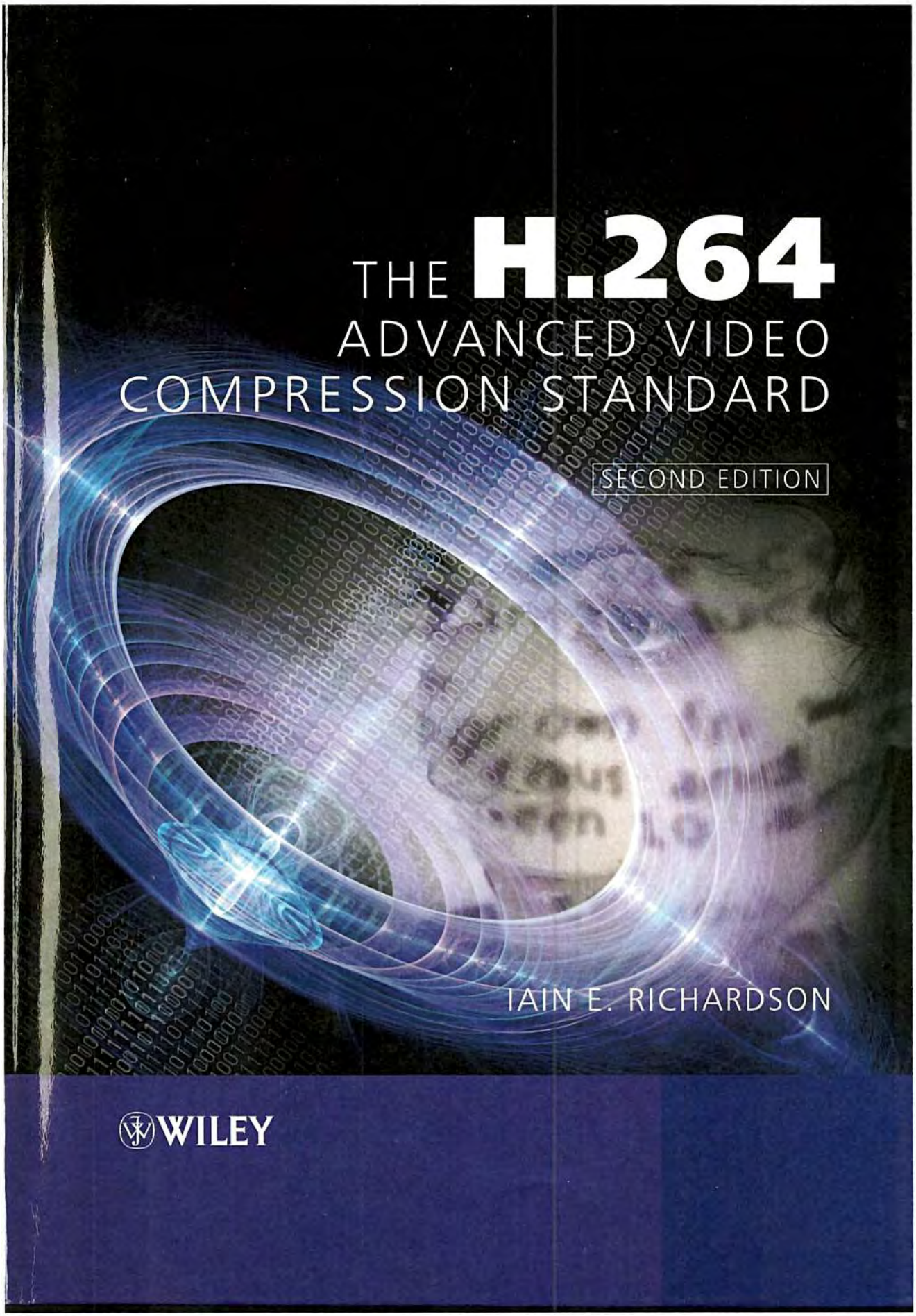


Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000001
IPR2024-00627 (Amazon.com, Inc., Amazon.com Services LLC v. Nokia Technologies Oy)

