

Video Codec Design
Iain E. G. Richardson
Copyright © 2002 John Wiley & Sons, Ltd
ISBNs: 0-471-48553-5 (Hardback); 0-470-84783-2 (Electronic)

Video Codec Design

To
Freya and Hugh

Video Codec Design

Developing Image and Video Compression Systems

Iain E. G. Richardson

The Robert Gordon University, Aberdeen, UK



JOHN WILEY & SONS, LTD

Copyright © 2002 by John Wiley & Sons Ltd,
Baffins Lane, Chichester,
West Sussex PO19 1UD, England

National 01243 779777
International (+44) 1243 779777

e-mail (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on <http://www.wiley-europe.com>

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London, UK W1P 0LP, without the permission in writing of the publisher.

Neither the authors nor John Wiley & Sons Ltd accept any responsibility or liability for loss or damage occasioned to any person or property through using the material, instructions, methods or ideas contained herein, or acting or refraining from acting as a result of such use. The authors and Publisher expressly disclaim all implied warranties, including merchantability of fitness for any particular purpose.

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons is aware of a claim, the product names appear in initial capital or capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Other Wiley Editorial Offices

John Wiley & Sons, Inc., 605 Third Avenue,
New York, NY 10158-0012, USA

WILEY-VCH Verlag GmbH, Pappelallee 3,
D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 33 Park Road, Milton,
Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01,
Jin Xing Distripark, Singapore 129809

John Wiley & Sons (Canada) Ltd, 22 Worcester Road,
Rexdale, Ontario M9W 1L1, Canada

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0 471 48553 5

Typeset in 10/12 Times by Thomson Press (India) Ltd., New Delhi

Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham, Wiltshire

This book is printed on acid-free paper responsibly manufactured from sustainable forestry, in which at least two trees are planted for each one used for paper production.

Contents

1	Introduction	1
1.1	Image and Video Compression	1
1.2	Video CODEC Design	2
1.3	Structure of this Book	2
2	Digital Video	5
2.1	Introduction	5
2.2	Concepts, Capture and Display	5
2.2.1	The Video Image	5
2.2.2	Digital Video	5
2.2.3	Video Capture	7
2.2.4	Sampling	7
2.2.5	Display	9
2.3	Colour Spaces	10
2.3.1	RGB	11
2.3.2	YCrCb	12
2.4	The Human Visual System	16
2.5	Video Quality	16
2.5.1	Subjective Quality Measurement	17
2.5.2	Objective Quality Measurement	19
2.6	Standards for Representing Digital Video	23
2.7	Applications	24
2.7.1	Platforms	25
2.8	Summary	25
	References	26
3	Image and Video Compression Fundamentals	27
3.1	Introduction	27
3.1.1	Do We Need Compression?	27
3.2	Image and Video Compression	28
3.2.1	DPCM (Differential Pulse Code Modulation)	30
3.2.2	Transform Coding	31
3.2.3	Motion-compensated Prediction	31
3.2.4	Model-based Coding	32
3.3	Image CODEC	33
3.3.1	Transform Coding	33
3.3.2	Quantisation	35

3.3.3	Entropy Coding	37
3.3.4	Decoding.	40
3.4	Video CODEC	41
3.4.1	Frame Differencing.	42
3.4.2	Motion-compensated Prediction	43
3.4.3	Transform, Quantisation and Entropy Encoding	45
3.4.4	Decoding.	45
3.5	Summary.	45
4	Video Coding Standards: JPEG and MPEG	47
4.1	Introduction	47
4.2	The International Standards Bodies.	47
4.2.1	The Expert Groups	48
4.2.2	The Standardisation Process	50
4.2.3	Understanding and Using the Standards	50
4.3	JPEG (Joint Photographic Experts Group).	51
4.3.1	JPEG	51
4.3.2	Motion JPEG.	56
4.3.3	JPEG-2000	56
4.4	MPEG (Moving Picture Experts Group)	58
4.4.1	MPEG-1	58
4.4.2	MPEG-2	64
4.4.3	MPEG-4	67
4.5	Summary.	76
	References.	76
5	Video Coding Standards: H.261, H.263 and H.26L	79
5.1	Introduction	79
5.2	H.261	80
5.3	H.263	80
5.3.1	Features.	81
5.4	The H.263 Optional Modes/H.263+	81
5.4.1	H.263 Profiles	86
5.5	H.26L	87
5.6	Performance of the Video Coding Standards	90
5.7	Summary.	91
	References.	92
6	Motion Estimation and Compensation	93
6.1	Introduction	93
6.2	Motion Estimation and Compensation.	94
6.2.1	Requirements for Motion Estimation and Compensation	94
6.2.2	Block Matching	95
6.2.3	Minimising Difference Energy.	97
6.3	Full Search Motion Estimation.	99
6.4	Fast Search	102
6.4.1	Three-Step Search (TSS).	102

6.4.2	Logarithmic Search	103
6.4.3	Cross Search	104
6.4.4	One-at-a-Time Search	105
6.4.5	Nearest Neighbours Search	105
6.4.6	Hierarchical Search	107
6.5	Comparison of Motion Estimation Algorithms	109
6.6	Sub-Pixel Motion Estimation	111
6.7	Choice of Reference Frames	113
6.7.1	Forward Prediction	113
6.7.2	Backwards Prediction	113
6.7.3	Bidirectional Prediction	113
6.7.4	Multiple Reference Frames	114
6.8	Enhancements to the Motion Model	115
6.8.1	Vectors That can Point Outside the Reference Picture	115
6.8.2	Variable Block Sizes	115
6.8.3	Overlapped Block Motion Compensation (OBMC)	116
6.8.4	Complex Motion Models	116
6.9	Implementation	117
6.9.1	Software Implementations	117
6.9.2	Hardware Implementations	122
6.10	Summary	125
	References	125
7	Transform Coding	127
7.1	Introduction	127
7.2	Discrete Cosine Transform	127
7.3	Discrete Wavelet Transform	133
7.4	Fast Algorithms for the DCT	138
7.4.1	Separable Transforms	138
7.4.2	Flowgraph Algorithms	140
7.4.3	Distributed Algorithms	144
7.4.4	Other DCT Algorithms	145
7.5	Implementing the DCT	146
7.5.1	Software DCT	146
7.5.2	Hardware DCT	148
7.6	Quantisation	150
7.6.1	Types of Quantiser	152
7.6.2	Quantiser Design	153
7.6.3	Quantiser Implementation	156
7.6.4	Vector Quantisation	157
7.7	Summary	160
	References	161
8	Entropy Coding	163
8.1	Introduction	163
8.2	Data Symbols	164
8.2.1	Run-Level Coding	164

8.2.2	Other Symbols	167
8.3	Huffman Coding	169
8.3.1	'True' Huffman Coding	169
8.3.2	Modified Huffman Coding	174
8.3.3	Table Design	174
8.3.4	Entropy Coding Example	177
8.3.5	Variable Length Encoder Design	180
8.3.6	Variable Length Decoder Design	184
8.3.7	Dealing with Errors	186
8.4	Arithmetic Coding	188
8.4.1	Implementation Issues	191
8.5	Summary	192
	References	193
9	Pre- and Post-processing	195
9.1	Introduction	195
9.2	Pre-filtering	195
9.2.1	Camera Noise	196
9.2.2	Camera Movement	198
9.3	Post-filtering	199
9.3.1	Image Distortion	199
9.3.2	De-blocking Filters	206
9.3.3	De-ringing Filters	207
9.3.4	Error Concealment Filters	208
9.4	Summary	208
	References	209
10	Rate, Distortion and Complexity	211
10.1	Introduction	211
10.2	Bit Rate and Distortion	212
10.2.1	The Importance of Rate Control	212
10.2.2	Rate-Distortion Performance	215
10.2.3	The Rate-Distortion Problem	217
10.2.4	Practical Rate Control Methods	220
10.3	Computational Complexity	226
10.3.1	Computational Complexity and Video Quality	226
10.3.2	Variable Complexity Algorithms	228
10.3.3	Complexity-Rate Control	231
10.4	Summary	232
	References	232
11	Transmission of Coded Video	235
11.1	Introduction	235
11.2	Quality of Service Requirements and Constraints	235
11.2.1	QoS Requirements for Coded Video	235
11.2.2	Practical QoS Performance	239
11.2.3	Effect of QoS Constraints on Coded Video	241

11.3	Design for Optimum QoS	244
11.3.1	Bit Rate	244
11.3.2	Error Resilience	244
11.3.3	Delay	247
11.4	Transmission Scenarios	249
11.4.1	Digital Television Broadcasting: MPEG-2 Systems/Transport	249
11.4.2	Packet Video: H.323 Multimedia Conferencing	252
11.5	Summary	254
	References	255
12	Platforms	257
12.1	Introduction	257
12.2	General-purpose Processors	257
12.2.1	Capabilities	258
12.2.2	Multimedia Support	258
12.3	Digital Signal Processors	260
12.4	Embedded Processors	262
12.5	Media Processors	263
12.6	Video Signal Processors	264
12.7	Custom Hardware	266
12.8	Co-processors	267
12.9	Summary	269
	References	270
13	Video CODEC Design	271
13.1	Introduction	271
13.2	Video CODEC Interface	271
13.2.1	Video In/Out	271
13.2.2	Coded Data In/Out	274
13.2.3	Control Parameters	276
13.2.4	Status Parameters	277
13.3	Design of a Software CODEC	278
13.3.1	Design Goals	278
13.3.2	Specification and Partitioning	279
13.3.3	Designing the Functional Blocks	282
13.3.4	Improving Performance	283
13.3.5	Testing	284
13.4	Design of a Hardware CODEC	284
13.4.1	Design Goals	284
13.4.2	Specification and Partitioning	285
13.4.3	Designing the Functional Blocks	286
13.4.4	Testing	286
13.5	Summary	287
	References	287
14	Future Developments	289
14.1	Introduction	289

14.2	Standards Evolution	289
14.3	Video Coding Research	290
14.4	Platform Trends	290
14.5	Application Trends	291
14.6	Video CODEC Design	292
	References	293
	Bibliography	295
	Glossary	297
	Index	301

Index

- Application Programming Interface, 276
- artefacts
 - blocking, 199
 - ringing, 200
- block matching, 43, 95
- blockiness. *See* artefacts:blocking
- B-picture, 59
- chrominance, 12
- CODEC
 - entropy, 37, 45, 163
 - image, 33
 - video, 41
- coded block pattern, 70, 167
- coding
 - arithmetic, 188
 - channel, 29
 - content-based, 70
 - entropy, 163
 - field, 64
 - Huffman, 169
 - lossless, 28
 - lossy, 28
 - mesh, 74
 - model based, 32
 - object-based, 70
 - run-level, 37, 164
 - scalable, 65, 73
 - shape, 70
 - source, 29
 - sprite, 73
 - transform, 31, 127
- colour space, 10
 - RGB, 11
 - YCrCb, 12
- complexity
 - complexity-rate control, 231
 - computational, 226
 - variable complexity algorithms, 228
- compression
 - image and video, 28
- DCT, 31, 127
- basis functions, 130
- distributed, 144
- fast algorithms, 138
- flowgraph, 140
- forward, 127
- hardware, 148
- inverse, 127
- pruned, 228
- software, 146
- design
 - hardware CODEC, 284
 - performance, 283
 - software CODEC, 278
- Digital Cinema, 291
- Digital Versatile Disk, 24
- displaced frame difference, 94
- DPCM, 30
- error concealment, 244
- error resilience, 73, 81, 244
- errors, 244
- filters
 - de-blocking, 82, 206
 - de-ringing, 207
 - error concealment, 208
 - loop, 202
 - pre, 195
 - stabilization, 198
- formats
 - 4CIF, 24
 - CIF, 24
 - ITU-R 601, 23
 - QCIF, 24
- frame rates, 17, 8
- GOB. *See* Group of Blocks
- Group of Blocks, 70
- Group of Pictures, 62
- H.261, 80
- H.263, 80
 - annexes, 81
 - baseline, 81

