in Figures F.4 and F.5.

Denoting either DC differences (DIFF) or AC coefficients as V, the non-zero signed integer value of V is encoded by the sequence shown in Figure F.6. This sequence first codes the sign of V. It then (after converting V to a magnitude and decrementing it by 1 to give Sz) codes the magnitude category of Sz (code_log2_Sz), and then codes the low order magnitude bits (code_Sz_bits) to identify the exact magnitude value.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

There are two significant differences between this sequence and the similar set of operations described in F.1.2 for Huffman coding. First, the sign is encoded before the magnitude category is identified, and second, the magnitude is decremented by 1 before the magnitude category is identified.

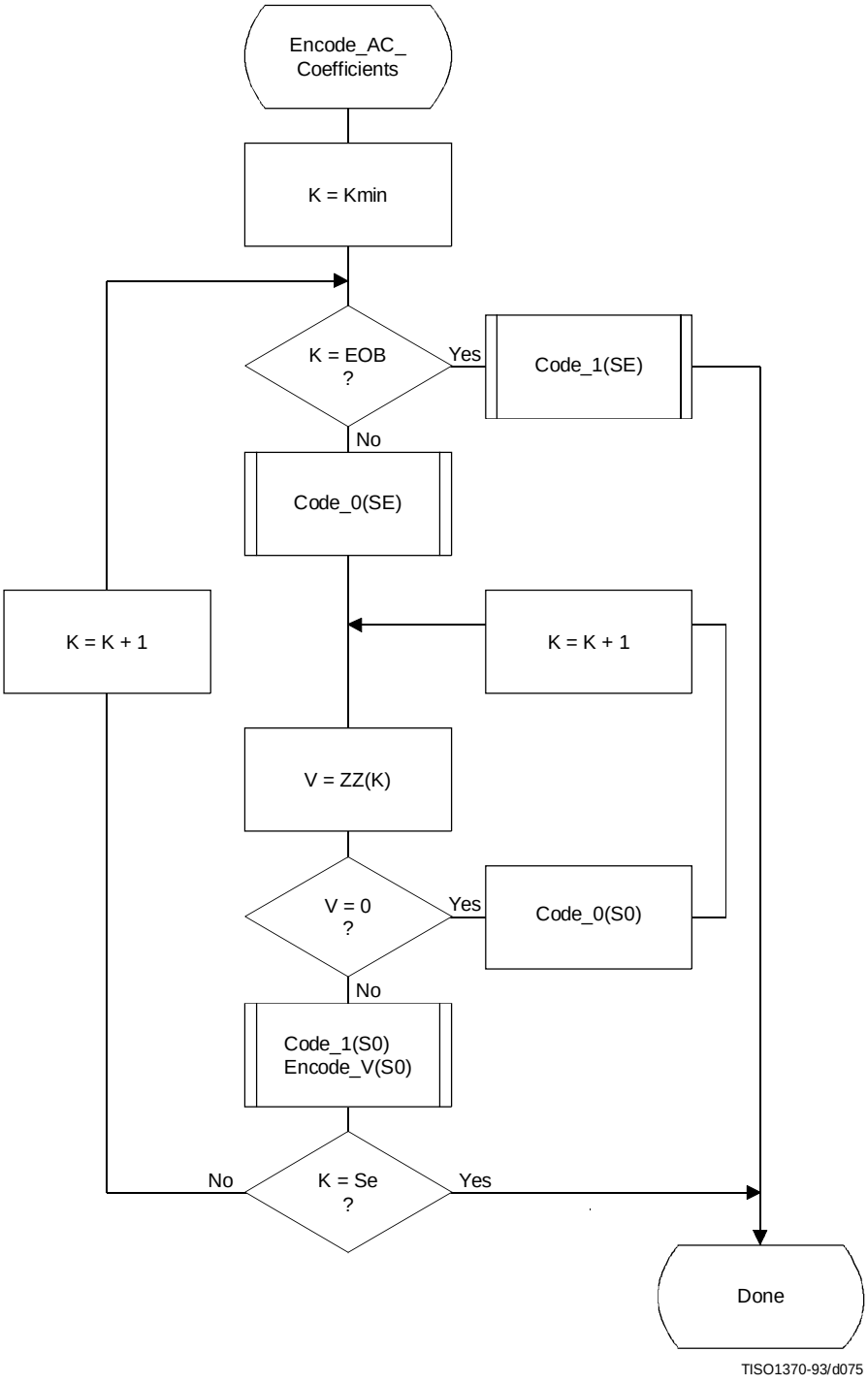


Figure F.5 – AC coding model for arithmetic coding

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

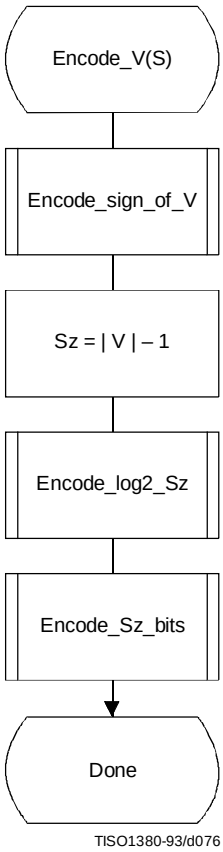


Figure F.6 – Sequence of procedures in encoding non-zero values of V

F.1.4.3.1.1 Encoding the sign

The sign is encoded by coding a 0-decision when the sign is positive and a 1-decision when the sign is negative (see Figure F.7).

The context-indices SS, SN and SP are defined for DC coding in Table F.4 and for AC coding in Table F.5. After the sign is coded, the context-index S is set to either SN or SP, establishing an initial value for Encode_log2_Sz.

F.1.4.3.1.2 Encoding the magnitude category

The magnitude category is determined by a sequence of binary decisions which compares Sz against an exponentially increasing bound (which is a power of 2) in order to determine the position of the leading 1-bit. This establishes the magnitude category in much the same way that the Huffman encoder generates a code for the value associated with the difference category. The flow chart for this procedure is shown in Figure F.8.

The starting value of the context-index S is determined in Encode_sign_of_V, and the context-index values X1 and X2 are defined for DC coding in Table F.4 and for AC coding in Table F.5. In Figure F.8, M is the exclusive upper bound for the magnitude and the abbreviations “SLL” and “SRL” refer to the shift-left-logical and shift-right-logical operations – in this case by one bit position. The SRL operation at the completion of the procedure aligns M with the most significant bit of Sz (see Table F.3).

The highest precision allowed for the DCT is 15 bits. Therefore, the highest precision required for the coding decision tree is 16 bits for the DC coefficient difference and 15 bits for the AC coefficients, including the sign bit.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

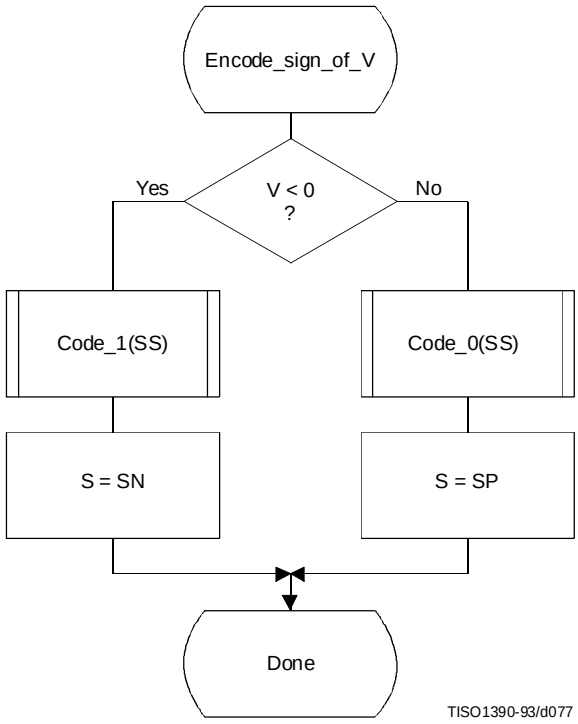


Figure F.7 – Encoding the sign of V

Table F.3 – Categories for each maximum bound

Exclusive upper bound (M)	Sz range	Number of low order magnitude bits
1	0	0
2	1	0
4	2,3	1
8	4,...,7	2
16	8,...,15	3
32	16,...,31	4
64	32,...,63	5
128	64,...,127	6
256	128,...,255	7
512	256,...,511	8
1 024	512,...,1 023	9
2 048	1 024,...,2 047	10
4 096	2 048,...,4 095	11
8 192	4 096,...,8 191	12
16 384	8 192,...,16 383	13
32 768	16 384,...,32 767	14

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

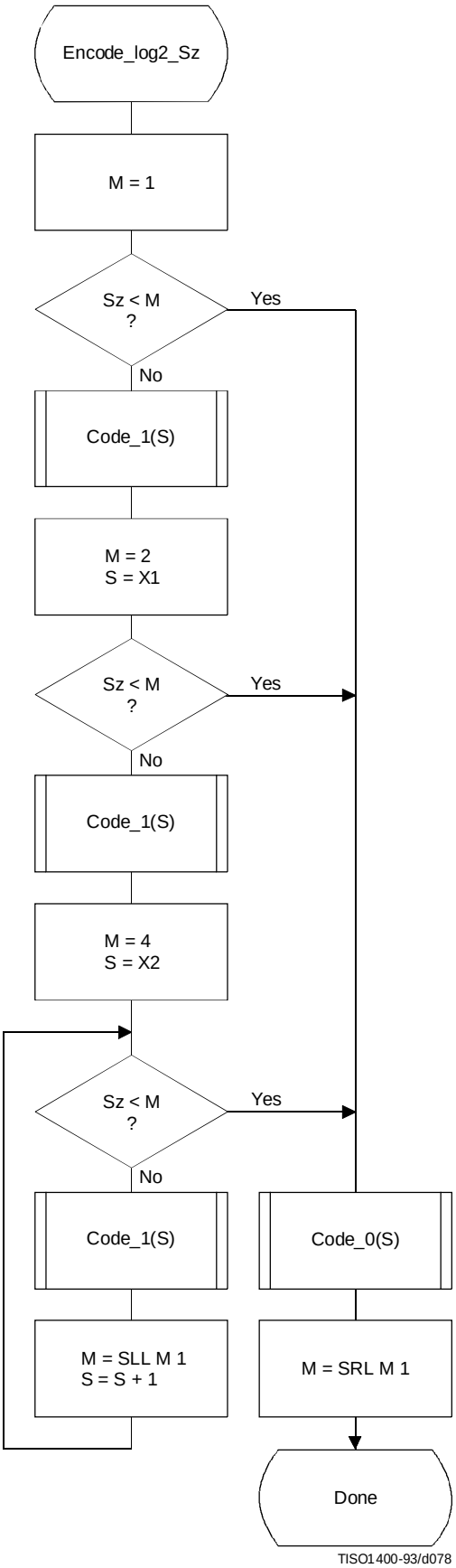


Figure F.8 – Decision sequence to establish the magnitude category

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

F.1.4.3.1.3 Encoding the exact value of the magnitude

After the magnitude category is encoded, the low order magnitude bits are encoded. These bits are encoded in order of decreasing bit significance. The procedure is shown in Figure F.9. The abbreviation "SRL" indicates the shift-right-logical operation, and M is the exclusive bound established in Figure F.8. Note that M has only one bit set – shifting M right converts it into a bit mask for the logical "AND" operation.

The starting value of the context-index S is determined in Encode_log2_Sz. The increment of S by 14 at the beginning of this procedure sets the context-index to the value required in Tables F.4 and F.5.

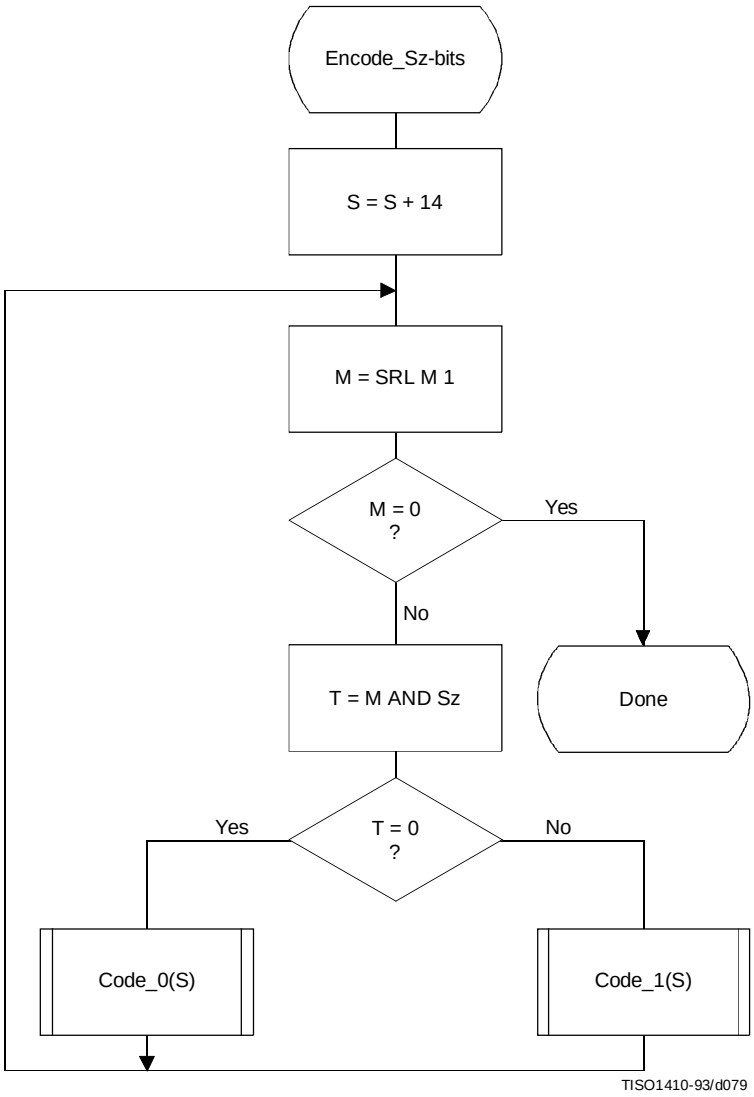


Figure F.9 – Decision sequence to code the magnitude bit pattern

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

F.1.4.4 Statistical models

An adaptive binary arithmetic coder requires a statistical model. The statistical model defines the contexts which are used to select the conditional probability estimates used in the encoding and decoding procedures.

Each decision in the binary decision trees is associated with one or more contexts. These contexts identify the sense of the MPS and the index in Table D.3 of the conditional probability estimate Qe which is used to encode and decode the binary decision.

The arithmetic coder is adaptive, which means that the probability estimates for each context are developed and maintained by the arithmetic coding system on the basis of prior coding decisions for that context.

F.1.4.4.1 Statistical model for coding DC prediction differences

The statistical model for coding the DC difference conditions some of the probability estimates for the binary decisions on previous DC coding decisions.

F.1.4.4.1.1 Statistical conditioning on sign

In coding the DC coefficients, four separate statistics bins (probability estimates) are used in coding the zero/not-zero (V = 0) decision, the sign decision and the first magnitude category decision. Two of these bins are used to code the V = 0 decision and the sign decision. The other two bins are used in coding the first magnitude decision, Sz < 1; one of these bins is used when the sign is positive, and the other is used when the sign is negative. Thus, the first magnitude decision probability estimate is conditioned on the sign of V.

F.1.4.4.1.2 Statistical conditioning on DC difference in previous block

The probability estimates for these first three decisions are also conditioned on Da, the difference value coded for the previous DCT block of the same component. The differences are classified into five groups: zero, small positive, small negative, large positive and large negative. The relationship between the default classification and the quantization scale is shown in Figure F.10.

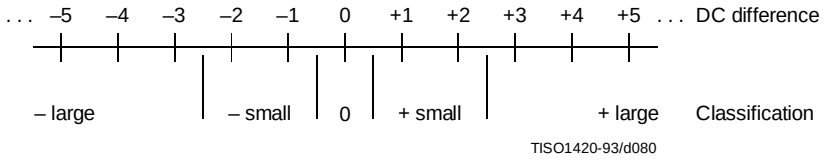


Figure F.10 – Conditioning classification of difference values

The bounds for the “small” difference category determine the classification. Defining L and U as integers in the range 0 to 15 inclusive, the lower bound (exclusive) for difference magnitudes classified as “small” is zero for L = 0, and is 2^{L-1} for L > 0.

The upper bound (inclusive) for difference magnitudes classified as “small” is 2^{U}.

L shall be less than or equal to U.

These bounds for the conditioning category provide a segmentation which is identical to that listed in Table F.3.

F.1.4.4.1.3 Assignment of statistical bins to the DC binary decision tree

As shown in Table F.4, each statistics area for DC coding consists of a set of 49 statistics bins. In the following explanation, it is assumed that the bins are contiguous. The first 20 bins consist of five sets of four bins selected by a context-index S0. The value of S0 is given by DC_Context(Da), which provides a value of 0, 4, 8, 12 or 16, depending on the difference classification of Da (see F.1.4.4.1.2). The remaining 29 bins, X1,...,X15,M2,...,M15, are used to code magnitude category decisions and magnitude bits.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table F.4 – Statistical model for DC coefficient coding

Context-index	Value	Coding decision
S0	DC_Context(Da)	V = 0
SS	S0 + 1	Sign of V
SP	S0 + 2	Sz < 1 if V > 0
SN	S0 + 3	Sz < 1 if V < 0
X1	20	Sz < 2
X2	X1 + 1	Sz < 4
X3	X1 + 2	Sz < 8
.	.	.
.	.	.
X15	X1 + 14	$Sz < 2^{15}$
M2	X2 + 14	Magnitude bits if Sz < 4
M3	X3 + 14	Magnitude bits if Sz < 8
.	.	.
.	.	.
M15	X15 + 14	Magnitude bits if $Sz < 2^{15}$

F.1.4.4.1.4 Default conditioning for DC statistical model

The bounds, L and U, for determining the conditioning category have the default values L = 0 and U = 1. Other bounds may be set using the DAC (Define Arithmetic coding Conditioning) marker segment, as described in Annex B.

F.1.4.4.1.5 Initial conditions for DC statistical model

At the start of a scan and at the beginning of each restart interval, the difference for the previous DC value is defined to be zero in determining the conditioning state.

F.1.4.4.2 Statistical model for coding the AC coefficients

As shown in Table F.5, each statistics area for AC coding consists of a contiguous set of 245 statistics bins. Three bins are used for each value of the zig-zag index K, and two sets of 28 additional bins X2,...,X15,M2,...,M15 are used for coding the magnitude category and magnitude bits.

The value of SE (and also S0, SP and SN) is determined by the zig-zag index K. Since K is in the range 1 to 63, the lowest value for SE is 0 and the largest value for SP is 188. SS is not assigned a value in AC coefficient coding, as the signs of the coefficients are coded with a fixed probability value of approximately 0.5 (Qe = X'5A1D', MPS = 0).

The value of X2 is given by AC_Context(K). This gives X2 = 189 when K ≤ Kx and X2 = 217 when K > Kx, where Kx is defined using the DAC marker segment (see B.2.4.3).

Note that a X1 statistics bin is not used in this sequence. Instead, the 63 × 1 array of statistics bins for the magnitude category is used for two decisions. Once the magnitude bound has been determined – at statistics bin Xn, for example – a single statistics bin, Mn, is used to code the magnitude bit sequence for that bound.

F.1.4.4.2.1 Default conditioning for AC coefficient coding

The default value of Kx is 5. This may be modified using the DAC marker segment, as described in Annex B.

F.1.4.4.2.2 Initial conditions for AC statistical model

At the start of a scan and at each restart, all statistics bins are re-initialized to the standard default value described in Annex D.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table F.5 – Statistical model for AC coefficient coding

Context-index	Value	Coding decision
SE	$3 \times (K - 1)$	K = EOB
S0	SE + 1	V = 0
SS	Fixed estimate	Sign of V
SN,SP	S0 + 1	Sz < 1
X1	S0 + 1	Sz < 2
X2	AC_Context(K)	Sz < 4
X3	X2 + 1	Sz < 8
.	.	.
.	.	.
X15	X2 + 13	$Sz < 2^{15}$
M2	X2 + 14	Magnitude bits if Sz < 4
M3	X3 + 14	Magnitude bits if Sz < 8
.	.	.
.	.	.
M15	X15 + 14	Magnitude bits if $Sz < 2^{15}$

F.1.5 Extended sequential DCT-based Huffman encoding process for 12-bit sample precision

This process is identical to the sequential DCT process for 8-bit precision extended to four Huffman table destinations as documented in F.1.3, with the following changes.

F.1.5.1 Structure of DC code table for 12-bit sample precision

The two's complement difference magnitudes are grouped into 16 categories, SSSS, and a Huffman code is created for each of the 16 difference magnitude categories.

The Huffman table for DC coding (see Table F.1) is extended as shown in Table F.6.

Table F.6 – Difference magnitude categories for DC coding

SSSS	Difference values
12	–4 095..–2 048,2 048..4 095
13	–8 191..–4 096,4 096..8 191
14	–16 383..–8 192,8 192..16 383
15	–32 767..–16 384,16 384..32 767

F.1.5.2 Structure of AC code table for 12-bit sample precision

The general structure of the code table is extended as illustrated in Figure F.11. The Huffman table for AC coding is extended as shown in Table F.7.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

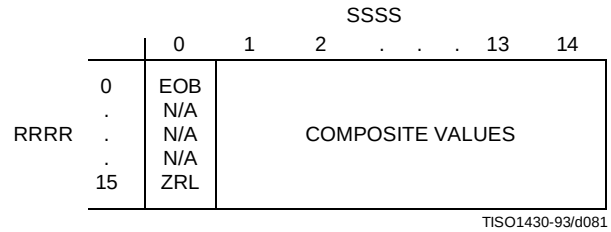


Figure F.11 – Two-dimensional value array for Huffman coding

Table F.7 – Values assigned to coefficient amplitude ranges

SSSS	AC coefficients
11	–2 047..–1 024,1 024..2 047
12	–4 095..–2 048,2 048..4 095
13	–8 191..–4 096,4 096..8 191
14	–16 383..–8 192,8 192..16 383

F.1.6 Extended sequential DCT-based arithmetic encoding process for 12-bit sample precision

The process is identical to the sequential DCT process for 8-bit precision except for changes in the precision of the FDCT computation.

The structure of the encoding procedure is identical to that specified in F.1.4 which was already defined for a 12-bit sample precision.

F.2 Sequential DCT-based decoding processes

F.2.1 Sequential DCT-based control procedures and coding models

F.2.1.1 Control procedures for sequential DCT-based decoders

The control procedures for decoding compressed image data and its constituent parts – the frame, scan, restart interval and MCU – are given in Figures E.6 to E.10. The procedure for decoding a MCU (Figure E.10) repetitively calls the procedure for decoding a data unit. For DCT-based decoders the data unit is an 8×8 block of samples.

F.2.1.2 Procedure for decoding an 8×8 block data unit

In the sequential DCT-based decoding process, decoding an 8×8 block data unit consists of the following procedures:

a) decode DC coefficient for 8×8 block using the DC table destination specified in the scan header;

b) decode AC coefficients for 8×8 block using the AC table destination specified in the scan header;

c) dequantize using table destination specified in the frame header and calculate the inverse 8×8 DCT.

F.2.1.3 Decoding models for the sequential DCT procedures

Two decoding procedures are used, one for the DC coefficient ZZ(0) and the other for the AC coefficients ZZ(1)...ZZ(63). The coefficients are decoded in the order in which they occur in the zig-zag sequence order, starting with the DC coefficient. The coefficients are represented as two's complement integers.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

F.2.1.3.1 Decoding model for DC coefficients

The decoded difference, DIFF, is added to PRED, the DC value from the most recently decoded 8×8 block from the same component. Thus ZZ(0) = PRED + DIFF.

At the beginning of the scan and at the beginning of each restart interval, the prediction for the DC coefficient is initialized to zero.

F.2.1.3.2 Decoding model for AC coefficients

The AC coefficients are decoded in the order in which they occur in ZZ. When the EOB is decoded, all remaining coefficients in ZZ are initialized to zero.

F.2.1.4 Dequantization of the quantized DCT coefficients

The dequantization of the quantized DCT coefficients as described in Annex A, is accomplished by multiplying each quantized coefficient value by the quantization table value for that coefficient. The decoder shall be able to use up to four quantization table destinations.

F.2.1.5 Inverse DCT (IDCT)

The mathematical definition of the IDCT is given in A.3.3.

After computation of the IDCT, the signed output samples are level-shifted, as described in Annex A, converting the output to an unsigned representation. For 8-bit precision the level shift is performed by adding 128. For 12-bit precision the level shift is performed by adding 2 048. If necessary, the output samples shall be clamped to stay within the range appropriate for the precision (0 to 255 for 8-bit precision and 0 to 4 095 for 12-bit precision).

F.2.2 Baseline Huffman Decoding procedures

The baseline decoding procedure is for 8-bit sample precision. The decoder shall be capable of using up to two DC and two AC Huffman tables within one scan.

F.2.2.1 Huffman decoding of DC coefficients

The decoding procedure for the DC difference, DIFF, is:

T = DECODE

DIFF = RECEIVE(T)

DIFF = EXTEND(DIFF,T)

where DECODE is a procedure which returns the 8-bit value associated with the next Huffman code in the compressed image data (see F.2.2.3) and RECEIVE(T) is a procedure which places the next T bits of the serial bit string into the low order bits of DIFF, MSB first. If T is zero, DIFF is set to zero. EXTEND is a procedure which converts the partially decoded DIFF value of precision T to the full precision difference. EXTEND is shown in Figure F.12.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

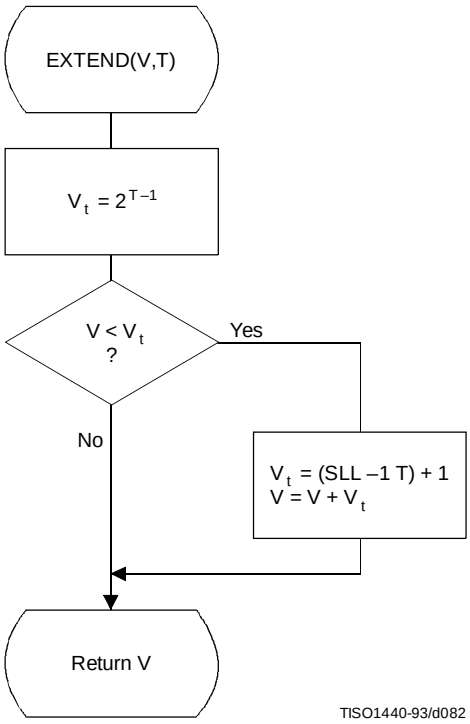


Figure F.12 – Extending the sign bit of a decoded value in V

F.2.2.2 Decoding procedure for AC coefficients

The decoding procedure for AC coefficients is shown in Figures F.13 and F.14.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

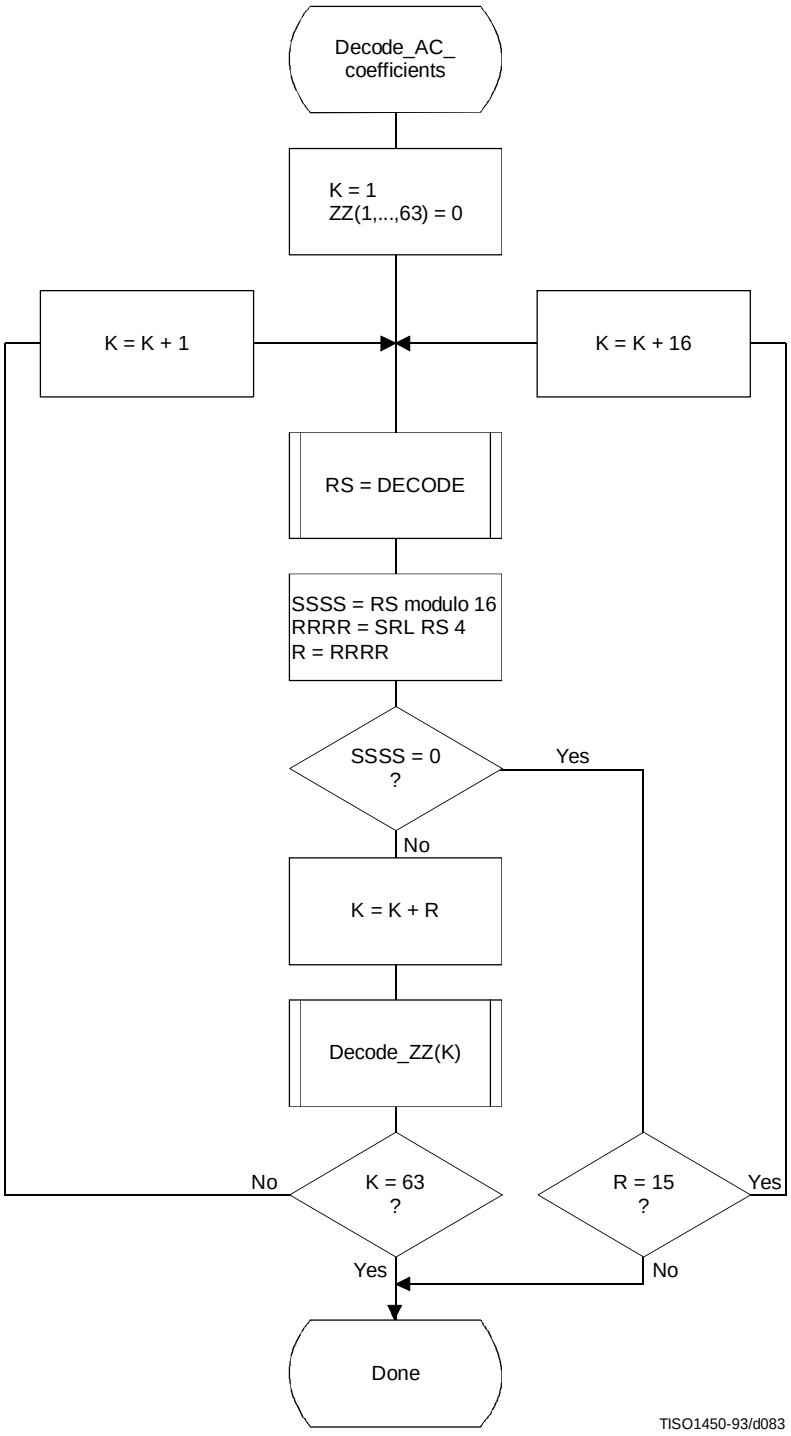


Figure F.13 – Huffman decoding procedure for AC coefficients

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

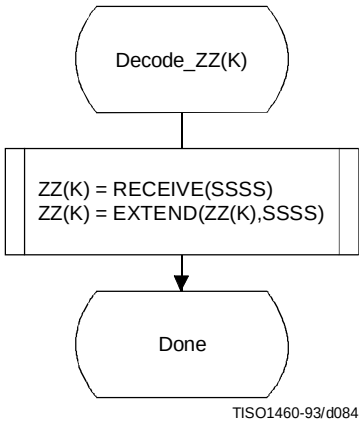


Figure F.14 – Decoding a non-zero AC coefficient

The decoding of the amplitude and sign of the non-zero coefficient is done in the procedure "Decode_ZZ(K)", shown in Figure F.14.

DECODE is a procedure which returns the value, RS, associated with the next Huffman code in the code stream (see F.2.2.3). The values SSSS and R are derived from RS. The value of SSSS is the four low order bits of the composite value and R contains the value of RRRR (the four high order bits of the composite value). The interpretation of these values is described in F.1.2.2. EXTEND is shown in Figure F.12.

F.2.2.3 The DECODE procedure

The DECODE procedure decodes an 8-bit value which, for the DC coefficient, determines the difference magnitude category. For the AC coefficient this 8-bit value determines the zero run length and non-zero coefficient category.

Three tables, HUFFVAL, HUFFCODE, and HUFFSIZE, have been defined in Annex C. This particular implementation of DECODE makes use of the ordering of the Huffman codes in HUFFCODE according to both value and code size. Many other implementations of DECODE are possible.

NOTE – The values in HUFFVAL are assigned to each code in HUFFCODE and HUFFSIZE in sequence. There are no ordering requirements for the values in HUFFVAL which have assigned codes of the same length.

The implementation of DECODE described in this subclause uses three tables, MINCODE, MAXCODE and VALPTR, to decode a pointer to the HUFFVAL table. MINCODE, MAXCODE and VALPTR each have 16 entries, one for each possible code size. MINCODE(I) contains the smallest code value for a given length I, MAXCODE(I) contains the largest code value for a given length I, and VALPTR(I) contains the index to the start of the list of values in HUFFVAL which are decoded by code words of length I. The values in MINCODE and MAXCODE are signed 16-bit integers; therefore, a value of –1 sets all of the bits.

The procedure for generating these tables is shown in Figure F.15. The procedure for DECODE is shown in Figure F.16. Note that the 8-bit "VALUE" is returned to the procedure which invokes DECODE.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

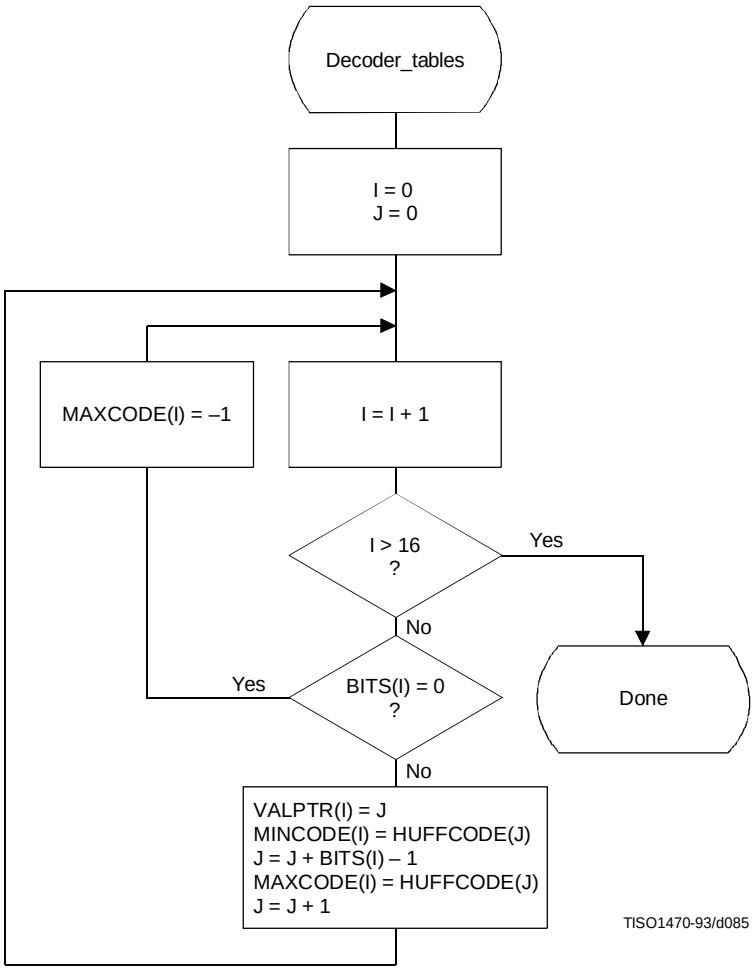


Figure F.15 – Decoder table generation

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

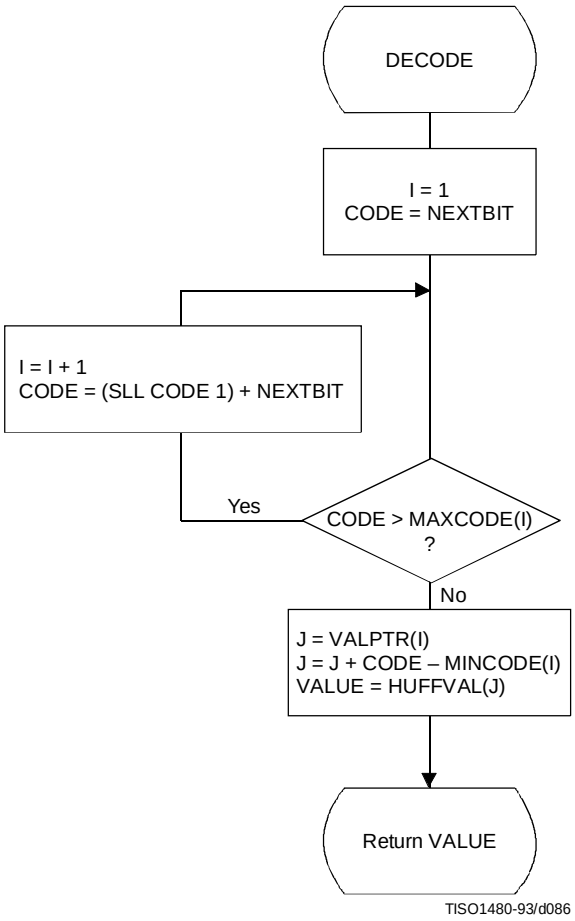


Figure F.16 – Procedure for DECODE

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

F.2.2.4 The RECEIVE procedure

RECEIVE(SSSS) is a procedure which places the next SSSS bits of the entropy-coded segment into the low order bits of DIFF, MSB first. It calls NEXTBIT and it returns the value of DIFF to the calling procedure (see Figure F.17).

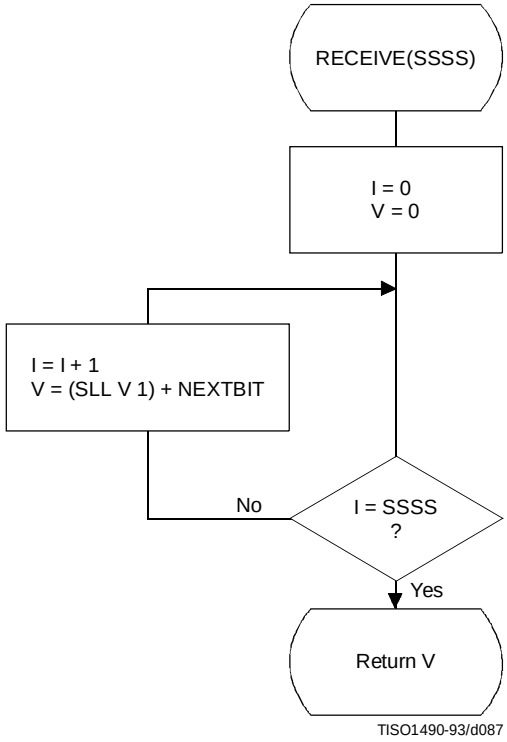


Figure F.17 – Procedure for RECEIVE(SSSS)

F.2.2.5 The NEXTBIT procedure

NEXTBIT reads the next bit of compressed data and passes it to higher level routines. It also intercepts and removes stuff bytes and detects markers. NEXTBIT reads the bits of a byte starting with the MSB (see Figure F.18).

Before starting the decoding of a scan, and after processing a RST marker, CNT is cleared. The compressed data are read one byte at a time, using the procedure NEXTBYTE. Each time a byte, B, is read, CNT is set to 8.

The only valid marker which may occur within the Huffman coded data is the RST_m marker. Other than the EOI or markers which may occur at or before the start of a scan, the only marker which can occur at the end of the scan is the DNL (define-number-of-lines).

Normally, the decoder will terminate the decoding at the end of the final restart interval before the terminating marker is intercepted. If the DNL marker is encountered, the current line count is set to the value specified by that marker. Since the DNL marker can only be used at the end of the first scan, the scan decode procedure must be terminated when it is encountered.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

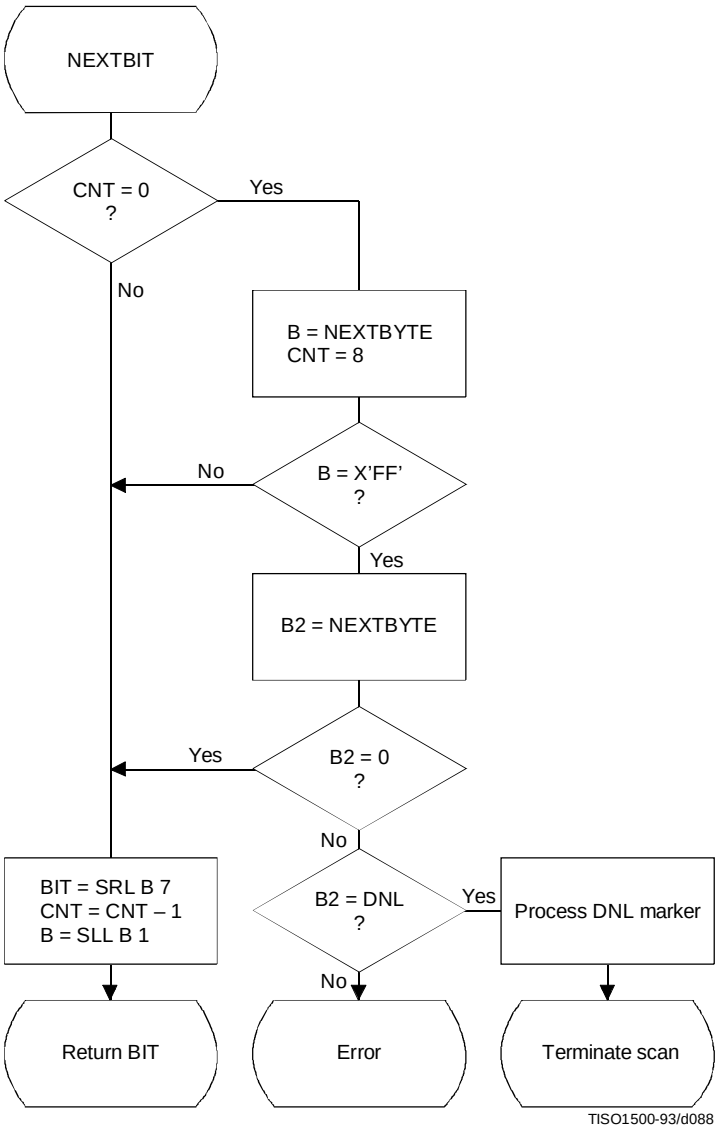


Figure F.18 – Procedure for fetching the next bit of compressed data

F.2.3 Sequential DCT decoding process with 8-bit precision extended to four sets of Huffman tables

This process is identical to the Baseline decoding process described in F.2.2, with the exception that the decoder shall be capable of using up to four DC and four AC Huffman tables within one scan. Four DC and four AC Huffman tables is the maximum allowed by this Specification.

F.2.4 Sequential DCT decoding process with arithmetic coding

This subclause describes the sequential DCT decoding process with arithmetic decoding.

The arithmetic decoding procedures for decoding binary decisions, initializing the statistical model, initializing the decoder, and resynchronizing the decoder are listed in Table D.4 of Annex D.

Some of the procedures in Table D.4 are used in the higher level control structure for scans and restart intervals described in F.2. At the beginning of scans and restart intervals, the probability estimates used in the arithmetic decoder are reset to the standard initial value as part of the Initdec procedure which restarts the arithmetic coder.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The statistical models defined in F.1.4.4 also apply to this decoding process.

The decoder shall be capable of using up to four DC and four AC conditioning tables and associated statistics areas within one scan.

F.2.4.1 Arithmetic decoding of DC coefficients

The basic structure of the decision sequence for decoding a DC difference value, DIFF, is shown in Figure F.19. The equivalent structure for the encoder is found in Figure F.4.

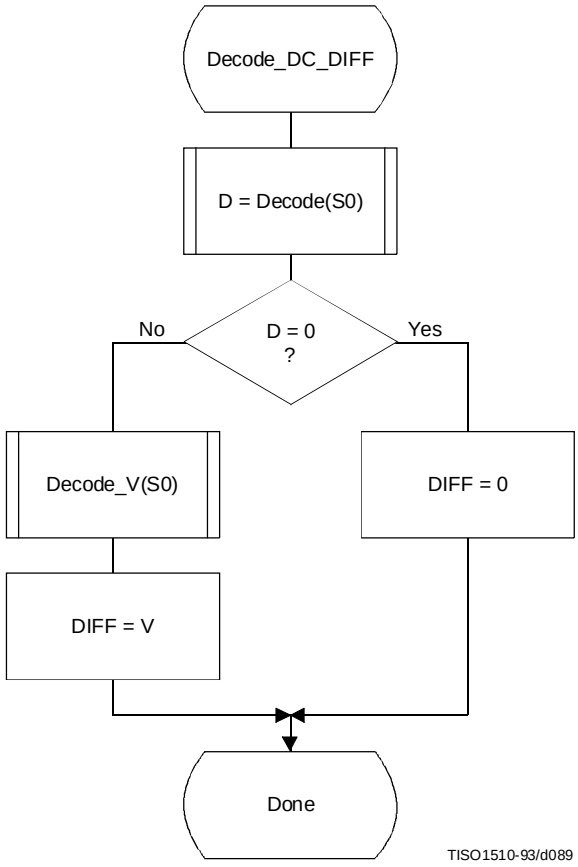


Figure F.19 – Arithmetic decoding of DC difference

The context-indices used in the DC decoding procedures are defined in Table F.4 (see F.1.4.4.1.3).

The “Decode” procedure returns the value “D” of the binary decision. If the value is not zero, the sign and magnitude of the non-zero DIFF must be decoded by the procedure “Decode_V(S0)”.

F.2.4.2 Arithmetic Decoding of AC coefficients

The AC coefficients are decoded in the order that they occur in ZZ(1,...,63). The encoder procedure for the coding process is found in Figure F.5. Figure F.20 illustrates the decoding sequence.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

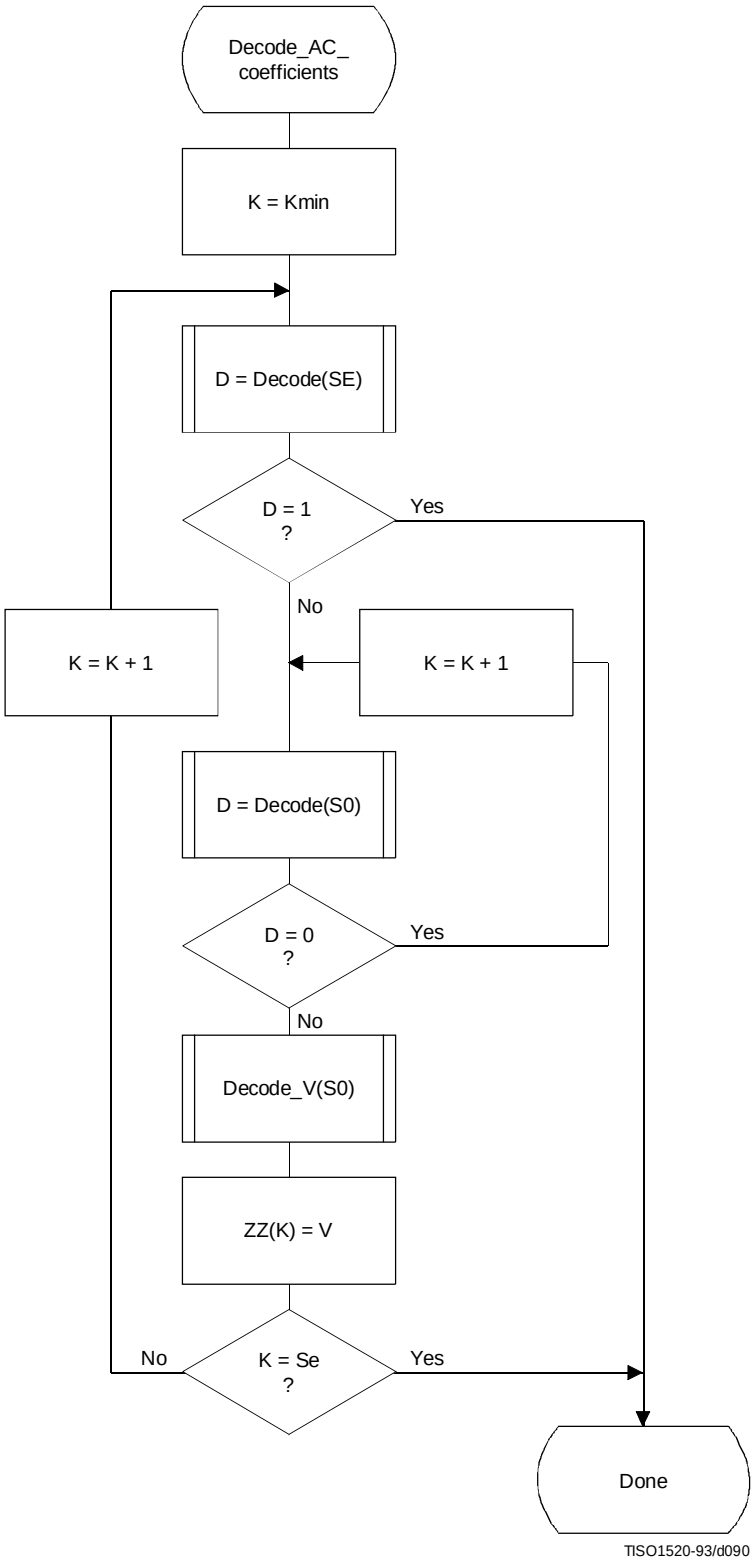


Figure F.20 – Procedure for decoding the AC coefficients

The context-indices used in the AC decoding procedures are defined in Table F.5 (see F.1.4.4.2).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

In Figure F.20, K is the index to the zig-zag sequence position. For the sequential scan, Kmin = 1 and Se = 63. The decision at the top of the loop is the EOB decision. If the EOB occurs (D = 1), the remaining coefficients in the block are set to zero. The inner loop just below the EOB decoding decodes runs of zero coefficients. Whenever the coefficient is non-zero, "Decode_V" decodes the sign and magnitude of the coefficient. After each non-zero coefficient is decoded, the EOB decision is again decoded unless K = Se.

F.2.4.3 Decoding the binary decision sequence for non-zero DC differences and AC coefficients

Both the DC difference and the AC coefficients are represented as signed two's complement 16-bit integer values. The decoding decision tree for these signed integer values is the same for both the DC and AC coding models. Note, however, that the statistical models are not the same.

F.2.4.3.1 Arithmetic decoding of non-zero values

Denoting either DC differences or AC coefficients as V, the non-zero signed integer value of V is decoded by the sequence shown in Figure F.21. This sequence first decodes the sign of V. It then decodes the magnitude category of V (Decode_log2_Sz), and then decodes the low order magnitude bits (Decode_Sz_bits). Note that the value decoded for Sz must be incremented by 1 to get the actual coefficient magnitude.

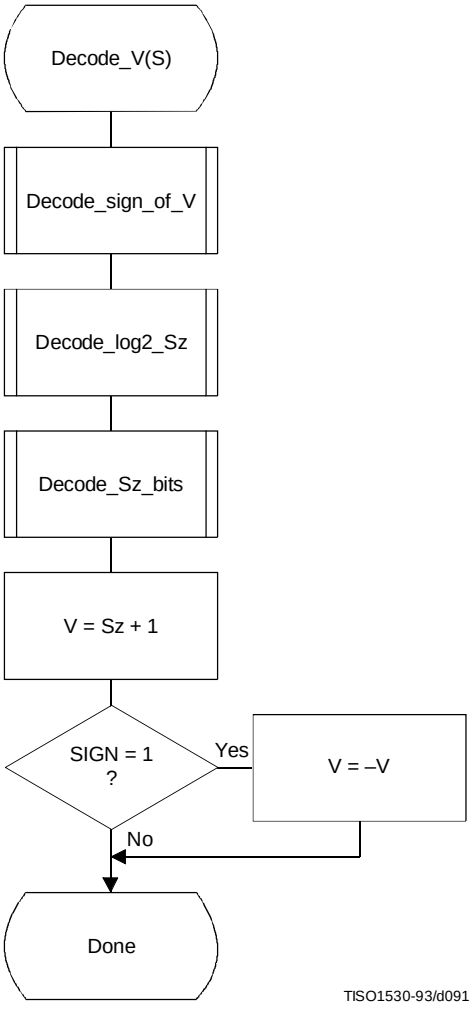


Figure F.21 – Sequence of procedures in decoding non-zero values of V

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

F.2.4.3.1.1 Decoding the sign

The sign is decoded by the procedure shown in Figure F.22.

The context-indices are defined for DC decoding in Table F.4 and AC decoding in Table F.5.

If SIGN = 0, the sign of the coefficient is positive; if SIGN = 1, the sign of the coefficient is negative.

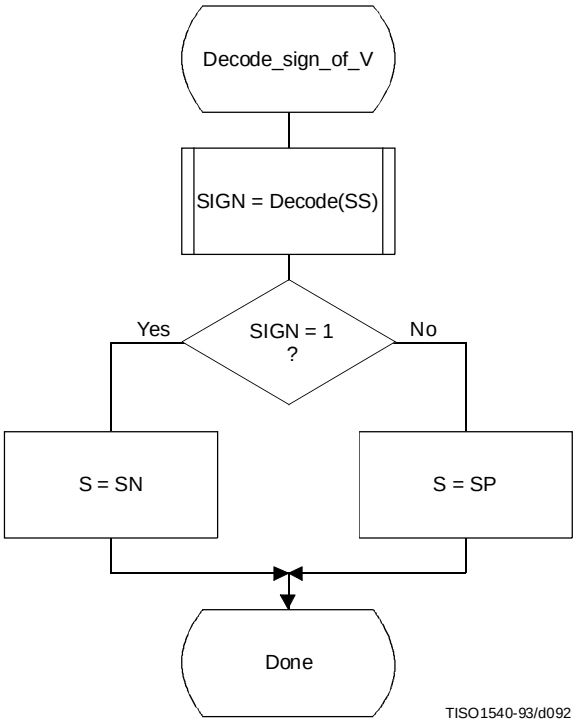


Figure F.22 – Decoding the sign of V

F.2.4.3.1.2 Decoding the magnitude category

The context-index S is set in Decode_sign_of_V and the context-index values X1 and X2 are defined for DC coding in Table F.4 and for AC coding in Table F.5.

In Figure F.23, M is set to the upper bound for the magnitude and shifted left until the decoded decision is zero. It is then shifted right by 1 to become the leading bit of the magnitude of Sz.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

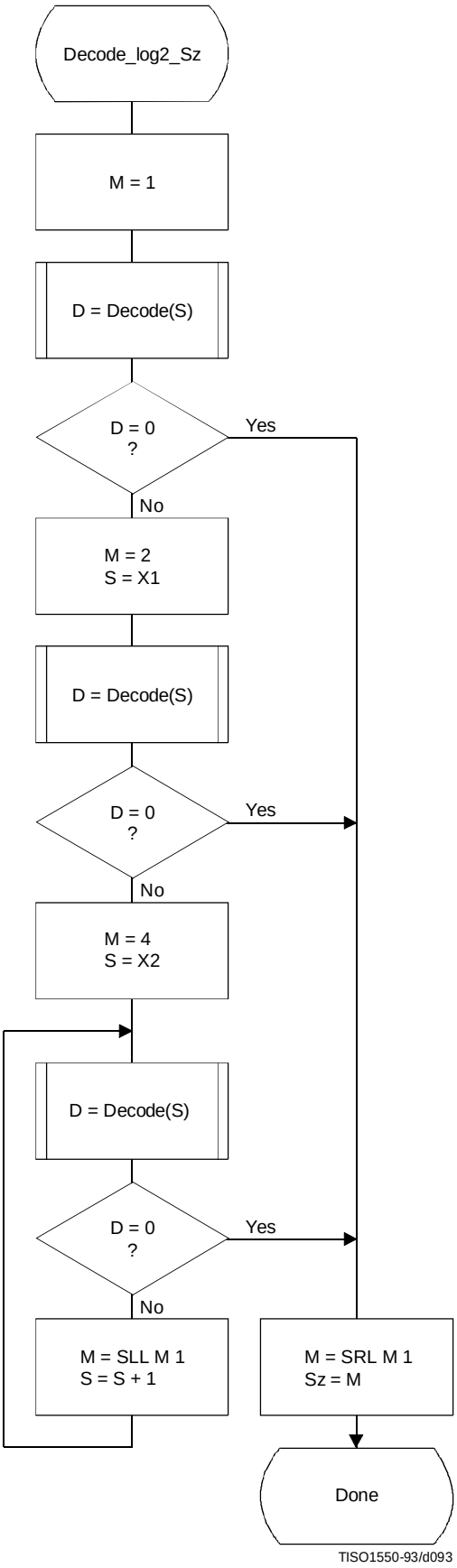


Figure F.23 – Decoding procedure to establish the magnitude category

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

F.2.4.3.1.3 Decoding the exact value of the magnitude

After the magnitude category is decoded, the low order magnitude bits are decoded. These bits are decoded in order of decreasing bit significance. The procedure is shown in Figure F.24.

The context-index S is set in Decode_log2_Sz.

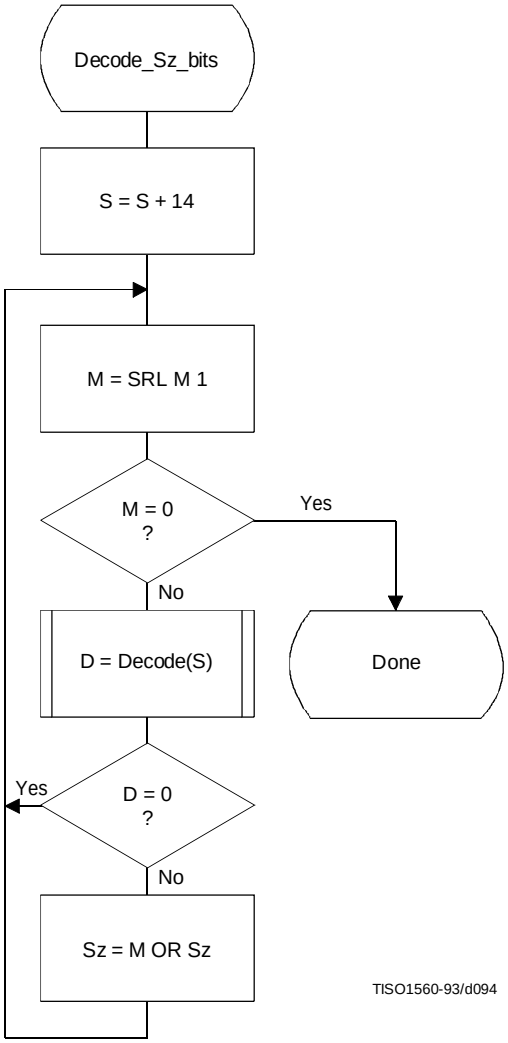


Figure F.24 – Decision sequence to decode the magnitude bit pattern

F.2.4.4 Decoder restart

The RST_m markers which are added to the compressed data between each restart interval have a two byte value which cannot be generated by the coding procedures. These two byte sequences can be located without decoding, and can therefore be used to resynchronize the decoder. RST_m markers can therefore be used for error recovery.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Before error recovery procedures can be invoked, the error condition must first be detected. Errors during decoding can show up in two places:

a) The decoder fails to find the expected marker at the point where it is expecting resynchronization.

b) Physically impossible data are decoded. For example, decoding a magnitude beyond the range of values allowed by the model is quite likely when the compressed data are corrupted by errors. For arithmetic decoders this error condition is extremely important to detect, as otherwise the decoder may reach a condition where it uses the compressed data very slowly.

NOTE – Some errors will not cause the decoder to lose synchronization. In addition, recovery is not possible for all errors; for example, errors in the headers are likely to be catastrophic. The two error conditions listed above, however, almost always cause the decoder to lose synchronization in a way which permits recovery.

In regaining synchronization, the decoder can make use of the modulo 8 coding restart interval number in the low order bits of the RST_m marker. By comparing the expected restart interval number to the value in the next RST_m marker in the compressed image data, the decoder can usually recover synchronization. It then fills in missing lines in the output data by replication or some other suitable procedure, and continues decoding. Of course, the reconstructed image will usually be highly corrupted for at least a part of the restart interval where the error occurred.

F.2.5 Sequential DCT decoding process with Huffman coding and 12-bit precision

This process is identical to the sequential DCT process defined for 8-bit sample precision and extended to four Huffman tables, as documented in F.2.3, but with the following changes.

F.2.5.1 Structure of DC Huffman decode table

The general structure of the DC Huffman decode table is extended as described in F.1.5.1.

F.2.5.2 Structure of AC Huffman decode table

The general structure of the AC Huffman decode table is extended as described in F.1.5.2.

F.2.6 Sequential DCT decoding process with arithmetic coding and 12-bit precision

The process is identical to the sequential DCT process for 8-bit precision except for changes in the precision of the IDCT computation.

The structure of the decoding procedure in F.2.4 is already defined for a 12-bit input precision.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex G

Progressive DCT-based mode of operation

(This annex forms an integral part of this Recommendation | International Standard)

This annex provides a **functional specification** of the following coding processes for the progressive DCT-based mode of operation:

1) spectral selection only, Huffman coding, 8-bit sample precision;
2) spectral selection only, arithmetic coding, 8-bit sample precision;
3) full progression, Huffman coding, 8-bit sample precision;
4) full progression, arithmetic coding, 8-bit sample precision;
5) spectral selection only, Huffman coding, 12-bit sample precision;
6) spectral selection only, arithmetic coding, 12-bit sample precision;
7) full progression, Huffman coding, 12-bit sample precision;
8) full progression, arithmetic coding, 12-bit sample precision.

For each of these, the encoding process is specified in G.1, and the decoding process is specified in G.2. The functional specification is presented by means of specific flow charts for the various procedures which comprise these coding processes.

NOTE – There is **no requirement** in this Specification that any encoder or decoder which embodies one of the above-named processes shall implement the procedures in precisely the manner specified by the flow charts in this annex. It is necessary only that an encoder or decoder implement the **function** specified in this annex. The sole criterion for an encoder or decoder to be considered in compliance with this Specification is that it satisfy the requirements given in clause 6 (for encoders) or clause 7 (for decoders), as determined by the compliance tests specified in Part 2.

The number of Huffman or arithmetic conditioning tables which may be used within the same scan is four.

Two complementary progressive procedures are defined, spectral selection and successive approximation.

In spectral selection the DCT coefficients of each block are segmented into frequency bands. The bands are coded in separate scans.

In successive approximation the DCT coefficients are divided by a power of two before coding. In the decoder the coefficients are multiplied by that same power of two before computing the IDCT. In the succeeding scans the precision of the coefficients is increased by one bit in each scan until full precision is reached.

An encoder or decoder implementing a full progression uses spectral selection within successive approximation. An allowed subset is spectral selection alone.

Figure G.1 illustrates the spectral selection and successive approximation progressive processes.

G.1 Progressive DCT-based encoding processes

G.1.1 Control procedures and coding models for progressive DCT-based procedures

G.1.1.1 Control procedures for progressive DCT-based encoders

The control procedures for encoding an image and its constituent parts – the frame, scan, restart interval and MCU – are given in Figures E.1 through E.5.

The control structure for encoding a frame is the same as for the sequential procedures. However, it is convenient to calculate the FDCT for the entire set of components in a frame before starting the scans. A buffer which is large enough to store all of the DCT coefficients may be used for this progressive mode of operation.

The number of scans is determined by the progression defined; the number of scans may be much larger than the number of components in the frame.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

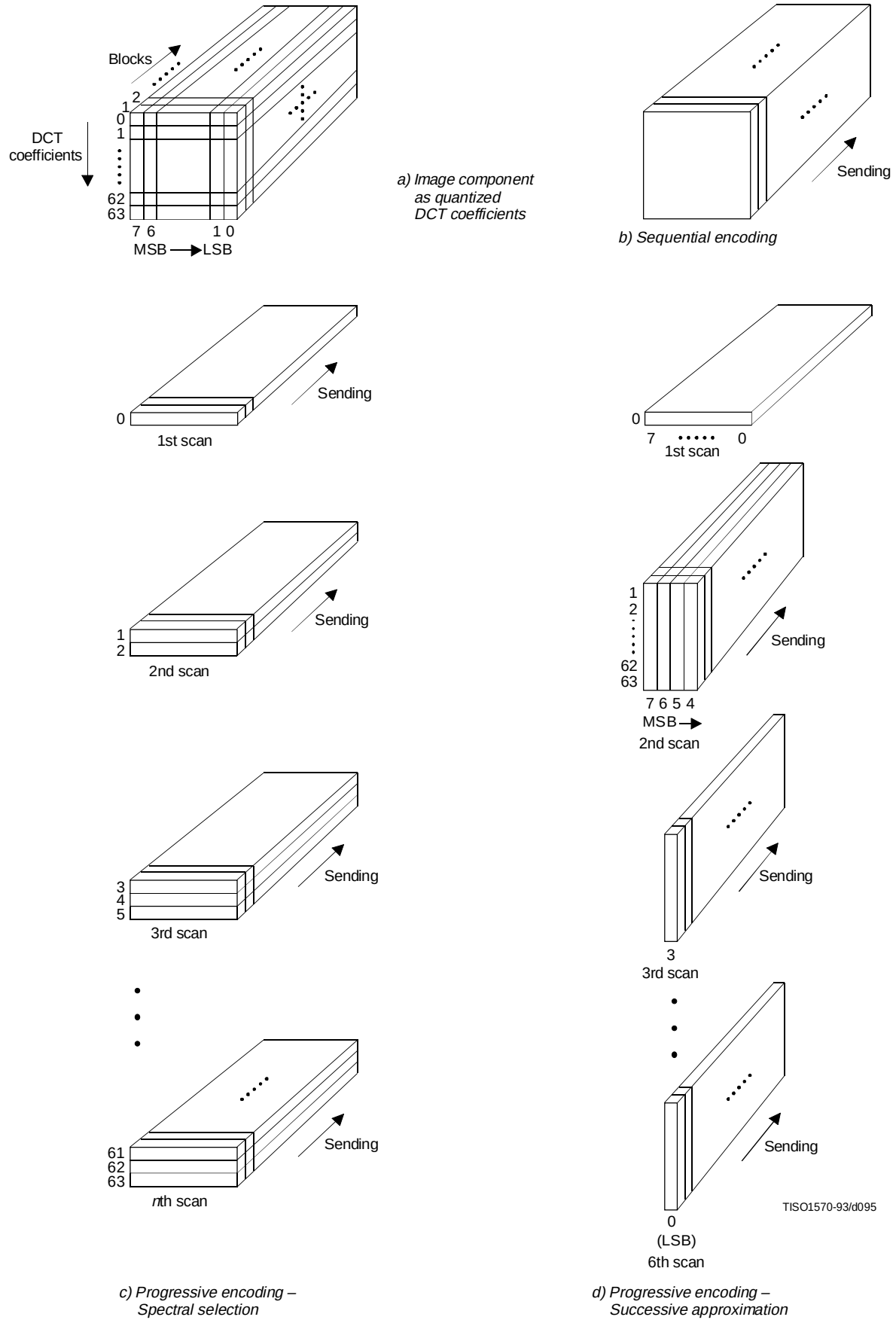


Figure G.1 – Spectral selection and successive approximation progressive processes

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The procedure for encoding a MCU (see Figure E.5) repetitively invokes the procedure for coding a data unit. For DCT-based encoders the data unit is an 8×8 block of samples.

Only a portion of each 8×8 block is coded in each scan, the portion being determined by the scan header parameters Ss, Se, Ah, and Al (see B.2.3). The procedures used to code portions of each 8×8 block are described in this annex. Note, however, that where these procedures are identical to those used in the sequential DCT-based mode of operation, the sequential procedures are simply referenced.

G.1.1.1.1 Spectral selection control

In spectral selection the zig-zag sequence of DCT coefficients is segmented into bands. A band is defined in the scan header by specifying the starting and ending indices in the zig-zag sequence. One band is coded in a given scan of the progression. DC coefficients are always coded separately from AC coefficients, and only scans which code DC coefficients may have interleaved blocks from more than one component. All other scans shall have only one component. With the exception of the first DC scans for the components, the sequence of bands defined in the scans need not follow the zig-zag ordering. For each component, a first DC scan shall precede any AC scans.

G.1.1.1.2 Successive approximation control

If successive approximation is used, the DCT coefficients are reduced in precision by the point transform (see A.4) defined in the scan header (see B.2.3). The successive approximation bit position parameter Al specifies the actual point transform, and the high four bits (Ah) – if there are preceding scans for the band – contain the value of the point transform used in those preceding scans. If there are no preceding scans for the band, Ah is zero.

Each scan which follows the first scan for a given band progressively improves the precision of the coefficients by one bit, until full precision is reached.

G.1.1.2 Coding models for progressive DCT-based encoders

If successive approximation is used, the DCT coefficients are reduced in precision by the point transform (see A.4) defined in the scan header (see B.2.3). These models also apply to the progressive DCT-based encoders, but with the following changes.

G.1.1.2.1 Progressive encoding model for DC coefficients

If Al is not zero, the point transform for DC coefficients shall be used to reduce the precision of the DC coefficients. If Ah is zero, the coefficient values (as modified by the point transform) shall be coded, using the procedure described in Annex F. If Ah is not zero, the least significant bit of the point transformed DC coefficients shall be coded, using the procedures described in this annex.

G.1.1.2.2 Progressive encoding model for AC coefficients

If Al is not zero, the point transform for AC coefficients shall be used to reduce the precision of the AC coefficients. If Ah is zero, the coefficient values (as modified by the point transform) shall be coded using modifications of the procedures described in Annex F. These modifications are described in this annex. If Ah is not zero, the precision of the coefficients shall be improved using the procedures described in this annex.

G.1.2 Progressive encoding procedures with Huffman coding

G.1.2.1 Progressive encoding of DC coefficients with Huffman coding

The first scan for a given component shall encode the DC coefficient values using the procedures described in F.1.2.1. If the successive approximation bit position parameter Al is not zero, the coefficient values shall be reduced in precision by the point transform described in Annex A before coding.

In subsequent scans using successive approximation the least significant bits are appended to the compressed bit stream without compression or modification (see G.1.2.3), except for byte stuffing.

G.1.2.2 Progressive encoding of AC coefficients with Huffman coding

In spectral selection and in the first scan of successive approximation for a component, the AC coefficient coding model is similar to that used by the sequential procedures. However, the Huffman code tables are extended to include coding of runs of End-Of-Bands (EOBs). See Table G.1.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table G.1 – EOBn code run length extensions

EOBn code	Run length
EOB0	1
EOB1	2,3
EOB2	4..7
EOB3	8..15
EOB4	16..31
EOB5	32..63
EOB6	64..127
EOB7	128..255
EOB8	256..511
EOB9	512..1 023
EOB10	1 024..2 047
EOB11	2 048..4 095
EOB12	4 096..8 191
EOB13	8 192..16 383
EOB14	16 384..32 767

The end-of-band run structure allows efficient coding of blocks which have only zero coefficients. An EOB run of length 5 means that the current block and the next four blocks have an end-of-band with no intervening non-zero coefficients. The EOB run length is limited only by the restart interval.

The extension of the code table is illustrated in Figure G.2.

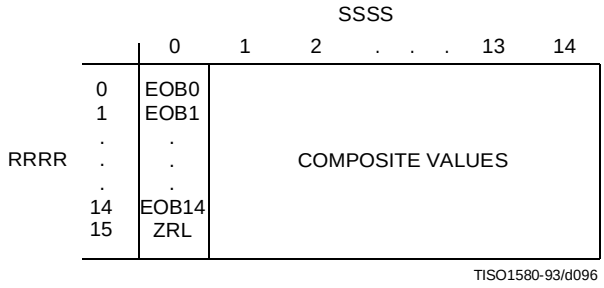


Figure G.2 – Two-dimensional value array for Huffman coding

The EOBn code sequence is defined as follows. Each EOBn code is followed by an extension field similar to the extension field for the coefficient amplitudes (but with positive numbers only). The number of bits appended to the EOBn code is the minimum number required to specify the run length.

If an EOB run is greater than 32 767, it is coded as a sequence of EOB runs of length 32 767 followed by a final EOB run sufficient to complete the run.

At the beginning of each restart interval the EOB run count, EOBRUN, is set to zero. At the end of each restart interval any remaining EOB run is coded.

The Huffman encoding procedure for AC coefficients in spectral selection and in the first scan of successive approximation is illustrated in Figures G.3, G.4, G.5, and G.6.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

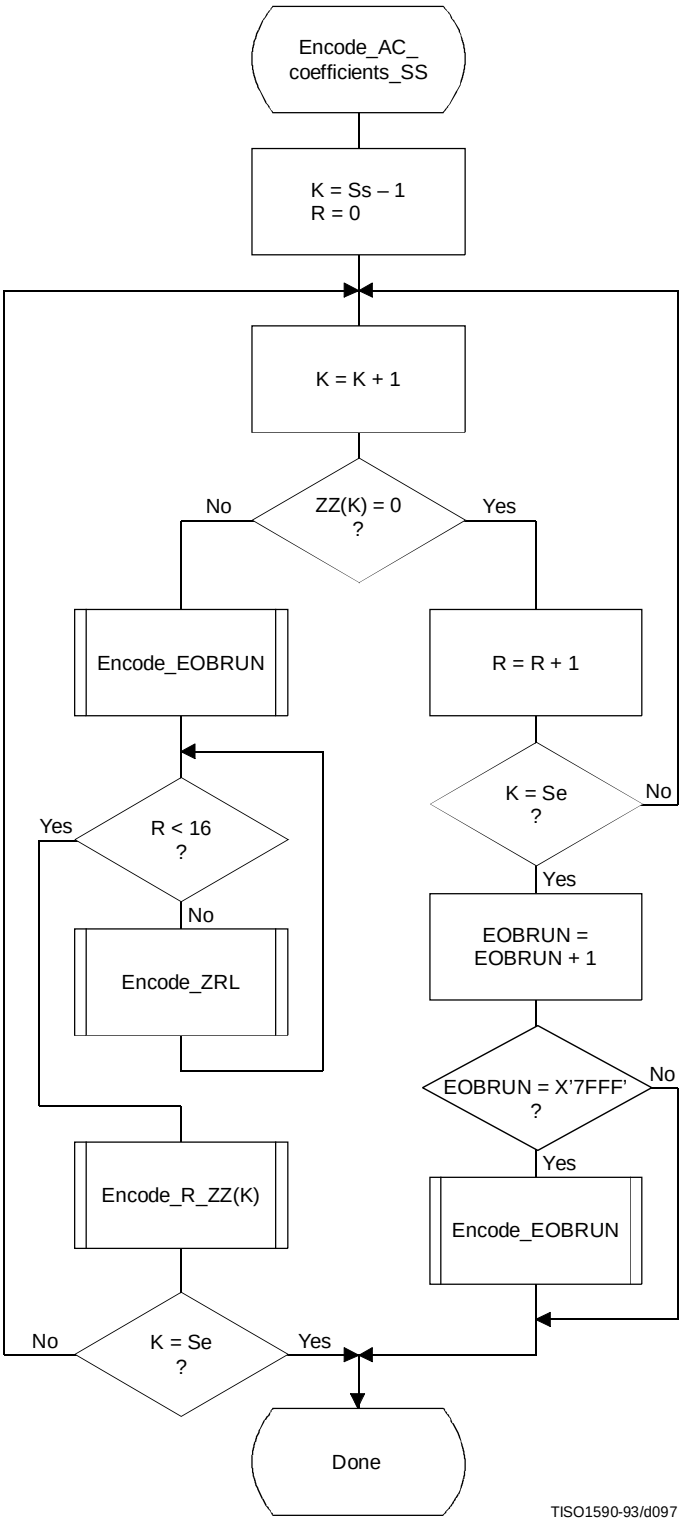


Figure G.3 – Procedure for progressive encoding of AC coefficients with Huffman coding

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

In Figure G.3, Ss is the start of spectral selection, Se is the end of spectral selection, K is the index into the list of coefficients stored in the zig-zag sequence ZZ, R is the run length of zero coefficients, and EOBRUN is the run length of EOBs. EOBRUN is set to zero at the start of each restart interval.

If the scan header parameter Al (successive approximation bit position low) is not zero, the DCT coefficient values ZZ(K) in Figure G.3 and figures which follow in this annex, including those in the arithmetic coding section, shall be replaced by the point transformed values ZZ'(K), where ZZ'(K) is defined by:

$$ZZ'(K) = \frac{ZZ(K)x}{2^{Al}}$$

EOBSIZE is a procedure which returns the size of the EOB extension field given the EOB run length as input. CSIZE is a procedure which maps an AC coefficient to the SSSS value defined in the subclauses on sequential encoding (see F.1.1 and F.1.3).

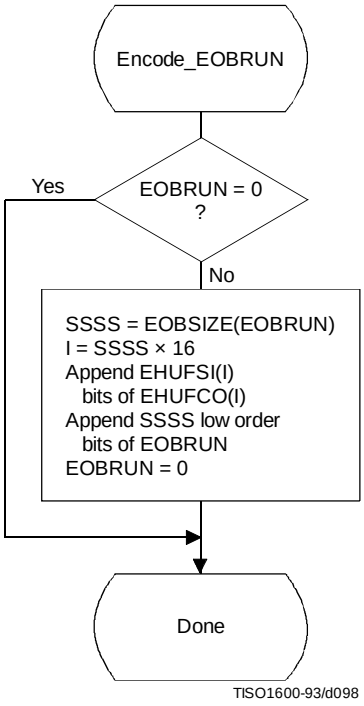


Figure G.4 – Progressive encoding of a non-zero AC coefficient

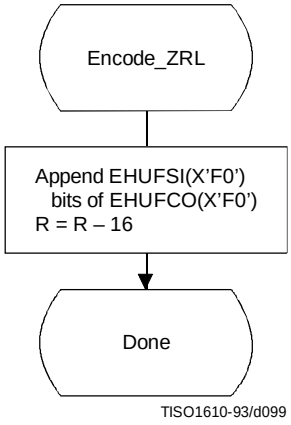


Figure G.5 – Encoding of the run of zero coefficients

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

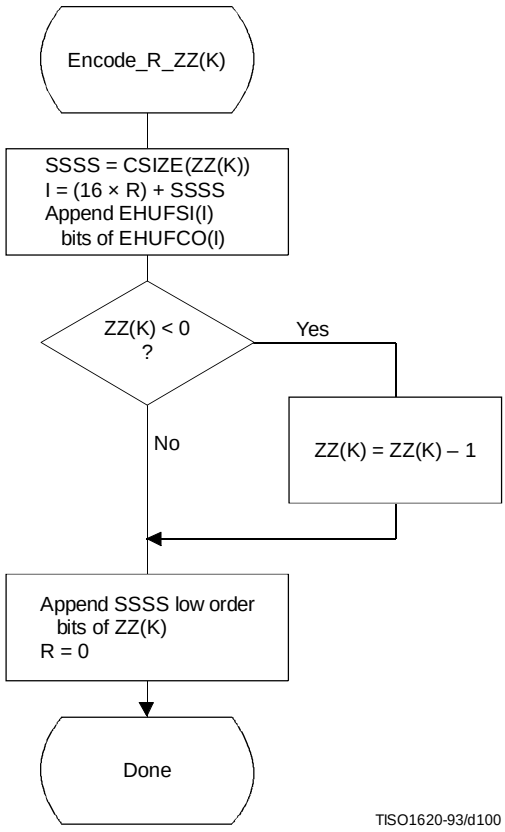


Figure G.6 – Encoding of the zero run and non-zero coefficient

G.1.2.3 Coding model for subsequent scans of successive approximation

The Huffman coding structure of the subsequent scans of successive approximation for a given component is similar to the coding structure of the first scan of that component.

The structure of the AC code table is identical to the structure described in G.1.2.2. Each non-zero point transformed coefficient that has a zero history (i.e. that has a value ± 1, and therefore has not been coded in a previous scan) is defined by a composite 8-bit run length-magnitude value of the form:

RRRRSSSS

The four most significant bits, RRRR, give the number of zero coefficients that are between the current coefficient and the previously coded coefficient (or the start of band). Coefficients with non-zero history (a non-zero value coded in a previous scan) are skipped over when counting the zero coefficients. The four least significant bits, SSSS, provide the magnitude category of the non-zero coefficient; for a given component the value of SSSS can only be one.

The run length-magnitude composite value is Huffman coded and each Huffman code is followed by additional bits:

a) One bit codes the sign of the newly non-zero coefficient. A 0-bit codes a negative sign; a 1-bit codes a positive sign.

b) For each coefficient with a non-zero history, one bit is used to code the correction. A 0-bit means no correction and a 1-bit means that one shall be added to the (scaled) decoded magnitude of the coefficient.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Non-zero coefficients with zero history are coded with a composite code of the form:

HUFFCO(RRRRSSSS) + additional bit (rule a) + correction bits (rule b)

In addition whenever zero runs are coded with ZRL or EOBn codes, correction bits for those coefficients with non-zero history contained within the zero run are appended according to rule b above.

For the Huffman coding version of Encode_AC_Coefficients_SA the EOB is defined to be the position of the last point transformed coefficient of magnitude 1 in the band. If there are no coefficients of magnitude 1, the EOB is defined to be zero.

NOTE – The definition of EOB is different for Huffman and arithmetic coding procedures.

In Figures G.7 and G.8 BE is the count of buffered correction bits at the start of coding of the block. BE is initialized to zero at the start of each restart interval. At the end of each restart interval any remaining buffered bits are appended to the bit stream following the last EOBn Huffman code and associated appended bits.

In Figures G.7 and G.9, BR is the count of buffered correction bits which are appended to the bit stream according to rule b. BR is set to zero at the beginning of each Encode_AC_Coefficients_SA. At the end of each restart interval any remaining buffered bits are appended to the bit stream following the last Huffman code and associated appended bits.

G.1.3 Progressive encoding procedures with arithmetic coding

G.1.3.1 Progressive encoding of DC coefficients with arithmetic coding

The first scan for a given component shall encode the DC coefficient values using the procedures described in F.1.4.1. If the successive approximation bit position parameter is not zero, the coefficient values shall be reduced in precision by the point transform described in Annex A before coding.

In subsequent scans using successive approximation the least significant bits shall be coded as binary decisions using a fixed probability estimate of 0.5 (Qe = X'5A1D', MPS = 0).

G.1.3.2 Progressive encoding of AC coefficients with arithmetic coding

Except for the point transform scaling of the DCT coefficients and the grouping of the coefficients into bands, the first scan(s) of successive approximation is identical to the sequential encoding procedure described in F.1.4. If Kmin is equated to Ss, the index of the first AC coefficient index in the band, the flow chart shown in Figure F.5 applies. The EOB decision in that figure refers to the "end-of-band" rather than the "end-of-block". For the arithmetic coding version of Encode_AC_Coefficients_SA (and all other AC coefficient coding procedures) the EOB is defined to be the position following the last non-zero coefficient in the band.

NOTE - The definition of EOB is different for Huffman and arithmetic coding procedures.

The statistical model described in F.1.4 also holds. For this model the default value of Kx is 5. Other values of Kx may be specified using the DAC marker code (Annex B). The following calculation for Kx has proven to give good results for 8-bit precision samples:

$$Kx = Kmin + SRL\ (8 + Se - Kmin)\ 4$$

This expression reduces to the default of Kx = 5 when the band is from index 1 to index 63.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

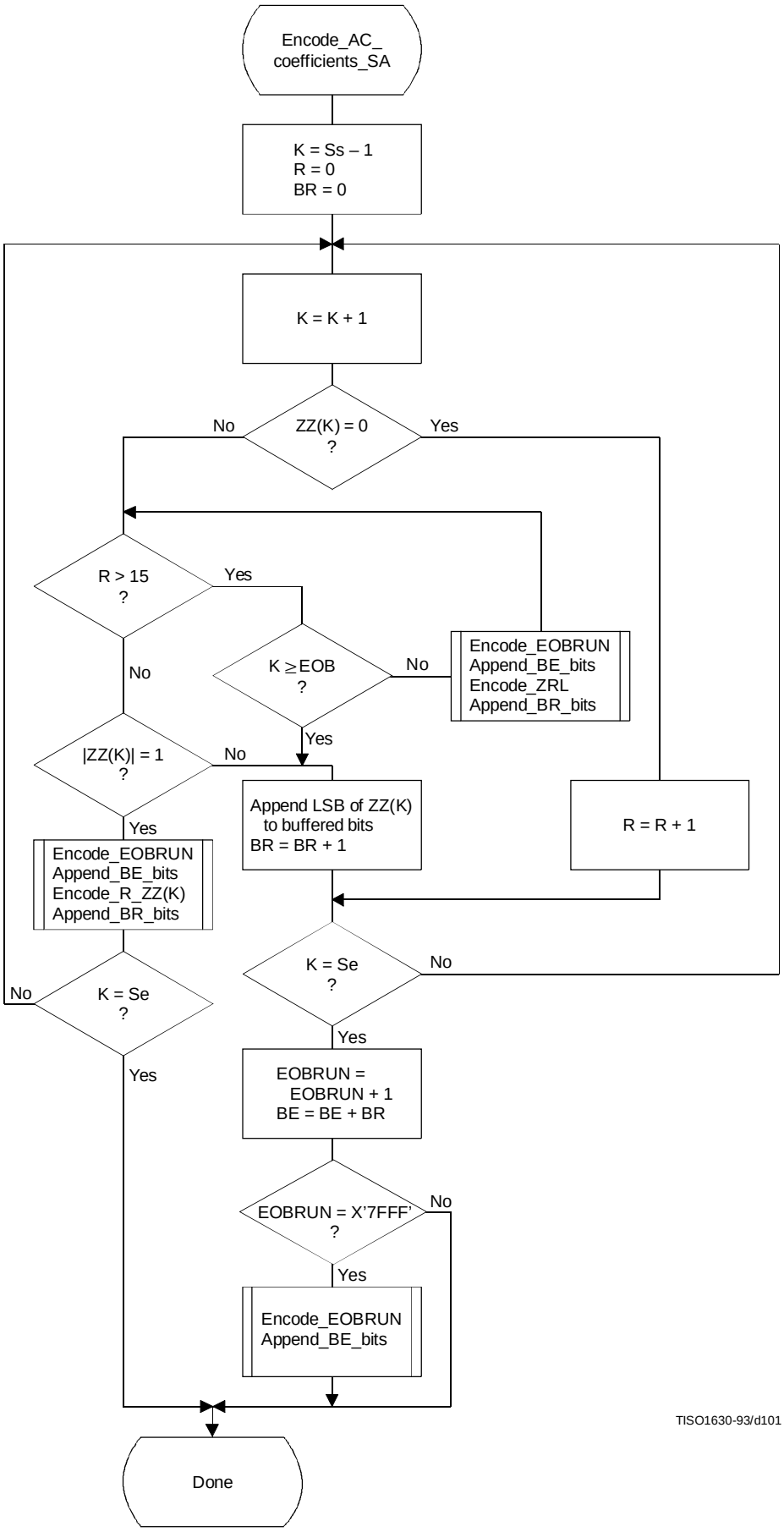


Figure G.7 – Successive approximation coding of AC coefficients using Huffman coding

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

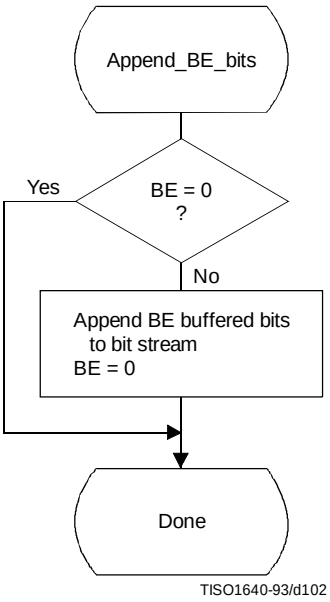


Figure G.8 – Transferring BE buffered bits from buffer to bit stream

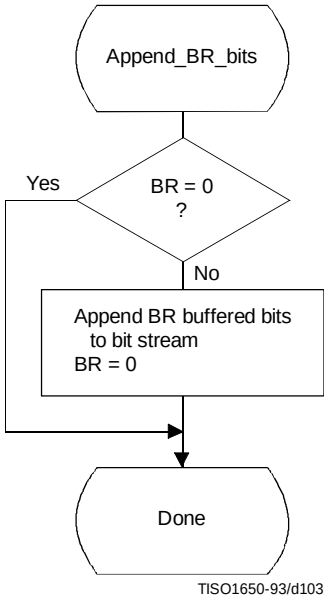


Figure G.9 – Transferring BR buffered bits from buffer to bit stream

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

G.1.3.3 Coding model for subsequent scans of successive approximation

The procedure “Encode_AC_Coefficient_SA” shown in Figure G.10 increases the precision of the AC coefficient values in the band by one bit.