THE H.264 ADVANCED VIDEO COMPRESSION STANDARD

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000002

THE H.264 ADVANCED VIDEO COMPRESSION STANDARD

Second Edition

Iain E. Richardson
Vcodex Limited, UK

WILEY
A John Wiley and Sons, Ltd., Publication

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000003

This edition first published 2010
© 2010, John Wiley & Sons, Ltd

First Edition published in 2003

Registered office
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

For details of our global editorial offices, for customer services and for information about how to apply for permission to reuse the copyright material in this book please see our website at www.wiley.com.

The right of the author to be identified as the author of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988.

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 or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior permission of the publisher.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book. This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Library of Congress Cataloguing-in-Publication Data

Richardson, Iain E. G.
The H.264 advanced video compression standard / Iain E. Richardson. – 2nd ed.
p. cm.
Rev. ed. of: H.264 and MPEG-4 video compression. c2003.
Includes bibliographical references and index.
ISBN 978-0-470-51692-8 (pbk.)
1. Digital video–Standards. 2. Video compression–Standards. 3. MPEG (Video coding standard) 4. Multimedia systems. I. Richardson, Iain E. G. H.264 and MPEG-4 video compression. II. Title.
TK6680.5.R52 2010
006.6'96–dc22

2009054387

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

ISBN: 978-0-470-51692-8

Typeset in 10/12pt Times by Aptara Inc., New Delhi, India

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000004

To Pat

Language is living.
but what is most important
goes deeper than words.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000005

Contents

About the Author		**xiii**
Preface		**xv**
Glossary		**xvii**
List of Figures		**xxi**
List of Tables		**xxix**
1	**Introduction**	**1**
1.1	A change of scene	1
1.2	Driving the change	4
1.3	The role of standards	4
1.4	Why H.264 Advanced Video Coding is important	4
1.5	About this book	5
1.6	Reference	6
2	**Video formats and quality**	**7**
2.1	Introduction	7
2.2	Natural video scenes	7
2.3	Capture	8
2.3.1	Spatial sampling	9
2.3.2	Temporal sampling	9
2.3.3	Frames and fields	11
2.4	Colour spaces	12
2.4.1	RGB	12
2.4.2	YCrCb	13
2.4.3	YCrCb sampling formats	14
2.5	Video formats	16
2.5.1	Intermediate formats	16
2.5.2	Standard Definition	17
2.5.3	High Definition	18
2.6	Quality	19
2.6.1	Subjective quality measurement	20
2.6.2	Objective quality measurement	21

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000006

2.7	Summary	24
2.8	References	24
3	**Video coding concepts**	**25**
3.1	Introduction	25
3.2	Video CODEC	26
3.3	Prediction model	28
	3.3.1 Temporal prediction	28
	3.3.2 Spatial model: intra prediction	38
3.4	Image model	40
	3.4.1 Predictive image coding	41
	3.4.2 Transform coding	42
	3.4.3 Quantization	50
	3.4.4 Reordering and zero encoding	52
3.5	Entropy coder	57
	3.5.1 Predictive coding	57
	3.5.2 Variable-length coding	58
	3.5.3 Arithmetic coding	65
3.6	The hybrid DPCM/DCT video CODEC model	68
3.7	Summary	79
3.8	References	79
4	**What is H.264?**	**81**
4.1	Introduction	81
4.2	What is H.264?	81
	4.2.1 A video compression format	81
	4.2.2 An industry standard	82
	4.2.3 A toolkit for video compression	83
	4.2.4 Better video compression	83
4.3	How does an H.264 codec work?	83
	4.3.1 Encoder processes	85
	4.3.2 Decoder processes	89
4.4	The H.264/AVC Standard	91
4.5	H.264 Profiles and Levels	92
4.6	The H.264 Syntax	94
4.7	H.264 in practice	97
	4.7.1 Performance	97
	4.7.2 Applications	98
4.8	Summary	98
4.9	References	98
5	**H.264 syntax**	**99**
5.1	Introduction	99
	5.1.1 A note about syntax examples	99
5.2	H.264 syntax	100
5.3	Frames, fields and pictures	101