- H.263 (Continued)
 - PB-frames, 82
 - profiles, 86
- H.26L, 87, 289
- H.323, 252
- high definition television, 67
- Human Visual System, 16
- HVS. *See* Human Visual System
- interface
 - coded data, 274
 - control and status, 276, 277
 - to video CODEC, 271
 - video, 271
- interframe, 41
- International Standards Organisation, 47
- International Telecommunications Union, 47
- intraframe, 41
- I-picture, 59
- ISO. *See* International Standards Organisation
- ITU. *See* International Telecommunications Union
- JPEG, 51
 - baseline CODEC, 51
 - hierarchical, 55
 - lossless, 54
 - Motion, 56
- JPEG2000, 56
- KLT, 31
- latency, 240, 237, 243
- luminance, 12
- memory bandwidth, 274
- MJPEG. *See* JPEG:motion
- motion
 - compensation, 43, 94
 - estimation, 43, 94, 109
 - Cross Search, 104
 - full search, 99
 - hardware, 122
 - hierarchical, 107
 - Logarithmic Search, 103
 - nearest neighbours search, 105
 - OTA, 105
 - performance, 109
 - software, 117
 - sub-pixel, 111
 - Three Step Search, 102
 - vectors, 43, 94, 167
- MPEG, 47
- MPEG-1, 58
 - syntax, 61
- MPEG-2, 64
 - Program Stream, 250
 - systems, 249
 - Transport Stream, 250
 - video, 64
- MPEG-21, 49, 289
- MPEG-4, 67
 - Binary Alpha Blocks, 71
 - profiles and levels, 74
 - Short Header, 68
 - Very Low Bitrate Video core, 68
 - Video Object, 68
 - Video Object Plane, 68
 - video packet, 73
- MPEG-7, 49, 289
- Multipoint Control Unit, 253
- OBMC. *See* motion compensation
- prediction
 - backwards, 113
 - bidirectional, 113
 - forward, 113
- processors
 - co-processor, 267
 - DSP, 260
 - embedded, 262
 - general purpose, 257
 - media, 263
 - PC, 257
 - video signal, 264
- profiles and levels, 66, 74
- quality, 16
 - DSCQS, 17
 - ITU-R 500-10, 17
 - objective, 19
 - PSNR, 19
 - recency, 18
 - subjective, 17
- Quality of Service, 235
- quantisation, 35, 150
 - scale factor, 35
 - vector, 157
- rate, 212
 - control, 212, 220
 - Lagrangian optimization, 218
 - rate-distortion, 217
- Real Time Protocol, 252, 254
- redundancy
 - statistical, 29
 - subjective, 30
- reference picture selection, 84, 247

- re-ordering
 - pictures, 60
 - modified scan, 166
 - zigzag, 166, 37
- RGB. *See* colour space
- ringing. *See* artefacts: ringing
- RVLC. *See* variable length codes
- sampling
 - 4-2-0, 272, 12
 - 4-2-2, 272, 12
 - 4-4-4, 12
 - spatial, 7
 - temporal, 7
- scalability. *See* coding: scalable
- Single Instruction Multiple Data, 258
- slice, 63, 83
- source model, 28
- still image, 5
- sub pixel motion estimation. *See* motion: estimation
- test model, 50
- transform
 - DCT. *See* DCT
 - fractal, 35
 - integer, 145
 - wavelet, 35, 57, 133
- variable length codes
 - reversible, 73, 187
 - table design, 174
 - universal, 175
- variable length decoder, 184
- variable length encoder, 180
- vectors. *See* motion:vectors
- Very Long Instruction Word, 263
- video
 - capture, 7
 - digital, 5
 - interlaced, 9, 64
 - progressive, 9
 - stereoscopic, 7, 65
- Video Coding Experts Group, 48
- Video Quality Experts Group, 19
- VOP. *See* MPEG-4: Video Object Plane
- wavelet transform. *See* transform:wavelet
- YCrCb. *See* colour space

3

Image and Video Compression Fundamentals

3.1 INTRODUCTION

Representing video material in a digital form requires a large number of bits. The volume of data generated by digitising a video signal is too large for most storage and transmission systems (despite the continual increase in storage capacity and transmission 'bandwidth'). This means that compression is essential for most digital video applications.

The ITU-R 601 standard (described in Chapter 2) describes a digital format for video that is roughly equivalent to analogue television, in terms of spatial resolution and frame rate. One channel of ITU-R 601 television, broadcast in uncompressed digital form, requires a transmission bit rate of 216 Mbps. At this bit rate, a 4.7 Gbyte DVD could store just 87 seconds of uncompressed video.

Table 3.1 shows the uncompressed bit rates of several popular video formats. From this table it can be seen that even QCIF at 15 frames per second (i.e. relatively low-quality video, suitable for video telephony) requires 4.6 Mbps for transmission or storage. Table 3.2 lists typical capacities of popular storage media and transmission networks.

There is a clear gap between the high bit rates demanded by uncompressed video and the available capacity of current networks and storage media. The purpose of video compression (video coding) is to fill this gap. A video compression system aims to reduce the amount of data required to store or transmit video whilst maintaining an 'acceptable' level of video quality. Most of the practical systems and standards for video compression are 'lossy', i.e. the volume of data is reduced (compressed) at the expense of a loss of visual quality. The quality loss depends on many factors, but in general, higher compression results in a greater loss of quality.

3.1.1 Do We Need Compression?

The following statement (or something similar) has been made many times over the 20-year history of image and video compression: 'Video compression will become redundant very soon, once transmission and storage capacities have increased to a sufficient level to cope with uncompressed video.' It is true that both storage and transmission capacities continue to increase. However, an efficient and well-designed video compression system gives very significant performance advantages for visual communications at both low and high transmission bandwidths. At low bandwidths, compression enables applications that would not otherwise be possible, such as basic-quality video telephony over a standard telephone

Table 3.1 Uncompressed bit rates

Format	Luminance resolution	Chrominance resolution	Frames per second	Bits per second (uncompressed)
ITU-R 601	858×525	429×525	30	216 Mbps
CIF	352×288	176×144	30	36.5 Mbps
QCIF	176×144	88×72	15	4.6 Mbps

Table 3.2 Typical transmission/storage capacities

Media/network	Capacity
Ethernet LAN (10 Mbps)	Max. 10 Mbps/Typical 1–2 Mbps
ADSL	Typical 1–2 Mbps (downstream)
ISDN-2	128 kbps
V.90 modem	56 kbps downstream/33 kbps upstream
DVD-5	4.7 Gbytes
CD-ROM	640 Mbytes

connection. At high bandwidths, compression can support a much higher visual quality. For example, a 4.7 Gbyte DVD can store approximately 2 hours of uncompressed QCIF video (at 15 frames per second) or 2 hours of compressed ITU-R 601 video (at 30 frames per second). Most users would prefer to see ‘television-quality’ video with smooth motion rather than ‘postage-stamp’ video with jerky motion.

Video compression and video CODECs will therefore remain a vital part of the emerging multimedia industry for the foreseeable future, allowing designers to make the most efficient use of available transmission or storage capacity. In this chapter we introduce the basic components of an image or video compression system. We begin by defining the concept of an image or video encoder (compressor) and decoder (decompressor). We then describe the main functional blocks of an image encoder/decoder (CODEC) and a video CODEC.

3.2 IMAGE AND VIDEO COMPRESSION

Information-carrying signals may be *compressed*, i.e. converted to a representation or form that requires fewer bits than the original (uncompressed) signal. A device or program that compresses a signal is an *encoder* and a device or program that decompresses a signal is a *decoder*. An enCOder/DECOder pair is a *CODEC*.

Figure 3.1 shows a typical example of a CODEC as part of a communication system. The original (uncompressed) information is encoded (compressed): this is *source coding*. The source coded signal is then encoded further to add error protection (*channel coding*) prior to transmission over a *channel*. At the receiver, a *channel decoder* detects and/or corrects transmission errors and a *source decoder* decompresses the signal. The decompressed signal may be identical to the original signal (*lossless compression*) or it may be distorted or degraded in some way (*lossy compression*).

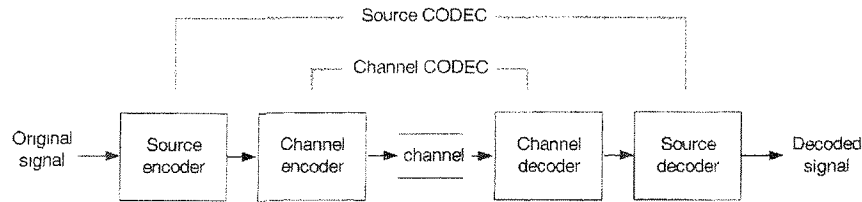


Figure 3.1 Source coder, channel coder, channel

General-purpose compression CODECs are available that are designed to encode and compress data containing *statistical redundancy*. An information-carrying signal usually contains redundancy, which means that it may (in theory) be represented in a more compact way. For example, characters within a text file occur with varying frequencies: in English, the letters E, T and A occur more often than the letters Q, Z and X. This makes it possible to compress a text file by representing frequently occurring characters with short codes and infrequently occurring characters with longer codes (this principle is used in *Huffman coding*, described in Chapter 8). Compression is achieved by reducing the statistical redundancy in the text file. This type of general-purpose CODEC is known as an *entropy CODEC*.

Photographic images and sequences of video frames are not amenable to compression using general-purpose CODECs. Their contents (pixel values) tend to be highly correlated, i.e. neighbouring pixels have similar values, whereas an entropy encoder performs best with data values that have a certain degree of independence (decorrelated data). Figure 3.2 illustrates the poor performance of a general-purpose entropy encoder with image data. The original image (a) is compressed and decompressed using a ZIP program to produce

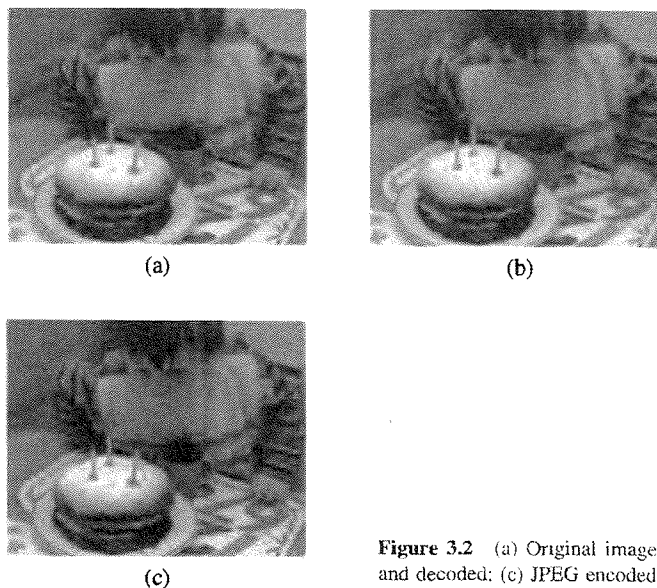


Figure 3.2 (a) Original image; (b) ZIP encoded and decoded; (c) JPEG encoded and decoded

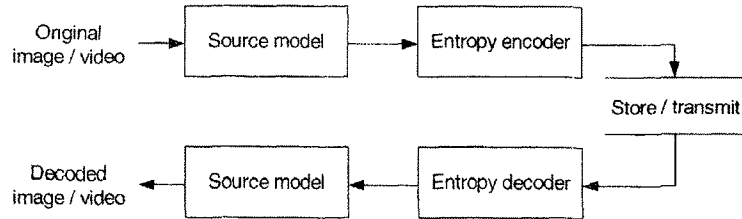


Figure 3.3 Image or video CODEC

image (b). This is identical to the original (lossless compression), but the compressed file is only 92% of the size of the original, i.e. there is very little compression. Image (c) is obtained by compressing and decompressing the original using the JPEG compression method. The compressed version is less than a quarter of the size of the original (over $4 \times$ compression) and the decompressed image looks almost identical to the original. (It is in fact slightly 'degraded' due to the lossy compression process.)