As in the first scan of successive approximation for a component, an EOB decision is coded at the start of the band and after each non-zero coefficient.

However, since the end-of-band index of the previous successive approximation scan for a given component, EOBx, is known from the data coded in the prior scan of that component, this decision is bypassed whenever the current index, K, is less than EOBx. As in the first scan(s), the EOB decision is also bypassed whenever the last coefficient in the band is not zero. The decision ZZ(K) = 0 decodes runs of zero coefficients. If the decoder is at this step of the procedure, at least one non-zero coefficient remains in the band of the block being coded. If ZZ(K) is not zero, the procedure in Figure G.11 is followed to code the value.

The context-indices in Figures G.10 and G.11 are defined in Table G.2 (see G.1.3.3.1). The signs of coefficients with magnitude of one are coded with a fixed probability value of approximately 0.5 (Qe = X’5A1D’, MPS = 0).

G.1.3.3.1 Statistical model for subsequent successive approximation scans

As shown in Table G.2, each statistics area for subsequent successive approximation scans of AC coefficients consists of a contiguous set of 189 statistics bins. The signs of coefficients with magnitude of one are coded with a fixed probability value of approximately 0.5 (Qe = X’5A1D’, MPS = 0).

G.2 Progressive decoding of the DCT

The description of the computation of the IDCT and the dequantization procedure contained in A.3.3 and A.3.4 apply to the progressive operation.

Progressive decoding processes must be able to decompress compressed image data which requires up to four sets of Huffman or arithmetic coder conditioning tables within a scan.

In order to avoid repetition, detailed flow diagrams of progressive decoder operation are not included. Decoder operation is defined by reversing the function of each step described in the encoder flow charts, and performing the steps in reverse order.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

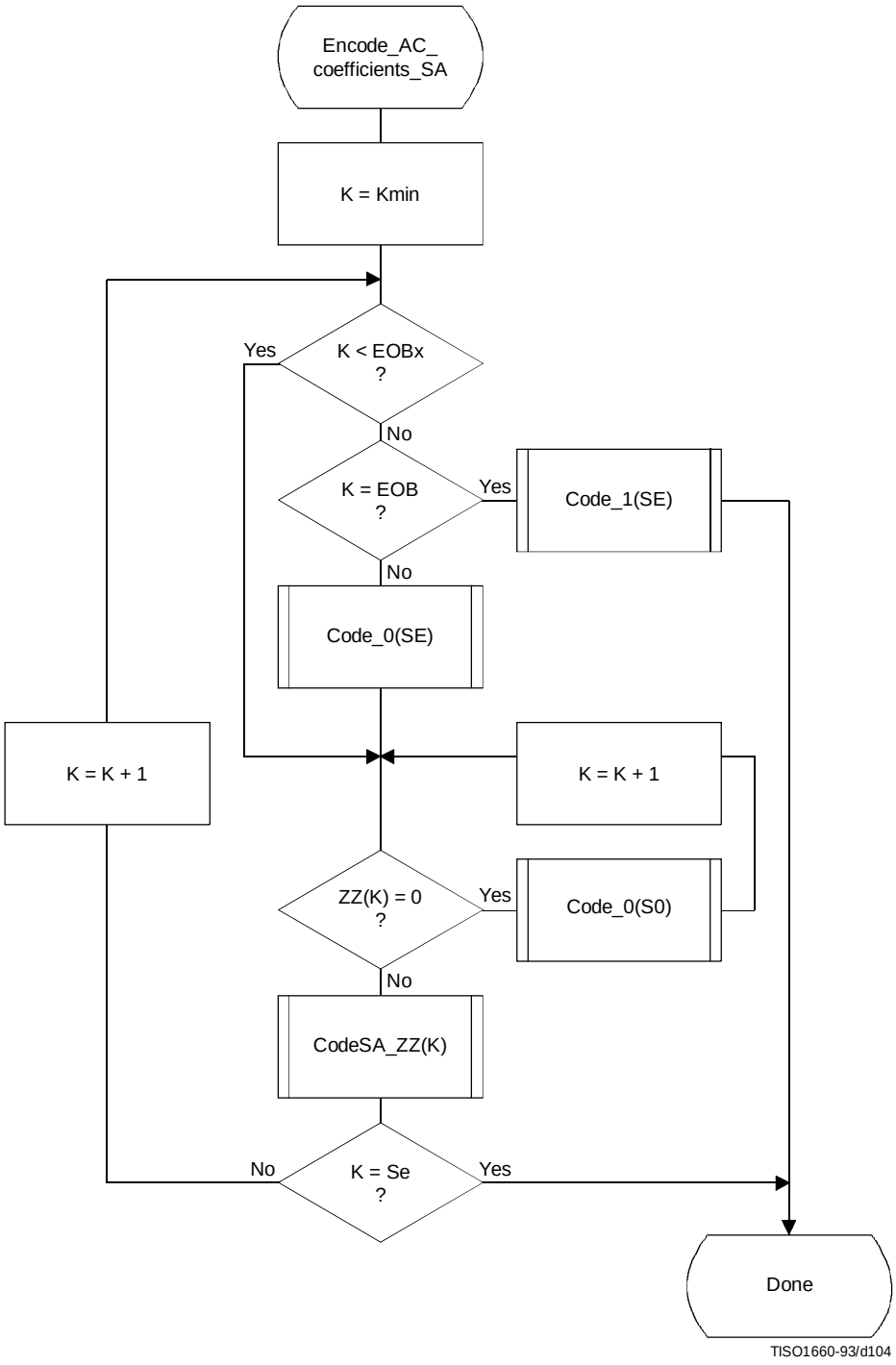


Figure G.10 – Subsequent successive approximation scans for coding of AC coefficients using arithmetic coding

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

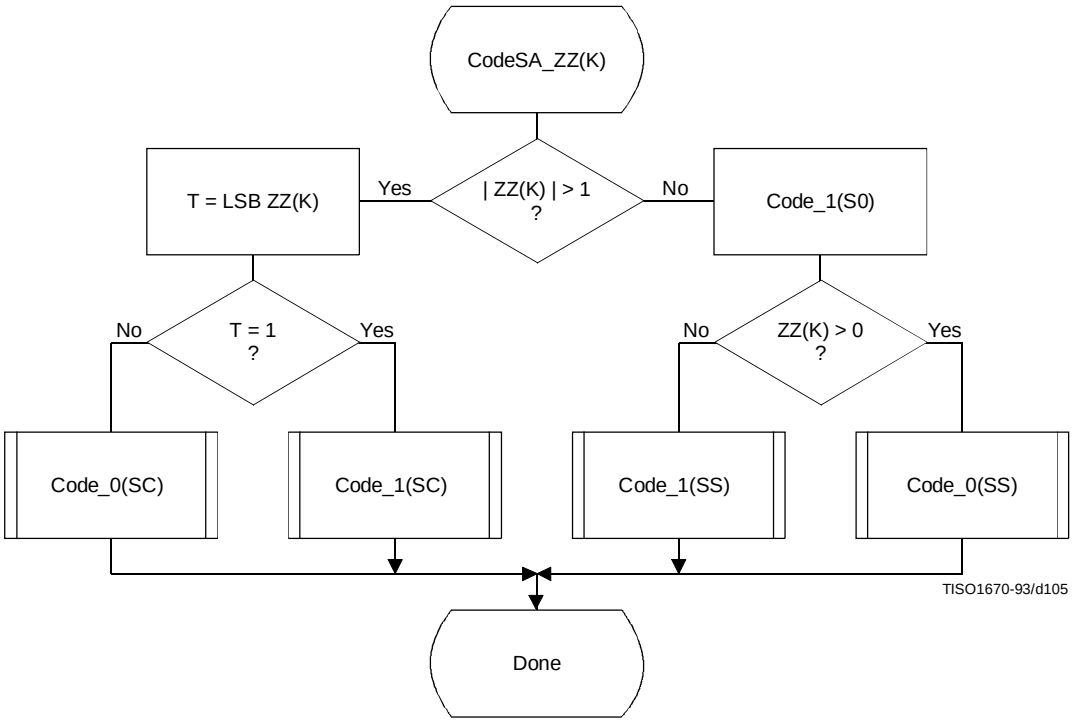


Figure G.11 – Coding non-zero coefficients for subsequent successive approximation scans

Table G.2 – Statistical model for subsequent scans of successive approximation coding of AC coefficient

Context-index	AC coding	Coding decision
SE	$3 \times (K-1)$	K = EOB
S0	SE + 1	V = 0
SS	Fixed estimate	Sign
SC	S0 + 1	LSB ZZ(K) = 1

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex H

Lossless mode of operation

(This annex forms an integral part of this Recommendation | International Standard)

This annex provides a **functional specification** of the following coding processes for the lossless mode of operation:

1) lossless processes with Huffman coding;

2) lossless processes with arithmetic coding.

For each of these, the encoding process is specified in H.1, and the decoding process is specified in H.2. The functional specification is presented by means of specific procedures which comprise these coding processes.

NOTE – There is **no requirement** in this Specification that any encoder or decoder which embodies one of the above-named processes shall implement the procedures in precisely the manner specified in this annex. It is necessary only that an encoder or decoder implement the **function** specified in this annex. The sole criterion for an encoder or decoder to be considered in compliance with this Specification is that it satisfy the requirements given in clause 6 (for encoders) or clause 7 (for decoders), as determined by the compliance tests specified in Part 2.

The processes which provide for sequential lossless encoding and decoding are not based on the DCT. The processes used are spatial processes based on the coding model developed for the DC coefficients of the DCT. However, the model is extended by incorporating a set of selectable one- and two-dimensional predictors, and for interleaved data the ordering of samples for the one-dimensional predictor can be different from that used in the DCT-based processes.

Either Huffman coding or arithmetic coding entropy coding may be employed for these lossless encoding and decoding processes. The Huffman code table structure is extended to allow up to 16-bit precision for the input data. The arithmetic coder statistical model is extended to a two-dimensional form.

H.1 Lossless encoder processes

H.1.1 Lossless encoder control procedures

Subclause E.1 contains the encoder control procedures. In applying these procedures to the lossless encoder, the data unit is one sample.

Input data precision may be from 2 to 16 bits/sample. If the input data path has different precision from the input data, the data shall be aligned with the least significant bits of the input data path. Input data is represented as unsigned integers and is not level shifted prior to coding.

When the encoder is reset in the restart interval control procedure (see E.1.4), the prediction is reset to a default value. If arithmetic coding is used, the statistics are also reset.

For the lossless processes the restart interval shall be an integer multiple of the number of MCU in an MCU-row.

H.1.2 Coding model for lossless encoding

The coding model developed for encoding the DC coefficients of the DCT is extended to allow a selection from a set of seven one-dimensional and two-dimensional predictors. The predictor is selected in the scan header (see Annex B). The same predictor is used for all components of the scan. Each component in the scan is modeled independently, using predictions derived from neighbouring samples of that component.

H.1.2.1 Prediction

Figure H.1 shows the relationship between the positions (a, b, c) of the reconstructed neighboring samples used for prediction and the position of x, the sample being coded.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

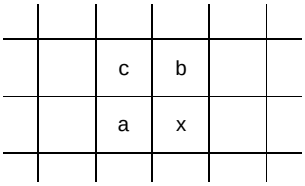


Figure H.1 – Relationship between sample and prediction samples

Define Px to be the prediction and Ra, Rb, and Rc to be the reconstructed samples immediately to the left, immediately above, and diagonally to the left of the current sample. The allowed predictors, one of which is selected in the scan header, are listed in Table H.1.

Table H.1 – Predictors for lossless coding

Selection-value	Prediction
0	No prediction (See Annex J)
1	Px = Ra
2	Px = Rb
3	Px = Rc
4	Px = Ra + Rb – Rc
5	Px = Ra + ((Rb – Rc)/2)[a)]
6	Px = Rb + ((Ra – Rc)/2)[a)]
7	Px = (Ra + Rb)/2

a) Shift right arithmetic operation

Selection-value 0 shall only be used for differential coding in the hierarchical mode of operation. Selections 1, 2 and 3 are one-dimensional predictors and selections 4, 5, 6, and 7 are two-dimensional predictors.

The one-dimensional horizontal predictor (prediction sample Ra) is used for the first line of samples at the start of the scan and at the beginning of each restart interval. The selected predictor is used for all other lines. The sample from the line above (prediction sample Rb) is used at the start of each line, except for the first line. At the beginning of the first line and at the beginning of each restart interval the prediction value of 2^{P-1} is used, where P is the input precision.

If the point transformation parameter (see A.4) is non-zero, the prediction value at the beginning of the first lines and the beginning of each restart interval is 2^{P-Pt-1}, where Pt is the value of the point transformation parameter.

Each prediction is calculated with full integer arithmetic precision, and without clamping of either underflow or overflow beyond the input precision bounds. For example, if Ra and Rb are both 16-bit integers, the sum is a 17-bit integer. After dividing the sum by 2 (predictor 7), the prediction is a 16-bit integer.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

For simplicity of implementation, the divide by 2 in the prediction selections 5 and 6 of Table H.1 is done by an arithmetic-right-shift of the integer values.

The difference between the prediction value and the input is calculated modulo 2^{16}. In the decoder the difference is decoded and added, modulo 2^{16}, to the prediction.

H.1.2.2 Huffman coding of the modulo difference

The Huffman coding procedures defined in Annex F for coding the DC coefficients are used to code the modulo 2^{16} differences. The table for DC coding contained in Tables F.1 and F.6 is extended by one additional entry. No extra bits are appended after SSSS = 16 is encoded. See Table H.2.

Table H.2 – Difference categories for lossless Huffman coding

SSSS	Difference values
0	0
1	–1,1
2	–3,–2,2,3
3	–7..–4,4..7
4	–15..–8,8..15
5	–31..–16,16..31
6	–63..–32,32..63
7	–127..–64,64..127
8	–255..–128,128..255
9	–511..–256,256..511
10	–1 023..–512,512..1 023
11	–2 047..–1 024,1 024..2 047
12	–4 095..–2 048,2 048..4 095
13	–8 191..–4 096,4 096..8 191
14	–16 383..–8 192,8 192..16 383
15	–32 767..–16 384,16 384..32 767
16	32 768

H.1.2.3 Arithmetic coding of the modulo difference

The statistical model defined for the DC coefficient arithmetic coding model (see F.1.4.4.1) is generalized to a two-dimensional form in which differences coded for the sample to the left and for the line above are used for conditioning.

H.1.2.3.1 Two-dimensional statistical model

The binary decisions are conditioned on the differences coded for the neighbouring samples immediately above and immediately to the left from the same component. As in the coding of the DC coefficients, the differences are classified into 5 categories: zero(0), small positive (+S), small negative (–S), large positive (+L), and large negative (–L). The two independent difference categories combine to give 25 different conditioning states. Figure H.2 shows the two-dimensional array of conditioning indices. For each of the 25 conditioning states probability estimates for four binary decisions are kept.

At the beginning of the scan and each restart interval the conditioning derived from the line above is set to zero for the first line of each component. At the start of each line, the difference to the left is set to zero for the purposes of calculating the conditioning.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Difference to left (position a) \ Difference above (position b)	0	+S	–S	+L	–L
0	0	4	8	12	16
+S	20	24	28	32	36
–S	40	44	48	52	56
+L	60	64	68	72	76
–L	80	84	88	92	96

TISO1690-93/d107

Figure H.2 – 5 × 5 Conditioning array for two-dimensional statistical model

H.1.2.3.2 Assignment of statistical bins to the DC binary decision tree

Each statistics area for lossless coding consists of a contiguous set of 158 statistics bins. The first 100 bins consist of 25 sets of four bins selected by a context-index S0. The value of S0 is given by L_Context(Da,Db), which provides a value of 0, 4,..., 92 or 96, depending on the difference classifications of Da and Db (see H.1.2.3.1). The value for S0 provided by L_Context(Da,Db) is from the array in Figure H.2.

The remaining 58 bins consist of two sets of 29 bins, X1, ..., X15, M2, ..., M15, which are used to code magnitude category decisions and magnitude bits. The value of X1 is given by X1_Context(Db), which provides a value of 100 when Db is in the zero, small positive or small negative categories and a value of 129 when Db is in the large positive or large negative categories.

The assignment of statistical bins to the binary decision tree used for coding the difference is given in Table H.3.

Table H.3 – Statistical model for lossless coding

Context-index	Value	Coding decision
S0	L_Context(Da,Db)	V = 0
SS	S0 + 1	Sign
SP	S0 + 2	Sz < 1 if V > 0
SN	S0 + 3	Sz < 1 if V < 0
X1	X1_Context(Db)	Sz < 2
X2	X1 + 1	Sz < 4
X3	X1 + 2	Sz < 8
.	.	.
.	.	.
X15	X1 + 14	$Sz < 2^{15}$
M2	X2 + 14	Magnitude bits if Sz < 4
M3	X3 + 14	Magnitude bits if Sz < 8
.	.	.
.	.	.
M15	X15 + 14	Magnitude bits if $Sz < 2^{15}$

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

H.1.2.3.3 Default conditioning bounds

The bounds, L and U, for determining the conditioning category have the default values L = 0 and U = 1. Other bounds may be set using the DAC (Define-Arithmetic-Conditioning) marker segment, as described in Annex B.

H.1.2.3.4 Initial conditions for statistical model

At the start of a scan and at each restart, all statistics bins are re-initialized to the standard default value described in Annex D.

H.2 Lossless decoder processes

Lossless decoders may employ either Huffman decoding or arithmetic decoding. They shall be capable of using up to four tables in a scan. Lossless decoders shall be able to decode encoded image source data with any input precision from 2 to 16 bits per sample.

H.2.1 Lossless decoder control procedures

Subclause E.2 contains the decoder control procedures. In applying these procedures to the lossless decoder the data unit is one sample.

When the decoder is reset in the restart interval control procedure (see E.2.4) the prediction is reset to the same value used in the encoder (see H.1.2.1). If arithmetic coding is used, the statistics are also reset.

Restrictions on the restart interval are specified in H.1.1.

H.2.2 Coding model for lossless decoding

The predictor calculations defined in H.1.2 also apply to the lossless decoder processes.

The lossless decoders, decode the differences and add them, modulo 2^{16}, to the predictions to create the output. The lossless decoders shall be able to interpret the point transform parameter, and if non-zero, multiply the output of the lossless decoder by 2^{Pt}.

In order to avoid repetition, detailed flow charts of the lossless decoding procedures are omitted.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex J

Hierarchical mode of operation

(This annex forms an integral part of this Recommendation | International Standard)

This annex provides a **functional specification** of the coding processes for the hierarchical mode of operation.

In the hierarchical mode of operation each component is encoded or decoded in a non-differential frame. Such frames may be followed by a sequence of differential frames. A non-differential frame shall be encoded or decoded using the procedures defined in Annexes F, G and H. Differential frame procedures are defined in this annex.

The coding process for a hierarchical encoding containing DCT-based processes is defined as the highest numbered process listed in Table J.1 which is used to code any non-differential DCT-based or differential DCT-based frame in the compressed image data format. The coding process for a hierarchical encoding containing only lossless processes is defined to be the process used for the non-differential frames.

Table J.1 – Coding processes for hierarchical mode

Process	Non-differential frame specification	
1	Extended sequential DCT, Huffman, 8-bit	Annex F, process 2
2	Extended sequential DCT, arithmetic, 8-bit	Annex F, process 3
3	Extended sequential DCT, Huffman, 12-bit	Annex F, process 4
4	Extended sequential DCT, arithmetic, 12-bit	Annex F, process 5
5	Spectral selection only, Huffman, 8-bit	Annex G, process 1
6	Spectral selection only, arithmetic, 8-bit	Annex G, process 2
7	Full progression, Huffman, 8-bit	Annex G, process 3
8	Full progression, arithmetic, 8-bit	Annex G, process 4
9	Spectral selection only, Huffman, 12-bit	Annex G, process 5
10	Spectral selection only, arithmetic, 12-bit	Annex G, process 6
11	Full progression, Huffman, 12-bit	Annex G, process 7
12	Full progression, arithmetic, 12-bit	Annex G, process 8
13	Lossless, Huffman, 2 through 16 bits	Annex H, process 1
14	Lossless, arithmetic, 2 through 16 bits	Annex H, process 2

Hierarchical mode syntax requires a DHP marker segment that appears before the non-differential frame or frames. It may include EXP marker segments and differential frames which shall follow the initial non-differential frame. The frame structure in hierarchical mode is identical to the frame structure in non-hierarchical mode.

Either all non-differential frames within an image shall be coded with DCT-based processes, or all non-differential frames shall be coded with lossless processes. All frames within an image must use the same entropy coding procedure, either Huffman or arithmetic, with the exception that non-differential frames coded with the baseline process may occur in the same image with frames coded with arithmetic coding processes.

If the non-differential frames use DCT-based processes, all differential frames except the final frame for a component shall use DCT-based processes. The final differential frame for each component may use a differential lossless process.

If the non-differential frames use lossless processes, all differential frames shall use differential lossless processes.

For each of the processes listed in Table J.1, the encoding processes are specified in J.1, and decoding processes are specified in J.2.

NOTE – There is **no requirement** in this Specification that any encoder or decoder which embodies one of the above-named processes shall implement the procedures in precisely the manner specified by the flow charts in this annex. It is necessary only that an encoder or decoder implement the **function** specified in this annex. The sole criterion for an encoder or decoder to be considered in compliance with this Specification is that it satisfy the requirements given in clause 6 (for encoders) or clause 7 (for decoders), as determined by the compliance tests specified in Part 2.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

In the hierarchical mode of operation each component is encoded or decoded in a non-differential frame followed by a sequence of differential frames. A non-differential frame shall use the procedures defined in Annexes F, G, and H. Differential frame procedures are defined in this annex.

J.1 Hierarchical encoding

J.1.1 Hierarchical control procedure for encoding an image

The control structure for encoding of an image using the hierarchical mode is given in Figure J.1.

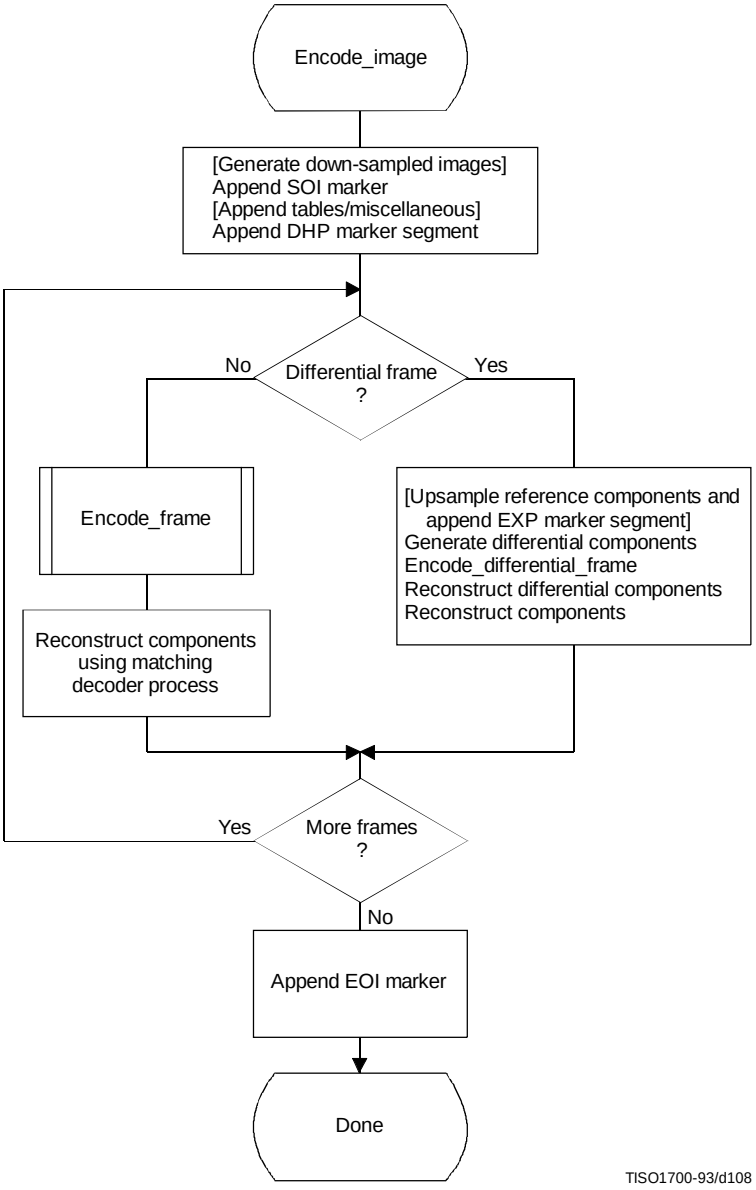


Figure J.1 – Hierarchical control procedure for encoding an image

In Figure J.1 procedures in brackets shall be performed whenever the particular hierarchical encoding sequence being followed requires them.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

In the hierarchical mode the define-hierarchical-progression (DHP) marker segment shall be placed in the compressed image data before the first start-of-frame. The DHP segment is used to signal the size of the image components of the completed image. The syntax of the DHP segment is specified in Annex B.

The first frame for each component or group of components in a hierarchical process shall be encoded by a non-differential frame. Differential frames shall then be used to encode the two's complement differences between source input components (possibly downsampled) and the reference components (possibly upsampled). The reference components are reconstructed components created by previous frames in the hierarchical process. For either differential or non-differential frames, reconstructions of the components shall be generated if needed as reference components for a subsequent frame in the hierarchical process.

Resolution changes may occur between hierarchical frames in a hierarchical process. These changes occur if downsampling filters are used to reduce the spatial resolution of some or all of the components of the source image. When the resolution of a reference component does not match the resolution of the component input to a differential frame, an upsampling filter shall be used to increase the spatial resolution of the reference component. The EXP marker segment shall be added to the compressed image data before the start-of-frame whenever upsampling of a reference component is required. No more than one EXP marker segment shall precede a given frame.

Any of the marker segments allowed before a start-of-frame for the encoding process selected may be used before either non-differential or differential frames.

For 16-bit input precision (lossless encoder), the differential components which are input to a differential frame are calculated modulo 2^{16}. The reconstructed components calculated from the reconstructed differential components are also calculated modulo 2^{16}.

If a hierarchical encoding process uses a DCT encoding process for the first frame, all frames in the hierarchical process except for the final frame for each component shall use the DCT encoding processes defined in either Annex F or Annex G, or the modified DCT encoding processes defined in this annex. The final frame may use a modified lossless process defined in this annex.

If a hierarchical encoding process uses a lossless encoding process for the first frame, all frames in the hierarchical process shall use a lossless encoding process defined in Annex H, or a modified lossless process defined in this annex.

J.1.1.1 Downsampling filter

The downsampled components are generated using a downsampling filter that is not specified in this Specification. This filter should, however, be consistent with the upsampling filter. An example of a downsampling filter is provided in K.5.

J.1.1.2 Upsampling filter

The upsampling filter increases the spatial resolution by a factor of two horizontally, vertically, or both. Bi-linear interpolation is used for the upsampling filter, as illustrated in Figure J.2.

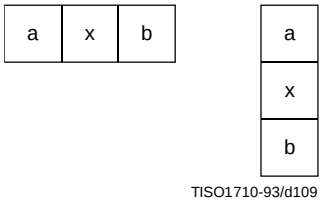


Figure J.2 – Diagram of sample positions for upsampling rules

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The rule for calculating the interpolated value is:

$$P_x = (Ra + Rb) / 2$$

where Ra and Rb are sample values from adjacent positions a and b of the lower resolution image and Px is the interpolated value. The division indicates truncation, not rounding. The left-most column of the upsampled image matches the left-most column of the lower resolution image. The top line of the upsampled image matches the top line of the lower resolution image. The right column and the bottom line of the lower resolution image are replicated to provide the values required for the right column edge and bottom line interpolations. The upsampling process always doubles the line length or the number of lines.

If both horizontal and vertical expansions are signalled, they are done in sequence – first the horizontal expansion and then the vertical.

J.1.2 Control procedure for encoding a differential frame

The control procedures in Annex E for frames, scans, restart intervals, and MCU also apply to the encoding of differential frames, and the scans, restart intervals, and MCU from which the differential frame is constructed. The differential frames differ from the frames of Annexes F, G, and H only at the coding model level.

J.1.3 Encoder coding models for differential frames

The coding models defined in Annexes F, G, and H are modified to allow them to be used for coding of two's complement differences.

J.1.3.1 Modifications to encoder DCT encoding models for differential frames

Two modifications are made to the DCT coding models to allow them to be used in differential frames. First, the FDCT of the differential input is calculated without the level shift. Second, the DC coefficient of the DCT is coded directly – without prediction.

J.1.3.2 Modifications to lossless encoding models for differential frames

One modification is made to the lossless coding models. The difference is coded directly – without prediction. The prediction selection parameter in the scan header shall be set to zero. The point transform which may be applied to the differential inputs is defined in Annex A.

J.1.4 Modifications to the entropy encoders for differential frames

The coding of two's complement differences requires one extra bit of precision for the Huffman coding of AC coefficients. The extension to Tables F.1 and F.7 is given in Table J.2.

Table J.2 – Modifications to table of AC coefficient amplitude ranges

SSSS	AC coefficients
15	–32 767..–16 384, 16 384..32 767

The arithmetic coding models are already defined for the precision needed in differential frames.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

J.2 Hierarchical decoding

J.2.1 Hierarchical control procedure for decoding an image

The control structure for decoding an image using the hierarchical mode is given in Figure J.3.

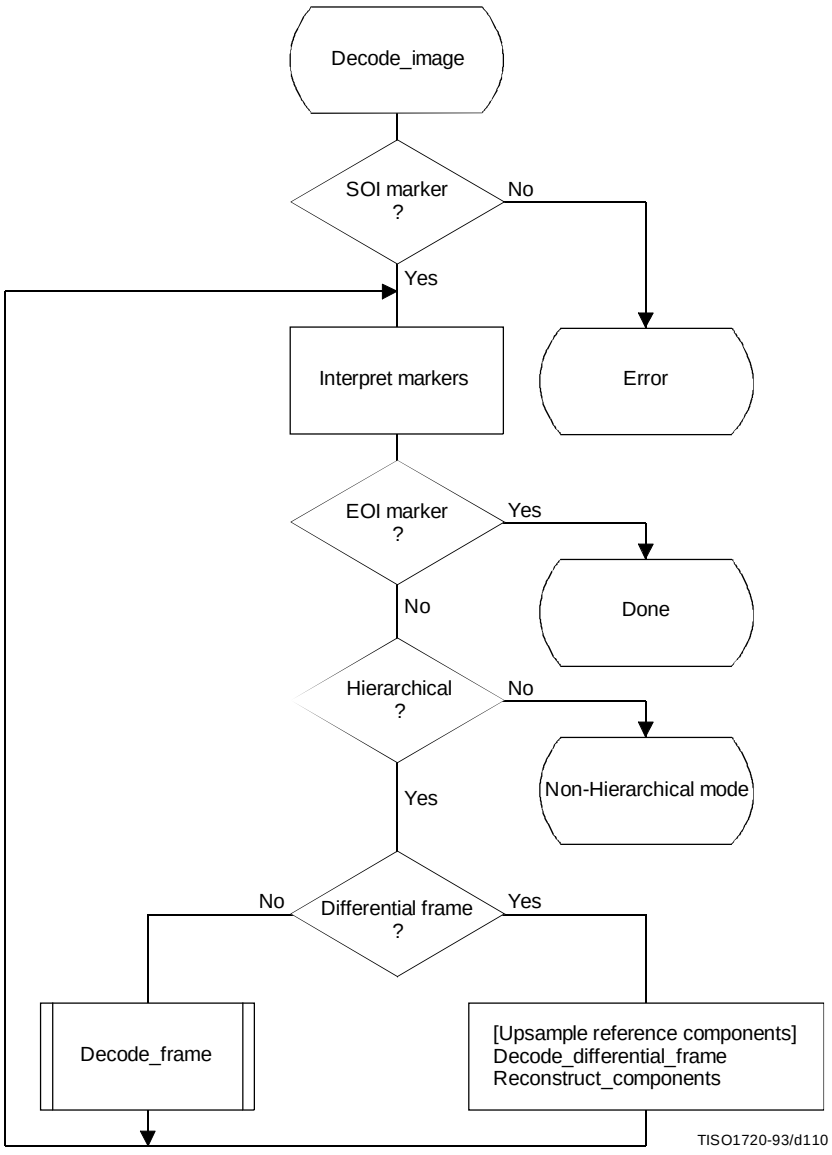


Figure J.3 – Hierarchical control procedure for decoding an image

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The Interpret markers procedure shall decode the markers which may precede the SOF marker, continuing this decoding until either a SOF or EOI marker is found. If the DHP marker is encountered before the first frame, a flag is set which selects the hierarchical decoder at the “hierarchical?” decision point. In addition to the DHP marker (which shall precede any SOF) and the EXP marker (which shall precede any differential SOF requiring resolution changes in the reference components), any other markers which may precede a SOF shall be interpreted to the extent required for decoding of the compressed image data.

If a differential SOF marker is found, the differential frame path is followed. If the EXP was encountered in the Interpret markers procedure, the reference components for the frame shall be upsampled as required by the parameters in the EXP segment. The upsampling procedure described in J.1.1.2 shall be followed.

The Decode_differential_frame procedure generates a set of differential components. These differential components shall be added, modulo 2^{16}, to the upsampled reference components in the Reconstruct_components procedure. This creates a new set of reference components which shall be used when required in subsequent frames of the hierarchical process.

J.2.2 Control procedure for decoding a differential frame

The control procedures in Annex E for frames, scans, restart intervals, and MCU also apply to the decoding of differential frames and the scans, restart intervals, and MCU from which the differential frame is constructed. The differential frame differs from the frames of Annexes F, G, and H only at the decoder coding model level.

J.2.3 Decoder coding models for differential frames

The decoding models described in Annexes F, G, and H are modified to allow them to be used for decoding of two’s complement differential components.

J.2.3.1 Modifications to the differential frame decoder DCT coding model

Two modifications are made to the decoder DCT coding models to allow them to code differential frames. First, the IDCT of the differential output is calculated without the level shift. Second, the DC coefficient of the DCT is decoded directly – without prediction.

J.2.3.2 Modifications to the differential frame decoder lossless coding model

One modification is made to the lossless decoder coding model. The difference is decoded directly – without prediction. If the point transformation parameter in the scan header is not zero, the point transform, defined in Annex A, shall be applied to the differential output.

J.2.4 Modifications to the entropy decoders for differential frames

The decoding of two’s complement differences requires one extra bit of precision in the Huffman code table. This is described in J.1.4. The arithmetic coding models are already defined for the precision needed in differential frames.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex K

Examples and guidelines

(This annex does not form an integral part of this Recommendation | International Standard)

This annex provides examples of various tables, procedures, and other guidelines.

K.1 Quantization tables for luminance and chrominance components

Two examples of quantization tables are given in Tables K.1 and K.2. These are based on psychovisual thresholding and are derived empirically using luminance and chrominance and 2:1 horizontal subsampling. These tables are provided as examples only and are not necessarily suitable for any particular application. These quantization values have been used with good results on 8-bit per sample luminance and chrominance images of the format illustrated in Figure 13. Note that these quantization values are appropriate for the DCT normalization defined in A.3.3.

If these quantization values are divided by 2, the resulting reconstructed image is usually nearly indistinguishable from the source image.

Table K.1 – Luminance quantization table

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table K.2 – Chrominance quantization table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

K.2 A procedure for generating the lists which specify a Huffman code table

A Huffman table is generated from a collection of statistics in two steps. The first step is the generation of the list of lengths and values which are in accord with the rules for generating the Huffman code tables. The second step is the generation of the Huffman code table from the list of lengths and values.

The first step, the topic of this section, is needed only for custom Huffman table generation and is done only in the encoder. In this step the statistics are used to create a table associating each value to be coded with the size (in bits) of the corresponding Huffman code. This table is sorted by code size.

A procedure for creating a Huffman table for a set of up to 256 symbols is shown in Figure K.1. Three vectors are defined for this procedure:

FREQ(V)	Frequency of occurrence of symbol V
CODESIZE(V)	Code size of symbol V
OTHERS(V)	Index to next symbol in chain of all symbols in current branch of code tree

where V goes from 0 to 256.

Before starting the procedure, the values of FREQ are collected for V = 0 to 255 and the FREQ value for V = 256 is set to 1 to reserve one code point. FREQ values for unused symbols are defined to be zero. In addition, the entries in CODESIZE are all set to 0, and the indices in OTHERS are set to –1, the value which terminates a chain of indices. Reserving one code point guarantees that no code word can ever be all “1” bits.

The search for the entry with the least value of FREQ(V) selects the largest value of V with the least value of FREQ(V) greater than zero.

The procedure “Find V1 for least value of FREQ(V1) > 0” always selects the value with the largest value of V1 when more than one V1 with the same frequency occurs. The reserved code point is then guaranteed to be in the longest code word category.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

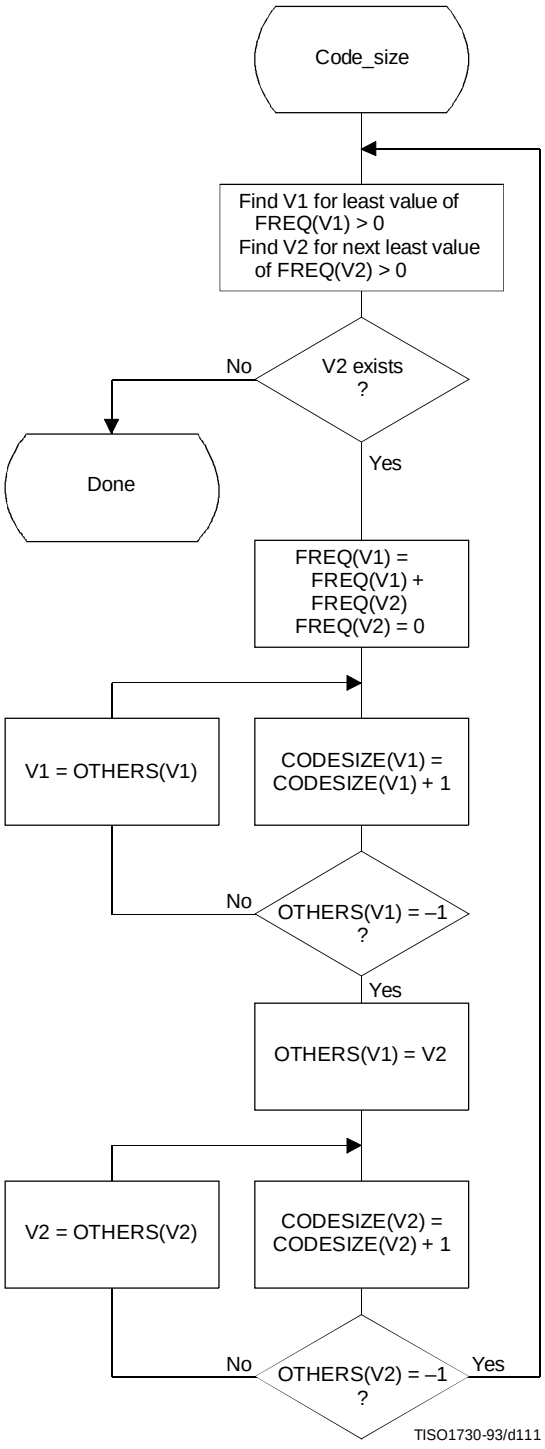


Figure K.1 – Procedure to find Huffman code sizes

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Once the code lengths for each symbol have been obtained, the number of codes of each length is obtained using the procedure in Figure K.2. The count for each size is contained in the list, BITS. The counts in BITS are zero at the start of the procedure. The procedure assumes that the probabilities are large enough that code lengths greater than 32 bits never occur. Note that until the final Adjust_BITS procedure is complete, BITS may have more than the 16 entries required in the table specification (see Annex C).

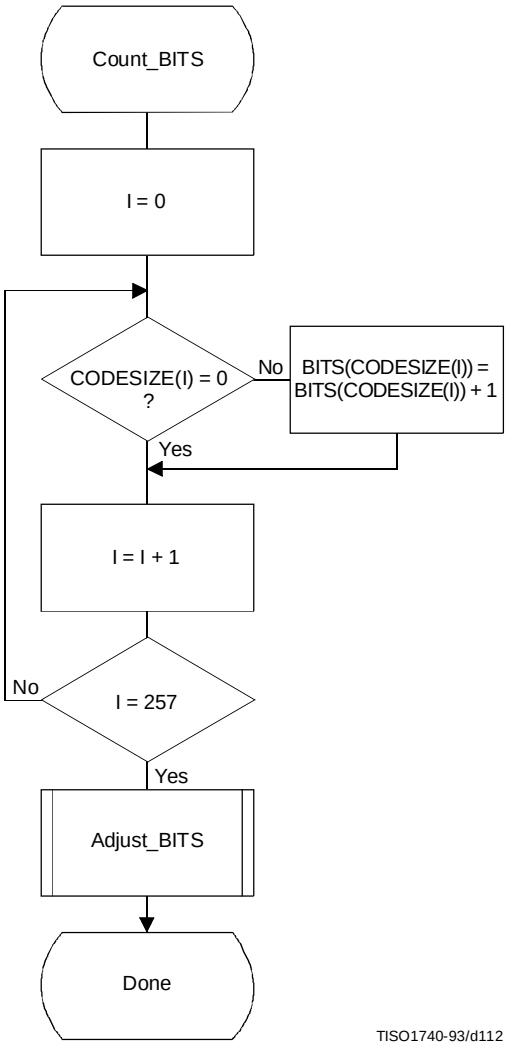


Figure K.2 – Procedure to find the number of codes of each size

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Figure K.3 gives the procedure for adjusting the BITS list so that no code is longer than 16 bits. Since symbols are paired for the longest Huffman code, the symbols are removed from this length category two at a time. The prefix for the pair (which is one bit shorter) is allocated to one of the pair; then (skipping the BITS entry for that prefix length) a code word from the next shortest non-zero BITS entry is converted into a prefix for two code words one bit longer. After the BITS list is reduced to a maximum code length of 16 bits, the last step removes the reserved code point from the code length count.

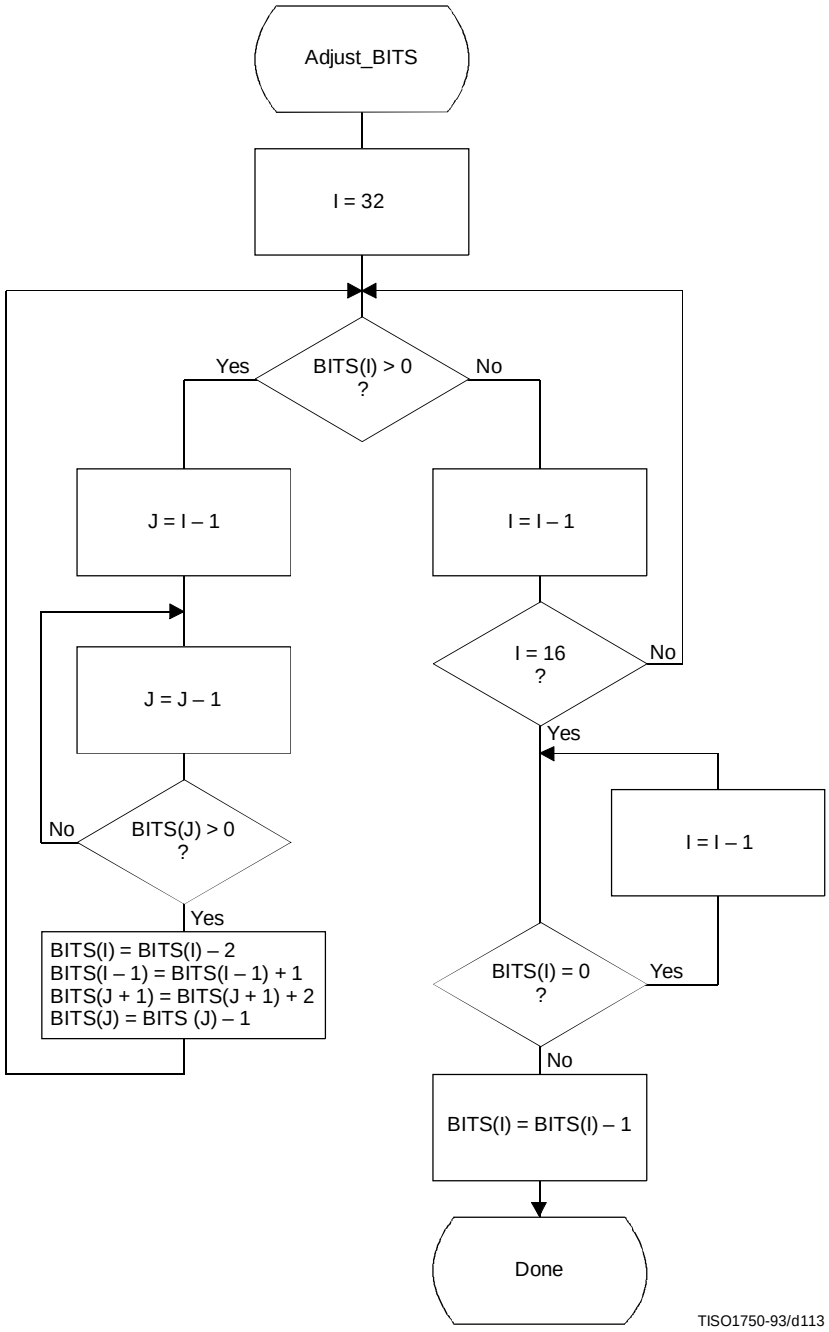


Figure K.3 – Procedure for limiting code lengths to 16 bits

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The input values are sorted according to code size as shown in Figure K.4. HUFFVAL is the list containing the input values associated with each code word, in order of increasing code length.

At this point, the list of code lengths (BITS) and the list of values (HUFFVAL) can be used to generate the code tables. These procedures are described in Annex C.

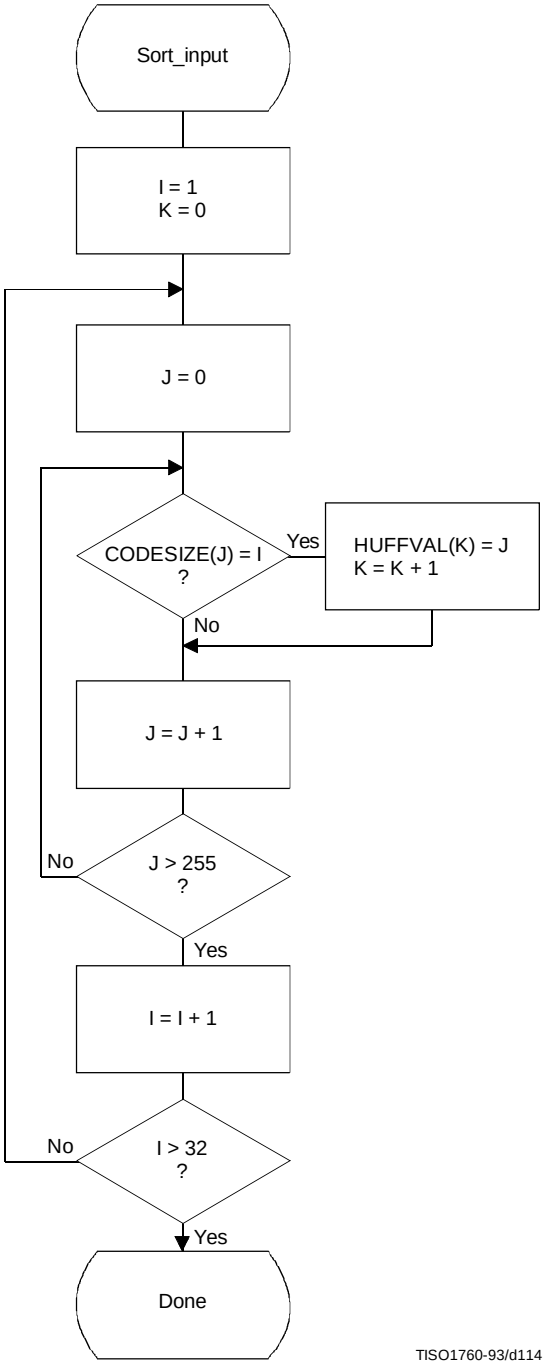


Figure K.4 – Sorting of input values according to code size

K.3 Typical Huffman tables for 8-bit precision luminance and chrominance

Huffman table-specification syntax is specified in B.2.4.2.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

K.3.1 Typical Huffman tables for the DC coefficient differences

Tables K.3 and K.4 give Huffman tables for the DC coefficient differences which have been developed from the average statistics of a large set of video images with 8-bit precision. Table K.3 is appropriate for luminance components and Table K.4 is appropriate for chrominance components. Although there are no default tables, these tables may prove to be useful for many applications.

Table K.3 – Table for luminance DC coefficient differences

Category	Code length	Code word
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

Table K.4 – Table for chrominance DC coefficient differences

Category	Code length	Code word
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

K.3.2 Typical Huffman tables for the AC coefficients

Tables K.5 and K.6 give Huffman tables for the AC coefficients which have been developed from the average statistics of a large set of images with 8-bit precision. Table K.5 is appropriate for luminance components and Table K.6 is appropriate for chrominance components. Although there are no default tables, these tables may prove to be useful for many applications.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.5 – Table for luminance AC coefficients (sheet 1 of 4)

Run/Size	Code length	Code word
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	11111110110
1/6	16	1111111110000100
1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	1111111110001001
2/6	16	1111111110001010
2/7	16	1111111110001011
2/8	16	1111111110001100
2/9	16	1111111110001101
2/A	16	1111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	1111111110001111
3/5	16	1111111110010000
3/6	16	1111111110010001
3/7	16	1111111110010010
3/8	16	1111111110010011
3/9	16	1111111110010100
3/A	16	1111111110010101

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.5 (sheet 2 of 4)

Run/Size	Code length	Code word
4/1	6	111011
4/2	10	1111111000
4/3	16	1111111110010110
4/4	16	1111111110010111
4/5	16	1111111110011000
4/6	16	1111111110011001
4/7	16	1111111110011010
4/8	16	1111111110011011
4/9	16	1111111110011100
4/A	16	1111111110011101
5/1	7	1111010
5/2	11	11111110111
5/3	16	1111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001
5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101
6/1	7	1111011
6/2	12	111111110110
6/3	16	1111111110100110
6/4	16	1111111110100111
6/5	16	1111111110101000
6/6	16	1111111110101001
6/7	16	1111111110101010
6/8	16	1111111110101011
6/9	16	1111111110101100
6/A	16	1111111110101101
7/1	8	11111010
7/2	12	111111110111
7/3	16	1111111110101110
7/4	16	1111111110101111
7/5	16	1111111110110000
7/6	16	1111111110110001
7/7	16	1111111110110010
7/8	16	1111111110110011
7/9	16	1111111110110100
7/A	16	1111111110110101
8/1	9	111111000
8/2	15	111111111000000

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.5 (sheet 3 of 4)

Run/Size	Code length	Code word
8/3	16	1111111110110110
8/4	16	1111111110110111
8/5	16	1111111110111000
8/6	16	1111111110111001
8/7	16	1111111110111010
8/8	16	1111111110111011
8/9	16	1111111110111100
8/A	16	1111111110111101
9/1	9	111111001
9/2	16	1111111110111110
9/3	16	1111111110111111
9/4	16	1111111111000000
9/5	16	1111111111000001
9/6	16	1111111111000010
9/7	16	1111111111000011
9/8	16	1111111111000100
9/9	16	1111111111000101
9/A	16	1111111111000110
A/1	9	111111010
A/2	16	1111111111000111
A/3	16	1111111111001000
A/4	16	1111111111001001
A/5	16	1111111111001010
A/6	16	1111111111001011
A/7	16	1111111111001100
A/8	16	1111111111001101
A/9	16	1111111111001110
A/A	16	1111111111001111
B/1	10	1111111001
B/2	16	1111111111010000
B/3	16	1111111111010001
B/4	16	1111111111010010
B/5	16	1111111111010011
B/6	16	1111111111010100
B/7	16	1111111111010101
B/8	16	1111111111010110
B/9	16	1111111111010111
B/A	16	1111111111011000
C/1	10	1111111010
C/2	16	1111111111011001
C/3	16	1111111111011010
C/4	16	1111111111011011

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.5 (sheet 4 of 4)

Run/Size	Code length	Code word
C/5	16	1111111111011100
C/6	16	1111111111011101
C/7	16	1111111111011110
C/8	16	1111111111011111
C/9	16	1111111111100000
C/A	16	1111111111100001
D/1	11	11111111000
D/2	16	1111111111100010
D/3	16	1111111111100011
D/4	16	1111111111100100
D/5	16	1111111111100101
D/6	16	1111111111100110
D/7	16	1111111111100111
D/8	16	1111111111101000
D/9	16	1111111111101001
D/A	16	1111111111101010
E/1	16	1111111111101011
E/2	16	1111111111101100
E/3	16	1111111111101101
E/4	16	1111111111101110
E/5	16	1111111111101111
E/6	16	1111111111110000
E/7	16	1111111111110001
E/8	16	1111111111110010
E/9	16	1111111111110011
E/A	16	1111111111110100
F/0 (ZRL)	11	11111111001
F/1	16	1111111111110101
F/2	16	1111111111110110
F/3	16	1111111111110111
F/4	16	1111111111111000
F/5	16	1111111111111001
F/6	16	1111111111111010
F/7	16	1111111111111011
F/8	16	1111111111111100
F/9	16	1111111111111101
F/A	16	1111111111111110

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.6 – Table for chrominance AC coefficients (sheet 1 of 4)

Run/Size	Code length	Code word
0/0 (EOB)	2	00
0/1	2	01
0/2	3	100
0/3	4	1010
0/4	5	11000
0/5	5	11001
0/6	6	111000
0/7	7	1111000
0/8	9	111110100
0/9	10	1111110110
0/A	12	111111110100
1/1	4	1011
1/2	6	111001
1/3	8	11110110
1/4	9	111110101
1/5	11	11111110110
1/6	12	111111110101
1/7	16	1111111110001000
1/8	16	1111111110001001
1/9	16	1111111110001010
1/A	16	1111111110001011
2/1	5	11010
2/2	8	11110111
2/3	10	1111110111
2/4	12	111111110110
2/5	15	111111111000010
2/6	16	1111111110001100
2/7	16	1111111110001101
2/8	16	1111111110001110
2/9	16	1111111110001111
2/A	16	1111111110010000
3/1	5	11011
3/2	8	11111000
3/3	10	1111111000
3/4	12	111111110111
3/5	16	1111111110010001
3/6	16	1111111110010010
3/7	16	1111111110010011
3/8	16	1111111110010100
3/9	16	1111111110010101
3/A	16	1111111110010110
4/1	6	111010

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.6 (sheet 2 of 4)

Run/Size	Code length	Code word
4/2	9	111110110
4/3	16	1111111110010111
4/4	16	1111111110011000
4/5	16	1111111110011001
4/6	16	1111111110011010
4/7	16	1111111110011011
4/8	16	1111111110011100
4/9	16	1111111110011101
4/A	16	1111111110011110
5/1	6	111011
5/2	10	1111111001
5/3	16	1111111110011111
5/4	16	1111111110100000
5/5	16	1111111110100001
5/6	16	1111111110100010
5/7	16	1111111110100011
5/8	16	1111111110100100
5/9	16	1111111110100101
5/A	16	1111111110100110
6/1	7	1111001
6/2	11	11111110111
6/3	16	1111111110100111
6/4	16	1111111110101000
6/5	16	1111111110101001
6/6	16	1111111110101010
6/7	16	1111111110101011
6/8	16	1111111110101100
6/9	16	1111111110101101
6/A	16	1111111110101110
7/1	7	1111010
7/2	11	11111111000
7/3	16	1111111110101111
7/4	16	1111111110110000
7/5	16	1111111110110001
7/6	16	1111111110110010
7/7	16	1111111110110011
7/8	16	1111111110110100
7/9	16	1111111110110101
7/A	16	1111111110110110
8/1	8	11111001
8/2	16	1111111110110111
8/3	16	1111111110111000

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.6 (sheet 3 of 4)

Run/Size	Code length	Code word
8/4	16	1111111110111001
8/5	16	1111111110111010
8/6	16	1111111110111011
8/7	16	1111111110111100
8/8	16	1111111110111101
8/9	16	1111111110111110
8/A	16	1111111110111111
9/1	9	111110111
9/2	16	1111111111000000
9/3	16	1111111111000001
9/4	16	1111111111000010
9/5	16	1111111111000011
9/6	16	1111111111000100
9/7	16	1111111111000101
9/8	16	1111111111000110
9/9	16	1111111111000111
9/A	16	1111111111001000
A/1	9	111111000
A/2	16	1111111111001001
A/3	16	1111111111001010
A/4	16	1111111111001011
A/5	16	1111111111001100
A/6	16	1111111111001101
A/7	16	1111111111001110
A/8	16	1111111111001111
A/9	16	1111111111010000
A/A	16	1111111111010001
B/1	9	111111001
B/2	16	1111111111010010
B/3	16	1111111111010011
B/4	16	1111111111010100
B/5	16	1111111111010101
B/6	16	1111111111010110
B/7	16	1111111111010111
B/8	16	1111111111011000
B/9	16	1111111111011001
B/A	16	1111111111011010
C/1	9	111111010
C/2	16	1111111111011011
C/3	16	1111111111011100
C/4	16	1111111111011101
C/5	16	1111111111011110

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.6 (sheet 4 of 4)

Run/Size	Code length	Code word
C/6	16	1111111111011111
C/7	16	1111111111100000
C/8	16	1111111111100001
C/9	16	1111111111100010
C/A	16	1111111111100011
D/1	11	11111111001
D/2	16	1111111111100100
D/3	16	1111111111100101
D/4	16	1111111111100110
D/5	16	1111111111100111
D/6	16	1111111111101000
D/7	16	1111111111101001
D/8	16	1111111111101010
D/9	16	1111111111101011
D/A	16	1111111111101100
E/1	14	11111111100000
E/2	16	1111111111101101
E/3	16	1111111111101110
E/4	16	1111111111101111
E/5	16	1111111111110000
E/6	16	1111111111110001
E/7	16	1111111111110010
E/8	16	1111111111110011
E/9	16	1111111111110100
E/A	16	1111111111110101
F/0 (ZRL)	10	1111111010
F/1	15	111111111000011
F/2	16	1111111111110110
F/3	16	1111111111110111
F/4	16	1111111111111000
F/5	16	1111111111111001
F/6	16	1111111111111010
F/7	16	1111111111111011
F/8	16	1111111111111100
F/9	16	1111111111111101
F/A	16	1111111111111110

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

K.3.3 Huffman table-specification examples

K.3.3.1 Specification of typical tables for DC difference coding

A set of typical tables for DC component coding is given in K.3.1. The specification of these tables is as follows:

For Table K.3 (for luminance DC coefficients), the 16 bytes which specify the list of code lengths for the table are

X'00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00'

The set of values following this list is

X'00 01 02 03 04 05 06 07 08 09 0A 0B'

For Table K.4 (for chrominance DC coefficients), the 16 bytes which specify the list of code lengths for the table are

X'00 03 01 01 01 01 01 01 01 01 01 00 00 00 00 00'

The set of values following this list is

X'00 01 02 03 04 05 06 07 08 09 0A 0B'

K.3.3.2 Specification of typical tables for AC coefficient coding

A set of typical tables for AC component coding is given in K.3.2. The specification of these tables is as follows:

For Table K.5 (for luminance AC coefficients), the 16 bytes which specify the list of code lengths for the table are

X'00 02 01 03 03 02 04 03 05 05 04 04 00 00 01 7D'

The set of values which follows this list is

X'01 02 03 00 04 11 05 12 21 31 41 06 13 51 61 07
22 71 14 32 81 91 A1 08 23 42 B1 C1 15 52 D1 F0
24 33 62 72 82 09 0A 16 17 18 19 1A 25 26 27 28
29 2A 34 35 36 37 38 39 3A 43 44 45 46 47 48 49
4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69
6A 73 74 75 76 77 78 79 7A 83 84 85 86 87 88 89
8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7
A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5
C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E1 E2
E3 E4 E5 E6 E7 E8 E9 EA F1 F2 F3 F4 F5 F6 F7 F8
F9 FA'

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

For Table K.6 (for chrominance AC coefficients), the 16 bytes which specify the list of code lengths for the table are

X'00 02 01 02 04 04 03 04 07 05 04 04 00 01 02 77'

The set of values which follows this list is:

X'00 01 02 03 11 04 05 21 31 06 12 41 51 07 61 71
13 22 32 81 08 14 42 91 A1 B1 C1 09 23 33 52 F0
15 62 72 D1 0A 16 24 34 E1 25 F1 17 18 19 1A 26
27 28 29 2A 35 36 37 38 39 3A 43 44 45 46 47 48
49 4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68
69 6A 73 74 75 76 77 78 79 7A 82 83 84 85 86 87
88 89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5
A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3
C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA
E2 E3 E4 E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8
F9 FA'

K.4 Additional information on arithmetic coding

K.4.1 Test sequence for a small data set for the arithmetic coder

The following 256-bit test sequence (in hexadecimal form) is structured to test many of the encoder and decoder paths:

X'00020051 000000C0 0352872A AAAAAAAA 82C02000 FCD79EF6 74EAABF7 697EE74C'

Tables K.7 and K.8 provide a symbol-by-symbol list of the arithmetic encoder and decoder operation. In these tables the event count, EC, is listed first, followed by the value of Qe used in encoding and decoding that event. The decision D to be encoded (and decoded) is listed next. The column labeled MPS contains the sense of the MPS, and if it is followed by a CE (in the "CX" column), the conditional MPS/LPS exchange occurs when encoding and decoding the decision (see Figures D.3, D.4 and D.17). The contents of the A and C registers are the values before the event is encoded and decoded. ST is the number of X'FF' bytes stacked in the encoder waiting for a resolution of the carry-over. Note that the A register is always greater than X'7FFF'. (The starting value has an implied value of X'10000'.)

In the encoder test, the code bytes (B) are listed if they were completed during the coding of the preceding event. If additional bytes follow, they were also completed during the coding of the preceding event. If a byte is listed in the Bx column, the preceding byte in column B was modified by a carry-over.

In the decoder the code bytes are listed if they were placed in the code register just prior to the event EC.

For this file the coded bit count is 240, including the overhead to flush the final data from the C register. When the marker X'FFD9' is appended, a total of 256 bits are output. The actual compressed data sequence for the encoder is (in hexadecimal form)

X'655B5144 F7969D51 7855BFFF 00FC5184 C7CEF939 00287D46 708ECBC0 F6FFD900'

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.7 – Encoder test sequence (sheet 1 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	ST	Bx	B
1	0	0		5A1D	0000	00000000	11	0		
2	0	0	CE	5A1D	A5E3	00000000	11	0		
3	0	0		2586	B43A	0000978C	10	0		
4	0	0		2586	8EB4	0000978C	10	0		
5	0	0		1114	D25C	00012F18	9	0		
6	0	0		1114	C148	00012F18	9	0		
7	0	0		1114	B034	00012F18	9	0		
8	0	0		1114	9F20	00012F18	9	0		
9	0	0		1114	8E0C	00012F18	9	0		
10	0	0		080B	F9F0	00025E30	8	0		
11	0	0		080B	F1E5	00025E30	8	0		
12	0	0		080B	E9DA	00025E30	8	0		
13	0	0		080B	E1CF	00025E30	8	0		
14	0	0		080B	D9C4	00025E30	8	0		
15	1	0		080B	D1B9	00025E30	8	0		
16	0	0		17B9	80B0	00327DE0	4	0		
17	0	0		1182	D1EE	0064FBC0	3	0		
18	0	0		1182	C06C	0064FBC0	3	0		
19	0	0		1182	AEEA	0064FBC0	3	0		
20	0	0		1182	9D68	0064FBC0	3	0		
21	0	0		1182	8BE6	0064FBC0	3	0		
22	0	0		0CEF	F4C8	00C9F780	2	0		
23	0	0		0CEF	E7D9	00C9F780	2	0		
24	0	0		0CEF	DAEA	00C9F780	2	0		
25	0	0		0CEF	CDFB	00C9F780	2	0		
26	1	0		0CEF	C10C	00C9F780	2	0		
27	0	0		1518	CEF0	000AB9D0	6	0		65
28	1	0		1518	B9D8	000AB9D0	6	0		
29	0	0		1AA9	A8C0	005AF480	3	0		
30	0	0		1AA9	8E17	005AF480	3	0		
31	0	0		174E	E6DC	00B5E900	2	0		
32	1	0		174E	CF8E	00B5E900	2	0		
33	0	0		1AA9	BA70	00050A00	7	0		5B
34	0	0		1AA9	9FC7	00050A00	7	0		
35	0	0		1AA9	851E	00050A00	7	0		
36	0	0		174E	D4EA	000A1400	6	0		

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.7 – Encoder test sequence (sheet 2 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	ST	Bx	B
37	0	0		174E	BD9C	000A1400	6	0		
38	0	0		174E	A64E	000A1400	6	0		
39	0	0		174E	8F00	000A1400	6	0		
40	0	0		1424	EF64	00142800	5	0		
41	0	0		1424	DB40	00142800	5	0		
42	0	0		1424	C71C	00142800	5	0		
43	0	0		1424	B2F8	00142800	5	0		
44	0	0		1424	9ED4	00142800	5	0		
45	0	0		1424	8AB0	00142800	5	0		
46	0	0		119C	ED18	00285000	4	0		
47	0	0		119C	DB7C	00285000	4	0		
48	0	0		119C	C9E0	00285000	4	0		
49	0	0		119C	B844	00285000	4	0		
50	0	0		119C	A6A8	00285000	4	0		
51	0	0		119C	950C	00285000	4	0		
52	0	0		119C	8370	00285000	4	0		
53	0	0		0F6B	E3A8	0050A000	3	0		
54	0	0		0F6B	D43D	0050A000	3	0		
55	0	0		0F6B	C4D2	0050A000	3	0		
56	0	0		0F6B	B567	0050A000	3	0		
57	1	0		0F6B	A5FC	0050A000	3	0		
58	1	0		1424	F6B0	00036910	7	0		51
59	0	0		1AA9	A120	00225CE0	4	0		
60	0	0		1AA9	8677	00225CE0	4	0		
61	0	0		174E	D79C	0044B9C0	3	0		
62	0	0		174E	C04E	0044B9C0	3	0		
63	0	0		174E	A900	0044B9C0	3	0		
64	0	0		174E	91B2	0044B9C0	3	0		
65	0	0		1424	F4C8	00897380	2	0		
66	0	0		1424	E0A4	00897380	2	0		
67	0	0		1424	CC80	00897380	2	0		
68	0	0		1424	B85C	00897380	2	0		
69	0	0		1424	A438	00897380	2	0		
70	0	0		1424	9014	00897380	2	0		
71	1	0		119C	F7E0	0112E700	1	0		
72	1	0		1424	8CE0	001E6A20	6	0		44
73	0	0		1AA9	A120	00F716E0	3	0		

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.7 – Encoder test sequence (sheet 3 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	ST	Bx	B
74	1	0		1AA9	8677	00F716E0	3	0		
75	0	0		2516	D548	00041570	8	0		F7
76	1	0		2516	B032	00041570	8	0		
77	0	0		299A	9458	00128230	6	0		
78	0	0		2516	D57C	00250460	5	0		
79	1	0		2516	B066	00250460	5	0		
80	0	0		299A	9458	00963EC0	3	0		
81	1	0		2516	D57C	012C7D80	2	0		
82	0	0		299A	9458	0004B798	8	0		96
83	0	0		2516	D57C	00096F30	7	0		
84	0	0		2516	B066	00096F30	7	0		
85	0	0		2516	8B50	00096F30	7	0		
86	1	0		1EDF	CC74	0012DE60	6	0		
87	1	0		2516	F6F8	009C5FA8	3	0		
88	1	0		299A	9458	0274C628	1	0		
89	0	0		32B4	A668	0004C398	7	0		9D
90	0	0		2E17	E768	00098730	6	0		
91	1	0		2E17	B951	00098730	6	0		
92	0	0		32B4	B85C	002849A8	4	0		
93	1	0		32B4	85A8	002849A8	4	0		
94	0	0		3C3D	CAD0	00A27270	2	0		
95	1	0		3C3D	8E93	00A27270	2	0		
96	0	0		415E	F0F4	00031318	8	0		51
97	1	0		415E	AF96	00031318	8	0		
98	0	0	CE	4639	82BC	000702A0	7	0		
99	1	0		415E	8C72	000E7E46	6	0		
100	0	0	CE	4639	82BC	001D92B4	5	0		
101	1	0		415E	8C72	003B9E6E	4	0		
102	0	0	CE	4639	82BC	0077D304	3	0		
103	1	0		415E	8C72	00F01F0E	2	0		
104	0	0	CE	4639	82BC	01E0D444	1	0		
105	1	0		415E	8C72	0002218E	8	0		78
106	0	0	CE	4639	82BC	0004D944	7	0		
107	1	0		415E	8C72	000A2B8E	6	0		
108	0	0	CE	4639	82BC	0014ED44	5	0		
109	1	0		415E	8C72	002A538E	4	0		
110	0	0	CE	4639	82BC	00553D44	3	0		

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.7 – Encoder test sequence (sheet 4 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	ST	Bx	B
111	1	0		415E	8C72	00AAF38E	2	0		
112	0	0	CE	4639	82BC	01567D44	1	0		
113	1	0		415E	8C72	0005738E	8	0		55
114	0	0	CE	4639	82BC	000B7D44	7	0		
115	1	0		415E	8C72	0017738E	6	0		
116	0	0	CE	4639	82BC	002F7D44	5	0		
117	1	0		415E	8C72	005F738E	4	0		
118	0	0	CE	4639	82BC	00BF7D44	3	0		
119	1	0		415E	8C72	017F738E	2	0		
120	0	0	CE	4639	82BC	02FF7D44	1	0		
121	1	0		415E	8C72	0007738E	8	0		BF
122	0	0	CE	4639	82BC	000F7D44	7	0		
123	1	0		415E	8C72	001F738E	6	0		
124	0	0	CE	4639	82BC	003F7D44	5	0		
125	1	0		415E	8C72	007F738E	4	0		
126	0	0	CE	4639	82BC	00FF7D44	3	0		
127	1	0		415E	8C72	01FF738E	2	0		
128	0	0	CE	4639	82BC	03FF7D44	1	0		
129	1	0		415E	8C72	0007738E	8	1		
130	0	0	CE	4639	82BC	000F7D44	7	1		
131	0	0		415E	8C72	001F738E	6	1		
132	0	0		3C3D	9628	003EE71C	5	1		
133	0	0		375E	B3D6	007DCE38	4	1		
134	0	0		32B4	F8F0	00FB9C70	3	1		
135	1	0		32B4	C63C	00FB9C70	3	1		
136	0	0		3C3D	CAD0	03F0BFE0	1	1		
137	1	0		3C3D	8E93	03F0BFE0	1	1		
138	1	0		415E	F0F4	000448D8	7	0		FF00FC
139	0	0	CE	4639	82BC	0009F0DC	6	0		
140	0	0		415E	8C72	00145ABE	5	0		
141	0	0		3C3D	9628	0028B57C	4	0		
142	0	0		375E	B3D6	00516AF8	3	0		
143	0	0		32B4	F8F0	00A2D5F0	2	0		
144	0	0		32B4	C63C	00A2D5F0	2	0		
145	0	0		32B4	9388	00A2D5F0	2	0		
146	0	0		2E17	C1A8	0145ABE0	1	0		

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.7 – Encoder test sequence (sheet 5 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	ST	Bx	B
147	1	0		2E17	9391	0145ABE0	1	0		
148	0	0		32B4	B85C	00084568	7	0		51
149	0	0		32B4	85A8	00084568	7	0		
150	0	0		2E17	A5E8	00108AD0	6	0		
151	0	0		299A	EFA2	002115A0	5	0		
152	0	0		299A	C608	002115A0	5	0		
153	0	0		299A	9C6E	002115A0	5	0		
154	0	0		2516	E5A8	00422B40	4	0		
155	0	0		2516	C092	00422B40	4	0		
156	0	0		2516	9B7C	00422B40	4	0		
157	0	0		1EDF	ECCC	00845680	3	0		
158	0	0		1EDF	CDED	00845680	3	0		
159	0	0		1EDF	AF0E	00845680	3	0		
160	0	0		1EDF	902F	00845680	3	0		
161	1	0		1AA9	E2A0	0108AD00	2	0		
162	1	0		2516	D548	000BA7B8	7	0		84
163	1	0		299A	9458	00315FA8	5	0		
164	1	0		32B4	A668	00C72998	3	0		
165	1	0		3C3D	CAD0	031E7530	1	0		
166	1	0		415E	F0F4	000C0F0C	7	0		C7
167	0	0	CE	4639	82BC	00197D44	6	0		
168	0	0		415E	8C72	0033738E	5	0		
169	1	0		3C3D	9628	0066E71C	4	0		
170	1	0		415E	F0F4	019D041C	2	0		
171	0	0	CE	4639	82BC	033B6764	1	0		
172	1	0		415E	8C72	000747CE	8	0		CE
173	0	0	CE	4639	82BC	000F25C4	7	0		
174	1	0		415E	8C72	001EC48E	6	0		
175	1	0	CE	4639	82BC	003E1F44	5	0		
176	1	0		4B85	F20C	00F87D10	3	0		
177	1	0	CE	504F	970A	01F2472E	2	0		
178	0	0	CE	5522	8D76	03E48E5C	1	0		
179	0	0		504F	AA44	00018D60	8	0		F9
180	1	0		4B85	B3EA	00031AC0	7	0		
181	1	0	CE	504F	970A	0007064A	6	0		
182	1	0	CE	5522	8D76	000E0C94	5	0		
183	1	0		59EB	E150	00383250	3	0		

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.7 – Encoder test sequence (sheet 6 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	ST	Bx	B
184	0	1		59EB	B3D6	0071736A	2	0		
185	1	0		59EB	B3D6	00E39AAA	1	0		
186	1	1		59EB	B3D6	0007E92A	8	0		38
187	1	1		5522	B3D6	000FD254	7	0		
188	1	1		504F	BD68	001FA4A8	6	0		
189	0	1		4B85	DA32	003F4950	5	0		
190	1	1	CE	504F	970A	007FAFFA	4	0		
191	1	1		4B85	A09E	00FFED6A	3	0		
192	0	1		4639	AA32	01FFDAD4	2	0		
193	0	1	CE	4B85	8C72	04007D9A	1	0		
194	1	1	CE	504F	81DA	0000FB34	8	0	39	00
195	1	1		4B85	A09E	0002597E	7	0		
196	1	1		4639	AA32	0004B2FC	6	0		
197	0	1		415E	C7F2	000965F8	5	0		
198	1	1	CE	4639	82BC	0013D918	4	0		
199	0	1		415E	8C72	00282B36	3	0		
200	0	1	CE	4639	82BC	0050EC94	2	0		
201	1	1		4B85	F20C	0003B250	8	0		28
202	1	1		4B85	A687	0003B250	8	0		
203	1	1		4639	B604	000764A0	7	0		
204	0	1		415E	DF96	000EC940	6	0		
205	1	1	CE	4639	82BC	001ECEF0	5	0		
206	0	1		415E	8C72	003E16E6	4	0		
207	1	1	CE	4639	82BC	007CC3F4	3	0		
208	0	1		415E	8C72	00FA00EE	2	0		
209	1	1	CE	4639	82BC	01F49804	1	0		
210	0	1		415E	8C72	0001A90E	8	0		7D
211	1	1	CE	4639	82BC	0003E844	7	0		
212	0	1		415E	8C72	0008498E	6	0		
213	1	1	CE	4639	82BC	00112944	5	0		
214	0	1		415E	8C72	0022CB8E	4	0		
215	1	1	CE	4639	82BC	00462D44	3	0		
216	1	1		415E	8C72	008CD38E	2	0		
217	1	1		3C3D	9628	0119A71C	1	0		
218	1	1		375E	B3D6	00034E38	8	0		46
219	1	1		32B4	F8F0	00069C70	7	0		
220	1	1		32B4	C63C	00069C70	7	0		

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.7 – Encoder test sequence (sheet 7 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	ST	Bx	B
221	0	1		32B4	9388	00069C70	7	0		
222	1	1		3C3D	CAD0	001BF510	5	0		
223	1	1		3C3D	8E93	001BF510	5	0		
224	1	1		375E	A4AC	0037EA20	4	0		
225	0	1		32B4	DA9C	006FD440	3	0		
226	1	1		3C3D	CAD0	01C1F0A0	1	0		
227	1	1		3C3D	8E93	01C1F0A0	1	0		
228	0	1		375E	A4AC	0003E140	8	0		70
229	1	1		3C3D	DD78	00113A38	6	0		
230	0	1		3C3D	A13B	00113A38	6	0		
231	0	1		415E	F0F4	00467CD8	4	0		
232	1	1	CE	4639	82BC	008E58DC	3	0		
233	0	1		415E	8C72	011D2ABE	2	0		
234	1	1	CE	4639	82BC	023AEBA4	1	0		
235	1	1		415E	8C72	0006504E	8	0		8E
236	1	1		3C3D	9628	000CA09C	7	0		
237	1	1		375E	B3D6	00194138	6	0		
238	1	1		32B4	F8F0	00328270	5	0		
239	1	1		32B4	C63C	00328270	5	0		
240	0	1		32B4	9388	00328270	5	0		
241	1	1		3C3D	CAD0	00CB8D10	3	0		
242	1	1		3C3D	8E93	00CB8D10	3	0		
243	1	1		375E	A4AC	01971A20	2	0		
244	0	1		32B4	DA9C	032E3440	1	0		
245	0	1		3C3D	CAD0	000B70A0	7	0		CB
246	1	1		415E	F0F4	002FFCCC	5	0		
247	1	1		415E	AF96	002FFCCC	5	0		
248	1	1		3C3D	DC70	005FF998	4	0		
249	0	1		3C3D	A033	005FF998	4	0		
250	1	1		415E	F0F4	01817638	2	0		
251	0	1		415E	AF96	01817638	2	0		
252	0	1	CE	4639	82BC	0303C8E0	1	0		
253	1	1		4B85	F20C	000F2380	7	0		C0
254	1	1		4B85	A687	000F2380	7	0		
255	0	1		4639	B604	001E4700	6	0		
256	0	1	CE	4B85	8C72	003D6D96	5	0		
Flush:					81DA	007ADB2C	4	0		F6
										FFD9

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.8 – Decoder test sequence (sheet 1 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	B
1	0	0		5A1D	0000	655B0000	0	65 5B
2	0	0	CE	5A1D	A5E3	655B0000	0	
3	0	0		2586	B43A	332AA200	7	51
4	0	0		2586	8EB4	332AA200	7	
5	0	0		1114	D25C	66554400	6	
6	0	0		1114	C148	66554400	6	
7	0	0		1114	B034	66554400	6	
8	0	0		1114	9F20	66554400	6	
9	0	0		1114	8E0C	66554400	6	
10	0	0		080B	F9F0	CCAA8800	5	
11	0	0		080B	F1E5	CCAA8800	5	
12	0	0		080B	E9DA	CCAA8800	5	
13	0	0		080B	E1CF	CCAA8800	5	
14	0	0		080B	D9C4	CCAA8800	5	
15	1	0		080B	D1B9	CCAA8800	5	
16	0	0		17B9	80B0	2FC88000	1	
17	0	0		1182	D1EE	5F910000	0	
18	0	0		1182	C06C	5F910000	0	
19	0	0		1182	AEEA	5F910000	0	
20	0	0		1182	9D68	5F910000	0	
21	0	0		1182	8BE6	5F910000	0	
22	0	0		0CEF	F4C8	BF228800	7	44
23	0	0		0CEF	E7D9	BF228800	7	
24	0	0		0CEF	DAEA	BF228800	7	
25	0	0		0CEF	CDFB	BF228800	7	
26	1	0		0CEF	C10C	BF228800	7	
27	0	0		1518	CEF0	B0588000	3	
28	1	0		1518	B9D8	B0588000	3	
29	0	0		1AA9	A8C0	5CC40000	0	
30	0	0		1AA9	8E17	5CC40000	0	
31	0	0		174E	E6DC	B989EE00	7	F7
32	1	0		174E	CF8E	B989EE00	7	
33	0	0		1AA9	BA70	0A4F7000	4	
34	0	0		1AA9	9FC7	0A4F7000	4	
35	0	0		1AA9	851E	0A4F7000	4	
36	0	0		174E	D4EA	149EE000	3	
37	0	0		174E	BD9C	149EE000	3	
38	0	0		174E	A64E	149EE000	3	
39	0	0		174E	8F00	149EE000	3	
40	0	0		1424	EF64	293DC000	2	

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.8 – Decoder test sequence (sheet 2 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	B
41	0	0		1424	DB40	293DC000	2	
42	0	0		1424	C71C	293DC000	2	
43	0	0		1424	B2F8	293DC000	2	
44	0	0		1424	9ED4	293DC000	2	
45	0	0		1424	8AB0	293DC000	2	
46	0	0		119C	ED18	527B8000	1	
47	0	0		119C	DB7C	527B8000	1	
48	0	0		119C	C9E0	527B8000	1	
49	0	0		119C	B844	527B8000	1	
50	0	0		119C	A6A8	527B8000	1	
51	0	0		119C	950C	527B8000	1	
52	0	0		119C	8370	527B8000	1	
53	0	0		0F6B	E3A8	A4F70000	0	
54	0	0		0F6B	D43D	A4F70000	0	
55	0	0		0F6B	C4D2	A4F70000	0	
56	0	0		0F6B	B567	A4F70000	0	
57	1	0		0F6B	A5FC	A4F70000	0	
58	1	0		1424	F6B0	E6696000	4	96
59	0	0		1AA9	A120	1EEB0000	1	
60	0	0		1AA9	8677	1EEB0000	1	
61	0	0		174E	D79C	3DD60000	0	
62	0	0		174E	C04E	3DD60000	0	
63	0	0		174E	A900	3DD60000	0	
64	0	0		174E	91B2	3DD60000	0	
65	0	0		1424	F4C8	7BAD3A00	7	9D
66	0	0		1424	E0A4	7BAD3A00	7	
67	0	0		1424	CC80	7BAD3A00	7	
68	0	0		1424	B85C	7BAD3A00	7	
69	0	0		1424	A438	7BAD3A00	7	
70	0	0		1424	9014	7BAD3A00	7	
71	1	0		119C	F7E0	F75A7400	6	
72	1	0		1424	8CE0	88B3A000	3	
73	0	0		1AA9	A120	7FBD0000	0	
74	1	0		1AA9	8677	7FBD0000	0	
75	0	0		2516	D548	9F7A8800	5	51
76	1	0		2516	B032	9F7A8800	5	
77	0	0		299A	9458	517A2000	3	
78	0	0		2516	D57C	A2F44000	2	
79	1	0		2516	B066	A2F44000	2	
80	0	0		299A	9458	5E910000	0	

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.8 – Decoder test sequence (sheet 3 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	B
81	1	0		2516	D57C	BD22F000	7	78
82	0	0		299A	9458	32F3C000	5	
83	0	0		2516	D57C	65E78000	4	
84	0	0		2516	B066	65E78000	4	
85	0	0		2516	8B50	65E78000	4	
86	1	0		1EDF	CC74	CBCF0000	3	
87	1	0		2516	F6F8	F1D00000	0	
88	1	0		299A	9458	7FB95400	6	55
89	0	0		32B4	A668	53ED5000	4	
90	0	0		2E17	E768	A7DAA000	3	
91	1	0		2E17	B951	A7DAA000	3	
92	0	0		32B4	B85C	72828000	1	
93	1	0		32B4	85A8	72828000	1	
94	0	0		3C3D	CAD0	7E3B7E00	7	BF
95	1	0		3C3D	8E93	7E3B7E00	7	
96	0	0		415E	F0F4	AF95F800	5	
97	1	0		415E	AF96	AF95F800	5	
98	0	0	CE	4639	82BC	82BBF000	4	
99	1	0		415E	8C72	8C71E000	3	
100	0	0	CE	4639	82BC	82BBC000	2	
101	1	0		415E	8C72	8C718000	1	
102	0	0	CE	4639	82BC	82BB0000	0	
103	1	0		415E	8C72	8C71FE00	7	FF 00
104	0	0	CE	4639	82BC	82BBFC00	6	
105	1	0		415E	8C72	8C71F800	5	
106	0	0	CE	4639	82BC	82BBF000	4	
107	1	0		415E	8C72	8C71E000	3	
108	0	0	CE	4639	82BC	82BBC000	2	
109	1	0		415E	8C72	8C718000	1	
110	0	0	CE	4639	82BC	82BB0000	0	
111	1	0		415E	8C72	8C71F800	7	FC
112	0	0	CE	4639	82BC	82BBF000	6	
113	1	0		415E	8C72	8C71E000	5	
114	0	0	CE	4639	82BC	82BBC000	4	
115	1	0		415E	8C72	8C718000	3	
116	0	0	CE	4639	82BC	82BB0000	2	
117	1	0		415E	8C72	8C700000	1	
118	0	0	CE	4639	82BC	82B80000	0	
119	1	0		415E	8C72	8C6AA200	7	51
120	0	0	CE	4639	82BC	82AD4400	6	

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.8 – Decoder test sequence (sheet 4 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	B
121	1	0		415E	8C72	8C548800	5	
122	0	0	CE	4639	82BC	82811000	4	
123	1	0		415E	8C72	8BFC2000	3	
124	0	0	CE	4639	82BC	81D04000	2	
125	1	0		415E	8C72	8A9A8000	1	
126	0	0	CE	4639	82BC	7F0D0000	0	
127	1	0		415E	8C72	85150800	7	84
128	0	0	CE	4639	82BC	74021000	6	
129	1	0		415E	8C72	6EFE2000	5	
130	0	0	CE	4639	82BC	47D44000	4	
131	0	0		415E	8C72	16A28000	3	
132	0	0		3C3D	9628	2D450000	2	
133	0	0		375E	B3D6	5A8A0000	1	
134	0	0		32B4	F8F0	B5140000	0	
135	1	0		32B4	C63C	B5140000	0	
136	0	0		3C3D	CAD0	86331C00	6	C7
137	1	0		3C3D	8E93	86331C00	6	
138	1	0		415E	F0F4	CF747000	4	
139	0	0	CE	4639	82BC	3FBCE000	3	
140	0	0		415E	8C72	0673C000	2	
141	0	0		3C3D	9628	0CE78000	1	
142	0	0		375E	B3D6	19CF0000	0	
143	0	0		32B4	F8F0	339F9C00	7	CE
144	0	0		32B4	C63C	339F9C00	7	
145	0	0		32B4	9388	339F9C00	7	
146	0	0		2E17	C1A8	673F3800	6	
147	1	0		2E17	9391	673F3800	6	
148	0	0		32B4	B85C	0714E000	4	
149	0	0		32B4	85A8	0714E000	4	
150	0	0		2E17	A5E8	0E29C000	3	
151	0	0		299A	EFA2	1C538000	2	
152	0	0		299A	C608	1C538000	2	
153	0	0		299A	9C6E	1C538000	2	
154	0	0		2516	E5A8	38A70000	1	
155	0	0		2516	C092	38A70000	1	
156	0	0		2516	9B7C	38A70000	1	
157	0	0		1EDF	ECCC	714E0000	0	
158	0	0		1EDF	CDED	714E0000	0	
159	0	0		1EDF	AF0E	714E0000	0	
160	0	0		1EDF	902F	714E0000	0	

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.8 – Decoder test sequence (sheet 5 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	B
161	1	0		1AA9	E2A0	E29DF200	7	F9
162	1	0		2516	D548	D5379000	4	
163	1	0		299A	9458	94164000	2	
164	1	0		32B4	A668	A5610000	0	
165	1	0		3C3D	CAD0	C6B4E400	6	39
166	1	0		415E	F0F4	E0879000	4	
167	0	0	CE	4639	82BC	61E32000	3	
168	0	0		415E	8C72	4AC04000	2	
169	1	0		3C3D	9628	95808000	1	
170	1	0		415E	F0F4	EE560000	7	00
171	0	0	CE	4639	82BC	7D800000	6	
172	1	0		415E	8C72	81FA0000	5	
173	0	0	CE	4639	82BC	6DCC0000	4	
174	1	0		415E	8C72	62920000	3	
175	1	0	CE	4639	82BC	2EFC0000	2	
176	1	0		4B85	F20C	BBF00000	0	
177	1	0	CE	504F	970A	2AD25000	7	28
178	0	0	CE	5522	8D76	55A4A000	6	
179	0	0		504F	AA44	3AA14000	5	
180	1	0		4B85	B3EA	75428000	4	
181	1	0	CE	504F	970A	19BB0000	3	
182	1	0	CE	5522	8D76	33760000	2	
183	1	0		59EB	E150	CDD80000	0	
184	0	1		59EB	B3D6	8CE6FA00	7	7D
185	1	0		59EB	B3D6	65F7F400	6	
186	1	1		59EB	B3D6	1819E800	5	
187	1	1		5522	B3D6	3033D000	4	
188	1	1		504F	BD68	6067A000	3	
189	0	1		4B85	DA32	C0CF4000	2	
190	1	1	CE	504F	970A	64448000	1	
191	1	1		4B85	A09E	3B130000	0	
192	0	1		4639	AA32	76268C00	7	46
193	0	1	CE	4B85	8C72	245B1800	6	
194	1	1	CE	504F	81DA	48B63000	5	
195	1	1		4B85	A09E	2E566000	4	
196	1	1		4639	AA32	5CACC000	3	
197	0	1		415E	C7F2	B9598000	2	
198	1	1	CE	4639	82BC	658B0000	1	
199	0	1		415E	8C72	52100000	0	
200	0	1	CE	4639	82BC	0DF8E000	7	70

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.8 – Decoder test sequence (sheet 6 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	B
201	1	1		4B85	F20C	37E38000	5	
202	1	1		4B85	A687	37E38000	5	
203	1	1		4639	B604	6FC70000	4	
204	0	1		415E	DF96	DF8E0000	3	
205	1	1	CE	4639	82BC	82AC0000	2	
206	0	1		415E	8C72	8C520000	1	
207	1	1	CE	4639	82BC	827C0000	0	
208	0	1		415E	8C72	8BF31C00	7	8E
209	1	1	CE	4639	82BC	81BE3800	6	
210	0	1		415E	8C72	8A767000	5	
211	1	1	CE	4639	82BC	7EC4E000	4	
212	0	1		415E	8C72	8483C000	3	
213	1	1	CE	4639	82BC	72DF8000	2	
214	0	1		415E	8C72	6CB90000	1	
215	1	1	CE	4639	82BC	434A0000	0	
216	1	1		415E	8C72	0D8F9600	7	CB
217	1	1		3C3D	9628	1B1F2C00	6	
218	1	1		375E	B3D6	363E5800	5	
219	1	1		32B4	F8F0	6C7CB000	4	
220	1	1		32B4	C63C	6C7CB000	4	
221	0	1		32B4	9388	6C7CB000	4	
222	1	1		3C3D	CAD0	2EA2C000	2	
223	1	1		3C3D	8E93	2EA2C000	2	
224	1	1		375E	A4AC	5D458000	1	
225	0	1		32B4	DA9C	BA8B0000	0	
226	1	1		3C3D	CAD0	4A8F0000	6	C0
227	1	1		3C3D	8E93	4A8F0000	6	
228	0	1		375E	A4AC	951E0000	5	
229	1	1		3C3D	DD78	9F400000	3	
230	0	1		3C3D	A13B	9F400000	3	
231	0	1		415E	F0F4	E9080000	1	
232	1	1	CE	4639	82BC	72E40000	0	
233	0	1		415E	8C72	6CC3EC00	7	F6
234	1	1	CE	4639	82BC	435FD800	6	
235	1	1		415E	8C72	0DB9B000	5	
236	1	1		3C3D	9628	1B736000	4	
237	1	1		375E	B3D6	36E6C000	3	
238	1	1		32B4	F8F0	6DCD8000	2	
239	1	1		32B4	C63C	6DCD8000	2	
240	0	1		32B4	9388	6DCD8000	2	

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Table K.8 – Decoder test sequence (sheet 7 of 7)

EC	D	MPS	CX	Qe (hexadecimal)	A (hexadecimal)	C (hexadecimal)	CT	B
241	1	1		3C3D	CAD0	33E60000	0	
242	1	1		3C3D	8E93	33E60000	0	
Marker detected: zero byte fed to decoder								
243	1	1		375E	A4AC	67CC0000	7	
244	0	1		32B4	DA9C	CF980000	6	
245	0	1		3C3D	CAD0	9EC00000	4	
246	1	1		415E	F0F4	40B40000	2	
247	1	1		415E	AF96	40B40000	2	
248	1	1		3C3D	DC70	81680000	1	
249	0	1		3C3D	A033	81680000	1	
Marker detected: zero byte fed to decoder								
250	1	1		415E	F0F4	75C80000	7	
251	0	1		415E	AF96	75C80000	7	
252	0	1	CE	4639	82BC	0F200000	6	
253	1	1		4B85	F20C	3C800000	4	
254	1	1		4B85	A687	3C800000	4	
255	0	1		4639	B604	79000000	3	
256	0	1	CE	4B85	8C72	126A0000	2	

K.5 Low-pass downsampling filters for hierarchical coding

In this section simple examples are given of downsampling filters which are compatible with the upsampling filter defined in J.1.1.2.