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000007

	5.3.1	Decoding order	104
	5.3.2	Display order	104
	5.3.3	Reference picture lists	106
	5.3.4	Frame and field coding	111
5.4	NAL unit		114
5.5	Parameter Sets		115
5.6	Slice layer		117
	5.6.1	Slice types	117
	5.6.2	Slice header	117
	5.6.3	Slice data	118
5.7	Macroblock layer		119
	5.7.1	Overview	119
	5.7.2	The Intra PCM mode	121
	5.7.3	Macroblock prediction	122
	5.7.4	Residual data	124
	5.7.5	Macroblock syntax examples	127
5.8	Summary		134
5.9	References		135
6	**H.264 Prediction**		**137**
6.1	Introduction		137
6.2	Macroblock prediction		137
6.3	Intra prediction		138
	6.3.1	4×4 luma prediction modes	143
	6.3.2	16×16 luma prediction modes	146
	6.3.3	Chroma prediction modes	147
	6.3.4	8×8 luma prediction. High profiles	148
	6.3.5	Signalling intra prediction modes	148
6.4	Inter prediction		149
	6.4.1	Reference pictures	151
	6.4.2	Interpolating reference pictures	152
	6.4.3	Macroblock partitions	157
	6.4.4	Motion vector prediction	158
	6.4.5	Motion compensated prediction	162
	6.4.6	Inter prediction examples	164
	6.4.7	Prediction structures	169
6.5	Loop filter		171
	6.5.1	Boundary strength	172
	6.5.2	Filter decision	173
	6.5.3	Filter implementation	174
	6.5.4	Loop filter example	174
6.6	Summary		177
6.7	References		177
7	**H.264 transform and coding**		**179**
7.1	Introduction		179

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000008

7.2	Transform and quantization		179
	7.2.1	The H.264 transforms	179
	7.2.2	Transform processes	180
	7.2.3	Integer transform and quantization: 4 × 4 blocks	185
	7.2.4	Integer transform and quantization: 8 × 8 blocks	198
	7.2.5	DC transforms	203
	7.2.6	Transform and quantization extensions in the High profiles	204
7.3	Block scan orders		206
7.4	Coding		207
	7.4.1	Exp-Golomb Coding	208
	7.4.2	Context Adaptive Variable Length Coding, CAVLC	210
	7.4.3	Context Adaptive Binary Arithmetic Coding, CABAC	217
7.5	Summary		220
7.6	References		221
8	**H.264 conformance, transport and licensing**		**223**
8.1	Introduction		223
8.2	Conforming to the Standard		223
	8.2.1	Profiles	224
	8.2.2	Levels	226
	8.2.3	Hypothetical Reference Decoder	230
	8.2.4	Conformance testing	236
8.3	H.264 coding tools for transport support		237
	8.3.1	Redundant slices	237
	8.3.2	Arbitrary Slice Order (ASO)	238
	8.3.3	Slice Groups/Flexible Macroblock Order (FMO)	238
	8.3.4	SP and SI slices	240
	8.3.5	Data partitioned slices	243
8.4	Transport of H.264 data		244
	8.4.1	Encapsulation in RBSPs, NALUs and packets	244
	8.4.2	Transport protocols	245
	8.4.3	File formats	247
	8.4.4	Coding and transport issues	247
8.5	Supplemental Information		248
	8.5.1	Supplemental Enhancement Information (SEI)	248
	8.5.2	Video Usability Information (VUI)	248
8.6	Licensing H.264/AVC		248
	8.6.1	Video coding patents	250
	8.6.2	Video coding standards and patents	252
	8.6.3	Licensing H.264/AVC patents	252
8.7	Summary		253
8.8	References		253
9	**H.264 performance**		**255**
9.1	Introduction		255
9.2	Experimenting with H.264		256