In this example, the JPEG method achieved good compression performance by applying a *source model* to the image before compression. The source model attempts to exploit the properties of video or image data and to represent it in a form that can readily be compressed by an entropy encoder. Figure 3.3 shows the basic design of an image or video CODEC consisting of a source model and an entropy encoder/decoder.

Images and video signals have a number of properties that may be exploited by source models. Neighbouring samples (pixels) within an image or a video frame tend to be highly correlated and so there is significant spatial redundancy. Neighbouring regions within successive video frames also tend to be highly correlated (temporal redundancy). As well as these statistical properties (statistical redundancy), a source model may take advantage of *subjective redundancy*, exploiting the sensitivity of the human visual system to various characteristics of images and video. For example, the HVS is much more sensitive to low frequencies than to high ones and so it is possible to compress an image by eliminating certain high-frequency components. Image (c) in Figure 3.2 was compressed by discarding certain subjectively redundant components of the information: the decoded image is not identical to the original but the information loss is not obvious to the human viewer. Examples of image and video source models include the following:

3.2.1 DPCM (Differential Pulse Code Modulation)

Each sample or pixel is predicted from one or more previously transmitted samples. The simplest prediction is formed from the previous pixel (pixel A in Figure 3.4). A more accurate prediction can be obtained using a weighted average of neighbouring pixels (for example, A, B and C in Figure 3.4). The actual pixel value X is subtracted from the prediction and the difference (the prediction error) is transmitted to the receiver. The prediction error will typically be small due to spatial correlation, and compression can be achieved by representing common, small prediction errors with short binary codes and larger, less common errors with longer codes. Further compression may be achieved by quantising the prediction error and reducing its precision: this is lossy compression as it

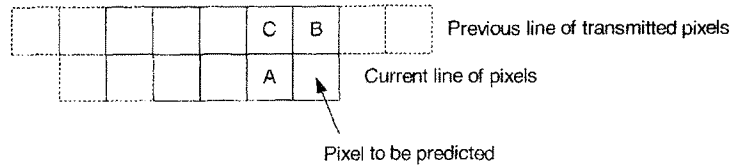


Figure 3.4 DPCM

becomes impossible to exactly reproduce the original values at the decoder. DPCM may be applied spatially (using adjacent pixels in the same frame) and/or temporally (using adjacent pixels in a previous frame to form the prediction) and gives modest compression with low complexity.

3.2.2 Transform Coding

The image samples are transformed into another domain (or representation) and are represented by transform *coefficients*. In the 'spatial domain' (i.e. the original form of the image), samples are highly spatially correlated. The aim of transform coding is to reduce this correlation, ideally leaving a small number of visually significant transform coefficients (important to the appearance of the original image) and a large number of insignificant coefficients (that may be discarded without significantly affecting the visual quality of the image). The transform process itself does not achieve compression: a lossy quantisation process in which the insignificant coefficients are removed, leaving behind a small number of significant coefficients, usually follows it. Transform coding (Figure 3.5) forms the basis of most of the popular image and video compression systems and is described in more detail in this chapter and in Chapter 7.

3.2.3 Motion-compensated Prediction

Using a similar principle to DPCM, the encoder forms a model of the current frame based on the samples of a previously transmitted frame. The encoder attempts to 'compensate' for motion in a video sequence by translating (moving) or warping the samples of the previously transmitted 'reference' frame. The resulting *motion-compensated* predicted frame (the model of the current frame) is subtracted from the current frame to produce a residual 'error' frame (Figure 3.6). Further coding usually follows motion-compensated prediction, e.g. transform coding of the residual frame.

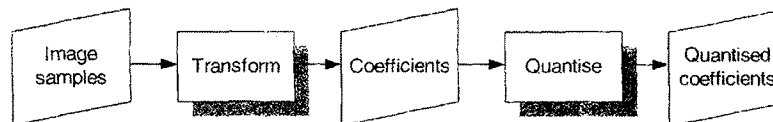


Figure 3.5 Transform coding

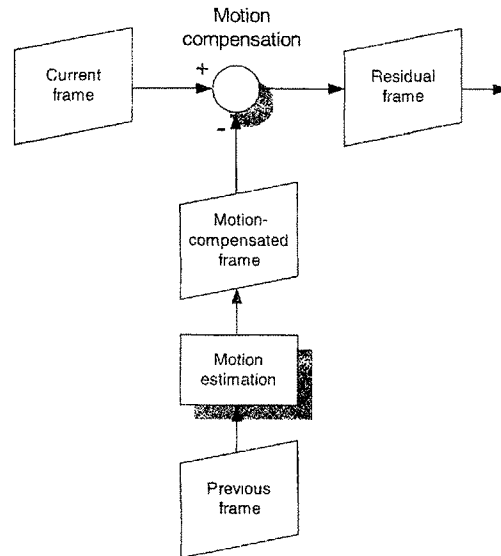


Figure 3.6 Motion-compensated prediction

3.2.4 Model-based Coding

The encoder attempts to create a semantic model of the video scene, for example by analysing and interpreting the content of the scene. An example is a 'talking head' model: the encoder analyses a scene containing a person's head and shoulders (a typical video conferencing scene) and models the head as a 3-D object. The decoder maintains its own 3-D model of the head. Instead of transmitting information that describes the entire image, the encoder sends only the animation parameters required to 'move' the model, together with an error signal that compensates for the difference between the modelled scene and the actual video scene (Figure 3.7). Model-based coding has the potential for far greater

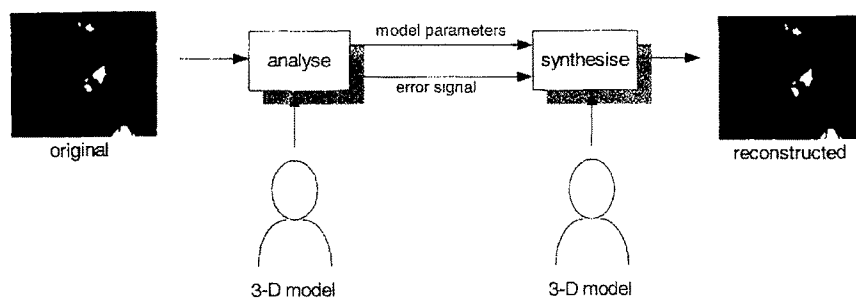


Figure 3.7 Model-based coding

compression than the other source models described above: however, the computational complexity required to analyse and synthesise 3-D models of a video scene in real time is very high.

3.3 IMAGE CODEC

An image CODEC encodes and decodes single images or individual frames from a video sequence (Figure 3.8) and may consist of a *transform coding* stage followed by *quantisation* and *entropy coding*.

3.3.1 Transform Coding

The transform coding stage converts (transforms) the image from the spatial domain into another domain in order to make it more amenable to compression. The transform may be applied to discrete blocks of the image (block transform) or to the entire image.

Block transforms

The spatial image samples are processed in discrete blocks, typically 8×8 or 16×16 samples. Each block is transformed using a 2-D transform to produce a block of transform coefficients. The performance of a block-based transform for image compression depends on how well it can decorrelate the information in each block.

The Karhunen–Loeve transform (KLT) has the ‘best’ performance of any block-based image transform. The coefficients produced by the KLT are decorrelated and the energy in the block is packed into a minimal number of coefficients. The KLT is, however, very computationally inefficient, and it is impractical because the functions required to carry out the transform (‘basis functions’) must be calculated in advance and transmitted to the decoder for every image. The discrete cosine transform (DCT) performs nearly as well as the KLT and is much more computationally efficient. Figure 3.9 shows a 16×16 block of

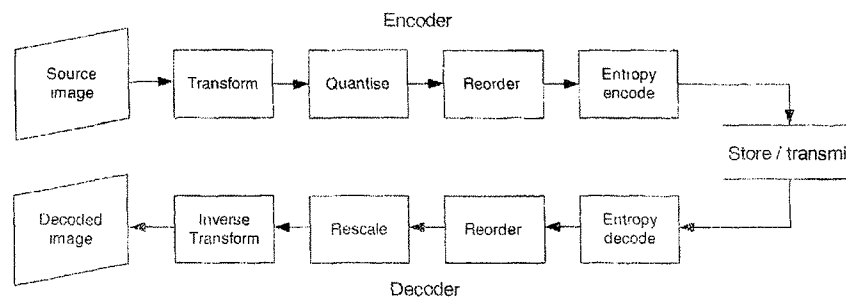


Figure 3.8 Image CODEC

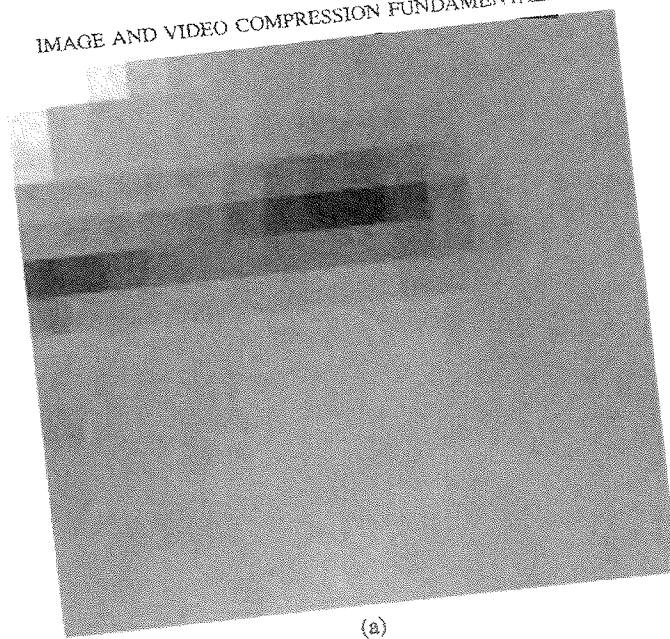
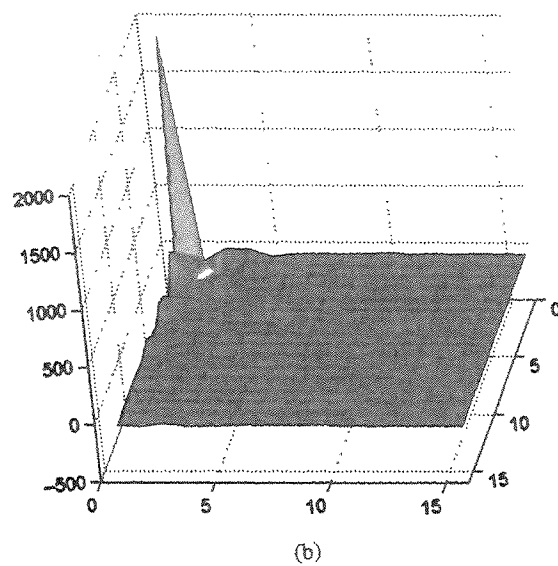


Figure 3.9 (a) 16×16 block of pixels; (b) DCT coefficients

image samples (a) and the corresponding block of coefficients produced by the DCT (b). In the original block, the energy is distributed across the 256 samples and the latter are clearly closely interrelated (correlated). In the coefficient block, the energy is concentrated into a few significant coefficients (at the top left). The coefficients are decorrelated; this means that the smaller-valued coefficients may be discarded (for example by quantisation) without significantly affecting the quality of the reconstructed image block at the decoder.