Figure K.5 shows the weighting of neighbouring samples for simple one-dimensional horizontal and vertical low-pass filters. The output of the filter must be normalized by the sum of the neighbourhood weights.

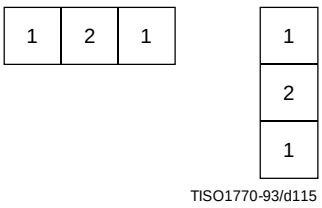


Figure K.5 – Low-pass filter example

The centre sample in Figure K.5 should be aligned with the left column or top line of the high resolution image when calculating the left column or top line of the low resolution image. Sample values which are situated outside of the image boundary are replicated from the sample values at the boundary to provide missing edge values.

If the image being downsampled has an odd width or length, the odd dimension is increased by 1 by sample replication on the right edge or bottom line before downsampling.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

K.6 Domain of applicability of DCT and spatial coding techniques

The DCT coder is intended for lossy coding in a range from quite visible loss to distortion well below the threshold for visibility. However in general, DCT-based processes cannot be used for true lossless coding.

The lossless coder is intended for completely lossless coding. The lossless coding process is significantly less effective than the DCT-based processes for distortions near and above the threshold of visibility.

The point transform of the input to the lossless coder permits a very restricted form of lossy coding with the "lossless" coder. (The coder is still lossless after the input point transform.) Since the DCT is intended for lossy coding, there may be some confusion about when this alternative lossy technique should be used.

Lossless coding with a point transformed input is intended for applications which cannot be addressed by DCT coding techniques. Among these are

- true lossless coding to a specified precision;
- lossy coding with precisely defined error bounds;
- hierarchical progression to a truly lossless final stage.

If lossless coding with a point transformed input is used in applications which can be met effectively by DCT coding, the results will be significantly less satisfactory. For example, distortion in the form of visible contours usually appears when precision of the luminance component is reduced to about six bits. For normal image data, this occurs at bit rates well above those for which the DCT gives outputs which are visually indistinguishable from the source.

K.7 Domain of applicability of the progressive coding modes of operation

Two very different progressive coding modes of operation have been defined, progressive coding of the DCT coefficients and hierarchical progression. Progressive coding of the DCT coefficients has two complementary procedures, spectral selection and successive approximation. Because of this diversity of choices, there may be some confusion as to which method of progression to use for a given application.

K.7.1 Progressive coding of the DCT

In progressive coding of the DCT coefficients two complementary procedures are defined for decomposing the 8×8 DCT coefficient array, spectral selection and successive approximation. Spectral selection partitions zig-zag array of DCT coefficients into "bands", one band being coded in each scan. Successive approximation codes the coefficients with reduced precision in the first scan; in each subsequent scan the precision is increased by one bit.

A single forward DCT is calculated for these procedures. When all coefficients are coded to full precision, the DCT is the same as in the sequential mode. Therefore, like the sequential DCT coding, progressive coding of DCT coefficients is intended for applications which need very good compression for a given level of visual distortion.

The simplest progressive coding technique is spectral selection; indeed, because of this simplicity, some applications may choose – despite the limited progression that can be achieved – to use only spectral selection. Note, however, that the absence of high frequency bands typically leads – for a given bit rate – to a significantly lower image quality in the intermediate stages than can be achieved with the more general progressions. The net coding efficiency at the completion of the final stage is typically comparable to or slightly less than that achieved with the sequential DCT.

A much more flexible progressive system is attained at some increase in complexity when successive approximation is added to the spectral selection progression. For a given bit rate, this system typically provides significantly better image quality than spectral selection alone. The net coding efficiency at the completion of the final stage is typically comparable to or slightly better than that achieved with the sequential DCT.

K.7.2 Hierarchical progression

Hierarchical progression permits a sequence of outputs of increasing spatial resolution, and also allows refinement of image quality at a given spatial resolution. Both DCT and spatial versions of the hierarchical progression are allowed, and progressive coding of DCT coefficients may be used in a frame of the DCT hierarchical progression.

The DCT hierarchical progression is intended for applications which need very good compression for a given level of visual distortion; the spatial hierarchical progression is intended for applications which need a simple progression with a truly lossless final stage. Figure K.6 illustrates examples of these two basic hierarchical progressions.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

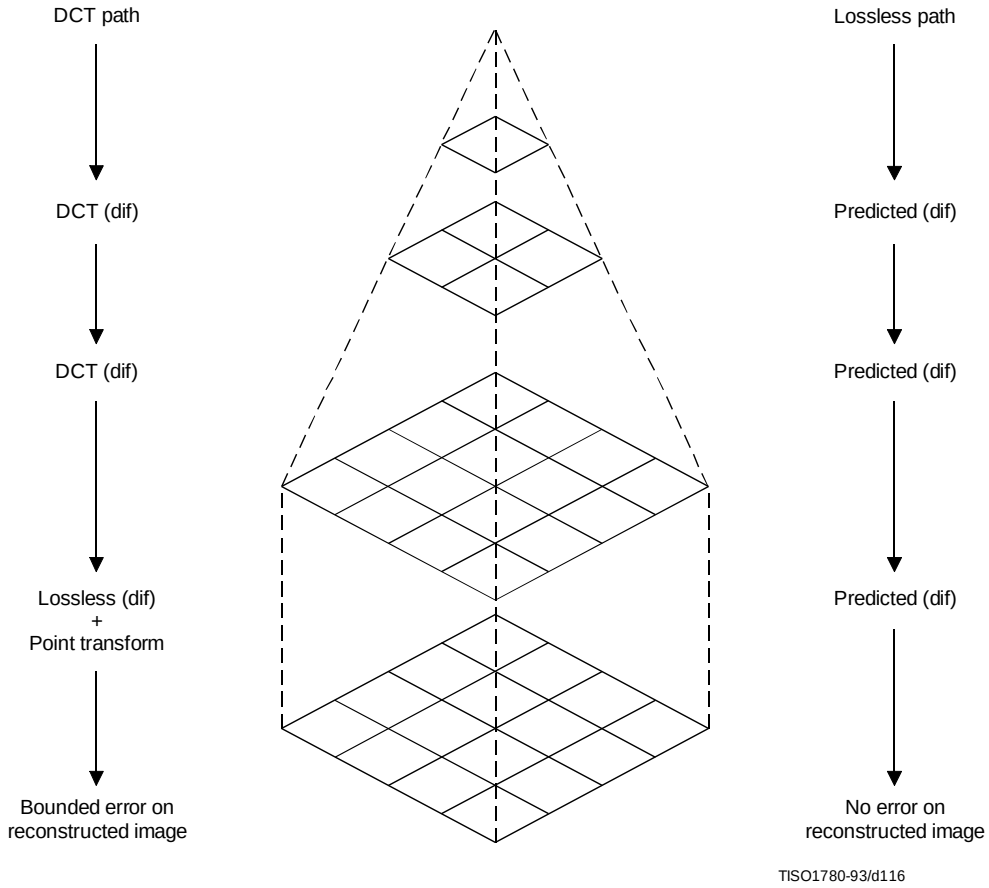


Figure K.6 – Sketch of the basic operations of the hierarchical mode

K.7.2.1 DCT Hierarchical progression

If a DCT hierarchical progression uses reduced spatial resolution, the early stages of the progression can have better image quality for a given bit rate than the early stages of non-hierarchical progressive coding of the DCT coefficients. However, at the point where the distortion between source and output becomes indistinguishable, the coding efficiency achieved with a DCT hierarchical progression is typically significantly lower than the coding efficiency achieved with a non-hierarchical progressive coding of the DCT coefficients.

While the hierarchical DCT progression is intended for lossy progressive coding, a final spatial differential coding stage can be used. When this final stage is used, the output can be almost lossless, limited only by the difference between the encoder and decoder IDCT implementations. Since IDCT implementations can differ significantly, truly lossless coding after a DCT hierarchical progression cannot be guaranteed. An important alternative, therefore, is to use the input point transform of the final lossless differential coding stage to reduce the precision of the differential input. This allows a bounding of the difference between source and output at a significantly lower cost in coded bits than coding of the full precision spatial difference would require.

K.7.2.2 Spatial hierarchical progression

If lossless progression is required, a very simple hierarchical progression may be used in which the spatial lossless coder with point transformed input is used as a first stage. This first stage is followed by one or more spatial differential coding stages. The first stage should be nearly lossless, such that the low order bits which are truncated by the point transform are essentially random – otherwise the compression efficiency will be degraded relative to non-progressive lossless coding.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

K.8 Suppression of block-to-block discontinuities in decoded images

A simple technique is available for suppressing the block-to-block discontinuities which can occur in images compressed by DCT techniques.

The first few (five in this example) low frequency DCT coefficients are predicted from the nine DC values of the block and the eight nearest-neighbour blocks, and the predicted values are used to suppress blocking artifacts in smooth areas of the image.

The prediction equations for the first five AC coefficients in the zig-zag sequence are obtained as follows:

K.8.1 AC prediction

The sample field in a 3 by 3 array of blocks (each block containing an 8 × 8 array of samples) is modeled by a two-dimensional second degree polynomial of the form:

$$P(x,y) = A1(x^2y^2) + A2(x^2y) + A3(xy^2) + A4(x^2) + A5(xy) + A6(y^2) + A7(x) + A8(y) + A9$$

The nine coefficients A1 through A9 are uniquely determined by imposing the constraint that the mean of P(x,y) over each of the nine blocks must yield the correct DC-values.

Applying the DCT to the quadratic field predicting the samples in the central block gives a prediction of the low frequency AC coefficients depicted in Figure K.7.

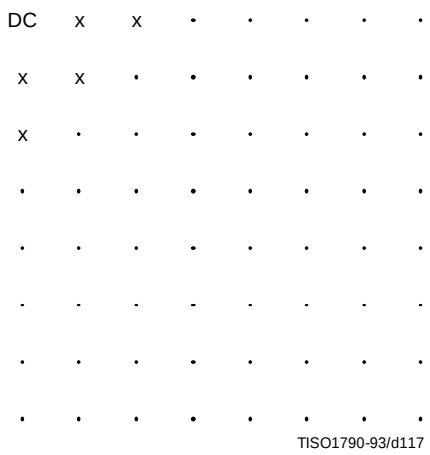


Figure K.7 – DCT array positions of predicted AC coefficients

The prediction equations derived in this manner are as follows:

For the two dimensional array of DC values shown

DC_1 DC_2 DC_3
DC_4 DC_5 DC_6
DC_7 DC_8 DC_9

The unquantized prediction equations are

$$AC_{01} = 1{,}13885\ (DC_4 - DC_6)$$
$$AC_{10} = 1{,}13885\ (DC_2 - DC_8)$$
$$AC_{20} = 0{,}27881\ (DC_2 + DC_8 - 2 \times DC_5)$$
$$AC_{11} = 0{,}16213\ ((DC_1 - DC_3) - (DC_7 - DC_9))$$
$$AC_{02} = 0{,}27881\ (DC_4 + DC_6 - 2 \times DC_5)$$

The scaling of the predicted AC coefficients is consistent with the DCT normalization defined in A.3.3.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

K.8.2 Quantized AC prediction

The prediction equations can be mapped to a form which uses quantized values of the DC coefficients and which computes quantized AC coefficients using integer arithmetic. The quantized DC coefficients need to be scaled, however, such that the predicted coefficients have fractional bit precision.

First, the prediction equation coefficients are scaled by 32 and rounded to the nearest integer. Thus,

$$1{,}13885 \times 32 = 36$$

$$0{,}27881 \times 32 = 9$$

$$0{,}16213 \times 32 = 5$$

The multiplicative factors are then scaled by the ratio of the DC and AC quantization factors and rounded appropriately. The normalization defined for the DCT introduces another factor of 8 in the unquantized DC values. Therefore, in terms of the quantized DC values, the predicted quantized AC coefficients are given by the equations below. Note that if (for example) the DC values are scaled by a factor of 4, the AC predictions will have 2 fractional bits of precision relative to the quantized DCT coefficients.

$$QAC_{01} = ((R_d \times Q_{01}) + (36 \times Q_{00} \times (QDC_4 - QDC_6)))/(256 \times Q_{01})$$
$$QAC_{10} = ((R_d \times Q_{10}) + (36 \times Q_{00} \times (QDC_2 - QDC_8)))/(256 \times Q_{10})$$
$$QAC_{20} = ((R_d \times Q_{20}) + (9 \times Q_{00} \times (QDC_2 + QDC_8 - 2 \times QDC_5)))/(256 \times Q_{20})$$
$$QAC_{11} = ((R_d \times Q_{11}) + (5 \times Q_{00} \times ((QDC_1 - QDC_3) - (QDC_7 - QDC_9))))/(256 \times Q_{11})$$
$$QAC_{02} = ((R_d \times Q_{02}) + (9 \times Q_{00} \times (QDC_4 + QDC_6 - 2 \times QDC_5)))/(256 \times Q_{02})$$

where QDC_x and QAC_{xy} are the quantized and scaled DC and AC coefficient values. The constant Rd is added to get a correct rounding in the division. Rd is 128 for positive numerators, and –128 for negative numerators.

Predicted values should not override coded values. Therefore, predicted values for coefficients which are already non-zero should be set to zero. Predictions should be clamped if they exceed a value which would be quantized to a non-zero value for the current precision in the successive approximation.

K.9 Modification of dequantization to improve displayed image quality

For a progression where the first stage successive approximation bit, Al, is set to 3, uniform quantization of the DCT gives the following quantization and dequantization levels for a sequence of successive approximation scans, as shown in Figure K.8:

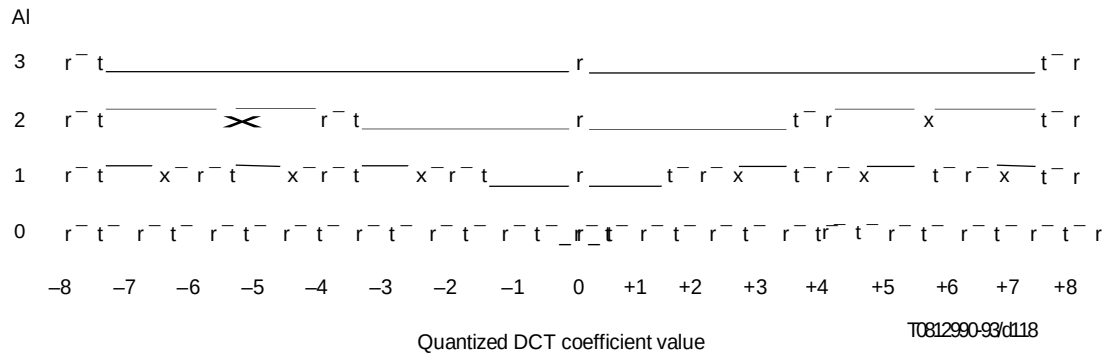


Figure K.8 – Illustration of two reconstruction strategies

The column to the left labelled “Al” gives the bit position specified in the scan header. The quantized DCT coefficient magnitudes are therefore divided by 2^{Al} during that scan.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Referring to the final scan (Al = 0), the points marked with “t” are the threshold values, while the points marked with “r” are the reconstruction values. The unquantized output is obtained by multiplying the horizontal scale in Figure K.8 by the quantization value.

The quantization interval for a coefficient value of zero is indicated by the depressed interval of the line. As the bit position Al is increased, a “fat zero” quantization interval develops around the zero DCT coefficient value. In the limit where the scaling factor is very large, the zero interval is twice as large as the rest of the quantization intervals.

Two different reconstruction strategies are shown. The points marked “r” are the reconstruction obtained using the normal rounding rules for the DCT for the complete full precision output. This rule seems to give better image quality when high bandwidth displays are used. The points marked “x” are an alternative reconstruction which tends to give better images on lower bandwidth displays. “x” and “r” are the same for slice 0. The system designer must determine which strategy is best for the display system being used.

K.10 Example of point transform

The difference between the arithmetic-shift-right by Pt and divide by 2^{Pt} can be seen from the following:

After the level shift the DC has values from +127 to –128. Consider values near zero (after the level shift), and the case where Pt = 1:

Before level shift	Before point transform	After divide by 2	After shift-right-arithmetic 1
131	+3	+1	+1
130	+2	+1	+1
129	+1	0	0
128	0	0	0
127	–1	0	–1
126	–2	–1	–1
125	–3	–1	–2
124	–4	–2	–2
123	–5	–2	–3

The key difference is in the truncation of precision. The divide truncates the magnitude; the arithmetic shift truncates the LSB. With a divide by 2 we would get non-uniform quantization of the DC values; therefore we use the shift-right-arithmetic operation.

For positive values, the divide by 2 and the shift-right-arithmetic by 1 operations are the same. Therefore, the shift-right-arithmetic by 1 operation effectively is a divide by 2 when the point transform is done before the level shift.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex L

Patents

(This annex does not form an integral part of this Recommendation | International Standard)

L.1 Introductory remarks

The user's attention is called to the possibility that – for some of the coding processes specified in Annexes F, G, H, and J – compliance with this Specification may require use of an invention covered by patent rights.

By publication of this Specification, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. However, for each patent listed in this annex, the patent holder has filed with the Information Technology Task Force (ITTF) and the Telecommunication Standardization Bureau (TSB) a statement of willingness to grant a license under these rights on reasonable and non-discriminatory terms and conditions to applicants desiring to obtain such a license.

The criteria for including patents in this annex are:

a) the patent has been identified by someone who is familiar with the technical fields relevant to this Specification, and who believes use of the invention covered by the patent is *required* for implementation of one or more of the coding processes specified in Annexes F, G, H, or J;

b) the patent-holder has written a letter to the ITTF and TSB, stating willingness to grant a license to an unlimited number of applicants throughout the world under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

This list of patents shall be updated, if necessary, upon publication of any revisions to the Recommendation | International Standard.

L.2 List of patents

The following patents may be required for implementation of any one of the processes specified in Annexes F, G, H, and J which uses arithmetic coding:

US 4,633,490, December 30, 1986, IBM, MITCHELL (J.L.) and GOERTZEL (G.): *Symmetrical Adaptive Data Compression/Decompression System.*

US 4,652,856, February 4, 1986, IBM, MOHIUDDIN (K.M.) and RISSANEN (J.J.): *A Multiplication-free Multi-Alphabet Arithmetic Code.*

US 4,369,463, January 18, 1983, IBM, ANASTASSIOU (D.) and MITCHELL (J.L.): *Grey Scale Image Compression with Code Words a Function of Image History.*

US 4,749,983, June 7, 1988, IBM, LANGDON (G.): *Compression of Multilevel Signals.*

US 4,935,882, June 19, 1990, IBM, PENNEBAKER (W.B.) and MITCHELL (J.L.): *Probability Adaptation for Arithmetic Coders*.

US 4,905,297, February 27, 1990, IBM, LANGDON (G.G.), Jr., MITCHELL (J.L.), PENNEBAKER (W.B.), and RISSANEN (J.J.): *Arithmetic Coding Encoder and Decoder System*.

US 4,973,961, November 27, 1990, AT&T, CHAMZAS (C.), DUTTWEILER (D.L.): *Method and Apparatus for Carry-over Control in Arithmetic Entropy Coding.*

US 5,025,258, June 18, 1991, AT&T, DUTTWEILER (D.L): *Adaptive Probability Estimator for Entropy Encoding/Decoding*.

US 5,099,440, March 24, 1992, IBM, PENNEBAKER (W.B.) and MITCHELL (J.L.): *Probability Adaptation for Arithmetic Coders*.

Japanese Patent Application 2-46275, February 26, 1990, MEL ONO (F.), KIMURA (T.), YOSHIDA (M.), and KINO (S.): *Coding System*.

The following patent may be required for implementation of any one of the hierarchical processes specified in Annex H when used with a lossless final frame:

US 4,665,436, May 12, 1987, EI OSBORNE (J.A.) and SEIFFERT (C.): *Narrow Bandwidth Signal Transmission*.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

No other patents required for implementation of any of the other processes specified in Annexes F, G, H, or J had been identified at the time of publication of this Specification.

L.3 Contact addresses for patent information

Director, Telecommunication Standardization Bureau (formerly CCITT)
International Telecommunication Union
Place des Nations
CH-1211 Genève 20, Switzerland
Tel. +41 (22) 730 5111
Fax: +41 (22) 730 5853

Information Technology Task Force
International Organization for Standardization
1, rue de Varembé
CH-1211 Genève 20, Switzerland
Tel: +41 (22) 734 0150
Fax: +41 (22) 733 3843

Program Manager, Licensing
Intellectual Property and Licensing Services
IBM Corporation
208 Harbor Drive
P.O. Box 10501
Stamford, Connecticut 08904-2501, USA
Tel: +1 (203) 973 7935
Fax: +1 (203) 973 7981 or +1 (203) 973 7982

Mitsubishi Electric Corp.
Intellectual Property License Department
1-2-3 Morunouchi, Chiyoda-ku
Tokyo 100, Japan
Tel: +81 (3) 3218 3465
Fax: +81 (3) 3215 3842

AT&T Intellectual Property Division Manager
Room 3A21
10 Independence Blvd.
Warren, NJ 07059, USA
Tel: +1 (908) 580 5392
Fax: +1 (908) 580 6355

Senior General Manager
Corporate Intellectual Property and Legal Headquarters
Canon Inc.
30-2 Shimomaruko 3-chome
Ohta-ku Tokyo 146 Japan
Tel: +81 (3) 3758 2111
Fax: +81 (3) 3756 0947

Chief Executive Officer
Electronic Imagery, Inc.
1100 Park Central Boulevard South
Suite 3400
Pompano Beach, FL 33064, USA
Tel: +1 (305) 968 7100
Fax: +1 (305) 968 7319

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Annex M

Bibliography

(This annex does not form an integral part of this Recommendation | International Standard)

M.1 General references

LEGER (A.), OMACHI (T.), and WALLACE (G.K.): JPEG Still Picture Compression Algorithm, *Optical Engineering*, Vol. 30, No. 7, pp. 947-954, 1991.

RABBANI (M.) and JONES (P.): Digital Image Compression Techniques, *Tutorial Texts in Optical Engineering*, Vol. TT7, SPIE Press, 1991.

HUDSON (G.), YASUDA (H.) and SEBESTYEN (I.): The International Standardization of a Still Picture Compression Technique, *Proc. of IEEE Global Telecommunications Conference*, pp. 1016-1021, 1988.

LEGER (A.), MITCHELL (J.) and YAMAZAKI (Y.): Still Picture Compression Algorithm Evaluated for International Standardization, *Proc. of the IEEE Global Telecommunications Conference*, pp. 1028-1032, 1988.

WALLACE (G.), VIVIAN (R.) and POULSEN (H.): Subjective Testing Results for Still Picture Compression Algorithms for International Standardization, *Proc. of the IEEE Global Telecommunications Conference*, pp. 1022-1027, 1988.

MITCHELL (J.L.) and PENNEBAKER (W.B.): Evolving JPEG Colour Data Compression Standard, *Standards for Electronic Imaging Systems*, M. Nier, M.E. Courtot, Editors, SPIE, Vol. CR37, pp. 68-97, 1991.

WALLACE (G.K.): The JPEG Still Picture Compression Standard, *Communications of the ACM*, Vol. 34, No. 4, pp. 31-44, 1991.

NETRAVALI (A.N.) and HASKELL (B.G.): *Digital Pictures: Representation and Compression*, Plenum Press, New York 1988.

PENNEBAKER (W.B.) and MITCHELL (J.L.): *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, New York 1993.

M.2 DCT references

CHEN (W.), SMITH (C.H.) and FRALICK (S.C.): A Fast Computational Algorithm for the Discrete Cosine Transform, *IEEE Trans. on Communications*, Vol. COM-25, pp. 1004-1009, 1977.

AHMED (N.), NATARAJAN (T.) and RAO (K.R.): Discrete Cosine Transform, *IEEE Trans. on Computers*, Vol. C-23, pp. 90-93, 1974.

NARASINHA (N.J.) and PETERSON (A.M.): On the Computation of the Discrete Cosine Transform, *IEEE Trans. on Communications*, Vol. COM-26, No. 6, pp. 966-968, 1978.

DUHAMEL (P.) and GUILLEMOT (C.): Polynomial Transform Computation of the 2-D DCT, *Proc. IEEE ICASSP-90*, pp. 1515-1518, Albuquerque, New Mexico 1990.

FEIG (E.): A Fast Scaled DCT Algorithm, in *Image Processing Algorithms and Techniques*, Proc. SPIE, Vol. 1244, K.S. Pennington and R. J. Moorhead II, Editors, pp. 2-13, Santa Clara, California, 1990.

HOU (H.S.): A Fast Recursive Algorithm for Computing the Discrete Cosine Transform, *IEEE Trans. Acoust. Speech and Signal Processing*, Vol. ASSP-35, No. 10, pp. 1455-1461.

LEE (B.G.): A New Algorithm to Compute the Discrete Cosine Transform, *IEEE Trans. on Acoust., Speech and Signal Processing*, Vol. ASSP-32, No. 6, pp. 1243-1245, 1984.

LINZER (E.N.) and FEIG (E.): New DCT and Scaled DCT Algorithms for Fused Multiply/Add Architectures, *Proc. IEEE ICASSP-91*, pp. 2201-2204, Toronto, Canada, 1991.

VETTERLI (M.) and NUSSBAUMER (H.J.): Simple FFT and DCT Algorithms with Reduced Number of Operations, *Signal Processing*, 1984.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

VETTERLI (M.): Fast 2-D Discrete Cosine Transform, *Proc. IEEE ICASSP-85*, pp. 1538-1541, Tampa, Florida, 1985.

ARAI (Y.), AGUI (T.), and NAKAJIMA (M.): A Fast DCT-SQ Scheme for Images, *Trans. of IEICE*, Vol. E.71, No. 11, pp. 1095-1097, 1988.

SUEHIRO (N.) and HATORI (M.): Fast Algorithms for the DFT and other Sinusoidal Transforms, *IEEE Trans. on Acoust., Speech and Signal Processing*, Vol ASSP-34, No. 3, pp. 642-644, 1986.

M.3 Quantization and human visual model references

CHEN (W.H.) and PRATT (W.K.): Scene adaptive coder, *IEEE Trans. on Communications*, Vol. COM-32, pp. 225-232, 1984.

GRANRATH (D.J.): The role of human visual models in image processing, *Proceedings of the IEEE*, Vol. 67, pp. 552-561, 1981.

LOHSCHELLER (H.): Vision adapted progressive image transmission, *Proceedings of EUSIPCO*, Vol. 83, pp. 191-194, 1983.

LOHSCHELLER (H.) and FRANKE (U.): Colour picture coding – Algorithm optimization and technical realization, *Frequenze*, Vol. 41, pp. 291-299, 1987.

LOHSCHELLER (H.): A subjectively adapted image communication system, *IEEE Trans. on Communications*, Vol. COM-32, pp. 1316-1322, 1984.

PETERSON (H.A.) *et al*: Quantization of colour image components in the DCT domain, *SPIE/IS&T 1991 Symposium on Electronic Imaging Science and Technology*, 1991.

M.4 Arithmetic coding references

LANGDON (G.): An Introduction to Arithmetic Coding, *IBM J. Res. Develop.*, Vol. 28, pp. 135-149, 1984.

PENNEBAKER (W.B.), MITCHELL (J.L.), LANGDON (G.) Jr., and ARPS (R.B.): An Overview of the Basic Principles of the Q-Coder Binary Arithmetic Coder, *IBM J. Res. Develop.*, Vol. 32, No. 6, pp. 717-726, 1988.

MITCHELL (J.L.) and PENNEBAKER (W.B.): Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder Binary Arithmetic Coder, *IBM J. Res. Develop.*, Vol. 32, No. 6, pp. 727-736, 1988.

PENNEBAKER (W.B.) and MITCHELL (J.L.): Probability Estimation for the Q-Coder, *IBM J. Res. Develop.*, Vol. 32, No. 6, pp. 737-752, 1988.

MITCHELL (J.L.) and PENNEBAKER (W.B.): Software Implementations of the Q-Coder, *IBM J. Res. Develop.*, Vol. 32, No. 6, pp. 753-774, 1988.

ARPS (R.B.), TRUONG (T.K.), LU (D.J.), PASCO (R.C.) and FRIEDMAN (T.D.): A Multi-Purpose VLSI Chip for Adaptive Data Compression of Bilevel Images, *IBM J. Res. Develop.*, Vol. 32, No. 6, pp. 775-795, 1988.

ONO (F.), YOSHIDA (M.), KIMURA (T.) and KINO (S.): Subtraction-type Arithmetic Coding with MPS/LPS Conditional Exchange, *Annual Spring Conference of IECED*, Japan, D-288, 1990.

DUTTWEILER (D.) and CHAMZAS (C.): Probability Estimation in Arithmetic and Adaptive-Huffman Entropy Coders, submitted to *IEEE Trans. on Image Processing*.

JONES (C.B.): An Efficient Coding System for Long Source Sequences, *IEEE Trans. Inf. Theory*,Vol. IT-27, pp. 280-291, 1981.

LANGDON (G.): Method for Carry-over Control in a Fifo Arithmetic Code String, *IBM Technical Disclosure Bulletin*, Vol. 23, No.1, pp. 310-312, 1980.

M.5 Huffman coding references

HUFFMAN (D.A.): A Method for the Construction of Minimum Redundancy codes, *Proc. IRE*, Vol. 40, pp. 1098-1101, 1952.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T T.6

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

TERMINAL EQUIPMENT AND PROTOCOLS FOR TELEMATIC SERVICES

FACSIMILE CODING SCHEMES AND CODING CONTROL FUNCTIONS FOR GROUP 4 FACSIMILE APPARATUS

ITU-T Recommendation T.6

(Extract from the *Blue Book*)

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

NOTES

1 ITU-T Recommendation T.6 was published in Fascicle VII.3 of the *Blue Book*. This file is an extract from the *Blue Book*. While the presentation and layout of the text might be slightly different from the *Blue Book* version, the contents of the file are identical to the *Blue Book* version and copyright conditions remain unchanged (see below).

2 In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

© ITU 1988, 1993

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

Recommendation T.6

FACSIMILE CODING SCHEMES AND CODING CONTROL FUNCTIONS FOR GROUP 4 FACSIMILE APPARATUS

(Malaga-Torremolinos, 1984; amended at Melbourne, 1988)

1 General

1.1 *Scope*

1.1.1 Recommendation T.6 defines the facsimile coding schemes and their control functions to be used in the Group 4 facsimile.

1.1.2 This Recommendation should be read in conjunction with the following Recommendations:

T.503 A document application profile for the interchange of Group 4 facsimile documents

T.521 Communication application profile for document bulk transfer based on the session service (according to the rules defined in T.62 *bis*)

T.563 Terminal characteristics for Group 4 facsimile apparatus

T.73 Document interchange protocol for the Telematic services

T.62 Control procedures for Teletex and Group 4 facsimile services

T.62 *bis* Control procedures for Teletex and Group 4 facsimile services based on Recommendations X.215/X.225

T.70 Network-independent basic transport service for Telematic services

F. 161 International Group 4 facsimile service

In addition, in the case of Group 4 Class II/III (Teletex or mixed mode of operation), the following Recommendations should also be read:

T.60 Terminal equipment for use in the Teletex service

T.61 Character repertoire and coded character sets for the international Teletex service

T.72 Terminal capabilities for mixed mode of operation

1.2 *Fundamental principles*

1.2.1 *Facsimile coding schemes and coding control functions*

Facsimile coding schemes consist of the basic facsimile coding scheme and optional facsimile coding schemes. They are defined in § 2 and §§ 3 and 4, respectively.

Facsimile coding schemes are specified assuming that transmission errors are corrected by control procedures at a lower level.

The basic facsimile coding scheme is the two-dimensional coding scheme which is in principle the same as the two-dimensional coding scheme of Group 3 facsimile specified in Recommendation T.4.

Optional facsimile coding schemes are specified not only for black and white images but also for grey scale images and colour images.

Facsimile coding control functions are used in facsimile user information in order to change facsimile parameters or to invoke the end of facsimile block. They are defined in § 2.4.

2 Facsimile coding schemes and coding control functions for black and white images

2.1 *General*

This section specifies the facsimile coding schemes, and associated control functions for black and white images.

Facsimile coding schemes consist of the basic facsimile coding scheme and optional coding schemes.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

The use of the optional facsimile coding schemes is subject to mutual agreement between terminals and shall be initiated by the appropriate procedural steps.

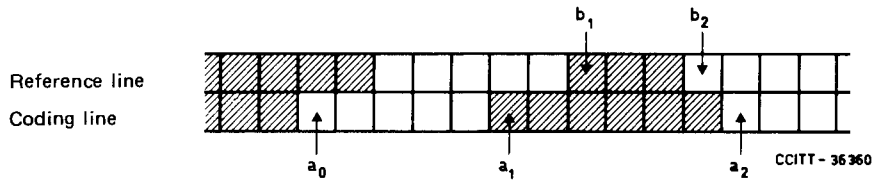
2.2 *Basic facsimile coding scheme*

2.2.1 *Principle of the coding scheme*

The coding scheme uses a two-dimensional line-by-line coding method in which the position of each changing picture element on the current coding line is coded with respect to the position of a corresponding reference element situated on either the coding line or the reference line which is immediately above the coding line. After the coding line has been coded, it becomes the reference line for the next coding line. The reference line for the first coding line in a page is an imaginary white line.

2.2.2 *Definition of changing picture elements* (see Figure 1 /T.6)

A changing element is defined as an element whose "colour" (i.e. black or white) is different from that of the previous element along the same scan line.



a_0 : The reference or starting changing element on the coding line. At the start of the line a_0 is set on an imaginary white changing element situated just before the first element on the line. During the coding of the coding line, the position of a_0 is defined by the previous coding mode (see § 2.2.3).

a_1 : The next changing element to the right of a_0 on the coding line.

a_2 : The next changing element to the right of a_1 on the coding line.

b_1 : The first changing element on the reference line to the right of a_0 and of opposite colour to a_0.

b_2 : The next changing element to the right of b_1 on the reference line.

FIGURE 1/T.6

Changing picture elements

2.2.3 *Coding modes*

One of the three coding modes are chosen according to the coding procedure described in § 2.2.4 to code the position of each changing element along the coding line. Examples of the three coding modes are given in Figure 2/T.6, 3/T.6 and 4/T.6.

2.2.3.1 *Pass mode*

This mode is identified when the position of b_2 lies to the left of a_1. (See Figure 2/T.6.)

However, the state where b_2 occurs just above a_1 as shown in Figure 3/T.6 is not considered as a pass mode.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

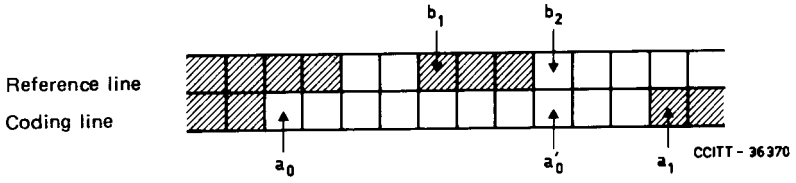


FIGURE 2/T.6

Pass mode

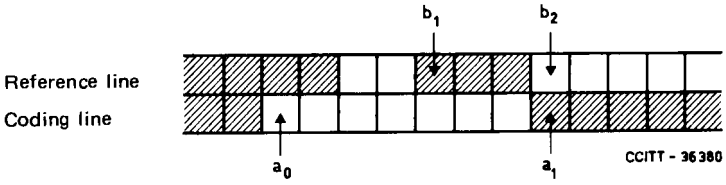


FIGURE 3/T.6

An example not corresponding to a Pass mode

2.2.3.2 *Vertical mode*

When this mode is identified, the position of a_1 is coded relative to the position of b_1. The relative distance a_1b_1 can take on one of seven values V(0), $V_R(1)$, $V_R(2)$, $V_R(3)$, $V_L(1)$, $V_L(2)$ and $V_L(3)$, each of which is represented by a separate code word. The subscripts R and L indicate that a_1 is to the right or left respectively of b_1, and the number in brackets indicates the value of the distance a_1b_1 (see Figure 4/T.6).

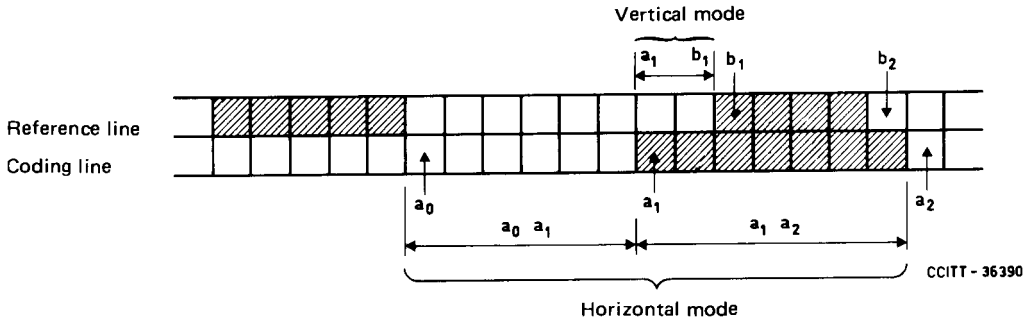


FIGURE 4/T.6

Vertical mode and horizontal mode

2.2.3.3 *Horizontal mode*

When this mode is identified, both the run-lengths a_0a_1 and a_1a_2 are coded using the code words H + M(a_0a_1) + M(a_1a_2). H is the flag code word 001 taken from the two-dimensional code table (Table 1/T.6). M(a_0a_1) and M(a_1a_2) are code words which represent the length and "colour" of the runs a_0a_1 and a_1a_2 respectively and are taken from the appropriate white or black run-length code tables (Tables 2/T.6 and 3/T.6).

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

TABLE 1/T.6

Code table

Mode	Elements to be coded		Notation	Code word
Pass	b_1, b_2		P	0001
Horizontal	a_0a_1, a_1a_2		H	001 + M(a_0a_1) + M(a_1a_2) (see Note)
Vertical	a_1 just under b_1	$a_1b_1 = 0$	V(0)	1
	a_1 to the right of b_1	$a_1b_1 = 1$	$V_R(1)$	011
		$a_1b_1 = 2$	$V_R(2)$	000011
		$a_1b_1 = 3$	$V_R(3)$	0000011
	a_1 to the left of b_1	$a_1b_1 = 1$	$V_L(1)$	010
		$a_1b_1 = 2$	$V_L(2)$	000010
		$a_1b_1 = 3$	$V_L(3)$	0000010
Extension				0000001xxx

Note – Code M( ) of the horizontal mode represents the code words in Tables 2/T.6 and 3/T.6.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

TABLE 2/T.6

Terminating codes

White run length	Code word	Black run length	Code word
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	00001100111
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000011001010
27	0100100	27	000011001011
28	0011000	28	000011001100
29	00000010	29	000011001101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	00010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011010011
36	00010101	36	000011010100
37	00010110	37	000011010101
38	00010111	38	000011010110
39	00101000	39	000011010111
40	00101001	40	000001101100
41	00101010	41	000001101101
42	00101011	42	000011011010
43	00101100	43	000011011011
44	00101101	44	000001010100
45	00000100	45	000001010101
46	00000101	46	000001010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101
50	01010011	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111
54	00100101	54	000000111000
55	01011000	55	000000100111
56	01011001	56	000000101000
57	01011010	57	000001011000
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

TABLE 3/T.6

Make-up codes between 64 and 1728

White run length	Code word	Black run length	Code word
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	0000001101100
576	01101000	576	0000001101101
640	01100111	640	0000001001010
704	011001100	704	0000001001011
768	011001101	768	0000001001100
832	011010010	832	0000001001101
896	011010011	896	0000001110010
960	011010100	960	0000001110011
1024	011010101	1024	0000001110100
1088	011010110	1088	0000001110101
1152	011010111	1152	0000001110110
1216	011011000	1216	0000001110111
1280	011011001	1280	0000001010010
1344	011011010	1344	0000001010011
1408	011011011	1408	0000001010100
1472	010011000	1472	0000001010101
1536	010011001	1536	0000001011010
1600	010011010	1600	0000001011011
1664	011000	1664	0000001100100
1728	010011011	1728	0000001100101

Make-up codes between 1792 and 2560

Run length (black and white)	Make-up codes
1792	00000001000
1856	00000001100
1920	00000001101
1984	000000010010
2048	000000010011
2112	000000010100
2176	000000010101
2240	000000010110
2304	000000010111
2368	000000011100
2432	000000011101
2496	000000011110
2560	000000011111

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

2.2.4 *Coding procedure*

The coding procedure identifies the coding mode that is to be used to code each changing element along the coding line. When one of the three coding modes has been identified according to Step 1 or Step 2 mentioned below, an appropriate code word is selected from the code table given in Table 1/T.6. The coding procedure is as shown in the flow diagram of Figure 5/T.6.

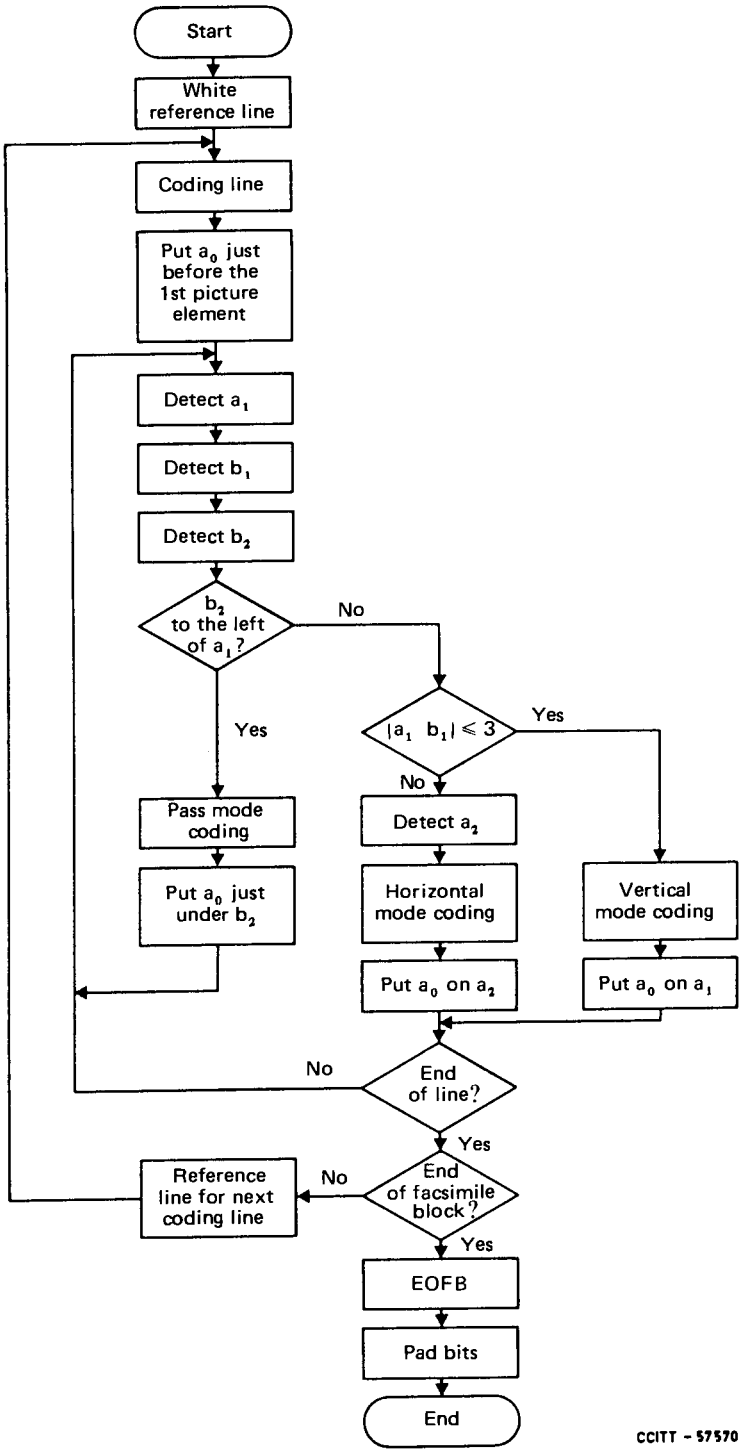


FIGURE 5/T.6

Coding flow diagram

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Step 1

i) If a pass mode is identified, this is coded using the word 0001 (Table 1/T.6). After this processing, picture element a'_0 just under b_2 is regarded as the new starting picture element a_0 for the next coding (see Figure 2/T.6).

ii) If a pass mode is not detected, then proceed to Step 2.

Note – It does not affect compatibility to restrict the use of pass mode in the encoder to a single pass mode. Variations of the algorithm which do not affect compatibility should be the subject of further study.

Step 2

i) Determine the absolute value of the relative distance a_1b_1.

ii) If $| a_1b_1 | \leq 3$, as shown in Table 1/T.6, a_1b_1 is coded by the vertical mode, after which position a_1 is regarded as the new starting picture element a_0 for the next coding.

iii) If $| a_1b_1 | > 3$, as shown in Table 1/T.6, following horizontal mode code 001, a_0a_1 and a_1a_2 are respectively coded by one-dimensional run length coding.

Run lengths in the range of 0 to 63 pels are encoded with their appropriate terminating code word of Table 2/T.6. Note that there is a different list of code words for black and white run lengths. Run lengths in the range of 64 to 2623 pels are encoded first by the make-up code word representing the run length which is nearest, not longer, to that required. This is then followed by the terminating code word representing the difference between the required run length and the run length represented by the make-up code. Run lengths in the range of lengths longer than or equal to 2624 pels are coded first by the make-up code of 2560. If the remaining part of the run (after the first make-up code of 2560) is 2560 pels or greater, additional make-up code(s) of 2560 are issued until the remaining part of the run becomes less than 2560 pels. Then the remaining part of the run is encoded by terminating code or by make-up code plus Terminating code according to the range as mentioned above.

After this processing, position a_2 is regarded as the new starting picture element a_0 for the next coding.

Note – Coding examples are given in Recommendation T.4, § 4.2.5.

2.2.5 *Processing the first and last picture element in a line*

2.2.5.1 *Processing the first picture element*

The first starting picture element a_0 on each coding line is imaginarily set at a position just before the first picture element, and is regarded as a white picture element (see § 2.2.2).

The first run length on a line a_0a_1 is replaced by $a_0a_1 - 1$. Therefore, if the first actual run is black and is deemed to be coded by horizontal mode coding, then the first code word $M(a_0a_1)$ corresponds to an imaginary white run of zero length (see Figure 10/T.4).

2.2.5.2 *Processing the last picture element*

The coding of the coding line continues until the position of the imaginary changing element situated just after the last actual element has been coded. This may be coded as a_1 or a_2. Also, if b_1 and/or b_2 are not detected at any time during the coding of the line, they are positioned on the imaginary changing element situated just after the last actual picture element on the reference line.

2.3 *Optional facsimile coding schemes for black and white images*

2.3.1 *Uncompressed mode*

Uncompressed mode is an optional coding scheme associated to the basic facsimile coding scheme and is used to transmit the image information without data compression techniques as shown in Table 4/T.6.

The extension code in § 2.2.4 with the xxx bits set to 111 is used as an entrance code from the basic coding scheme in § 2.2 to the uncompressed mode.

While using the uncompressed mode, the last picture elements of the end of the line and the first picture elements of the beginning of the following line are concatenated to one pattern.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

TABLE 4/T.6

Uncompressed mode code words

Entrance code to uncompressed mode	Basic coding scheme: 0000001111	
	Image pattern	*Code word*
Uncompressed mode code	1 01 001 0001 00001 00000	1 01 001 0001 00001 000001
Exit from uncompressed mode code	0 00 000 0000	0000001T 00000001T 000000001T 0000000001T 00000000001T

T denotes a tag bit which tells the colour of the next run (black = 1, white = 0).

2.4 *Facsimile coding control functions*

2.4.1 *Control functions for basic facsimile coding scheme*

2.4.1.1 *End-of-facsimile block*

The end-of-facsimile block (EOFB) code is added to the end of every coded facsimile block. The format of EOFB is as follows:

Format: 000000000001000000000001

24 bits

2.4.1.2 *Pad bits*

Pad bits may be used after the end-of-facsimile block code if it is necessary to align on octet boundaries or to a fixed block size. The format used is as follows.

Format: Variable length string of 0s.

2.4.1.3 *Extension*

Extension code is used to indicate the change from the current mode to another mode, e.g., another coding scheme.

Format: 0000001xxx,

where xxx = 111 indicates uncompressed mode which is specified in § 2.3.1.

Further study is needed to define other unspecified xxx bit assignments and their use for any further extensions.

3 Optional grey scale facsimile coding schemes and their coding control functions

For further study.

4 Optional colour facsimile coding schemes and their coding control functions

For further study.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

US005956467A

United States Patent [19]

Rabbani et al.

[11] **Patent Number: 5,956,467**

[45] **Date of Patent: Sep. 21, 1999**

[54] **ENCODING COLOR IMAGE DATA FOR MULTIPASS COLOR PRINTERS**

[75] Inventors: **Majid Rabbani**, Pittsford; **Chris W. Honsinger**, Webster, both of N.Y.

[73] Assignee: **Eastman Kodak Company**, Rochester, N.Y.

[21] Appl. No.: **08/767,613**

[22] Filed: **Dec. 17, 1996**

[51] **Int. Cl.**[6] **H04N 1/64**

[52] **U.S. Cl.** **395/109**; 382/166; 382/236; 382/238; 358/539

[58] **Field of Search** 395/109, 104, 395/114, 116, 115; 382/238, 236, 166, 232, 250, 248; 358/539, 430, 432, 433

[56] **References Cited**

U.S. PATENT DOCUMENTS

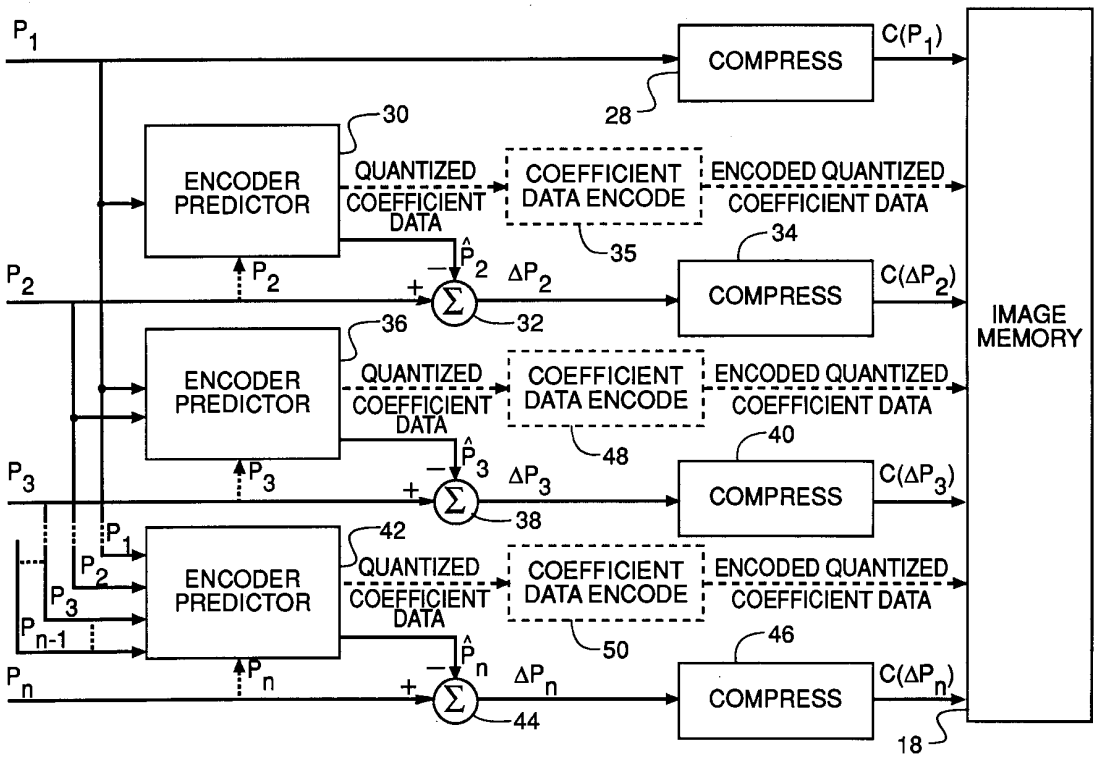
4,809,350	2/1989	Shimoni	382/56
5,539,842	7/1996	Schwartz	382/232
5,663,764	9/1997	Kondo et al.	348/414
5,761,346	6/1998	Moody	382/254

Primary Examiner—Scott Rogers
Attorney, Agent, or Firm—Thomas H. Close

[57] **ABSTRACT**

A method for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence includes the steps of: compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data; using information relating to the image data in the previous color plane(s) to predict the image data in the next color plane to be printed; forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane; and compressing the difference image data and storing the resulting compressed data. These steps are repeated until the image data for all of the color planes have been compressed and stored.

16 Claims, 6 Drawing Sheets



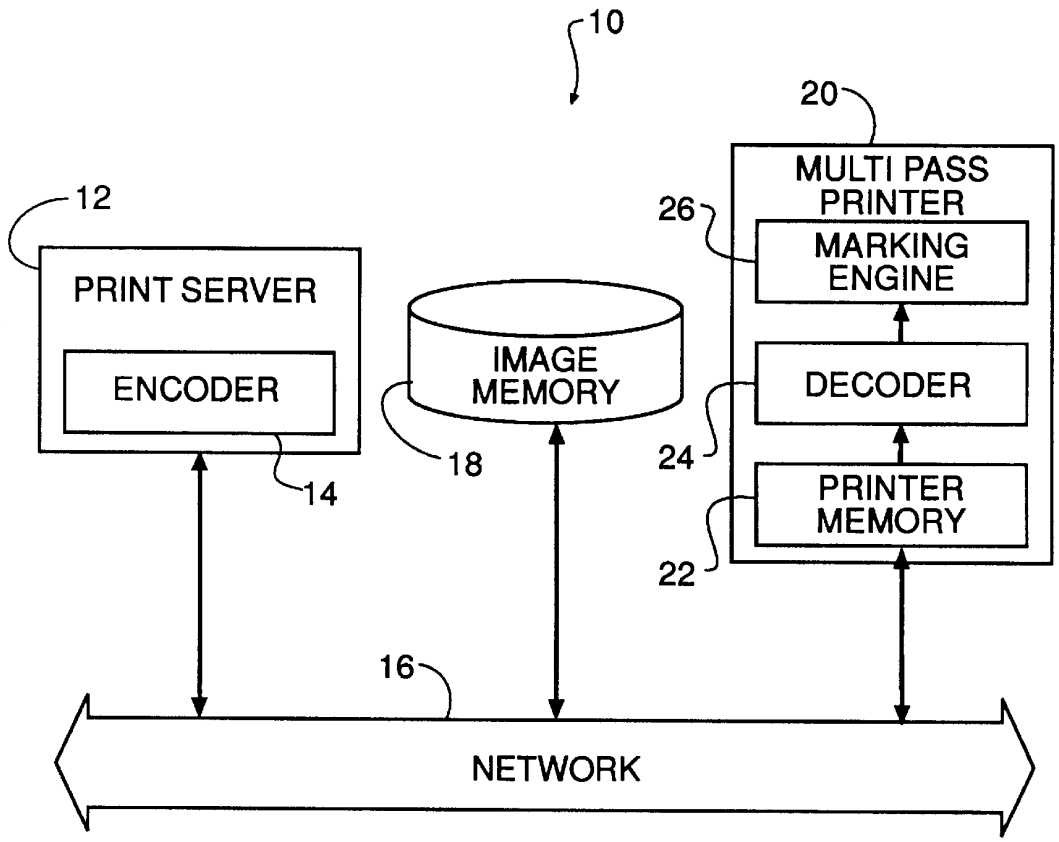

10
20
MULTI PASS PRINTER
26
MARKING ENGINE
12
PRINT SERVER
ENCODER
14
IMAGE MEMORY
18
DECODER
24
PRINTER MEMORY
22
16
NETWORK

FIG. 1

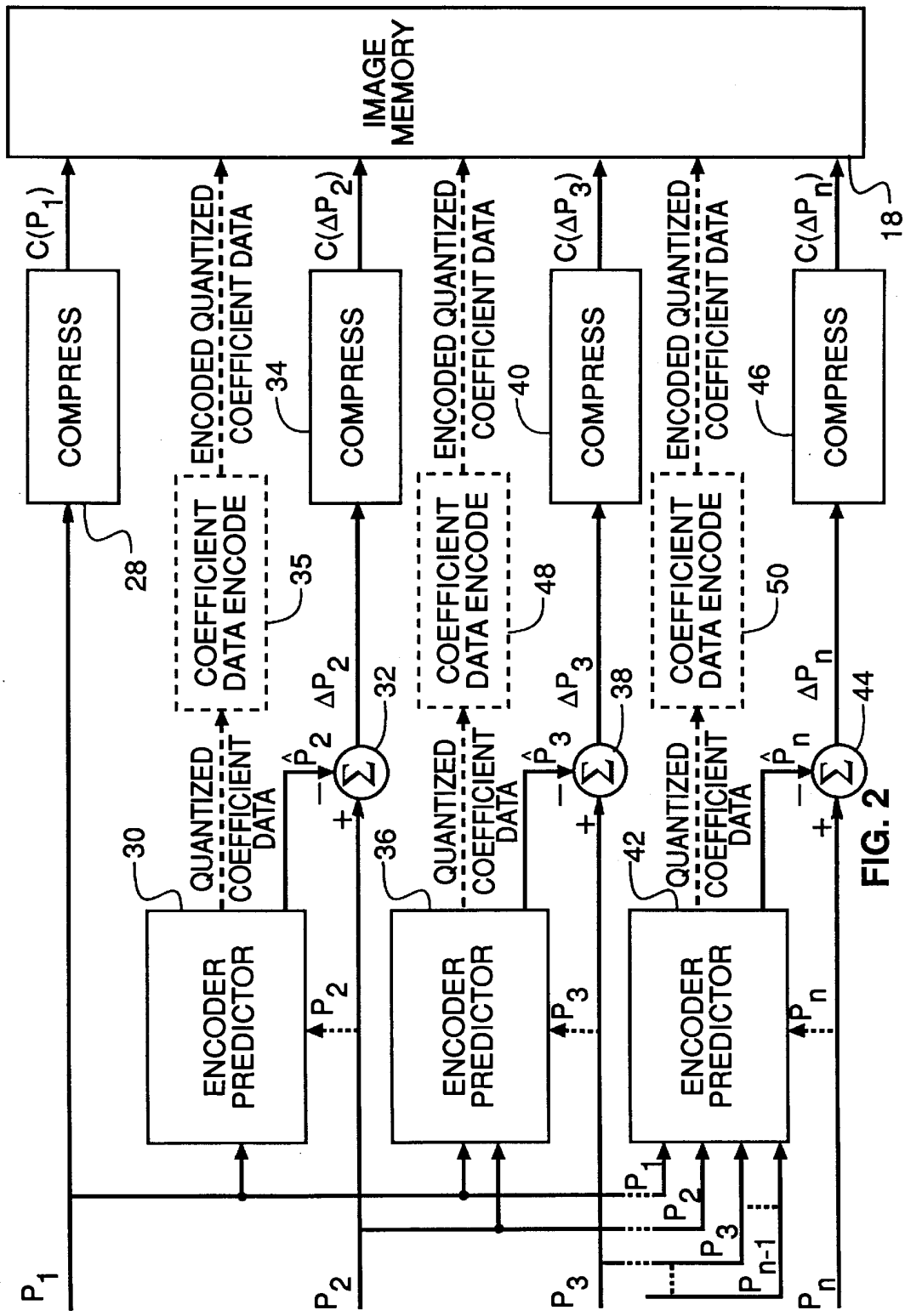

IMAGE MEMORY
18
C(P1)
COMPRESS
28
ENCODED QUANTIZED COEFFICIENT DATA
34
C(ΔP2)
COMPRESS
ENCODED QUANTIZED COEFFICIENT DATA
40
C(ΔP3)
COMPRESS
ENCODED QUANTIZED COEFFICIENT DATA
46
C(ΔPn)
COMPRESS
COEFFICIENT DATA ENCODE
35
COEFFICIENT DATA ENCODE
48
COEFFICIENT DATA ENCODE
50
ΔP2
32
ΔP3
38
ΔPn
44
QUANTIZED COEFFICIENT DATA
30
QUANTIZED COEFFICIENT DATA
36
QUANTIZED COEFFICIENT DATA
42
ENCODER PREDICTOR
ENCODER PREDICTOR
ENCODER PREDICTOR
P1
P2
P3
Pn
P1
P2
P3
Pn-1
Pn

FIG. 2

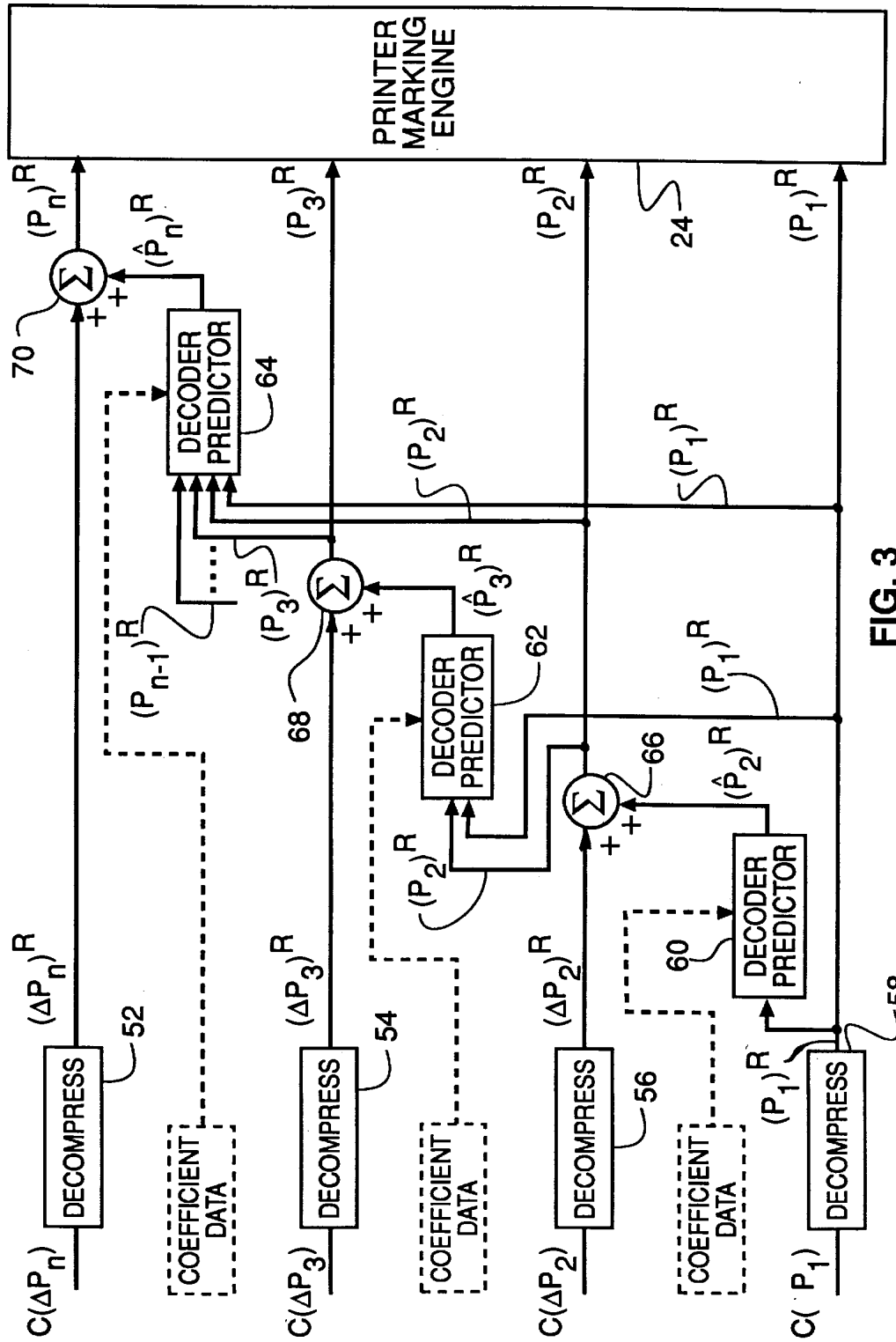

PRINTER MARKING ENGINE
24
$(P_n)^R$
$(\hat{P}_n)^R$
$(P_3)^R$
$(P_2)^R$
$(P_1)^R$
70
DECODER PREDICTOR
64
$(P_{n-1})^R$
68
62
66
60
58
$(\Delta P_n)^R$
$(\Delta P_3)^R$
$(\Delta P_2)^R$
52
54
56
DECOMPRESS
COEFFICIENT DATA
$C(\Delta P_n)$
$C(\Delta P_3)$
$C(\Delta P_2)$
$C(P_1)$

FIG. 3

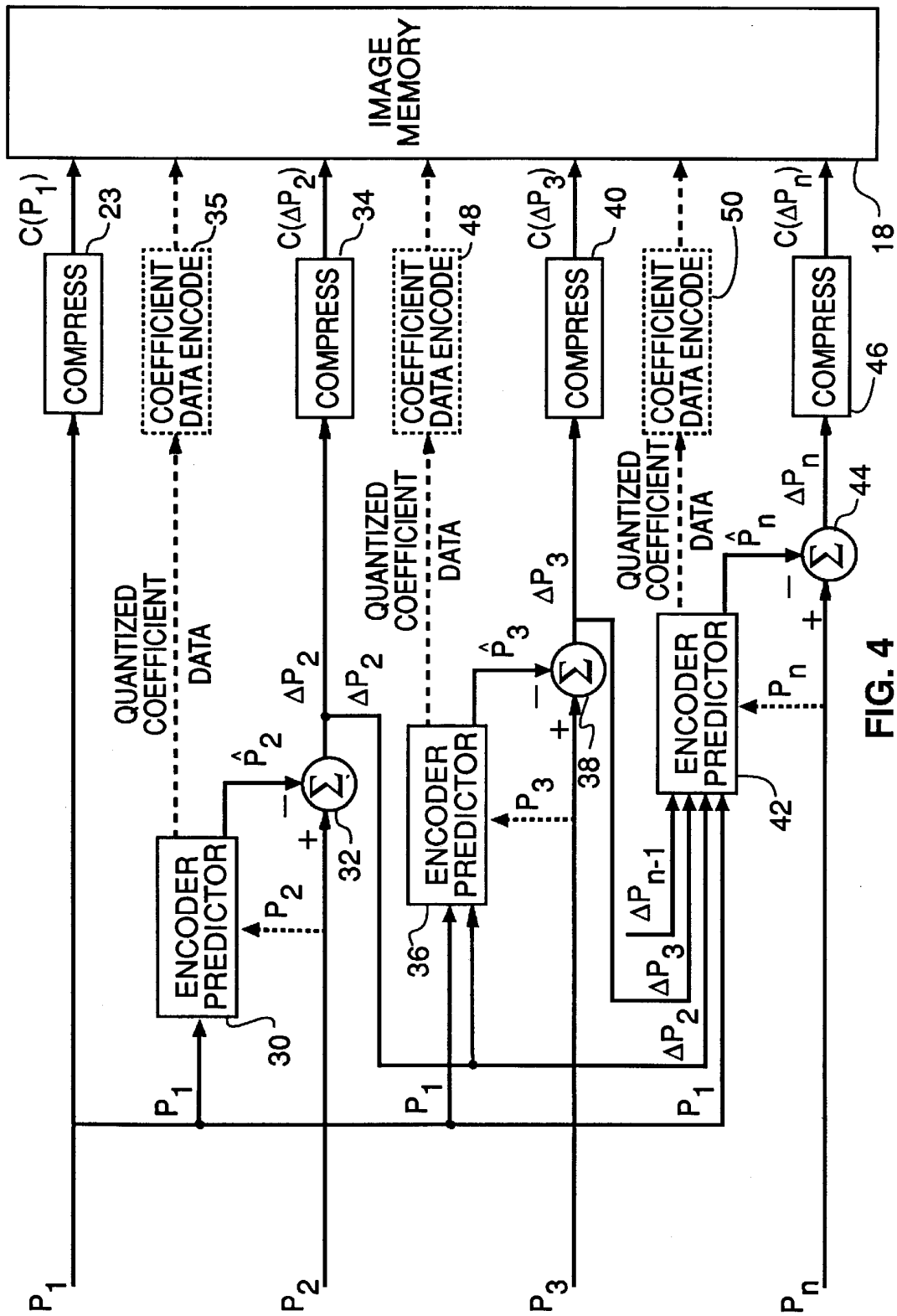

IMAGE MEMORY
18
C(P1)
23
COMPRESS
COEFFICIENT DATA ENCODE
35
QUANTIZED COEFFICIENT DATA
C(ΔP2)
34
COMPRESS
COEFFICIENT DATA ENCODE
48
QUANTIZED COEFFICIENT DATA
C(ΔP3)
40
COMPRESS
COEFFICIENT DATA ENCODE
50
QUANTIZED COEFFICIENT DATA
C(ΔPn)
COMPRESS
46
ENCODER PREDICTOR
30
ENCODER PREDICTOR
36
ENCODER PREDICTOR
42
P1
P2
P3
Pn
ΔP2
ΔP3
ΔPn
ΔPn-1
32
38
44
+
−
Σ

FIG. 4

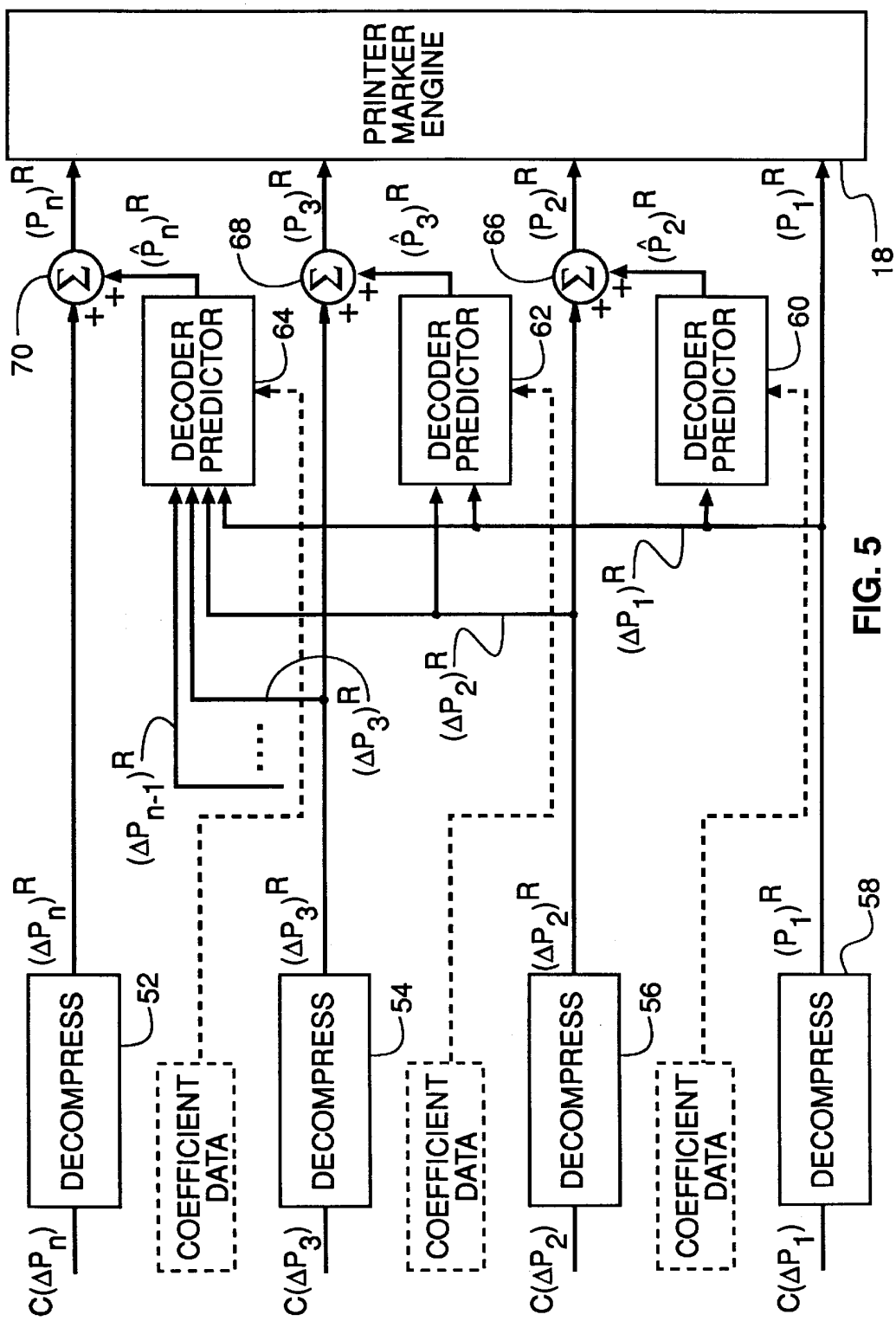

PRINTER MARKER ENGINE
18
$(P_n)^R$
$(\hat{P}_n)^R$
$(P_3)^R$
$(\hat{P}_3)^R$
$(P_2)^R$
$(\hat{P}_2)^R$
$(P_1)^R$
70
68
66
64
62
60
DECODER PREDICTOR
DECODER PREDICTOR
DECODER PREDICTOR
$(\Delta P_1)^R$
$(\Delta P_2)^R$
$(\Delta P_3)^R$
$(\Delta P_{n-1})^R$
$(\Delta P_n)^R$
$(\Delta P_3)^R$
$(\Delta P_2)^R$
$(P_1)^R$
52
54
56
58
DECOMPRESS
DECOMPRESS
DECOMPRESS
DECOMPRESS
COEFFICIENT DATA
COEFFICIENT DATA
COEFFICIENT DATA
$C(\Delta P_n)$
$C(\Delta P_3)$
$C(\Delta P_2)$
$C(\Delta P_1)$

FIG. 5

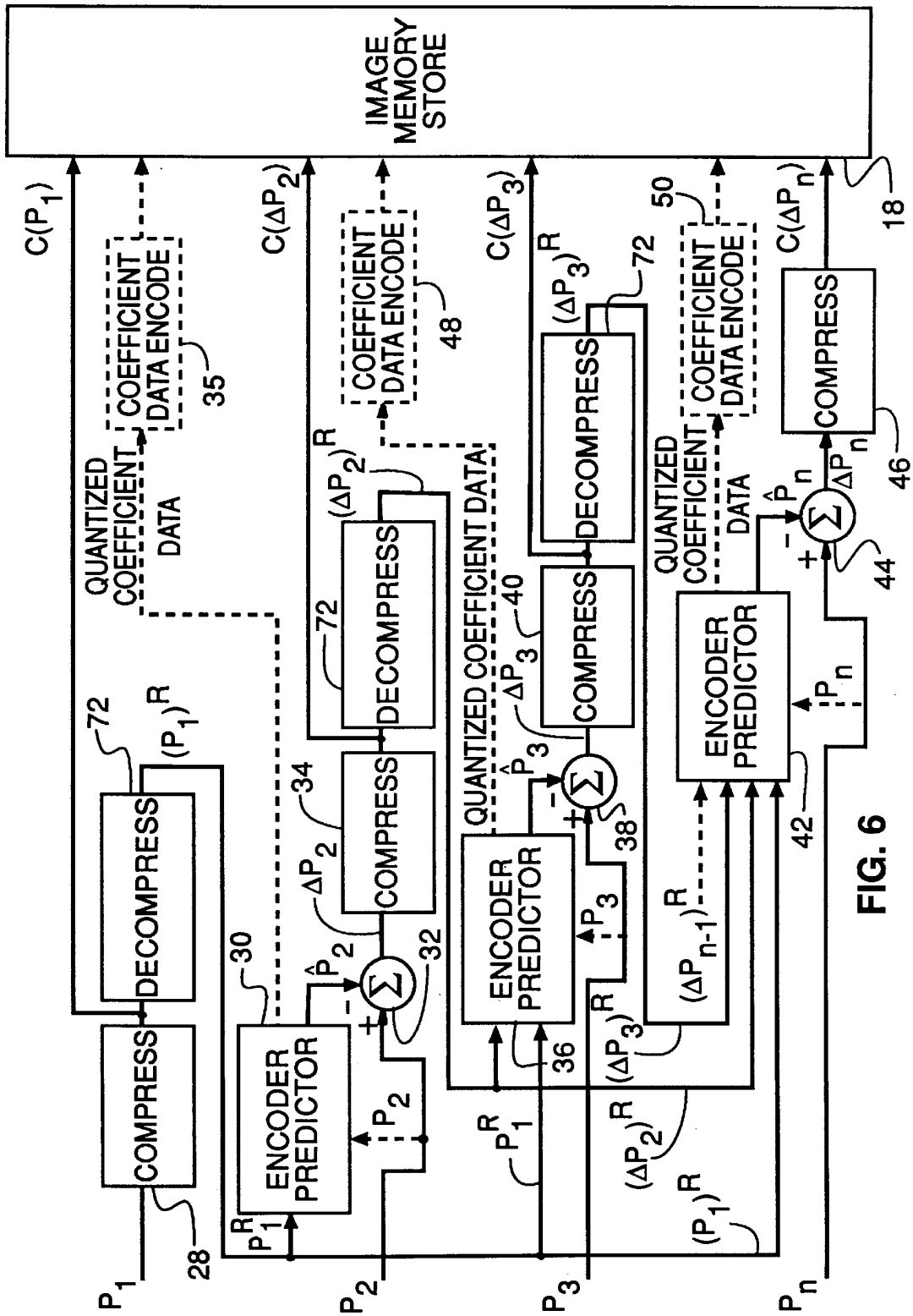

IMAGE MEMORY STORE
C(P1)
C(ΔP2)
C(ΔP3)
C(ΔPn)
QUANTIZED COEFFICIENT DATA
COEFFICIENT DATA ENCODE
35
48
50
18
72
DECOMPRESS
COMPRESS
ENCODER PREDICTOR
30
34
40
42
44
46
32
36
38
28
P1
P2
P3
Pn

FIG. 6

ENCODING COLOR IMAGE DATA FOR MULTIPASS COLOR PRINTERS

APPENDIX

The disclosure in the appendix of this patent document contains material to which a claim of copyright protection is made. The copyright owner has no objection to the facsimile reproduction of any one of the patent documents or the patent disclosure as it appears in the U. S. Patent and Trademark Office patent file or records, but reserves all other rights whatsoever.