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000009

	9.2.1	The JM Reference Software	256
	9.2.2	Other software encoders/decoders	261
	9.2.3	H.264 stream analysis	263
9.3	Performance comparisons		265
	9.3.1	Performance criteria	265
	9.3.2	Performance examples: Foreman sequence, QCIF resolution	265
	9.3.3	Performance examples: Foreman and Container sequences	269
	9.3.4	Performance examples: Inter prediction structures	271
	9.3.5	Performance example: H.264 vs. MPEG-4 Visual	273
9.4	Rate control		274
	9.4.1	Rate control in the JM reference encoder	276
9.5	Mode selection		279
	9.5.1	Rate Distortion Optimized mode selection	281
9.6	Low complexity coding		283
	9.6.1	Approximating the cost function	283
	9.6.2	Reducing the set of tested modes	284
	9.6.3	Early termination	285
9.7	Summary		285
9.8	References		285
10	**Extensions and directions**		**287**
10.1	Introduction		287
10.2	Scalable Video Coding		288
	10.2.1	Simulcast transmission	288
	10.2.2	Scalable transmission	289
	10.2.3	Applications of Scalable Video Coding	290
	10.2.4	Scalable Video Coding in H.264	290
	10.2.5	Temporal scalability	292
	10.2.6	Quality scalability: overview	294
	10.2.7	Spatial scalability: overview	294
	10.2.8	Spatial scalability in detail	294
	10.2.9	Quality scalability in detail	298
	10.2.10	Combined scalability	299
	10.2.11	SVC performance	299
10.3	Multiview Video Coding		302
	10.3.1	H.264 Multiview Video Coding	304
10.4	Configurable Video Coding		306
	10.4.1	MPEG Reconfigurable Video Coding	307
	10.4.2	Fully Configurable Video Coding	308
10.5	Beyond H.264/AVC		310
10.6	Summary		310
10.7	References		311
Index			**313**

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000010

About the Author

Professor Iain Richardson is an internationally known expert on the MPEG and H.264 video compression standards.

The author of *H.264 and MPEG-4 Video Compression*, a widely cited work in the research literature, Professor Richardson has written two further books and over 70 journal and conference papers on video compression. He regularly advises companies on video compression technology, video coding patents and company acquisitions in the video coding industry. Professor Richardson leads an internationally renowned video coding research team, contributes to the MPEG industry standards group and is sought after as an expert witness. Based in Aberdeen, Scotland, he regularly travels to the US and Europe.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000011

Preface

The last decade has seen a quiet revolution in digital video technology. Digital video is everywhere: on our televisions, our DVD and Blu-Ray players, our computers, our music players and our mobile handsets. Only recently, a video image in a web page was an unusual sight. Nowadays, many of us are just as likely to catch the latest news on the web as on the TV. With the explosion of digital video applications, a billion-dollar industry has developed and expanded, with new companies and niche markets emerging, thriving and disappearing faster than anyone can easily track. Video compression is essential to all of these applications and markets, and the H.264 format is considered by many to be the state of the art in video compression.

When I wrote the first edition of this book in 2003, H.264 Advanced Video Compression had just been published as an International Standard and it was hard to predict its impact on industry. Its predecessor, MPEG-4 Visual, had arguably failed to live up to its promise, with only limited adoption in the market. Since 2003, the significant performance improvements that are built into H.264 have made it the clear successor to the older MPEG video standards in many applications, from mobile video to High Definition broadcasting. At the time of writing, the MPEG and VCEG standards committees are debating the possible successor to H.264. It is likely to be several years before a new standard is released, and several years after that before H.264 begins to become obsolete.