The 16×16 array of coefficients shown in Figure 3.9 represent *spatial frequencies* in the original block. At the top left of the array are the low-frequency components, representing the gradual changes of brightness (luminance) in the original block. At the bottom right of the array are high-frequency components and these represent rapid changes in brightness. These frequency components are analogous to the components produced by Fourier analysis of a time-varying signal (and in fact the DCT is closely related to the discrete Fourier transform) except that here the components are 2-D. The example shown in Figure 3.9 is typical for a photographic image: most of the coefficients produced by the DCT are insignificant and can be discarded. This makes the DCT a powerful tool for image and video compression.

Image transforms

The DCT is usually applied to small, discrete blocks of an image, for reasons of practicality. In contrast, an *image transform* may be applied to a complete video image (or to a large 'tile' within the image). The most popular transform of this type is the *discrete wavelet transform*. A 2-D wavelet transform is applied to the original image in order to decompose it into a series of filtered 'sub-band' images (Figure 3.10). Image (a) is processed in a series of stages to produce the 'wavelet decomposition' image (b). This is made up of a series of components, each containing a subset of the spatial frequencies in the image. At the top left is a low-pass filtered version of the original and moving to the bottom right, each component contains progressively higher-frequency information that adds the 'detail' of the image. It is clear that the higher-frequency components are relatively 'sparse', i.e. many of the values (or 'coefficients') in these components are zero or insignificant. The wavelet transform is thus an efficient way of decorrelating or concentrating the important information into a few significant coefficients.

The wavelet transform is particularly effective for still image compression and has been adopted as part of the JPEG-2000 standard and for still image 'texture' coding in the MPEG-4 standard. Wavelet-based compression is discussed further in Chapter 7.

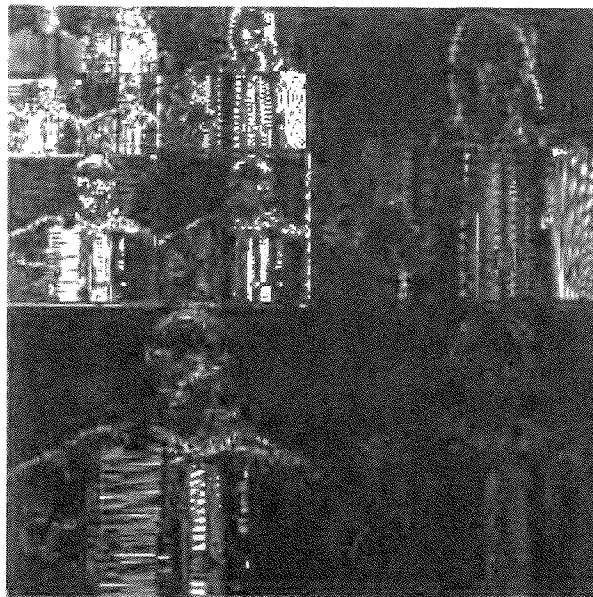
Another image transform that has received much attention is the so-called *fractal transform*. A fractal transform coder attempts to represent an image as a set of scaled and translated arbitrary 'basis patterns'. Fractal-based coding has not, however, shown sufficiently good performance to be included in any of the international standards for video and image coding and so we will not discuss it in detail.

3.3.2 Quantisation

The block and image transforms described above do not themselves achieve any compression. Instead, they represent the image in a different domain in which the image data is



(a)



(b)

Figure 3.10 Wavelet decomposition of image

separated into components of varying 'importance' to the appearance of the image. The purpose of quantisation is to remove the components of the transformed data that are unimportant to the visual appearance of the image and to retain the visually important components. Once removed, the less important components cannot be replaced and so quantisation is a lossy process.

Example

1. The DCT coefficients shown earlier in Figure 3.9 are quantised by dividing each coefficient by an integer. The resulting array of quantised coefficients is shown in Figure 3.11(a); the large-value coefficients map to non-zero integers and the small-value coefficients map to zero.
2. Rescaling the quantised array (multiplying each coefficient by the same integer) gives Figure 3.11(b). The magnitudes of the larger coefficients are similar to the original coefficients; however, the smaller coefficients (set to zero during quantisation) cannot be recreated and remain at zero.
3. Applying an inverse DCT to the rescaled array gives the block of image samples shown in Figure 3.12: this looks superficially similar to the original image block but some of the information has been lost through quantisation.

It is possible to vary the 'coarseness' of the quantisation process (using a quantiser 'scale factor' or 'step size'). 'Coarse' quantisation will tend to discard most of the coefficients, leaving only the most significant, whereas 'fine' quantisation will tend to leave more coefficients in the quantised block. Coarse quantisation usually gives higher compression at the expense of a greater loss in image quality. The quantiser scale factor or step size is often the main parameter used to control image quality and compression in an image or video CODEC. Figure 3.13 shows a small original image (left) and the effect of compression and decompression with fine quantisation (middle) and coarse quantisation (right).

3.3.3 Entropy Coding

A typical image block will contain a few significant non-zero coefficients and a large number of zero coefficients after block transform coding and quantisation. The remaining non-zero data can be efficiently compressed using a statistical compression method ('entropy coding'):

1. *Reorder the quantised coefficients.* The non-zero quantised coefficients of a typical image block tend to be clustered around the 'top-left corner', i.e. around the low frequencies (e.g. Figure 3.9). These non-zero values can be grouped together in sequence by reordering the 64 coefficients, for example in a zigzag scanning order (Figure 3.14). Scanning through in a zigzag sequence from the top-left (lowest frequency) to the bottom-right (highest frequency) coefficients groups together the significant low-frequency coefficients.

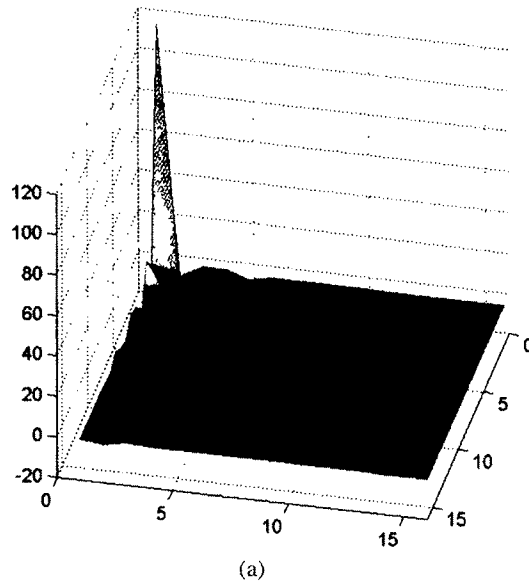
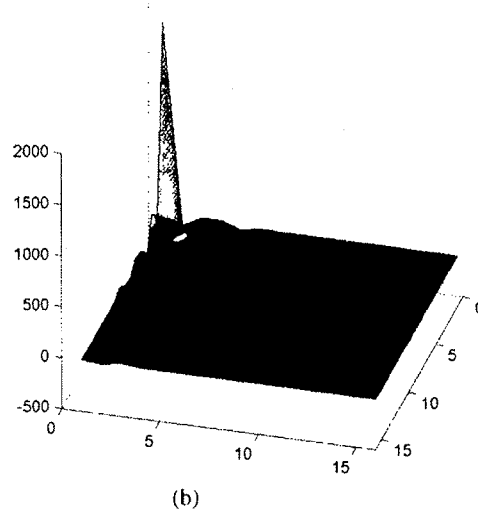


Figure 3.11 (a) Quantised DCT coefficients; (b) rescaled

2. Run-level coding. The reordered coefficient array is usually 'sparse', consisting of a group of non-zero coefficients followed by zeros (with occasional non-zero higher-frequency coefficients). This type of array may be compactly represented as a series of (run, level) pairs, as shown in the example in Table 3.3. The first number in the (run,

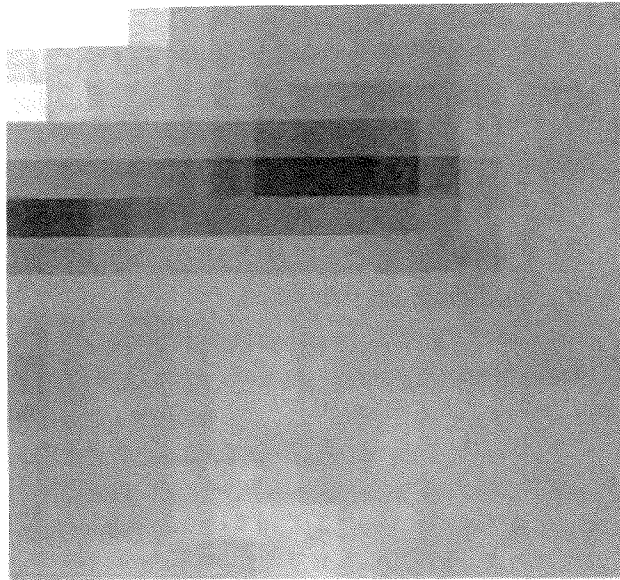


Figure 3.12 Reconstructed block of image samples

level) pair represents the number of preceding zeros and the second number represents a non-zero value (level). For example, (5, 12) represents five zeros followed by 12.

3. *Entropy coding.* A statistical coding algorithm is applied to the (run, level) data. The purpose of the entropy coding algorithm is to represent frequently occurring (run, level) pairs with a short code and infrequently occurring (run, level) pairs with a longer code. In this way, the run-level data may be compressed into a small number of bits.

Huffman coding and arithmetic coding are widely used for entropy coding of image and video data.