FIELD OF THE INVENTION

The invention relates generally to the field of electronic imaging, and in particular to encoding of color digital images that are to be printed by multipass digital color printers.

BACKGROUND OF THE INVENTION

In the practice of electronic color imaging it is often desirable to print a digital color image using a digital color printer such as a thermal printer. The color image may be represented as a set of three color planes, usually red, green, and blue (RGB). Alternatively, it may be represented as the set of four color planes, usually cyan, magenta, yellow and black (CMYK). The color printer usually prints the image by printing one color plane at a time in a prescribed order, e.g., in a four color system, the cyan plane may be printed first, the magenta next, followed by the yellow and finally the black plane.

Two classes of memory configurations are common for color printers. The first class applies to low cost printers. Here only a stripe (subsection) of a single color plane is employed. When the printer completes the printing of this stripe, the printer waits until it receives more data. There are three concerns with this type of configuration. The first is that the server (i.e. the computer asking the printer for service) must buffer the data while the printer is printing. This can consume large amounts of disk space and slow down the computer response time due to the system interruptions necessary to manage this buffer. Second, the network line to the printer will be busy throughout the printing process, which causes a prolonged delay for other users who are sharing the network. Finally, delays in the network connection can cause a delay in providing the image data to the printer. This requires a print head that is capable of stopping and resuming in the middle of the printing process, creating additional engineering and hardware costs that are not desirable.

The second class of printers includes higher end color printers that contain sufficient local memory for storing the entire image at the time of printing. In general, given the high price of solid state memory (e.g., SRAM and DRAM), it is desirable to keep the printer memory requirements to a minimum. Also, in order to expedite the data transfer process for the printing of the next image, it is desirable to discard the data related to each color plane of the current image once it has been printed, so that the memory thus made available can be used to download the color planes of the next image in the printer queue.

In order to satisfy these requirements, in the prior art, each color plane of the image to be printed is compressed independently using a compression scheme such as the lossy or the lossless JPEG compression standard. The JPEG compression standard is explained in detail in ITU Recommendations T.81, T.83 and T.84, or alternatively, in ISO/IEC International Standards, 10918-1, 10918-2, and 10918-3.

It is well known that, in general, there exists a significant correlation between the various color planes of a color image. Consequently, to increase the amount of compression achieved on a color image, this correlation should be reduced to the extent possible prior to the compression process. A conventional method of reducing the correlation among the color planes is to apply a linear decorrelating transformation to the color planes. This is sometimes referred to also as a color space rotation. For example, if the color image is represented as RGB tri-stimulus, it can be transformed into another color space representation such as the YUV luminance-chrominance space. Many different standard and proprietary color space transformations exist and the equations for such transformations have been documented in the literature.

Although applying a conventional color space transformation to the color image data prior to compression (e.g., with the JPEG standard) increases the amount of compression achieved in printers, it does not allow the data that corresponds to a certain color plane to be discarded after that color plane has been printed. As an example, consider the YUV luminance-chrominance color space representation of an RGB color image. Further, assume that the order in which the color planes are printed is blue, red, and green, respectively, and that the decompression and printing of each color plane is done in stripes. The first step in this process would be to decompress a stripe of the compressed YUV color planes. Next, a stripe of the blue color plane would be reconstructed from the decompressed YUV stripes and printed. Since the same YUV color planes are needed in order to also reconstruct the red and the green color planes, no stripes of the compressed data can be discarded until the whole color image has been printed. That is why in prior art each RGB (or CMYK) color plane is independently compressed and then decompressed at the time of printing.

There is a need therefore to improve the performance of image compression in this second class of printers to minimize the cost of the printer memory.

SUMMARY OF THE INVENTION

The present invention is aimed at reducing the memory requirements on board a multipass printer, while allowing the printer to discard the color plane image data after it has been printed. Briefly summarized, according to one aspect of the present invention, a method for encoding color image data of the type having multiple color planes for a multipass color printer that prints the color planes in a given sequence is disclosed. The method includes the steps of: compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data; using information relating to the image data in the previous color plane(s) to predict the image data in the next color plane to be printed; forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane; and compressing the difference image data and storing the resulting compressed data. These steps are repeated until the image data for all of the color planes have been compressed and stored.

ADVANTAGEOUS EFFECT OF THE INVENTION

The present invention has the advantage of reducing memory requirements on board a multipass printer, while allowing the printer to discard the color plane image data after it has been printed.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is schematic diagram of a printing system including a multi-pass printer useful with the present invention

FIG. **2** is a schematic diagram of the encoder described in a preferred embodiment of this invention

FIG. **3** is a schematic diagram of the decoder described in a preferred embodiment of this invention

FIG. **4** is a schematic diagram of the encoder described in an alternative embodiment of this invention

FIG. **5** is a schematic diagram of the decoder described in an alternative embodiment of this invention

FIG. **6** is a schematic diagram of the encoder described in another alternative embodiment of this invention

To facilitate understanding, identical reference numerals have been used, where possible, to designate identical elements that are common to the figures.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. **1**, a printing system **10** useful with the present invention includes a print server **12**, for example, a suitably programmed personal computer, having an encoder **14** for enoding a color image according to the present invention. The print server communicates over a network server **16**, with an image memory **18**, and a multi-pass printer **20**, for example, a Kodak XL8600 thermal printer. The multi-pass printer **20** includes a printer memory **22** for the temporary storage of the color image from the image memory **18**, a decoder **24** for decoding the color image signal stored in printer memory **22**, and a marking engine **26** for printing the color image signal. At the time of printing, the decoder **24** decodes the data residing in the printer memory **22** to reconstruct the color planes of the color image and prints them in a prescribed order.

The present invention improves the image compression efficiency and hence reduces the cost of memory in digital color printers by reducing the correlation among the color planes prior to compression. This correlation reduction is performed in such a way that at the decoder, once a given color plane has been printed, the related encoded image data can be deleted from the memory to accommodate the loading of the data corresponding to the next image in the printer queue.

For the purpose of describing the encoding and the decoding procedures according to the present invention, assume that the color image consists of n color planes denoted by P_i, i=1,2, . . . , n. Further, without any loss of generality, assume that the printer requires these color planes to be printed in the following sequence of P_n, P_{n-1}, . . . , P_1. For example, a color image can consist of four components, cyan, magenta, yellow, and black, denoted by CMYK, and the order of the printing can be cyan, followed by magenta, yellow, and black, respectively. Although the present invention applies to any choice of compression schemes, it is convenient to use the JPEG compression standard to compress the image data. The JPEG compression standard compresses the data by dividing it into 8×8 blocks and compressing each block separately. As a result, the JPEG compression in addition to being an international standard, is also suitable for compressing the image data in stripes. The JPEG compression can provide either lossless compression or lossy compression. In lossless compression, the decompressed image data are mathematically identical to the original image data, while in lossy compression, the decompressed image data are different from the original image data

where the extent of this difference depends on the degree of compression achieved. Since lossless compression typically results in only a modest amount of compression (usually around 2:1), lossy compression is often preferred in printer applications. With JPEG compression, the amount of loss and the degree of compression can be controlled by changing the elements of an 8×8 array called the quantization table. In general, a lossy compression scheme such as the JPEG standard includes three steps: transformation or decomposition, quantization, and symbol encoding. The transformation step reduces the correlation or the redundancy in the data to make it more amenable to compression. The quantization step, which is where the loss in lossy compression is introduced, maps ranges of input values into a single quantized value and thus reduces the size of the data to be encoded. Finally, the symbol encoding step translates the quantizer output values into a string of bits (binary "1" and "0" digits).

According to the present invention, color correlation reduction is achieved by ensuring that the redundant information in a given color plane is reduced by using only the color plane information available at the print buffer at the time of printing. In general, compression performance is improved over prior art since the information present in a given color plane that is also contained in the succeeding color planes in the order of printing is reduced. Either lossy or lossless compression algorithms may be chosen.

The encoding and decoding of the color plane data need not be performed using the entire color plane. Sub-regions of the color planes such as stripes or small rectangles may be used to perform the operations. Breaking the color plane into sub-regions has the advantage of requiring less memory and for certain embodiments, improved compression performance.

In a preferred embodiment of the present invention as shown in FIG. **2**, the color plane that will be printed last, namely, P_1, is first compressed **28** independent of any other color plane and is stored in the image memory **18**. For the encoding of the color plane P_2, first the correlation between that color plane and all the color planes that have been previously encoded, namely P_1 in this case, is reduced. In particular, the previous color planes are used to form a prediction of P_2 denoted by $\hat{P}_2$ in encoder predictor **30**, which is then subtracted **32** from P_2 to form a difference image ΔP_2. This difference image ΔP_2 is compressed **34** and stored in image memory.

The prediction used at the encoder predictor **30** can be either linear or nonlinear as will be described later. Furthermore, the prediction can be either causal or noncausal. In causal predictors, only the data that has been encoded prior to the encoding of the current data is used for prediction. Since the decoder also has access to all the past data used by the encoder for prediction, the causal prediction operation can be exactly repeated at the decoder. An example of causal prediction is when a previous color plane, e.g., P_1, is used as the prediction for the next color plane to be encoded, e.g., P_2. In this case, the difference image is simply the pixel by pixel difference between the color planes P_1 and P_2.

On the other hand, in noncausal prediction both the information contained in the previously encoded data and the information that has not yet been encoded may be used to form the prediction. Since the noncausal information is not available at the decoder, the decoder has to be instructed as to how to construct the exact same prediction. This is usually accomplished by including additional information,

i.e. overhead information, along with the prediction. As an example of noncausal prediction, consider finding the optimum linear prediction of the color plane P_2 using the color plane P_1. This procedure, which will be described in detail later, uses the color plane P_2 in addition to the color plane P_1. As a result, the encoder appends the prediction coefficient data to the prediction as overhead. At the decoder, this overhead information is used to construct the exact same prediction that was used at the encoder. In FIG. **2**, it is assumed that the noncausal information, if used at all by the encoder predictor **30**, is the actual color plane which is shown as a dotted line input to the predictor. Also, the overhead information resulting from noncausal prediction is shown with dotted lines at the output of the encoder predictor **30** and is referred to herein as the "quantized coefficient data". This quantized coefficient data may then be further compressed in the encoder **35** prior to storing in image memory **18**.

Continuing in the same vein for the encoding of the other color planes, when encoding color plane P_3, first the correlation between that color plane and all the color planes that have been previously encoded, namely P_1 and P_2 in this case, is reduced in encoder predictor **36**. In particular, the previous color planes (in addition to the current color plane in the case of noncausal prediction) are used to form a prediction of P_3 denoted by $\hat{P}_3$, which is then subtracted **38** from P_3 to form the difference image ΔP_3. This difference image is compressed **40** and stored in image memory **18**. This process is continued until all of the color planes have been compressed and stored in image memory **18**, as shown generally by encoder predictor **42**, difference operator **44**, and compressor **46**. Similarly, coefficient data resulting from predictors **36** and **42** may be further compressed in encoders **48** and **50** respectively.

It should be noted that in the general case shown in FIG. **2**, any or all combinations of the previous color planes may be used in order to predict the color plane that is currently being processed. However, for practical reasons, in many situations only a single color plane (e.g., the one printed last) or a limited number of previous color planes are actually used to predict the current color plane. Our findings indicate that even using a single color plane for prediction in many cases results in a significant improvement in compression efficiency.

The decoding is performed at the printer by simply reversing the above process as is shown in FIG. **3**. At the decoder the color planes can be recovered from the compressed data in the desired order for printing while the compressed data corresponding to a particular color plane can be discarded after that color plane has been printed. To reconstruct a stripe of the first color plane to be printed P_n, the stripe of compressed color data $C(\Delta P_n)$ is decompressed **52**, and a corresponding stripe of all of the previously compressed data $C(P_1)$, $C(\Delta P_2)$, . . . , $C(\Delta P_{n-1})$ is first decompressed **54**, **56**, and **58**. Since in most cases the compression scheme used is a lossy compression scheme, the decompressed values will not be exactly the same as the original values prior to compression. As noted above, the degree to which the decompressed values approximate the original values depends on the amount of quantization employed during the compression process. In our notation, we refer to the decompressed image values as the reconstructed values and we denote them by the superscript R. For example, if the compressed color plane P_1 is represented by $C(P_1)$, the decompressed (reconstructed) color plane P_1 is denoted by $(P_1)^R$. Obviously, if lossless compression has been used, then $(P_1)^R$ is the same as P_1.

In reference to FIG. **3**, to reconstruct the ith color plane, first the reconstructed values of the previous color planes are used to form a prediction of that color plane, $(\hat{P}_i)^R$, using the decoder predictors **60**, **62** and **64** consecutively. Additionally, in the case of noncausal prediction the overhead information labeled as "quantized coefficient data" (the dotted lines in FIG. **3**) is also used in constructing the prediction $(\hat{P}_i)^R$. This prediction is then added **66**, **68**, and **70**, consecutively, to the reconstructed difference image for that color plane, $(\Delta P_i)^R$, to yield the reconstructed value of that color plane denoted by $(P_i)^R$. The reconstructed color plane data for the ith color plane $(P_i)^R$ is then sent to the printer marking engine **24** for printing, after which it may be purged from memory. Similarly, the compressed data $C(\Delta P_i)$, corresponding to the ith color plane just printed can be purged from memory.

As noted above, for practical reasons, in most cases only a single color plane or a limited number of previous color planes are actually used to predict the current color plane. For example, our findings indicate that by using a single color plane, e.g., P_1, to predict all of the other color planes, the operations of both the encoder and the decoder are significantly simplified while the compression efficiency is still significantly improved compared to the prior art.

In an alternative embodiment of the present invention as described in FIG. **4**, the color plane that will be printed last, namely, P_1, is first compressed **28** independent of any other color plane and is stored in the image memory **18**. Next, the color plane P_1 is used to form a prediction **30** of the next color plane P_2 denoted by $\hat{P}_2$, which is then subtracted **32** from P_2 to form the difference image ΔP_2. This difference image is compressed **34** and stored in image memory **18**. As noted before, the prediction can either be linear or nonlinear or it can be either causal or noncausal. In general, for the encoding of any given color plane P_i, the last color plane to be printed P_1 and all of the color plane differences formed up to that stage, i.e., $\Delta P_2, \ldots, \Delta P_{i-1}$, are used to predict that color plane. The difference image ΔP_i formed by subtracting the prediction image values $\tilde{P}_i$ from the original image values P_i is compressed and stored. This process is continued until all of the color planes have been compressed and stored. The main difference between this embodiment and the one described in FIG. **2** is that in predicting a given color plane P_i, the set of $P_1, \Delta P_2, \ldots, \Delta P_{i-1}$ values are used instead of the set $P_1, P_2, \ldots, P_{i-1}$. This generally results in a simpler encoder and decoder configuration. The decoder process is depicted in FIG. **5** and is simply the reverse of the operations performed at the encoder.

A potential limitation of the embodiments described in FIG. **2** and FIG. **4** is that the predictions computed at the encoder are based on the original color plane values whereas at the decoder, only the decompressed (reconstructed) values are available. If the compression process is lossless, the predictions calculated at the encoder which are based on the original data, and the predictions calculated at the decoder, which are based on the decompressed data, are identical. However, in most cases lossy compression is used in printers and the predictions made at the encoder and the decoder will be different. This problem is addressed in a further aspect of the present invention, where the compression efficiency of the technique of the invention is further improved by using at the encoder the same reconstructed values that are available at the decoder to construct the prediction. One way of achieving this is to decompress the compressed data at the encoder prior to forming the prediction. A short cut to this process can be introduced by recognizing the fact that the discrepancies between the original values and the recon-

structed (compressed/decompressed) values result from the quantizer in the lossy compression scheme. Thus, to access the reconstructed values, it is not necessary to decompress the compressed bit stream. Instead, during the process of compression, the output of the quantizer step can be used to produce the exact same data that will be reconstructed at the decoder.

According to a further aspect of the present invention as depicted in FIG. **6**, the color plane that will be printed last, namely, P_1, is first compressed independent of any other color plane and is stored in the memory. Next, the compressed color plane P_1 is reconstructed (e.g., decompressed **72**) and the reconstructed color plane $(P_1)^R$ is used to form a prediction **30** of P_2 denoted by, $\hat{P}_2$ which is then subtracted from P_2 **32** to form the difference image ΔP_2. As noted above, the reconstructed color plane $(P_1)^R$ can alternatively be produced from the output of the quantizer during the compression process. This difference image is compressed **34** and stored in image memory **18**. In general, for the encoding of any given color plane P_i, the reconstructed last color plane $(P_1)^R$ and all of the reconstructed color plane differences formed up to that stage, i.e., $(\Delta P_2)^R$, . . . , $(\Delta P_{i-1})^R$, are used to predict that color plane. The difference image ΔP_i formed by subtracting the prediction image values from the original image values is compressed and stored. This process is continued until all of the color planes have been compressed and stored. The decoder process, which is simply the reverse of the operations performed at the encoder, is the same as the previous embodiment described with reference to FIG. **5**.

The preferred linear predictors employed in encoder predictors **30**, **36** and **42** will now be described in further detail. As noted before, in one embodiment of the invention as described in FIG. **2**, the ith color plane is predicted as a function of the previously processed color planes. In the case of a linear predictor, the prediction is expressed as a linear weighting of the previous color planes as in the following equation:

$$\hat{P}_i=\phi_{i,1}P_1+\phi_{i,2}P_2+\ldots\phi_{i,i-1}P_{i-1},$$

where $\phi_{i,j}$ are constants. The above equation represents each pixel of the predicted color plane as a linear combination of the corresponding pixels in the previous color planes, where the weights for the color plane i are given by the set of constants $\{\phi_{i,j}\}$, j=1,2, . . . , i−1, also referred to as the predictor coefficients. If the predictor is causal, then the predictor coefficients are fixed for all images and are also known to the decoder. A simple but important example is using only the last color plane P_1 for prediction and using a single fixed predictor coefficient α. In this case the predictor simply becomes $\hat{P}_i=\alpha P_1$, where the value of α can be determined empirically for each color plane and can then be used for all images. When using noncausal predictors, the predictor coefficients can be optimized for each individual image and the particular color plane that is being predicted. Since these image-dependent predictor coefficients are not known to the decoder, they should be included as overhead information with the encoded file. Furthermore, instead of using the same set of predictor coefficients for the entire color plane, the color plane can be broken down into stripes or any other arbitrary shaped regions, and an optimum set of predictors can be found for each region which again is included as overhead information.

In the case of noncausal prediction, the optimum set of linear predictor coefficients can be found by minimizing the square of the prediction error $\Delta P_i \equiv \hat{P}_i - P_i$. In particular, it can be shown that for the ith color plane, the optimum set of i−1 coefficients, $\{\phi_{i,j}\}$, j=1,2, . . . , i−1, is the solution to the following set of i−1 equations:

$$\frac{\partial}{\partial \phi_{i,j}}\left[(\hat{P}_i - P_i)\bullet(\hat{P}_i - P_i)\right] = 0,\ j = 1, 2, \cdots i-1.$$

The operator "•" represents the "dot product" between the two images. The dot product between two images A and B, with dimensions of N_x pixels per line and N_y lines per image, is defined as the sum of the product of each pixel value from image A with the corresponding pixel value from image B, namely:

$$A \bullet B = \sum_{k=1}^{N_y}\sum_{l=1}^{N_x} A_{k,l}B_{k,l},$$

where, e.g., $A_{k,l}$ represents the pixel value at row k and column l of image A.

To better illustrate the noncausal prediction process, consider the encoder predictor **42** in FIG. **2** as an example of all the encoder predictors. The prediction part of the output of the encoder predictors is

$$\hat{P}_n=\phi_{n,1}P_1+\phi_{n,2}P_2+\ldots\phi_{n,n-1}P_{n-1},$$

where the n−1 weights are found by solving the following set of n−1 equations:

$$\frac{\partial[(P_n - \phi_{n,1}P_1 + \phi_{n,2}P_2 + \cdots \phi_{n,n-1}P_{n-1})\bullet (P_n - \phi_{n,1}P_1 + \phi_{n,2}P_2 + \cdots \phi_{n,n-1}P_{n-1})]}{\partial \phi_{n,j}} = 0,\ j = 1, 2, \cdots n-1.$$

Since the values of these weights are not available at the decoder, they need to be included as overhead information. This is accomplished by the dotted line output of encoder predictor labeled "quantized coefficient data". Finally, if the image has been broken into stripes or any other arbitrary shaped regions, for each region the above set of equations is solved and an optimum set of predictor coefficients is determined. In this case, the "quantized coefficient data" output of encoder predictor includes all the coefficient sets for the different regions in addition to any information needed to identify each coefficient set with its corresponding region or stripe.

In another embodiment of the present invention as depicted in FIG. **4**, the set of color difference data are used in the encoder predictors boxes **30**, **36** and **42** to predict the current color plane. Using the color difference data instead of the color plane data usually results in a simpler decoder. Recall that in all of the embodiments of the present invention, the compressed files are of the color difference data. As a result, instead of having to reconstruct all the previous color planes, the decoder simply uses the corresponding reconstructed color difference data as input to the decoder predictor boxes **60**, **62** and **64** of FIG. **5**. The linear predictor in this case is given by

$$\hat{P}_i=\theta_{i,1}P_1+\theta_{i,2}\Delta P_2+\ldots\theta_{i,i-1}\Delta P_{i-1},$$

where the i−1 coefficients, $\{\theta_{i,j}\}$, j=1,2, . . . , i−1, are constants. As noted before, for causal prediction, the above constants are image-independent and are fixed for each color plane. In the case of noncausal prediction, the minimum

means-squared-error optimum predictor coefficients are the solution to the following set of i−1 equations:

$$\frac{\partial}{\partial \theta_{i,j}}\left[\left(\hat{P}_i - P_i\right) \bullet \left(\hat{P}_i - P_i\right)\right] = 0,\ j = 1, 2, \cdots i-1.$$

It can be shown that solving the above equations would result in the following value for the weight $\theta_{i,j}$:

$$\theta_{ij} = \begin{cases} \dfrac{(\Delta P_i \bullet P_1)}{(P_1 \bullet P_1)}, & j = 1 \\ \dfrac{(\Delta P_i \bullet \Delta P_j)}{(\Delta P_i \bullet \Delta P_i)}, & j \neq 1 \end{cases} \quad \text{for } i \geq 2.$$

The above procedure is also referred to as the Gram-Schmidt orthogonalization process. In essence, the linear correlation between the color plane P_i and all the previous data, P_1, ΔP_2, . . . , ΔP_{i-1}, is removed by creating a color plane difference ΔP_i that is orthogonal to all the previous data, i.e.,

$$\Delta P_i \cdot P_1 = 0, \text{ and } \Delta P_i \cdot \Delta P_j = 0,\ 1 < j < i.$$

As before, the optimum set of coefficients needs to be included in the encoded file as overhead information. The coefficients are represented by line outputs from the encoder predictors **30**,**36**,**42** in FIG. **4**. Also, if the image has been broken into stripes or any other arbitrary shaped regions, for each region the above set of equations is solved and an optimum set of predictor coefficients is determined. It can be mathematically shown that when using optimum linear prediction, the predictions resulting from the corresponding encoder predictors depicted in FIG. **2** and FIG. **4** are identical. Since the decoder of FIG. **5** is much simpler to implement in practice as compared to the decoder of FIG. **3**, this latter embodiment is preferred in conjunction with using optimum linear predictors.

It should be noted that the coefficient data resulting from optimum linear prediction is generally expressed in floating point notation. In order to efficiently encode and store this data, it needs to be quantized to a finite number of bits. This quantization takes place in the encoder predictors **30**, **36** and **42**. The quantization can be either uniform or nonuniform. A preferred method of uniform quantization in this invention is the following

$$\phi_{i,j}{}^{q} = q \times \mathrm{NINT}\left(\frac{\phi_{i,j}}{q}\right),$$

where q is typically a negative power of two and is much smaller than one, e.g., $q=2^{-7}$, and the NINT operation denotes rounding to the nearest integer, e.g., NINT(3.2)=3, while NINT(3.8)=4. The q value is the granularity of the quantizer. Uniform quantization is simple to implement and has optimal properties when used in conjunction with variable length coding in the "coefficient data encode" unit **35**, **48**, **50**. On the other hand, if fixed-length coding is used to encode the coefficient data, it is advantageous to perform nonuniform quantization. We have observed that the coefficient data has a probability distribution that is closely approximated by a double-sided exponential distribution, also known as a Laplacian distribution. As a result, a preferred method of nonuniform quantization in this invention is a Lloyd-Max nonuniform minimum mean-squared-error (MMSE) quantizer designed for a Laplacian probability distribution.

It should be noted that the calculation of the coefficient data can be strongly influenced by the mean value of the image sub-regions. As a result, it is often advantageous to remove the mean value of each sub-region that is input to the encoder predictors **30**, **36**, **42** prior to the calculation of the coefficient data. This will tend to stabilize the distribution of the coefficient data towards a Laplacian and thus make it more amenable to quantization and encoding. However, this will also create a zero-mean prediction of the subregion, whereas the color plane being predicted usually has a zero mean. As a result, this will create a bias in the color plane difference data. This bias is encoded by the compressors **28**, **34**, **40** and **46**. Similarly, at the decoder, the mean value of the subregions input to the decoder predictors **60**, **62** and **64** are removed prior to the prediction calculation.

When the present invention is used in conjunction with the JPEG compression standard, instead of removing the mean value of each sub-region, the mean value of each 8×8 block can be removed prior to the calculation of the coefficient data at the encoder. This is particularly advantageous with the JPEG standard, since JPEG, when applied to the color difference data, performs a differential (DPCM) encoding of the mean values of each 8×8 block.

Since the quantized coefficients are not equal to the original coefficients, it is important to use the quantized coefficients to form the prediction in the encoder predictor boxes **30**, **36**, and **42**. Otherwise the prediction at the decoder, which is based on the quantized coefficient values, will differ from the prediction made at the encoder.

In general, when using optimum linear prediction in a color image with n color planes, n(n−1)/2 predictor coefficients are generated for every stripe or sub-region of the image. These quantized coefficients are then encoded by the "coefficient data encode" box **35**, **48**, **50** before being stored in the image memory **18**. The encoding process converts the quantized coefficient data values into fixed-length or variable-length codewords. When using variable-length codes, the data corresponding to the different sub-regions of the image can be differentially encoded to further reduce the filesize corresponding to the quantized coefficient data. For example, for each color plane, the difference between the value of a quantized coefficient of a given sub-region and the corresponding value of the quantized coefficient of the previous sub-region is found and variable-length coded. A preferred method of variable-length coding in the present invention is Huffman coding. However, in many cases the file size associated with the predictor coefficient data is only a small fraction of the overall compressed image file size. As a result, the quantized coefficient data can also be fixed-length coded. In such a case, it is usually advantageous to use a Lloyd-Max nonuniform MMSE quantizer designed for a Laplacian probability as described before.

It was mentioned before that the general method described in the present invention is applicable to any method used to compress the color difference data. The JPEG compression standard has found widespread use in many applications involving compression of continuous tone images due to its compression efficiency and availability of hardware and software implementations. If the JPEG standard is used as a preferred method of compression in conjunction with the present invention, the considerations noted below can help improve the efficiency of the overall system.

In general, the JPEG compression standard segments the image into 8×8 blocks and applies a discrete cosine transform to each 8×8 block. The DCT coefficients that are obtained in this manner are uniformly quantized using a user-defined quantization matrix and are then variable-length coded using a Huffman coder. If the JPEG standard

is used in conjunction with the optimum linear predictor described before, the predictor can use the DCT-domain representation of each color plane (or each color difference image) instead of its image-domain representation. This constitutes no compromise in the optimality of the process since the DCT transform is also a linear transformation. As a result, the optimum prediction coefficients in the DCT domain will be similar in value to the ones derived in the image domain. However, using the DCT-domain representation can result in a simpler encoder and decoder structure, especially in the case of the encoder depicted in FIG. **6**.

Also, when choosing the size of the stripe or the region for prediction optimization, it is advantageous to choose the dimensions of the region to be an integer multiple of 8 in order to align the regions boundaries with the DCT block boundaries. Our findings indicate that for photographic type imagery, a square region between 32–64 pixels wide delivers optimum compression. For regions less than this, the cost of transmitting and storing the associated coefficient data may out-weigh the benefit of the reduced dimensions for color plane differencing.

Finally, in certain applications it may be advantageous to use a nonlinear predictor instead of a linear predictor. A common method of applying a nonlinear predictor is to first process a given color plane with a look-up table (LUT) and then apply a linear predictor as described above. An example of such an application is when the image is generated from a desktop application program such as a word processor or a desktop publisher and is stored in an RGB format. Since the high quality printers usually use four color planes, e.g., CMYK, a transformation from RGB to the CMYK space is applied to the RGB data prior to printing. If the transformation from RGB to CMYK were a linear color transform, the optimum linear predictor described in this invention would be sufficient to remove the resulting correlation. However, the color transform applied to the RGB data is generally designed to compensate for the gamut limitations of the printer inks and specific printer mechanisms which are likely to be of non-linear nature. As a result, a linear operation would not be sufficient in removing that correlation. Fortunately, the color transformation is often known, and it is possible to effectively invert it by constructing a LUT. This LUT is applied to each color plane prior to the linear prediction stage. Thus, the combination of this LUT operation and the linear prediction creates a nonlinear prediction process. Finally, it should be noted that the nonlinear effect needs to be reintroduced at the decoder end before the data is rendered to the print medium.

A computer program written in the C++ language for performing the encoding and decoding process of the present invention is included in Appendix A.

The invention has been described with reference to a preferred embodiment. However, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.

APPENDIX A

```
#include "image.h"
float thresh = 0.001f;
#define COEFF_MAX (.75)
#define COEFF_MIN (-.75)
int dct_dim=8;
```

APPENDIX A-continued

```
image image::remove_dc_from_blocks(int dimx,int dimy)
{
    int i,j,line,pixel,fast;
    float sum=0.0f;
    image idc (get_nbands( ),get_npix( )/dimx,get_nlin( )/dimy,32);
    if(get_nbits( )!=32) deepcpy(change_bits_to(32));
    for(int ib=0;ib<get_nbands( );ib++)
    {
        for(line=0;line<get_nlin( );line+=dimy)
        {
            for(pixel=0;pixel<get_npix( );pixel+=dimx)
            {
                sum=0.0f;
                for(j=line;j<(line+dimy);j++)
                {
                    fast = (pixel+dimx);
                    for(i=pixel;i<fast;i++)
                    {
                        sum+=get_pixel(ib,i,j);
                    }
                }
                sum/=(dimx*dimy);
                idc.put_pixel(sum,ib,pixel/dimx,line/dimy);
                for(j=line;j<(line+dimy);j++)
                {
                    fast = (pixel+dimx);
                    for(i=pixel;i<fast;i++)
                    {
                        put_pixel( (get_pixel(ib,i,j)-sum) ,ib,i,j);
                    }
                }
            }
        }
    }
    return(idc);
}
void image::add_dc_to_blocks(const image &idc, int dimx,int dimy)
{
    int i,j,line,pixel,fast;
    float dc=0.0f
    if(get_nbits( )!=32) deepcpy(change_bits_to(32));
    for(int ib=0;ib<get_nbands( );ib++)
    {
        for(line=0;line<get_nlin( );line+=dimy)
        {
            for(pixel=0;pixel<get_npix( );pixel+=dimx)
            {
                dc=idc.get_pixel(ib,pixel/dimx,line/dimy);
                for(j=line;j<(line+dimy);j++)
                {
                    fast = (pixel+dimx);
                    for(i=pixel;i<fast;i++)
                    {
                        put_pixel( (get_pixel(ib,i,j)+dc) ,ib,i,j);
                    }
                }
            }
        }
    }
}
double dot(const image& v1,const image& v2,int ib1,int ib2)
{
    int line=0;
    int pixel=0;
    double dot = 0;
    for(line=0;line<v2.get_nlin( );line++)
    {
        for(pixel=0;pixel<v2.get_npix( );pixel++)
        {
            dot+=(v1.get_pixel(ib1,pixel,line)*v2.get_pixel
            (ib2,pixel,line));
        }
    }
    return(dot/(v1.get_npix( )*v1.get_nlin( )));
}
```

APPENDIX A-continued

```
void main(int argc, char *argv[ ])
{
    if(argc<5)
    {
        printf("\n\nUsage:: gs block_size gs_quant dct_quant
        infile direction\n");
        printf("Decorrelates input bands\n");
    }
    else if (atoi(argv[5])>0)   // decorrelate
    {
        printf("\n Forward adaptive GS\n");
        int j,sj=0,si=0;
        thresh = .1f;
        float dct_q=(float)atof(argv[3]);
        float gs_q=(float)atof(argv[2]);
        int dim=atoi(argv[1]);
        image a(argv[4]);
        {
            image b=a;
            a.put_band(b,0,3);
            a.put_band(b,1,2);
            a.put_band(b,2,1);
            a.put_band(b,3,0);
        }
        image idc=a.remove_dc_from_blocks(dim,dim);
        idc.save("dc.tif");
        int nbc= (a.get_nbands( )-1)*a.get_nbands( )/2;
        image coeff(nbc,a.get_npix( )/dim,a.get_nlin( )/dim,32);
        image small;
        for( j=0;j<a.get_nlin( );j+=dim)
        {
            for(int i=0;i<a.get_npix( );i+=dim)
            {
                small = a.copy(i,j,i+dim-1,j+dim-1);
                small= small.decorr(&coeff,i/dim,j/dim,gs_q);
                a.paste(i,j,small);
            }
        }
        a.save("gs.tif");
        a.forward_dct_blocks(dct_dim);
        image mom= a.get_band(0);
        a.zero_high_frequencies_of_dct_block(dct_dim);
        a.put_band(mom,0,0);
        a.quantize_dct_blocks(dct_q,dct_dim,"c.tif");
        idc.compress_dct_blocks(MIN(idc.get_npix( ),dim),
        "dc.bin","dc.his");
        coeff.compress_dct_blocks(MIN(idc.get_npix( ),dim),
        "coeff.bin","coeff.his");
        a.compress_dct_blocks(MIN(idc.get_npix( ),dct_dim),
        "gs.bin","p.his");
        coeff.save("coeff.tif");
    }
    else // recorrelate
    {
        printf("\n Inverse adaptive GS\n");
        int j,sj=0,si=0;
        float dct_q=(float)atof(argv[3]);
        float gs_q=(float)atof(argv[2]);
        int dim=atoi(argv[1]);
        int dim2=dim/2;
        image a=decompress_dct_blocks("gs.bin");
        image coeff =decompress_dct_blocks("coeff.bin");
        image idc = decompress_dct_blocks("dc.bin");
        image small;
        a.invert_quantization_dct_blocks(dct_q,dct_dim,"c.tif");
        a.inverse_dct_blocks(dct_dim);
        for( j=0;j<a.get_nlin( );j+=dim)
        {
            for(int i=0;i<a.get_npix( );i+=dim)
            {
                small = a.copy(i,j,i+dim-1,j+dim-1)
                small= small.recorr(&coeff,i/dim,j/dim,gs_q);
                a.paste(i,j,small);
            }
        }
```

APPENDIX A-continued

```
        a.add_dc_to_blocks(idc,dim,dim);
        {
            if(a.get_nbands( )==4)
            {
                image b=a;
                a.put_band(b,0,3);
                a.put_band(b,1,2);
                a.put_band(b,2,1);
                a.put_band(b,3,0);
            }
        }
        a.deepcpy(a.change_bits_to(8));
        a.save("decomp.tif");
    }
}
image image::decorr( image *c, int bx, int by,float q)
{
        int         sum=0;
        double      *var, *coeff;
        image       *gs;
        gs        = new  image       [get_nbands( )];
        var       = new  double      [get_nbands( )];
        coeff     = new double       [get_nbands( )];
        image ret=*this;
        gs[0] = ret.get_band(0);
        var[0] = dot(gs[0],gs[0],0,0);
        ret.put_band(gs[0],0,0);
        sum=0;
        for(int ib=1;ib<(ret.get_nbands( ));ib++)
        {
            image v= ret.get_band(ib);
            float val;
            image vp=v;
            val=0.0f;
            for(int kk=0;kk<ib;kk++)
            {
                if (var[kk]>thresh)
                {
                    coeff[kk]=dot(v,gs[kk],0,0)/var[kk];
                    {
                        if (coeff[kk]>COEFF_MAX)
                        coeff[kk]=COEFF_MAX;
                        else if(coeff[kk]<COEFF_MIN)
                        coeff[kk]=COEFF_MIN;
                    }
                    coeff[kk]/=q;
                    if(coeff[kk]>0)
                    {
                        coeff[kk]=((int)(coeff[kk]+.5));
                    }
                    else
                    {
                        coeff[kk]=((int)coeff[kk]-.5);
                    }
                }
                else
                {
                    coeff[kk]=0.0;
                }
                for(int jj=0;jj<v.get_nlin( );jj++)
                {
                    for(int ii=0;ii<vp.get_npix( );ii++)
                    {
                        val = vp.get_pixel(0,ii,jj);
                        val-=(float)(q*coeff[kk]*gs[kk].get_pixel(0,
                        ii,jj));
                        vp.put_pixel((float)(int)(val),0,ii,jj);
                    }
                }
            }
            gs[ib]=vp;
            ret.put_band(gs[ib],ib,0);
            var[ib]=dot(gs[ib],gs[ib],0,0);
            for(int uu=0;uu<ib;uu++)
            {
                c->put_pixel((float)coeff[uu];sum,bx,by);
                sum++;
            }
        }
```

APPENDIX A-continued

```
        delete [ ] gs;
        delete [ [ var;
        delete [ ] coeff;
        return(ret);
}
image image::recorr( image *c, int bx, int by,float   q)
{
        float v;
        image gs=*this;
        for (int 1=0;1<get_nlin( );1++)
        {
            for(int p=0;p<get_npix( );p++)
            {
                // 1
                if(get_nbands( )>1)
                {
                    v = get_pixel(1,p,l)+   q*c->get_pixel(0,
                    bx,by)*get_pixel(0,p,l);
                    gs.put_pixel((float)(int)v,1,p,l);
                }
                // 2
                if(get_nbands( )>2)
                {
                    v = get_pixel(2,p,l)+   q*c->get_pixel(1,
                    bx,by)*get_pixel(0,p,l)+
                            q*c->get_pixel(2,bx,by)*get_pixel(1,p,
                            l);
                    gs.put_pixel((float)(int)v,2,p,l);
                }
                // 3
                if(get_nbands( )>3)
                {
                    v = get_pixel(3,p,l)+   q*c->get_pixel(3,
                    bx,by)*get_pixel(0,p,l)+
                            q*c->get_pixel(4,bx,by)*get_pixel(1,p,
                            l)+
                            q*c->get_pixel(5,bx,by)*get_pixel(2,p,
                            l);
                    gs.put_pixel((float)(int)v,3,p,l);
                }
            }
        }
        if(get_nbands( )>4)
        {
            printf("Too many bands!");
            exit(1);
        }
        return(gs);
}
image image::get_band(int bnd_no)
        {
            float val;
            if(bnd_no>=get_nbands( ))
            {
                error("Not enough bands for requested data!");
            }
            image ref(1,get_npix( ),get_nlin( ),get_nbits( ));
            for(int j=0;j<get_nlin( );j++)
            {
                for(register int i=0;i<get_npix( );i++)
                {
                    val = get_pixel(bnd_no,i,j);
                    ref.put_pixel(val,0,i,j);
                }
            }
            return(ref);
        }
        void image::put_band(image &ref,int dest_bno,int ref_bno)
        {
            char util [128];
            if( (ref.get_nbands( )<=ref_bno)||
            (get_nbands( )<=dest_bno) )
            {
                ref.report("Image Reference");
                report("Image destination");
                sprintf(util,"Trying to put band %d into
                band %d", ref_bno ,dest_bno);
                error(util);
            }
```

APPENDIX A-continued

```
            float val;
            int ppp=MIN(get_npix( ),ref.get_npix( ));
            int lll=MIN(get_nlin( ),ref.get_nlin( ));
            for(int j=0;j<lll;j++)
            {
                for(register int i=0;i<ppp;i++)
                {
                    val = ref.get_pixel(ref_bno,i,j);
                    put_bounded_pixel(val,dest_bno,i,j);
                }
            }
        }
#include "huffman.h"
#define HIST_FILE_LENGTH 16
void image::quantize_dct_blocks(float quant,int dim,char *csf_file)
{
    int i,j,ii,jj,ib
    float v;
    image csf(csf_file)
    if(dim!=csf.get_npix( )) csf=csf.resample(dim,dim);
    int dim2=dim/2
    float test;
    for( ib=0;ib<get_nbands( );ib++)
    {
        if(ib==0) test=1.0*quant;
        else test =quant,
        for(jj=0,jj<get_nlin( );jj+=(dim))
        {
            for( ii=0;ii<get_npix( );ii+=(dim))
            {
                for(j=jj;j<(jj+dim);j++)
                {
                    for( i=ii;i<(ii+dim);i++)
                    {
                        v = get_pixel(ib,i,j);
                        v/=
                        (test*csf.get_pixel(0,i-ii,j-jj) );
                        if(v<0) v-=.5f;
                        else v+=.5f;
                        put_pixel((float)((int)(v)),
                        ib,i,j);
                    }
                }
            }
        }
    }
}
void image::zero_high_frequencies_of_dct_block(int
dct_dim)
{
    int i,j,ii,jj,ib;
    int sx,sy,ex,ey;
    int dim2=dct_dim
    for( ib=0;ib<get_nbands( );ib++)
    {
        for(jj=0;jj<get_nlin( );jj+=(dct_dim))
        {
            for( ii=0;ii<get_npix( );ii+=(dct_dim))
            {
                sy=jj+dim2
                ey=jj+dct_dim;
                for(j=sy;j<(ey);j++)
                {
                    sx=ii+dim2;
                    exii+dct_dim;
                    for( i=sx;i<(ex);i++)
                    {
                        put_pixel(0.0f, ib,i,j)
                    }
                }
            }
        }
    }
}
void image::invert_quantization_dct_blocks(float
quant,int dim,char *csf_file)
{
    int i,j,ii,jj,ib;
    float v;
```

APPENDIX A-continued

```
        float test;
        image csf(csf_file);
        if(dim!=csf.get_npix( )) csf=csf.resample(dim,dim);
        for( ib=0;ib<get_nbands( );ib++)
    {
        if(ib==0) test=1.0*quant;
        else test =quant;
        for(jj=0;jj<get_nlin( );jj+=(dim))
        {
            for( ii=0;ii<get_npix( );ii+=(dim))
            {
                for(j=jj;j<(jj+dim);j++)
                {
                    for( i=ii;i<(ii+dim);i++)
                    {
                        //    if(i!=ii&&j!=jj)
                        {
                            v= (get_pixel(ib,i,j)
                            *csf.get_pixel
                            (0.i-ii,j-jj)
                            *test) ;
                            put_pixel((float)v,
                            ib,i,j);
                        }
                    }
                }
            }
        }
    }
}
void image::compress_dct_blocks(int dim,char *out_file,
char *hist_file)
{
        int blck_elms,hist_elms,ib,j,i,k,fmin,fmax,nrun,val,ii,jj;
        int amp_offset,run_offset;
        hist *prob;
        prob = new hist;
        int *xarray=get_x_array( dim);
        int *yarray=get_y_array( dim);
        // Find the bounds for the histogram table
        fmin=(int)get_pixel(0,0,0);
        fmax=fmin;
        for( ib=0;ib<get_nbands( );ib++)
        {
            for(j=0;j<get_nlin( );j++)
            {
                for( i=0;i<get_npix( );i++)
                {
                    fmin=(int)MIN(fmin,get_pixel(ib,i,j));
                    fmax=(int)MAX(fmax,get_pixel(ib,i,j));
                }
            }
        }
        fmin--;
        fmax++;
        blck_elms = dim*dim;
        amp_offset=-fmin;
        run_offset = fmax -fmin +1;
        hist_elms = fmax -fmin +1 + blck_elms +1;
        prob->initialize(hist_elms);
        nrun=0;
        for(jj=0;jj<get_nlin( );jj+=(dim))
        {
            for( ii=0;ii<get_npix( );ii+=(dim))
            {
                for( ib=0;ib<get_nbands( );ib++)
                {
                    nrun=0;
                    for(k=0;k<blck_elms;k++)
                    {
                        i =xarray[k]+ii;
                        j = yarray[k]+jj;
                        val = (int)get_pixel(ib,i,j);
                        if(val!=0)
                        {
                            // If prior was a run, update...
                            if(nrun!=0)
                            {
                                prob->increment
                                (nrun+run_offset);
                                nrun=0;
                            }
                            prob->increment(val+amp_offset);
                        }
                        else
                        {
                            nrun++;
                        }
                    }
                    if(nrun!=0) // Last k check
                    {
                        prob->increment(nrun+run_offset);
                        nrun=0;
                    }
                }
            }
        }
}
huffman *huff;
huff = new huffman;
huff->initialize(prob->get_ptr( ),hist_elms,1);
char *stream;
long pos=0;
int stream_size = get_npix( )*get_nlin( )*get_nbits( )
*get_nbands( )/8;
stream = new char [stream_size];
memset(stream,0,stream_size);
nrun=0;
for(jj=0;jj<get_nlin( );jj+=(dim))
{
    for( ii=0;ii<get_npix( );ii+=(dim))
    {
        for( ib=0;ib<get_nbands( );ib++)
        {
            nrun=0;
            for(k=0;k<blck_elms;k++)
            {
                i =xarray[k]+ii;
                j = yarray[k]+jj;
                val = (int)get_pixel(ib,i,j);
                if(val!=0)
                {
                    if(nrun!=0)
                    {
                        huff->encode_symbol
                        (nrun+run_offset,stream,&pos);
                        nrun=0;
                    }
                    huff->encode_symbol
                    (val+amp_offset,stream,&pos);
                }
                else
                {
                    nrun++;
                }
            } // End k
            if(nrun!=0) // Last k check
            {
                huff->encode_symbol(nrun+run_offset,
                stream,&pos);
                nrun=0;
            }
        }
    }
}
int pix=get_npix( ),lin=get_nlin( ),bnds=get_nbands( ),
bits=get_nbits( );
float quant =0.0f;
long stream_extent=((pos+7)/8);
FILE *fp;
char hf[HIST_FILE_LENGTH];
sprintf(hf, "%s",hist_file);
fp = fopen(out_file, "wb");
fwrite(&pix,sizeof(long),1,fp);
fwrite(&lin,sizeof(long),1,fp);
fwrite(&bnds,sizeof(long),1,fp);
fwrite(&bits,sizeof(long),1,fp);
fwrite(&hist_elms,sizeof(long),1,fp);
fwrite(&fmin,sizeof(long),1,fp);
```

APPENDIX A-continued

```
        fwrite(&fmax,sizeof(long),1,fp);
        fwrite(&blck_elms,sizeof(long),1,fp);
        fwrite(&dim,sizeof(long),1,fp);
        fwrite(&stream_extent,sizeof(long),1,fp);
        fwrite(&quant,sizeof(float),1,fp);
        fwrite(hf,sizeof(char),HIST_FILE_LENGTH,fp);
        prob->save(hf);
        fwrite(&stream[0],1,stream_extent,fp);
        fclose(fp);
        delete huff;
        delete prob;
        delete [ ] xarray;
        delete [ ] yarray;
}
image decompress_dct_blocks(char *infile)
{
        float quant;
        int blck_elms,hist_elms,ib,j,i,k,fmin,fmax,val,ii,jj,dim;
        int pix,lin,bnds,bits;
        int amp_offset,run_offset,nrun=0;
        long stream_extent;
        huffman *huff;
        huff = new huffman;
        char *stream;
        char hist_file_name[HIST_FILE_LENGTH];
        FILE *fp;
        fp = fopen(infile, "rb");
        fread(&pix,sizeof(long),1,fp);
        fread(&lin,sizeof(long),1,fp);
        fread(&bnds,sizeof(long),1,fp);
        fread(&bits,sizeof(long),1,fp);
        image a(bnds,pix,lin,bits);
        long pos=0;
        int stream_size = a.get_npix( )*a.get_nlin( )
        *a.get_nbits( )*a.get_nbands( )/8;
        stream = new char [stream_size];
        memset(stream,0,stream_size);
        fread(&hist_elms,sizeof(long),1,fp);
        fread(&fmin,sizeof(long),1,fp);
        fread(&fmax,sizeof(long),1,fp);
        fread(&blck_elms,sizeof(long),1,fp);
        fread(&dim,sizeof(long),1,fp);
        fread(&stream_extent,sizeof(long),1,fp);
        fread(&quant,sizeof(float),1,fp);
        fread(hist_file_name,sizeof(char),HIST_FILE_LENGTH,fp);
        hist *prob;
        prob = new hist;
        prob->initialize(hist_file_name);
        huff->initialize(prob->get_ptr( ),prob->get_nelm( ),1);
        delete prob;
        //huff->initialize_from_serialization(fp);
        fread(&stream[0],1,stream_extent,fp);
        fclose(fp);
        int *xarray=get_x_array( dim);
        int *yarray=get_y_array( dim);
        amp_offset=-fmin;
        run_offset = fmax -fmin +1;
        pos =0;
        int raw_value;
        for(jj=0;jj<a.get_nlin( );jj+=(dim))
        {
            for( ii=0;ii<a.get_npix( );ii+=(dim))
            {
                for( ib=0;ib<a.get_nbands( );ib++)
                {
                    nrun=0;
                    for(k=0;k<blck_elms;k++)
                    {
                        i =xarray[k]+ii;
                        j = yarray[k]+jj;
                        // If we're not unraveling a run
                        if(nrun==0) raw_value =
                        huff->decode_symbol(stream,&pos);
                        if(raw_value<run_offset)
                        {
                                val = raw_value-amp_offset;
                                a.put_pixel((float)val,ib,i,j);
                        }
                        else
                        {
                            if(nrun==0)
                            nrun=raw_value-run_offset;
                            a.put_pixel((float)0.0,ib,i,j);
                            nrun--;
                        }
                    }
                }
            }
        }
        fclose(fp);
        delete [ ] stream;
        delete huff;
        delete [ ] xarray;
        delete [ ] yarray;
        return(a);
}
```

PARTS LIST	
10	printing system
12	print server
14	encoder
16	network server
18	image memory
20	multi-pass printer
22	printer memory
24	decoder
26	marking engine
28	compression step
30	encoder predictor
32	subtraction step
34	compression step
35	encoder
36	encoder predictor
38	subtraction step
40	compression step
42	encoder predictor
44	difference operator
46	compressor
48	encoder
50	encoder
52–58	decompression steps
60–64	decoder predictors
66–70	addition step
72	decompression step

We claim:

1. A method for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, comprising the steps of:

a) compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data;

b) using information relating to the image data in the previous color plane(s) to predict the image data in the next color plane to be printed;

c) forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane;

d) compressing the difference image data formed in step c) and storing the resulting compressed data;

e) repeating steps b)–d) until the image data for all of the color planes have been compressed and stored.

2. The method for encoding color image data claimed in claim **1**, wherein the information relating to the image data in the previous color plane(s) is the image data.

3. The method for encoding color image data claimed in claim **1**, wherein the information relating to the image data in the previous color plane(s) is the image data from the last

color plane in the printing sequence and difference image data from subsequent color plane(s) in the image sequence.

4. The method for encoding color image data claimed in claim **1**, wherein the compressing steps are performed using JPEG compression.

5. The method for encoding color image data claimed in claim **1**, wherein the information relating to the image data in the previous color plane(s) is decompressed image data from the last color plane in the printing sequence and decompressed difference image data from subsequent color plane(s) in the image sequence.

6. The method for encoding color image data claimed in claim **1**, wherein the prediction steps are performed using an optimum linear predictor.

7. The method for encoding color image data claimed in claim **6**, wherein the optimum linear predictor is derived via a Gram-Schmidt orthogonalization process.

8. The method for encoding color image data claimed in claim **6**, wherein the previous color plane(s) used in step b) to predict the image data is a single color plane.

9. The method for encoding color image data claimed in claim **8**, wherein the single color plane is the last color plane to be printed.

10. A method for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, comprising the steps of:

a) compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data;

b) using the image data in the last color plane to predict the image data in the next color plane to be printed;

c) forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane;

d) compressing the difference image data formed in step c) and storing the resulting compressed data;

e) using the image data in the last color plane and the difference image data in the previous color plane(s) to predict the image data in the next color plane to be printed;

f) forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane;

g) compressing the difference image data formed in step f) and storing the resulting compressed data;

h) repeating steps e)–g) until the image data for all of the color planes have been compressed and stored.

11. A method for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, comprising the steps of:

a) compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data;

b) decompressing the compressed data from step a);

c) using the decompressed image data from step b) to predict the image data in the next color plane to be printed;

d) forming difference image data between the image data in the next color plane to be printed and the predicted image data formed in step c) for that color plane;

e) compressing the difference image data formed in step d) and storing the resulting compressed data;

f) decompressing the compressed difference image data from step e);

g) using the decompressed image data in the last color plane and the decompressed difference image data in the previous color plane(s) to predict the image data in the next color plane to be printed;

h) forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane predicted in step g);

i) compressing the difference image data formed in step h) and storing the resulting compressed data;

j) decompressing the compressed difference image data from step i);

h) repeating steps g)–j) until the image data for all of the color planes have been compressed and stored.

12. The method of encoding color image data claimed in claim **1**, further comprising the decoding steps of:

a) decompressing the stored compressed image data that was used to predict the color plane to be printed;

b) using the decompressed image data from step a) to predict the color plane to be printed;

c) decompressing the stored compressed difference image data corresponding to the color plane to be printed;

d) adding the predicted image data from step b) to the decompressed difference from step c) to reconstruct the image data for the color plane to be printed;

e) repeating steps a)–d) until the last color plane to be printed; and

f) decompressing the last color plane to be printed.

13. Apparatus for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, comprising:

a) means for compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data in a memory;

b) means for using information relating to the image data in the previous color plane(s) to predict the image data in the next color plane to be printed;

c) means for forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane; and

d) means for compressing the difference image data and storing the resulting compressed difference image data in the memory.

14. Apparatus for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, comprising:

a) means for compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data in a memory;

b) means for using the image data in the last color plane to predict the image data in the next color plane to be printed;

c) means for forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane;

d) means for compressing the difference image data formed in step c) and storing the resulting compressed data in the memory;

e) means for using the image data in the last color plane and the difference image data in the previous color plane(s) to predict the image data in the next color plane to be printed;

f) means for forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane; and

g) means for compressing the difference image data formed in step f) and storing the resulting compressed data in memory.

15. Apparatus for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, comprising:

a) means for compressing the image data in the last color plane in the printing sequence and and storing the resulting compressed data in a memory;

b) means for decompressing the compressed data from step a);

c) means for using the decompressed image data from step b) to predict the image data in the next color plane to be printed;

d) means for forming difference image data between the image data in the next color plane to be printed and the predicted image data formed in step c) for that color plane;

e) means for compressing the difference image data formed in step d) and storing the resulting compressed data in the memory;

f) means for decompressing the compressed difference image data from step e);

g) means for using the decompressed image data in the last color plane and the decompressed difference image data in the previous color plane(s) to predict the image data in the next color plane to be printed;

h) means for forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane predicted in step g);

i) means for compressing the difference image data formed in step h) and storing the resulting compressed data;

j) means for decompressing the compressed difference image data from step i).

16. A computer program product, comprising: a computer readable storage medium having a computer program stored thereon for encoding color image data of the type having multiple color planes for a printer that prints the color planes in a given sequence, the computer program performing the steps of:

a) compressing the image data in the last color plane in the printing sequence and storing the resulting compressed data;

b) using information relating to the image data in the previous color plane(s) to predict the image data in the next color plane to be printed;

c) forming difference image data between the image data in the next color plane to be printed and the predicted image data for that color plane;

d) compressing the difference image data formed in step c) and storing the resulting compressed data;

e) repeating steps b)–d) until the image data for all of the color planes have been compressed and stored.

* * * * *

US006608928B1

(12) **United States Patent**
Queiroz

(10) **Patent No.: US 6,608,928 B1**
(45) **Date of Patent: Aug. 19, 2003**

(54) **GENERIC PRE-PROCESSING OF MIXED RASTER CONTENT PLANES**

(75) Inventor: **Ricardo de Queiroz**, Pittsford, NY (US)

(73) Assignee: **Xerox Corporation**, Stamford, CT (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/432,600**

(22) Filed: **Nov. 3, 1999**

(51) **Int. Cl.**[7] **G06K 9/34**
(52) **U.S. Cl.** **382/173**; 382/166; 382/232
(58) **Field of Search** 382/173, 282–284, 382/176, 257, 166, 232; 375/240.1

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,778,092 A * 7/1998 MacLeod et al. 382/176
6,437,881 B1 * 8/2002 Baba et al. 382/176

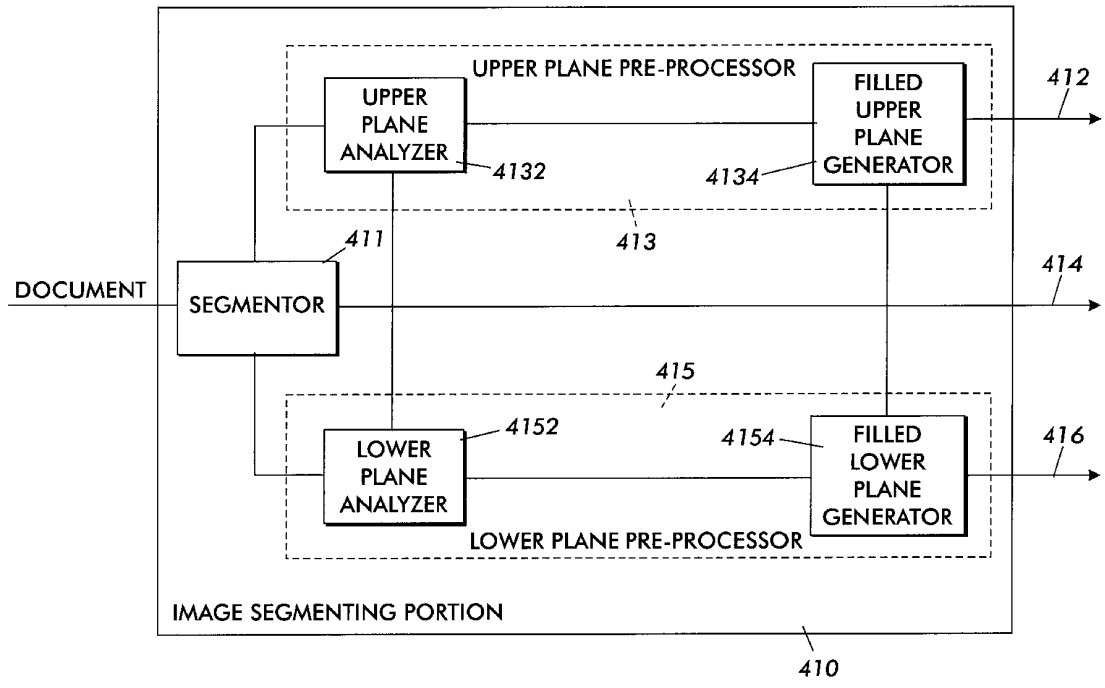
* cited by examiner

Primary Examiner—Amelia M. Au
Assistant Examiner—Colin LaRose
(74) *Attorney, Agent, or Firm*—Oliff & Berridge, PLC

(57) **ABSTRACT**

A mixed raster content system and method improves the performance of the compression process by reducing the amount of data. The system and method uses a selector plane as a reference to aid in reducing the amount of data necessary to encode each of associated planes. A smoothing technique is used to pre-process the associated planes using the information contained in the selector plane, thereby reducing the amount of data that will be subjected to further processing. The system and method determines the useless data, smooths the boundary of the useless data, and replaces the useless data by values that improve compression.

33 Claims, 9 Drawing Sheets



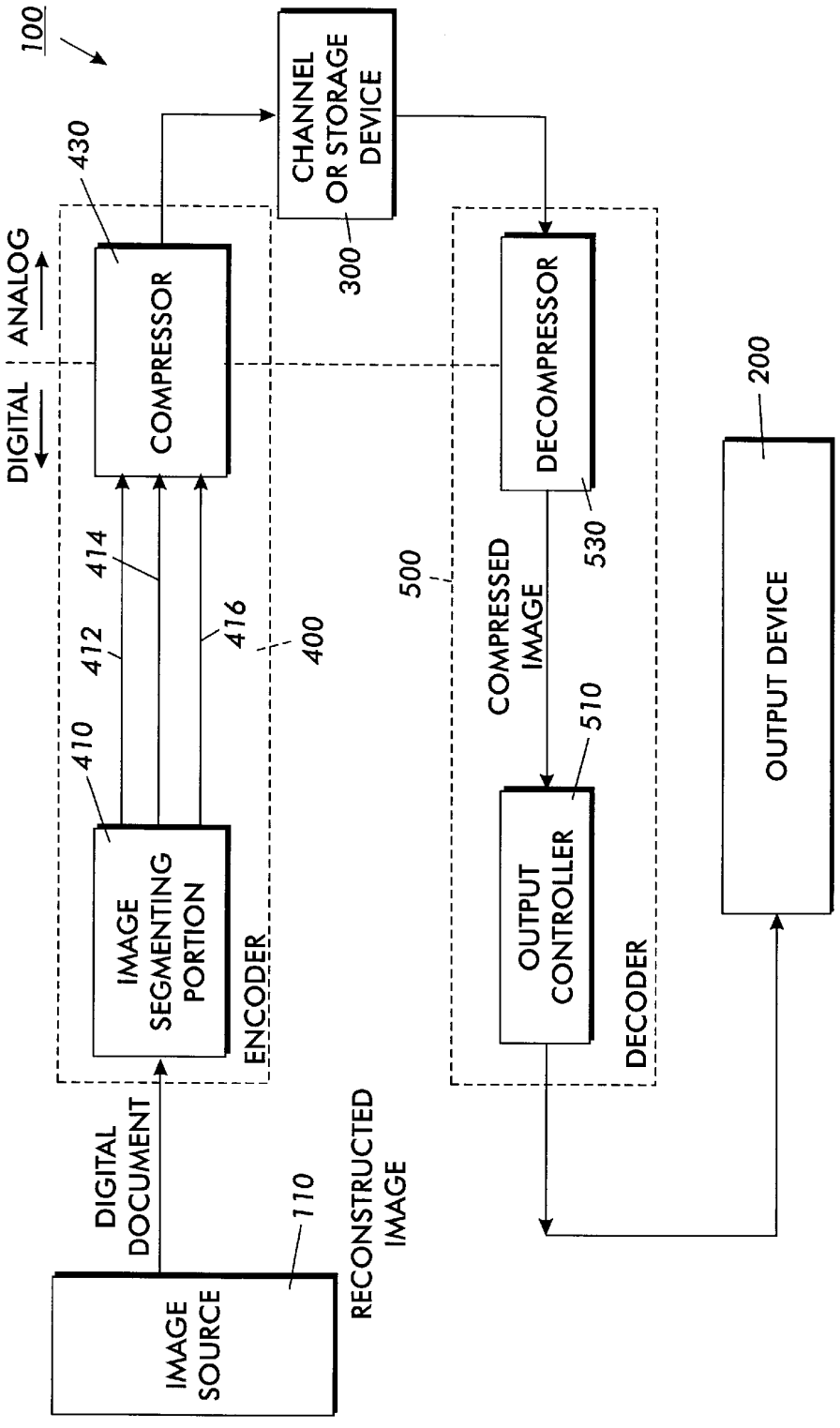

100
CHANNEL OR STORAGE DEVICE
300
430
ANALOG
DIGITAL
COMPRESSOR
DECOMPRESSOR
200
OUTPUT DEVICE
414
412
416
400
500
COMPRESSED IMAGE
530
410
510
IMAGE SEGMENTING PORTION
ENCODER
OUTPUT CONTROLLER
DECODER
DIGITAL DOCUMENT
110
RECONSTRUCTED IMAGE
IMAGE SOURCE

FIG. 1

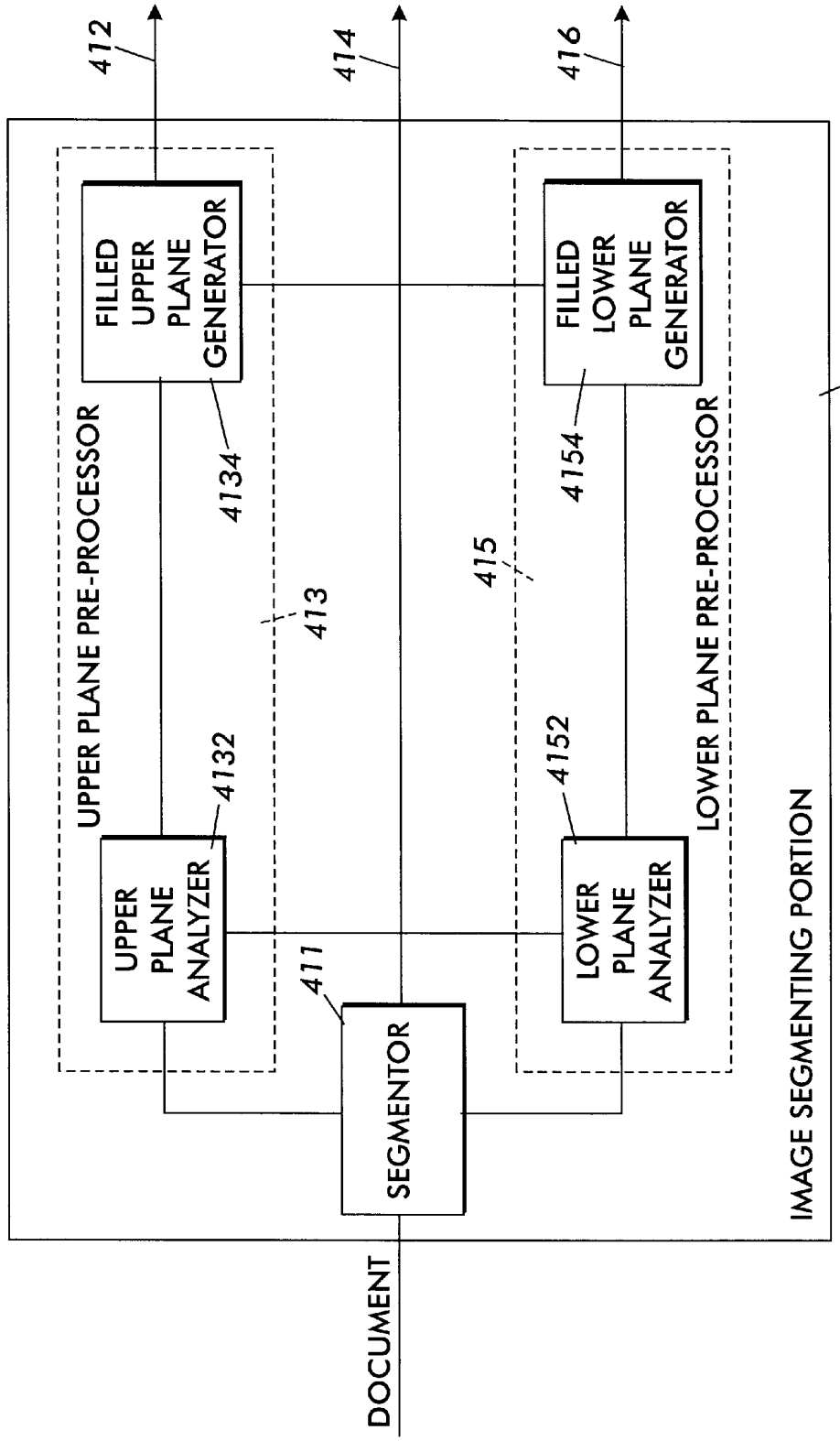

412
414
416
FILLED UPPER PLANE GENERATOR
FILLED LOWER PLANE GENERATOR
4134
4154
UPPER PLANE PRE-PROCESSOR
LOWER PLANE PRE-PROCESSOR
413
415
410
4132
4152
UPPER PLANE ANALYZER
LOWER PLANE ANALYZER
411
SEGMENTOR
IMAGE SEGMENTING PORTION
DOCUMENT

FIG. 2

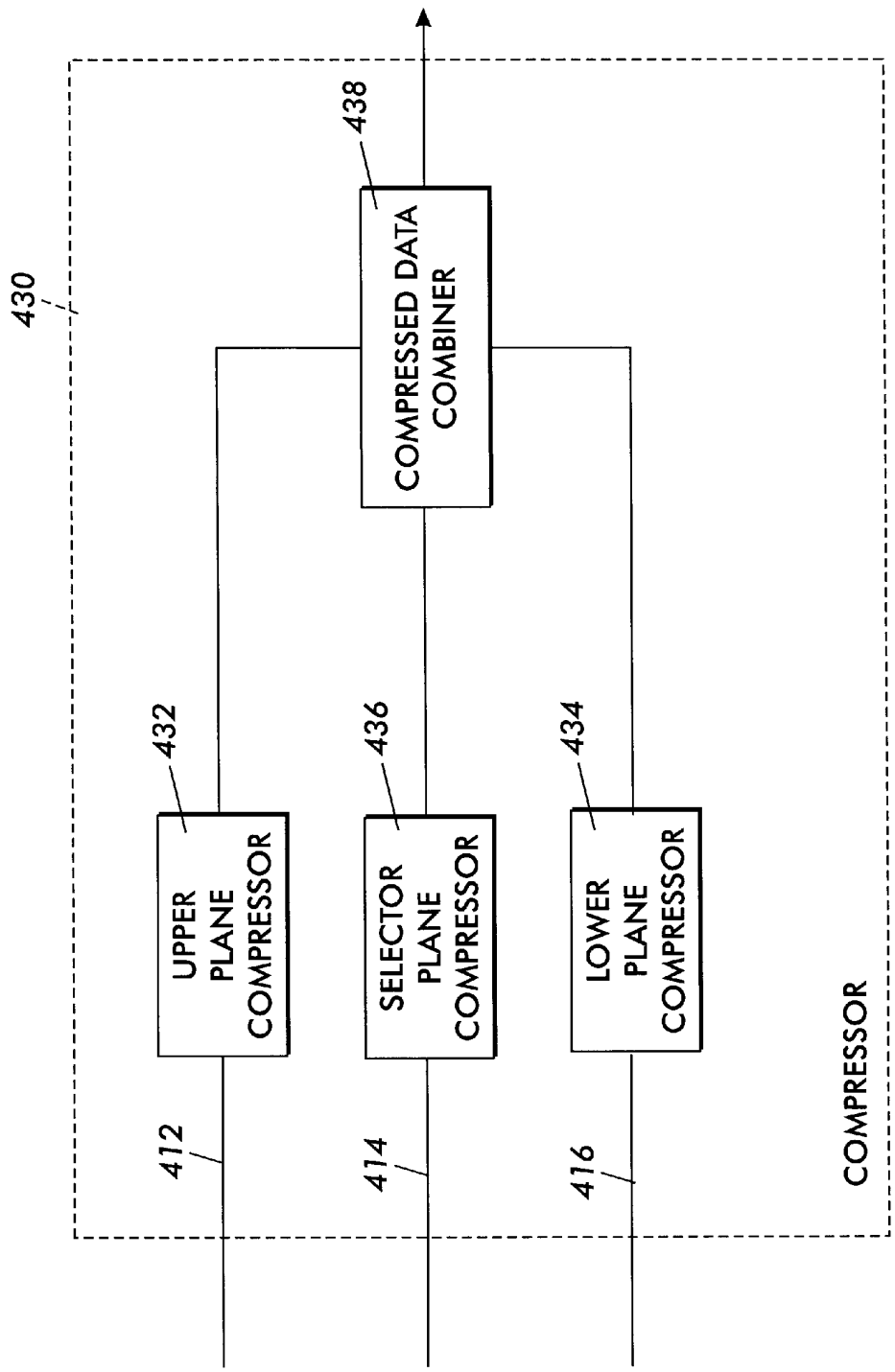

430
438
COMPRESSED DATA COMBINER
432
UPPER PLANE COMPRESSOR
436
SELECTOR PLANE COMPRESSOR
434
LOWER PLANE COMPRESSOR
412
414
416
COMPRESSOR

FIG. 3

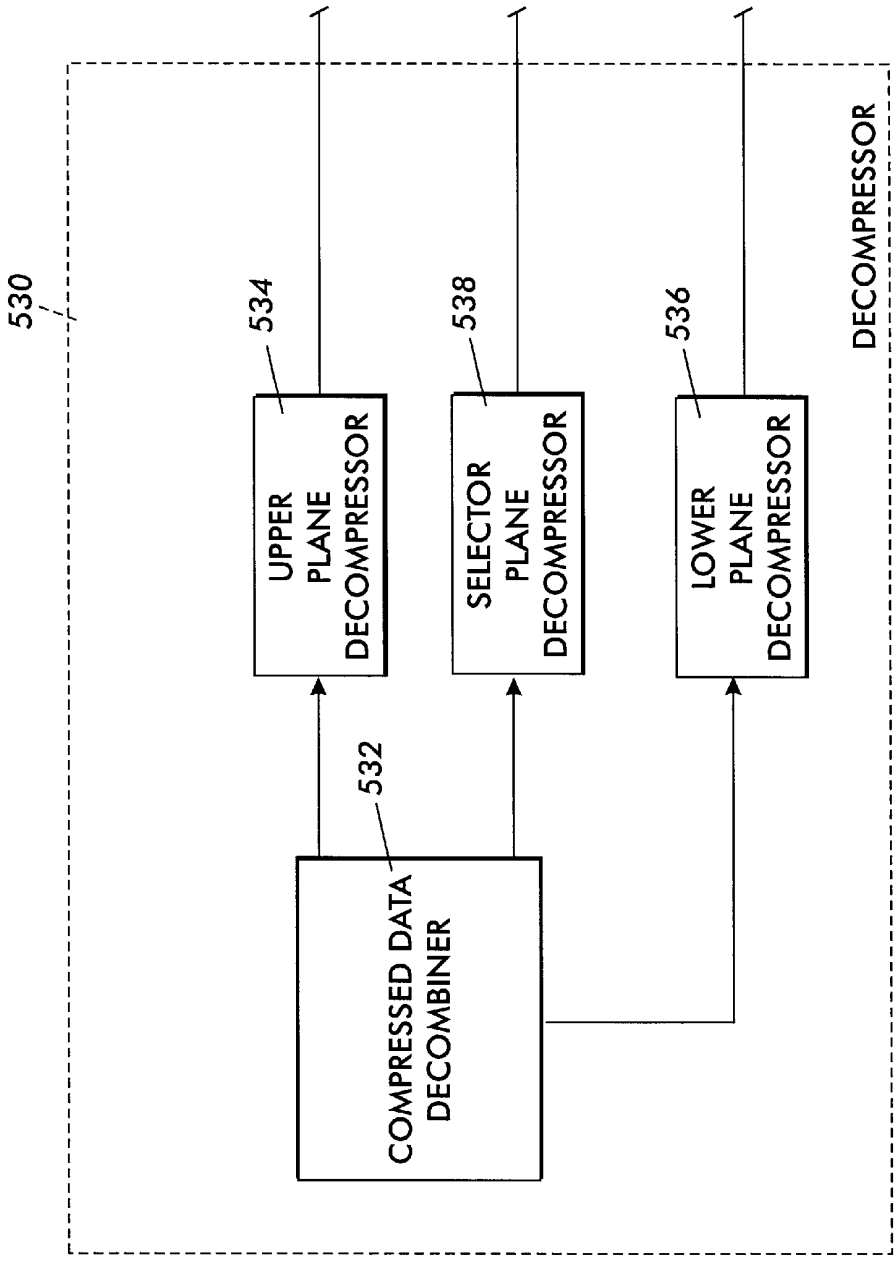

530
534
538
536
DECOMPRESSOR
UPPER PLANE DECOMPRESSOR
SELECTOR PLANE DECOMPRESSOR
LOWER PLANE DECOMPRESSOR
532
COMPRESSED DATA DECOMBINER

FIG. 4

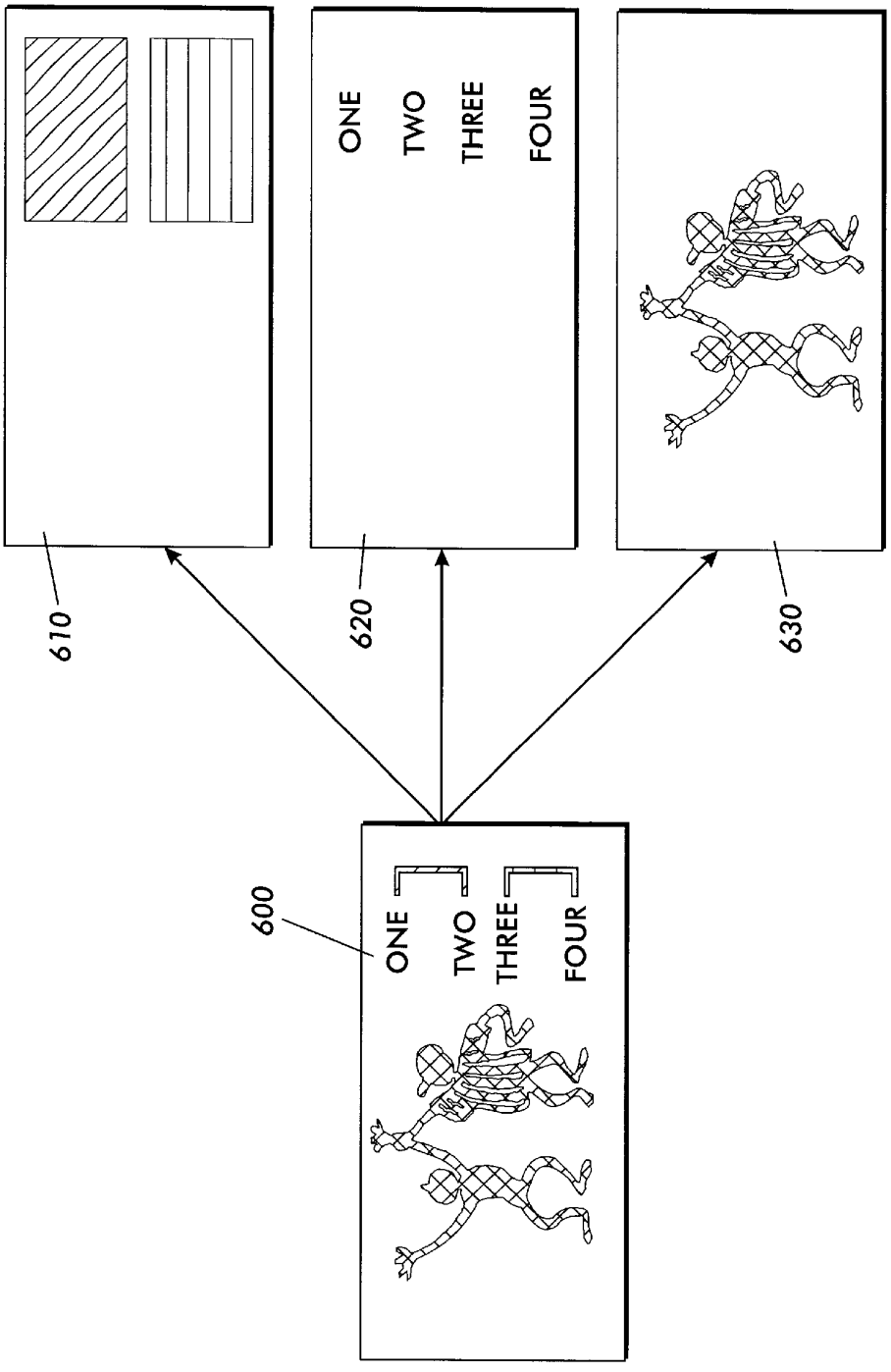


FIG. 5

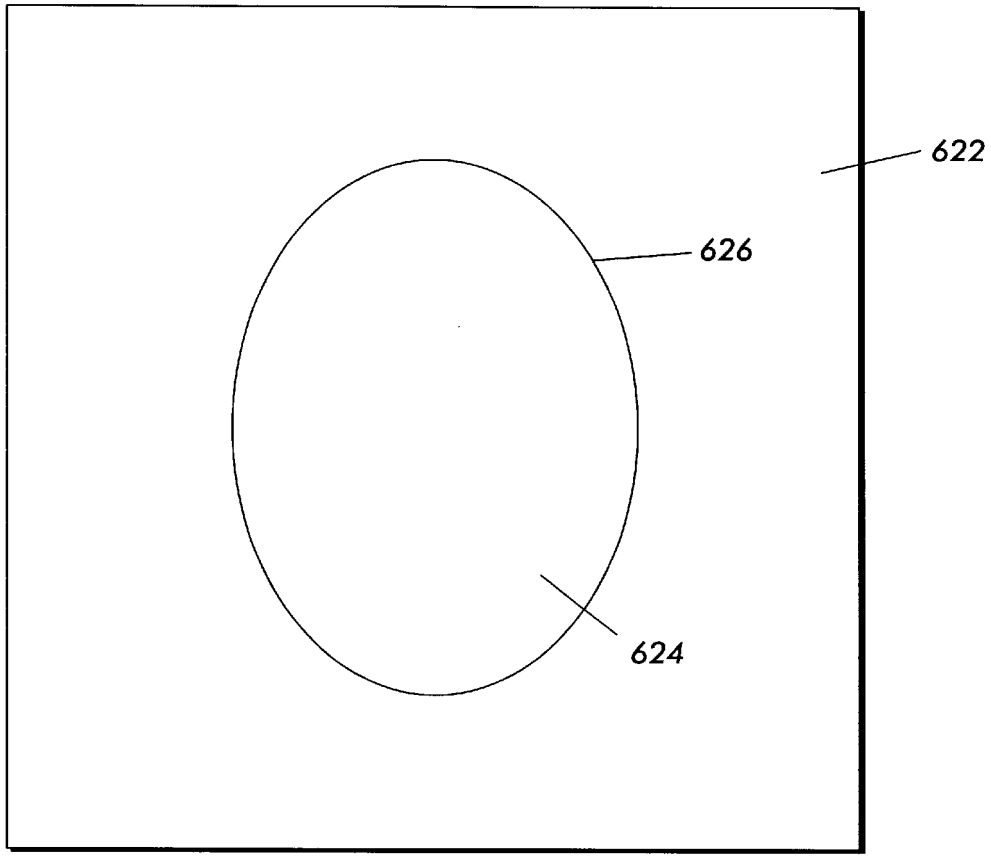


FIG. 6

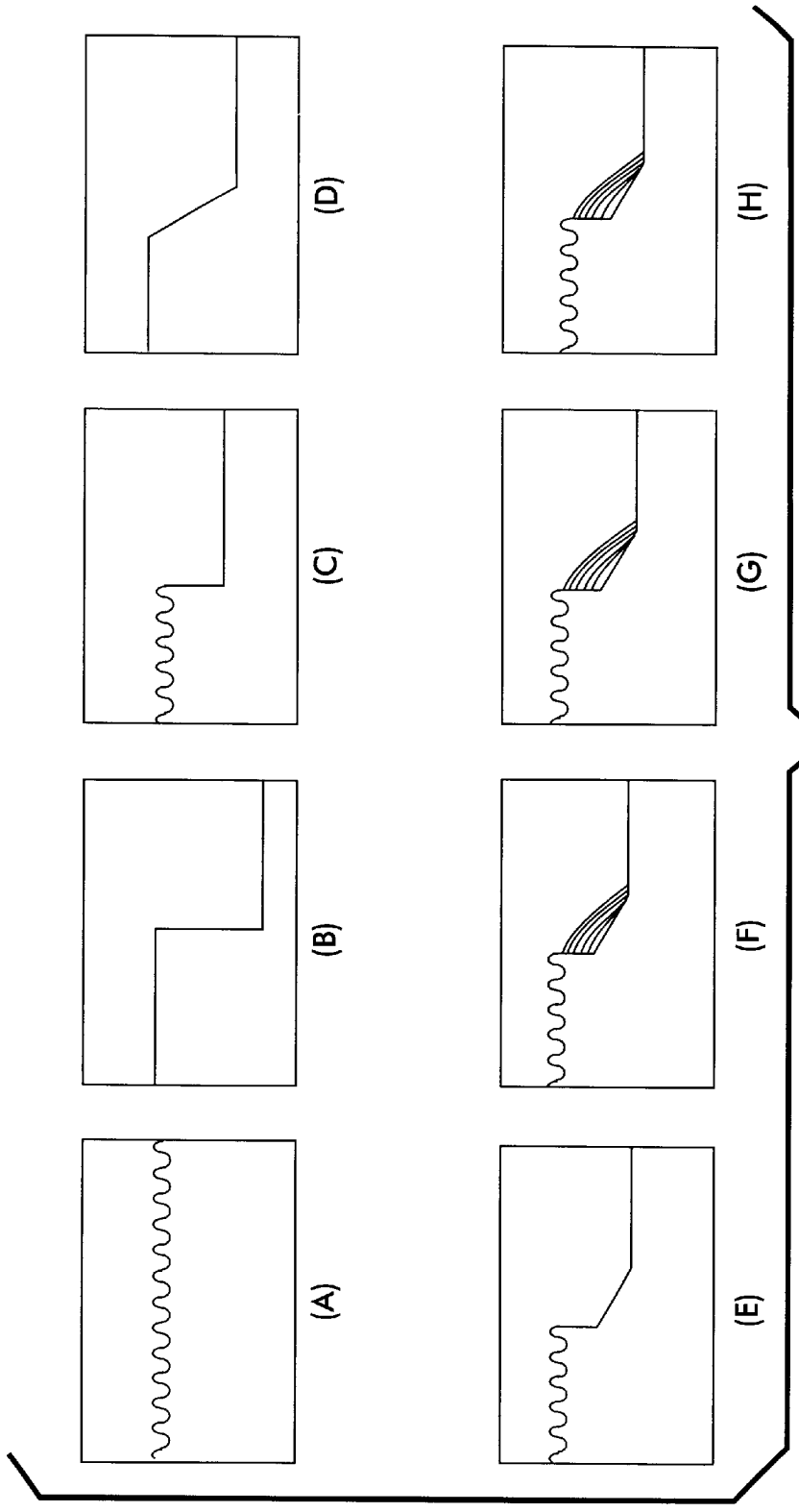


FIG. 7

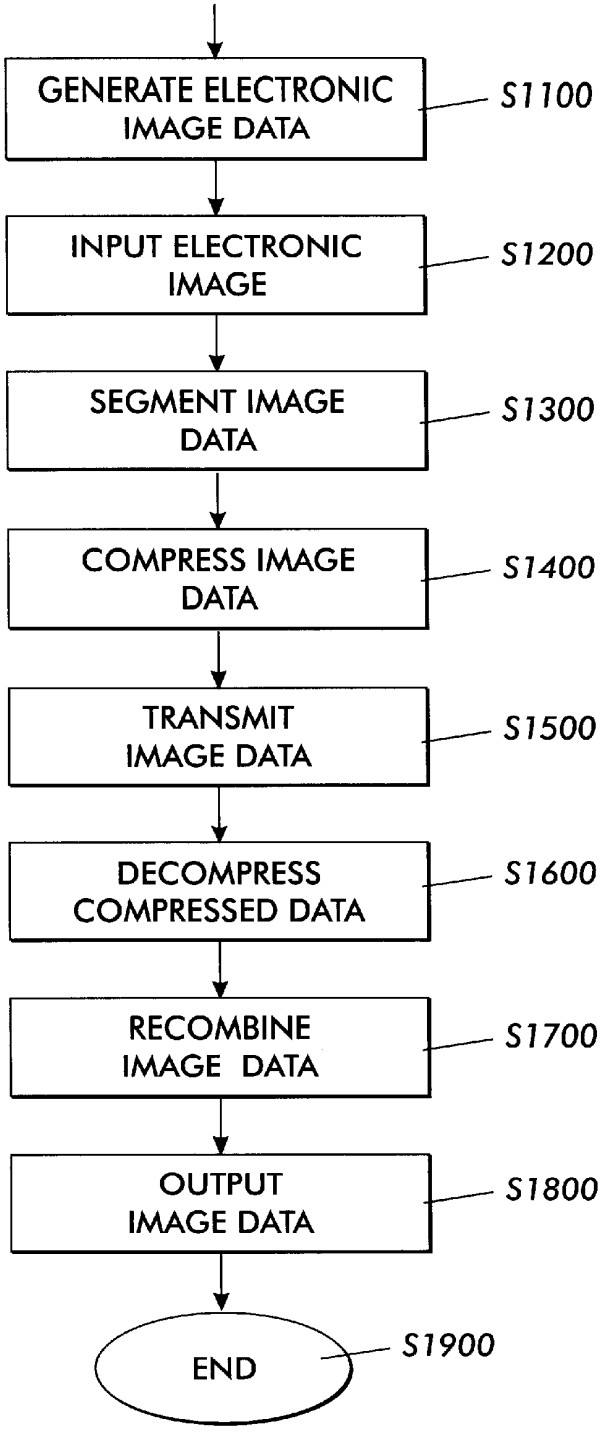

GENERATE ELECTRONIC IMAGE DATA
S1100
INPUT ELECTRONIC IMAGE
S1200
SEGMENT IMAGE DATA
S1300
COMPRESS IMAGE DATA
S1400
TRANSMIT IMAGE DATA
S1500
DECOMPRESS COMPRESSED DATA
S1600
RECOMBINE IMAGE DATA
S1700
OUTPUT IMAGE DATA
S1800
END
S1900

FIG. 8

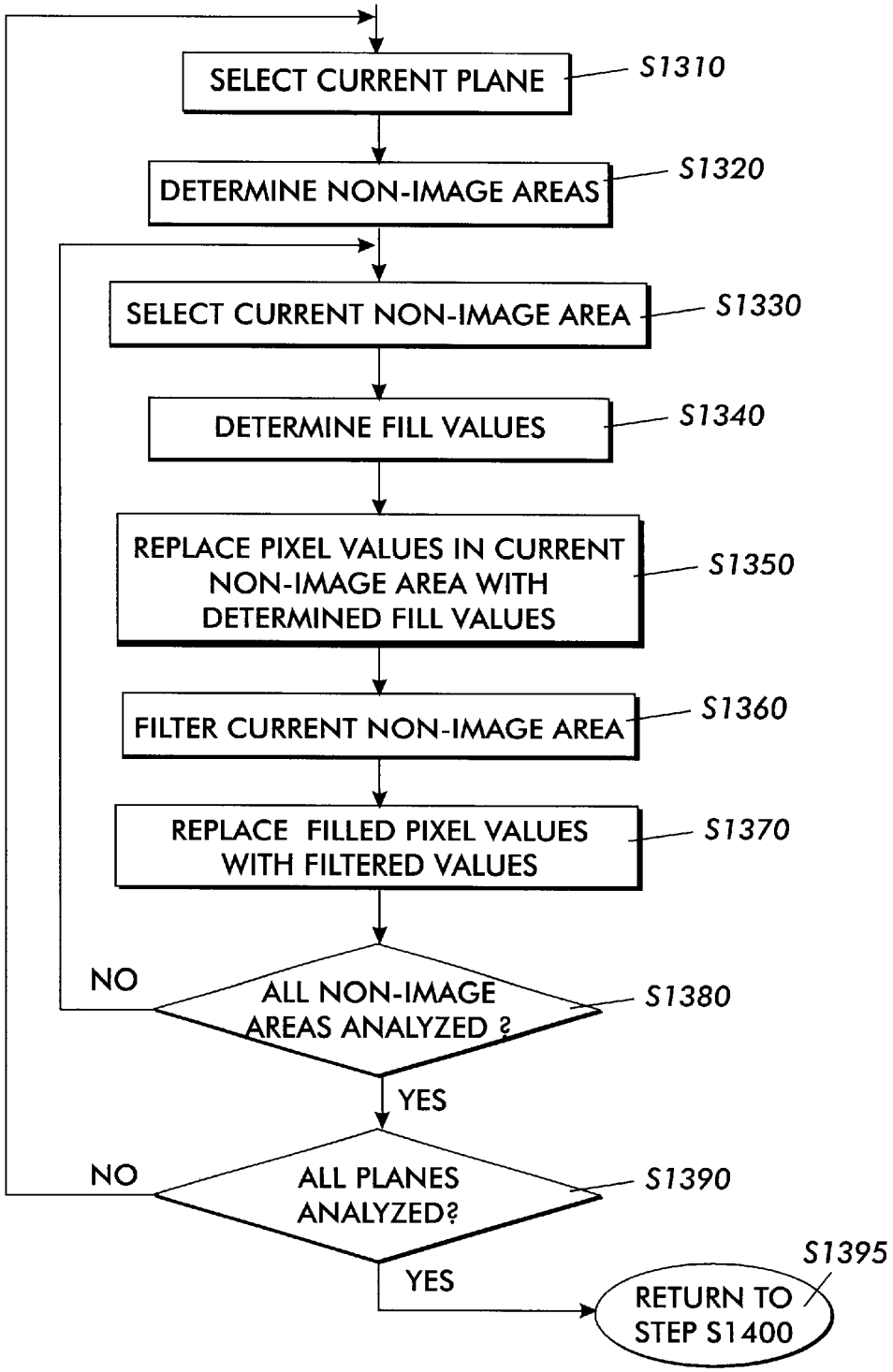

SELECT CURRENT PLANE
S1310
DETERMINE NON-IMAGE AREAS
S1320
SELECT CURRENT NON-IMAGE AREA
S1330
DETERMINE FILL VALUES
S1340
REPLACE PIXEL VALUES IN CURRENT NON-IMAGE AREA WITH DETERMINED FILL VALUES
S1350
FILTER CURRENT NON-IMAGE AREA
S1360
REPLACE FILLED PIXEL VALUES WITH FILTERED VALUES
S1370
NO
ALL NON-IMAGE AREAS ANALYZED ?
S1380
YES
NO
ALL PLANES ANALYZED?
S1390
YES
S1395
RETURN TO STEP S1400

FIG. 9

GENERIC PRE-PROCESSING OF MIXED RASTER CONTENT PLANES

BACKGROUND OF THE INVENTION

1. Field of Invention

This invention generally relates to processing compressed digital images.