Huffman coding replaces each 'symbol' (e.g. a [run, level] pair) with a codeword containing a variable number of bits. The codewords are allocated based on the statistical distribution of

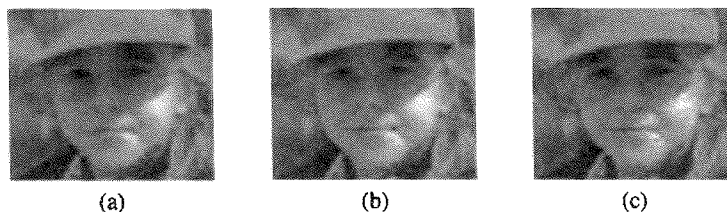


Figure 3.13 (a) Original image; (b) fine quantisation; (c) coarse quantisation

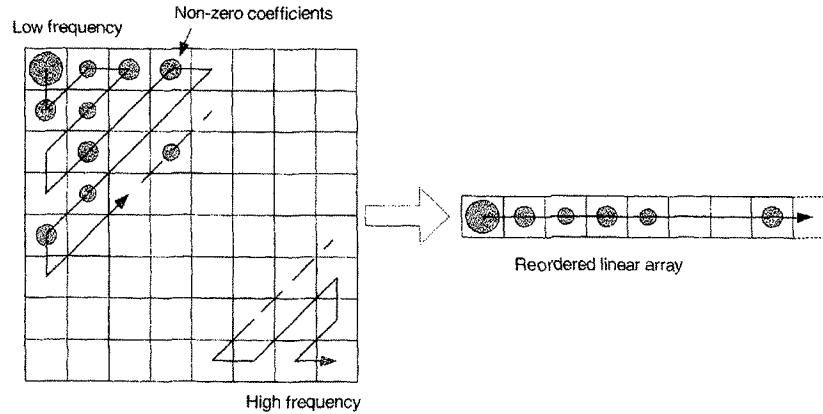


Figure 3.14 Zigzag reordering of quantised coefficients

the symbols. Short codewords are allocated to common symbols and longer codewords are allocated to infrequent symbols. Each codeword is chosen to be 'uniquely decodeable', so that a decoder can extract the series of variable-length codewords without ambiguity. Huffman coding is well suited to practical implementation and is widely used in practice.

Arithmetic coding maps a series of symbols to a fractional number (see Chapter 8) that is then converted into a binary number and transmitted. Arithmetic coding has the potential for higher compression than Huffman coding. Each symbol may be represented with a fractional number of bits (rather than just an integral number of bits) and this means that the bits allocated per symbol may be more accurately matched to the statistical distribution of the coded data.

3.3.4 Decoding

The output of the entropy encoder is a sequence of binary codes representing the original image in compressed form. In order to recreate the image it is necessary to decode this sequence and the decoding process (shown in Figure 3.8) is almost the reverse of the encoding process.

An entropy decoder extracts run-level symbols from the bit sequence. These are converted to a sequence of coefficients that are reordered into a block of quantised coefficients. The decoding operations up to this point are the inverse of the equivalent encoding operations. Each coefficient is multiplied by the integer scale factor ('rescaled'). This is often described

Table 3.3 Run-level coding example

Reordered coefficient data	24, 3, -9, 0, -2, 0, 0, 0, 0, 12, 0, 0, 0, 2, ...
Run-level pairs	(0, 24), (0, 3), (0, -9), (1, -2), (5, 12), (3, 2), ...

as 'inverse quantisation', but in fact the loss of precision due to quantisation cannot be reversed and so the rescaled coefficients are not identical to the original transform coefficients.

The rescaled coefficients are transformed with an inverse transform to reconstruct a decoded image. Because of the data loss during quantisation, this image will not be identical to the original image: the amount of difference depends partly on the 'coarseness' of quantisation.

3.4 VIDEO CODEC

A video signal consists of a sequence of individual frames. Each frame may be compressed individually using an image CODEC as described above: this is described as *intra-frame coding*, where each frame is 'intra' coded without any reference to other frames. However, better compression performance may be achieved by exploiting the temporal redundancy in a video sequence (the similarities between successive video frames). This may be achieved by adding a 'front end' to the image CODEC, with two main functions:

1. *Prediction*: create a prediction of the current frame based on one or more previously transmitted frames.
2. *Compensation*: subtract the prediction from the current frame to produce a 'residual frame'.

The residual frame is then processed using an 'image CODEC'. The key to this approach is the prediction function: if the prediction is accurate, the residual frame will contain little data and will hence be compressed to a very small size by the image CODEC. In order to decode the frame, the decoder must 'reverse' the compensation process, adding the prediction to the decoded residual frame (*reconstruction*) (Figure 3.15). This is *inter-frame coding*: frames are coded based on some relationship with other video frames, i.e. coding exploits the interdependencies of video frames.

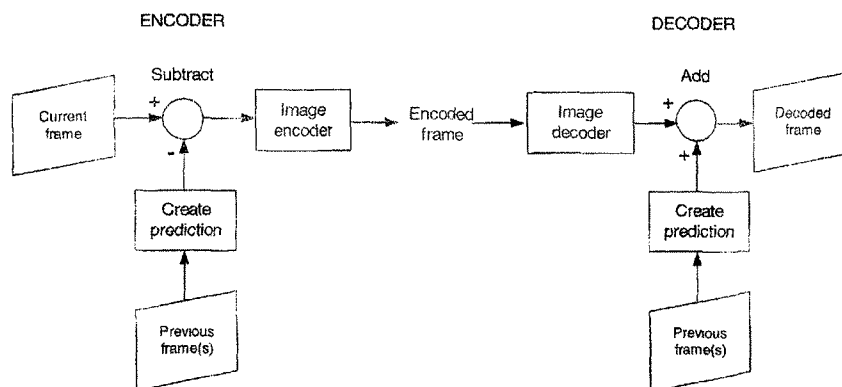


Figure 3.15 Video CODEC with prediction

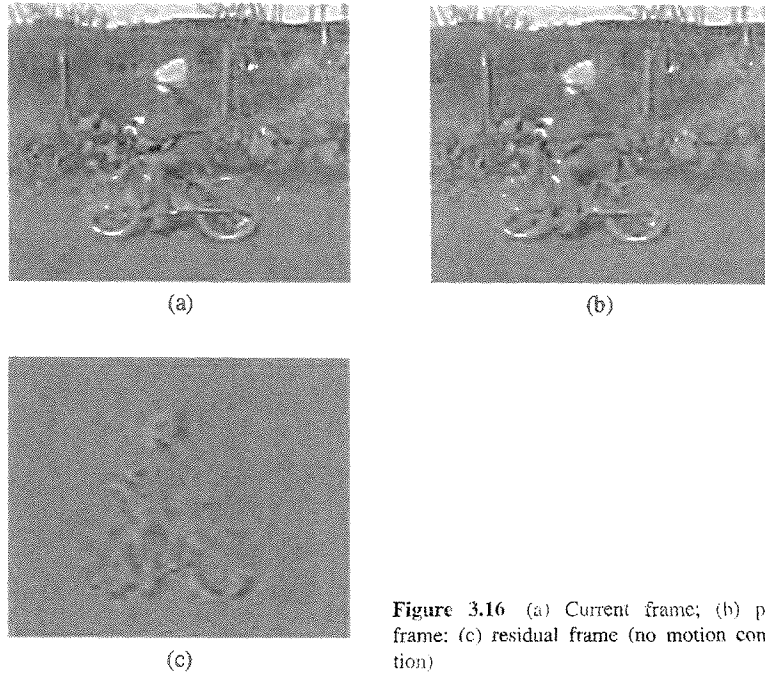


Figure 3.16 (a) Current frame; (b) previous frame; (c) residual frame (no motion compensation)

3.4.1 Frame Differencing

The simplest predictor is just the previous transmitted frame. Figure 3.16 shows the residual frame produced by subtracting the previous frame from the current frame in a video sequence. Mid-grey areas of the residual frame contain zero data: light and dark areas indicate positive and negative residual data respectively. It is clear that much of the residual data is zero: hence, compression efficiency can be improved by compressing the residual frame rather than the current frame.

Table 3.4 Prediction 'drift'

Encoder input	Encoder prediction	Encoder output/ decoder input	Decoder prediction	Decoder output
Original frame 1	Zero	Compressed frame 1	Zero	Decoded frame 1
Original frame 2	Original frame 1	Compressed residual frame 2	Decoded frame 1	Decoded frame 2
Original frame 3	Original frame 2	Compressed residual frame 2	Decoded frame 2	Decoded frame 3
...				...

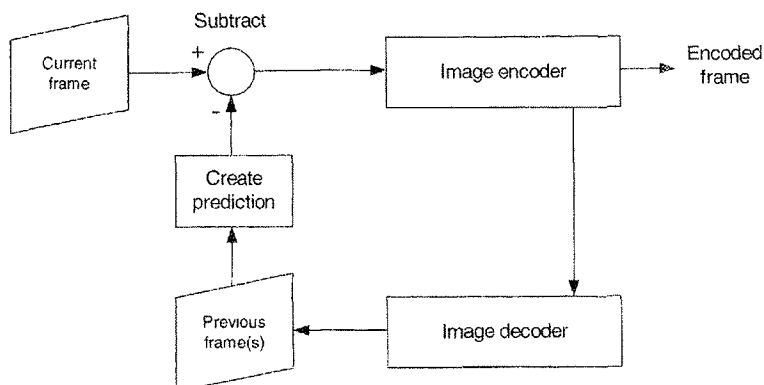


Figure 3.17 Encoder with decoding loop

The decoder faces a potential problem that can be illustrated as follows. Table 3.4 shows the sequence of operations required to encode and decode a series of video frames using frame differencing. For the first frame the encoder and decoder use no prediction. The problem starts with frame 2: the encoder uses the original frame 1 as a prediction and encodes the resulting residual. However, the decoder only has the decoded frame 1 available to form the prediction. Because the coding process is lossy, there is a difference between the decoded and original frame 1 which leads to a small error in the prediction of frame 2 at the decoder. This error will build up with each successive frame and the encoder and decoder predictors will rapidly 'drift' apart, leading to a significant drop in decoded quality.

The solution to this problem is for the encoder to use a decoded frame to form the prediction. Hence the encoder in the above example decodes (or reconstructs) frame 1 to form a prediction for frame 2. The encoder and decoder use the same prediction and drift should be reduced or removed. Figure 3.17 shows the complete encoder which now includes a decoding 'loop' in order to reconstruct its prediction reference. The reconstructed (or 'reference') frame is stored in the encoder and in the decoder to form the prediction for the next coded frame.

3.4.2 Motion-compensated Prediction

Frame differencing gives better compression performance than intra-frame coding when successive frames are very similar, but does not perform well when there is a significant change between the previous and current frames. Such changes are usually due to movement in the video scene and a significantly better prediction can be achieved by *estimating* this movement and *compensating* for it.

Figure 3.18 shows a video CODEC that uses motion-compensated prediction. Two new steps are required in the encoder:

1. **Motion estimation:** a region of the current frame (often a rectangular block of luminance samples) is compared with neighbouring regions of the previous reconstructed frame.

compression does not come without a price: motion estimation can be very computationally intensive. The design of a motion estimation algorithm can have a dramatic effect on the compression performance and computational complexity of a video CODEC.

3.4.3 Transform, Quantisation and Entropy Encoding

A block or image transform is applied to the residual frame and the coefficients are quantised and reordered. Run-level pairs are entropy coded as before (although the statistical distribution and hence the coding tables are generally different for inter-coded data). If motion-compensated prediction is used, motion vector information must be sent in addition to the run-level data. The motion vectors are typically entropy encoded in a similar way to run-level pairs, i.e. commonly occurring motion vectors are coded with shorter codes and uncommon vectors are coded with longer codes.

3.4.4 Decoding

A motion-compensated decoder (Figure 3.18) is usually simpler than the corresponding encoder. The decoder does not need a motion estimation function (since the motion information is transmitted in the coded bit stream) and it contains only a decoding path (compared with the encoding and decoding paths in the encoder).

3.5 SUMMARY

Efficient coding of images and video sequences involves creating a model of the source data that converts it into a form that can be compressed. Most image and video CODECs developed over the last two decades have been based around a common set of 'building blocks'. For motion video compression, the first step is to create a motion-compensated prediction of the frame to be compressed, based on one or more previously transmitted frames. The difference between this model and the actual input frame is then coded using an image CODEC. The data is transformed into another domain (e.g. the DCT or wavelet domain), quantised, reordered and compressed using an entropy encoder. A decoder must reverse these steps to reconstruct the frame: however, quantisation cannot be reversed and so the decoded frame is an imperfect copy of the original.

An encoder and decoder must clearly use a compatible set of algorithms in order to successfully exchange compressed image or video data. Of prime importance is the syntax or structure of the compressed data. In the past 15 years there has been a significant worldwide effort to develop standards for video and image compression. These standards generally describe a syntax (and a decoding process) to support video or image communications for a wide range of applications. Chapters 4 and 5 provide an overview of the main standards bodies and JPEG, MPEG and H.26x video and image coding standards.

5

Video Coding Standards: H.261, H.263 and H.26L

5.1 INTRODUCTION

The ISO MPEG video coding standards are aimed at storage and distribution of video for entertainment and have tried to meet the needs of providers and consumers in the 'media industries'. The ITU has (historically) been more concerned about the telecommunications industry, and its video coding standards (H.261, H.263, H.26L) have consequently been targeted at real-time, point-to-point or multi-point communications.

The first ITU-T video coding standard to have a significant impact, H.261, was developed during the late 1980s/early 1990s with a particular application and transmission channel in mind. The application was video conferencing (two-way communications via a video 'link') and the channel was N-ISDN. ISDN provides a constant bit rate of $p \times 64$ kbps, where p is an integer in the range 1–30: it was felt at the time that ISDN would be the medium of choice for video communications because of its guaranteed bandwidth and low delay. Modem channels over the analogue POTS/PSTN (at speeds of less than 9600 bps at the time) were considered to be too slow for visual communications and packet-based transmission was not considered to be reliable enough.

H.261 was quite successful and continues to be used in many legacy video conferencing applications. Improvements in processor performance, video coding techniques and the emergence of analogue Modems and Internet Protocol (IP) networks as viable channels led to the development of its successor, H.263, in the mid-1990s. By making a number of improvements to H.261, H.263 provided significantly better compression performance as well as greater flexibility. The original H.263 standard (Version 1) had four optional modes which could be switched on to improve performance (at the expense of greater complexity). These modes were considered to be useful and Version 2 ('H.263+') added 12 further optional modes. The latest (and probably the last) version (v3) will contain a total of 19 modes, each offering improved coding performance, error resilience and/or flexibility.

Version 3 of H.263 has become a rather unwieldy standard because of the large number of options and the need to continue to support the basic ('baseline') CODEC functions. The latest initiative of the ITU-T experts group VCEG is the H.26L standard (where 'L' stands for 'long term'). This is a new standard that makes use of some of the best features of H.263 and aims to improve compression performance by around 50% at lower bit rates. Early indications are that H.26L will outperform H.263+ (but possibly not by 50%).

5.2 H.261¹

Typical operating bit rates for H.261 applications are between 64 and 384 kbps. At the time of development, packet-based transmission over the Internet was not expected to be a significant requirement, and the limited video compression performance achievable at the time was not considered to be sufficient to support bit rates below 64 kbps.

A typical H.261 CODEC is very similar to the 'generic' motion-compensated DCT-based CODEC described in Chapter 3. Video data is processed in 4:2:0 Y:Cr:Cb format. The basic unit is the 'macroblock', containing four luminance blocks and two chrominance blocks (each 8×8 samples) (see Figure 4.6). At the input to the encoder, 16×16 macroblocks may be (optionally) motion compensated using integer motion vectors. The motion-compensated residual data is coded with an 8×8 DCT followed by quantisation and zigzag reordering. The reordered transform coefficients are run-level coded and compressed with an entropy encoder (see Chapter 8).

Motion compensation performance is improved by use of an optional *loop filter*, a 2-D spatial filter that operates on each 8×8 block in a macroblock prior to motion compensation (if the filter is switched on). The filter has the effect of 'smoothing' the reference picture which can help to provide a better prediction reference. Chapter 9 discusses loop filters in more detail (see for example Figures 9.11 and 9.12).

In addition, a forward error correcting code is defined in the standard that should be inserted into the transmitted bit stream. In practice, this code is often omitted from practical implementations of H.261: the error rate of an ISDN channel is low enough that error correction is not normally required, and the code specified in the standard is not suitable for other channels (such as a noisy wireless channel or packet-based transmission).

Each macroblock may be coded in 'intra' mode (no motion-compensated prediction) or 'inter' mode (with motion-compensated prediction). Only two frame sizes are supported, CIF (352×288 pixels) and QCIF (176×144 pixels).

H.261 was developed at a time when hardware and software processing performance was limited and therefore has the advantage of low complexity. However, its disadvantages include poor compression performance (with poor video quality at bit rates of under about 100 kbps) and lack of flexibility. It has been superseded by H.263, which has higher compression efficiency and greater flexibility, but is still widely used in installed video conferencing systems.

5.3 H.263²

In developing the H.263 standard, VCEG aimed to improve upon H.261 in a number of areas. By taking advantage of developments in video coding algorithms and improvements in processing performance, it provides better compression. H.263 provides greater flexibility than H.261: for example, a wider range of frame sizes is supported (listed in Table 4.2). The first version of H.263 introduced four optional modes, each described in an annex to the standard, and further optional modes were introduced in Version 2 of the standard ('H.263+'). The target application of H.263 is low-bit-rate, low-delay two-way video communications. H.263 can support video communications at bit rates below 20 kbps (at a very limited visual quality) and is now widely used both in 'established' applications such as video telephony and video conferencing and an increasing number of new applications (such as Internet-based video).

5.3.1 Features

The baseline H.263 CODEC is functionally identical to the MPEG-4 'short header' CODEC described in Section 4.4.3. Input frames in 4:2:0 format are motion compensated (with half-pixel resolution motion vectors), transformed with an 8×8 DCT, quantised, reordered and entropy coded. The main factors that contribute to the improved coding performance over H.261 are the use of half-pixel motion vectors (providing better motion compensation) and redesigned variable-length code (VLC) tables (described further in Chapter 8). Features such as I- and P-pictures, more frame sizes and optional coding modes give the designer greater flexibility to deal with different application requirements and transmission scenarios.

5.4 THE H.263 OPTIONAL MODES/H.263+

The original H.263 standard (Version 1) included four optional coding modes (Annexes D, E, F and G). Version 2 of the standard added 12 further modes (Annexes I to T) and a new release is scheduled with yet more coding modes (Annexes U, V and W). CODECs that implement some of the optional modes are sometimes described as 'H.263--' or 'H.263++' CODECs depending on which modes are implemented.

Each mode adds to or modifies the functionality of H.263, usually at the expense of increased complexity. An H.263-compliant CODEC must support the 'baseline' syntax described above: the use of optional modes may be negotiated between an encoder and a decoder prior to starting a video communications session. The optional modes have a number of potential benefits: some of the modes improve compression performance, others improve error resilience or provide tools that are useful for particular transmission environments such as packet-based transmission.

Annex D, Unrestricted motion vectors The optional mode described in Annex D of H.263 allows motion vectors to point outside the boundaries of the picture. This can provide a coding performance gain, particularly if objects are moving into or out of the picture. The pixels at the edges of the picture are extrapolated to form a 'border' outside the picture that vectors may point to (Figure 5.1). In addition, the motion vector range is extended so that

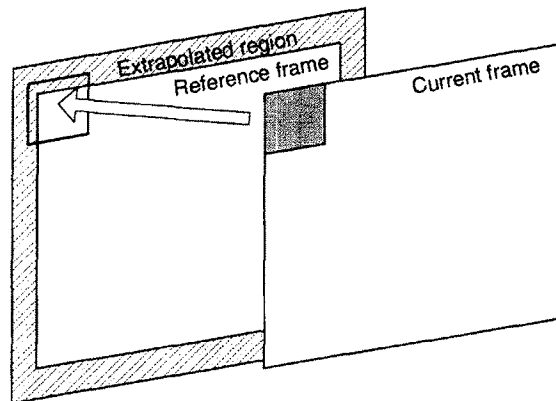


Figure 5.1 Unrestricted motion vectors

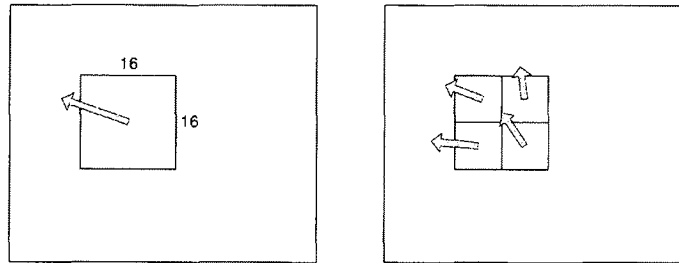


Figure 5.2 One or four motion vectors per macroblock

longer vectors are allowed. Finally, Annex D contains an optional alternative set of VLCs for encoding motion vector data. These VLCs are reversible, making it easier to recover from transmission errors (see Chapter 11).

Annex E, Syntax-based arithmetic coding Arithmetic coding is used instead of variable-length coding. Each of the VLCs defined in the standard is replaced with a probability value that is used by an arithmetic coder (see Chapter 8).

Annex F, Advanced prediction The efficiency of motion estimation and compensation is improved by allowing the use of four vectors per macroblock (a separate motion vector for each 8×8 luminance block, Figure 5.2). Overlapped block motion compensation (described in Chapter 6) is used to improve motion compensation and reduce 'blockiness' in the decoded image. Annex F requires the CODEC to support unrestricted motion vectors (Annex D).

Annex G, PB-frames A PB-frame is a pair of frames coded as a combined unit. The first frame is coded as a 'B-picture' and the second as a P-picture. The P-picture is forward predicted from the previous I- or P-picture and the B-picture is bidirectionally predicted from the previous and current I- or P-pictures. Unlike MPEG-1 (where a B-picture is coded as a separate unit), each macroblock of the PB-frame contains data from both the P-picture and the B-picture (Figure 5.3). PB-frames can give an improvement in compression efficiency.

Annex I, Advanced intra-coding This mode exploits the correlation between DCT coefficients in neighbouring intra-coded blocks in an image. The DC coefficient and the first row or column of AC coefficients may be predicted from the coefficients of neighbouring blocks (Figure 5.4). The zigzag scan, quantisation procedure and variable-length code tables are modified and the result is an improvement in compression efficiency for intra-coded macroblocks.

Annex J, Deblocking filter The edges of each 8×8 block are 'smoothed' using a spatial filter (described in Chapter 9). This reduces 'blockiness' in the decoded picture and also improves motion compensation performance. When the deblocking filter is switched on, four

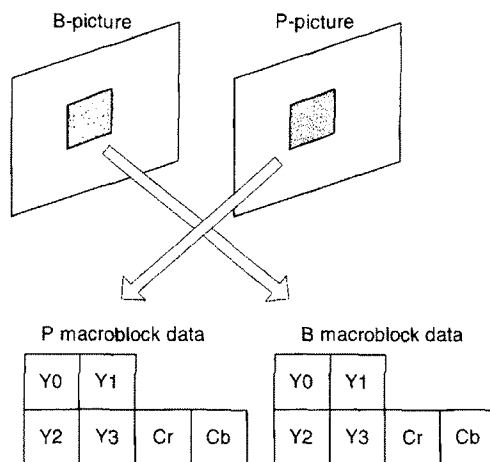


Figure 5.3 Macroblock in PB-frame

motion vectors per macroblock and unrestricted motion vectors are also enabled (Annexes D and F).