2. Description of Related Art

Data compression is required in data handling processes, where too much data is present for practical applications using the data. Commonly, compression is used in communication links to reduce the transmission time or required bandwidth. Similarly, compression is preferred in image storage systems, including digital printers and copiers, where "pages" of a document to be printed are stored temporarily in precollation memory. The amount of media space on which the image data is stored can be substantially reduced with compression. Generally speaking, scanned images, i.e., electronic representations of hard copy documents, are often large, and thus make desirable candidates for compression.

During compression, a scanned image may take a form of a pixel map, where the mapped image is created by digitizing light reflected off of physical media using the scanner. A pixel map is one in which each pixel represents some "value" which indicates the color of the image, or, in the case of gray scale document, how light or dark the image is, at a particular point. Pixel maps may have values that are taken from a set of discrete, non-negative integers. The term bitmap is used to mean a binary pixel map in which pixels can take one of two values, 1 or 0.

SUMMARY OF THE INVENTION

This invention is directed to compressing images from documents which contain different types of data. Mixed Raster Content (MRC) is one approach to satisfying the compression needs of differing types of data. MRC involves separating a composite image into a plurality of planes, and separately applying an appropriate compression technique to each image plane. The document is represented by a pixel map which is decomposed into a three-plane representation of a upper plane, a lower plane, and a binary selector plane that is used to select between the upper and lower planes, for example.

While the MRC technique has shown to be successful at separately processing planes, the segmentation process leaves data in the associated planes in areas that will not be chosen by the selector plane. Thus, the associated planes often will have pixels that are not used to reconstruct the final image. This often results in an increase in the number of bits of useless data that are required to encode the entire image, thereby decreasing the compression ratio.

The systems and methods of this invention improve the performance of the compression process by reducing the amount of data. This invention provides systems and methods that use the selector plane as a reference to aid in reducing the amount of data necessary to encode each of the associated planes.

The systems and methods of this invention provide a smoothing technique used to pre-process the upper and lower planes using the information contained in the selector plane, thereby reducing the amount of data that will be subjected to further processing.

The systems and methods of this invention provide techniques that determine useless data and then smooth the boundary between the actual image data and the useless data. In the systems and methods of this invention, the determined useless data is replaced by values that improve compression.

In various exemplary embodiments of the systems and methods of this invention, useless data in non-image areas of a plane is replaced by a constant value and the boundary regions are smoothed by filtering using a segmented-computing filter structure. In various exemplary embodiments of the systems and methods of this invention, the boundary regions are smoothed independently of the compression techniques to be used. Thus the boundary regions reduce the distortion and increase the compression rate achieved by compressing the planes using generic compressors.

It should be appreciated that the compression systems and methods of this invention can be applied to any set of compression techniques for separately processing the various types of data contained in a composite image. Accordingly, the compression/decompression methods and systems of this invention can be applied to any number of systems, including digital printers and copiers, that need to provide compressed or decompressed images. Systems implementing this invention can include data or image processing systems capable of compressing or decompressing images. The systems and methods described herein are efficient and result in smoother boundaries between regions of different image data types.

These and other features and advantages of this invention are described in or are apparent from the following detailed description of various exemplary embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

Various exemplary embodiments of this invention will be described in detail, with reference to the following figures, wherein:

FIG. **1** is a generalized block diagram of one embodiment of a compression and decompression system according to this invention;

FIG. **2** shows one embodiment of a preferred image segmenting portion according to this invention;

FIG. **3** shows one embodiment of a preferred compressor according to this invention;

FIG. **4** shows one embodiment of a decompressor according to this invention;

FIG. **5** shows one embodiment of a pixel map representation of a document according to this invention;

FIG. **6** illustrates one embodiment of a composite image according to this invention.

FIGS. **7**(A)–(H) show a smoothing operation according to this invention;

FIG. **8** is a flowchart outlining one embodiment of an image compression and decompression method according to this invention; and

FIG. **9** is a flowchart outlining in greater detail the image segmentation according to this invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

FIG. **1** shows one exemplary embodiment of a generalized functional block diagram of a compression and decompression system **100** according to this invention. The compression and decompression system **100** includes an image source **110** that may be any one of a number of different

sources, such as a scanner, a digital copier, a facsimile device suitable for generating electronic image data, or a device suitable for storing and/or transmitting the electronic image data, such as a client or server of a network. The electronic image data from the image source **110** is provided to an encoder **400** of the compression and decompression system **100**.

In particular, the encoder **400** includes an image segmenting portion **410** that divides the electronic image data into a plurality of planes. The image is segmented into at least two different types of image planes and at least one selector plane. In one exemplary embodiment, the segmenting operation may be accomplished by windowing or cropping that enables identifying and copying data associated with each of the image planes from the input document to a memory of the encoder **400**.

Once stored, the planes of data are output by the image segmenting portion **410** over at least the signal lines **412**, **414** and **416** to a compressor **430**. In the compressor **430**, each plane of the image data may be operated on separately using one of a variety of compression operations associated with the particular compression technique to compress the image data within that plane.

Once compressed, each plane of the image data then is preferably transferred to the channel or storage device **300**. The channel or storage device **300** can be either or both of a channel device for transmitting the compressed image data to the decoder **500** or a storage device for indefinitely storing the compressed image data until there arises a need to decompress the compressed image data. The channel device can be any known structure or apparatus for transmitting the compressed image data from a first apparatus implementing the encoder **400** according to this invention to a physically remote decoder **500** according to this invention. Thus, the channel device can be a public switched telephone network, a local or wide area network, an intranet, the Internet, a wireless transmission channel, any other distributed network, or the like.

Similarly, the storage device can be any known structure or apparatus for indefinitely storing compressed image data, such as a RAM, a hard drive and disk, a floppy drive and disk, an optical drive and disk, flash memory or the like. Moreover, the storage device can be physically remote from the encoder **400** and/or the decoder **500**, and reachable over the channel device described above.

When the image is to be decompressed, in one exemplary embodiment, the planes of data are then provided to and processed by the decoder **500**. In particular, the decoder **500** includes a decompressor **530** that receives the planes of data from the channel or storage device **300** and an output controller **510** that combines the decompressed planes from the decompressor **530** to form the decompressed image. Though the decoder **500** is shown in FIG. **1** as physically separate from the encoder **400**, it should be understood that the decoder **500** and the encoder **400** may be different functional and/or structural aspects of a single physical device.

The output controller **510** sends the reconstructed image to the output device **200**. It should be understood that the output device **200** can be any device that is capable of outputting or storing the decompressed image data generated according to the invention such as a printer, facsimile device, a display device, a memory, or the like.

FIG. **2** shows one exemplary embodiment of the image segmenting portion **410** of the encoder **400** according to this invention. The image segmenting portion **410** includes a segmentor **411**. The image segmenting portion **410** also includes an upper plane pre-processor **413** and a lower plane pre-processor **415**, each connected to the segmentor **411**. The upper plane pre-processor **413** includes an upper plane analyzer **4132** and a filled upper plane generator **4134** connected to the upper plane analyzer **4132**. The lower plane pre-processor **415** includes a lower plane analyzer **4152** and a filled lower plane generator **4154** connected to the lower plane analyzer **4152**.

In practice, the segmentor **411** decides which pixels of the image data are to be stored into the upper plane or the lower plane and generates a selector plane that indicates, for each pixel, whether that pixel is to be stored in the upper or lower plane. The upper plane analyzer **4132** analyzes the selector plane to determine the non-image areas, i.e., those areas containing useless data, in the upper plane. Similarly, the lower plane analyzer **4152** analyzes the selector plane to determine the non-image areas, i.e., those areas containing useless data, in the lower plane. The filled upper plane generator **4134** receives the output from the upper plane analyzer **4132** and generates "false" image data from the actual image data so that the useless data in the non-image areas will be completely replaced with false image data. The filled lower plane generator **4154** receives the output from the lower plane analyzer **4152** and generates "false" image data from the actual image data so that the useless data in the non-image areas will be completely replaced with false image data.

The upper plane pre-processor **413** outputs the filled upper plane on a signal line **412** to the compressor **430**. The segmentor **411** outputs the selector plane on a signal line **414** to the compressor **430**. The lower plane pre-processor **415** outputs the filled lower plane on a signal line **416** to the compressor **430**.

It should be appreciated that, depending on the image to be compressed, any known or later developed data types may be pre-processed by the upper plane pre-processor **413** and the lower plane pre-processor **415**, such as continuous tone or picture data, halftone data, line art or graphics data, text data or the like. Thus, any set of compression techniques for these various data types can be used, and the system and method of this invention can be used with any such set of compression techniques.

It should be appreciated that one of ordinary skill in the art can easily modify the structure shown in FIG. **2** using any complementary known or later-developed image type and/or any known or later developed image segmentation technique without any undue experimentation or any unpredictability in the results of such modifications. Thus, while the scope of this disclosure is specifically intended to encompass all such combinations or sets of data types and/or image segmentation techniques, currently known or later developed, it is not necessary to describe to one of ordinary skill in the art each such combination and the resulting modifications to the structure shown in FIG. **2**.

It should be further appreciated that the systems and methods of this invention are not limited to one upper plane and one lower plane, and that any number of upper-lower pairs of planes may be segmented. In particular, as discussed above, the segmentor **411** may decompose the image into an upper plane, a lower plane and a selector plane. Then, the filled upper plane from the filled upper plane generator **4134** can be output to a second image segmenting portion, where the upper plane is further decomposed into a lower sub-upper plane, an upper sub-upper plane, and a second selector plane. In this case, the filled upper plane data output on line

412 is input to an additional segmentor in the second image segmenting portion. Furthermore, the filled upper sub-upper plane can further be decomposed into separate planes, and any number of upper planes may be segmented as appropriate. Thus, any number of additional image segmenting portions may be cascaded together.

FIG. 3 shows one exemplary embodiment of the compressor 430 of the encoder 400 according to this invention. The compressor 430 includes an upper plane compressor 432, a lower plane compressor 434 and a selector plane compressor 436. The compressor 430 further includes a compressed data combiner 438 connected to the upper plane compressor 432, the lower plane compressor 434, and the selector plane compressor 436. It should be appreciated that, depending on the image to be compressed, the upper and lower planes may include any known or later developed data type, such as continuous tone or picture data, halftone data, line art or graphics data, text data or the like. Thus, any set of compression techniques for these various data types can be used, and the system and method of this invention can be used with any such set of compression techniques.

It should further be appreciated that one of ordinary skill in the art can easily modify the structure shown in FIG. 3 using any complementary known or later-developed image type and/or only known or later developed compression technique without any undue experimentation or any unpredictability in the results of such modifications. Thus, while the scope of this disclosure is specifically intended to encompass all such combinations or sets of data types and/or compression techniques, currently known or later developed, that include one or more segment compression techniques, it is not necessary to describe to one of ordinary skill in the art each such combination and the resulting modifications to the structure shown in FIG. 3.

In operation, the "filled" upper plane is input from the image segmenting portion 410 to the upper plane compressor 432, while the "filled" lower plane is input to the lower plane compressor 434 and the selector plane is input to the selector plane compressor 436. The upper and lower planes are each compressed by the upper and lower plane compressors 432 and 434 using an appropriate compression technique for the type of image data stored in the upper and lower planes, respectively. The compressed upper and lower planes are then output to the compressed data combiner 438. Similarly, the selector plane is compressed by an appropriate compression technique by the selector phase compressor 436. The compressed selector plane is also output to the compressed data compressor 438.

The compressed data combiner 438 combines the compressed data in the upper and lower planes, along with the compressed selector plane, into a single data structure, such as a file. The compressed data combiner 438 can also optionally apply a further level of compression, such as Huffman encoding, to the single data structure. The compressed data combiner 438 then outputs the single data file, whether further compressed or not, to the channel or storage device 300.

FIG. 4 shows one embodiment of a decompressor according to this invention. In particular, the compressed image data is input into the decompressor 530. The decompressor 530 includes a compressed data decombiner 532, and an upper plane decompressor 534, a lower plane decompressor 536 and a selector plane decompressor 538, each connected to the compressed data decombiner 532. The compressed data decombiner 532 inputs the single data structure from the channel or storage device 300. If the single data structure has been further compressed, the compressed data decombiner 532 will first decompress it. The compressed data decombiner 532 then splits the single data structure into the compressed upper plane data, the compressed lower plane data, and the compressed selector plane data. The compressed upper plane data is output to the upper plane decompressor 534. The compressed lower plane data is output to the lower plane decompressor 536. Similarly, the compressed selector plane data is output to the selector plane decompressor 538.

The upper plane decompressor 534 decompresses the compressed upper plane data using the complimentary decompression technique to the appropriate compression technique used to compress the upper plane. The upper plane decompressor 534 then outputs the decompressed upper plane data to the output controller 510. Similarly, the lower plane decompressor 536 decompresses the compressed lower plane data using the complimentary decompression technique to the appropriate compression technique used to compress the lower plane. The lower plane decompressor 536 then outputs the decompressed lower plane data to the output controller 510.

Likewise, the selector plane decompressor 538 decompresses the compressed selector plane data using the complimentary decompression technique to the appropriate compression technique used to compress the selector plane. The selector plane decompressor 538 then outputs the decompressed selector plane to the output controller 510.

The output controller 510 recombines the decompressed upper and lower planes into a single output image based on the decompressed selector plane output from the selector plane decompressor 538. In particular, in one exemplary embodiment, the output controller 510 first pastes one of the lower plane or the upper plane, including the filled data, into its appropriate location in the decompressed image buffer. The output controller 510 then pastes the image areas of the other of the lower plane or the upper plane into each area's appropriate location in the decompressed image buffer, based on the selector plane. Thus, each image portion of the other of the lower plane or the upper plane is pasted over the corresponding unused portion of the one of the lower or upper plane. The planes are pasted in this manner because the filled useless data in the one of the lower or upper plane is replaced with image data of the other of the lower or upper plane. This filled useless data is false, and is removed by pasting the data of the image areas of the other of the lower or upper plane over the non-image areas of the one of the lower or upper plane.

In another exemplary embodiment, the output controller 510, using the selector plane, pastes each image portion of the upper plane and the lower plane into its appropriate location in the decompressed image. Thus, only the image portions, without the filled useless data in the upper and lower plane, is pasted into the corresponding portion of the decompressed image. That is, the false image data and optionally the false useless data of one of the planes is not used in the pasting process.

It should be appreciated that the image portions of the upper plane and the lower plane may be pasted into the corresponding portion in using any known or later developed methods and systems. Thus, it should be appreciated that the methods and systems of this invention are not limited to the above-described exemplary embodiments, and that any methods and systems for pasting the image areas while omitting the non-image areas may be used.

It should also be appreciated that the above-outlined structures for the image segmenting portion 410, the com-

pressor **430** and the decompressor **530** are functional in nature. That is, FIGS. **2**, **3** and **4** show the functional organization of the image segmenting portion **410**, the compressor **430** and the decompressor **530**, with each structure being implemented, in these various exemplary embodiments, on a general purpose computer. However, each of the image segmenting portion **410**, the compressor **430**, the output controller **510** and the decompressor **530** can also be implemented on a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA or PAL, or the like. Furthermore, as set forth above, all of the image segmenting portion **410**, the compressor **430**, the output controller **510** and the decompressor **530** can be implemented on a single general purpose computer, a single special purpose computer, a single programmed microprocessor or microcontroller and peripheral integrated circuit elements, a single ASIC or other integrated circuit, a single digital signal processor, a single hardwired electronic or logic circuit such as a discrete element circuit, a single programmable logic device such as a PLD, PLA, FPGA or PAL, or the like.

As shown in FIG. **5**, a pixel map representation of a document **600** usable with various exemplary embodiments of this invention may be segmented into three separate planes using the MRC technique. An upper plane **610** contains color signals for a first subset of image data in the document. A lower plane **630** contains image signals for a second subset of image data in the document. A selector plane **620** selects between the first subset signals and the second subset signals when decompressing the compressed pixel map representation of the document and preprocessing the upper and lower image planes based upon data contained in the selector plane. The upper and lower planes may be stored at the bit depth same or at different bit depths. However, it should be appreciated that the planes can include a reduction in the bit depth. It should also be appreciated that compression of the planes may be done in any suitable format, including, but not limited to, JPEG, MMR, MR, MH, JBIG, JPEG2000, JBIG2 and the like.

It should be appreciated that although the terms "upper" and "lower" are used to describe the image planes on which data resides, it is not intended to limit this invention to any particular arrangement. Accordingly, the planes are not limited to particular data types.

In one embodiment, the upper plane **610** contains the "ink colors" of upper items such as text. The lower plane **630** contains both information that is pertinent to the background color of the page and the continuous-tone pictures that are found on the page. The selector plane **620** may be stored at higher resolution than the upper and lower planes **610** and **630**. The selector plane **620** is used to identify, for each pixel in the selector plane **620**, whether to use the pixel value found in the upper plane **610** or the lower plane **630** during reconstruction. In various exemplary embodiments, a "white" pixel in the selector plane **620**, i.e., a logical "0" value, means the pixel value should be taken from the corresponding pixel from the lower plane **630**, and a "black" pixel in the selector plane **620**, i.e. a logical "1" value, means that the pixel value should be taken from the corresponding pixel from the upper plane **610**.

It should also be appreciated that, although FIG. **5** shows only one upper plane and one lower plane, as discussed above, any number of upper-lower pairs of planes may be used. In particular, the planes may be decomposed into upper plane, lower plane and a selector plane. Then, the upper plane can be decomposed into a lower sub-upper plane, an upper sub-upper plane, and a second selector plane. Furthermore, it should be appreciated that the systems and methods of this invention may be used with planes that are composed of multiple superimposed separations. Accordingly, the planes are not limited to any particular combinations, or number, of image data types.

The selector plane **620** may be a bitmap determined using a technique suitable for finding text or the like on an original pixel map. In one exemplary embodiment of this invention, the bitmap forming the selector plane **620** includes pixels having a 1 value to indicate the upper, or text plane, and a 0 to indicate the lower, or everything else, plane.

The upper plane **610**, which may be stored at a reduced resolution relative to the original pixel map, contains color, or gray scale, information of items such as text. The upper plane **610** is determined using the selector plane **620**. The lower plane **630**, which also may be stored at a reduced resolution relative to the original pixel map, is then determined. The lower plane **630** contains background color information, as well as continuous tone image information, i.e., information including tints as well as color or gray scale data, but with all small, text-like features deleted.

FIG. **6** shows one exemplary embodiment of a selector plane according to this invention. As shown in FIG. **6**, the selector plane **620** contains a first region **622**, or image area, that includes image data, a second region **624**, or non-image area, that includes non-image, or useless, data, and a third region **626** of the non-image area **624** that includes pixels close to the transition region between the data types of image and non-image areas **622** and **624**. During compression, the total bit-rate and distortion achieved will be given by the individual contributions of each of the first-third regions **622–626**. In various exemplary embodiments of this invention, to reduce the bit-rate and the distortion, image values are created for the non-image, or useless, data region, i.e., the second and third regions **624** and **626**.

Since the second region **624** contains non-image, or useless, data, the second region **624** does not contribute to the reconstructed image and its distortion is null. The image data in the first region **622** is beyond the reach of the padding procedure, being only dependent on the encoder **400** and on the input image data. Thus, in various exemplary embodiments of this invention, to reduce the bit-rate and distortion, the bit-rate in the second region **624** is reduced, while the distortion in the third region **626** is reduced. Accordingly, in these various exemplary embodiments, in order to reduce the bit-rate, all pixels are set in the second region **624** to a single constant pixel value. To reduce distortion in the third, transition, or boundary, region **626**, the boundary region **626** is smoothed to avoid discontinuities. This is done, for example, to reduce possible ringing artifacts. As smooth regions tend to be well encoded without spending too many bits, the bit-rate for the third region **626** is assumed to be insignificant.

In one exemplary embodiment of this invention, the constant value to be used for the pixels in the second region **624** is a predefined image value for all pixels in the second region **624**, such as 128, which is the middle of the pixel range. The value of 255, i.e., white, can also be used to emulate background paper. In other exemplary embodiments, the constant value is determined by identifying the second region **624** as a single connected region. In this case, for example, the average of the border pixels is

used as the constant value. It should be appreciated that, depending on the region to be smoothed, the pixel value to be used may be any constant value. Thus the smoothing operation of the system and method of this invention can be done with any such values.

FIGS. **7**(A)–(H) show an exemplary smoothing operation of the boundary region of the systems and methods of this invention. FIG. **7**(A) shows the original image signal output from the image source to the image segmenting portion **410**. The segmentor **411** obtains a selector plane which decides which pixels of the image data input to the image segmenting portion **410** are to be stored into the upper plane or the lower plane, FIG. **7**(B) shows the respective image and non-image regions **622** and **624** in the selector plane. As shown in FIG. **7**(C), a filled signal is obtained by replacing the non-image data region **624** with the constant fill value, such as the intermediate value **128**, as discussed above. As shown in FIG. **7**(D), the filtered region is obtained by filtering the non-image area. FIG. **7**(E) shows the smoothing result when the filled signal of FIG. **7**(C) is replaced by its filtered region in FIG. **7**(D).

As shown in FIG. **7**(E), a discontinuity may be obtained at the boundary point when filtering the boundary region **626** using the systems and methods of this invention. However, this discontinuity is half of the size of the discontinuity which existed before the boundary region **626** was filtered. The smoothing filter of the systems and methods of this invention decreases the edge by nearly half. This may occur because the filter approach of the systems and methods of this invention smoothes the discontinuity by blurring both sides of the edge, i.e., both ends would meet halfway along the edge. Because about one half is to be filtered in the above embodiment, FIG. **7**(E) shows the result as the discontinuity at half height.

FIGS. **7**(F)–(H) show exemplary embodiments of results of filtering the boundary region **626** using the systems and methods of this invention using uniform, Hamming and triangular coefficients, respectively. As shown in FIGS. **7**(F)–(H), by filtering using these exemplary coefficients, further reductions in the depth of the discontinuity in the boundary region **626** can be obtained. This provides a smoother transition between the image region **622** and the non-image region **624**.

The filter approach of the systems and methods of this invention smoothes the discontinuity by blurring both sides of the edge using various embodiments of a segmented-filter. In one exemplary embodiment of the systems and methods of this invention, in determining the filtered signal, more weight is given to the samples in one region of the selector plane, such as the image data region **622**, and less weight is given for samples in the other region, i.e., the non-image data region **624**. Using the input signal, the selector plane signal, and the weight to be given to the regions, the filtered signal may be obtained.

One exemplary embodiment of a segmented-filter is:

$$y(n) = \frac{\sum_{k=-L}^{L} h(k,n)x(n+k)}{\sum_{k=-L}^{L} h(k,n)}; \text{ and}$$

$$h(k,n) = \begin{cases} 1 & \text{if } m(n+k)=0 \\ M+1 & \text{if } m(n+k)=1 \end{cases}$$

where:

x(n) is the input signal to be filtered;

y(n) is the filtered signal;

m(n) is the selector plane signal;

n is the index of the pixel;

L is the size of the filter; and

M is the weight given to the region, where M=0 is equivalent to not incorporating segmentation and biasing into the filtering process.

In another exemplary embodiment of the filter of the systems and methods of this invention, the segmented-filter is:

$$y(n) = \frac{\sum_{k=-L}^{L} h(k,n)x(n+k)}{\sum_{k=-L}^{L} h(k,n)}; \text{ and}$$

$$h(k,n) = \begin{cases} 1 & \text{if } m(n+k)=0 \\ Mf(k)+1 & \text{if } m(n+k)=1 \end{cases}$$

where:

x(n) is the input signal to be filtered;

y(n) is the filtered signal;

m(n) is the selector plane signal;

n is the index of the pixel;

L is the size of the filter;

M is the weight given to the region, where M=0 is equivalent to not incorporating segmentation and biasing into the filtering process; and

f(k) is a low-pass filter with decaying coefficients.

It should be appreciated that the above-outlined segmented-filtering processes are not limiting to this invention and that the systems and methods of this invention may be accomplished any filtering processes which smooth the discontinuity between regions by blurring both sides of the edge of the regions. In particular, any non-linear filter, such as, for example, morphological filters, can be used to smooth the edges of the non-image areas. Thus, by using these filters, reductions in the depth of the discontinuity in the boundary region can be obtained. This provides a smoother transition between the image region and the non-image region.

FIG. **8** is a flowchart outlining one embodiment of an image compression and decompression method according to this invention. Beginning in step **S1000**, control continues to step **S1100**, where electronic image data is generated from an original image. Then, in step **S1200**, the electronic image data is input from the image source.

It should be appreciated that, while the flowchart of FIG. **8** shows generating the electronic image data as part of the process, this step is not necessarily needed. That is, while the electronic image data can be generated by scanning an original image, or the like, the electronic image data could have been generated at any time in the past. Moreover, the electronic image data need not have been generated from an original physical image, but could have been created from scratch electronically. Accordingly, if electronic image data of the image is already available to the image source, step **S1100** can be skipped, with control continuing directly from step **S1000** to step **S1200**.

In step **S1300**, image data is segmented into planes of different data types. Then, in step **S1400**, compressed image data is generated from the image planes. Next, in step **S1500**, the compressed image data is transmitted, and pos-

sibly stored before being transmitted, to a device for decompressing the compressed image data.

In step **S1600**, the compressed image data is decompressed. Subsequently, in step **S1700**, the decompressed image-area data is recombined to form a decompressed image. Next, in step **S1800**, the decompressed and recombined image data is output. Then, in step **S1900**, the control routine ends.

FIG. **9** outlines in greater detail the image segmenting process of step **S1300**. Beginning in step **S1300**, control continues to step **S1310**, where the first plane is selected as the current plane. Then in step **S1320**, the non-image areas in the current plane are identified. Control then continues to step **S1330**.

In step **S1330**, the first identified non-image area is selected as the current non-image area. Next, in step **S1340**, a one or more image fill values to fill the current non-image areas are determined. Then, in step **S1350**, the pixels values in the current non-image area is replaced by determined image fill value or values. Control then continues to step **S1360**.

In step **S1360**, the current non-image area with the one or more image fill values is filtered. Then, in step **S1370**, the current filled non-image area is replaced with the filtered values. Next, in step **S1380**, a determination is made whether all non-image areas have been analyzed. If not, control returns to step **1330**. Otherwise, control continues to step **S1390**.

In step **S1390**, a determination is made whether all planes have been analyzed. If not, control returns to step **1310**. Otherwise, control continues to step **S1395**. In step **S1395**, control returns to step **S1400**.

As shown in FIG. **1**, the encoder **400** is implemented on a programmed general purpose computer. However, the encoder **400** can also be implemented on a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA or PAL, or the like. In general, any device, which is capable of implementing step **S1300** of FIGS. **8** and **9**, can be used to implement the encoder **400**.

Similarly, though FIG. **1** shows the decoder **500** being implemented on a programmed general purpose computer, the decoder **500** can also be implemented on a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA or PAL, or the like. In general, any device, which is capable of implementing step **S1600** of FIG. **8** can be used to implement the decoder **500**.

This invention has been described in connection with the preferred embodiments. However it should be understood that there is no intent to limit the invention to the embodiments described above. On the contrary, the intent to cover all altematives, modification, and equivalents as may be included within the spirit and scope of the invention.

What is claimed is:

1. A method for compressing an image comprising:

segmenting the image into a plurality of image planes including a selector plane and a first plane and a second plane, each image plane comprising a plurality of pixels, the selector plane indicating whether a pixel is to be stored in the first plane or the second plane;

determining non-image areas in each of the first plane and the second plane based on the selector plane;

generating false image data for each determined non-image area;

replacing the non-image areas with the false image data; and

compressing each image plane using an appropriate compression technique for that image plane.

2. The method of claim **1**, wherein, for each determined non-image area, generating false image data for that non-image area comprises generating the false image data based on image data of at least one image area adjacent that non-image area.

3. The method of claim **1**, further comprising filtering the generated false image data.

4. The method of claim **3**, wherein replacing the non-image areas with false image data comprises replacing the non-image areas with the filtered false image data.

5. The method of claim **3**, wherein filtering the generated false image data for a non-image area comprises reducing a discontinuity between an image data value of that false image data and an image data value of at least one image area adjacent that non-image area.

6. The method of claim **1**, wherein the compression technique for one of the first and second planes is one of JPEG, MMR, MR, MH, JBIG, JPEG2000 and JBIG2.

7. The method of claim **1**, wherein segmenting the image into image planes further comprises segmenting one of the first and second planes into a third plane and a fourth plane.

8. The method of claim **1**, wherein compressing each image plane containing the false image data using the appropriate compression technique results in decreased distortion relative to compressing an image plane that does not contain the false image data.

9. The method of claim **1**, wherein compressing each image plane containing the false image data using the appropriate compression technique results in increased compression relative to compressing an image plane that does not contain the false image data.

10. The method of claim **1**, wherein an image data value of the false image data is predetermined.

11. The method of claim **1**, wherein, for each determined non-image area, generating false image data for that non-image area comprises generating the false image data based on a boundary region between that non-image area and at least one image area adjacent that non-image area.

12. An image compression system, comprising:

an image segmentor that segments the image into a plurality of image planes including a selector plane and a first plane and a second plane, each image plane comprising a plurality of pixels, the selector plane indicating whether a pixel is to be stored in the first plane or the second plane;

a plane analyzer that determines non-image areas in each of the first plane and the second plane based on the selector plane;

a false image data generator that generates false image data for each determined non-image area and replaces the non-image areas with the false image data; and

a compressor that compresses each image plane using an appropriate compression technique for that image plane.

13. The image compression system of claim **12**, wherein the false image data generator generates, for each determined non-image area, false image data for that non-image area based on image data of at least one image area adjacent that non-image area.

14. The image compression system of claim 12, further comprising a filter that filters the generated false image data.

15. The image compression system of claim 14, wherein the false image data generator replaces the non-image areas with the filtered false image data.

16. The image compression system of claim 14, wherein the filter reduces a discontinuity between an image data value of that false image data and an image data value of at least one image area adjacent that non-image area.

17. The image compression system of claim 12, wherein the compression technique for one of the first and second planes is one of JPEG, MMR, MR, MH, JBIG, JPEG2000 and JBIG2.

18. The image compression system of claim 12, wherein the image segmentor further segments one of the first and second planes into a third plane and a fourth plane.

19. The image compression system of claim 12, wherein an image plane compressed using the image compression system has a reduced distortion ratio.

20. The image compression system of claim 12, wherein an image plane compressed using the image compression system has an increased compression ratio.

21. The image compression system of claim 12, wherein an image data value of the false image data is predetermined.

22. The image compression system of claim 12, wherein, for each determined non-image area, the false image data generator generates the false image data based on a boundary region between that non-image area and at least one image area adjacent that non-image area.

23. An image compression system, comprising:

means for segmenting the image into a plurality of image planes including a selector plane and a first plane and a second plane, each image plane comprising a plurality of pixels, the selector plane indicating whether a pixel is to be stored in the first plane or the second plane;

means for determining non-image areas in each of the first plane and the second plane based on the selector plane;

means for generating false image data for each determined non-image area;

means for replacing the non-image areas with the false image data; and

means for compressing each image plane using an appropriate compression technique for that image plane.

24. The image compression system of claim 23, wherein, for each determined non-image area, the means for generating false image data generates false image data for that non-image area based on image data of at least one image area adjacent that non-image area.

25. The image compression system of claim 23, further comprising means for filtering the generated false image data.

26. The image compression system of claim 25, wherein the means for replacing the non-image areas with false image data replaces the non-image areas with the filtered false image data.

27. The image compression system of claim 25, wherein the means for filtering the generated false image data for a non-image area reduces a discontinuity between an image data value of that false image data and an image data value of at least one image area adjacent that non-image area.

28. The image compression system of claim 23, wherein the compression technique for one of the first and second planes is one of JPEG, MMR, MR, MH, JBIG, JPEG2000 and JBIG2.

29. The image compression system of claim 23, wherein the means for segmenting the image into image planes further segments one of the first and second planes into a third plane and a fourth plane.

30. The image compression system of claim 23, wherein an image plane compressed using the image compression system has a reduced distortion ratio.

31. The image compression system of claim 23, wherein an image plane compressed using the image compression system has an increased compression ratio.

32. The image compression system of claim 23, wherein an image data value of the false image data is predetermined.

33. The image compression system of claim 23, wherein, for each determined non-image area, the means for generating false image data for that non-image area generates the false image data based on a boundary region between that non-image area and at least one image area adjacent that non-image area.

* * * * *

US006285889B1

(12) **United States Patent**
Nykänen et al.

(10) **Patent No.: US 6,285,889 B1**
(45) **Date of Patent: *Sep. 4, 2001**

(54) **INFORMATION OUTPUT SYSTEM, METHOD FOR OUTPUTTING INFORMATION AND TERMINAL DEVICES FOR OUTPUTTING INFORMATION VIA MOBILE COMMUNICATION NETWORK**

(75) Inventors: **Petri Nykänen**; **Lasse Siitonen**, both of Tampere (FI)

(73) Assignee: **Nokia Mobile Phones Limited**, Espoo (FI)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/905,557**

(22) Filed: **Aug. 4, 1997**

(30) **Foreign Application Priority Data**

Aug. 8, 1996 (FI) 963116

(51) **Int. Cl.**[7] **H04Q 7/20**

(52) **U.S. Cl.** **455/557**; 455/517; 455/414; 455/466

(58) **Field of Search** 455/517, 557, 455/422, 554, 457, 403, 414, 466, 556; 395/200.57; 358/448

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,041,918		8/1991	Ishida et al.	358/442
5,175,762		12/1992	Kochis et al.	379/100
5,732,074	*	3/1998	Spaur et al.	455/457 X
5,809,415	*	9/1998	Rossmann	455/422
5,848,413	*	12/1998	Wolff	358/448 X
5,867,661	*	2/1999	Bittinger et al.	395/200.57
6,055,441	*	4/2000	Wieand et al.	455/557

FOREIGN PATENT DOCUMENTS

0 538 059	4/1993	(EP) .
0 574 222	12/1993	(EP) .
WO 95/03669	2/1995	(WO) .

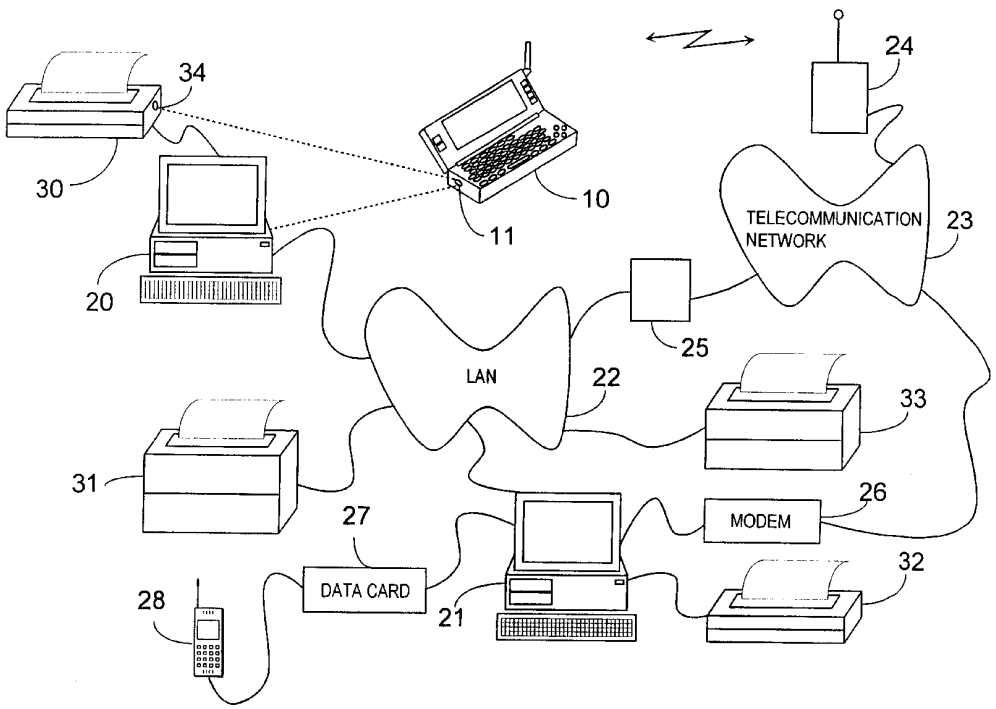
* cited by examiner

Primary Examiner—William Trost
Assistant Examiner—Philip J. Sobutka
(74) *Attorney, Agent, or Firm*—Perman & Green LLP

(57) **ABSTRACT**

In an information output system according to the invention data **(48)** processed by an application **(40)** is processed in a terminal device **(10)** into a data transfer format **(44)** independent of the application **(40)** using a specific format generator **(42)**. After this the data **(48)** is transferred to a second terminal device **(20)** via mobile communication network. In the receiving terminal device a specific data transfer application decodes the data **(48)** in the independent data format **(44)** into a form suitable for the output devices, and automatically transfers it to the output devices for outputting.

15 Claims, 6 Drawing Sheets



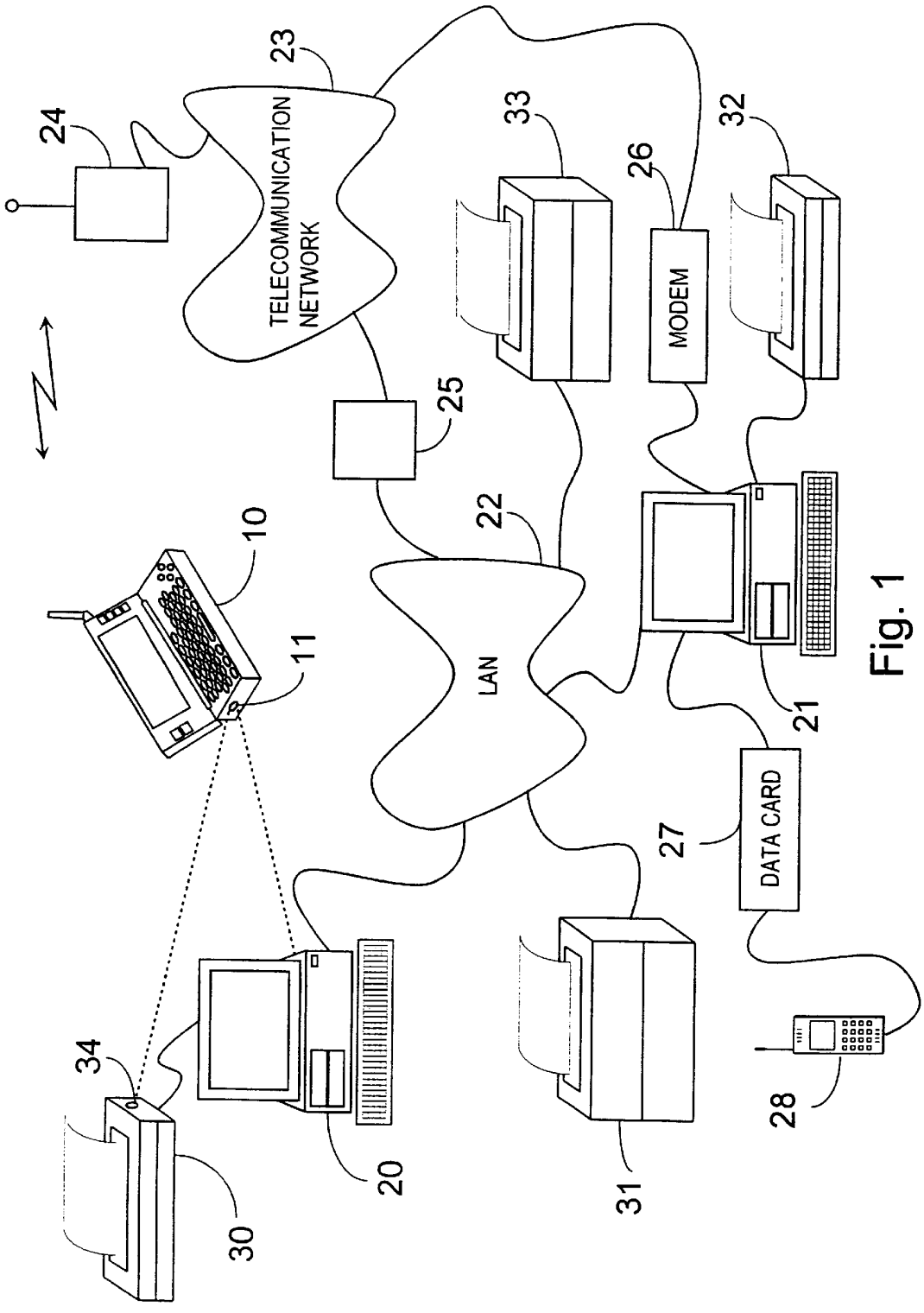

TELECOMMUNICATION NETWORK
23
24
25
26
MODEM
32
33
22
LAN
10
11
21
27
DATA CARD
28
20
30
31
34

Fig. 1

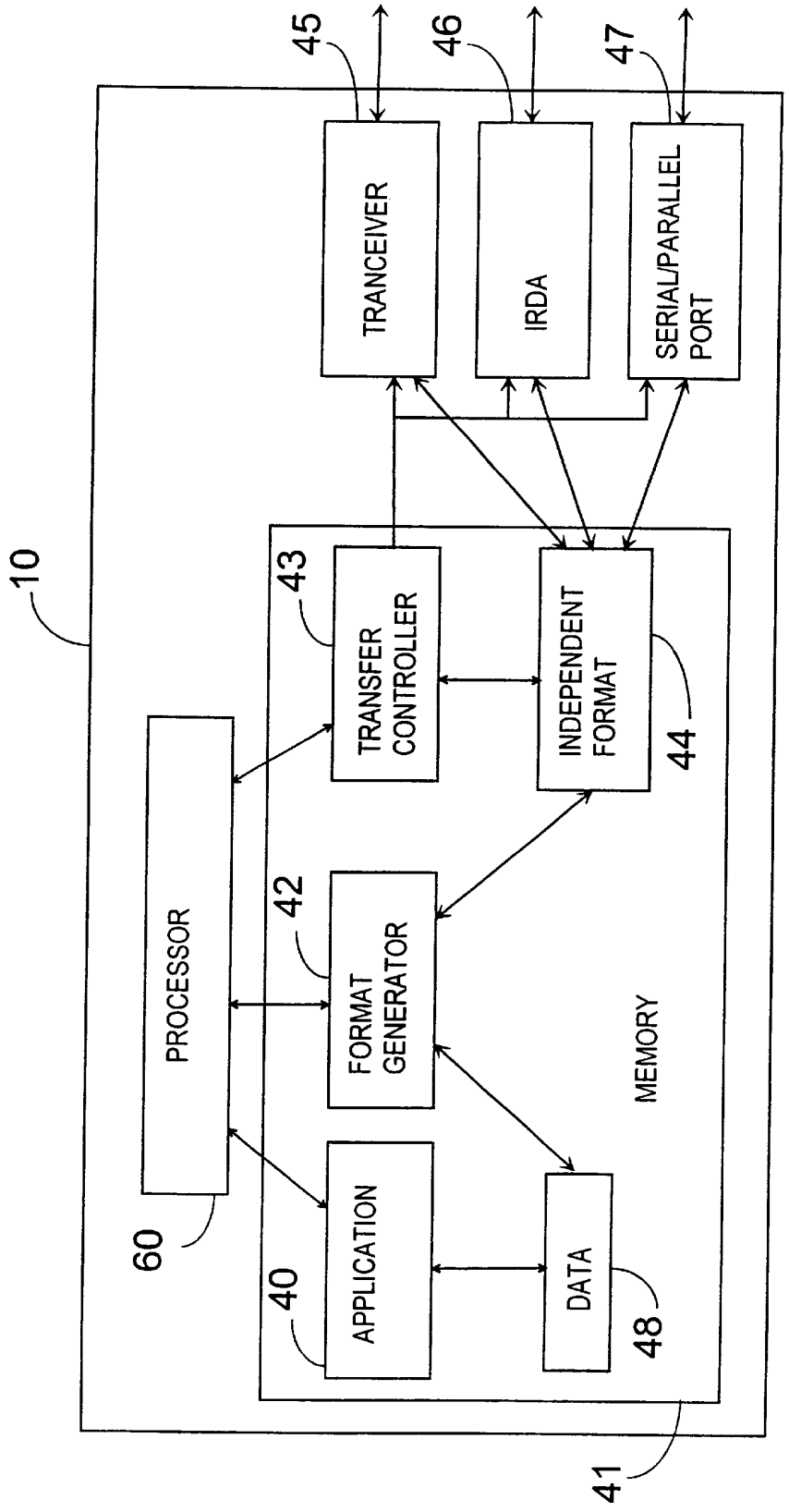

10
45
46
47
TRANCEIVER
IRDA
SERIAL/PARALLEL PORT
43
TRANSFER CONTROLLER
INDEPENDENT FORMAT
44
PROCESSOR
60
42
FORMAT GENERATOR
MEMORY
40
APPLICATION
DATA
48
41

Fig. 2A

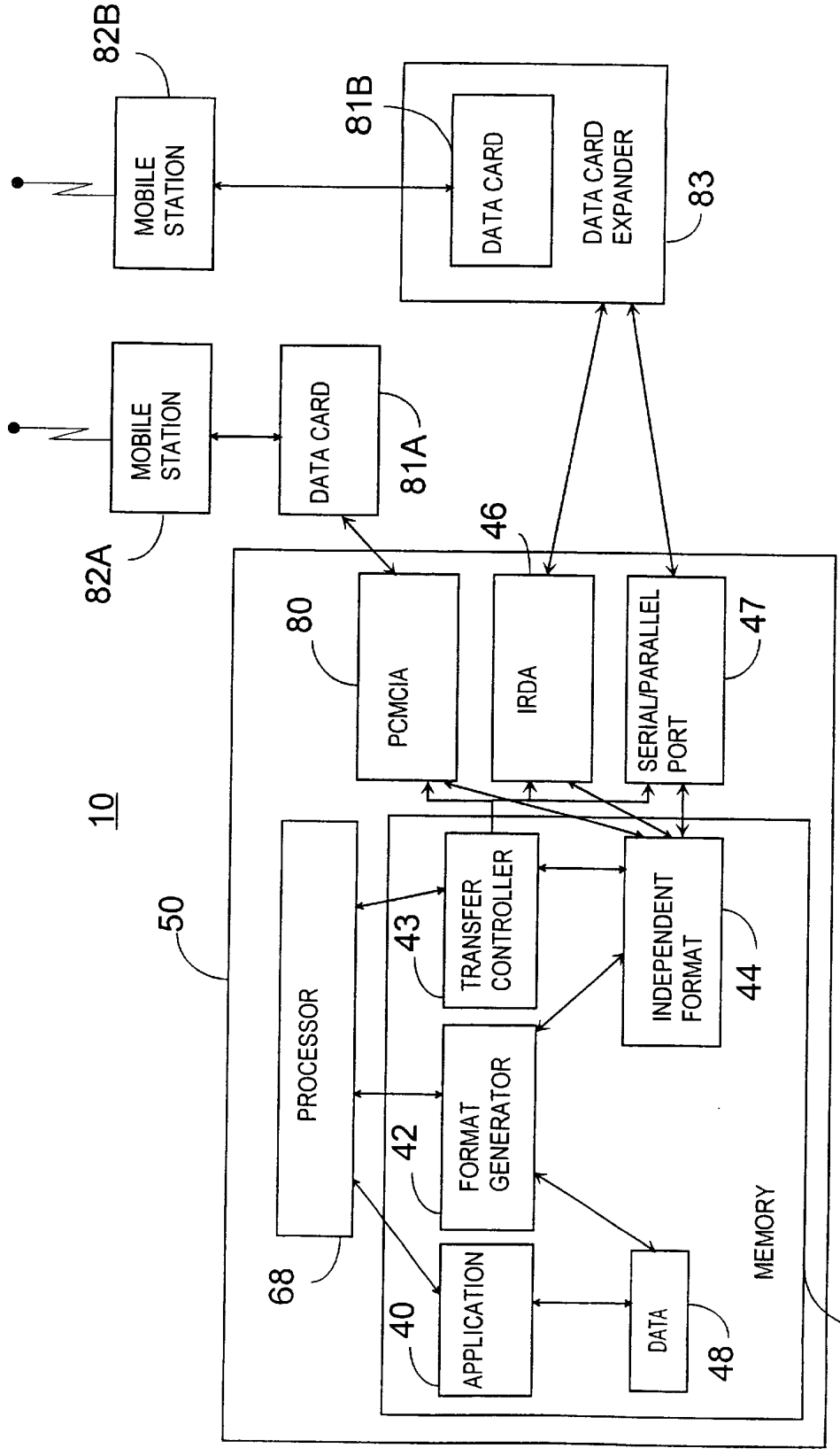

82B
MOBILE STATION
81B
DATA CARD
DATA CARD EXPANDER
83
MOBILE STATION
DATA CARD
81A
82A
46
80
PCMCIA
IRDA
SERIAL/PARALLEL PORT
47
10
50
PROCESSOR
TRANSFER CONTROLLER
43
INDEPENDENT FORMAT
44
FORMAT GENERATOR
42
68
APPLICATION
40
DATA
48
MEMORY
41

Fig. 2B

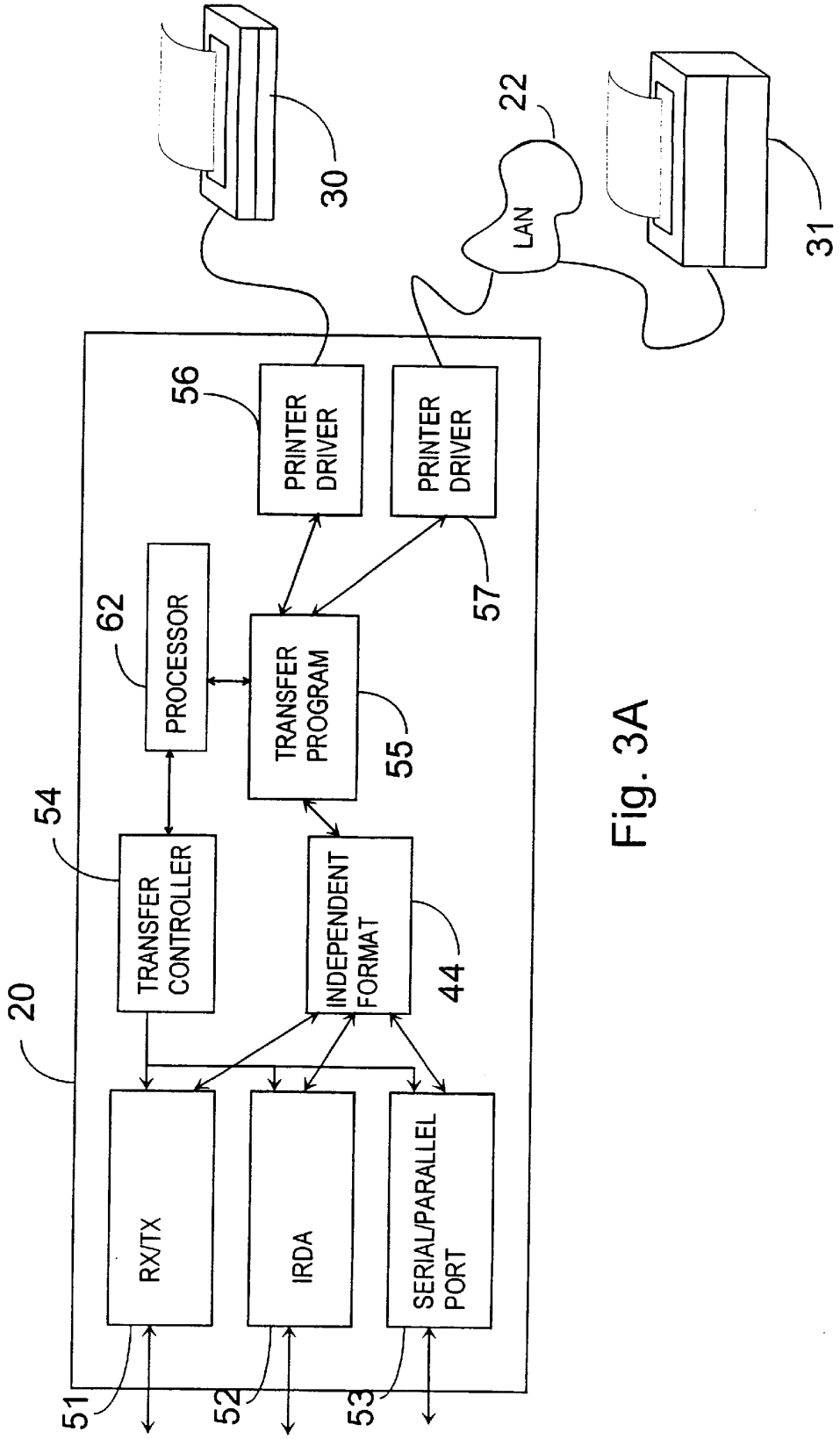

30
22
LAN
31
56
PRINTER DRIVER
PRINTER DRIVER
57
62
PROCESSOR
TRANSFER PROGRAM
55
54
TRANSFER CONTROLLER
INDEPENDENT FORMAT
44
20
RX/TX
IRDA
SERIAL/PARALLEL PORT
51
52
53

Fig. 3A

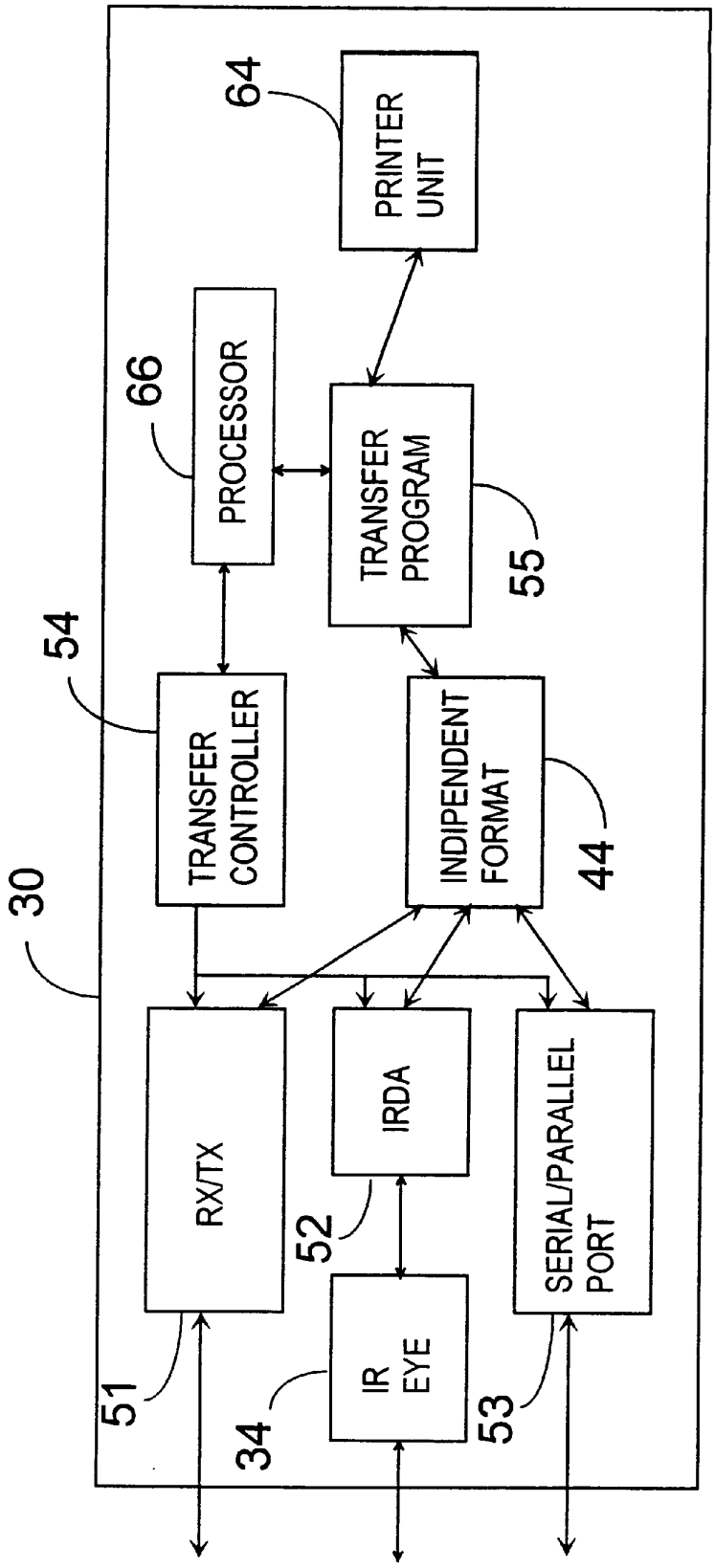

64
PRINTER UNIT
66
PROCESSOR
TRANSFER PROGRAM
55
54
TRANSFER CONTROLLER
INDIPENDENT FORMAT
44
30
RX/TX
IRDA
SERIAL/PARALLEL PORT
52
IR EYE
51
34
53

Fig. 3B

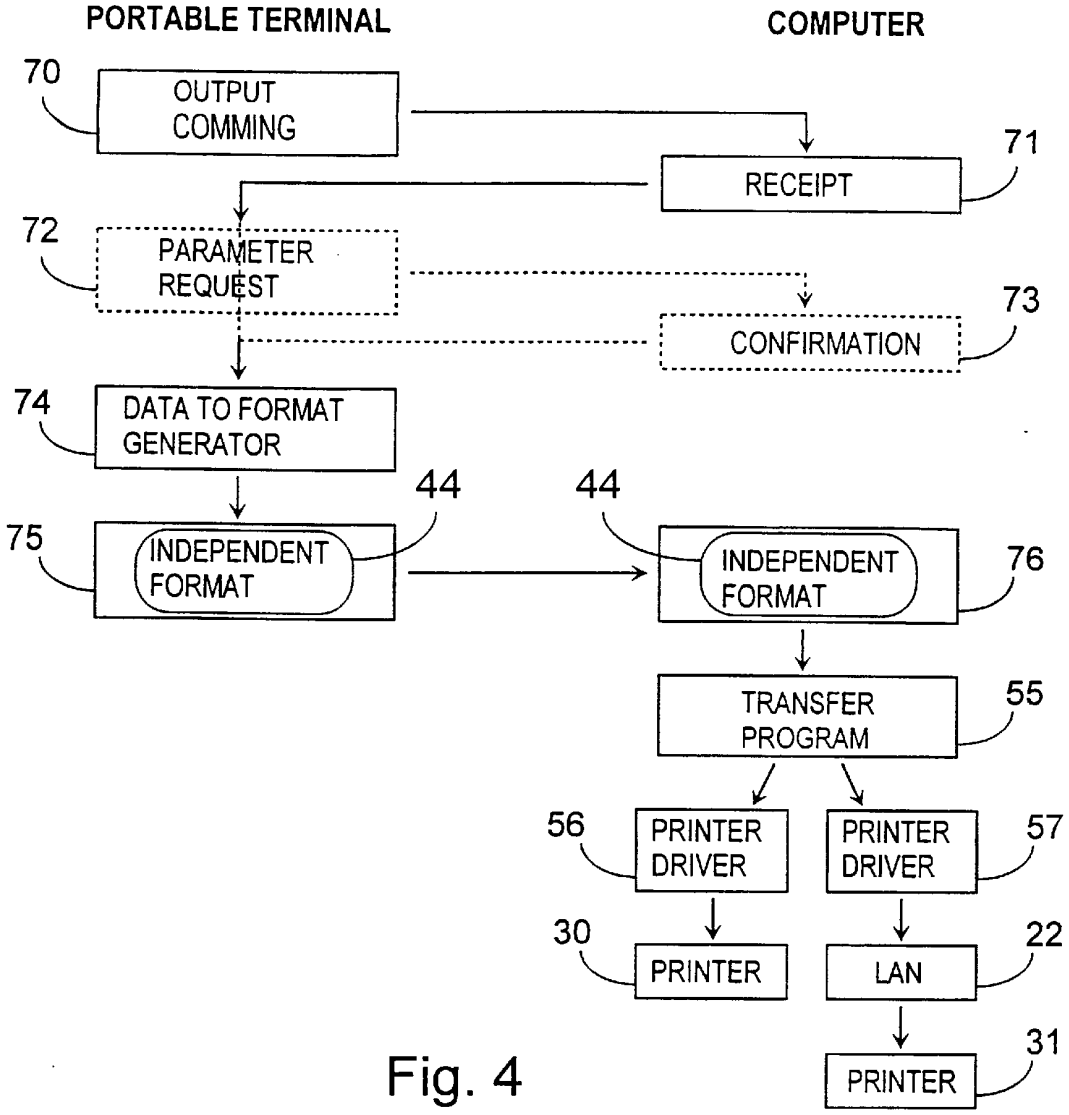

PORTABLE TERMINAL
COMPUTER
70
OUTPUT COMMING
71
RECEIPT
72
PARAMETER REQUEST
73
CONFIRMATION
74
DATA TO FORMAT GENERATOR
44
44
75
INDEPENDENT FORMAT
INDEPENDENT FORMAT
76
TRANSFER PROGRAM
55
56
PRINTER DRIVER
PRINTER DRIVER
57
30
PRINTER
LAN
22
PRINTER
31

Fig. 4

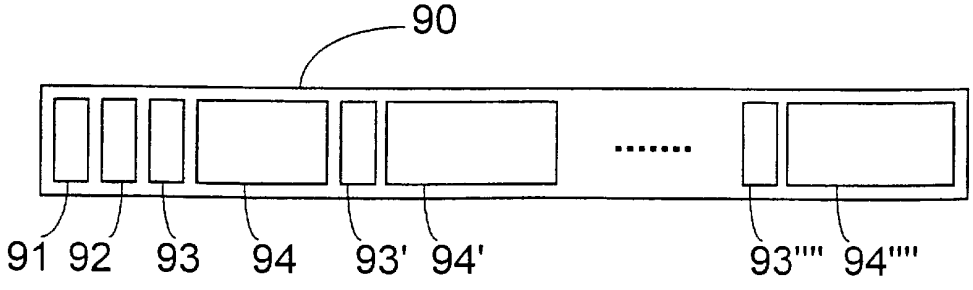

90
91
92
93
94
93'
94'
93''''
94''''

Fig. 5

INFORMATION OUTPUT SYSTEM, METHOD FOR OUTPUTTING INFORMATION AND TERMINAL DEVICES FOR OUTPUTTING INFORMATION VIA MOBILE COMMUNICATION NETWORK

FIELD OF THE INVENTION

The present invention relates to a method for outputting information, in which data is transferred between a first terminal device and a second terminal device via mobile communication network, and in which first terminal device an application is used, the data processed by which is converted into a form suitable for the data transfer connection and is transferred to a second terminal device, in which the data is received and processed into a form suitable for outputting devices for outputting.

BACKGROUND OF THE INVENTION

In the modern information society people are more and more dependent on telecommunication and data processing services. Utilization of services has been experienced to be so important that people do no more want to be dependent on the limitations of traditional office environment. That is why many kinds of portable telecommunication terminal devices have been developed, a very popular representant of which is a portable microcomputer. In order to utilize various telecommunication services a portable microcomputer can, using auxiliary devices, be connected to telecommunication networks e.g. utilizing the data channel provided by GSM (Global System for Mobile Communications) -mobile telephone system. Telecommunication terminal devices slightly different in their functions are so called electronic notebooks, which are commonly called PDA (Personal Digital Assistant) -devices. These devices offer typically calendar services and other services increasing the efficiency of utilization of time.

A telecommunication terminal device which is even more advanced than the above mentioned ones is a so called multi-service mobile station. In a multi-service mobile station features have been collected from a portable microcomputer, a mobile station and a PDA-device. A multi-service mobile station is a small but versatile wireless telecommunication terminal device. This kind of device can be used as a mobile station, because it comprises e.g. the functions of a GSM mobile telephone, and with it one can send and receive facsimile messages, browse WWW (World Wide Web) -pages through Internet and create and edit text files. One example of this kind of multi-service mobile station is NOKIA 9000 Communicator manufactured by Nokia Mobile Phones Inc. All above mentioned devices have been optimized for a certain purpose, and they facilitate the utilization of various telecommunication and data processing services outside the office.

While using the above described telecommunication terminal devices one usually meets the following problem: one wishes to output information processed with the telecommunication terminal device on paper. While moving outside an office, a personal printer is not available very often. In order to be able to connect a telecommunication terminal device to a printer, a printer driver specific to the printer in question is required, which driver converts the data processed with the telecommunication terminal device into a form suitable for the printer. This means in practice that several different printer drivers have to be installed in a telecommunication terminal device in order to be able to use printers of different types. This can be somewhat successfully done in a portable computer which has a big mass storage. As is generally known, computers usually have a large number of printer drivers for various printers. Still, even then one may meet the problem that the necessary printer driver is not found in the mass memory of the computer when it is needed. This happens especially in foreign environment when it is impossible to predetermine an available printer. The version of an available printer driver can also be so old that printing is not successful when a more recent printer version is used. These problems are easily met e.g. while traveling, when one attempts to connect a computer to such a printer which one has not used previously.

Several printer drivers as such are a big problem for pocket-size, light telecommunication terminal devices. Their memory capacity is small, and as big a share of the available memory as possible is desired to be reserved for the information processed by the user. In addition to that, increasing memory capacity increases manufacturing cost. On the other hand, if memory capacity has to be increased because of the number of printer drivers, the power consumption of the memory is normally increased, and through it also the total power consumption of the telecommunication terminal device, which on its behalf has an unfavorable effect on the operating time of battery powered telecommunication terminal devices.

Patent publication U.S. Pat. No. 5,041,918 describes a separate adapter unit connected to a fixed ISDN (Integrated Services digital Network) -line, which adapter is further connected to a computer and a printer. The adapter unit comprises among other things ISDN -interface circuits, a memory and a processor. By utilizing this adapter unit it is possible to use a computer and a printer as a facsimile device. The adapter unit can relay transfer data received from the computer through ISDN-network to another corresponding adapter unit, and from there further to a printer. The information may be a teletext, a facsimile message and in so called mixed form, which comprises text and pictures. If information is transferred using a protocol other than the data transfer protocol prior known from facsimile devices, we meet the following drawback: one has to determine in advance for each connection which data format is to be used for data transfer. For example, if the receiving terminal device cannot process data in mixed form, the sending terminal device has to be commanded manually into facsimile mode.

Patent publication U.S. Pat. No. 5,175,762 presents a remote outputting method which utilizes in a data transfer connection through telephone lines the picture coding standard prior known from normal facsimile devices. Printer commands required for outputting have though to be forwarded in connection with the message, which means that one has to know before transmitting, what kind of printer is available in the other end. This is laborious. In addition to that, the resolution used in facsimile devices does not make it possible to transfer pictures of good quality. In the system according to the publication it is also possible to transfer data in non-standard (NSF, Non-Standard Facility) format. NSF has been specified in Group 3 of CCITT. In this case files can be transferred in the original file format, specific to each application. The disadvantage of this system is that outputting requires that the receiving end has access to the same software with which the file was originally created in the transmitting end. It is often even required that the software must be of the same version. Even if both computers would have the same software, the outputting has to be executed manually, that is, someone must be present in

the receiving end to open the received file using said software and to give the Print-command.

SUMMARY OF THE INVENTION

Now an information outputting system utilizing a data transfer format independent of an application and an output device, and a telecommunication terminal devices have been invented, with which the above mentioned problems can be reduced. In the information output system according to the invention data intended for outputting is transferred between two telecommunication terminal devices or between a telecommunication terminal device and an output device. Telecommunication terminal devices may be either portable or fixed. The telecommunication terminal devices can be connected to a telecommunication network through wired connection or the connection is established wirelessly using either a radio connection or an infrared connection. Most preferably the benefits of the information output system according to the invention become evident while transferring data by radio from a portable telecommunication terminal device to a fixed telecommunication terminal device connected to an output device. Because of clarity and simplicity in the following one telecommunication terminal device is called portable terminal device and the other is called fixed device.

In the information output system according to the invention the information (data) from portable terminal device desired to be output is first converted into a data transfer format independent of application, which format is also independent of the physical realization of the data outputting system. After this the data desired to be output is transferred in the data transfer format independent of application from the portable terminal device to a fixed device connected to the desired telecommunication network, e.g. to a personal computer.

In order to execute the outputting the processor of the fixed device runs a device-independent data transfer program. The purpose of the data transfer program is to forward the data further for outputting. The transfer program recognizes that the received data is in the application-independent data transfer format e.g. from an output code added in the transferred data. It is also possible to define an individual data address for the application-independent data transfer format, in which case the transfer program automatically assumes that data coming from this address is in said application-independent data transfer format. In this case there is no need in the transmitting end to add any additional outputting code in the data to be transferred. Having received the data in the application-independent data transfer format, the transfer program decodes it into a format suitable for operating system and the output device. The output device can be, e.g., a monitor or a printer. If it is a printer the transfer program directs the decoded data further to the printer driver, from which it is further transferred to the printer controlled by the operating system. Depending on the type off the output device the data is transferred to the corresponding driver. The outputting from the fixed device to the output device takes place in a way similar to as if the data to be output would have been created in the fixed device. The printer driver and the output device do not even know that the data to be output is originated outside the fixed device. A difference with said patent publication U.S. Pat. No. 5,175,762 is among other things, that preferably the software with which the data was originally created in the portable terminal device, is not required in the fixed device.

Because the application-independent data transfer format used in connection with the data outputting system according to the invention is also hardware-independent, the one and same transfer program can preferably be used in all for outputting suitable fixed devices and networks, independent of the printer drivers or other drivers of the output devices and independent of the types of output devices. For the same reason the types of available output devices need not be known even when the transfer program is installed in a fixed device. The transfer program can preferably be installed in the fixed device from a diskette, but is also possible to install it through the telecommunication network. If the outputting device selected for use understands said independent data transfer format, the desired data can be transferred in the information output system according to the invention in the application-independent data transfer format from the portable terminal device directly to the output device.

The transfer of the application-independent data transfer format from a portable terminal device to an output device or to a fixed device connected to the same telecommunication network with the output device can preferably be carried out by radio, using data transfer over an infrared connection, or over a wire connection. It is possible to realize the wire connection using optical fiber or a cable providing galvanic connection.

The information output system according to the invention is characterized in, that the data conversion means of a first terminal device comprise a format generator which has been arranged to process the data to be transmitted into an application-independent data transfer format, means of data transmission have been arranged to transfer data in said independent data transfer format to another terminal device, the other terminal device comprises means for format decoding, which have been arranged to decode the data in said independent data transfer format into a form suitable for data outputting devices for outputting.

It is characteristic of the method for outputting information according to the invention, that for data transfer the data is processed into a data transfer format which is independent of application and outputting devices.

The information output system according to the invention utilizing application-independent data transfer format is described in detail in the following referring to enclosed figures.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** presents an information output system according to the invention utilizing an application-independent data transfer format,

FIG. **2**A presents an embodiment of a portable terminal device according to the invention utilizing an application-independent data transfer format, in form of a block diagram,

FIG. **2**B presents another embodiment of a portable terminal device according to the invention utilizing an application-independent data transfer format, in form of a block diagram,

FIG. **3**A presents the structure of a terminal device according to the invention utilizing an application-independent data transfer format and connected to telecommunication network and outputting devices, in form of a block diagram,

FIG. **3**B presents the structure of an output device according to the invention utilizing an application-independent data transfer format and capable of directly outputting the application-independent data transfer format, in form of a block diagram,

FIG. **4** presents an example of the communication between a portable terminal device and a fixed device according to the invention utilizing an application-independent data transfer format in connection with outputting, and

FIG. **5** presents an example of a transfer frame, in which data in an application-independent data transfer format can be transferred between terminal devices according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following the information output system according to the invention is explained using some preferable embodiments. However, nothing limits using the invention also in other systems different in their realization but belonging to the scope of the invention.

FIG. **1** presents a preferable information output system according to the invention utilizing application-independent data transfer format. It comprises portable terminal device **10**, which is connected to a telecommunication network, in this case local network **22**. A typical and very common local network **22** is the LAN (Local Area Network) -network. In FIG. **1** two computers **20**, **21** and two printers **31**, **33** have been connected to local network **22**. Fixed devices connected to local network **22**, that is computers **20**, **21** and outputting devices **31**,**33** can transfer data between each other. This e.g. makes it possible that computer **20** can, when required output data using both printer **31** and printer **33**. Additionally, computers **20**, **21** connected to the telecommunication system have been connected to personal printers **30**, **32**. These personal printers **30**, **32** are used mainly only locally, but if required also the capacity of personal printers **30**, **32** can be, as prior known, shared between all computers **20**, **21** connected to the local network, as e.g. in Microsoft Windows for Workgroups environment.

Portable terminal device **10** is connected to local network **22** e.g. utilizing data transfer over infrared connection between infrared transceiver element **11** installed in portable terminal device **10** and an infrared transceiver element (not shown in the figure) installed in computer **20**. Alternatively the connection can be realized also over an optical or galvanic wire connection. When portable terminal device **10** is connected to local network **22** through computer **20**, all printers **31**, **33** connected to local network **22** are available for portable terminal device **10**. If desired, it is also possible to execute the outputting using personal printers **30**. The outputting can also be successfully carried out using printer **32**, provided that the resources of printer **32** have been set to be shared between other computers **20** connected to the local network and that both computer **21** and printer **32** are ready for operation.

In the information output system according to the invention utilizing application-independent data transfer format **44**, portable terminal device **10** need preferably not know the types of printers **31**, **33** connected to local network **22** nor the types of personal printers **30**, **32** connected to computers **20**, **21**. For this reason no printer drivers are needed in portable terminal device **10** and the settings of portable terminal device **10** need not be changed while selecting printer **30**, **31**, **32**, **33**. The printer selection is executed in computer **20** connected to the local network, in which computer **20** the selection is carried out by data transfer program **55** (see FIGS. **3**A and **3**B) installed in computer **20**. The default printer of the computer can preferably be used as a printer, in which case data transfer program **55** need not select a printer. If for example a video monitor or a video projector is used as a output device instead of a printer, the settings are made in a similar manner under the control of computer **20** independently of the portable terminal device **10**. Data transfer program **55**, its functions, and application-independent data transfer format **44**, utilized by data transfer program **55**, are described in more detail in connection with the explanation of FIGS. **2**A, **2**B, **3**A and **3**B.