Annex K, Slice structured mode This mode provides support for resynchronisation intervals that are similar to MPEG-1 'slices'. A slice is a series of coded macroblocks

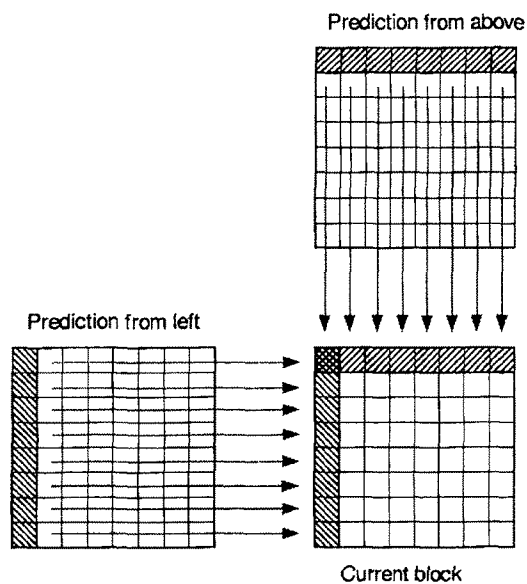


Figure 5.4 Prediction of intra-coefficients, H.263 Annex I

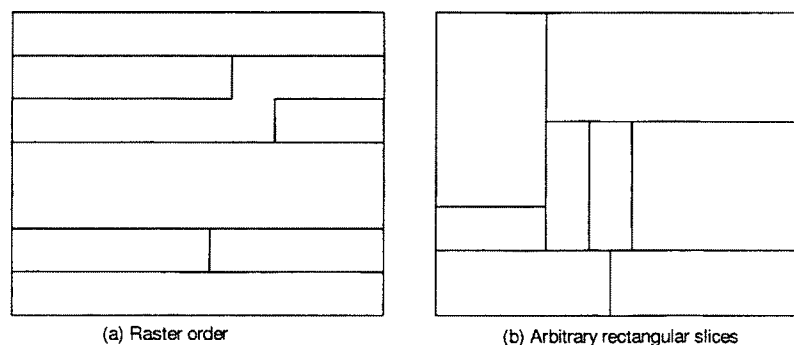


Figure 5.5 H.263 Annex K: slice options

starting with a slice header. Slices may contain macroblocks in raster order, or in any rectangular region of the picture (Figure 5.5). Slices may optionally be sent in an arbitrary order. Each slice may be decoded independently of any other slice in the picture and so slices can be useful for error resilience (see Chapter 11) since an error in one slice will not affect the decoding of any other slice.

Annex L, Supplemental enhancement information This annex contains a number of supplementary codes that may be sent by an encoder to a decoder. These codes indicate display-related information about the video sequence, such as picture freeze and timing information.

Annex M, Improved PB-frames As the name suggests, this is an improved version of the original PB-frames mode (Annex G). Annex M adds the options of forward or backward prediction for the B-frame part of each macroblock (as well as the bidirectional prediction defined in Annex G), resulting in improved compression efficiency.

Annex N, Reference picture selection This mode enables an encoder to choose from a number of previously coded pictures for predicting the current picture. The use of this mode to limit error propagation in a noisy transmission environment is discussed in Chapter 11. At the start of each GOB or slice, the encoder may choose the preferred reference picture for prediction of macroblocks in that GOB or slice.

Annex O, Scalability Temporal, spatial and SNR scalability are supported by this optional mode. In a similar way to the MPEG-2 optional scalability modes, spatial scalability increases frame resolution, SNR scalability increases picture quality and temporal scalability increases frame rate. In each case, a 'base layer' provides basic performance and the increased performance is obtained by decoding the base layer together with an 'enhancement layer'. Temporal scalability is particularly useful because it supports B-pictures: these are similar to the 'true' B-pictures in the MPEG standards (where a B-picture is a separate coded unit) and are more flexible than the combined PB-frames described in Annexes G and M.

Annex P, Reference picture resampling The prediction reference frame used by the encoder and decoder may be resampled prior to motion compensation. This has several possible applications. For example, an encoder can change the frame resolution 'on the fly' whilst continuing to use motion-compensated prediction. The prediction reference frame is resampled to match the new resolution and the current frame can then be predicted from the resampled reference. This mode may also be used to support *warping*, i.e. the reference picture is warped (deformed) prior to prediction, perhaps to compensate for nonlinear camera movements such as zoom or rotation.

Annex Q, Reduced resolution update An encoder may choose to update selected macroblocks at a lower resolution than the normal spatial resolution of the frame. This may be useful, for example, to enable a CODEC to refresh moving parts of a frame at a low resolution using a small number of coded bits whilst keeping the static parts of the frame at the original higher resolution.

Annex R, Independent segment decoding This annex extends the concept of the independently decodeable slices (Annex K) or GOBs. Segments of the picture (where a segment is one slice or an integral number of GOBs) may be decoded completely independently of any other segment. In the slice structured mode (Annex K), motion vectors can point to areas of the reference picture that are outside the current slice; with independent segment decoding, motion vectors and other predictions can only reference areas within the current segment in the reference picture (Figure 5.6). A segment can be decoded (over a series of frames) independently of the rest of the frame.

Annex S, Alternative inter-VLC The encoder may use an alternative variable-length code table for transform coefficients in inter-coded blocks. The alternative VLCs (actually the same VLCs used for intra-coded blocks in Annex I) can provide better coding efficiency when there are a large number of high-valued quantised DCT coefficients (e.g. if the coded bit rate is high and/or there is a lot of variation in the video scene).

Annex T, Modified quantisation This mode introduces some changes to the way the quantiser and rescaling operations are carried out. Annex T allows the encoder to change the

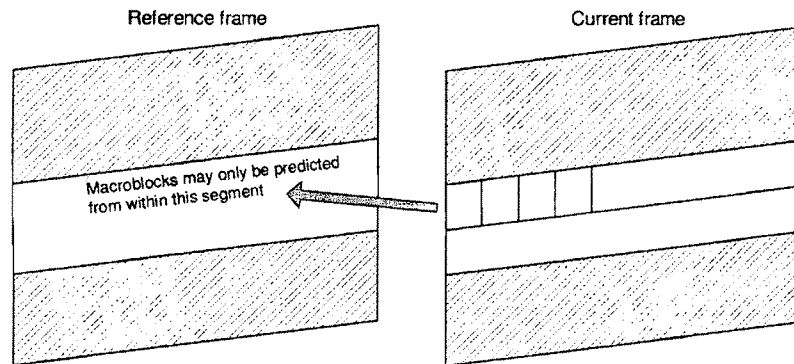


Figure 5.6 Independent segments

quantiser scale factor in a more flexible way during encoding, making it possible to control the encoder output bit rate more accurately.

Annex U, Enhanced reference picture selection Annex U modifies the reference picture selection mode of Annex N to provide improved error resilience and coding efficiency. There are a number of changes, including a mechanism to reduce the memory requirements for storing previously coded pictures and the ability to select a reference picture for motion compensation on a macroblock-by-macroblock basis. This means that the ‘best’ match for each macroblock may be selected from any of a number of stored previous pictures (also known as *long-term memory prediction*).

Annex V, Data partitioned slice Modified from Annex K, this mode improves the resilience of slice structured data to transmission errors. Within each slice, the macroblock data is rearranged so that all of the macroblock headers are transmitted first, followed by all of the motion vectors and finally by all of the transform coefficient data. An error occurring in header or motion vector data usually has a more serious effect on the decoded picture than an error in transform coefficient data: by rearranging the data in this way, an error occurring part-way through a slice should only affect the less-sensitive transform coefficient data.

Annex W, Additional supplemental enhancement information Two extra enhancement information items are defined (in addition to those defined in Annex L). The ‘fixed-point IDCT’ function indicates that an approximate inverse DCT (IDCT) may be used rather than the ‘exact’ definition of the IDCT given in the standard: this can be useful for low-complexity fixed-point implementations of the standard. The ‘picture message’ function allows the insertion of a user-definable message into the coded bit stream.

5.4.1 H.263 Profiles

It is very unlikely that all 19 optional modes will be required for any one application. Instead, certain combinations of modes may be useful for particular transmission scenarios. In common with MPEG-2 and MPEG-4, H.263 defines a set of recommended *profiles* (where a profile is a subset of the optional tools) and *levels* (where a level sets a maximum value on certain coding parameters such as frame resolution, frame rate and bit rate). Profiles and levels are defined in the final annex of H.263, Annex X. There are a total of nine profiles, as follows.

Profile 0, Baseline This is simply the baseline H.263 functionality, without any optional modes.

Profile 1, Coding efficiency (Version 2) This profile provides efficient coding using only tools available in Versions 1 and 2 of the standard (i.e. up to Annex T). The selected optional modes are Annex I (Advanced Intra-coding), Annex J (De-blocking Filter), Annex L (Supplemental Information: only the full picture freeze function is supported) and Annex T (Modified Quantisation). Annexes I, J and T provide improved coding efficiency compared with the baseline mode. Annex J incorporates the ‘best’ features of the first version of the standard, four motion vectors per macroblock and unrestricted motion vectors.

Profile 2, Coding efficiency (Version 1) Only tools available in Version 1 of the standard are used in this profile and in fact only Annex F (Advanced Prediction) is included. The other three annexes (D, E, G) from the original standard are not (with hindsight) considered to offer sufficient coding gains to warrant their use.

Profiles 3 and 4, Interactive and streaming wireless These profiles incorporate efficient coding tools (Annexes I, J and T) together with the slice structured mode (Annex K) and, in the case of Profile 4, the data partitioned slice mode (Annex V). These slice modes can support increased error resilience which is important for 'noisy' wireless transmission environments.

Profiles 5, 6, 7, Conversational These three profiles support low-delay, high-compression 'conversational' applications (such as video telephony). Profile 5 includes tools that provide efficient coding; Profile 6 adds the slice structured mode (Annex K) for Internet conferencing; Profile 7 adds support for interlaced camera sources (part of Annex W).

Profile 8, High latency For applications that can tolerate a higher latency (delay), such as streaming video, Profile 8 adds further efficient coding tools such as B-pictures (Annex O) and reference picture resampling (Annex P). B-pictures increase coding efficiency at the expense of a greater delay.

The remaining tools within the 19 annexes are not included in any profile, either because they are considered to be too complex for anything other than special-purpose applications, or because more efficient tools have superseded them.

5.5 H.26L³

The 19 optional modes of H.263 improved coding efficiency and transmission capabilities; however, development of H.263 standard is constrained by the requirement to continue to support the original 'baseline' syntax. The latest standardisation effort by the Video Coding Experts Group is to develop a new coding syntax that offers significant benefits over the older H.261 and H.263 standards. This new standard is currently described as 'H.26L', where the L stands for 'long term' and refers to the fact that this standard was planned as a long-term solution beyond the 'near-term' additions to H.263 (Versions 2 and 3).

The aim of H.26L is to provide a 'next generation' solution for video coding applications offering significantly improved coding efficiency whilst reducing the 'clutter' of the many optional modes in H.263. The new standard also aims to take account of the changing nature of video coding applications. Early applications of H.261 used dedicated CODEC hardware over the low-delay, low-error-rate ISDN. The recent trend is towards software-only or mixed software/hardware CODECs (where computational resources are limited, but greater flexibility is possible than with a dedicated hardware CODEC) and more challenging transmission scenarios (such as wireless links with high error rates and packet-based transmission over the Internet).

H.26L is currently at the test model development stage and may continue to evolve before standardisation. The main features can be summarised as follows.

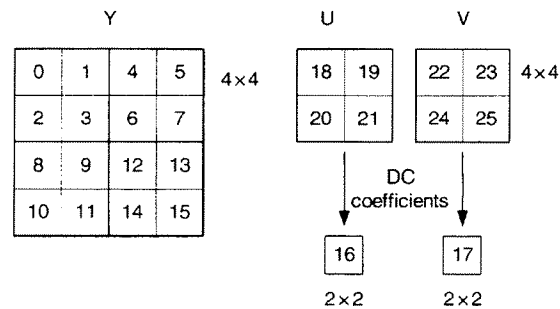


Figure 5.7 H.26L blocks in a macroblock

Processing units The basic unit is the macroblock, as with the previous standards. However, the subunit is now a 4×4 block (rather than an 8×8 block). A macroblock contains 26 blocks in total (Figure 5.7): 16 blocks for the luminance (each 4×4), four 4×4 blocks each for the chrominance components and two 2×2 'sub-blocks' which hold the DC coefficients of each of the eight chrominance blocks. It is more efficient to code these DC coefficients together because they are likely to be highly correlated.

Intra-prediction Before coding a 4×4 block within an intra-macroblock, each pixel in the block is predicted from previously coded pixels. This prediction reduces the amount of data coded in low-detail areas of the picture.

Prediction reference for inter-coding In a similar way to Annexes N and U of H.263, the reference frame for predicting the current inter-coded macroblock may be selected from a range of previously coded frames. This can improve coding efficiency and error resilience at the expense of increased complexity and storage.

Sub-pixel motion vectors H.26L supports motion vectors with $\frac{1}{4}$ pixel and (optionally) $\frac{1}{8}$ pixel accuracy; $\frac{1}{4}$ -pixel vectors can give an appreciable improvement in coding efficiency over $\frac{1}{2}$ -pixel vectors (e.g. H.263, MPEG-4) and $\frac{1}{8}$ -pixel vectors can give a small further improvement (at the expense of increased complexity).

Motion vector options H.26L offers seven different options for allocating motion vectors within a macroblock, ranging from one vector per macroblock (Mode 1 in Figure 5.8) to an individual vector for each of the 16 luminance blocks (Mode 7 in Figure 5.8). This makes it possible to model the motion of irregular-shaped objects with reasonable accuracy. More motion vectors require extra bits to encode and transmit and so the encoder must balance the choice of motion vectors against coding efficiency.

De-blocking filter The de-blocking filter defined in Annex J of H.263 significantly improves motion compensation efficiency because it improves the 'smoothness' of the reference frame used for motion compensation. H.26L includes an integral de-blocking filter that operates across the edges of the 4×4 blocks within each macroblock.

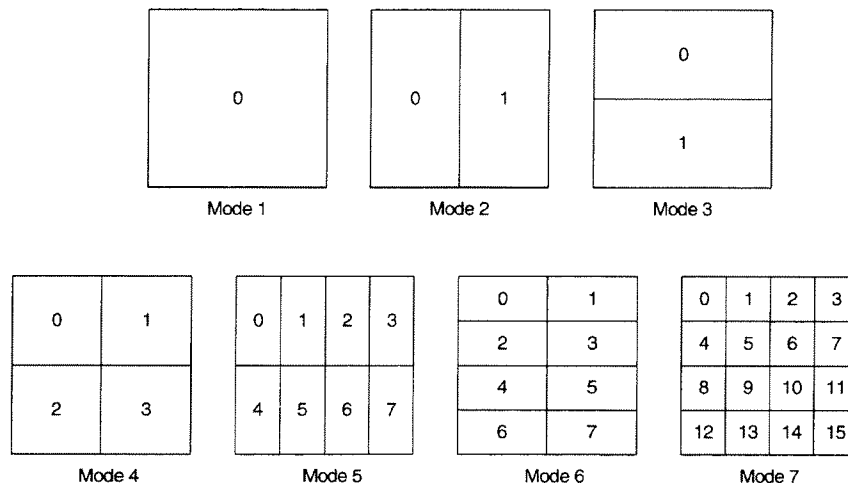


Figure 5.8 H.26L motion vector modes

4 × 4 Block transform After motion compensation, the residual data within each block is transformed using a 4 × 4 block transform. This is based on a 4 × 4 DCT but is an integer transform (rather than the floating-point ‘true’ DCT). An integer transform avoids problems caused by mismatches between different implementations of the DCT and is well suited to implementation in fixed-point arithmetic units (such as low-power embedded processors, Chapter 13).

Universal variable-length code The VLC tables in H.263 are replaced with a single ‘universal’ VLC. A transmitted code is created by building up a regular VLC from the ‘universal’ codeword. These codes have two advantages: they can be implemented efficiently in software without the need for storage of large tables and they are reversible, making it easier to recover from transmission errors (see Chapters 8 and 11 for further discussion of VLCs and error resilience).

Content-based adaptive binary arithmetic coding This alternative entropy encoder uses arithmetic coding (described in Chapter 8) to give higher compression efficiency than variable-length coding. In addition, the encoder can adapt to local image statistics, i.e. it can generate and use accurate probability statistics rather than using predefined probability tables.

B-pictures These are recognised to be a very useful coding tool, particularly for applications that are not very sensitive to transmission delays. H.26L supports B-pictures in a similar way to MPEG-1 and MPEG-2, i.e. there is no restriction on the number of B-pictures that may be transmitted between pairs of I- and/or P-pictures.

At the time of writing it remains to be seen whether H.26L will supersede the popular H.261 and H.263 standards. Early indications are that it offers a reasonably impressive performance gain over H.263 (see the next section): whether these gains are sufficient to merit a ‘switch’ to the new standard is not yet clear.

5.6 PERFORMANCE OF THE VIDEO CODING STANDARDS

Each of the image and video coding standards described in Chapters 4 and 5 was designed for a different purpose and includes different features. This makes it difficult to compare them directly. Figure 5.9 compares the PSNR performance of each of the video coding standards for one particular test video sequence, 'Foreman', encoded at QCIF resolution and a frame rate of 10 frames per second. The results shown in the figure should be interpreted with caution, since different performance will be measured depending on the video sequence, frame rate and so on. However, the trend in performance is clear. MJPEG performs poorly (i.e. it requires a relatively high data rate to support a given picture 'quality') because it does not use any inter-frame compression. H.261 achieves a substantial gain over MJPEG, due to the use of integer-pixel motion compensation. MPEG-2 (with half-pixel motion compensation) is next, followed by H.263/MPEG-4 (which achieve a further gain by using four motion vectors per macroblock). The emerging H.26L test model achieves the best performance of all. (Note that MPEG-1 achieves the same performance as MPEG-2 in this test because the video sequence is not interlaced.)

This comparison is not the complete picture because it does not take into account the special features of particular standards (for example, the content-based tools of MPEG-4 or the interlaced video tools of MPEG-2). Table 5.1 compares the standards in terms of coding performance and features. At the present time, MPEG-2, H.263 and MPEG-4 are each viable

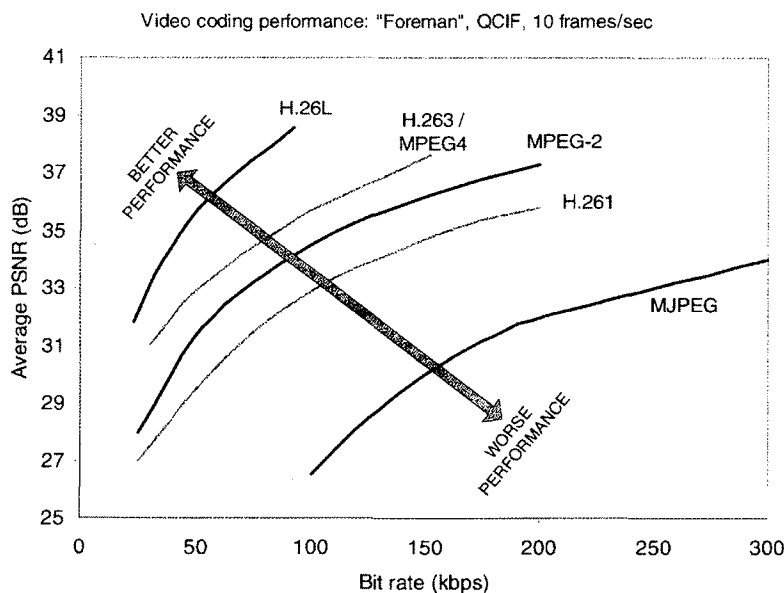


Figure 5.9 Coding performance comparison

Table 5.1 Comparison of the video coding standards

Standard	Target application	Coding performance	Features
MJPEG	Image coding	1 (worst)	Scalable and lossless coding modes
H.261	Video conferencing	2	Integer-pixel motion compensation
MPEG-1	Video-CD	3 (equal)	I, P, B-pictures, half-pixel compensation
MPEG-2	Digital TV	3 (equal)	As above: field coding, scalable coding
H.263	Video conferencing	4 (equal)	Optimised for low bit rates; many optional modes
MPEG-4	Multimedia coding	4 (equal)	Many options including content-based tools
H.26L	Video conferencing	5 (best)	Full feature set not yet defined

alternatives for designers of video communication systems. MPEG-2 is a relatively mature technology for the mass-market digital television applications; H.263 offers good coding performance and options to support a range of transmission scenarios; MPEG-4 provides a large toolkit with the potential for new and innovative content-based applications. The emerging H.26L standard promises to outperform the H.263 and MPEG-4 standards in terms of video compression efficiency⁴ but is not yet finalised.

5.7 SUMMARY

The ITU-T Video Coding Experts Group developed the H.261 standard for video conferencing applications which offered reasonable compression performance with relatively low complexity. This was superseded by the popular H.263 standard, offering better performance through features such as half-pixel motion compensation and improved variable-length coding. Two further versions of H.263 have been released, each offering additional optional coding modes to support better compression efficiency and greater flexibility. The latest version (Version 3) includes 19 optional modes, but is constrained by the requirement to support the original, 'baseline' H.263 CODEC. The H.26L standard, under development at the time of writing, incorporates a number of new coding tools such as a 4×4 block transform and flexible motion vector options and promises to outperform earlier standards.

Comparing the performance of the various coding standards is difficult because a direct 'rate-distortion' comparison does not take into account other factors such as features, flexibility and market penetration. It seems clear that the H.263, MPEG-2 and MPEG-4 standards each have their advantages for designers of video communication systems. Each of these standards makes use of common coding technologies: motion estimation and compensation, block transformation and entropy coding. In the next section of this book we will examine these core technologies in detail.

REFERENCES

1. ITU-T Recommendation H.261, 'Video CODEC for audiovisual services at px64 kbit/s', 1993.
2. ITU-T Recommendation H.263, 'Video coding for low bit rate communication', Version 2, 1998.
3. ITU-T Q6/SG16 VCEG-L45, 'H.26L Test Model Long Term Number 6 (TML-6) draft 0', March 2001.
4. ITU-T Q6/SG16 VCEG-M08, 'Objective coding performance of [H.26L] TML 5.9 and H.263+', March 2001.