When an infrared connection or a physical wire connection is used, a high transfer speed can be used in the transfer connection in order to make the outputting faster. If portable terminal device **10** is used outside office, the outputting is carried out wirelessly using a radio connection. This is one of the most preferable fields of use of the invention. An example of a wireless data transfer method is the data channel of the GSM (Global System for Mobile Communications) mobile communication system prior known to a person skilled in the art. In order to establish a data transfer connection from portable terminal device **10** through base station **24** and telecommunication network **23** to local network **22** it is prior known to arrange adapter **25** between telecommunication network **23** and local network **22**. A data transfer connection realized using local network **22** and adapter **25** also enables high speed data transfer to other networks. This type of realization is particularly suitable for large companies which need high data transfer capacity and several simultaneous data transfer connections.

If computer **21** is equipped with modem **26**, portable terminal device **10** can be connected directly to computer **21** without local network **22** and adapter **25**. In this case portable terminal device **10** opens a radio connection to base station **24**, which is connected to telecommunication network **23**. The connection from telecommunication network **23** to computer **21** is established over modem **26** using conventional telephone lines. It is evident to a person skilled in the art that establishing a connection from computer **21** to portable terminal device **10** takes place according to the same principle but in reversed order. A connection over modem **26** is an easy and economical way to realize and it is particularly suitable for private users and small enterprises. Using the same modem **26**, computer **21** can be connected to also other telecommunication networks, such as Internet, according to the same principle. Modem **26** can also be a so-called card-modem, in which case it is installed inside computer **21**. It is also possible to realize a connection from portable terminal device **10** to computer **21** directly over radio connection without modem **26** and telecommunication network **23**. This can be realized in such a way that computer **21** is equipped with data card **27**, which is connected to its PCMCIA (Personal Computer Memory Card International Association) -bus, and a mobile station **28**. In this case it is possible to realize the data connection from portable terminal device **10** to mobile station **28** using e.g. the above mentioned data channel provided by the GSM-system.

FIG. **2**A presents the structure of portable terminal device **10** according to the invention utilizing application-independent data transfer format **44**, in form of a block diagram. Portable terminal device **10** comprises at least one application **40**. Application **40**, as well as even eventual other applications, has been stored in form of programs in memory **41** of portable terminal device **10**. Application **40** may be e.g. a word processing application, a facsimile service or an Internet-service. Processor **60** of portable terminal device **10** handles and processes data **48** of application **40**, which data is in a form characteristic of application **40**, in memory **41** of portable terminal device **10**. If

there are several applications **40**, e.g. a word processing program, a facsimile service, a figure drawing application or an Internet-service, data **48** stored in memory **41** by each application **40** is not necessarily in the same form, or format. The reason of the differences between the formats is e.g. the accuracy or resolution used by application **40** when saving text or figures, or whether application **40** saves data **48** in text format (e.g. as standard ASCII-characters, in which case each character is represented by eight bits) or in a form characteristic of application **40**, e.g. in binary form.

An essential component of the data transfer system according to the invention utilizing application-independent data transfer format **44** is format generator **42**. It is possible to realize it e.g. utilizing a program stored in memory **41** of portable terminal device **10**, which program is executed by processor **60**. Format generator program **42** can be built-in in portable terminal device **10**, but preferably it can also be installed afterward e.g. from a diskette, if the operating system of portable terminal device **10** and the structural realization facilitate it. Format generator **42** converts data **48** in original format characteristic of application **40** into format **44** which is independent of application **40**, which is here also called independent format **44** for shortness.

In format generator **42** the conversion from one format to the other is carried out using mathematical methods, which are characteristic of each data format. In the following an example of format conversion is described, in which a circle presented in mathematical form is converted into a bit map. It has to be emphasized that the structure of independent format **44** decides which is the preferable method of executing the format conversion. The conversion into a bit map need not be carried out often.

In the beginning the necessary part of two-dimensional memory space is reserved, in which part the bits of bit map to be generated are saved. The size of the picture page to be output, the resolution used in data transfer and the number of colors used determine the size of the memory space required. Let us assume for the sake of simplicity that the figure is black and white, and that logical "0" corresponds to white, and that logical "1" corresponds to black. To begin with, all bits of the required part of two-dimensional memory space are set to "0" corresponding a white picture element. If the original circle has been presented utilizing the coordinates of the center, the radius of the circle and line thickness, the conversion into a bit map is started at the center point. The coordinates of the center correspond with a certain memory block of the two-dimensional memory space, in this case one memory bit. By calculating two-dimensionally with sufficient calculation accuracy the distance in bits indicated by the radius, from the memory bit corresponding with the center of the circle to every direction, and by setting the nearest bits to "1" corresponding with black, the circle can be converted into a bit map. Using this method, the thickness of the circumference would be only approximately one bit (corresponds typically to one picture element or pixel when displayed on a screen), and accordingly with bigger line thickness several adjoining bits at the circumference are set to "1", that is black. The above format conversion is an example of the mathematical method, prior known to a person skilled in the art, by which information presented in a digital form can be transferred from one format into another.

If format generator **42** is capable of directly handling data **48**, application **40** need not necessarily participate in the processing of data **48** at all. This is the case e.g. when a file desired to be output is directly in form of a WMF (Windows Meta File) file according to Windows-environment. Application **40** need not in this case even be activated, but format generator **42** retrieves the WMF-file desired to be transferred directly from memory **41**.

In many cases format generator **42** is not capable of converting data **48** specific to application **40** directly to application-independent format **44**. This is due to the fact that most computer programs use for data saving a format specific of the application (proprietary), which format other computer programs cannot decode. In this case application **40** first carries out the conversion of data **48** specific of application **40** into a form suitable for format generator **42**, and processor **60** forwards the data to format generator **42**. There can be several of these data formats suitable for format generator **42**, for example WMF- and BMP- (bitmap) formats prior known from Windows-environment. Because processor **60** preferably carries out the functions of both application **40** and of format generator **42**, transfer of information between these two programs can be arranged effectively independent of the format.

Independent format **44** presents data **48** in such a form that in fixed device **20** (FIG. **3**A) receiving independent format **44**, independent of output devices such as printers **30**, **31**, monitors and their corresponding drivers **56**, **57**, data **48** can be decoded and further forwarded to the output devices. As an example of an output device in the following we use printers **30** and **31**. The outputting using printers **30** and **31** has been achieved in such a way that data transfer program **55** executed by processor **62** of fixed device **20** converts independent format **44** into a format suitable for printer drivers **56**, **57** for outputting. The conversion of data from one format into another itself is carried out applying mathematical methods corresponding with methods described above in connection with the explanation of format generator **42**.

Suitable formats for printer drivers **56**, **57** in the receiving end are e.g. the same WMF- and BMP-formats which can be used for data transfer in the transmitting end. The functions of format generator **42** and data transfer program **55** and outputting are described in more detail in the latter part of the specification. Both the transmitting and the receiving device have data transfer controllers **43** and **54** (FIGS. **2**A and **3**A) which together with processors **60** and **62** control data transfer between portable terminal device **10** and fixed device **20**. This information output system according to the invention utilizing independent format **44** has in its preferable embodiment access to three separate data transfer paths. Controlled by processor **60** of portable terminal device **10**, transfer controller **43** selects, if desired, for transfer path a radio channel, in which case data transfer from portable terminal device **10** to fixed device **20** takes place utilizing transceivers **45** and **51**. Transceivers **45** and **51** have been realized e.g. using radio components prior known from GSM mobile telephones. Transceiver **45** of portable terminal device **10** comprises in this case also the means of modulation required for GSM-data transfer.

Correspondingly, data transfer over infrared connection is provided for by infrared transceiver **46** and infrared transceiver **52**. Data transfer takes place e.g. using the IrDA-protocol of Infrared Data Association prior known to a person skilled in the art. If portable terminal device **10** and fixed device **20** have been connected using a physical wire connection, the data transfer is executed by data transfer controllers **43** and **54** utilizing preferably serial or parallel port **47** and serial or parallel port **53**.

FIG. **2**B presents another embodiment of the portable terminal device **10** according to the invention utilizing

independent data transfer format **44**. In it portable terminal device **10** comprises portable computer **50**, data card **81**A connected to its PCMCIA (Personal Computer Memory Card International Association) -bus and conventional mobile station **82**A. Alternatively is possible to use, instead of a portable computer, a traditional desktop computer, if it is equipped with PCMCIA-bus **80**. The conversion of data **48** handled by application **40** executed with processor **68** of the portable computer into independent format **44** is carried out as described in above chapters. A major difference with the embodiment presented in FIG. **2**A is that the radio components are not fixedly incorporated in portable terminal device **10**. When in the embodiment according to the invention presented in FIG. **2**B data is transmitted in independent format **44** by radio, it preferably is carried out utilizing data card **81**A connected to PCMCIA-bus **80** of portable computer **50**. Data card **81**A comprises among other things the data transfer protocols for transferring data between computer **50** and mobile station **82**A. Data card **81**A and mobile station **82**A are connected together with an interface cable. Alternatively data card **81**A to be connected to the PCMCIA-bus can be replaced with a particular radio module connected to the PCMCIA-bus, in which module the functions of data card **81**A and mobile station **82**A have been integrated in one module in order to save space and power.

If portable computer **50** does not have PCMCIA-bus **80** suitable for data card **81**A or if the bus is already occupied, the data connection can also be realized using infrared connection **46** or e.g. serial/parallel port **47**. In this case data card expander **83** will be required between portable computer **50** and data card **81**B. Data card expander **83** is connected to portable computer **50** either using infrared connection **46** or using an interconnection cable to serial/parallel port **47**. Data card expander **83** can be connected to data card **81**B and mobile station **82**B e.g. directly using connectors (not shown in the figure). The data card expander converts data **48** transferred in independent format **44** from infrared connection **46** or from serial/parallel port **47** into a form suitable for data card **81**B. From data card **81**B the data is forwarded to mobile station **82**B and further to base station **24**, which was the case also in the alternative described in the previous chapter.

In one outputting system according to the invention, which is utilizing independent format **44**, portable terminal **10** can be realized in such a way that format generator **42** presented in FIG. **2**B is placed in specific data card **81**A connected to PCMCIA-bus **80**. It is possible to realize format generator **42** as a program installed in the memory of data card **81**A (not shown in the figure), but it can be realized also integrated in an ASIC (Application Specific Integrated Circuit) -circuit. When data transfer over PCMCIA-bus **80** is carried out using a data transfer program which utilizes only API (Application Programming Interface) -commands characteristic of Windows environment and format generator **44** has been realized outside portable computer **50**, transfer of data in independent format **44** can preferably be realized without needing to install separate format generator application program **42** in portable computer **50**. In the most advanced systems the operating system is capable of automatically initializing a PCMCIA-card installed in a PCMCIA-card slot, and accordingly very easy installation is achieved using this method. If processing of data **48** into independent format **44** is carried out outside portable computer **50**, processor **68** of the portable computer is preferable not loaded by it, and the capacity of processor **68** can be utilized for other purposes.

FIG. **3**B presents the realization of output device **30** according to the invention as a block diagram. Output device **30** is equipped with infrared transceiver **34**, and accordingly it is capable of communicating with portable terminal device **10**. Output device **30** can directly decode independent data transfer format **44**, due to which computer **20** or local network **22** is preferably not needed. Processor **66** of output device **30** together with data transfer program **55** caters for decoding independent format **44** into a form suitable for output unit **64**. The structure of output unit **64** is prior known to persons skilled in the art, e.g. from laser printers or inkjet printers, and it outputs the desired information e.g. on paper. Before transferring actual data **48** processed into independent format **44** in portable terminal device **10**, portable terminal device **10** shown in FIGS. **2**A and **2**B and fixed devices **20** and **30** shown in FIGS. **3**A and **3**B can communicate with each other in order to determine the optimal parameters to be used at outputting. An example of communication between portable terminal **10** and fixed device **20**, of using independent format **44**, of using a direct outputting code, of transferring independent format **44**, of decoding independent format **44** into a form suitable for printer drivers **56**, **57**, and of the outputting itself is described in detail in the following utilizing FIGS. **4** and **5**. If no handshake operation alike the above described is carried out, both portable terminal device **10** and fixed devices **20** and **30** use default settings.

In block "output coming" **70** processor **60** of portable terminal device **10** sends, using transfer path **45**, **46**, **47** used by data transfer controller **43**, a message to fixed device **20**, in which message it indicates that it wants to output to output device **30**–**33** of fixed device **20**. Portable terminal device **10** needs preferably not to know anything about printers **30**–**33** themselves or about their settings. After receiving the message, processor **62** of fixed device **20** transmits acknowledgment message **71** to portable terminal device **10**. Processor **62** of fixed device **20** can include in acknowledgment message **71** the parameters of the available printers, such as the resolutions of the printers, the available paper sizes, the paper position (horizontal or vertical) and the size of the largest printable picture area. Based upon this information portable terminal device **10** can, when it so wishes, select the printer to be used or alternatively use the default printer. This kind of exchange of information between two terminals, in order to agree on the parameters to be used in the connection, is prior known to a person skilled in the art e.g. from modems.

After receiving acknowledgment **71** processor **60** of portable terminal device **10** can send to fixed device **20** parameter request **72**, in which it indicates which output device and which settings portable terminal device **10** wants fixed device **20** to use for outputting. If the settings included in acknowledgment message **71** correspond with the output settings of portable terminal device **10**, parameter request **72** need preferably not be sent at all, but the processor of portable terminal device **10** can direct data **48** processed by application **40** directly to format generator **42** for processing. This takes place in block **74**. If sending parameter request **72** was necessary, as e.g. when portable terminal device **10** wants to transmit using a resolution lower than what the output devices of fixed device **20** are capable of, fixed device **20** confirms the settings to be used with confirmation message **73**. If any problems occur during the possible handshake procedure, terminal devices **10**, **20** **30** can be set in a state in which they after the elapse of a certain predetermined time constant operate at default settings. In this way outputting is possible even if the handshake operation was unsuccessful.

In block **74** format generator **42** processes data **48** handled by application **40** into independent format **44** utilizing

processor **60**. Such a format is selected as independent format **44**, by use of which data transfer from portable terminal device **10** to fixed device **20** can be carried out as effectively as possible in current operating situation. As criteria for selection of independent format **44** can be used the amount of data to be transferred, the character of the information in data **48** or selected transfer path **45**, **46**, **47**. Such a format shall be used in data transfer, which is limited to the output settings which are determined in accordance with above described blocks **70–73**. In this way e.g. transfer of graphics information is done using the resolution facilitated by portable terminal device **10**, if it is lower than the resolution of output device **30–33** selected by fixed device **20**. Correspondingly, as a default value for data transfer the highest resolution facilitated by output device **30–33** is used if it is lower than the highest resolution facilitated by portable terminal device **10**. In this way more transfer capacity cannot be reserved for data transfer than what is needed. If the amount of data to be transferred is large compared with available transfer capacity, which is in general the case when operating by radio, it is possible to decrease the resolution in order to increase the data transfer speeds and to lower the cost for data transfer. In this case naturally the quality of the transferred text, graphics and other information is reduced.

As independent format itself it is in principle possible to use any page description language. It is most essential from the point of view of the present invention that format generator **42** of portable terminal device **10** forms such independent format **44**, which is decoded by data transfer program **55** of fixed device **20** into a form suitable for printer drivers **56**, **57** (FIG. **3**A) or output unit **64** (FIG. **3**B). An independent format **44** suitable for the information output system according to the invention is, in addition to WMF- and BMP-formats, the HTML (Hyper Text Markup Language) page description language. HTML page description language has been defined according to ISO (International Standardizing Organization) - standard (ISO 8879,1986).

An essential part of page description languages is also the possibility to include separate pictures in the pages, which pictures have been coded using some other coding method, such as e.g. JPEG-compression method. JPEG (Joint Photographic Experts Group) is an ISO standard for encoding fixed pictures and it facilitates transferring large pictures of good quality utilizing data transfer band effectively. The compression ratio can be adjusted using picture quality as a criterion, and thus JPEG-standard is very versatile. JPEG is well suited for information output systems according to the invention utilizing independent data transfer format **44**, in which systems the capacity available for data transfer can be very different in different operating situations: by radio in GSM-system it may be that only 9600 bit/s is available, whereas in infrared connections it is possible to use e.g. speeds 1.152 or 4 Mbit/s.

In the information output system according to the invention there can be several page description languages, in which case processor **60** of portable terminal device **10** selects for format generator **42** the page description language which is best suited for the current page intended to be output. This is preferable among other things because some of the page description languages are better suited for text dominated (e.g. HTML) and some for graphics (e.g. JPEG) -pages. As an example of other page description languages suitable for independent format **44** we mention HPGL (Hewlett-Packard Graphics Language) -language developed by Hewlett-Packard, which language has become a de-facto standard. HPGL-language has been designed for plotters, and thus it is well suited e.g. for pages which contain plenty of pictures consisting of thin lines. Another page description language, mainly suited for high transfer speeds because of the high volume of the information to be transferred, is EPS (Encapsulated Post Script).

In order to carry out data transfer from portable terminal device **10** to fixed device **20** format generator **42** places, if required, independent format **44** into transfer frame **90** (FIG. **5**). If a specific port (e.g. a COM-port) in fixed device **20** has been allocated for data to be received in independent format **44**, a transfer frame according to the following description is not necessarily needed, because that data is always automatically directed to the correct address. It is possible to include direct output code **91** in transfer frame **90**. Based upon the information marked in direct output code **91**, data transfer program **55** in fixed device **20** recognizes that data in independent format **44** is concerned and can act accordingly, that is to carry out the described conversions and transfer the received information directly to the output device. If so desired, an authorization for outputting can be requested from the operator e.g. using an inquiry window displayed on the computer screen. The data in independent format **42** has been placed in blocks **94–94**"". Language identifiers **93–93**"" are placed before the data blocks, which identifiers indicate which page description language is used in data blocks **94–94**"". It is also possible to include in transfer frame **90** fault correction parameters **92**, prior known to a person skilled in the art, for detection and/or correction of transfer errors. Error correction parameters can also be placed at the end of transfer frame **90** or after each data area **94–94**"". Direct output code **91** preferably contains information of the name and size of the file to be transmitted, and the revision numbers of the various parts of the system. If the file to be transmitted consists of several transfer frames **90**, also the numbers of the different frames are indicated in direct output code **91**. The numbering can be utilized e.g. at transfer failure, in which case it is possible to request retransmission of only those transfer frames which contained errors.

In the next stage processor **60** selects a page description language suitable for the first page, e.g. HTML language. The identifier of the page description language is placed by processor **60** in language identifier **93** of data transfer frame **90**. Now that format generator **42** has prepared the frames of data transfer frame **90**, format generator **42** of portable terminal device **10** processes data **48** in application **40** into independent format **44**, and places it in data transfer frame **90** in the place allocated to it, in data area **94**. Processor **60** changes the page description language if necessary either between pages or even in the middle of a page, and while doing so, always generates a new language identifier **93'**, after which it again places the data generated with the page description language in data area **94'**. When data transfer frame **90** is entirely ready, format generator **42** sets it ready in the data transfer path in block **75**. The size and quantity of data transfer frames **90** is optimized according to the amount of data to be transferred and to the quality of the transfer connection in order to have the data transfer connection operating as efficiently and reliably as possible. Data transfer frame **90** can be realized in a number of ways, and using it is not even necessary in the data transfer system according to the invention.

In the next stage independent format **44** is transferred, controlled by processors **60** and **62** and data transfer controllers **43** and **54**, in data transfer frames **90** from portable terminal device **10** to fixed device **20** using the selected

transfer path **45, 46, 47**. When data in independent format **44** is transferred directly to output device **30** (FIG. 3B), processor **66** of output device **30** is utilized correspondingly for the data transfer.

In fixed device **20** transfer program **55** decodes independent format **44** according to the requirements of operating system into a form suitable for printer drivers **56** or **57** and forwards the decoded data to the selected printer driver **56** or **57** for outputting. In the present data output system according to the invention utilizing independent data transfer format **44**, data transfer program **55** has been realized in processor **62** of fixed device **20**, which processor executes decoding and data transfer program code **55** stored in a memory (not shown in the figure). Processor **62** checks at regular intervals I/O (Input/Output) ports **51, 52,53** of fixed device **20** in order to find out if any data transferred in independent format **44** is coming. This so called polling takes place under the control of operating system. Respectively the data transfer program monitors regularly the result of said polling. Data transfer program **55** recognizes a file received in independent format **44** e.g. from the name of the received file or from direct outputting identifier **91** transferred in the data transfer frame. Depending on the operating system, the information of a file received in independent format **44** can be forwarded also using the message principle. In this case the operating system automatically generates a message, which tells that data in independent format **44** has been received, and transmits it to data transfer program **55**. In this case data transfer program **55** preferably needs not actively to monitor the result of the polling executed by the operating system. If the Input/Output port used has been reserved exclusively for data transferred in independent format **44**, there is no need to use a direct outputting identifier.

If e.g. Windows user environment is concerned, data transfer program **55** decodes independent format **44** into WMF (Windows Meta Format) -format. A special case occurs when WMF-format is used for independent format **44**, in which case naturally no decoding is preferably needed. WMF-format is a general format prior known to a person skilled in the art, which is used in Windows environment e.g. for data transfer between different computer programs. It is just because of this that all printer drivers **56, 57** are capable of outputting text and picture files in WMF-format. WMF-format preferably facilitates transfer of high quality graphics, because WMF-format is not limited to a certain resolution in the same way as e.g. BMP (Bit Map) -format, which also could be used in the information output system according to the invention.

Before transferring the data decoded into WMF-format to printer driver **56, 57** processor **62** of fixed device **20** selects for use a desired printer driver **56, 57** based upon the printer driver data of the operating system, preferably the printer driver of the default printer. When printer driver **56, 57** has been selected, processor **62** transfers, controlled by data transfer program **55**, the data decoded into WMF-format by itself to the selected printer driver, e.g. printer driver **56**, which further transfers the data to output queue (not shown in the figure). The print queue is prior known to a person skilled in the art, e.g. the queue of FIFO (First In First Out) -type arranged by processor **62** in the memory (not shown in the figure) of fixed device **20**, from where processor **62** of fixed device **20** further transfers the data to be output to output device **30**. It is also possible to arrange the print queue in connection with computer **21** connected to local network **22**, and also in output device **31** itself.

In the cases mentioned in the previous chapter the documents intended to be output are often output in the order they arrive in the print queue. It is also possible to change the printing order of the documents by giving the documents different priorities, in which case the documents classified as most urgent are printed first. When data transfer program **55** executed by processor **62** of fixed device **20** has transferred the documents to be printed to the print queue, printer **30–33** selected by fixed device **20** prints out the desired documents exactly alike in the case if the documents had been originally created and output in fixed device **20**. It preferably makes no difference to printer driver **56, 57** and output devices **30–33** were the documents desired to be printed came from fixed device **20** or portable terminal device **10**.

The information output system according to the invention solves the long known problem of outputting with almost any output device without the need for a output device dependent driver. With this method remarkably easy use is achieved and successful output can be trusted upon even in foreign environment. The invention is very useful particularly for portable terminal devices, which are often moved from one environment to another. They also have a limited memory capacity, which in the information output system according to the invention is released for effective use because printer drivers requiring plenty of memory space are preferably not needed any longer.

This has been a presentation of the realization of the invention and some of its embodiments using examples. It is evident to a person skilled in the art that the invention is not limited to the details of the above presented embodiments and that the invention can be realized also in other form without deviating from the characteristics of the present invention. The above presented examples should be regarded as illustrating, not as limiting. Thus the possibilities of realizing and using the invention are limited only by enclosed patent claims. Thus the various embodiments of the invention defined by the claims, including equivalent embodiments, are included in the scope of the invention.

What is claimed is:

1. An information transmission system for transmitting information from a mobile terminal device for printing at the receiving end, which system comprises:

a first terminal device, a second terminal device and means for establishing a data transfer connection between said first and second terminal devices, and in said first terminal device means for running an application, means for conversion of data processed by the application into a form suitable for said data transfer connection, and in said second terminal device means for receiving the data and in connection with said second terminal device means for automatically printing said data upon receiving the data in the second terminal device;

wherein said data conversion means of the first terminal device comprise a format generator which has been arranged to process the data to be transmitted into an application independent data transfer format, said data transmitting means have been arranged to establish the data transfer connection to the second terminal device, to generate a direct printing identifier, and to transfer the data in said independent data transfer format to said second terminal, and said second terminal device further comprises:

format decoding means which have been arranged to decode the data in said independent data transfer format into a form suitable for said data printing means for printing.

2. An information transmission system for printing, according to claim **1**, wherein said format decoding means

have been arranged to automatically decode the data in said independent data transfer format into a form suitable for said data outputting means for automatic output.

3. An information transmission system for printing, according to claim **1**, wherein said data transmitting means and said data receiving means comprise at least one of the following means of data transfer: a radio transceiver, an infrared transceiver, a PCMCIA-bus, a serial port and a parallel port.

4. An information transmission system for printing, according to claim **1**, wherein the information is printed via a mobile communication network, whereby the first terminal device comprises means for transmitting the data via said mobile communication network, the data transfer connection is via said mobile communication network, and the data is transferred in said independent data transfer format from said first terminal device to said second terminal device via said mobile communication network.

5. An information transmission system for printing, according to claim **1**, wherein the first terminal device is a pocket-size mobile communication device.

6. A mobile terminal device which comprises means for running an application, means for converting data processed by the application into a form suitable for a data transfer connection, data transmitting means for transmitting the data, wherein said data converting means of the mobile terminal device comprise a format generator which has been arranged to process the data to be transmitted into a data transfer format independent of the application, and a direct printing identifier generator to generate a direct printing identifier, and said data transmitting means have been arranged to establish the data transfer connection to a second terminal device and to transfer the data in said independent data transfer format including said direct printing identifier to a second terminal device for automatic printing upon receiving the data in the second terminal device.

7. A terminal device according to claim **6**, wherein said data transmitting means comprise at least one of the following means of data transfer: a radio transceiver, an infrared transceiver, a PCMCIA-bus, a serial port and a parallel port.

8. A terminal device according to claim **6**, wherein said data transmitting means have been arranged to establish the data transfer connection to said second terminal device via a mobile communication network and to transfer the data in said independent data transfer format to said second terminal device via said mobile communication network.

9. A terminal device according to claim **6**, wherein the terminal device is a pocket-size mobile communication device.

10. A method for printing information on a printing device from a first mobile terminal device, in which an application is used, wherein data processed by the application is processed into a form suitable for a data transfer connection, is established between the first mobile terminal device and a second terminal device, and further wherein, before transferring, the data is processed by the first terminal device into an application and printing device independent data transfer format having a direct printing identifier and at the second terminal device the received data is processed into a form suitable for printing devices and is transferred to the printing devices for automatic printing upon receiving the data in the second terminal device.

11. A method according to claim **10**, wherein the information is printed via a mobile communication network, whereby the data transfer connection is via said mobile communication network, and the data is transferred in said independent data transfer format from said first terminal device to said second terminal device via said mobile communication network.

12. A method according to claim **10**, wherein the first terminal device is a pocket-size mobile communication device.

13. An information transmission system for transmitting information from a mobile terminal for printing at the receiving end, which system comprises:

a mobile terminal device, a receiving terminal device and means for establishing a wireless data transfer connection between said mobile and receiving terminal devices, and in said mobile terminal device means for using printer driver application software resident on said receiving terminal device, means for conversion of data processed by the application into a form suitable for said wireless data transfer connection, and in said receiving terminal device means for receiving the data and in connection with said second terminal device means for automatically printing said data upon receiving the data in the receiving terminal device;

wherein said data conversion means of the mobile terminal device comprise a format generator which has been arranged to process the data to be transmitted into an application independent data transfer format, said data transmitting means have been arranged to establish the data transfer connection to the receiving terminal device and to transfer the data in said independent data transfer format to said receiving terminal, and said receiving terminal device further comprises

format decoding means which have been arranged to decode the data in said independent data transfer format into a form suitable for said data printing means for printing.

14. An information transmission system for transmitting information from a mobile terminal for printing at the receiving end, which system comprises:

a first terminal device, a second terminal device and means for establishing a data transfer connection between said first and second terminal devices, and in said first terminal device means for running an application, means for conversion of data processed by the application into a form suitable for said data transfer connection, and in said second terminal device means for receiving the data and in connection with said second terminal device means for automatically printing said data upon receiving the data in the second terminal device;

wherein said data conversion means of the first terminal device comprise a format generator which has been arranged to process the data to be transmitted into an application independent data transfer format, said data transmitting means have been arranged to establish the data transfer connection to the second terminal device and to transfer the data in said independent data transfer format to said second terminal, and said second terminal device further comprises

format decoding means which have been arranged to decode the data in said independent data transfer format into a form suitable for said data printing means for printing, and

wherein said data transmitting means of the first terminal device have been arranged to transmit the data in the independent data transfer format in data transfer frames, and that said data conversion means have been arranged to add a direct print identifier in the data transfer frames.

15. A terminal device which comprises means for running an application, means for converting data processed by the application into a form suitable for a data transfer connection, data transmitting means for transmitting the data, wherein said data converting means of the terminal device comprise a format generator which has been arranged to process the data to be transmitted into a data transfer format independent of the application, and said data transmitting means have been arranged to establish the data transfer connection to a second terminal device and to transfer the data in said independent data transfer format to a second terminal device for automatic printing upon receiving the data in the second terminal device, and

wherein said data transmitting means of the terminal device have been arranged to transmit the data in the independent data transfer format in data transfer frames, and that said data converting means have been arranged to add a direct printing identifier in the data transfer frames.

* * * * *

US007028102B1

(12) **United States Patent**
Larsson et al.

(10) **Patent No.: US 7,028,102 B1**
(45) **Date of Patent: Apr. 11, 2006**

(54) **METHOD AND SYSTEM FOR PRESENTING INFORMATION**

(75) Inventors: **Olof Larsson**, Lund (SE); **Lennart Fagerberg**, Genarp (SE)

(73) Assignee: **Axis, AB**, Lund (SE)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 549 days.

(21) Appl. No.: **09/639,011**

(22) Filed: **Aug. 15, 2000**

Related U.S. Application Data

(60) Provisional application No. 60/183,281, filed on Feb. 17, 2000.

(30) **Foreign Application Priority Data**

Dec. 13, 1999 (SE) 9904552

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/246**; 709/232

(58) **Field of Classification Search** 709/206, 709/227, 232, 236, 246; 379/93.24
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2,949,776	A *	8/1960	Field et al.	73/290 R
5,848,415	A	12/1998	Guck	
5,862,321	A	1/1999	Lamming et al.	
6,076,076	A *	6/2000	Gottfreid	705/45
6,141,688	A *	10/2000	Bi et al.	
6,249,809	B1 *	6/2001	Bro	
6,285,889	B1 *	9/2001	Nykanen et al.	455/557
6,400,810	B1 *	6/2002	Skladman et al.	
6,430,601	B1 *	8/2002	Eldridge et al.	
6,438,585	B1 *	8/2002	Mousseau et al.	
6,442,595	B1 *	8/2002	Kelly	709/206
6,557,029	B1 *	4/2003	Szymansky	
6,725,300	B1 *	4/2004	Nagasaka et al.	710/62
6,738,841	B1 *	5/2004	Wolff	710/62

FOREIGN PATENT DOCUMENTS

EP	0924628	6/1999
WO	WO 98/4424	10/1998

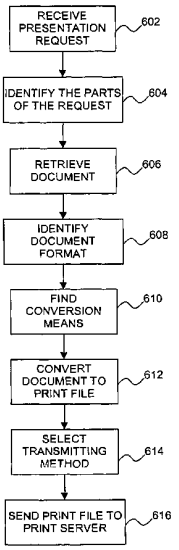
* cited by examiner

Primary Examiner—Paul H. Kang
(74) *Attorney, Agent, or Firm*—IP Creators; Charles C Cary

(57) **ABSTRACT**

A presentation of a selected document, stored at a node of a network, is accomplished by means of a system essentially comprising a handheld device, a communication device for a presentation unit and a device that handles presentation requests. By operating the handheld device, a user is able to get a presentation unit specification from said communication device. This specification is then sent, together with an address to a selected document, to a device handling presentation requests. In this device the selected document is then converted to a format suitable for the presentation unit of said communication device, where the converted document is presented. The system could be using WAP-technology for sending information from the handheld device to the device that handles presentation requests and for accessing information on the Internet. Further the system could, for example, be using Bluetooth or IrDA for the communication between the handheld device and said communication device. The system also makes it possible to achieve mobile printing and portable access to documents that cannot be presented on a small display.

15 Claims, 11 Drawing Sheets



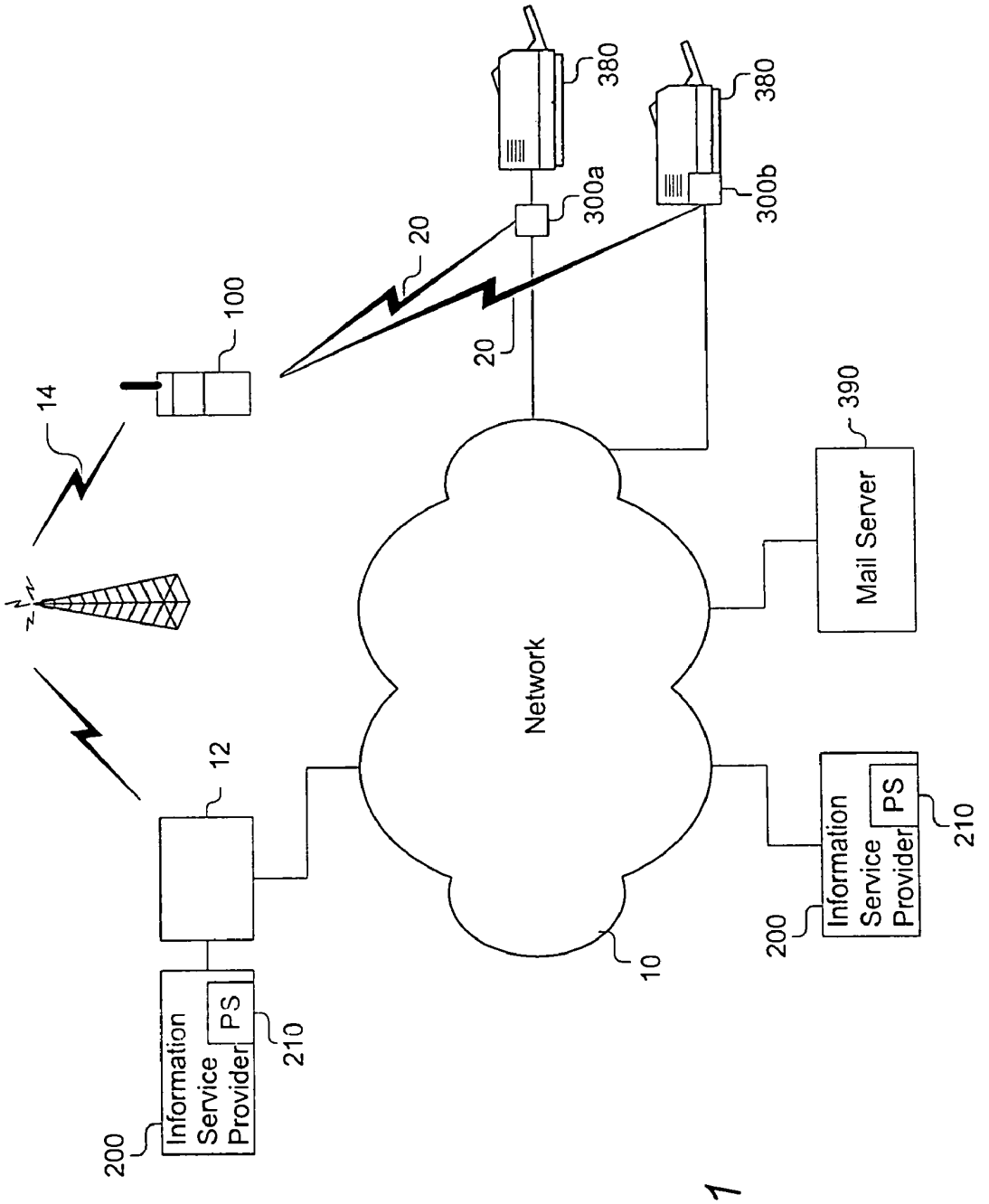

380
380
300a
300b
20
20
100
14
390
Mail Server
Network
12
Information Service Provider
PS
210
200
Information Service Provider
PS
210
200
10

FIG 1

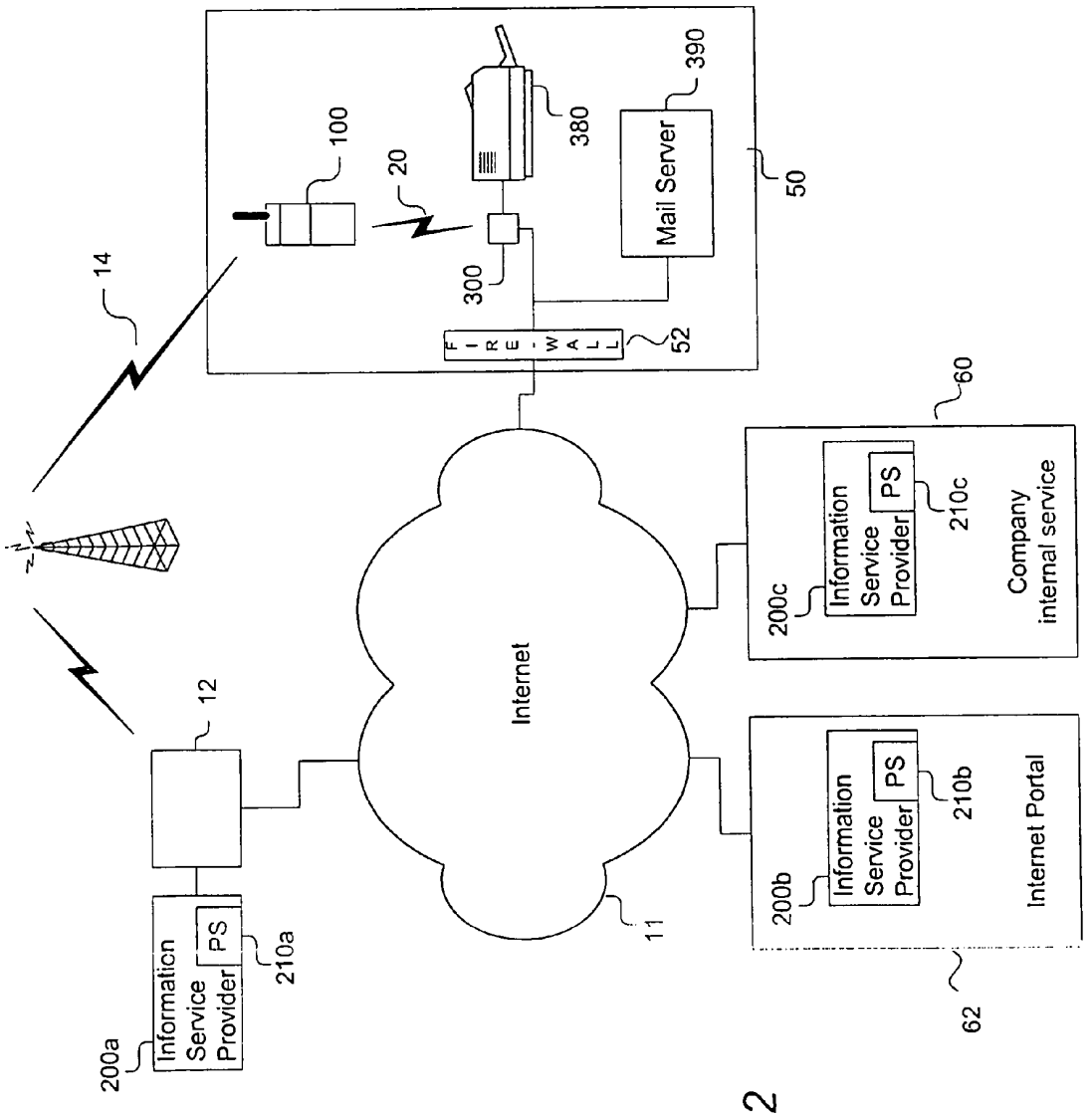

14
100
20
380
390
Mail Server
50
300
FIRE-WALL
52
60
Information Service Provider
PS
210c
Company internal service
200c
Internet
12
11
Information Service Provider
PS
210b
Internet Portal
200b
62
Information Service Provider
PS
210a
200a

FIG 2

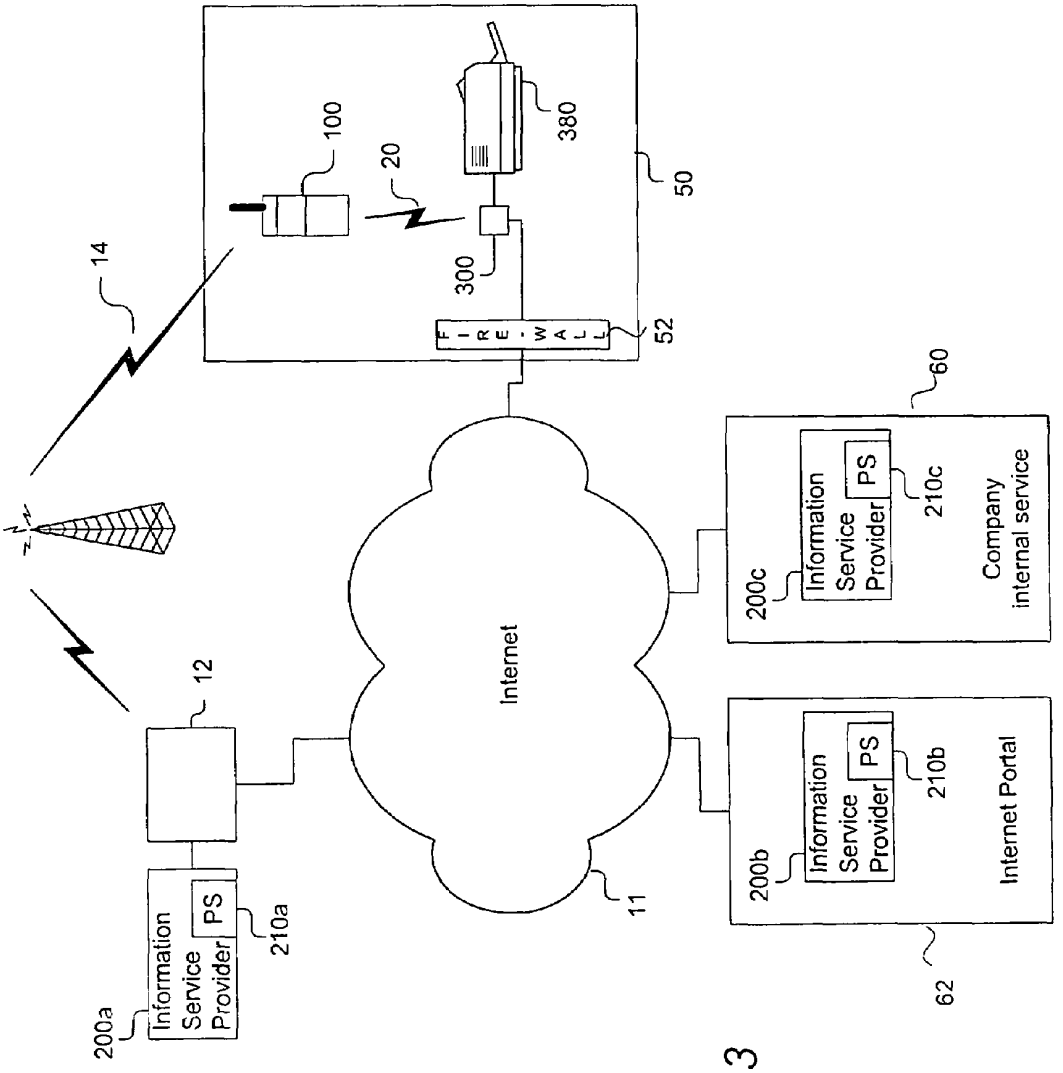

14
100
20
380
50
300
F I R E - W A L L
52
60
Information Service Provider
PS
210c
Company internal service
200c
Internet
12
11
200b
Information Service Provider
PS
210b
Internet Portal
62
200a
Information Service Provider
PS
210a

FIG 3

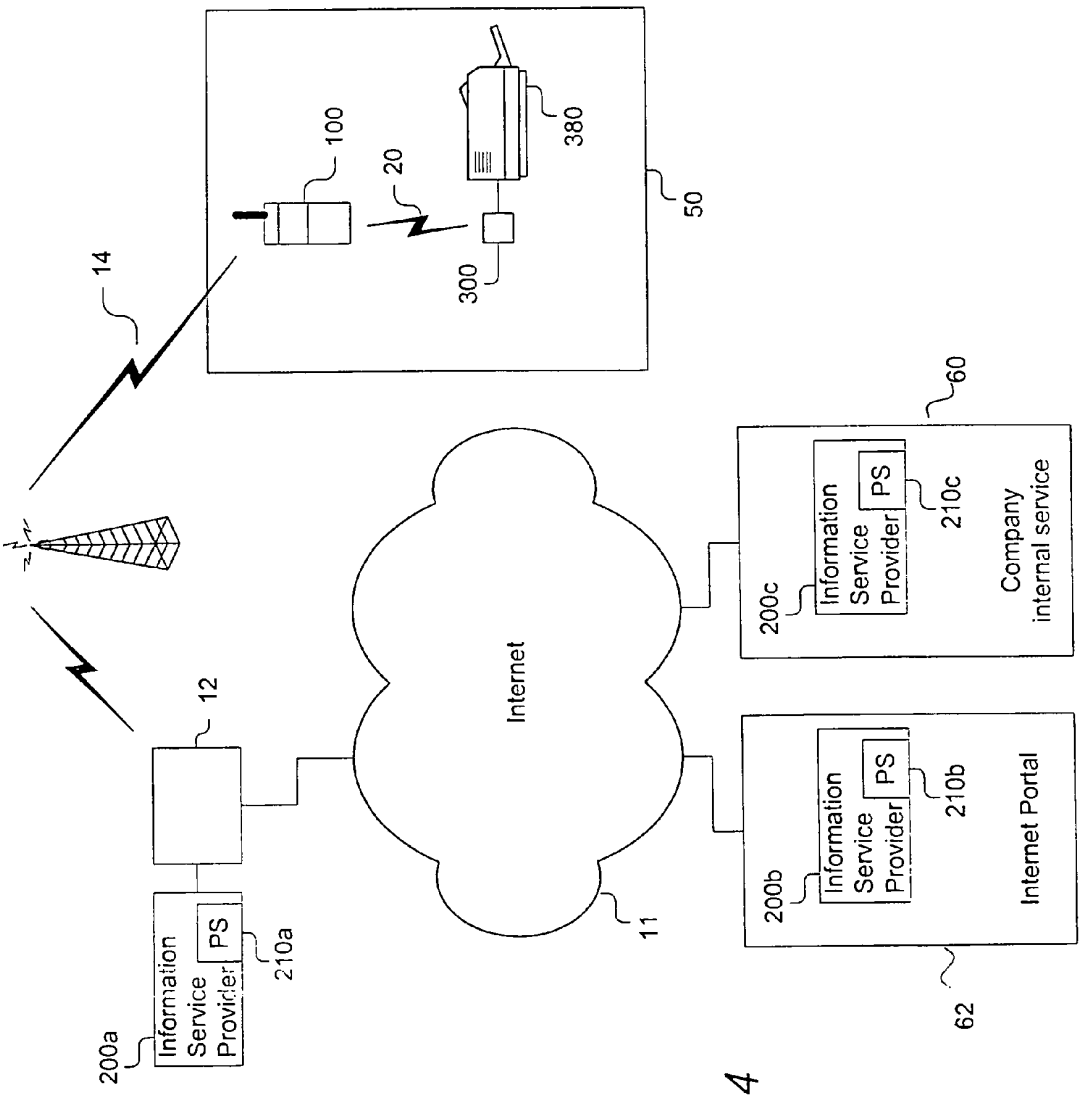

14
100
20
380
50
300
12
Internet
11
60
200c
Information Service Provider
PS
210c
Company internal service
200b
Information Service Provider
PS
210b
Internet Portal
62
200a
Information Service Provider
PS
210a

FIG 4

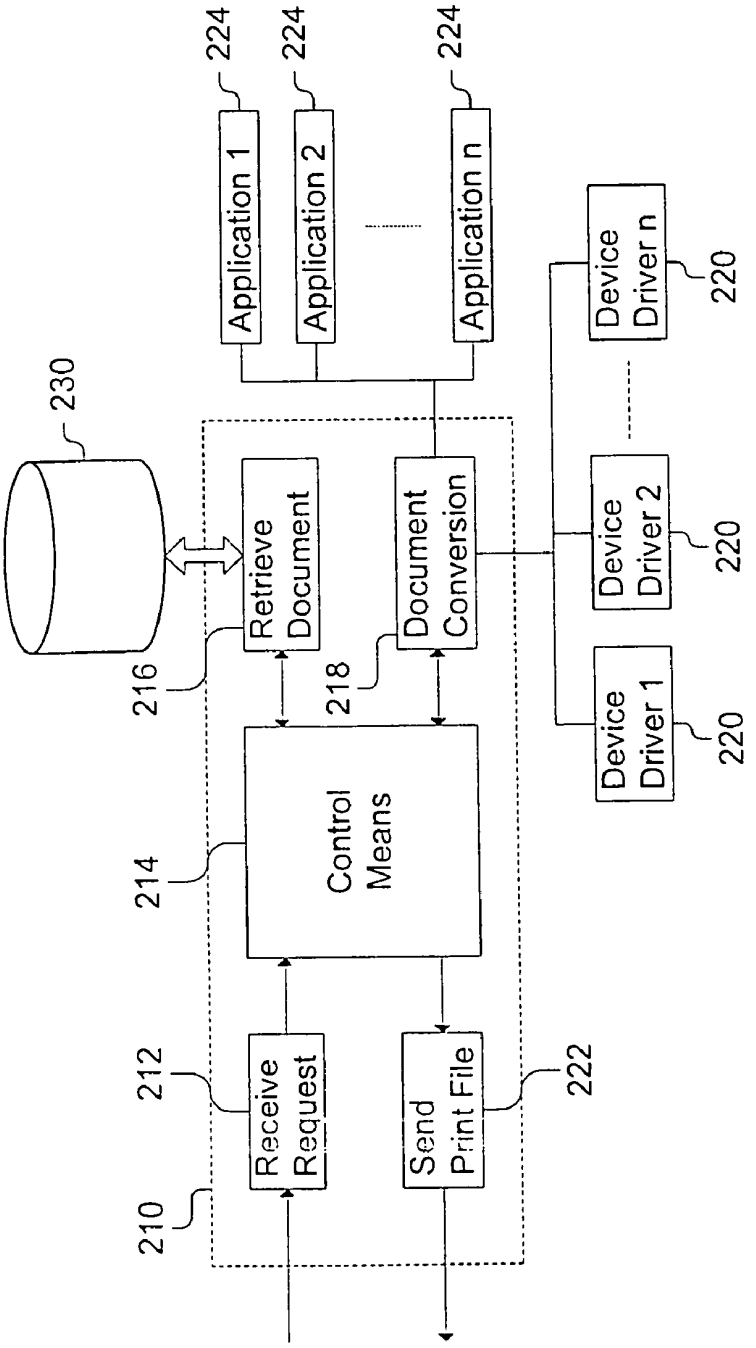

224
224
224
Application 1
Application 2
Application n
230
Retrieve Document
Document Conversion
Device Driver n
220
Device Driver 2
220
Device Driver 1
220
216
218
Control Means
214
212
Receive Request
Send Print File
222
210

FIG 5

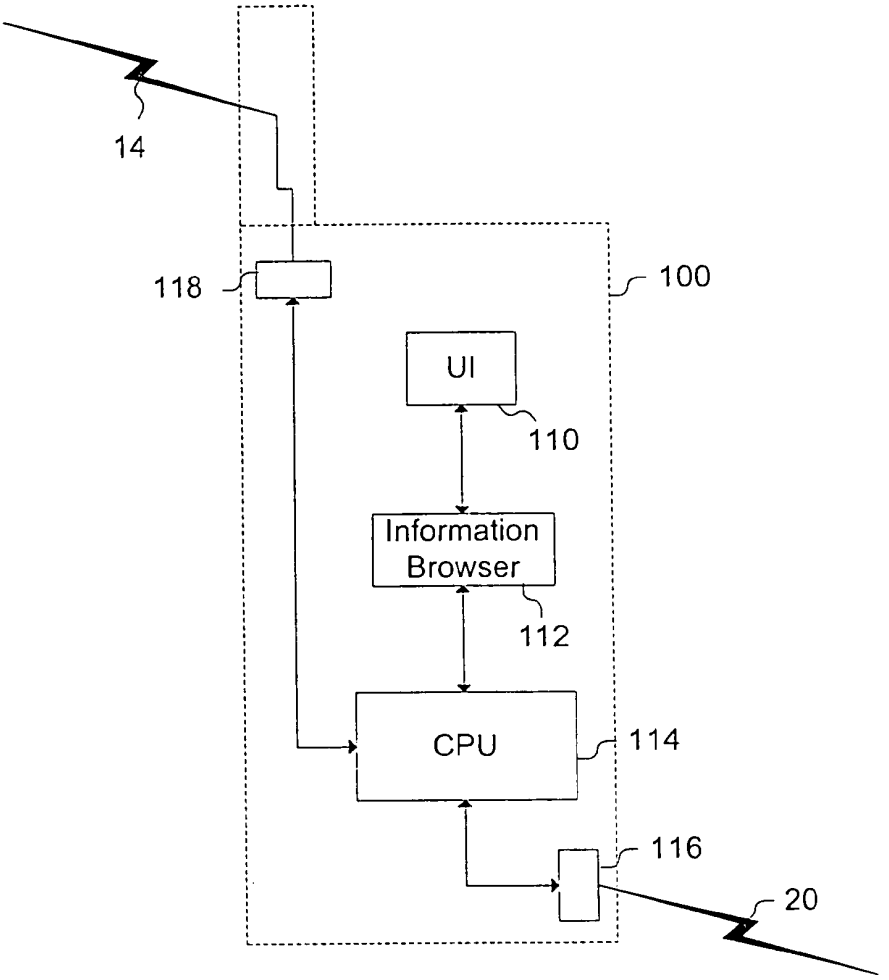


FIG 6

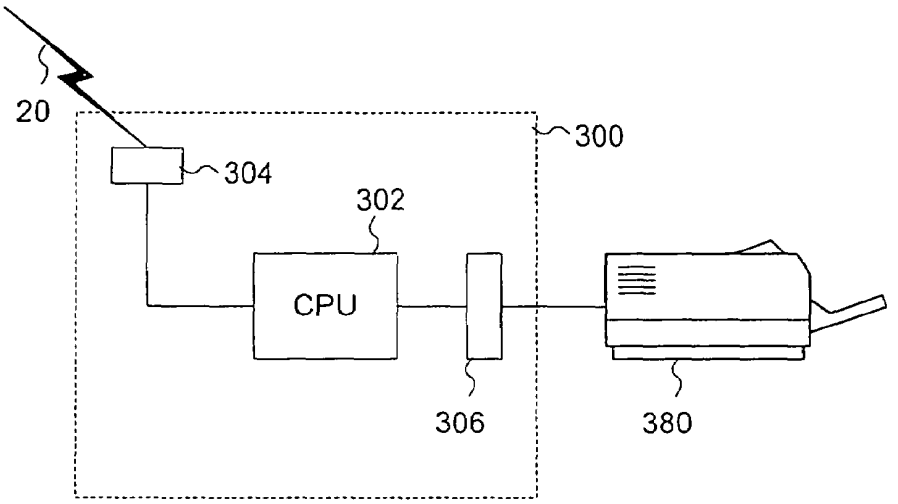

20
300
304
302
CPU
306
380

FIG 7a

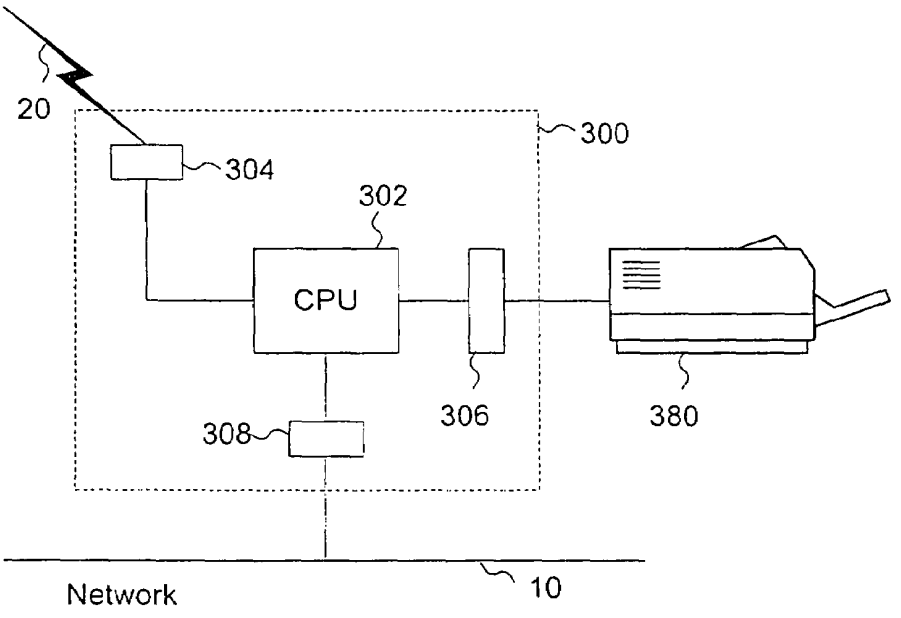

20
300
304
302
CPU
306
380
308
Network
10

FIG 7b

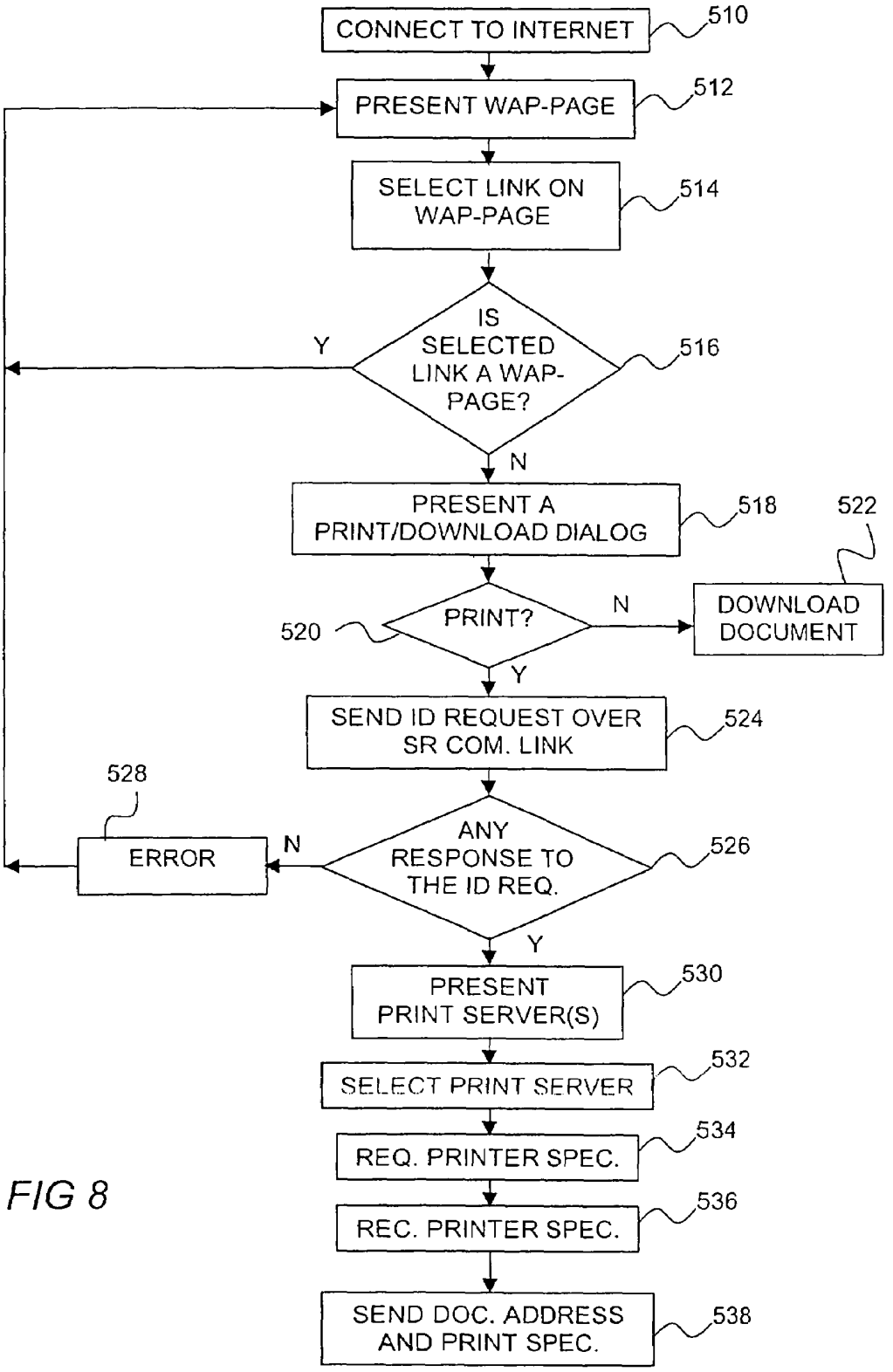

CONNECT TO INTERNET
510
PRESENT WAP-PAGE
512
SELECT LINK ON WAP-PAGE
514
IS SELECTED LINK A WAP-PAGE?
516
Y
N
PRESENT A PRINT/DOWNLOAD DIALOG
518
522
PRINT?
520
N
DOWNLOAD DOCUMENT
Y
SEND ID REQUEST OVER SR COM. LINK
524
528
ERROR
N
ANY RESPONSE TO THE ID REQ.
526
Y
PRESENT PRINT SERVER(S)
530
SELECT PRINT SERVER
532
REQ. PRINTER SPEC.
534
REC. PRINTER SPEC.
536
SEND DOC. ADDRESS AND PRINT SPEC.
538

FIG 8

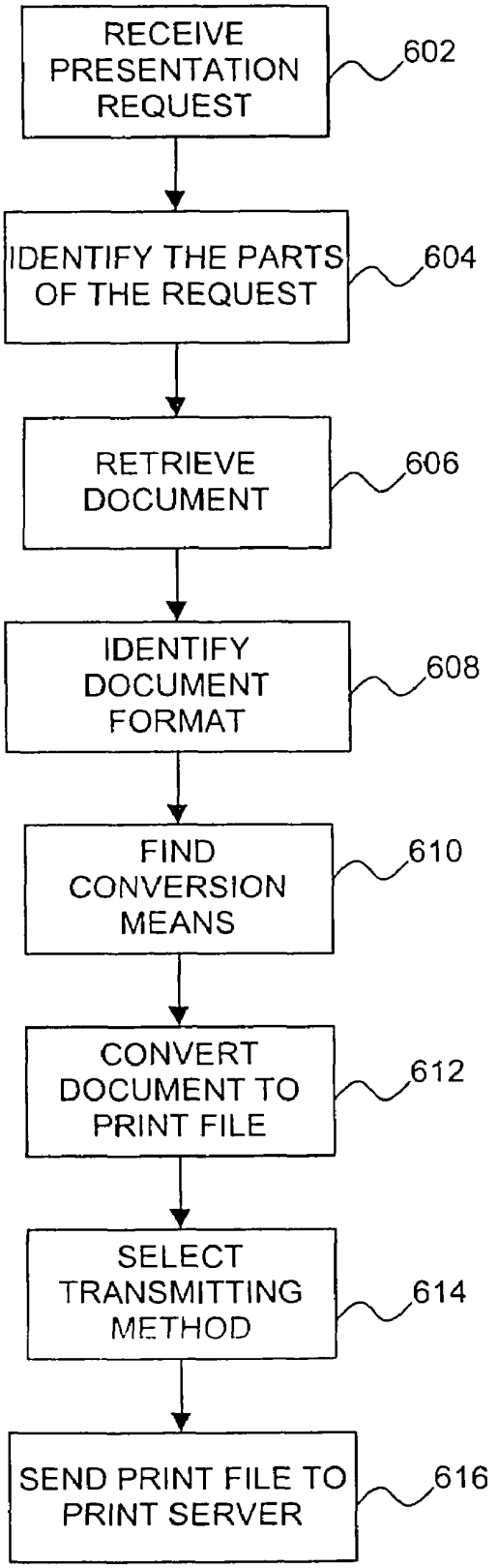

RECEIVE PRESENTATION REQUEST
602
IDENTIFY THE PARTS OF THE REQUEST
604
RETRIEVE DOCUMENT
606
IDENTIFY DOCUMENT FORMAT
608
FIND CONVERSION MEANS
610
CONVERT DOCUMENT TO PRINT FILE
612
SELECT TRANSMITTING METHOD
614
SEND PRINT FILE TO PRINT SERVER
616

FIG 9

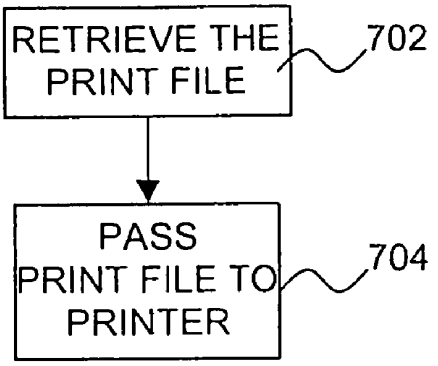


FIG 10a

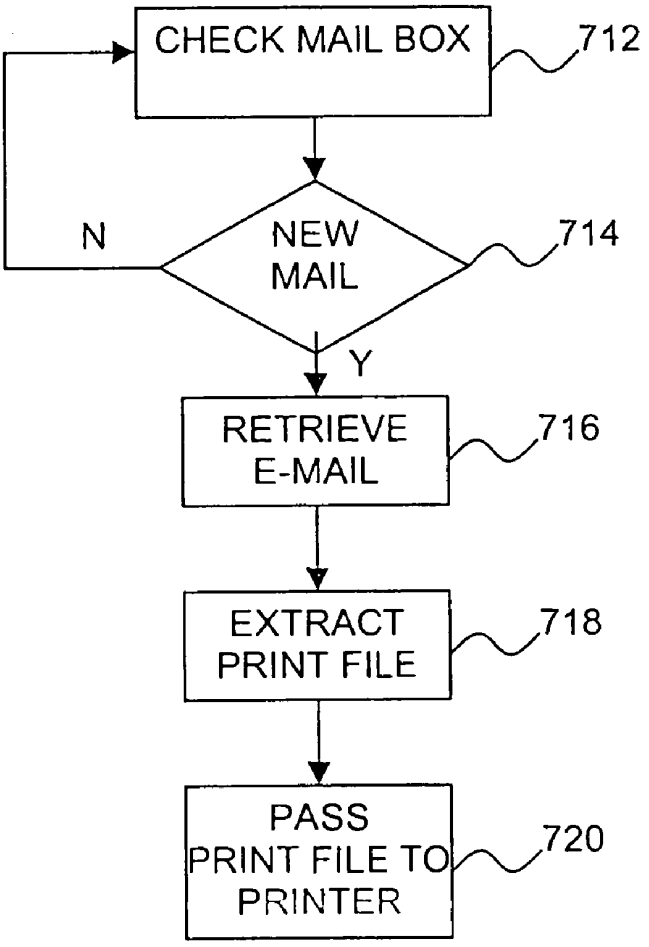


FIG 10b

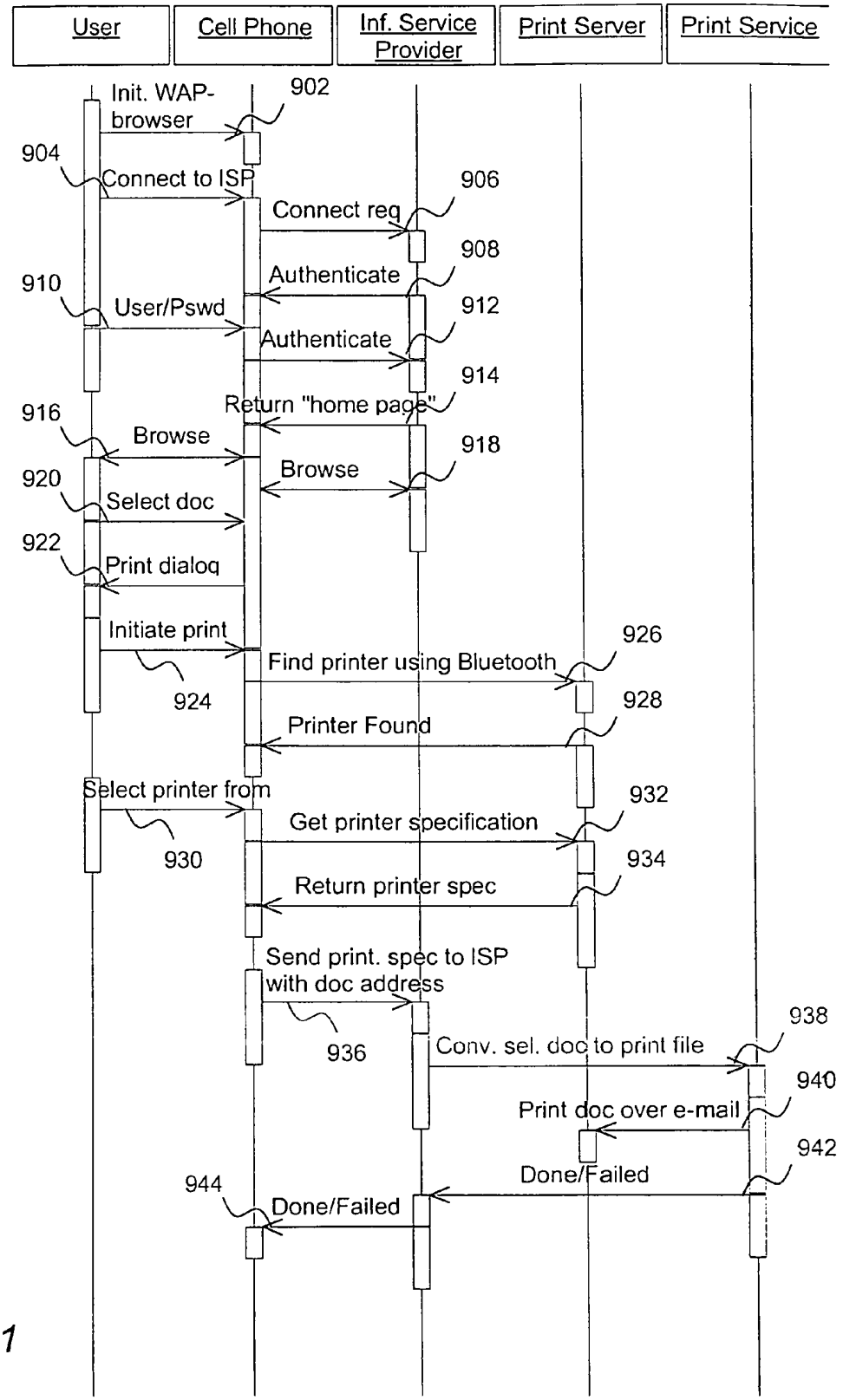

User
Cell Phone
Inf. Service Provider
Print Server
Print Service
Init. WAP-browser
902
904
Connect to ISP
906
Connect req
908
Authenticate
910
User/Pswd
912
Authenticate
914
Return "home page"
916
Browse
918
Browse
920
Select doc
922
Print dialog
Initiate print
926
Find printer using Bluetooth
924
928
Printer Found
Select printer from
932
Get printer specification
930
934
Return printer spec
Send print. spec to ISP with doc address
938
936
Conv. sel. doc to print file
940
Print doc over e-mail
942
Done/Failed
944
Done/Failed

FIG 11

METHOD AND SYSTEM FOR PRESENTING INFORMATION

This application claims the benefit of U.S. Provisional Application No. 60/183,281 filed on Feb. 17, 2000, the disclosure of which is incorporated herein by reference.

TECHNICAL FIELD OF THE INVENTION

The present invention relates to a system and methods for presenting electronically stored information on a presentation unit via a network.

BACKGROUND OF THE INVENTION

Today a lot of information is accessible over networks such as Intranets and the Internet. Even information that is to be accessible only for a specific group of users can be accessed over these networks by means of secure communication channels and different identification methods.

To have information presented on a presentation unit nearby, the user must have a computer having access to the network in which the information is accessible and a presentation unit connected to the computer. Thus, people that travel a lot and need access to information stored on, for example, a company server have to carry with them a computer to be able to access the desired information. Further, they either have to have a portable modem with them or have access to a network connection.

The access to information can be accomplished by means of a cellular telephone implementing e.g. the Wireless Application Protocol (WAP), a Personal Digital Assistant (PDA) or a similar device. Then, the user has access to information presented in a format presentable on the cellular telephone. However, if the information is a text document comprising a large amount of text, e.g. a patent application, or a blueprint of a machine to repair, then it is not suitable and often not even possible to present the information on the cellular telephone. Even if the telephone is capable of using a normal Internet Protocol (IP) connection for communication, the information has to be adapted to the small size screen of the telephone or the PDA, and thus it would be very difficult for the user to get an overview or even read such a document.

SUMMARY OF THE INVENTION

The object of the present invention is to make it possible for a person to have information presented to him in a form that is easy to read and overview, even if the information is in a form not suitable for easy reading on a small display.

Another object is to increase the mobility of people who require access to information of the above-mentioned character.

Yet another object is to make it possible for a person to access documents that are electronically stored at a remote location.

These objects are accomplished by means of a method in a system according to claim **1**, a system according to claim **6**, a method for a device that handles presentation requests according to claim **12**, a device that handles presentation requests according to claim **20**, a computer program product according to claim **26**, a method for a communication device according to claim **27**, a communication device according to claim **33**, a method for a handheld device according to claim **36**, or a handheld device according to claim **39**. Preferred embodiments of the invention are disclosed in the dependent claims.

More particularly, according to one aspect of the invention, a method in a system for selection and presentation of information comprises the steps of:

- receiving a presentation unit specification at a handheld device via a short-range wireless connection from a communication device for a presentation unit,
- sending a document address and said presentation unit specification from the handheld device via a network to a device that handles presentation requests,
- receiving said document address and presentation unit specification at the device that handles presentation requests,
- retrieving, at the device that handles presentation requests, the document corresponding to said document address,
- converting, at the device that handles presentation requests, the retrieved document to a format suitable for the presentation unit,
- sending the converted document from the device that handles presentation requests via a network to the communication device for a presentation unit, and
- receiving said converted document at the communication device for a presentation unit.

According to another aspect of the invention, a system for selection and presentation of information, comprises:

- a handheld device enabled to present information retrieved from an information provider on a screen in the handheld device, including:
  - an I/O-interface for short-range wireless communication with a communication device for a presentation unit,
  - means for receiving a presentation unit specification via said I/O-interface for short-range communication,
  - means for sending a document address to a device that handles presentation requests, and
  - means for sending a received presentation unit specification to a device that handles presentation requests,
- a device that handles presentation requests, including:
  - means for receiving a document address, referring to a document that is to be presented, from a handheld device,
  - means for receiving a presentation unit specification from a handheld device,
  - means for retrieving the document corresponding to the document address,
  - means for converting the retrieved document to a format that is suitable for the presentation unit according to the presentation unit specification, and
  - means for sending the converted document to the presentation address,
- a communication device for a presentation unit, including:
  - an I/O-interface for short-range wireless communication with a handheld device,
  - means for sending a presentation unit specification via said I/O-interface for short-range wireless communication, and
- means for receiving information to be presented by a presentation unit.

The system according to the invention solves the problems accounted for above. When the need for a certain document arises, a user of this system only has to pick up his handheld device, make it display a page where the document

is identified, select the desired document on said page for presentation on the presentation unit, make it access a communication device for a presentation unit nearby to exchange data regarding the presentation unit and then have the document presented on said presentation unit. The document can be located at any node of a network as long as there is an address identifying it. The address to the document and a specification of the presentation unit are sent from the handheld device to a device that handles presentation requests and then the document will be sent to the presentation unit in a format that the presentation unit understands.

The system is advantageous by being able to allow essentially any kind of document to be presented on essentially any kind of presentation unit. Further, the system provides this advantage without requiring any additional memory in the handheld device or the communication device for the presentation unit for storing conversion means for various document formats and presentation units. This is achieved by having the requested document converted in a device that handles presentation requests.

Another advantage of performing the conversion in the device that handles presentation requests is that the normal operation neither of the handheld device nor of the communication device is affected negatively by the conversion.

Yet another advantage of performing the conversion in a device that handles presentation requests is that the document does not necessarily need to be sent to the handheld device, at least not before it is converted. This is achieved by making the device that handles presentation requests retrieving a requested document on the basis of a document address received from the handheld device, and therefore the handheld device only needs to communicate the document address in order to make the document available to the device that handles presentation requests. Thus, by having the device that handles presentation requests retrieving the document, bandwidth is saved on the communication paths of the handheld device.

A further advantage of performing the conversion at the device that handles presentation requests, and thus positioning the means for conversion at this location, is that the process of adding new means for conversion or changing of old means for conversion is simplified to a great extent. One reason therefor is that there will be a greater number of handheld devices or presentation units than devices that handle presentation requests, whereby it will be easier for an administrator to perform an upgrade of the conversion means at the device that handles presentation requests. Another reason is that the handheld device is probably not connected to a network all the time. Thus, a user should be required to personally perform an update of conversion means if the means were located in the handheld device.

The system is also advantageous in that the handheld device is able to exchange data with presentation units nearby and can also be made to identify them, thanks to the short-range communication link. Thus, a user does not need to worry about manually identifying a presentation unit and typing in data needed to utilise the presentation unit. As a result, the user is not required to search for parameters needed to accomplish a presentation. These advantages are accomplished by means of the short-range communication link that is established between the handheld device and a communication device for the presentation unit. The short-range communication link can for example be implemented using Bluetooth, IrDA, Open Air, Home RF, Hiper Lan, IEEE 802.11, etc.

In the context of the invention, a presentation unit represents devices that are able to present electronically represented information that cannot be presented satisfactorily on a small display of a handheld device. The presentation unit could be, for example, a screen, a rather large display, an e-book, an electronic picture frame or a printer.

According to a preferred embodiment of the invention, the presentation unit is a device that creates a copy of an electronic document on or in a medium that is highly portable. In that case, a presentation unit could, for example, be a conventional printer or an electronically controllable presentation medium, e.g. e-books, electronic picture frames, etc.

In a preferred embodiment the presentation unit is a printer. By presenting the information via a printer on a printable medium, it is possible to have information presented in high resolution and it is easy get an overview of the information. Another advantage of using a printer as presentation unit is that a printed document format is well known and easy to utilise for essentially any person and that a printed document is easy to add comments to or sign, which still is required in regard of most contracts.

An information service provider is a system that allows access to browsable information on a network, e.g. electronically stored documents, e-mails, drawings, blueprints etc. The information service provider could, for example, be an Internet portal or a server for serving the documents of a company to employees or other people allowed access to them. The server for serving the documents of a company could, for example, be a system such as a unified messaging server.

A handheld device is a device that is enabled to communicate over a network and to display information found at nodes of a network, at least in the form of text but preferably also in the form of pictures. Further, it is preferably of such size that a user could hold it in one hand while operating it with the other. The device could for example be a cellular telephone, a PDA or a small computer. In a preferred embodiment, the device is provided with a browser for presenting and browsing the information available on the network.

In the context of the invention a network is, for example, a communication network for telephones, a Wide Area Network (WAN), a Local Area Network (LAN), the Internet or an Intranet.

In one preferred embodiment of the invention the search or selection of information is performed from the handheld device via a wireless telephone network and a station of the network operator giving the handheld device access to an information service provider, which could be directly accessed at the station or could be accessed via a network. In this embodiment the handheld device is a cellular telephone and the user can search for information as long as the telephone is able to establish contact with the network of the telephone operator. The embodiment has the advantage of making the information available at almost any location, allowing the user to search for a document to present wherever he may be. Then, when a document is found, the user can move to a location where the short-range communication link can be established between the telephone and a communication device for a presentation unit for presentation of the document.

In another preferred embodiment of the invention the search or selection of information is performed from the handheld device via the short-range communication link and the communication device for the presentation unit. In this embodiment the communication device is providing an

access point to the Internet, an Intranet or another computer network, from which the information service provider can be accessed. The handheld device of this embodiment could, for example, be a PDA or a small computer. The advantage of this embodiment is that the handheld device does not need any other connection than the short-range communication link for searching or selecting information.

According to a preferred embodiment, the device that handles presentation requests is a module in an information service provider. By providing the device that handles presentation requests as a module in an information service provider, the device is able to share functionality and interfaces with the information service. Further, if the information service provider serves confidential information, the security of the document for presentation is easily maintained.

The system can utilise different paths for sending the presentation file, which is the requested document converted to a suitable format. It is possible to manufacture devices supporting all of the paths or only one or some of the paths. Thus, the paths from the device that handles presentation requests to the presentation unit described below can all be supported by one device.

In one preferred embodiment of the invention, the presentation file is sent from the device that handles presentation requests to the communication device for the presentation unit via at least a wireless network for cellular telephones, the handheld device, and the short-range communication link. One advantage of this embodiment is that the presentation file can be sent to the presentation unit even if the communication device for the presentation unit is lacking a network connection. Another advantage is that the path for the sending of the presentation file has already been established, during the selection of document and presentation unit, and thus the user can be certain that there is a path for sending the presentation file to the presentation unit.

In another preferred embodiment of the invention, the presentation file is sent from the device that handles presentation requests to the communication device for the presentation unit via a network such as the Internet and/or an Intranet. An advantage of this embodiment is that the data rate of the transmission is very high. Another advantage of this embodiment is that the user is able to use the handheld device to perform other tasks when waiting for the presentation.

In yet another preferred embodiment of the invention, the presentation file is sent from the device that handles presentation requests to the communication device for the presentation unit by e-mail. Advantages of this embodiment are that there is no trouble in presenting a presentation file on a presentation unit that is located behind a firewall and that the data rate of the transmission is very high. According to one embodiment, the e-mail containing the presentation file is sent to a mail server handling the e-mails addressed to the presentation unit. Then, either the mail server sends a notice to the communication device regarding the newly received e-mail or the communication device frequently checks the mail server for new mails.

According to one aspect of the invention, a presentation of a selected document, stored at a node of a network, is accomplished by means of a system essentially comprising a handheld device, a communication device for a presentation unit and a device that handles presentation requests. By operating the handheld device, a user is able to get a presentation unit specification from said communication device. This specification is then sent, together with an address to a selected document, to a device handling presentation requests. In this device the selected document is then converted to a format suitable for the presentation unit of said communication device, where the converted document is presented. The system could be using WAP-technology for sending information from the handheld device to the device that handles presentation requests and for accessing information on the Internet. Further the system could, for example, be using Bluetooth or IrDA for the communication between the handheld device and said communication device. The system also makes it possible to achieve mobile printing and portable access to documents that cannot be presented on a small display.

Additionally, the communication device for a presentation unit could be arranged to receive simple information files stored in the handheld device via the short-range communication and to pass the simple information files to the presentation unit. Such files could, for example, be text files comprising the information of a WAP-page, a list of telephone numbers, an address list, etc. To initiate the sending of such information for presentation, the handheld device is provided with a "present" or a "print" command in one of its menus and could also be provided with a button dedicated to performing this action.

A further scope of applicability of the present invention will become apparent from the detailed description given below. However, it should be understood that the detailed description and specific examples, while indicating preferred embodiments of the invention, are given by way of illustration only, since various changes and modifications within the spirit and scope of the invention will become apparent to those skilled in the art from this detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

Other features and advantages of the present invention will become apparent from the following detailed description of a presently preferred embodiment, with reference to the accompanying drawings, in which

FIG. **1** is a schematic view of a preferred embodiment of the invention,

FIG. **2** is a schematic view of a preferred embodiment of the invention utilising one preferred communication device for a presentation unit,

FIG. **3** is a schematic view of a preferred embodiment of the invention utilizing another preferred communication device for a presentation unit,

FIG. **4** is a schematic view of a preferred embodiment of the invention utilising yet another preferred communication device for a presentation unit,

FIG. **5** is a schematic view of a device that handles presentation requests according to a preferred embodiment of the invention,

FIG. **6** is a schematic view of a handheld device according to a preferred embodiment of the invention,

FIG. **7***a* is a schematic view of one embodiment of a communication device for a presentation unit,

FIG. **7***b* is a schematic view of another embodiment of a communication device for a presentation unit,

FIG. **8** is a flowchart of a process in the handheld device according to a preferred embodiment of the invention,

FIG. **9** is a flowchart of a process in the device that handles presentation requests according to a preferred embodiment of the invention,

FIG. **10***a* and *b* are flowcharts of a process in the communication device for a presentation unit in a preferred embodiment of the invention, and

FIG. **11** is a diagram over signals sent between different units of the system according to one embodiment of the invention.

DETAILED DESCRIPTION OF A PRESENTLY PREFERRED EMBODIMENT

In FIG. **1**, there is shown a preferred embodiment of a document presentation system. The system comprises at least one cellular telephone **100**, at least one information service provider **200** and at least one communication device **300**, in FIG. **1** represented by **300***a* and **300***b*, for a presentation unit.

The cellular telephone **100** is connected to a network **10**, e.g. Internet, Intranet, Wide Area Network (WAN), Local Area Network (LAN), via a wireless network **14**, e.g. a GSM-network, a TDMA-network, or another wireless network for cellular telephones, and a station **12** of a network operator providing wireless access to the network **10**. The access to the network **10** is accomplished by providing the station **12** with a gateway implementing, for example, the Wireless Application Protocol (WAP), Hyper Text Transport Protocol (HTTP) or some other protocol for communicating information over a wireless telephone network. Further, the cellular telephone **100** is provided with an interface for short-range wireless communication, such as Bluetooth.

The information service provider **200** is either directly connected to the station **12** or connected to the network **10**. However, the information service provider **200** should have some access to a network **10**. The information service provider **200** is either an Internet portal or a server for serving the documents of a company, and is providing access to documents, e-mails and/or other browsable information on the network in a format suitable for the telephone **100**. A print service device **210** is preferably built into the information service provider **200**, but could also be positioned externally from the information service provider **200**. The print service device **210** is a device that handles the request of a document to be printed and is preferably implemented as a software code executed in the information service provider.

The communication devices **300***a,b* are connecting printers **380** to the network **10**. A communication device could, for example, be implemented in a print server. It can be connected to a printer either as an external device **300***a*, as an internal device **300***b* or as an embedded device **300***b*. It is provided with an interface for short-range wireless communication, such as Bluetooth. The main tasks of the communication devices **300***a,b* are to make sure that a print file sent to one of the communication devices for printing is printed and to provide a cellular telephone with a printer specification. The printer specification comprises data regarding the transport route for a print file that is to be printed, the characteristics of the printer, and is described in more detail below.

The transport route of a print file between the printer service device **210** and the communication device **300***a,b* can be of different types, depending on what kind of communication device that is arranged to receive the print file. One route is to send the print file directly to one of the communication devices **300***a,b* over the network **10** using IP protocols. For this purpose a File Transfer Protocol (FTP), an Internet Printing Protocol (IPP), etc can be used. Another route is to send the print file in an e-mail, for example by means of SMTP or IMAP, to an e-mail address dedicated to a chosen communication device **300***a,b*. The e-mail is then routed to an e-mail server **390**, which provides the chosen communication device **300***a,b* with the print file. Yet another route is to send the print file back to the cellular telephone **100**, for example by means of IP and/or WAP Protocols, e.g. as Wireless Session Protocol (WSP) and via the interface for short-range communication **20** transmit the print file to the chosen communication device **300***a,b*.

In FIGS. **2**, **3** and **4** said different communication device implementations are shown. The figures also show more details of one embodiment of the invention, e.g. the network is described as the Internet **11** and the site of the printer is surrounded by a frame **50** that defines the nearby surroundings of the printer. The nearby surroundings of the communication device **300** shall, in this embodiment, be seen as a LAN of a company that is restricted to an office or a building, or a zone in which the short-range communications **20** of a telephone **100** can be received by the communication device **300**. As illustrated in FIGS. **2** and **3** a company LAN that is connected to the Internet is usually connected thereto via a security system such as a firewall **52** in order to reduce the risk of unauthorised access to the LAN. Further, there are shown three possible information service providers **200***a–c*, generally referred to as **200**, one provided by the operator at the station **12**, one provided as an internet portal **62**, and one provided by a company **60** for serving company documents to employees. In all three figures, the procedure for requesting a document to print can be the same, only with a variation of the data in the specification in accordance with the present communication device implementation.

FIG. **2** shows the communication device implementation that receives the print file as an e-mail. The e-mail is sent from one of the print service devices **210***a–c*, generally referred to as **210**, to an e-mail server **390** via the Internet **11**, the firewall **52**, and finally a LAN. When the print file comprised in the e-mail is received at the mail server **390**, it is transmitted to the communication device **300** for printing. By using an e-mail as a carrier of the print file, a possible problem of getting the print file through the firewall **52** is avoided.

FIG. **3** shows the communication device implementation that receives the print file directly from one of the print service devices **210***a–c* via the Internet, the firewall, and a network connection, which could be either a LAN or a single Internet access point. In this embodiment the print file is addressed direct to the communication device **300**.

FIG. **4** shows the communication device implementation that receives the print file from one of the print service devices **210***a–c* via at least the wireless telephone network **14**, the telephone **100** and the short-range communication link **20**. In this embodiment the print file does not need to be sent over the Internet if the print service device **210***a* positioned at the site of the operator is utilised, but the Internet could be used if suitable. This communication device implementation is preferred if there is no network connection or if there is other reasons for not connecting the communication device to a network.

Now referring to FIG. **5**, an embodiment of a print service device **210** is shown. The print service device **210** is provided with means **212** for receiving a request for a document to be printed. The request comprises a document address, indicating where the document is situated, and a printer specification. The printer specification comprises, for instance, information regarding how to send the print file to the printer and the manufacturer and the model of the printer; further details will be described below. The request is passed from the means **212** for receiving a request to a control means **214**. The control means **214** then extracts the

document address from the request and passes it on to a means for retrieving documents **216**.

The means for retrieving documents **216** then retrieves the document from the specified address within a memory **230** that the information provider has access to. The memory **230** could be a storage memory of the information service provider or any memory connected to and accessible via a network such as the Internet or an Intranet. The control means **214** passes the retrieved document and a portion of the printer specification containing data for selecting an appropriate device driver **220** to a document conversion means **218**.

The document conversion means **218** checks the retrieved document to find out the format in which the document has been saved. The most simple method for checking the format is to read and interpret the file extension, e.g. ".txt" for a plain text file, ".doc" for a Microsoft® Word file, ".pdf" for an Acrobat® Reader file, ".tif" for a Tagged Image File, ".ps" for a postscript file, ".dwg" for a AutoCad® file, etc. However, only checking the extension of the file is a rather uncertain method for learning the format, because the extension could be wrong or another program could use the same extension. Therefore, in a preferred embodiment, also the contents of the file are checked for portions indicating the format of the file. When the document format has been determined, the document conversion means **218** selects a corresponding application **2.24** for interpreting the document. The applications **224** could be a program dedicated to interpret one or a plurality of formats, or it could be a full version of a program that is even able to create documents of the present format.

Further, the document conversion means **218** uses a portion of the printer specification containing data for selecting an appropriate device driver **220** to determine what device driver **220** to use for creating the print file. The determination can be based on data defining the manufacturer and the model of the printer and/or the Plug and Play (PnP) device ID of the printer.

The conversion means now utilises the selected application **224** and the selected device driver **220** to convert the requested document to a printer file that the target printer can use.

The control means **214** also checks the printer specification for the route that the print file is to be sent. For example, the specification could contain an e-mail address, the IP-address of the communication device, or a return address to the cellular telephone.

The control means **214** provides a send print file means **222** with the information of the route, e.g. an address, and the print file. Then the send print file means **222** sends the print file to the target printer via the requested route.

Now referring to FIG. **6**, there is shown a cellular telephone **100** arranged to request and receive a presentation unit specification via a short-range communication link **20** and to send a document request via a wireless telephone network. The cellular telephone is a WAP-enabled cellular telephone provided with a user interface **110** comprising a touch screen or a display and a keypad. The user interface is connected to an information browser means **112**, which, in accordance with the WAP-standard, browses information on a network in response to events at the user interface **110**. The information browser is implemented as a program that is processed by a Central Processing Unit (CPU) **114**. The communication via the short-range communication link is provided by means of a short-range communication interface **116**, preferably implementing Bluetooth, connected to the CPU **114**. For the communication over the wireless telephone network **14**, the cellular telephone is provided with an ordinary interface **118** for a cellular telephone.

Reference is now being made to FIGS. **7***a* and **7***b*, showing two embodiments of a communication device **300** according to the invention. Both embodiments of the communication device **300** are provided with a Central Processing Unit **302** (CPU) for handling the processes of the communication devices **300**. They are also provided with an interface **304** for communication via a short-range communication link **20** and an interface **0.306** for connecting a printer **380**. The communication device **300** according to FIG. **7***a* is arranged to receive print files via the short-range communication link **20**, in accordance with the route of FIG. **4**. The communication device **300** according to FIG. **7***b* is arranged to receive print files via the network **10**, in accordance with the routes of FIG. **2** or FIG. **3**. Therefore, the communication device **300** of FIG. **7***b* is provided with an additional interface **308** for communication over the network **10**. Both embodiments of the communication device **300** are provided with protocols for communication via the interfaces of each communication device **300**.

The primary object of the printer specification provided to the print service device via the telephone is to identify what formats the printer is able to print and to identify what route the file has to be sent to get to the printer. For these purposes, the print file comprises data regarding the manufacturer and the model of the printer and an address to which the print file is to be sent. The printer specification could, for example, be a text file like the one presented below:

```
<?xml version='1.0' standalone='yes'?>
<MobilePrint0.1>
<PrinterManufacturer></PrinterManufacturer>
<PrinterModel></PrinterModel>
<PrinterPnPID></PrinterPnPID>
<PrinterID></PrinterID>
<PrinterDeviceDriverURL></PrinterDeviceDriverURL>
<PrinterEmailAddress></PrinterEmailAddress>
<PrinterHostName></PrinterHostName>
<PrinterIPAddress></PrinterIPAddress>
<PrinterIPPURL></PrinterIPPURL>
<PrinterPDLs></PrinterPDLs>
<PrinterLocalName></PrinterLocalName>
<PrinterComment></PrinterComment>
</MobilePrint0.1>
```

In the specification above, the data of, for example, the printer manufacturer is to be positioned between the <PrinterManufacturer> and the </PrinterManufacturer>.

Now assume that a travelling service technician is travelling around the world servicing and repairing industrial robots. During an assignment he encounters a malfunction in a robot he is not very familiar with. He then needs the documentation of the robot. To get the documentation he asks for a printer having a Printer Server according to a preferred embodiment of the invention. He could also use his cellular telephone and instruct it to check the nearby surroundings for a communication device that is providing the service according to the invention.

Now referring to FIGS. **1** and **8**, when the service technician locates a communication device **300**, he uses his WAP-enabled cellular telephone **100** to connect, step **510**, to an operator that provides a WAP-gateway **12** to the Internet **10**. Then a WAP-page is presented, step **512**, on the display of the telephone **100** from which he can browse a web of linked WAP-pages by selecting links, **514**, presented on the display. Every time a link is selected, the telephone **100** checks if the link is pointing at another WAP-page or not,

step **516**. If the link points at another WAP-page, that page is downloaded and presented on the display, step **512**, and the procedure is repeated, steps **512**–**516**. A known WAP-page is also possible to get to by having the address of it stored in the telephone **100** and then selecting it or simply by typing the address in the information browser of the telephone **100**.

In one of the above-mentioned ways, the service technician gets an information page containing a link to the document he needs presented, step **512**. Now, he selects the link to the document, step **514**, which is not a WAP-page. The telephone **100** checks if the link is pointing at a WAP-page, step **516**, and finds out that it is not. Then the telephone **100** presents a print/download dialog, step **518**. If the service technician selects download, the document could be downloaded, step **522**, to a computer nearby, for example via the short-range communication link. However, if he selects print, the telephone **100** will broadcast an identity request via the short-range communication interface to all communication devices **300** within reach of the telephone **100** in regard of short-range communication **20**, step **524**. If there is no response to the request, an error message is presented in the display, step **528**, indicating that there is no compatible communication device **300** available. Then the telephone **100** returns to step **512** and presents the WAP-page again. If there is one or more communication devices **300** responding to the request, the communication devices **300** are presented on the display, step **530**. Then the service technician is enabled to select one of the communication devices **300** displayed, step **530**. When one communication device **300** is selected, the telephone **100** sends a request to the selected communication device **300** for a printer specification, step **534**. The communication device **300** returns a printer specification which is received at the telephone **100**, step **536**. Then a print request comprising a document address, given by the selected link, and the printer specification is sent to a print service device **210** of an information service provider **200**, step **538**, via the wireless telephone network **14**.

Now referring to FIGS. **1** and **9**, the print request is received at a print service device **210**, step **602**. The device **210** identifies the different parts of the request, step **604**, e.g. one document address and one printer specification, and uses the document address to retrieve the document that is to be printed, step **606**. Then the format of the retrieved document is identified, step **608**, as described above. When the format of the document is established, the print service device **210** uses this information and the presentation unit specification in the request to find an appropriate application for interpreting the document and a device driver, corresponding to the information of the printer specification, for converting the document to a print file suitable for the selected printer, step **610**. Then the document is converted to a print file, step **612**.

The print file is now ready to be sent to the selected communication device. However, before the print file is sent, the print service device selects a transmitting method in accordance with an address given in the printer specification, step **614**. Thereafter the print file is sent to the communication device via one of the routes described above, step **616**.

Now referring to FIGS. **1** and **10***a*, if the communication device **300** is arranged to receive print files via the telephone **100** and the interface for short-range communication **20** or directly via the Internet **10**, then the communication device **300** receives the print file, step **702**, and passes it to the printer for printing, step **704**.

Now referring to FIGS. **1** and **10***b*, if the communication device **300** is arranged to receive print files via e-mail, then the communication device could check the mail server **390** at predefined time intervals, step **712**. If a new mail is present, step **714**, then the communication device **300** retrieves the e-mail, step **716**, and thereafter extracts the print file from the e-mail, step **718**. When the print file is extracted, it is passed to the printer **380** for printing, step **720**.

Now the service technician can pick up the document from the printer and begin repairing the robot.

With reference to FIG. **11**, an example of communication between devices of the system and between the cellular telephone and a user is presented. The user initiates the WAP-browser of the cellular telephone, event **902**. Then he makes the telephone connect to an information service provider, event **904**. The telephone sends a connect request **906** to the information service provider. In response the information service provider sends an authentication request **908** to the telephone for authentication of the user. The user types a PIN-code into the telephone, event **910**, and the telephone passes the PIN-code and a SIM-code of the subscriber to the information service provider as an authentication message **912**. When the information service provider has accepted the user, it sends an initial WAP-page **914** to the telephone. The user is then able to select links on the pages presented on the telephone for browsing the information of the information service provider, event **916**. During the browsing the telephone requests pages according to selected links while the information provider responds to the request by sending the requested pages; this corresponds to the bi-directional communication of **918**. When the user selects a link to a document that is not a WAP-page, event **920**, the telephone presents a print dialog, event **922**. If the user chooses to initiate printing, event **924**, the telephone broadcasts a message **926** via the short-range communication interface, implementing for example Bluetooth, to all communication devices that are capable of acting as a communication device for the present invention. Said communication devices then respond to the request by sending a message **928**, comprising identification information, to the telephone. The user is then prompted to select one of the identified communication devices from a list, if there are more than one responding. If only one is identified, the user has to select that one or abandon the operation. When a communication device is selected, the telephone sends a request **932** for a printer specification to the selected communication device. The selected communication device responds by returning the printer specification **934** to the telephone. Thereafter the telephone sends a request **936**, comprising the printer specification and the document address, to the information service provider regarding printing the document at the specified printer. The request is passed **938** to the print service device of the information service provider for conversion of the document to a printable format and for sending the converted document to the selected printer. When the document is converted and if no errors occur, the print service device sends the print file **940** to the communication device, for example via e-mail. When the print file is sent, the information service provider gets a message **942** that the conversion was successful. If the conversion failed, the information provider gets a message **942** that the conversion failed. The message is passed along **944** to the telephone that presents it to the user.

What is claimed is:

1. A method for presentation of documents comprising the acts of:

selecting on a wireless communication device both a document provided by an information service provider and a presentation unit for presentment of the document;

receiving a specification on the wireless communication device from the presentation unit selected in the selecting act, and said specification specifying both a transport route to the presentation unit together with characteristics of the presentation unit; comprising at least one of:

a manufacturer name together with a model number of the presentation unit;

a filename of a device driver compatible with the presentation unit; and

a uniform resource locator (URL) of a device driver compatible with the presentation unit;

identifying both a device driver for the presentation unit based on the characteristics of the presentation unit specified in the specification and an application for interpreting the document selected in the selecting act based on a format of the document;

performing a conversion of the selected document to a presentation file suitable for presentment on the presentation unit, utilizing both the application and the device driver identified in the identifying act to perform the conversion; and

delivering the presentation file via the transport route specified in the specification to the presentation unit for presentment.

2. The method of claim **1**, wherein the transport route in the specification received by the wireless device from the presentation unit in the receiving act specifies at least one of:

an address of the presentation unit, URL of the presentation unit, and an e-mail address associated with the presentation unit.

3. The method of claim **1**, wherein the selecting act further comprises:

broadcasting an identity request from the wireless device;

receiving on the wireless device, responses to the identity request from proximate presentation units; and

selecting the presentation unit based on the responses to the identity request received in the receiving act.

4. The method of claim **1**, wherein the selecting act further comprises the acts of:

establishing a first communication link over a cellular telephonic network between the wireless communication device and the information service provider for selection of the document; and

establishing a second communication link over a short-range wireless communication protocol between the wireless communication device and the presentation unit for selection of the presentation unit.

5. A system for presentation of information comprising:

at least one presentation unit configured to respond to an identity request by delivering a specification specifying both a transport route to and characteristics of the presentation unit, comprising at least one of:

a manufacturer together with a model number of the at least one presentation unit;

a filename of a device driver compatible with the at least one presentation unit; and

a uniform resource locator (URL) of a device driver compatible with the at least one presentation unit;

at least one information service provider providing documents;

at least one wireless communication device configured to select one of the documents provided by the information service provider to receive the specification from the at least one presentation unit, and to generate a presentation request including the specification and information identifying the selected one of the documents; and

a device for handling presentation requests responsive to the presentation request to identify both a device driver for the at least one presentation unit based on the characteristics of the presentation unit in the specification and an application for interpreting the document based on a format of the selected one of the documents; and the device for handling presentation requests configured to convert the selected document to a presentation file suitable for presentment on the at least one presentation unit, utilizing both the identified application and the device driver to perform the conversion; and to deliver the presentation file via the transport route specified in the specification to the at least one presentation unit for presentment.

6. The system of claim **5**, wherein the transport route specified in the specification delivered by the at least one presentation unit specifies at least one of:

an address of the presentation unit, an URL of the presentation unit, and an e-mail address associated with the presentation unit.

7. The system of claim **5**, further comprising:

the wireless communication device further configured to establish a first communication link over a cellular telephonic network between the wireless communication device and the at least one information service provider for selection of the document; and to establish a second communication link by a broadcast of an identity request and a selection of the at least one presentation unit responding to the identity request utilizing a short-range wireless communication protocol.

8. The system of claim **5**, further comprising:

the wireless communication device further configured to establish a first communication link over a cellular telephonic network between the wireless communication device and the information service provider for selection of the document; and a second communication link over a short-range wireless communication protocol with the at least one presentation unit.

9. The system of claim **5**, further comprising:

an e-mail server coupled to the at least one presentation unit to accept delivery of the presentation file to the at least one presentation unit via e-mail when the transport route specified in the specification specifies an e-mail delivery of the presentation file.

10. The system of claim **5**, wherein further the information services provider comprises a server, and the device for handling presentation requests comprises processes executed on the server.

11. A presentation unit communication device for a presentation unit, comprising:

a short-range wireless communication interface for short-range wireless communications with a wireless communication device;

a presentation unit interface for communication with the presentation unit; and

a processor responsive to an identity request received via the short-range wireless communication interface to deliver a specification to the wireless communication device specifying both a transport route for delivery of a presentation file to the presentation unit together with characteristics of the presentation unit, including at least one of: a manufacturer

together with a model number of the presentation unit; a filename of a device driver compatible with the presentation unit; and a uniform resource locator (URL) of a device driver compatible with the presentation unit; and the processor further responsive to a delivery of a presentation file from an information service provider to communicate the presentation file via the presentation unit interface to the presentation unit for presentment.

12. The presentation unit communication device of claim **11**, wherein the transport route specified in the specification specifies at least one of:

an address of the presentation unit, an URL of the presentation unit, and an e-mail address associated with the presentation unit.

13. The presentation unit communication device of claim **11**, further comprising:

a network interface for communication over a network with the information service provider and receipt there from of the presentation file.

14. A wireless communication device comprising:

a short-range wireless communication interface for short range wireless communications;

a telephonic interface for cellular telephonic network communications; and

a processor responsive to user inputs to establish via the telephonic interface a telephonic communication link with an information service provider providing documents, to establish via the short-range wireless communication interface a short-range wireless communication link with a presentation unit, for receipt therefrom a specification specifying both a transport route to the presentation unit together with characteristics of the presentation unit, including at least one of; a manufacturer together with a model number of the presentation unit; a filename of a device driver compatible with the presentation unit; and a uniform resource locator (URL) of a device driver compatible with the presentation unit; and to forward the specification to the information service provider via the telephonic communication link, thereby to initiate conversion of a document provided by the information service provider to a corresponding presentation file, and delivery via the specified transport route of the corresponding presentation file to the presentation unit for presentment.

15. The wireless communication device of **14**, further comprising:

a processor further responsive to user inputs to broadcast an identity request via the short-range wireless communication interface and to establish the short-range communication link with a selected presentation unit responding to the identity request.

* * * * *

US007644018B1

(12) **United States Patent**
Yukie et al.

(10) **Patent No.: US 7,644,018 B1**
(45) **Date of Patent: Jan. 5, 2010**

(54) **SYSTEM AND METHOD FOR PROVIDING PUBLICLY VENDED CONTENT VIA A WIRELESS NETWORK**

(75) Inventors: **Satoru Yukie**, San Diego, CA (US); **Gina C. Eubanks**, Oceanside, CA (US); **Ken Aoki**, Tenafly, NJ (US)

(73) Assignees: **Sony Corporation**, Tokyo (JP); **Sony Electronics Inc.**, Park Ridge, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1517 days.

(21) Appl. No.: **09/542,139**

(22) Filed: **Apr. 4, 2000**

Related U.S. Application Data

(60) Provisional application No. 60/181,129, filed on Feb. 8, 2000, provisional application No. 60/180,990, filed on Feb. 8, 2000, provisional application No. 60/180,987, filed on Feb. 8, 2000, provisional application No. 60/180,985, filed on Feb. 8, 2000, provisional application No. 60/181,148, filed on Feb. 8, 2000, provisional application No. 60/181,144, filed on Feb. 8, 2000, provisional application No. 60/181,145, filed on Feb. 8, 2000, provisional application No. 60/180,992, filed on Feb. 8, 2000, provisional application No. 60/180,991, filed on Feb. 8, 2000, provisional application No. 60/181,105, filed on Feb. 8, 2000, provisional application No. 60/181,128, filed on Feb. 8, 2000, provisional application No. 60/180,988, filed on Feb. 8, 2000, provisional application No. 60/181,147, filed on Feb. 8, 2000, provisional application No. 60/181,127, filed on Feb. 8, 2000, provisional application No. 60/180,984, filed on Feb. 8, 2000, provisional application No. 60/180,993, filed on Feb. 8, 2000.

(51) **Int. Cl.**
G06Q 40/00 (2006.01)

(52) **U.S. Cl.** **705/34**; 705/35

(58) **Field of Classification Search** 705/10–44
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,128,755 A 7/1992 Fancher

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0 798 655 A	10/1997
EP	0 878 759 A	11/1998
EP	0 917 052 A	5/1999
EP	0 948 160 A	10/1999
WO	WO 97 41654 A	11/1997
WO	WO 98 45717 A	10/1998
WO	WO 99 18499 A	4/1999
WO	WO 99 48308 A	9/1999

OTHER PUBLICATIONS

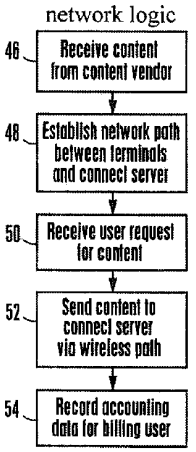
"Dynamic Icon Content" Research disclosure, Kenneth Mason Publications, Hampshire, GB. No. 298, Feb. 1, 1989, p. 99 XP000293031, ISSN: 0374-4353.
"Managing Devices With The Web" Howard M. et al. Byte, McGraw-Hill Inc. St. Peterborough, US. vol. 22, No. 9, pp. 45-46, XP000726362 ISSN: 0360-5280, Sep. 1, 1997.

Primary Examiner—Frantzy Poinvil
(74) *Attorney, Agent, or Firm*—John L. Rogitz

(57) **ABSTRACT**

A system and method for vending digitized content over an Internet network path, at least a portion of which is wireless, such as an SDMA high data rate directional wireless path. Content in the form of IP packets is provided to a user terminal via a connect server in response to user requests, which can be real-time requests or which can be requests embodied in a user-defined or system-defined user profile. The user is billed using one or more of several billing paradigms, including by the content piece, by the number of IP packets sent to the user, by the length of time the user accesses the system, by the subscription, and so on.

22 Claims, 1 Drawing Sheet



U.S. PATENT DOCUMENTS

5,617,526	A		4/1997	Oran et al. 345/779
5,634,012	A	*	5/1997	Stefik et al. 705/39
5,642,353	A	*	6/1997	Roy et al. 370/329
5,708,961	A	*	1/1998	Hylton et al. 725/81
5,754,939	A	*	5/1998	Herz et al. 455/3.04
5,799,285	A	*	8/1998	Klingman 705/26
5,813,007	A		9/1998	Nielsen 707/1
5,874,965	A		2/1999	Takai et al. 345/357
5,917,488	A		6/1999	Anderson et al. 345/838
5,949,876	A		9/1999	Ginter et al. 380/4
6,006,227	A		12/1999	Gelernter et al. 707/7
6,020,884	A		2/2000	MacNaughton et al. 345/747
6,020,916	A		2/2000	Gerszberg et al.
6,128,601	A	*	10/2000	Van Horne et al. 705/34
6,141,339	A		10/2000	Kaplan et al. 370/352
6,233,682	B1	*	5/2001	Fritsch 713/168
6,243,089	B1		6/2001	Gong 345/744
6,286,039	B1	*	9/2001	Van Horne et al. 709/221
6,288,716	B1		9/2001	Humpleman et al. 345/733
6,313,855	B1		11/2001	Shuping et al. 345/738
6,326,957	B1		12/2001	Nathan et al. 345/173
6,369,840	B1		4/2002	Barnett et al. 345/751
6,381,316	B2	*	4/2002	Joyce et al. 379/114.2
6,519,606	B2	*	2/2003	Burton et al. 707/102

* cited by examiner

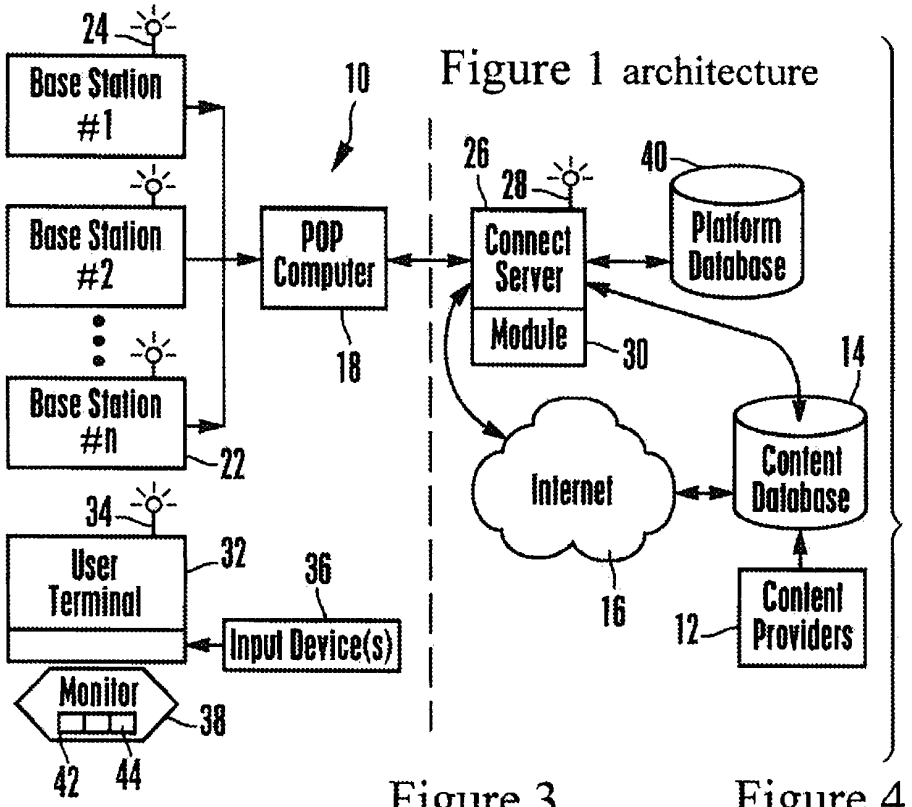


Figure 2
network logic

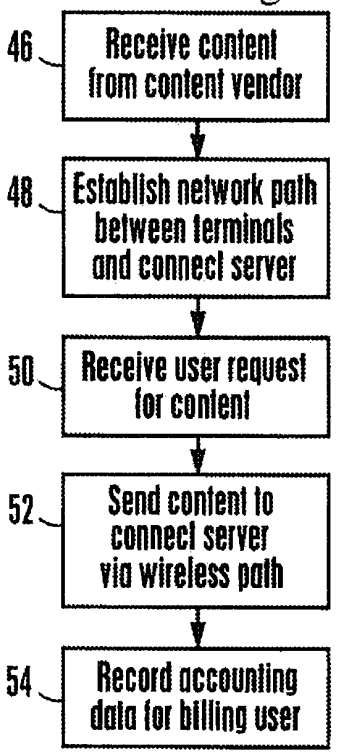


Figure 3
platform logic

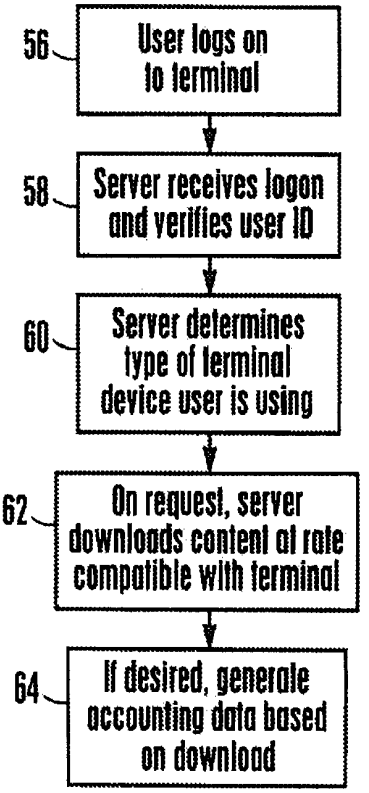
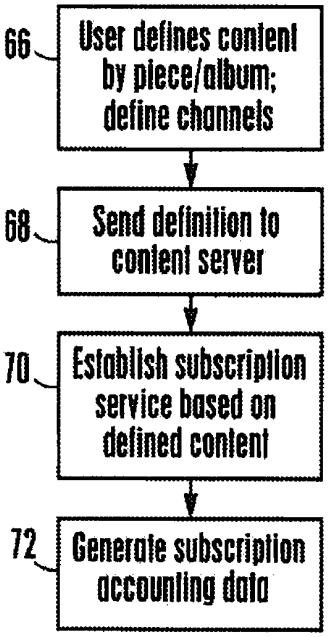


Figure 4
subscription logic



SYSTEM AND METHOD FOR PROVIDING PUBLICLY VENDED CONTENT VIA A WIRELESS NETWORK

RELATED APPLICATIONS

This application claims priority from the following U.S. provisional applications, all filed Feb. 8, 2000: Ser. Nos. 60/181,129, 60/180,990, 60/180,987, 60/180,985, 60/181,148, 60/181,144, 60/181,145, 60/180,992, 60/180,991, 60/181,105, 60/181,128, 60/180,988, 60/181,147, 60/181,127, 60/180,984, 60/180,993.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to vending digitized content via wireless networks.

2. Description of the Related Art

Most digitized content such as music, videos, games, and the like currently is sold in disk or tape format at retail outlets or based on catalogue orders or online orders. A disk or tape player can then be used to play the content. Other content is televised over cable channels that can be subscribed to, with payment for the content being based on a subscription service. Broadcast content, that is, wireless content, typically is not billed to the user, since use of the content by the user cannot be established. Such broadcast content consequently is subsidized by advertisers.

Accordingly, to buy publicly vended content, a user can purchase the content on a disk, which requires the user either to order the content and then wait for delivery or travel to a retail outlet. This is time consuming and consequently reduces the sales volume that content providers might otherwise achieve. Alternatively, the user can establish a subscription account for cable delivery, but this requires the installation of hardwired infrastructure, and it precludes the use of mobile, wireless devices.

The present invention, in critically observing the above-noted considerations relating to current models of vending content, has made further critical observations to develop the network-based, wireless content provision paradigm disclosed herein. Specifically, the present invention understands that personal computing devices such as personal digital assistants (PDAs) and the like have become ubiquitous. Along with this, instant worldwide computer communication has become available via the Internet. This juxtaposition of technologies has spawned thousands of new computer-driven opportunities to serve both consumers and content providers.

The present invention still further recognizes that unfortunately, in many cases Internet network data transfer rates are relatively low. Most personal computers, for example, currently have Internet data transfer rates of less than 100 kbps, making the vending of certain types of data, such as video, less than satisfactory over the Internet. Moreover, personal computers typically are connected to the Internet via telephone lines or fiber optic lines, meaning that the computers cannot be transported while connected to the Internet. Further, since the computers must be connected to hard-wired lines, cumbersome and costly line installation is required.

In the case of most wireless network paths, the data rates are relatively slow, and moreover the network paths are not continuous. That is, to establish a wireless network path, a user has had to dial up the wireless network and establish a connection through a telephony switch. This renders impractical applications that require continuous, high data rate, wireless network paths. The present invention recognizes the above-noted problems and provides the solutions disclosed herein, as well as various applications that take advantage of these solutions.

SUMMARY OF THE INVENTION

A general purpose computer is programmed according to the inventive steps herein. The invention can also be embodied as an article of manufacture—a machine component—that is used by a digital processing apparatus and which tangibly embodies a program of instructions that are executable by the digital processing apparatus to execute the present logic. This invention is realized in a critical machine component that causes a digital processing apparatus to perform the inventive method steps herein.

The invention can be implemented by a computer system including a computer network path, at least a portion of which is wireless. A user terminal is in the path, and a connect server communicates with the user terminal, with the connect server receiving requests for content generated from the user terminal and accessing a database of publicly vended content to fulfill the requests. A billing module is accessible to the connect server for generating accounting data based on the provision of the content in the database to users. Preferably, a portion of the network path is a directional wireless path that has no telephony switches therein, and the content is provided to the user terminal over the network in IP packets.

In a preferred embodiment, the billing module generates accounting data based on each request received. Alternatively, at least some requests can be embodied in a content subscription, and the billing module generates accounting data based on the subscription. As set forth in greater detail below, the subscription is established at least in part by a user profile. Still further, the billing module can generate accounting data based on a number of packets delivered to the user terminal, and/or based on at least one time period, and/or based on a type of the user terminal.

In another aspect, a method for vending content includes rendering the content in a digitized format, and making the content available on a computer network. The method further includes establishing at least one network path between the network and a user terminal, at least a portion of which is wireless. The content is sent to the user terminal via the network path, and accounting data representative of the provision of content to the user terminal is generated.

In still another aspect, a computer program device includes a computer program storage device that is readable by a digital processing apparatus. A program is on the program storage device. The program includes instructions that can be executed by the digital processing apparatus. The instructions include computer readable code means for billing a user for the provision of digitized packetized audio or video content from the Internet via a wireless network path to a user terminal associated with the user.

The details of the present invention, both as to its structure and operation, can best be understood in reference to the accompanying drawings, in which like reference numerals refer to like parts, and in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a schematic diagram showing the system of the present invention;

FIG. **2** is a flow chart showing the network logic of the invention;

FIG. **3** is a flow chart showing the platform logic; and

FIG. **4** is a flow chart showing the subscription logic.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring initially to FIG. **1**, a system is shown, generally designated **10**, for vending publicly available commercial content from multiple content providers **12**, in addition to also providing customized data mentioned further below. By "public content" is meant publicly available data that can be sold, such as but not limited to prerecorded commercial music and other audio including recordings, dictation, news, audio derived from video or text. Content can also include, commercial video such as movies, broadcast television, closed circuit monitoring and other feeds from video input devices. Content can also include non-commercial (e.g., governmental) audio and video that is broadcast over channels that are accessible to the public, whether by subscription or not. Also, content can include publicly-broadcast audio and video broadcasts of live events such as sporting events, music concerts, political speeches, and so on. Still further, public content can include still images from a camera or selection from a video data stream, computer-generated art or graphics, as well as games, such as video games (e.g., console-based, handheld-based, desktop-based, etc.) for download on a single-use basis, time-basis, purchase basis, and so on.

In contrast, by "customized data sources" is meant sources that provide data as defined by the user or that are associated with the user, and not with the general public. For instance, one customized data source might be a home security camera or a video camera located in the user's child's playroom or day care center as defined (set up) by the user, while another customized data source might be the user's home telephone, either wired or wireless or both, and yet another customized data source might be a stock market feed with portions of its stock information compiled into customized user-defined portfolio, or a customized database, such as, for example, an electronic journal of the user's personal finances. Still other customized sources include storage, such as data supplied by the user (e.g., images, audio, or other data stored in files), data requested by the user (e.g., search results), data stored on a device accessible by the server (e.g., an answering machine, desktop computer, different server). Personal data can also be provided, including but not limited to addresses, identifications, verification information such as biometric data, government documents such as passport images or passport data to replace a lost passport, financial information, credit information (e.g., for authorized third party access), user settings for electronic devices (e.g., channel selections or display parameters for a television where the device is a remote control, or operation settings for an appliance such as a refrigerator or oven). Customized data can be provided through a network connection to the connect server **26**, such as across the Internet or from another user terminal. Content can also be provided from a database within or otherwise accessible to the content server **26**, where the content server is also the content provider.

In the architecture shown, public content from the content providers **12** is made available in a content database **14** for provision of the data in Internet Protocol (IP) packets, with the database **14** preferably being accessible via the Internet **16**.

One or more point of presence (POP) network computers **18** controls plural base stations **22**, with the base stations **22** being in hard-wired or wireless communication with their associated POP computer **18**. A POP computer can be established by a network operation center (NOC) computer, an SMS computer, or equivalent. As shown in FIG. **1**, each base station **22** has an associated communication component **24**, schematically rendered in FIG. **1** as smart antennas.

FIG. **1** also shows a connect server **26** communicates with the POP computer **18** for receiving network information therefrom. The communication can be via hardwire, wireless, or the Internet. In the case of wireless communication the connect server **26** can have a communication component **28** that communicates with the POP computer **18** through one or more of the base stations **22**. Also, the connect server **26** accesses the content database, either through direct hardwired or wireless connection or through the Internet **16**.

The connect server **26** executes a software-implemented or hardware-implemented module **30** to undertake the novel logic set forth below to communicate with a user terminal **32**. The connect server **26** can connect one user terminal to another across various connections including wireline network such as the Internet, wireless system, or through the POP computer **18**. The user terminal **32** also has an associated communication component **34** that can communicate with the base stations **22** via a wireless network path and, hence, with the connect server **26**. It is to be understood that while FIG. **1** shows one exemplary architecture, the functions of the connect server **26** can be integrated into the user terminal **32** or integrated into the POP computer **18**. It is to be further understood that the communication components can be embedded in the respective computers or removable therefrom in the form of, e.g., PCMCIA cards. These components can be set to active, meaning that whenever the associated device is energized, the device communicates with the system **10**.

In any case, in accordance with present principles, the communication components **24**, **28**, **34** can be standard modems, in which case the network paths between the POP **18** (via the base stations **22**), connect server **26**, and user terminal **32** can be standard copper wire or optical fiber Internet path. In a preferred embodiment, however, as mentioned above at least a portion of the network path between the user terminal **32** and one or more of the remaining system components is a wireless path, such that the user terminal **32** can be mobile and portable while nonetheless communicating with the system **10** to receive digitized content, packetized in accordance with Internet Protocol (IP) principles, from the content database **14**. Thus, the preferred path between the user terminal **32** and the connect server **26** can be wireless.

The wireless path or paths can be implemented by the so-called iBurst™ or Ricochet™ communication systems, and more preferably can be implemented by a space division multiple access (SDMA) path as disclosed in, e.g., U.S. Pat. Nos. 5,515,378, 5,592,490, 5,642,353, 5,828,658, and 5,909,471 to ArrayComm, incorporated herein by reference, in which case one or more of the communication components **24**, **28**, **34** include appropriate interfaces for sending and receiving data in IP format. As set forth in the above-referenced patents, SDMA is essentially a directional version of time division multiple access (TDMA), which produces a very high rate of data transmission (i.e., in excess of one megabyte per second) as compared to other types of Internet communication protocols. Directional principles might also be applied to code division multiple access (CDMA) systems.

The terminal device **32** is a digital processing apparatus, such as a personal digital assistant made by Sony. Or, the terminal device **32** can be a personal computer made by International Business Machines Corporation (IBM) or Apple Computer, or the terminal device **32** may be any computer, including a laptop computer, a palmtop computer, a digital audio device, or any other appropriate digital appliance. Likewise, the POP computer **18** and the connect server

26 can be established by any of the above-named computing devices, and in addition can be established by larger computers, such as mainframe computers.

As shown in FIG. 1, the terminal device 32 is electrically connected to one or more input devices 36, e.g., a mouse and/or a keyboard and/or voice recognition device and/or Jog Dial™, which can be manipulated by a user of the system 10 as set forth further below to purchase content from the content database 14 for display on, e.g., an output device such as a monitor 38. Other output devices such as other computers, computer networks, audio speakers, printers, and so on can be used. In accordance with the below discussion, the connect server 26 accesses a platform database 40 that contains information on the user, including the below-described user profile, as well as information on the type of device that establishes the user terminal 32. Together, the connect server 26 and user terminal 32 can be thought of as establishing a platform, although a single connect server 26 might service multiple user terminals.

The flow charts herein illustrate the structure of the modules of the present invention as embodied in computer program software or in hardware. Those skilled in the art will appreciate that the flow charts illustrate the structures of logic elements, such as computer program code elements or electronic logic circuits, that function according to this invention. Manifestly, the invention is practiced in its essential embodiment by a machine component that renders the logic elements in a form that instructs a digital processing apparatus (that is, a computer) to perform a sequence of function steps corresponding to those shown.

In other words, the present logic may be established by a computer program that is executed by a processor within, e.g., the user terminal 32 or connect server 26 or POP computer 18 as a series of computer-executable instructions. In addition to residing on hard disk drives, these instructions may reside, for example, in RAM of the appropriate computer, or the instructions may be stored on a DASD array, magnetic tape, electronic read-only memory, or other appropriate data storage device. In an illustrative embodiment of the invention, the computer-executable instructions may be lines of compiled C++ compatible code.

As also shown in FIG. 1, a selector element 42 is provided. In the preferred embodiment the selector element 42 is a toolbar, although other graphical user interface (GUI) elements can be used, such as, for example, drop-down menus or text. Alternatively, an audio interface can be provided (e.g., selectable audio channels with audible indications for channel content and alerts).

As shown, the selector element 42 includes at least one and preferably plural source icons 44, with each icon 44 being associated with a respective data channel. Each channel can be defined by the user to receive a particular type of public content, such as video, audio, live broadcasts, games, and the above-mentioned customized data. In any case, the selector element 42 is presented on the monitor 38 so that a user can view the feeds as presented in the icons 44 and/or click on one or more of the icons 44 to enlarge or otherwise enhance the feed from the associated channel. An icon 44 can be established by a real-time feed of data from its associated source, or it can be static, or it can be established by a text display or audio display of data Referring now to FIG. 2, the logic of the connect server 26 can be seen.

Commencing at block 46, the server 26 receives public content from the content providers 12. Moving to block 48, a network communication path is established between the server 26 and user terminal 32. Then, at block 50 a user request for content is received as set forth in greater detail below. The content is sent to the user terminal 32 via a wireless path at block 52, and accounting data is recorded at block 54 for billing the user for the provision of the content.

FIG. 3 shows further details of the platform logic, i.e., the logic undertaken by the connect server 26/user terminal 32. Commencing at block 56, the user logs on to the system 10. This is preferably done automatically when the user energizes the user terminal 32. When the preferred SDMA system is used, no telephony switches need be used in the communication path. Preferably, the path includes a private, secure tunnel channel having plural threads, one for each content channel defined by the user. Logon can be accomplished using a handshaking protocol or a registration protocol, with a user ID being input by the user by means of the input device 36 or automatically by means of a removable card such as a PCMCIA card or Memory Stick™. Or, the user ID can be generated or confirmed through a biometric such as a fingerprint or by voice analysis.

At block 58, the connect server 26 receives the logon and verifies the user's identification and password if provided. When the user logs on, the server 26 correlates the user ID to the toolbar 42 and sends the toolbar 42 to the user terminal 32. Thus, the toolbar 42 is associated with a user, not necessarily with a particular network device, such that wherever the user logs on his or her own personalized toolbar is presented. The toolbar 42 is always displayed, although if desired the user can minimize the toolbar 42 and drag and drop the toolbar 42 on the display.

Moving to block 60, the connect server 26 determines the type of device the user is using as the terminal 32. On request including an explicit real-time request for content and/or according to a subscription established by a user-defined or platform-defined profile, at block 62 the connect server 26 accesses the requested content and sends the content to the terminal 32 at a rate compatible with the type of device being used as the terminal 32. Thus, a single user can access content using more than one type of device. Accounting data is then generated or collected at block 64.

This accounting data is generated based on one or more billing paradigms. For instance, billing for the above-described service can be accomplished by gathering accounting data either during the entire time a user is logged on, or on a monthly subscription basis, or on a piece basis as content is downloaded, i.e., the user can be billed for each song or video downloaded. Or, the user can be billed based on the number of IP packets sent to the user, i.e., by volume of content downloaded. Still further, when multiple channels are defined as set forth below, the user can be billed on a per-channel basis, and so on. Invoices are periodically sent to the user of the device 32 based on the accounting data.

FIG. 4 shows details of the subscription/profile logic. Commencing at block 66, the user defines a profile, or the system defines a profile, with the profile being stored in the platform database 40 and accessed by the connect server 26.

To store the profile in the database 40, at block 68 two-way communication can be established between the server 26 and terminal 32 and the profile transmitted to the server 26, or if one-way communication from the server to the terminal is available, then the profile is sent to a service provider and input into the database 40. If two-way communication is established, the user terminal 32 can send requests and commands to the server 26, such as a purchase request, a pause command, an answer phone command, a move camera command, etc.

In defining a profile, either the user or the system 10 or both can establish channels, with each channel being represented by a respective icon 44 on the toolbar 42. For instance, one

channel might be for video, another for audio (such as a user-selected radio station), and another for user-selected sporting events. Also, channels can be provided for receiving the above-described customized data that is personal to the user.

Channels can further be indicated or labelled by number, or by user-defined labels, such as "jogging", "sleeping", "relaxation", and so on, with, e.g., relaxing music being provided over the "sleeping" channel and energizing music or news reports being provided over the "jogging" channel.

Moreover, the user profile can include times when the user wants each channel to be active. For instance, the user might define the "jogging" channel to be active from 12 P.M. to 2 P.M. The server **26** transmits the associated content at the time indicated in the user profile. Accordingly, content over plural channels might be sent simultaneously to the user, by means of, e.g., the above-described multi-threaded tunnel channel. Or, the user can explicitly request a particular channel or a particular piece of content apart from the user profile. If desired, a public channel can be provided over which all user terminals in the system **10** can receive, e.g., weather news.

While the discussion above focusses on user-defined profiles, it is to be understood that the server **26** can access a profile that is generic for whatever type of device the user is employing, as determined by the server at block **60** of FIG. **3**. For example, if the terminal **32** is an audio-video system for a school, it might receive data on multiple predetermined channels according to a service contract between the school and the service provider.

Moreover, when the user defines the profile, the server **26** can access an expert system module to respond to the profile. For instance, the user might select "jazz" on channel **1**, and the server **26** would then access its expert system to determine which content in the content database **14** (or which real-time publicly broadcast radio station) plays jazz. Jazz music would then be provided on channel **1** at whatever times the user might have specified in the profile. Moreover, the user profile can include alert monitoring instructions, including but not limited to what channel to monitor at what time, what types of events the server **26** should report, when and how to report the event, etc. For example, the user can specify to be alerted at a particular time by causing the terminal **32** to emit a beep when a user-selected sporting event is to start, so that the user can view the terminal **32**. Likewise, a user can be alerted to events that occur attendant to customized data, such as when motion is sensed by an in-home security camera, or when a stock price in a user-customized portfolio reaches a predetermined threshold. The alerts can include audible signals, flashing icons **44**, text or video messages, etc.

In any case, the user can select one or more of the channels defined above for immediate display or for display at the time scheduled. It is to be appreciated that the user can thus be billed not only based on a song-by-song or program-by-program basis, but on the basis of the subscription services that are essentially defined by the user profile. The subscription service based on the profile is established at block **70**, and accounting data therefor is generated at block **72**.

When content is provided to the terminal **32**, the software of the terminal **32** receives the IP packets of the content, and the IP packets are sent to the software port corresponding to the associated icon **44** (i.e., corresponding with the associated channel). That is, when more than one content stream in being received, the packets of the various streams are separated from those of other streams and sent to the respective ports associated with the selected channels. The packets are processed in accordance with IP packet processing principles known in the art, and the processed data is then displayed on the monitor **38**.

While the particular SYSTEM AND METHOD FOR PROVIDING PUBLICLY VENDED CONTENT VIA A WIRELESS NETWORK as herein shown and described in detail is fully capable of attaining the above-described objects of the invention, it is to be understood that it is the presently preferred embodiment of the present invention and is thus representative of the subject matter which is broadly contemplated by the present invention, that the scope of the present invention fully encompasses other embodiments which may become obvious to those skilled in the art, and that the scope of the present invention is accordingly to be limited by nothing other than the appended claims, in which reference to an element in the singular means "at least one". All structural and functional equivalents to the elements of the above-described preferred embodiment that are known or later come to be known to those of ordinary skill in the art are expressly incorporated herein by reference and are intended to be encompassed by the present claims. Moreover, it is not necessary for a device or method to address each and every problem sought to be solved by the present invention, for it to be encompassed by the present claims. Furthermore, no element, component, or method step in the present disclosure is intended to be dedicated to the public regardless of whether the element, component, or method step is explicitly recited in the claims. No claim element herein is to be construed under the provisions of 35 U.S.C. §112, sixth paragraph, unless the element is expressly recited using the phrase "means for".

We claim:

1. A computer system, comprising:

a computer network path, at least a portion of which is wireless;

at least one user terminal in the path;

at least one connect server communicating with the user terminal, the connect server receiving requests for content generated from the user terminal, the connect server accessing a database of publicly vended content to fulfill the requests; and

at least one billing module accessible to the connect server for generating accounting data based on at least one of: each request for content received, each piece of content delivered to the user terminal, a content subscription, a type of user terminal, and a number of IP packets delivered to the user terminal.

2. The computer system of claim **1**, wherein at least a portion of the network path is a directional wireless path.

3. The computer system of claim **2**, wherein the directional wireless path has no telephony switches therein.

4. The computer system of claim **1**, wherein the content is provided to the user terminal over the network path in packets.

5. The computer system of claim **4**, wherein the packets are in IP protocol format.

6. The computer system of claim **1**, wherein the subscription is established at least in part by a user profile.

7. The computer system of claim **1**, wherein the billing module generates accounting data based at least in part on at least one time period.

8. The computer system of claim **1**, wherein the network path has a data transmission rate of at least one Mbps.

9. The computer system of claim **1**, wherein the user terminal is established by a device type, the server detecting the device type and providing content according thereto.

10. A method for vending content, comprising the acts of:

rendering the content in a digitized format;

making the content available on a computer network;

establishing at least one network path between the network and a user terminal, at least a portion of the path being a private wireless path;

sending the content to the user terminal via the network path; and

generating accounting data representative of the provision of content to the user terminal; wherein the accounting data is based on at least one of:

a number of packets delivered to the user terminal; and

a type of the user terminal.

11. The method of claim 10, wherein the private wireless path is a directional path.

12. The method of claim 11, wherein the private wireless path is an SDMA path.

13. The method of claim 11, wherein the wireless path has no telephony switches therein.

14. The method of claim 11, wherein the content is provided to the user terminal over the network in packets.

15. The method of claim 14, wherein the packets are in IP protocol format.

16. The method of claim 11, wherein the accounting data is generated based at least in part on at least one time period.

17. The method of claim 10, wherein the network path has a data transmission rate of at least one Mbps.

18. The method of claim 10, wherein the user terminal is established by a device type, the device type dictating the content provided thereto.

19. A computer program device comprising:

a computer program storage device readable by a digital processing apparatus; and

a program on the program storage device and including instructions executable by the digital processing apparatus, the program comprising:

computer readable code means for billing a user for the provision of digitized packetized audio or video content from the Internet via a wireless network path to a user terminal associated with the user, including at least one of:

means for receiving a request for a piece of content from the user, the means for billing generating accounting data based at least in part on the request;

means for generating accounting data based at least in part on a number of packets sent to the user via the path;

means for generating accounting data based at least in part on a content subscription for content;

and means for generating accounting data based at least in part on a type of user terminal;

wherein the means for billing generates accounting data based at least in part on a number of packets sent to the user via the path.

20. The computer program device of claim 19, wherein the subscription is based at least in part on a user profile.

21. The computer program device of claim 19, wherein the network path has a data transmission rate of at least one Mbps.

22. The computer program device of claim 19, further comprising computer readable code means for determining a user terminal device type, the device type dictating the content provided to the user terminal.

* * * * *

US007212229B2

(12) **United States Patent**
Parulski et al.

(10) **Patent No.: US 7,212,229 B2**
(45) **Date of Patent: May 1, 2007**

(54) **DIGITAL CAMERA PROVIDING IMAGE PROCESSING FOR AN ATTACHABLE PRINTER**

(75) Inventors: **Kenneth A. Parulski**, Rochester, NY (US); **Jeffrey A. Small**, Rochester, NY (US); **Douglas W. Couwenhoven**, Fairport, NY (US); **Jason C. VanBlargan**, Rochester, NY (US); **Raymond E. Wess**, Holley, NY (US)

(73) Assignee: **Eastman Kodak Company**, Rochester, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 907 days.

(21) Appl. No.: **09/800,158**

(22) Filed: **Mar. 6, 2001**

(65) **Prior Publication Data**

US 2001/0013894 A1 Aug. 16, 2001

Related U.S. Application Data

(63) Continuation-in-part of application No. 08/833,106, filed on Apr. 4, 1997, now Pat. No. 6,940,541.

(51) **Int. Cl.**
H04N 5/225 (2006.01)
H04N 1/60 (2006.01)
B41J 3/00 (2006.01)

(52) **U.S. Cl.** **348/207.2**; 358/1.9; 347/2

(58) **Field of Classification Search** 348/207.2, 348/222.1, 207.1, 207.11, 552; 358/1.9, 358/518, 521, 2.1; 347/2
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,642,678 A		2/1987	Cok	
4,774,574 A		9/1988	Daly et al.	
4,962,419 A		10/1990	Hibbard et al.	
5,189,511 A		2/1993	Parulski et al.	
5,652,621 A		7/1997	Adams, Jr. et al.	
5,668,596 A	*	9/1997	Vogel	348/222.1
5,689,344 A	*	11/1997	Ebner	358/2.1
6,618,553 B1	*	9/2003	Shiohara	386/117
6,668,596 B1	*	12/2003	Wagener	66/84 A
6,856,425 B2	*	2/2005	Ozawa et al.	348/207.2
6,952,222 B2	*	10/2005	Sekine	348/207.2
2002/0080250 A1	*	6/2002	Ogawa et al.	348/232
2004/0095469 A1	*	5/2004	Lin	348/207.2
2004/0165083 A1	*	8/2004	Chang	348/207.2

(Continued)

FOREIGN PATENT DOCUMENTS

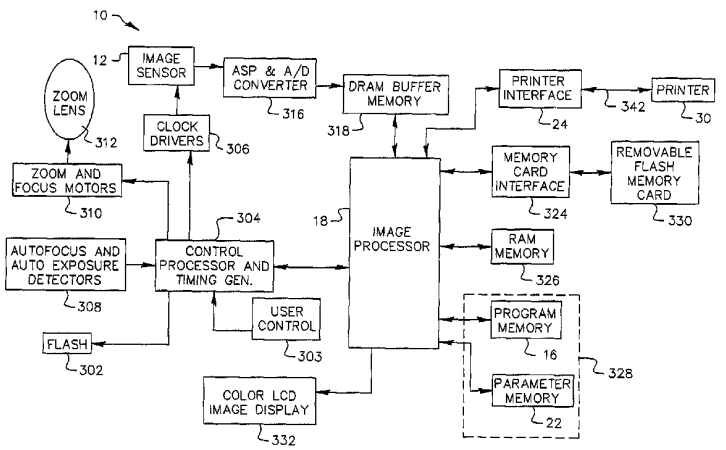
EP 0 869 656 A 10/1998

Primary Examiner—John M. Villecco
(74) *Attorney, Agent, or Firm*—Roland R. Schindler, II

(57) **ABSTRACT**

A system wherein already-existing computing and memory resources in an electronic camera are used to process an image for printing. Rather than duplicating, in printers, computing and memory resources that are already in digital cameras, significant computing and memory resources need exist only in the camera. A digital camera can support many different printers, each with its own set of parameters such as for example print size, pixel size, colorimetry, sensitometry, and artifacts compensation. Printer parameters are uploaded from the printer to the camera to provide a basis for image processing specific to the associated printer; whereby compensation may be done for variations in the printer characteristics which may occur as a result of printer manufacturing variations, and further so that compensation may be done for different media types which may be installed in the printer.

14 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

2004/0189811 A1* 9/2004 Ishiyama et al. 348/207.2
2005/0190265 A1* 9/2005 Small 348/207.2
2006/0109349 A1* 5/2006 Takashima 348/207.1
2006/0221191 A1* 10/2006 Nonaka et al. 348/207.1

* cited by examiner

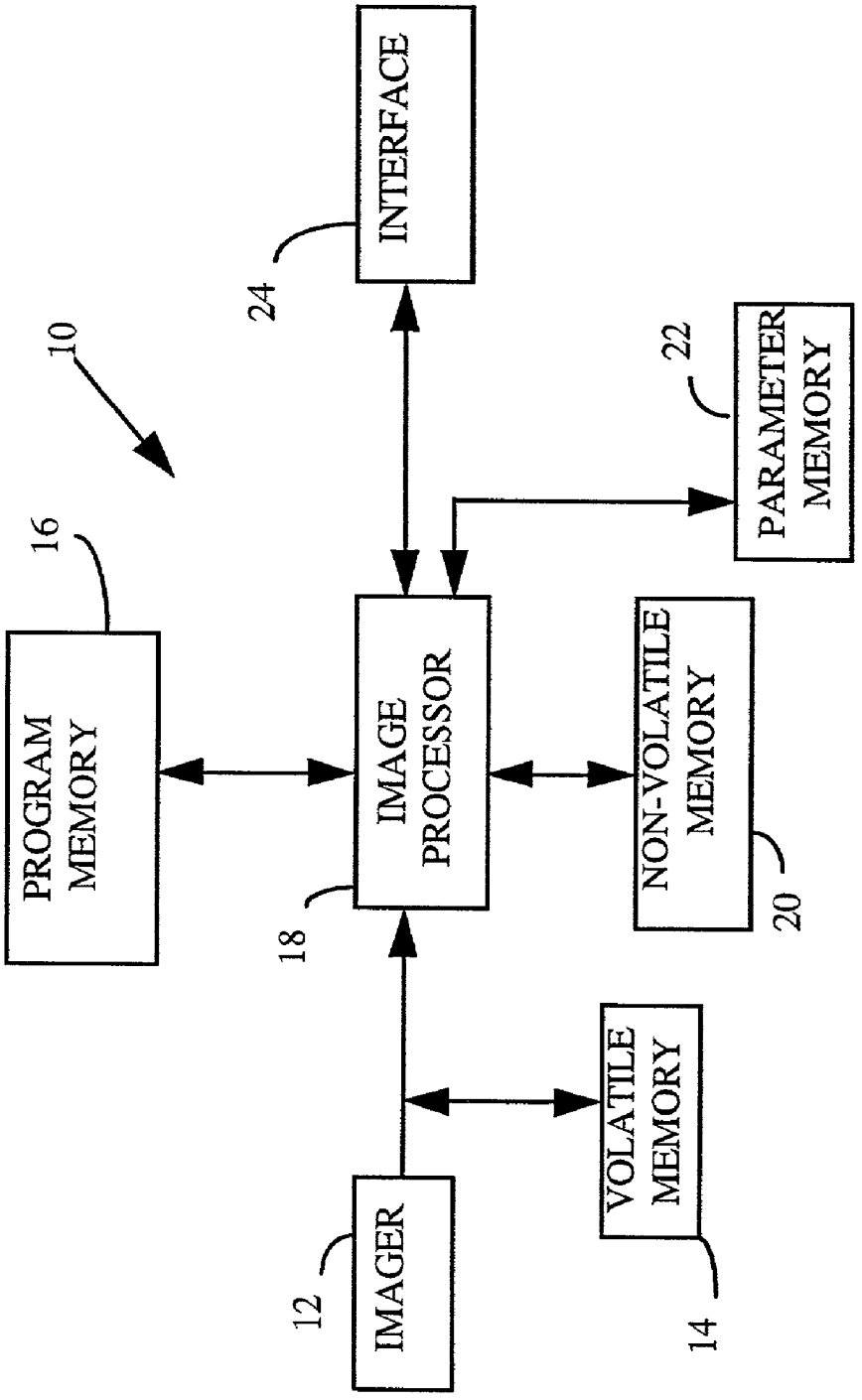

INTERFACE
24
10
22
PARAMETER MEMORY
16
PROGRAM MEMORY
IMAGE PROCESSOR
18
NON-VOLATILE MEMORY
20
VOLATILE MEMORY
14
IMAGER
12

FIG. 1

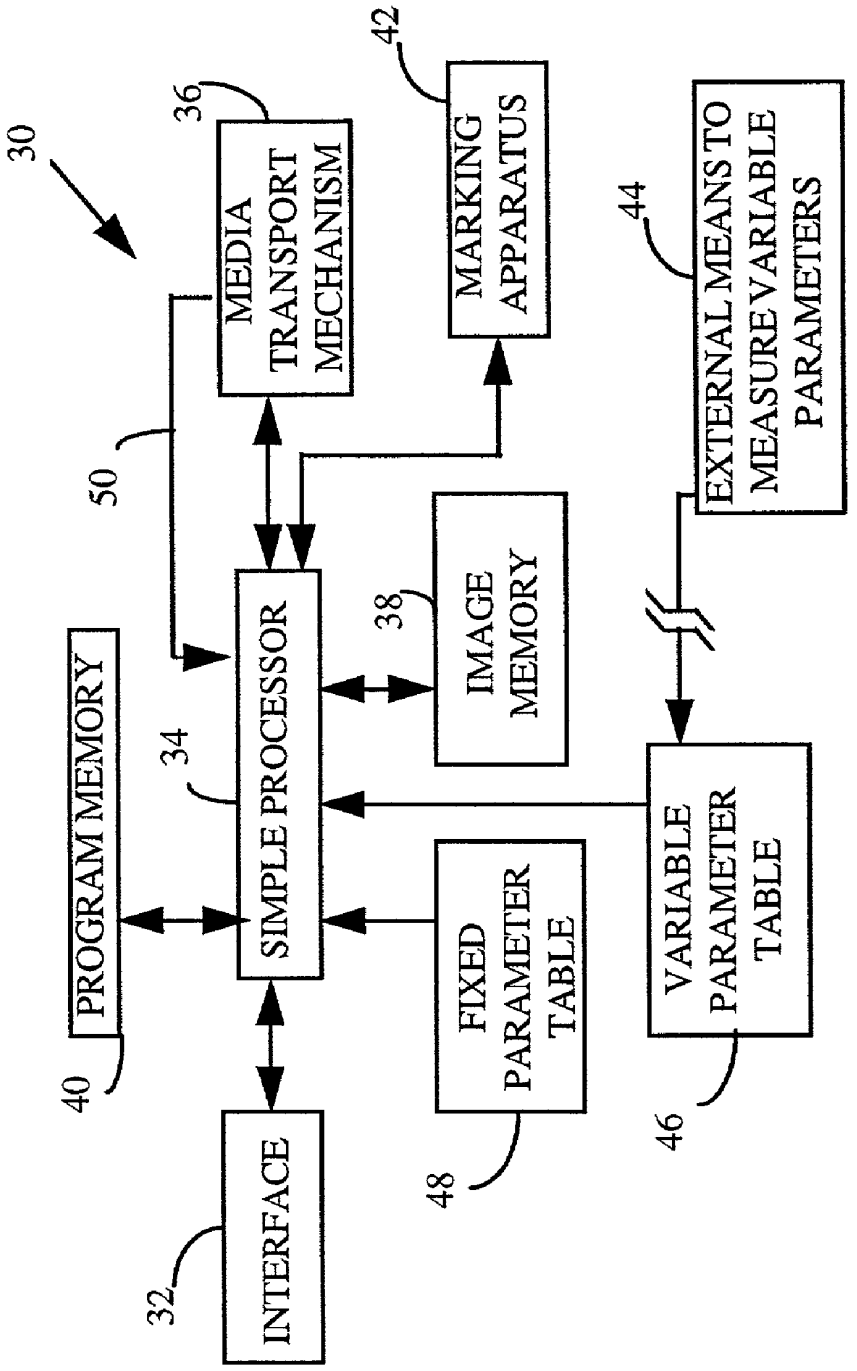

30
36
MEDIA TRANSPORT MECHANISM
42
MARKING APPARATUS
44
EXTERNAL MEANS TO MEASURE VARIABLE PARAMETERS
50
38
IMAGE MEMORY
34
SIMPLE PROCESSOR
PROGRAM MEMORY
40
FIXED PARAMETER TABLE
48
VARIABLE PARAMETER TABLE
46
INTERFACE
32

FIG. 2

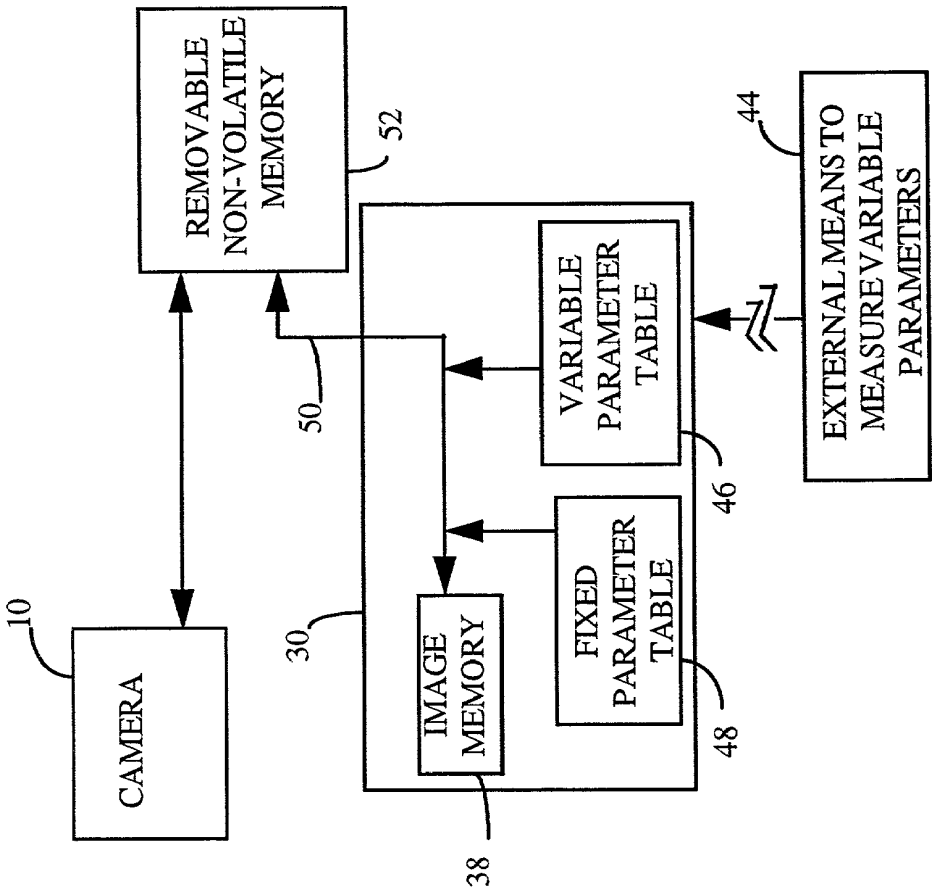

REMOVABLE NON-VOLATILE MEMORY
52
50
VARIABLE PARAMETER TABLE
46
FIXED PARAMETER TABLE
48
IMAGE MEMORY
38
30
CAMERA
10
EXTERNAL MEANS TO MEASURE VARIABLE PARAMETERS
44

FIG. 3

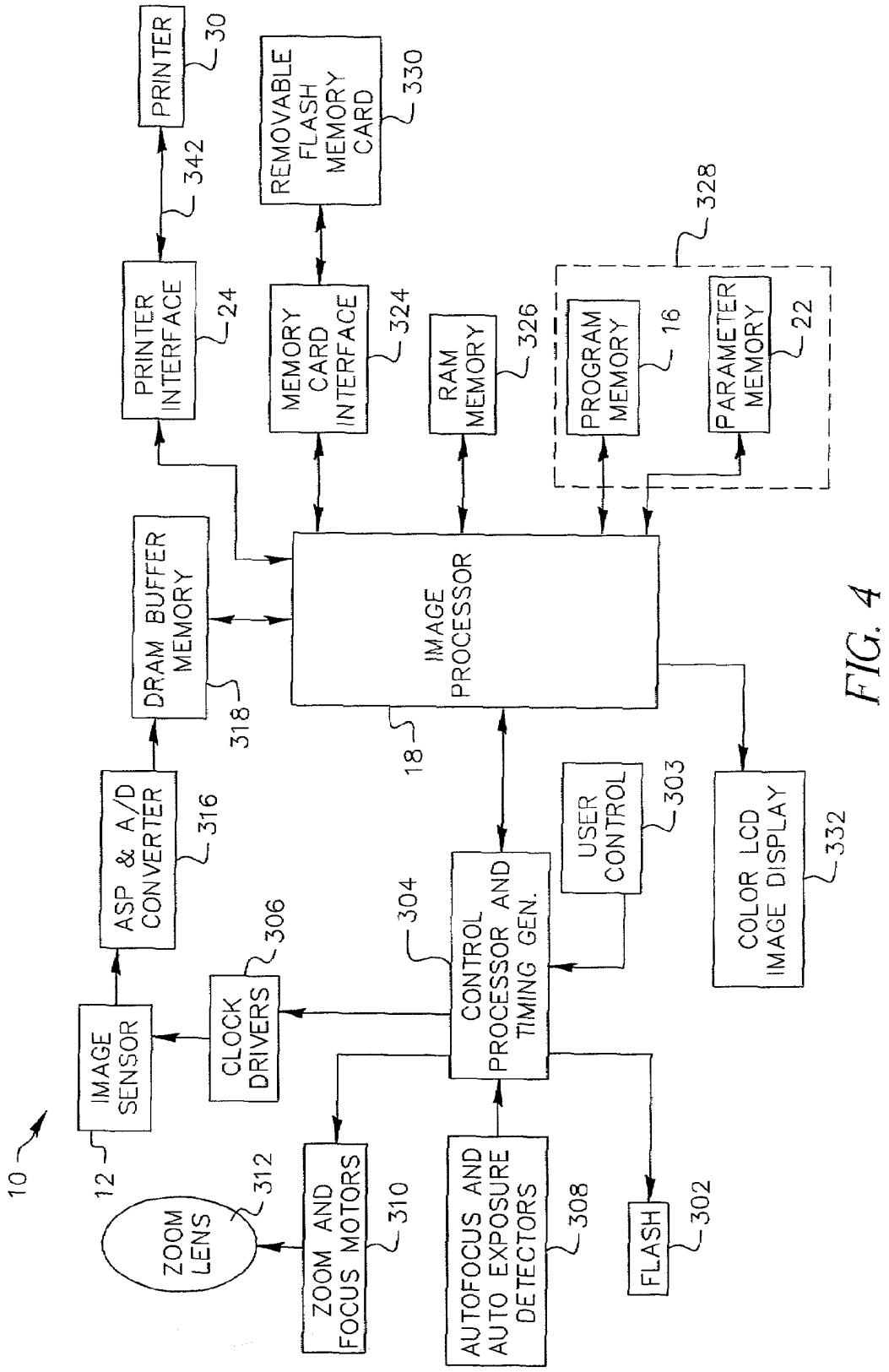

10
12
IMAGE SENSOR
ZOOM LENS
312
ZOOM AND FOCUS MOTORS
310
CLOCK DRIVERS
306
ASP & A/D CONVERTER
316
DRAM BUFFER MEMORY
318
IMAGE PROCESSOR
18
CONTROL PROCESSOR AND TIMING GEN.
304
AUTOFOCUS AND AUTO EXPOSURE DETECTORS
308
FLASH
302
USER CONTROL
303
COLOR LCD IMAGE DISPLAY
332
PRINTER INTERFACE
24
PRINTER
30
342
MEMORY CARD INTERFACE
324
REMOVABLE FLASH MEMORY CARD
330
RAM MEMORY
326
328
PROGRAM MEMORY
16
PARAMETER MEMORY
22

FIG. 4

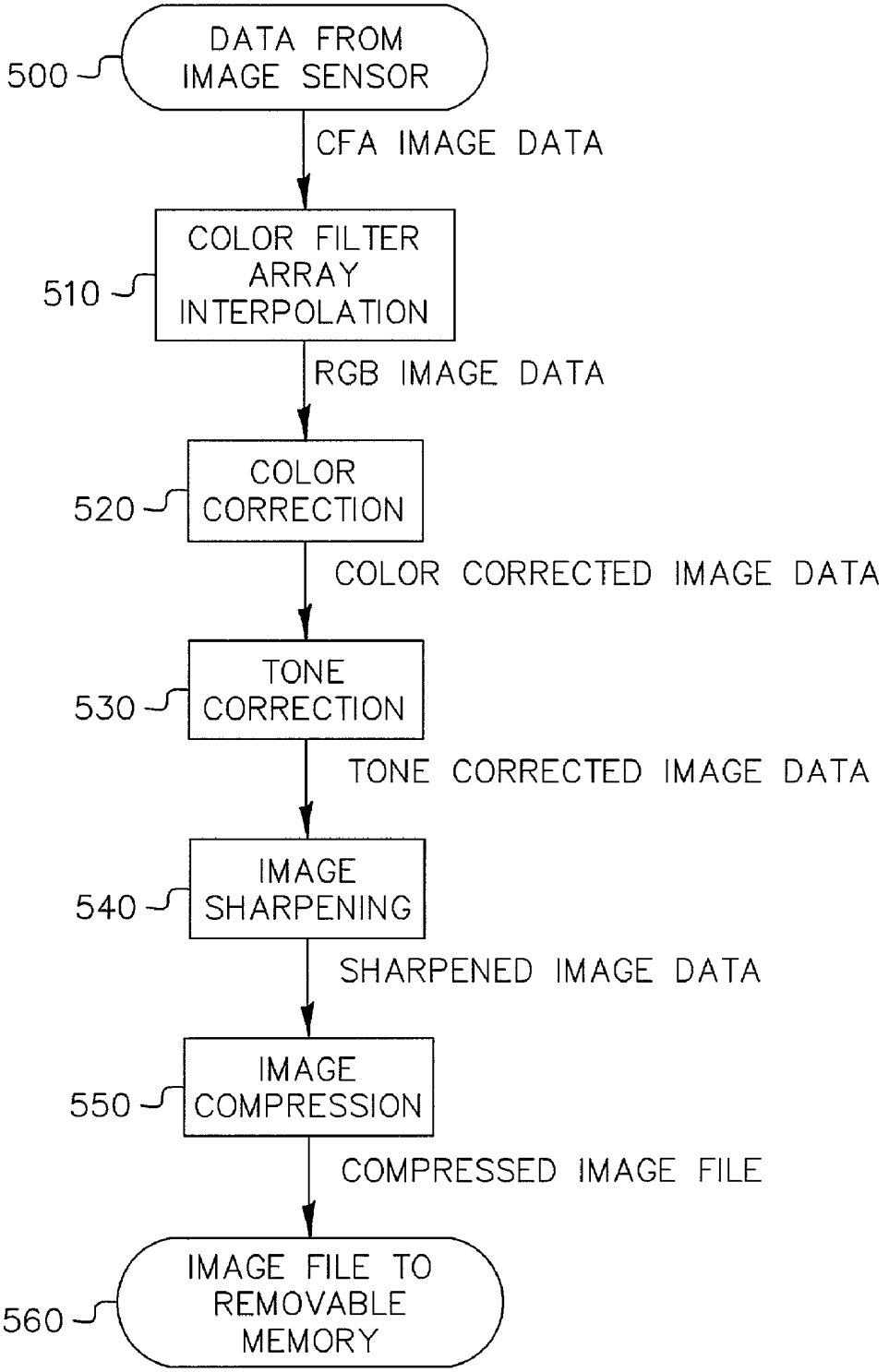

500
DATA FROM IMAGE SENSOR
CFA IMAGE DATA
510
COLOR FILTER ARRAY INTERPOLATION
RGB IMAGE DATA
520
COLOR CORRECTION
COLOR CORRECTED IMAGE DATA
530
TONE CORRECTION
TONE CORRECTED IMAGE DATA
540
IMAGE SHARPENING
SHARPENED IMAGE DATA
550
IMAGE COMPRESSION
COMPRESSED IMAGE FILE
560
IMAGE FILE TO REMOVABLE MEMORY

FIG. 5

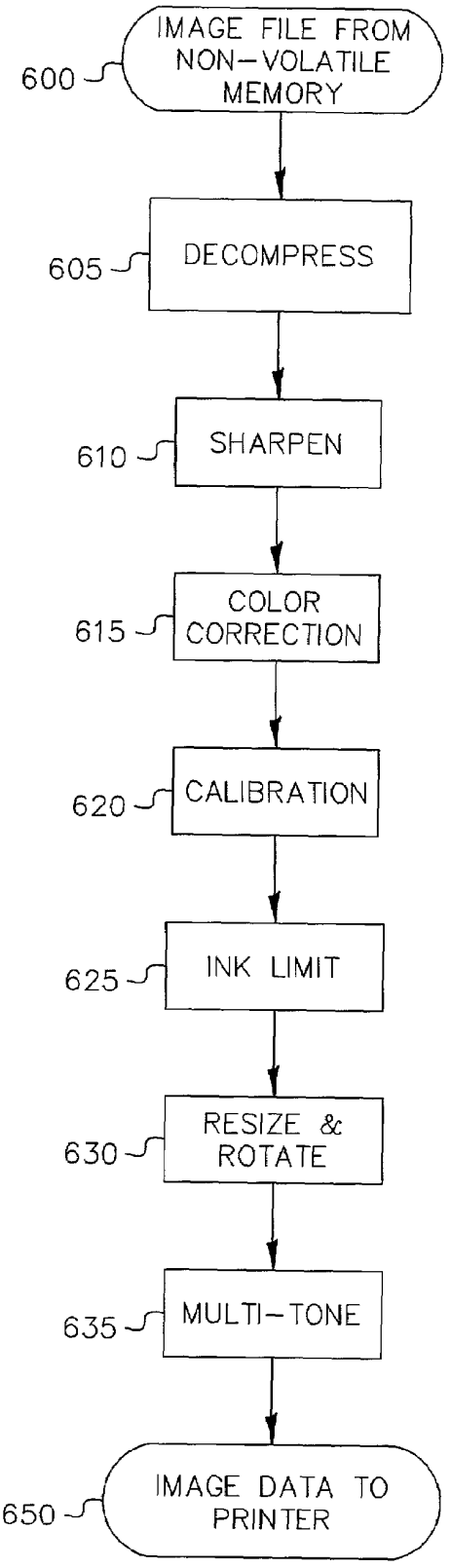

600
IMAGE FILE FROM NON-VOLATILE MEMORY
605
DECOMPRESS
610
SHARPEN
615
COLOR CORRECTION
620
CALIBRATION
625
INK LIMIT
630
RESIZE & ROTATE
635
MULTI-TONE
650
IMAGE DATA TO PRINTER

FIG. 6

DIGITAL CAMERA PROVIDING IMAGE PROCESSING FOR AN ATTACHABLE PRINTER

CROSS REFERENCE TO RELATED APPLICATIONS

This is a continuation in part of commonly assigned, U.S. patent application Ser. No. 08/833,106 filed in the name of Jeffrey A. Small on Apr. 4, 1997 now U.S. Pat. No. 6,940,541. Reference is made to commonly assigned U.S. patent application Ser. No. 08/977,382, filed in the name of Kenneth A. Parulski on Nov. 24, 1997.

FIELD OF THE INVENTION

The present invention relates to digital cameras and associated printers for producing hardcopy images captured by such cameras.

BACKGROUND OF THE INVENTION

Typically, images captured by digital cameras must be processed before they are printed. This processing is carried out in the printer. Significant computing and memory resources are required to process an image for printing. Accordingly, the printer must be provided with expensive computing and memory resources. One solution known in the prior art is to provide access to a stand-alone computer that is connectable to both the camera and the printer, either directly or by portable memory. This solution is inappropriate when the printer is to be used in remote locations distanced from the computer.

DISCLOSURE OF THE INVENTION

According to a feature of the present invention, we have come to appreciate that computing and memory resources, which already exist in electronic cameras in order for the camera to capture, process, compress, and store images, can be used to provide the computing and memory resources that are required to process an image for printing, particularly for printing on a portable, low cost ink jet printer.

It is an object of the present invention to provide a system wherein already-existing computing and memory resources in an electronic camera are used to process an image for printing. This is possible because the existing computing and memory resources are otherwise generally idle during the printing stage. Accordingly, it is a feature of the present invention that, rather than duplicating, in printers, computing and memory resources that are already in digital cameras, the present invention provides for camera and printer systems wherein significant computing and memory resources need exist only in the camera. Because such resources are already required by the camera in order to perform the camera functions, the cost of the camera is not increased. Because the resources are no longer required in the printer, the overall system cost is greatly reduced.

It is another object of the present invention to provide a digital camera that can support many different printers, each with its own set of parameters such as for example print size, pixel size, colorimetry, sensitometry, and artifacts compensation. Accordingly, it is a feature of the present invention to provide for uploading print drivers and printer parameters to the camera to provide a basis for image processing specific to an associated printer; whereby compensation may be done for variations in the printer characteristics which may occur

as a result of printer manufacturing variations, and further so that compensation may be done for different media types which may be installed in the printer, in particular different types of ink jet media installed in an ink jet printer.

According to another feature of the present invention, a digital camera includes an image sensor and a lens adapted to focus a scene onto the image sensor such that the image sensor captures analog image data. An analog-to-digital converter converts the analog image data captured by the image sensor to digital image data. An image processor performs first processing and compression of the digital image data to create a first-processed digital image file. A plurality of first-processed digital image files from the image processor are stored in removable digital memory. A digital image file, which is user-selected from the digital memory, is applied to a separate color printer having predetermined characteristics via an interface, wherein the image processor is adapted to perform second processing on the user-selected digital image file before the user-selected digital image file is applied to the interface.

According to a preferred embodiment of the present invention, color records of the user-selected digital image file are converted to multi-tone values during the second processing.

According to another preferred embodiment of the present invention, color records of the user-selected digital image file are processed during the second processing to provide ink limiting. The ink limiting is effected using type of printer, ink, and receiver media information provided by the separate color printer over the interface.

According to another preferred embodiment of the present invention, the separate color printer uses four ink colors, and the color records of the user-selected digital image file are converted to three image planes and are color corrected during said second processing to provide a set of color planes corresponding to each ink color of the separate color printer.

According to another preferred embodiment of the present invention, a color image display provides user-observable images of first-processed digital image files stored in the removable digital memory. User controls are coupled to the processor for user-selecting a digital image file to be second processed by the image processor.

According to another preferred embodiment of the present invention, the first processing includes: interpolation to provide red, green and blue image data values to provide red, green, and blue color planes; color correction of the red, green, and blue color planes; and image compression. The second processing includes decompression of the user-selected digital image file before the user-selected digital image file is applied to the interface.

The invention, and its objects and advantages, will become more apparent in the detailed description of the preferred embodiments presented below.

BRIEF DESCRIPTION OF THE DRAWINGS

In the detailed description of the preferred embodiments of the invention presented below, reference is made to the accompanying drawings, in which:

FIG. **1** is a schematic block diagram of a digital camera according to the present invention;

FIG. **2** is a schematic block diagram of a digital printer according to the present invention;

FIG. **3** is a schematic block diagram of a camera-printer system according to another embodiment of the present invention;

FIG. **4** is a detailed block diagram of a digital camera according to the present invention;

FIG. **5** is a flow diagram depicting the camera-related image processing operations provided by the digital camera of FIG. **3** in the process of capturing and storing images; and

FIG. **6** is a flow diagram depicting the printer-related image processing provided by the digital camera of FIG. **3** in the process of reading and printing images.

DETAILED DESCRIPTION OF THE INVENTION

The present description will be directed in particular to elements forming part of, or cooperating more directly with, apparatus in accordance with the present invention. It is to be understood that elements not specifically shown or described may take various forms well known to those skilled in the art.

Referring to FIG. **1**, a digital camera **10** provides significant image processing and memory resources to capture, process, compress, and store images. An imager **12** includes an array of image sensors such as, for example, light sensitive photoelements. Conventionally, a complete image frame is available in digital form from imager **12**, only for a short time interval. Thus, the captured image is temporarily stored in raw form in a volatile memory **14**. Various image processing algorithms are stored in a program memory **16**, and are executed by an image processor **18** in order to process the image data stored in volatile memory **14**. For example, the image processing algorithms may include all or some of the processes of image sensor tone scale compensation, color filter array interpolation, color space transformation, re-sizing, spatial filtering, and compression, as will be described in greater detail later in reference to FIG. **5**. The resulting processed image data is then typically stored in a nonvolatile memory **20**.

This stored image must be further processed prior to printing. Such further processing may include some or all of the steps of decompression, color space transformation into color planes that coincide with the process colors of the particular printer, re-sizing, rotation, and compensation for the printing process, as will be described in greater detail later in reference to FIG. **6**. In prior art systems, this further processing has been effected by computing and memory resources in the printer or in a stand-alone computer. According to the present invention, this further processing is performed using the resources which are already in camera **10**. It is advantageous to perform all of the processing using the resources in camera **10** in order to avoid the additional expense of including similar resources in the printer. To effect such image processing in camera **10**, the camera is provided with a parameter memory **22** and a printer interface **24**, both to be further described hereinafter.

Referring to FIG. **2**, a printer **30** includes a camera interface **32**, an inexpensive simple processor **34**, a media transport mechanism **36**, an image memory **38**, a program memory **40**, and a marking apparatus **42**. A processed image, received from camera **10** of FIG. **1** via interface **32**, may be stored by printer **30** in image memory **38** for subsequent printing by marking means **42** under the control of simple processor **34** and a program stored in program memory **40**, or the processed image may be printed immediately. Simple processor **34** need not be capable of executing printer compensation algorithms.

Parameters, which may vary as a result of manufacturing variations in the printer, may be measured by an external means **44** at the time of manufacture. Said parameters may

then be stored in a variable parameter table **46**, which is part of the printer. Camera **10** may query printer **30** to establish whether the printer will perform compensation for the variable parameters, or whether the camera should request and accept the variable parameters from the printer, and subsequently perform compensation for said variable parameters. The printer provides both fixed parameters from a fixed parameter table **48** and variable parameters from its variable parameter table **46** to the camera by means of camera and printer interfaces **24** and **32**, respectively. The camera stores these parameters in local parameter memory **22**.

When an image in either volatile memory **14** or nonvolatile memory **20** is selected for printing, image processor **18** processes the selected image, using the fixed and variable parameters which are stored in parameter memory **22**, and transmits the processed image to the printer by means of interfaces **24** and **32**. Processing may include all or part of the operations of image sensor tone scale compensation, color filter array interpolation, decompression, color space transformation, re-sizing, rotation, cropping, spatial filtering, and compensation for the printing process, but is not limited to these specific operations.

In addition, parameters which can vary during printing may also be transmitted by the printer to the camera during the printing process and used by image processor **18** to further compensate the image for printing process variations during the printing operation. The parameters may include temperature, ink viscosity, measured density, and any other parameters which are known to vary with the specific printing process employed by the printer.

Further, parameters characteristic of particular media material at media transport mechanism **36** may be determined by simple processor **34** over an interface **50** and transmitted to the camera. The media parameters may include parameters which vary with media type and parameters which vary between different batches of media due to manufacturing variations. Thus, compensation for the media parameters may be done by image processor **18** in the camera.

Any such media parameters, fixed parameters, and variable parameters may be transferred from printer **30** to camera **10** by means of a removable non-volatile memory cartridge **52** shown in FIG. **3**. The memory cartridge may also be used to transfer images between the camera and the printer. As used herein, the phrases "camera interface and printer interface" are intended to include cable connections, transferable memory, radiation transmission (light, microwave, infrared, etc.), and other forms of information transfer between components.

FIG. **4** is a block diagram showing portable digital camera **10** depicted in more detail than was shown in FIG. **1**. Digital camera **10** stores images on a removable flash memory card **330**, which is a specific type of non-volatile memory **14** (shown in FIG. **1**). Digital camera **10** includes a zoom lens **312** having zoom and focus motor drives **310** and an adjustable aperture and shutter (not shown). Zoom lens **312** focuses light from a scene (not shown) on image sensor **12**. Image sensor **12** may be, for example, a single-chip color CCD image sensor, such as a Toshiba model TCD5603D CCD sensor, available from Toshiba America Electronic Components, Irvine Calif. The model TCD5603D sensor has approximately 1536 columns and 1024 rows of photoelements, and uses the well-known Bayer color filter pattern. Other CCD or CMOS image sensors, having various image array sizes and color filter patterns, may alternatively be used.

Image sensor **12** is controlled by clock drivers **306**. Zoom and focus motors **310** and clock drivers **306** are controlled by control signals supplied by a control processor and timing generator circuit **304**. The control processor and timing generator **304** receives inputs from autofocus and autoexposure detectors **308** and controls a flash **302**. The analog output signal from image sensor **12** is amplified and converted to digital data by the analog signal processing (ASP) and analog-to-digital (A/D) converter circuit **316**. The A/D converter may alternatively be included a part of image sensor **12**, particularly if a CMOS image sensor is used. The digital data is stored in a DRAM buffer memory **318**, which is a specific type of volatile memory **14** (shown in FIG. **1**). The digital image data stored in DRAM buffer memory **318** is subsequently processed by a processor **18** controlled by the firmware stored in program memory **16**, which can be provided by a flash EPROM memory **328**. Flash EPROM memory **328** can be a single memory chip which can also provide parameter memory **22**.

The processed digital image file is provided to a memory card interface **324** which stores the digital image file on removable memory card **330**. Removable memory cards are known to those skilled in the art. For example, removable memory card **330** may be adapted to the Compact Flash interface standard, such as described in the CompactFlash Specification Version 1.3, published by the CompactFlash Association, Palo Alto, Calif., Aug. 5, 1998. Alternatively, removable memory card **330** can be adapted to the PCMCIA card interface standard, as described in the PC Card Standard, Release 2.0, published by the Personal Computer Memory Card International Association, Sunnyvale, Calif., September 1991. Removable memory card **330** can also be adapted to the well known secure digital (SD), solid state floppy disk card (SSFDC) or Memory Stick formats. Other types of non-volatile digital memory devices, such as magnetic hard drives, magnetic tape, or optical disks, could alternatively be used to store the digital images.

Processor **18** performs color interpolation followed by color and tone correction, in order to produce rendered sRGB image data as defined in IEC 61966-2-1 Multimedia systems and equipment—Color measurement and management—Part 2-1: Color management—Default RGB color space—sRGB available from the International Electrotechnical Commission, Geneva, Switzerland. The rendered sRGB image data is then JPEG compressed and stored as a JPEG image file on removable memory card **330** using an JPEG/Exif version 2.1 image file as defined in Digital Still Camera Image File Format Standard (Exchangeable Image File Format for Digital Still Camera: Exif), version 2.1, JBIDA-49-1998 available from the Japan Electronic Industry Development Association, Tokyo, Japan. The JPEG/Exif image files can be utilized by many different image capable devices, such as computers and imaging kiosks.

Processor **18** also creates a "thumbnail" size image that is stored in RAM memory **326** and supplied to color LCD image display **332**, which displays the captured image for the user to review. Electronic camera **300** is controlled by user controls **303**, such as a series of user buttons including a shutter release (e.g., capture button) (not shown) which initiates a picture taking operation. The graphical user interface displayed on color LCD image display **332** is controlled by the user interface portion of the firmware stored in program memory **16**. The graphical user interface is also used to select images for printing, and can optionally be used to select the number of copies and the print layout (e.g. the number images printed on one page). The images selected for printing may be immediately printed, if digital camera **10** is connected to printer **30**. If not, image processor **18** creates an "image utilization" file listing the image to be printed, the number copies for each image, and the print size, as described in commonly assigned U.S. patent application Ser. No. 08/977,382, filed by Parulski on Nov. 24, 1997, the disclosure of which is herein incorporated by reference. This "image utilization" file, which can conform to the well-known digital print order format (DPOF) is stored on removable flash memory card **330** along with the digital images captured by digital camera **10**.

FIG. **5** is a flow diagram depicting the image processing operations that are performed by image processor **18** in digital camera **10** in order to process the images from image sensor **12** stored in DRAM buffer memory **318**.

The Bayer pattern color filter array data (block **500**) which has been digitally converted by A/D converter **16** is interpolated in block **510** to provide red, green and blue (RGB) image data values at each pixel location in order to provide complete RGB color planes. Color filter array interpolation in block **510** can use the luminance CFA interpolation method described in commonly assigned U.S. Pat. No. 5,652,621, entitled "Adaptive color plane interpolation in single sensor color electronic camera" to Adams et al., the disclosure of which is herein incorporated by reference. The color filter array interpolation in block **510** can also use the chrominance CFA interpolation method described in commonly assigned U.S. Pat. No. 4,642,678, entitled "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal", to Cok, the disclosure of which is herein incorporated by reference.

A color space transformation is applied to the interpolated RGB color planes in order to provide color correction, prior to image storage. The RGB color planes are color corrected in block **520** using, for example, the 3×3 linear space color correction matrix **20** depicted in FIG. **4** of commonly assigned U.S. Pat. No. 5,189,511, entitled "Method and apparatus for improving the color rendition of hardcopy images from electronic cameras" to Parulski et al., the disclosure of which is incorporated herein by reference. The color correction matrix coefficients which are stored in program memory **16** in digital camera **10** can be, for example:

$$Rout = 1.50\ Rin - 0.30\ Gin - 0.20\ Bin$$

$$Gout = -0.40\ Rin + 1.80\ Gin - 0.40\ Bin$$

$$Bout = -0.20\ Rin - 0.20\ Gin + 1.40\ Bin$$

The color corrected color planes are tone corrected in block **530**. This tone correction **530** can use, for example, the lookup table corresponding to FIG. **2** of U.S. Pat. No. 5,189,511 cited above. This lookup table is stored in program memory **16** in digital camera **10**. Alternatively, color correction image processing operations **520** and tone correction image processing operations **530** can be provided by a three-dimensional lookup table (3D LUT). An example of such a 3D LUT is described in commonly assigned U.S. patent application Ser. No. 09/540,807 filed Mar. 31, 2000 in the names of Geoffrey Woolfe et al., the disclosure of which is incorporated herein by reference. The 3D LUT is more complex than the 3×3 matrix and single-channel LUT approach described above. However, it allows better control of color saturation. For example, it allows increased color saturation for most memory colors without increasing the saturation of flesh tone colors and near-neutral colors.

The image sharpening provided in block **540** of FIG. **5** can utilize the method described in commonly assigned U.S.

Pat. No. 4,962,419 ('419 patent), entitled "Detail processing method and apparatus providing uniform processing of horizontal and vertical detail components" to Hibbard et al., the disclosure of which is incorporated herein by reference.

The image compression provided in block **550** of FIG. **6** can use the method described in commonly assigned U.S. Pat. No. 4,774,574 (the '574 patent), entitled "Adaptive block transform image coding method and apparatus" to Daly et al., the disclosure of which is incorporated herein by reference.

The compressed image files are stored on removable flash memory card **330** as Exif image files. After a series of images have been taken and stored on removable memory card **330**, removable memory card **330** can optionally be inserted into a memory card reader in the user's host computer (not shown) in order to transfer the images captured by the digital camera to the host computer, where they can be viewed, e-mailed via the Internet, etc. To print images without using a host computer, an interface cable **342** can be used to connect between printer interface **24** in digital camera **10** and the corresponding camera interface in digital printer **30**. Printer interface **24** may conform to, for example, the well-know universal serial bus (USB) interface specification. Alternatively, printer interface **24** may conform to the RS-232 interface specification, the IEEE 1394 (Firewire) interface specification, or other cable interface specifications. Alternatively, the interface may utilize a wireless interface such as the well-known IrDA (Infrared Data Association) interface or an RF (radio frequency) interface such as the well-known Bluetooth RF interface.

FIG. **6** is a flow diagram depicting the printer-related image processing provided by image processor **18** in digital camera **10** in the process of reading and printing images on an ink jet printer. The images to be printed are selected by the user as described earlier. In block **600**, the image file to be printed is retrieved from non-volatile memory **20** in FIG. **1**, such as removable flash memory card **330** in FIG. **4**. If digital camera **10** compressed images prior to storage, for example by creating the JPEG/Exif image files described earlier, the image file is decompressed in block **605** to provide red, green and blue (RGB) color planes. In block **610**, each decompressed RGB color plane is sharpened in order to compensate for the sharpness degradation of the ink jet printing process. A preferred sharpening algorithm uses the well-known unsharp masking technique to produce a sharpened color plane Xs by creating a blurred version Xb of the original decompressed color plane Xo, and then computing

$$Xs=1+k(Xo-Xb)$$

where X is each of the R, G, and B color planes, and k is a gain factor. Gain factor k can be a parameter stored in fixed parameter table **48** in printer **30** for all printers of a given model, or alternately in variable parameter table **46** for a particular printer, which is measured for each batch of printers as they are manufactured. The gain factor is provided from printer **30** to digital camera **10** by means of camera and printer interfaces **24** and **32** respectively, when printer **30** is connected to camera **10**. The camera stores the gain factor k in camera parameter memory **22**.

The sharpened RGB color planes are color corrected in block **615**. The color correction block preferably uses a 3D LUT. The input to the 3D LUT is the RGB color plane, and the output is, for example, cyan, magenta, yellow, and black (CMYK) color planes corresponding to the color inks used as the process colors for printer **30**. This 3D LUT is preferably provided using the ICC profile format defined by the International Color Consortium. The 3D LUT profile values can be parameters stored in fixed parameter table **48** in printer **30** for all printers of a given model, or alternately in variable parameter table **46** for a particular printer, which is measured for each batch of printers as they are manufactured. The ICC profile is provided from printer **30** to digital camera **10** by means of camera and printer interfaces **24** and **32** respectively, when printer **30** is connected to camera **10**. The camera stores the ICC profile values in camera parameter memory **22**.

If printer **30** is an ink jet printer using more than four color inks, the CMYK color planes are further processed in block **615** to provide color planes corresponding to each ink. This processing preferably uses ink rendering processing to convert a single color plane (e.g. the cyan channel C) into two color planes (e.g. light cyan Cl and dark cyan Cd). Therefore, the output of color correction block **615** is set of color planes corresponding to the color inks used in the inkjet printer, which may for example use light cyan, dark cyan, light magenta, dark magenta, yellow, and black color inks as the process colors.

In block **620**, the color records are calibrated in order to correct for variations in tone scale. These variations may be may be the result of manufacturing variations in printer **10** or media (e.g. ink jet head or paper receiver) used by the printer. The calibration is provided by a one-dimensional lookup table applied to each color plane. The lookup table can be provided by parameters stored in variable parameter table **46** for a particular printer, which is measured for printer **30** as it is manufactured. Alternatively, the lookup table can be created by image processor **18** in digital camera **10** using parameters or settings provided by printer **30**. The parameters or settings can include, for example, data indicating the type of media (e.g. ink jet head or paper receiver) used by the printer, or data such as the ink viscosity, humidity, etc. The lookup tables, parameters, or settings are provided from printer **30** to digital camera **10** by means of camera and printer interfaces **24** and **32** respectively, when printer **30** is connected to camera **10**. The camera stores this data in camera parameter memory **22**.

In block **625**, the calibrated color planes corresponding to the inks of the inkjet printer are processed to provide ink limiting. This processing reduces the amount of ink that is deposited on the receiver media in high ink laydown areas. This is required in order to minimize deglossing and ink bleeding problems that reduce the image quality. It also reduces the stickiness, long drying time and delamination problems caused by laying down too much ink. The ink limiting step typically limits the total ink provided by all ink color planes to a maximum of 2 to 3 times the maximum amount of ink provided by a single color plane. The exact limit depends on the combination of the printer, ink, receiver media, and, to some extent, the humidity. To determine the appropriate limit to make a print, the type of printer, ink, and receiver media can be communicated from printer **30** to digital camera **10**. In some embodiments, ahumidity sensor in printer **30** can be used to sense the approximate humidity. A corresponding humidity parameter can be communicated, along with the type of printer, ink, and receiver media, from printer **30** to digital camera **10** by means of camera and printer interfaces **24** and **32** respectively, when printer **30** is connected to camera **10**. The camera stores this data in camera parameter memory **22**.

In block **630**, the color records corresponding to the process colors of the ink jet printer are resized and rotated if necessary. This converts the pixels captured by the digital

camera (e.g. the 1536 columns×1024 rows) to the appropriate number of pixels required by printer **30** in order to produce a selected image size. To perform this conversion, the number of pixels per inch used by printer **30** is communicated to digital camera **10**, when printer **30** is connected to camera **10**. The camera stores this data in camera parameter memory **22**.

In block **635**, the color records are converted to multi-tone values. Multi-toning is the process of reducing the bit depth of the image in a manner that reduces the spatial resolution while increasing the density resolution. Multi-toning is required in ink jet printers because the ink jet printers have few density levels (e.g. two density levels corresponding to ink or no ink, or four density levels corresponding to various ink drop sizes). Multi-toning using two density levels is also known as half-toning. Multi-toning may be provided using a variety of algorithms, such as the well-known "error diffusion" and "blue noise dithering" algorithms. In order for image processor **18** in digital camera **10** to provide multi-toning appropriate for printer **30**, the number of density levels, and the density of each level, is provided by printer **30**. More specifically, the density levels for each multi-tone level are stored in fixed parameter table **48** in printer **30** for all printers of a given model. The density levels are provided from printer **30** to digital camera **10** by means of camera and printer interfaces **24** and **32** respectively, when printer **30** is connected to camera **10**. The camera stores the density levels in camera parameter memory **22**.

In step **650**, the multi-tone color records corresponding to the inks used in printer **30** are communicated from digital camera **10** to printer **30** by means of camera and printer interfaces **24** and **32** respectively. Printer **30** produces an ink jet print using the multi-tone color records by controlling the marking apparatus **42** and media transport mechanism **36** using simple processor **34**.

In an alternative embodiment, some or all of the printer parameters are provided on a removable media, such as a floppy disk (not shown) or removable flash memory card **330**, rather than being stored in fixed parameter table **48** or variable parameter table **46**. The removable media is provided along with printer **30**, and is inserted into digital camera **10** so that the parameters can be downloaded and stored in parameter memory **22**. In the case of a floppy disk, the disk may be inserted into a separate host computer (not shown) and downloaded to the camera using a computer interface. The computer interface can use the same type of connection (e.g. USB, RS-232, IEEE 1394) as printer interface **24**. Alternatively, the parameters may be included as part of a printer driver which performs all of the processing described in relation to FIG. **6**. In this case, the printer driver firmware is downloaded from the removable media (supplied along with printer **30**) and stored in the program memory **16** of digital camera **10**.

In another alternative embodiment, some or all of the printer parameters, such as an ICC profile appropriate for particular "printer consumables" sold as a package, are provided as part of the printer consumables package. The printer consumables package can include, for example, printer receiver media (e.g. a quantity of photo grade ink jet paper) and a replacement color ink jet head for a particular type of printer. This printer consumables package can be provided with a nonvolatile digital memory, such as an EPROM, provided as part of the replacement color ink jet head. The parameters, such as the ICC profile, can be read from the EPROM memory by the simple processor **34** via the interface to the marking apparatus **42** when the color ink jet head is inserted into the printer **30**, and transferred to the digital camera **10** via the interface **32**.

The invention has been described in detail with particular reference to preferred embodiments thereof, but it will be understood that variations and modifications can be effected within the spirit and scope of the invention.

What is claimed is:

1. A digital camera for use with a separate color printer having predetermined characteristics, said camera comprising:

a housing;

an image sensor adapted to capture analog image data;

an analog-to-digital converter adapted to convert the analog image data captured by the image sensor to digital image data;

an image processor adapted to perform first processing and compression of the digital image data to create a first-processed digital image file;

digital memory in the camera housing, a plurality of first-processed digital image files from the image processor being stored in the digital memory; and

an interface to the separate color printer to which a digital image file, which is selected from the digital memory, is applied, wherein the image processor is adapted to perform second processing on the selected digital image file before the selected digital image file is applied to the interface.

2. A digital camera as set forth in claim **1** wherein color records of the selected digital image file are converted to multi-tone values during said second processing.

3. A digital camera as set forth in claim **2** wherein the multi-tone values are determined using a predetermined number of density levels provided by the separate color printer over the interface.

4. A digital camera as set forth in claim **3** wherein the multi-tone values are determined using a predetermined density for each of the density levels.

5. A digital camera as set forth in claim **1** wherein color records of the selected digital image file are processed during said second processing to provide ink limiting.

6. A digital camera as set forth in claim **5** wherein the ink limiting is effected using type of printer, ink, and receiver media information provided by the separate color printer over the interface.

7. A digital camera as set forth in claim **1** wherein:

the separate color printer uses four ink colors; and

the color records of the selected digital image file are converted to three image planes and are color corrected during said second processing to provide a set of color planes corresponding to each ink color of the separate color printer.

8. A digital camera as set forth in claim **7** wherein the provided set of color planes corresponding to each ink color of the separate color printer include at least four colors.

9. A digital camera as set forth in claim **7** wherein the provided set of color planes corresponding to each ink color of the separate color printer include light cyan, dark cyan, light magenta, dark magenta, yellow, and black.

10. A digital camera as set forth in claim **1** wherein:

the separate color printer uses four ink colors;

the color records of the selected digital image file are converted to three image planes; and

sequentially during said second processing:

the three image planes are color corrected to provide a set of color planes corresponding to each ink color of the separate color printer,

color records of the user-selected digital image file are processed to provide ink limiting, and

color records of the user-selected digital image file are converted to multi-tone values.

11. A digital camera as set forth in claim **1** wherein:

the housing includes a color image display for providing user-observable images of first-processed digital image files stored in the removable digital memory; and

user controls coupled to the processor for selecting a digital image file to be second processed by said image processor.

12. A digital camera as set forth in claim **1** wherein:

the first processing includes:

interpolation to provide red, green and blue image data values to provide red, green, and blue color planes,

color correction of the red, green, and blue color planes, and image compression; and

the second processing includes decompression of the selected digital image file before the selected digital image file is applied to the interface.

13. A digital camera as set forth in claim **12** wherein:

the separate color printer uses four ink colors;

the color records of the user-selected digital image file are converted to three image planes; and

sequentially during said second processing:

the decompression of the selected digital image file is effected,

the three image planes and are color corrected to provide a set of color planes corresponding to each ink color of the separate color printer,

color records of the user-selected digital image file are processed to provide ink limiting, and

color records of the selected digital image file are converted to multi-tone values.

14. A digital camera for use with a separate color printer having predetermined characteristics, said camera comprising:

a housing;

an image sensor adapted to capture analog image data;

an analog-to-digital converter adapted to convert the analog image data captured by the image sensor to digital image data;

an image processor adapted to perform first processing and compression of the digital image data to create a first-processed digital image file;

digital memory removably mounted in the camera housing, a plurality of first-processed digital image files from the image processor being stored in the removable digital memory; and

an interface to the separate color printer to which a digital image file, which is user-selected from the digital memory, is applied, wherein the image processor is adapted to perform second processing on the user-selected digital image file before the user-selected digital image file is applied to the interface.

* * * * *

Network Working Group
Request for Comments: 2393
Category: Standards Track

A. Shacham
Cisco
R. Monsour
Hi/fn
R. Pereira
TimeStep
M. Thomas
AltaVista Internet

December 1998

IP Payload Compression Protocol (IPComp)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1998). All Rights Reserved.

Abstract

This document describes a protocol intended to provide lossless compression for Internet Protocol datagrams in an Internet environment.

1. Introduction

IP payload compression is a protocol to reduce the size of IP datagrams. This protocol will increase the overall communication performance between a pair of communicating hosts/gateways ("nodes") by compressing the datagrams, provided the nodes have sufficient computation power, through either CPU capacity or a compression coprocessor, and the communication is over slow or congested links.

IP payload compression is especially useful when encryption is applied to IP datagrams. Encrypting the IP datagram causes the data to be random in nature, rendering compression at lower protocol layers (e.g., PPP Compression Control Protocol [RFC-1962]) ineffective. If both compression and encryption are required, compression MUST be applied before encryption.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

This document defines the IP payload compression protocol (IPComp), the IPComp packet structure, the IPComp Association (IPCA), and several methods to negotiate the IPCA.

Other documents shall specify how a specific compression algorithm can be used with the IP payload compression protocol. Such algorithms are beyond the scope of this document.

1.1. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC-2119].

2. Compression Process

The compression processing of IP datagrams has two phases: compressing of outbound IP datagrams ("compression") and decompressing of inbound datagrams ("decompression"). The compression processing MUST be lossless, ensuring that the IP datagram, after being compressed and decompressed, is identical to the original IP datagram.

Each IP datagram is compressed and decompressed by itself without any relation to other datagrams ("stateless compression"), as IP datagrams may arrive out of order or not arrive at all. Each compressed IP datagram encapsulates a single IP payload.

Processing of inbound IP datagrams MUST support both compressed and non-compressed IP datagrams, in order to meet the non-expansion policy requirements, as defined in section 2.2.

The compression of outbound IP datagrams MUST be done before any IP security processing, such as encryption and authentication, and before any fragmentation of the IP datagram. In addition, in IP version 6 [RFC-2460], the compression of outbound IP datagrams MUST be done before the addition of either a Hop-by-Hop Options header or a Routing Header, since both carry information that must be examined and processed by possibly every node along a packet’s delivery path, and therefore MUST be sent in the original form.

Similarly, the decompression of inbound IP datagrams MUST be done after the reassembly of the IP datagrams, and after the completion of all IP security processing, such as authentication and decryption.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

2.1. Compressed Payload

The compression is applied to a single array of octets, which are contiguous in the IP datagram. This array of octets always ends at the last octet of the IP packet payload. Note: a contiguous array of octets in the IP datagram may be not contiguous in physical memory.

In IP version 4 [RFC-0791], the compression is applied to the upper layer protocol (ULP) payload of the IP datagram. No portion of the IP header or the IP header options is compressed.

In the IPv6 context, IPComp is viewed as an end-to-end payload, and MUST not apply to hop-by-hop, routing, and fragmentation extension headers. The compression is applied starting at the first IP Header Option field that does not carry information that must be examined and processed by nodes along a packet's delivery path, if such IP Header Option field exists, and continues to the ULP payload of the IP datagram.

The size of a compressed payload, generated by the compression algorithm, MUST be in whole octet units.

As defined in section 3, an IPComp header is inserted immediately preceding the compressed payload. The original IP header is modified to indicate the usage of the IPComp protocol and the reduced size of the IP datagram. The original content of the Next Header (IPv6) or protocol (IPv4) field is stored in the IPComp header.

The decompression is applied to a single contiguous array of octets in the IP datagram. The start of the array of octets immediately follows the IPComp header and ends at the last octet of the IP payload. If the decompression process is successfully completed, the IP header is modified to indicate the size of the decompressed IP datagram, and the original next header as stored in the IPComp header. The IPComp header is removed from the IP datagram and the decompressed payload immediately follows the IP header.

2.2. Non-Expansion Policy

If the total size of a compressed ULP payload and the IPComp header, as defined in section 3, is not smaller than the size of the original ULP payload, the IP datagram MUST be sent in the original non-compressed form. To clarify: If an IP datagram is sent non-compressed, no IPComp header is added to the datagram. This policy ensures saving the decompression processing cycles and avoiding incurring IP datagram fragmentation when the expanded datagram is larger than MTU.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Small IP datagrams are likely to expand as a result of compression. Therefore, a numeric threshold should be applied before compression, where IP datagrams of size smaller than the threshold are sent in the original form without attempting compression. The numeric threshold is implementation dependent.

An IP datagram with payload that has been previously compressed tends not to compress any further. The previously compressed payload may be the result of external processes, such as compression applied by an upper layer in the communication stack, or by an off-line compression utility. An adaptive algorithm should be implemented to avoid the performance hit. For example, if the compression of i consecutive IP datagrams of an IPCA fails, the next k IP datagrams are sent without attempting compression. If the next j datagrams are also failing to compress, the next k+n datagrams are sent without attempting compression. Once a datagram is compressed successfully, the normal process of IPComp restarts. Such an adaptive algorithm, including all the related thresholds, is implementation dependent.

During the processing of the payload, the compression algorithm MAY periodically apply a test to determine the compressibility of the processed data, similar to the requirements of [V42BIS]. The nature of the test is algorithm dependent. Once the compression algorithm detects that the data is non-compressible, the algorithm SHOULD stop processing the data, and the payload is sent in the original non-compressed form.

3. Compressed IP Datagram Header Structure

A compressed IP datagram is encapsulated by modifying the IP header and inserting an IPComp header immediately preceding the compressed payload. This section defines the IP header modifications both in IPv4 and IPv6, and the structure of the IPComp header.

3.1. IPv4 Header Modifications

The following IPv4 header fields are set before transmitting the compressed IP datagram:

Total Length

The length of the entire encapsulated IP datagram, including the IP header, the IPComp header and the compressed payload.

Protocol

The Protocol field is set to 108, IPComp Datagram, [RFC-1700].

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Header Checksum

The Internet Header checksum [RFC-0791] of the IP header.

All other IPv4 header fields are kept unchanged, including any header options.

3.2. IPv6 Header Modifications

The following IPv6 header fields are set before transmitting the compressed IP datagram:

Payload Length

The length of the compressed IP payload.

Next Header

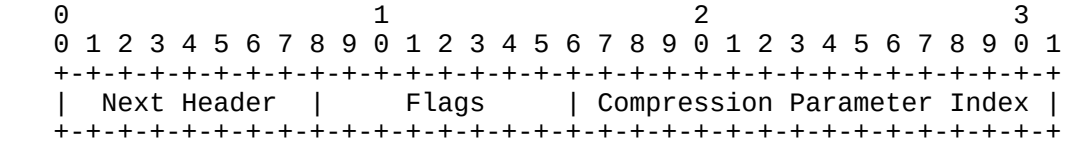
The Next Header field is set to 108, IPComp Datagram, [RFC-1700].

All other IPv6 header fields are kept unchanged, including any non-compressed header options.

The IPComp header is placed in an IPv6 packet using the same rules as the IPv6 Fragment Header. However if an IPv6 packet contains both an IPv6 Fragment Header and an IPComp header, the IPv6 Fragment Header MUST precede the IPComp header in the packet.

3.3. IPComp Header Structure

The four-octet header has the following structure:



Next Header

8-bit selector. Stores the IPv4 Protocol field or the IPv6 Next Header field of the original IP header.

Flags

8-bit field. Reserved for future use. MUST be set to zero. MUST be ignored by the receiving node.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Compression Parameter Index (CPI)

16-bit index. The CPI is stored in network order. The values 0-63 define well-known compression algorithms, which require no additional information, and are used for manual setup. The values themselves are identical to IPCOMP Transform identifiers as defined in [SECDOI]. Consult [SECDOI] for an initial set of defined values and for instructions on how to assign new values. The values 64-255 are reserved for future use. The values 256-61439 are negotiated between the two nodes in definition of an IPComp Association, as defined in section 4. Note: When negotiating one of the well-known algorithms, the nodes MAY select a CPI in the pre-defined range 0-63. The values 61440-65535 are for private use among mutually consenting parties. Both nodes participating can select a CPI value independently of each other and there is no relationships between the two separately chosen CPIs. The outbound IPComp header MUST use the CPI value chosen by the decompressing node. The CPI in combination with the destination IP address uniquely identifies the compression algorithm characteristics for the datagram.

4. IPComp Association (IPCA) Negotiation

To utilize the IPComp protocol, two nodes MUST first establish an IPComp Association (IPCA) between them. The IPCA includes all required information for the operation of IPComp, including the Compression Parameter Index (CPI), the mode of operation, the compression algorithm to be used, and any required parameter for the selected compression algorithm. The IPComp mode of operation is either a node-to-node policy where IPComp is applied to every IP packet between the nodes, or an ULP session based policy where only selected ULP sessions between the nodes are using IPComp. For each IPCA, a different compression algorithm may be negotiated in each direction, or only one direction may be compressed. The default is "no IPComp compression".

The IPCA is established by dynamic negotiations or by manual configuration. The dynamic negotiations SHOULD use the Internet Security Association and Key Management Protocol [ISAKMP], where IPSec is present. The dynamic negotiations MAY be implemented through a different protocol.

4.1. Use of ISAKMP

For IPComp in the context of IP Security, ISAKMP provides the necessary mechanisms to establish IPCA. IPComp Association is negotiated by the initiator using a Proposal Payload, which would

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

include one or more Transform Payloads. The Proposal Payload would specify a compression protocol in the protocol id field and each Transform Payload would contain the specific compression method(s) being offered to the responder.

In the Internet IP Security Domain of Interpretation (DOI), IPComp is negotiated as the Protocol ID PROTO_IPCOMP. The compression algorithm is negotiated as one of the defined IPCOMP Transform Identifiers.

4.2. Use of Non-ISAKMP Protocol

The dynamic negotiations MAY be implemented through a protocol other than ISAKMP. Such protocol is beyond the scope of this document.

4.3. Manual Configuration

Nodes may establish IPComp Associations using manual configuration. For this method, a limited number of Compression Parameters Indexes (CPIs) is designated to represent a list of specific compression methods.

5. Security Considerations

When IPComp is used in the context of IPSec, it is believed not to have an effect on the underlying security functionality provided by the IPSec protocol; i.e., the use of compression is not known to degrade or alter the nature of the underlying security architecture or the encryption technologies used to implement it.

When IPComp is used without IPSec, IP payload compression potentially reduces the security of the Internet, similar to the effects of IP encapsulation [RFC-2003]. For example, IPComp may make it difficult for border routers to filter datagrams based on header fields. In particular, the original value of the Protocol field in the IP header is not located in its normal positions within the datagram, and any transport layer header fields within the datagram, such as port numbers, are neither located in their normal positions within the datagram nor presented in their original values after compression. A filtering border router can filter the datagram only if it shares the IPComp Association used for the compression. To allow this sort of compression in environments in which all packets need to be filtered (or at least accounted for), a mechanism must be in place for the receiving node to securely communicate the IPComp Association to the border router. This might, more rarely, also apply to the IPComp Association used for outgoing datagrams.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

6. References

[RFC-0791] Postel, J., Editor, "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC-1700] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. Or see: http://www.iana.org/numbers.html

[RFC-2460] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC-1962] Rand, D., "The PPP Compression Control Protocol (CCP)", RFC 1962, June 1996.

[RFC-2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC-2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[ISAKMP] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[SECDOI] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[V42BIS] CCITT, "Data Compression Procedures for Data Circuit Terminating Equipment (DCE) Using Error Correction Procedures", Recommendation V.42 bis, January 1990.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Authors' Addresses

Abraham Shacham
Cisco Systems
170 West Tasman Drive
San Jose, California 95134
United States of America

EMail: shacham@cisco.com

Robert Monsour
Hi/fn Inc.
2105 Hamilton Avenue, Suite 230
San Jose, California 95125
United States of America

EMail: rmonsour@hifn.com

Roy Pereira
TimeStep Corporation
362 Terry Fox Drive
Kanata, Ontario K2K 2P5
Canada

EMail: rpereira@timestep.com

Matt Thomas
AltaVista Internet Software
30 Porter Road
Littleton, Massachusetts 01460
United States of America

EMail: matt.thomas@altavista-software.com

Working Group

The IP Payload Compression Protocol (IPPCP) working group can be contacted through its chair:

Naganand Dorswamy
Bay Networks

EMail: naganand@baynetworks.com

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Full Copyright Statement

Copyright (C) The Internet Society (1998). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Compression and Encryption in IPSec

06/24/1998

As more organizations open their intranets to remote users, suppliers and trading partners, security becomes increasingly important. Although encryption is not currently the most common security mechanism, its use will likely rise with the finalization of the IETF Internet Protocol Security (IPSec) standard in 1998.

However, encryption is very compute-intensive and can reduce performance dramatically. In many areas of performance, compression can help to mitigate performance delays. However, according to the IETF, the use of a higher-layer encryption protocol such as IPSec "reduces the effectiveness" of lower-layer compression protocols, such as PPP. This reduced effectiveness occurs because encrypted data cannot be compressed.

The compression process searches for redundant data strings in a file, and replaces the redundant strings with a token, thus reducing (compressing) the size of the file. The encryption process, on the other hand, randomizes data, making it unrecognizable. In an ideal randomization process, all redundancy is eliminated, so that compression becomes impossible or, at the very least, ineffective.

There are various Internet protocols and products that incorporate compression and/or encryption functions in their processes. However, problems arise if encryption is performed before compression. For example, IPSec, a layer 3 protocol, performs security functions but no compression functions; but PPP (Point-to-Point Protocol), a layer 2 protocol incorporates both compression and security functions. For example, when you are sending data, the network layer processing occurs first, which is where IPSec encryption occurs. Later, as the data gets closer to the wire, before sending the packet, PPP compression would normally be applied. But, according to Bob Monsour, vice president of marketing at Hi/fn (San Jose, Calif., www.hifn.com), an OEM networking products company, "once you do IPSec encryption, which randomizes the data, there is no more redundancy for the subsequent PPP compression to identify, which makes PPP compression ineffective. Therefore, the order in which compression and encryption are performed is important."

In order to overcome this problem, and incorporate the opportunity to use both compression and encryption effectively in IPSec, the IETF has established the IP Payload Compression (IPPC) working group to outline the proper process.

The IPPC working group is developing protocol specifications that define how payloads can be compressed before they are processed by an encryption protocol. The specifications will require that compression operations be performed before encryption of a payload by a protocol such as IPSec.

The two specifications, which currently will be negotiated, are LZS, by Hi/fn, and DEFLATE, designed by Phil Katz, president of PKWARE Inc. (Brown Deer, Wis., www.pkware.com) and creator of the PKUnzip compression utility. LZS compression is a lossless compression algorithm applied to IP datagram payloads. It is widely used today in networks in many routers, firewall, remote access servers and ISDN connectivity products. In order to solidify its position in the market, Hi/fn has recently acquired Cylan's Technologies and its IPSECure product, which performs compression and encryption according to IPPC working group guidelines. The DEFLATE protocol has been used for many years in networks on modem and other point-to-point links to transfer files for PCs and workstations.

The working group has already written a draft, which defines how to apply the Hi/fn LZS compression technique to the IPPC protocol. Although no date for finalization of the draft standard is available, users should know that in March 1998, at an interoperability workshop, 30 companies tested the implementation of IPSec, and six of those also tested the IPPC protocol.

Must Read Articles

APPENDIX O



Tokenization, Regulation and Expansion: Web3 Trends To Watch in 2023 (https://esj.com/articles/2023/01/09/we...

(https://esj.com/articles/2023/01/09/web3-trends-in-2023.aspx)



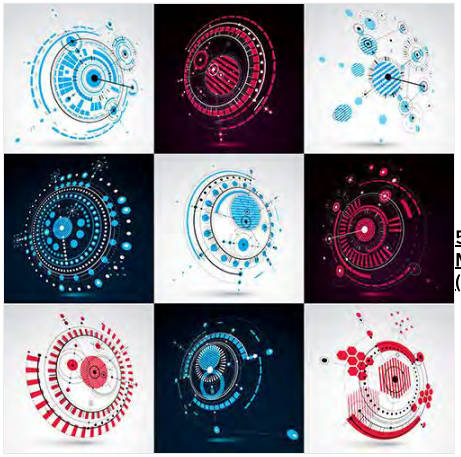
Top Web3, Metaverse and Blc Events in Q1 2023 (https://esj.com/articles/2022

(https://esj.com/articles/2022/12/05/blockchain-events-in-q1-2023.aspx)



Nike Steps Further into Web3 with Community-Based Virtual Sportswear Design...

(https://esj.com/articles/2022/12/07/nike-steps-further-into-web3-with-community-based-virtual-sportswear.aspx)



5 Takeaways from Microsoft's Metaverse Messaging (https://esj.com/articles/2022/

(https://esj.com/articles/2022/12/07/5-takeaways-microsoft-metaverse.aspx)

(https://converge360.com/Home.aspx)

(https://1105media.com/Home.aspx)

Application Development Trends (https://adtmag.com/home.aspx)

AWSInsider.net (https://awsinsider.net/home.aspx)

FutureTech360 (https://futuretech360.com/home.aspx)

Live! 360 (https://live360events.com/home.aspx)

MCPmag.com (https://mcpmag.com/home.aspx)

MedCloudInsider (https://medcloudinsider.com/home.aspx)

Prophyts (https://prophyts.com/home.aspx)

Pure AI (https://pureai.com/home.aspx)

Redmond (https://redmondmag.com/home.aspx)

Redmond Channel Partner (https://rcpmag.com/home.aspx)

TechMentor Events (https://techmentorevents.com/home.aspx)

Virtualization & Cloud Review (https://virtualizationreview.com/home.aspx)

Visual Studio Magazine (https://visualstudiomagazine.com/home.aspx)

Visual Studio Live! (https://vslive.com/home.aspx)

©1998-2023 1105 Media Inc (https://1105media.com/home.aspx). See our Privacy Policy (https://1105media.com/pages/privacy-policy.aspx), Cookie Policy (https://1105media.com/pages/cookie-policy.aspx) and Terms of Use (https://1105media.com/pages/terms-of-use.aspx). CA: Do Not Sell My Personal Info (https://1105media.com/pages/privacy-policy.aspx#_Toc520738059)

Problems? Questions? Feedback? E-mail us. (mailto:bnagel@1105media.com?subject=Regarding ESJ.com)

Performance Impact of Data Compression on Virtual Private Network Transactions

John P. McGregor and Ruby B. Lee
Department of Electrical Engineering
Princeton University
{mcgregor, rblee}@ee.princeton.edu

Abstract

Virtual private networks (VPNs) allow two or more parties to communicate securely over a public network. Using cryptographic algorithms and protocols, VPNs provide security services such as confidentiality, host authentication and data integrity. The computation required to provide adequate security, however, can significantly degrade performance. In this paper, we characterize the extent to which data compression can alleviate this performance problem in a VPN implemented with the IP Security Protocol (IPsec).

We use a system model for IPsec transactions to derive an inequality that specifies the conditions required for data compression to improve performance. We generate performance results for many combinations of network types, data types, packet sizes, and encryption, authentication and compression algorithms. We find that compression usually improves performance when using 10 Mbps or slower networks, but compression only improves performance in systems with 100 Mbps or 1 Gbps networks when using computationally intensive encryption algorithms.

1. Introduction

As Internet usage grows exponentially, network security issues become increasingly important. Using a virtual private network (VPN), multiple hosts can communicate securely over a public network. The details of VPN protocols vary, but most consist of two major steps: the handshake and bulk data encryption and authentication. The VPN is established during the handshake step. This step involves protocol and algorithm negotiation, authentication of hosts, and secret key exchanges between the hosts. The hosts can then communicate privately by encrypting and authenticating all of the data that travels over the public network.

The IP Security Protocol (IPsec) can be used to implement encryption and message authentication in virtual private networks in a vendor-independent, application-invisible manner [11]. The encryption and message authentication services provided by IPsec, however, require significant computation time. Consequently, IPsec can degrade performance when compared to unsecured transmissions.

In other work, researchers have improved the performance of secure network transactions using a variety of techniques. By adding new instructions to conventional instruction set architectures, the number of instructions in software implementations of cryptographic algorithms can be significantly reduced [22]. In addition, the computation associated with many cryptographic protocols is highly parallelizable. When performing encryption or message authentication, a multiprocessor system can achieve nearly linear speedup by assigning individual packets or connections to single processing elements [15], [16].

In this paper, we investigate the performance benefit of compressing IP packet payloads when using IPsec to implement a virtual private network. We restrict our investigation to the performance impact of compression on bulk data encryption and message authentication. Data compression does not improve the performance of the algorithms and protocols involved in the handshake step.

The IP Payload Compression Protocol (IPComp) employs data compression algorithms to reduce the size of packet payloads at the IP layer [21]. This size reduction decreases the time required to transmit IP packets by reducing the amount of information that is physically transferred over the network. In addition, IPComp can decrease the execution times of the encryption and authentication algorithms by reducing the amount of information to be encrypted and authenticated. IPComp does not always improve performance, however. The data compression algorithms used by IPComp consume a significant number of clock cycles. As a result, compressing the IP packet payload may increase the total time needed to complete a particular transaction, and therefore IPComp can degrade performance.

Measuring the actual performance of VPN transactions is complicated and time-consuming and does not reveal

0-7695-0912-6/00 $10.00 © 2000 IEEE

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

how individual components of the transactions affect performance. To show the dependence of performance on different algorithm and system parameters, we model a virtual private network between two parties. This model allows us to easily change parameters such as the encryption algorithm and network speed and observe the effects. Using speedup equations that describe the performance impact of compression, we derive an inequality that specifies the conditions required for compression to improve performance in the model. This inequality depends on several parameters such as network bandwidth, packet payload sizes, algorithm throughputs, and compression ratio. By measuring the values of these parameters, we can predict whether compression will improve performance in a given system. We obtain empirical performance results by executing the security and compression algorithms on a single machine that contains one 367 MHz HP PA-8500 processor.

The paper is organized as follows. In Section 2, we discuss our system model. Section 3 discusses the details of IPsec, and Section 4 explores the performance cost of using IPsec. In Section 5, we describe IPComp and investigate the compressibility and throughput achieved by different compression algorithms. In Section 6, we present speedup equations for calculating performance and derive inequalities that predict when compression will improve performance. In Section 7, we present and analyze experimental performance results of combining IPComp and IPsec. We summarize and discuss directions for future work in Section 8.

2. System Model

When combining IPComp with IPsec, the performance depends on the characteristics of the system. For example, the time needed for an individual to retrieve banking account information over the Internet consists of three major components. These components include the computational time required by the bank's servers to prepare the IP packets, the time needed to physically transmit the IP packets, and the computational time required by the individual's computer to interpret the packets. The *prepare packet* operation consists of the procedures such as encryption and hash computation required to prepare the packet for secure transmission. The *interpret packet* operation consists of the procedures such as decryption and hash verification required to extract the contents of the packet upon receipt. As we will explain in Section 3, hash computation and verification are used to perform message authentication. We describe encryption and decryption in Section 3, and we describe compression and decompression in Section 5.

The system model includes a sender, a network, and a receiver, as shown in Figure 1. Such a system can consist of multiple processors on both the receiver and sender ends, and the packets may pass through several gateways and network types before reaching their final destination. One approach to modeling such a system would be to treat the computation and transmission segments as a pipeline. In a pipelined model, the performance would depend on the most time-consuming segment. In this paper, however, we measure the performance in terms of total latency. More specifically, we are concerned with the total amount of time required by all computation and network resources to prepare, transmit, and interpret packets. We investigate the extent to which IPComp increases or decreases the total amount of time required to complete these tasks. The performance measure, therefore, is based upon the time needed to serially compress, encrypt, hash compute, transmit, hash verify, decrypt, and decompress.

3. IP Security Protocol

The IP Security Protocol (IPsec) provides a variety of security services at the IP layer [11]. IPsec confidentiality and message authentication services are implemented using the Encapsulating Security Payload and the Authentication Header. The Encapsulating Security Payload (ESP) provides for confidentiality of the IP packet payload using symmetric key encryption algorithms [10]. Both the Authentication Header (AH) and ESP insure the authenticity as well as the integrity of the IP packet payload using symmetric key encryption algorithms or secret-keyed one-way hash functions [9], [10]. In addition, AH provides protection against IP address spoofing. IPsec allows ESP and AH to be applied to IP packets either alone or in combination with each other [11].

Requests for Comments (RFCs) describing the IPsec protocols include instructions for integration in both IPv4 and IPv6. Only minor differences exist between the IPsec integration procedures for the two IP versions, and we discuss the algorithms, protocols, and procedures as specified for IPv6 in this paper.

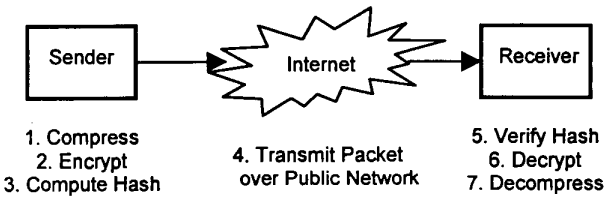


Figure 1. System model

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

3.1. IP Encapsulating Security Payload

ESP is used to provide confidentiality, data origin authentication, connectionless integrity, and anti-replay service (a form of partial sequence integrity) [10]. ESP employs symmetric-key encryption algorithms such as RC5 and 3DES to provide confidentiality. ESP uses keyed hash algorithms such as SHA-1 and MD5 to provide message authentication and anti-replay service.

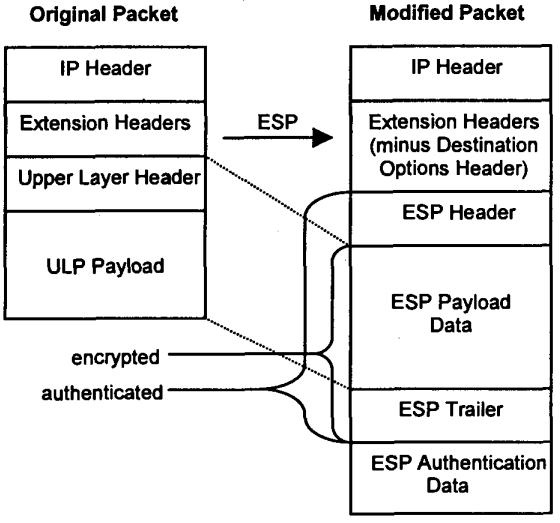


Figure 2. ESP in IPv6

ESP transforms IPv6 packets as illustrated in Figure 2 [10]. The original IPv6 packet depicted in Figure 2 consists of 4 components [4]. The IP header is the essential first component of every IPv6 packet. Following the IP header, the packet contains a variable number of Extension headers. These headers include (but are not limited to) the Routing header, the Fragmentation header, and the ESP header. The Upper Layer header immediately follows the Extension headers and is used by the protocol immediately above IPv6 (e.g., TCP, UDP, and ICMP). Last, the ULP (Upper Layer Protocol) Payload consists of the data to be sent and received by the ULP. The ESP header is inserted directly preceding the Destination Options header, if it exists. If a Destination Options header does not exist, the ESP header is inserted immediately preceding the ULP header and payload.

The first field in the ESP header consists of the Security Parameters Index (SPI), which, in conjunction with the IP destination and the security protocol, specifies the Security Association for the packet. The Security Association (SA) indicates which algorithms and modes of operation are being used to perform the encryption and authentication (see Section 3.3). The second field in the ESP header consists of the Sequence Number, which is a monotonically increasing counter value that prevents replay attacks (if authentication is employed).

The encrypted data block directly follows the ESP header. This block consists of at most three components: the initialization vector (IV), the ESP payload data, and the ESP trailer. The ESP Authentication Data field follows the ESP Trailer. This field contains a variable-length Integrity Check Value (ICV) that is computed over the ESP header, the ESP payload data and the ESP trailer. The ICV is calculated using either a secret-keyed hash algorithm or a symmetric key encryption algorithm.

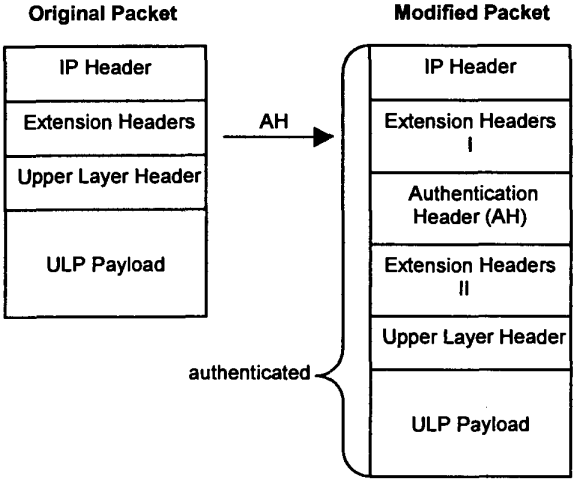


Figure 3. AH in IPv6

3.2. IP Authentication Header

The IP Authentication Header (AH) provides authentication, protection against replay attacks, and connectionless integrity for IP packets [9]. AH transforms IPv6 packets as illustrated in Figure 3 [9]. The original packet is constructed exactly as described in Section 3.1. As shown in Figure 3, the AH should be inserted immediately following Extension Headers I and immediately preceding Extension Headers II. Extension Headers I includes the Hop-by-Hop Options header, the Routing header, and the Fragment header. Extension Headers II includes the ESP header and the Destination Options header. In IPv6, the Authentication Header is also classified as an Extension header. IP packets are not required to contain any Extension headers [4].

The Sequence Number in the AH consists of a monotonically increasing counter value that is used to prevent replay attacks. Furthermore, the Authentication Data field contains the Integrity Check Value (ICV) for the packet. The ICV is the output of the chosen

Lexmark, Exhibit 1002
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.
Lexmark v. Flexiworld

authentication algorithm computed over the entire IP packet.

3.3. Encryption Algorithms

Several symmetric key encryption algorithms are defined for use in ESP [18]. In this study, we use two ciphers in CBC mode: RC5-CBC and 3DES-EDE-CBC. CBC, which stands for Cipher Block Chaining, is a mode of operation that provides more security than the ECB (Electronic Code Book) mode of operation, which is easier to implement [20].

DES, which is an acronym for Data Encryption Standard, is a symmetric-key block cipher that uses a 56-bit key to encrypt 64-bit blocks [20]. Triple DES (referred to as 3DES) provides more security than DES by encrypting a single block three times (with DES) using two or three different 56-bit keys [20]. We implement 3DES in EDE (encrypt-decrypt-encrypt) mode, as prescribed in [18]. To encrypt a block of data, we encrypt with the first key, then decrypt with the second key, and then encrypt with the third key. To decrypt a block of data, we decrypt with the third key, encrypt with the second key, and then decrypt with the first key. We apply CBC mode using the outer chaining technique as described in [20]. 3DES is cryptographically twice as strong as DES, but 3DES takes nearly three times as much computational time as DES to encrypt or decrypt blocks.

RC5 is a symmetric key block cipher that is patented by RSA Data Security, Inc. [20]. RC5 can operate over different block sizes and key lengths; in this study we use a block size of 64 bits and a key length of 128 bits. We implement RC5 in CBC mode as described in [2]. RC5 is considered to be one of the fastest secure symmetric-key block ciphers, whereas 3DES is one of the slowest.

3.4. Authentication Algorithms

Two authentication algorithms that are defined for use with AH and ESP are HMAC-MD5 and HMAC-SHA-1 [9], [10]. In this study, we shall evaluate the performance of both of these algorithms. HMAC is a mechanism that provides message authentication in which an iterative cryptographic hash function is used with a secret key [12].

MD5 is a cryptographic hash function that accepts plaintext blocks of size 64 bytes and outputs a 16-byte authentication value [13]. SHA-1, a government standard, is a cryptographic hash function that accepts 64-byte plaintext blocks and outputs a 20-byte authentication value [14]. SHA-1 is considered to be a cryptographically superior hash function, but MD5 is faster [12].

3.5. Implementing ESP and AH

When combining ESP and AH, the AH header directly precedes the ESP header in the modified IP packet. Since AH authenticates more data in an IP packet than the authentication services in ESP, we always use AH to perform authentication in this study; we only use ESP to perform encryption. In addition, hash computation and hash verification are equivalent operations: the sender and the receiver execute the same hash algorithm over identical keys and identical packets.

It is possible for ESP and AH to specify NULL for the encryption or authentication algorithms. When using the NULL algorithm, we treat the packet as if the protocol were not being employed at all. This policy reduces computation time by eliminating useless protocol processing.

4. Performance Impact of IPsec

In this section, we explore the performance costs of employing IPsec procedures. We obtain data concerning the throughput of IPsec procedures by implementing and executing the cryptographic algorithms on a HP Visualize C360 workstation. This workstation consists of a 64-bit 367 MHz HP PA-8500 processor with 1.5 MB of on-chip L1 cache and 128 MB of RAM. We implemented all four of the authentication and encryption algorithms in C. We use the HPUX C compiler (`cc`) to build the modules, and we employ full compiler and linker optimizations (i.e., `+O4`). We obtain the timing results for all of the authentication and encryption procedures using the UNIX `clock()` function. In order to avoid imprecision resulting from the relatively high granularity of the `clock()` output, we execute the IPsec procedures thousands of times. We obtain the final timing result by dividing the total time needed to complete the thousands of iterations by the total number of iterations executed.

4.1. Network Types and Payload Sizes

We consider the following network types: 56 kbps (phone line modem), 1.54 Mbps (T1, wireless), 10 Mbps (Ethernet), 100 Mbps (Ethernet), and 1 Gbps (Ethernet). Under ideal conditions, Ethernet networks can achieve a channel efficiency of over 90% [23]. As the number of network users increases, however, the channel efficiency drops to roughly 82% [23]. In this study, we assume all the network connections sustain 80% of their maximum throughput.

We evaluate the performance of the IPsec procedures using 3 ULP payload sizes: 1 kilobyte, 4 kilobytes, and 63 kilobytes. TCP, which stands for Transmission Control

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

Protocol, is a connection-based protocol that employs a 20-byte payload header [19]. We assume that all ULP payloads are TCP payloads. The maximum size of an IPv6 packet minus the IP header is 64 KB, so the 63 KB payload size represents the largest possible payload size but leaves room for ESP, AH, IPComp, and TCP headers. We set the network maximum transmission unit (MTU) to be 1280 bytes, the minimum MTU allowed in IPv6, and we fragment packets accordingly [4]. We assume that all the original IP packets solely consist of a 40-byte IPv6 header, an 8-byte fragmentation header (if necessary), a 20-byte TCP header, and a variable length TCP payload. The use of additional headers or a transport protocol other than TCP should not significantly affect the performance results.

Table 1. Performance of encryption and authentication algorithms (Mbps)

Algorithm Name	Payload Sizes			
	1K	4K	63K	Average
MD5	229	299	332	287
SHA-1	214	274	302	263
RC5	96	97	96	96
3DES	28	28	28	28

4.2. Performance Results and Analysis

The execution time of the encryption and authentication algorithms is a function of the input size and is independent of the statistical characteristics of the input data. Hence, for a given processing platform, the throughputs of the encryption and authentication algorithms are constant. Since hash computation and hash verification are equivalent operations, they will consume the same amount of time. Furthermore, DES is a Feistel cipher, so the encryption speed of 3DES is equivalent to the decryption speed of 3DES. Although RC5 is not a Feistel cipher, the speed of the encryption procedure is roughly the same as the speed of the decryption procedure.

We summarize the throughput of the encryption and authentication algorithms for the three packet sizes in Table 1. The algorithms are executed in the modes of operation described in Section 3. From Table 1, we see that the throughput of the authentication algorithms is much higher than that of the encryption algorithms. Furthermore, MD5 runs slightly faster than SHA-1 and RC5 runs three times as fast as 3DES.

We now calculate the performance degradation caused by different combinations of these algorithms. As described in Section 2, the system model consists of the computation required to serially compress, encrypt, compute the hash, verify the hash, decrypt, and decompress the IP packet. We quantitatively determine the level of performance degradation by calculating the speedup S as follows.

$$S = \frac{T_{NET}}{T_E + T_{HC} + T_{SECNET} + T_{HV} + T_D}$$

In this equation, T_{NET}, T_{SECNET}, T_E, T_{HC}, T_{HV}, and T_D represent the original packet transmission time, the encrypted/authenticated packet transmission time, the encryption time, the hash computation time, the hash verification time, and the decryption time, respectively. T_{NET} may not equal T_{SECNET} because of the headers that are added to the encrypted/authenticated packet. Values of S close or equal to 1.00 indicate minor performance degradation, whereas values closer to 0.0 indicate enormous performance degradation. We calculate the speedup results using a TCP payload size of 4 KB in our network model (see Section 4). The speedup results for the five network bandwidths and eight combinations of security algorithms are listed in Table 2.

From Table 2, we see that encryption and authentication never degrade performance more than 10% when using 56 kbps or 1.54 Mbps network connections. As bandwidth increases, however, the performance impact of authentication and encryption becomes more pronounced. Both encryption and authentication decimate performance when using a 1 Gbps link. The results also show that the encryption algorithms have a greater effect on the performance than the authentication algorithms, for the encryption routines require much more computation.

Table 2. Speedups (Slowdowns) resulting from authentication and encryption

Network Type	Algorithm Combination							
	MD5	SHA-1	RC5	RC5 MD5	RC5 SHA-1	3DES	3DES MD5	3DES SHA-1
56 kbps	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
1.54 Mbps	0.99	0.98	0.97	0.96	0.96	0.92	0.90	0.90
10 Mbps	0.94	0.94	0.86	0.82	0.82	0.64	0.62	0.62
100 Mbps	0.65	0.63	0.39	0.32	0.32	0.15	0.14	0.14
1 Gbps	0.16	0.15	0.06	0.04	0.04	0.02	0.02	0.02

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

This performance differential between encryption and authentication becomes more pronounced as bandwidth increases. We now investigate the degree to which data compression can alleviate this performance problem.

5. IP Payload Compression

The IP Payload Compression protocol (IPComp) employs data compression algorithms to reduce the size of IP packets [21]. This size reduction decreases the time required to transmit IP packets by decreasing the amount of information that is physically transferred over the network. Furthermore, IPComp decreases the execution time of encryption and authentication algorithms by reducing the amount of information to be encrypted and authenticated. In this section, we describe IPComp and analyze the compressibility and throughputs achieved by the LZS and DEFLATE compression algorithms.

5.1. Protocol Description

IPComp reduces the size of IP packets in IPv4 and in IPv6 using data compression algorithms [21]. The protocol specification includes several restrictions and guidelines concerning these compression algorithms. To preserve the consistency of the packet payload, the compression algorithm must be lossless. Furthermore, the compression of a packet payload must be completed before any IP security processing is performed: a cryptographically secure encryption algorithm outputs ciphertext that cannot be compressed. It follows that the decompression and reassembly of IP packets must also occur after decryption. In addition, each IP packet must be compressed and decompressed independently of other packets, since IP packets may arrive out of order or may never arrive at all. Lastly, the total size in bytes of the compressed payload and the IPComp header must be smaller than the size of the original payload. Otherwise, the original payload is sent without any IPComp header. This non-expansion policy saves clock cycles at the receiver end and guarantees that network traffic will not increase when using IPComp.

IPComp transforms an IPv6 packet as shown in Figure 4 [21]. The 4-byte IPComp header is inserted immediately preceding the Destination Options header in the IP packet. If the Destination Options header does not exist, the IPComp header is inserted immediately preceding the upper-layer header (e.g., the TCP header) in the IP packet. The compression algorithm used by IPComp compresses all the data that follows the IPComp header in the IP packet. The Destination Options header, the upper layer header, and the ULP payload are all compressed.

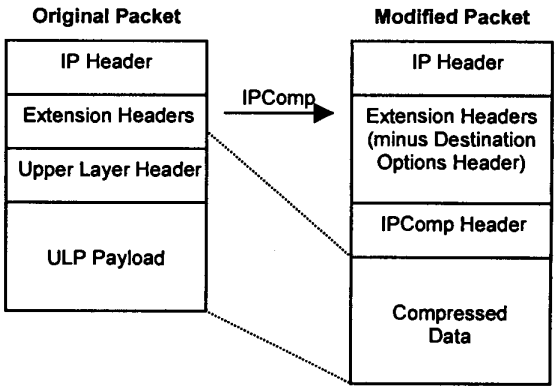


Figure 4. IPComp in IPv6

5.2. Data Compression Algorithms

We evaluate the performance of IPComp using two lossless compression algorithms: DEFLATE and LZS. DEFLATE compresses data using a combination of the LZ77 algorithm and Huffman coding [17]. In this study, we use zlib version 1.1.3, a freely distributed implementation of DEFLATE, with default compression settings [5], [6].

LZS is a lossless compression algorithm based on LZ77 that employs a sliding window of maximum size 2 kilobytes [1], [7]. LZS is an ANSI standard and is patented by Hi/fn, Inc. Our C implementation of LZS stores the compression dictionary as a quickly accessible and infrequently updated order-3 hash table. We also optimized our implementation by employing 64-bit features and by carefully reducing the compression capabilities of the algorithm. By not using the full compression power of LZS, we can obtain huge gains in throughput but pay only a small penalty in compression ratio. Our implementation requires approximately 50 KB of RAM and outperforms DEFLATE in throughput by an order of magnitude.

Unlike encryption and authentication algorithms, the performance of the compression algorithms depends on the statistical properties of the input data as well as its size. In other words, the throughput of the compression algorithms depends on the compressibility of the input data. Hence, we analyze the performance of the compression algorithms using an eclectic group of data benchmarks.

5.3. Data Benchmarks

In a real-world networking environment, a user may transmit a rich variety of data types and sizes. For example, a user may send or receive a compressed 100 KB text file, a 1.3 MB binary executable, or a 12 KB

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

bitmap almost entirely composed of white pixels. We chose 8 data benchmarks to obtain the performance results; their names and descriptions are listed in Table 3.

The first six are members of the Calgary corpus [3]. Researchers use the Calgary corpus to evaluate the practical performance of text compression algorithms. The remaining two benchmarks, gif and random, represent a GIF compressed image file and a randomly generated binary data file, respectively. We expect, therefore, that the gif and random benchmarks will be relatively incompressible. The 6 Calgary corpus benchmarks will exhibit different levels of compressibility, and therefore the performance will vary when using IPComp in conjunction with ESP and AH.

Table 3: Benchmark descriptions

Name	Description
obj2	Compiled code for Apple Macintosh: Knowledge support system
progl	Lisp source code for system software
paper2	A technical paper entitled "Computer (in)security" by Witten
trans	Transcript of a session on a terminal
book1	A book entitled *Far from the Madding Crowd* by Hardy
pic	Picture number 5 from the CCITT Facsimile test files (text + drawings)
gif	Compressed GIF file that contains a campus map
random	Randomly generated data

Table 4. Compression ratios

	DEFLATE			LZS		
	1K	4K	63K	1K	4K	63K
obj2	1.76	2.08	2.36	1.52	1.67	1.77
progl	2.31	3.16	4.23	1.76	2.16	2.33
paper2	1.74	2.08	2.70	1.28	1.50	1.61
trans	1.93	2.55	5.02	1.52	1.82	2.01
book1	1.68	1.94	2.34	1.22	1.40	1.46
pic	26.1	67.5	21.4	10.0	12.7	8.01
gif	0.99	1.00	1.00	0.89	0.89	0.89
random	0.99	1.00	1.00	0.89	0.89	0.89

5.4. Compressibility Results

We compute the compressibility results for both DEFLATE and LZS using the 8 benchmarks and the 3 payload sizes defined in Section 4.1. In order to avoid any compression anomalies associated with compressing the first few kilobytes of the benchmarks, we compress several different portions of the benchmarks when using the 1-KB and 4-KB payload sizes. For the 1-KB payloads, we compress the first 40 1-KB blocks of the benchmark, and the compressibility results are based upon the average compressibility of those 40 blocks. Similarly, for the 4-KB payload sizes, we compress the first 10 4-KB blocks of the benchmark. When using the 63-KB payload size, however, we only compress the first 63-KB block of the benchmark. In addition, to avoid timing imprecision resulting from the relatively high granularity of the result of the `clock()` function, we repeat the compression and decompression of the blocks thousands of times.

The eight benchmarks exhibit many different levels of compressibility. Table 4 summarizes the compression ratios achieved by DEFLATE and LZS. The size of the compressed data block includes the 4-byte IPComp header. The zlib implementation of DEFLATE achieves a higher compression ratio than LZS for each packet size and benchmark combination. DEFLATE usually outperforms LZS in compression ratio by a factor of 1.0 to 2.0, but the ratio of the compression ratios can exceed 5.0 for highly compressible data.

Table 5. DEFLATE performance (Mbps)

Benchmark Name	Compression			Decompression		
	1K	4K	63K	1K	4K	63K
obj2	14	21	23	66	121	365
progl	18	31	31	77	146	492
paper2	17	25	23	67	123	365
trans	17	29	43	70	143	492
book1	17	23	20	68	112	340
pic	34	72	76	166	299	703
gif	14	28	31	0	0	0
random	14	29	31	0	0	0

Table 6. LZS performance (Mbps)

Benchmark Name	Compression			Decompression		
	1K	4K	63K	1K	4K	63K
obj2	217	210	201	254	252	245
progl	234	245	252	253	263	263
paper2	178	179	182	224	214	211
trans	216	222	232	259	255	259
book1	172	169	169	220	207	203
pic	983	1047	735	498	537	476
gif	186	160	150	0	0	0
random	186	161	151	0	0	0

5.5. Throughput Results

The performance of the compression algorithms depends on both the size and compressibility of the benchmark. Tables 5 and 6 list the compression and decompression rates for DEFLATE and LZS, respectively. The zero values for gif and random in the

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

$$S = \frac{\frac{1}{R_E}N_P + \frac{1}{R_{HC}}N_P + \frac{1}{R_{SECNET}}N_P + \frac{1}{R_{HV}}N_P + \frac{1}{R_D}N_P}{\frac{1}{R_{COMP}}N_P + \frac{1}{R_E}\left(\frac{N_P}{X}\right) + \frac{1}{R_{HC}}\left(\frac{N_P}{X}\right) + \frac{1}{R_{SECNET}}\left(\frac{N_P}{X}\right) + \frac{1}{R_{HV}}\left(\frac{N_P}{X}\right) + \frac{1}{R_D}\left(\frac{N_P}{X}\right) + \frac{1}{R_{DECOMP}}N_P}$$

Figure 5. Speedup due to data compression

decompression columns are a consequence of IPComp's non-expansion policy (described earlier).

Tables 5 and 6 show that LZS always greatly outperforms DEFLATE in compression rate. LZS compresses data nearly as fast as SHA-1 authenticates data, and DEFLATE compresses data almost as slowly as 3DES encrypts data. In addition, LZS significantly outperforms DEFLATE in decompression rate for 1 KB and 4 KB payloads. DEFLATE, however, always achieves a higher decompression rate than LZS for 63 KB payloads. We observe that DEFLATE always achieves a higher compression ratio than LZS, but LZS usually compresses and decompresses data faster than DEFLATE. Which algorithm should we use to maximize performance? We address this issue in Section 7.

6. Calculating the Performance Impact of IPComp on IPsec Transactions

In this section, we present equations that describe the performance impact of combining ESP and AH with IPComp. First, we introduce the equation we use to calculate the speedup. Second, we use this speedup equation to derive an inequality that describes the conditions required for compression to improve performance.

6.1. Speedup Calculation

We calculate the speedup using a single equation. T_E, T_D, T_{HC}, T_{HV}, and T_{SECNET} represent the encryption time, decryption time, hash computation time, hash verification time, and transmission time, respectively. The variables C_E, C_D, C_{HC}, C_{HV}, and C_{SECNET} represent the time needed to encrypt, decrypt, compute the hash, verify the hash, and transmit a *compressed* packet, respectively. If encryption were not being used, for example, then T_E and C_E would both equal 0. Furthermore, the variables T_{COMP} and T_{DECOMP} represent the time required to perform the payload compression and decompression, respectively. We calculate the speedup S for the system model as follows:

$$S = \frac{T_E + T_{HC} + T_{SECNET} + T_{HV} + T_D}{T_{COMP} + C_E + C_{HC} + C_{SECNET} + C_{HV} + C_D + T_{DECOMP}}$$

A speedup greater than or equal to 1 indicates that IPComp reduces the total amount of time required by a secure transaction. Hence, a speedup greater than or equal to 1 means compression improves performance. A speedup less than 1 indicates that compression increases the total time needed; therefore compression degrades performance.

6.2. Predicting the Performance Impact of Data Compression

We now derive an inequality that describes the conditions required for data compression to improve system performance. We rewrite the speedup equation as shown in Figure 5. R_E, R_D, R_{HC}, R_{HV}, R_{COMP}, R_{DECOMP}, and N_P represent the encryption rate, the decryption rate, the hash computation rate, the hash verification rate, the compression rate, the decompression rate, and the size of the original (uncompressed) ULP payload, respectively. R_{SECNET} is the transmission rate of the network (i.e., effective network bandwidth). X is the compression ratio, which equals the size of the uncompressed payload divided by the size of the compressed payload. In order to make the relationships clear, we choose to neglect the overhead resulting from packet headers. For example, the amount of data to be hashed is not N_P as the previous equation indicates. The hash is calculated over the entire packet, so the amount of data to be hashed is at least N_P plus the size of the IP and AH headers. The total size of all the packet headers is usually less than 100 bytes, so if the payload size is relatively large, e.g., multiple kilobytes, the omission of the header overheads will not significantly affect the results.

Compression improves performance if the speedup is greater than or equal to 1. Hence, by setting the right side of the previous equation to be greater than or equal to 1, we obtain a relationship between the compression ratio and the algorithm throughputs such that compression will improve performance. N_P cancels out since we are not considering overhead from packet headers, and we now have the following equation:

$$\left(1 - \frac{1}{X}\right)\left(\frac{1}{R_E} + \frac{1}{R_D} + \frac{1}{R_{HC}} + \frac{1}{R_{HV}} + \frac{1}{R_{SECNET}}\right) \geq \frac{1}{R_{COMP}} + \frac{1}{R_{DECOMP}}$$

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

If the encryption rate, decryption rate, hash computation rate, hash verification rate, and network transmission rate are constant, we obtain the following expression:

$$K\left(1-\frac{1}{X}\right) \geq \frac{1}{R_{COMP}} + \frac{1}{R_{DECOMP}}, \text{ where}$$

$$K = \frac{1}{R_E} + \frac{1}{R_D} + \frac{1}{R_{HC}} + \frac{1}{R_{HV}} + \frac{1}{R_{SECNET}}$$

This inequality describes a relationship between compression rate, decompression rate, and compression ratio such that the resulting speedup will be greater than or equal to 1.

Using this inequality, one can determine whether compression will improve the performance of secure packet transactions without conducting extensive simulations. Secure network transactions in some systems involve other time-consuming steps such as bus transfer time, disk access time and protocol processing time. The speedup equations presented here could be modified to model a particular system more accurately by adding relevant latency terms.

Table 7: System parameters

Parameter	Values
Encryption algorithm	3DES, RC5, NULL
Authentication algorithm	MD5, SHA-1, NULL
Compression algorithm	DEFLATE, LZS, NULL
ULP payload size	1 KB, 4 KB, 63 KB
Data benchmark	obj2, progl, paper2, trans, book1, pic, gif, random
Network bandwidth	56 kbps, 1.54 Mbps, 10 Mbps, 100 Mbps, 1 Gbps

7. Experimental Results and Analysis

In this section, we evaluate the experimental performance results for all the combinations of parameter values listed in Table 7. We execute the security and compression algorithms on the HP Visualize C360 workstation described in Section 4. The simulations produced an enormous amount of performance information due to the large number (3240) of parameter value combinations; we only present the most valuable results here.

We describe results for 4 of the 8 benchmarks and for 2 of the 3 packet sizes. Using the compressibility results discussed in Section 5, we divide the data benchmarks into 4 groups according to level of compressibility. We select 1 benchmark from each of these 4 groups: pic, gif, book1, and trans. The benchmarks pic and gif represent highly compressible and incompressible payloads, respectively, and book1 and trans fall in between. In addition, we only present results for the 1 KB and 63 KB payload sizes. Most of the results for the 4 KB payload size closely resemble those for the 1 KB payload size.

The choice of compression algorithm depends on the network speed. We discover that compression ratio is more important than compression throughput when using slow network links, whereas the compression and decompression rates are more important than the compression ratio for fast network links. We find that DEFLATE usually yields a higher speedup than LZS for 56 kbps and 1.544 Mbps connections, and LZS usually yields a higher speedup than DEFLATE for 10 Mbps, 100 Mbps, and 1 Gbps links. It is important to note that these conclusions concerning compression algorithms and network types are highly implementation dependent. For example, if our LZS implementation were significantly slower, it would be advisable to use DEFLATE rather than LZS for 10 Mbps links. In general, fast compression algorithms should be used with fast network connections, and algorithms that achieve high compression ratios should be used with slow network connections.

Upon inspecting the speedup results, we discover that certain different algorithm combinations produce similar results. For example, consider the case where message authentication is used but encryption is not used. In almost every case, the speedups for MD5 and SHA-1 (when using compression) are both greater than 1 or are both less than 1. In other words, the decision to compress when using AH but not ESP is independent of the authentication algorithm being used. We expected this result, for the throughputs of the MD5 and SHA-1 authentication algorithms differ by only 7.5% when using 63 KB payloads.

We propose 4 algorithm combination classes: combinations that include 3DES (slow encryption), combinations that include RC5 (fast encryption), combinations that include authentication only, and the case where neither authentication nor encryption are employed (referred to as NULL). The results of the NULL case can be used to determine whether compression improves the performance of ordinary communications. We present performance results for one algorithm combination from each class. The algorithm combinations are NULL, SHA-1, RC5/SHA-1, and 3DES/SHA-1; we use SHA-1 since it is computationally more intensive than MD5. For each of these algorithm combinations, we use our earlier recommendations for the compression algorithm that yields the highest performance for a given network type.

Figures 6 through 9 illustrate the speedups obtained for the four algorithm combinations. In all four cases, compression improves performance for network bandwidths less than or equal to 10 Mbps. Figure 6

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

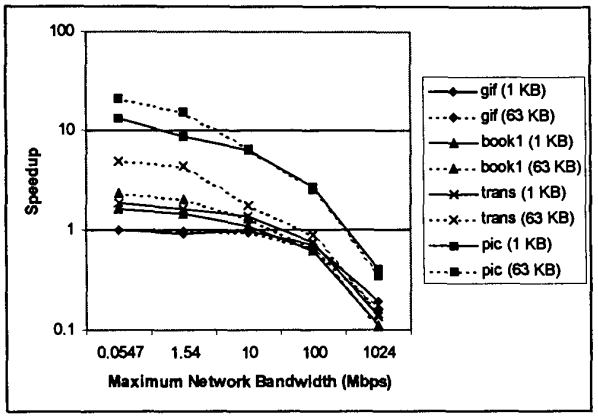


Figure 6. Speedups when using compression only

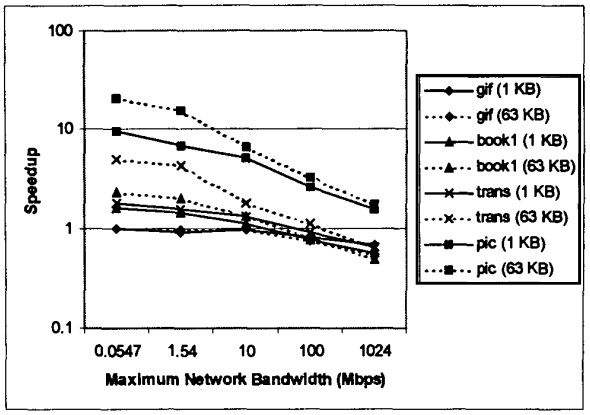


Figure 7. Speedups when using SHA-1

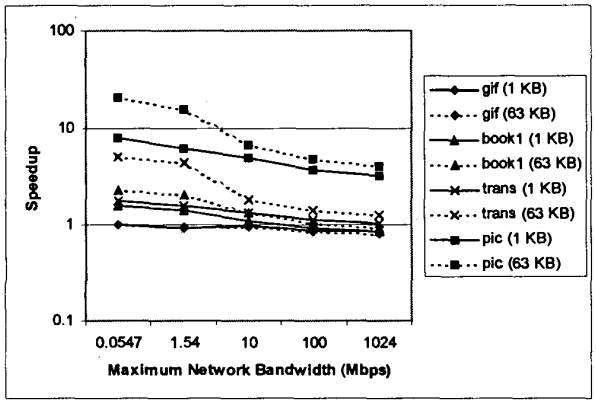


Figure 8. Speedups when using RC5/SHA-1

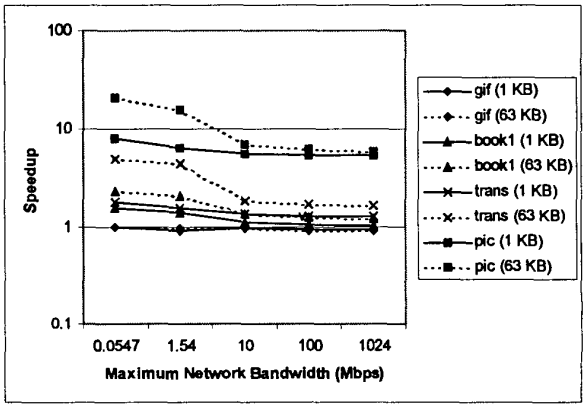


Figure 9. Speedups when using 3DES/SHA-1

depicts the speedups achieved when using compression without encryption or authentication. Compression almost always degrades performance when using 100 Mbps and 1 Gbps networks. Figure 7 illustrates the speedups obtained when using SHA-1. Compression used with message authentication only can improve performance for 100 Mbps and 1 Gbps networks if the packet payloads are expected to be highly compressible (i.e., compressible by a factor of 10). Figures 8 and 9 show the speedup results when using RC5/SHA-1 and 3DES/SHA-1, respectively. For slow encryption (3DES/SHA-1), compression almost always improves performance when using 100 Mbps or 1 Gbps network connections. For fast encryption (RC5/SHA-1), compression improves performance for 100 Mbps and 1 Gbps links if the data is reasonably compressible (i.e., compressible by a factor of 1.5).

If the security and compression algorithms are all executed on workstations similar to the one we used, compression will improve performance for network bandwidths less than or equal to 10 Mbps. In addition, compression improves performance when using network bandwidths as high as 1 Gbps if the payload data is reasonably compressible and encryption is used. We define reasonably compressible data to mean a compression algorithm can reduce the size of packet payload by at least a factor of 1.5.

If processing power increases relative to network speed, the speedups generated by compression will also increase. If processing power decreases relative to network speed, compression will be more likely to cause performance degradation. Similarly, if compression speed and ratio increase relative to encryption and authentication speed, performance speedups due to compression will increase. If compression speed and ratio decrease relative to encryption and authentication speed, compression will be less beneficial in improving system performance.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld
Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.

8. Conclusions

In this paper, we investigated the performance impact of combining data compression with encryption and message authentication to provide for fast, secure network transactions. We defined a system model that allows us to quickly estimate the performance of many combinations of packet, network and algortihm types. First, we demonstrate the extent to which IPsec can degrade performance. Encryption with 3DES can reduce system performance by 36% for 10 Mbps networks and 85% for 100 Mbps networks, i.e., an encrypted transaction can be 1.5 times slower than normal transactions over a 10 Mbps network and 6.6 times slower over a 100 Mbps network.

We derived an inequality that predicts whether data compression will improve performance for a given system. This inequality depends on the throughputs of the encryption, authentication and compression algorithms used, on the bandwidth of the network, and on the compressibility of the packets. We analyzed performance results for 3240 combinations of system parameter values and categorized them into four classes: compression with no authentication or encryption, compression with authentication (and no encryption), compression with fast encryption, and compression with slow encryption. In each class, compression usually improves performance for 10 Mbps or slower networks. For 100 and 1000 Mbps networks, compression only improves performance when encryption (fast or slow) is employed: compression does not improve performance for these faster networks when message authentication is used without encryption.

Future work includes exploring the impact of data compression on secure information processing using more complex models that include communications pipelines, multiple clients and servers, and multiprocessor systems.

9. Acknowledgements

We wish to thank Hewlett Packard Company for funding this research and providing equipment. We also thank Zhijie Shi for writing the encryption and message authentication routines used in this study.

References

[1] American National Standards Institute, Inc., "Data Compression Method – Adaptive Coding with Sliding Window for Information Interchange", ANSI X3.241-1994, August 1994.

[2] Baldwin, R., and R. Rivest, "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms", RFC 2040, October 1996.

[3] Bell, T.C., et al., *Text Compression*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[4] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[5] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[6] Deutsch, P., "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.

[7] Friend, R., and R. Monsour, "IP Payload Compression Using LZS", RFC 2395, December 1998.

[8] Glenn, R., and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998.

[9] Kent, S., and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[10] Kent, S., and R. Atkinson, "IP Encapsulating Security Payload", RFC 2406, November 1998.

[11] Kent, S., and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[12] Krawczyk, H., et al., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[13] Madson, C., and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", RFC 2403, November 1998.

[14] Madson, C., and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.

[15] Nahum, E., et al., "Parallelized Network Security Protocols", *Proceedings of the Internet Society Symposium on Network and Distributed System Security (SNDSS `96)*, February 1996.

[16] Nahum, E., et al., "Towards High Performance Cryptographic Software", *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communications Subsystems (HPCS '95)*, August 1995.

[17] Pereira, R., "IP Payload Compression Using DEFLATE", RFC 2394, December 1998.

[18] Pereira, R., and R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998.

[19] Postel, J., "Transmission Control Protocol", RFC 793, September 1981.

[20] Schneier, B., *Applied Cryptography Second Edition*, John Wiley & Sons, New York, NY, 1996.

[21] Shacham, A., et al., "IP Payload Compression Protocol", RFC 2393, December 1998.

[22] Shi, Z., and R. Lee, "Bit Permutation Instructions for Accelerating Software Cryptography", *Proceedings of the IEEE International Conference on Application-specific Systems, Architectures and Processors*, July 2000.

[23] Tanenbaum, A., *Computer Networks Third Edition*, Prentice-Hall, Upper Saddle River, NJ, 1996.

Lexmark, Exhibit 1002
Lexmark v. Flexiworld

Authorized licensed use limited to: Perkins Coie LLP. Downloaded on April 27,2023 at 23:03:10 UTC from IEEE Xplore. Restrictions apply.