This book is intended to be a practical, accessible and unbiased guide to the H.264 video compression standard. As always, I have chosen to explain the details of H.264 in my own way, concentrating on what I feel is important to the engineer, researcher or student who needs a 'way in' to this complex yet important technical subject. This book is not the final word on H.264. By definition, that final word is provided by the standard itself and I advise any serious developer or implementer of H.264 to get hold of a copy of the standard. There is a need for a guidebook to the standard that explains the concepts, tools, benefits and disadvantages of the format, just as a good guidebook helps the tourist to get to know a foreign country and to become more at home there. Some visitors may be disappointed that their favourite subject is not covered in as much depth as they would like. I have made a deliberate choice to cover certain topics such as Scalable and Multiview Video Coding only briefly as they are still, in my view, in the early stages of practical implementation.

My sincere thanks to the many people who have helped to shape this book, including the readers of my earlier books who told me what they liked and what they wanted; the many companies and individuals who have asked me to solve their video compression problems; Kourosh Soroushian for discussions on Hypothetical Reference Decoders; Abharana Bhat,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000012

Maja Bystrom, Sam Jansen, Sampath Kannangara and Yafan Zhao for reading and commenting on draft chapters; Gary Sullivan for many comments, corrections, suggestions and discussions; Nicky, Simone and the editorial team at John Wiley & Sons; and to Pat for reading the manuscript, cracking the whip and making me finish it.

I hope that you find the book useful; more importantly, I hope you enjoy it. Visit my website at www.vcodex.com and *tell me what you think*.

Iain Richardson
Aberdeen, 2010

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000013

Glossary

4:2:0 (sampling)	Sampling method: chrominance components have half the horizontal and vertical resolution of luminance component
4:2:2 (sampling)	Sampling method: chrominance components have half the horizontal resolution of luminance component
4:4:4 (sampling)	Sampling method: chrominance components have same resolution as luminance component
access unit	Complete coded frame or field
arithmetic coding	Coding method to reduce redundancy
artefact	Visual distortion in an image
ASO	Arbitrary Slice Order, in which slices may be coded out of raster sequence
block	Region of macroblock
block matching	Motion estimation carried out on rectangular picture areas
blocking	Square or rectangular distortion areas in an image
B slice	Coded slice predicted using bidirectional motion compensation
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
chrominance or chroma	Colour difference component
CIF	Common Intermediate Format, a colour image format
CODEC	COder / DECoder pair
Coded Picture Buffer (CPB)	Buffer containing coded frames or fields
colour space	Method of representing colour images
DCT	Discrete Cosine Transform, a mathematical transform and/or its practical approximation(s)
direct prediction	A coding mode in which no motion vector is transmitted
DPCM	Differential Pulse Code Modulation

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000014

DSCQS	Double Stimulus Continuous Quality Scale, a scale and method for subjective quality measurement
DWT	Discrete Wavelet Transform
entropy coding	Coding method to reduce redundancy
error concealment	Post-processing of a decoded image to remove or reduce visible error effects
Exp-Golomb or ExpG	Exponential Golomb variable length codes
field	Odd- or even-numbered lines from an interlaced video sequence
FMO	Flexible Macroblock Order, in which macroblocks may be coded out of raster sequence
Full Search	A motion estimation algorithm
Fully Configurable Video Coding	A framework for video coding in which a codec may be completely re-configured during a communication session
GOP	Group of Pictures, a set of coded video images
H.261	A video coding standard
H.263	A video coding standard
H.264	A video coding standard
HDTV	High Definition Television
Huffman coding	Coding method to reduce redundancy
HVS	Human Visual System, the system by which humans perceive and interpret visual images
hybrid (CODEC)	CODEC model featuring motion compensation and transform
Hypothetical Reference Decoder (HRD)	Decoder 'model' that may be used to test bitstream conformance
IEC	International Electrotechnical Commission, a standards body
inter (coding)	Coding of video frames using temporal prediction or compensation
interlaced (video)	Video data represented as a series of fields
intra (coding)	Coding of video frames without temporal prediction
I slice	Slice coded without reference to any other frame
ISO	International Standards Organisation, a standards body
ITU	International Telecommunication Union, a standards body
JPEG	Joint Photographic Experts Group, a committee of ISO (also an image coding standard)
latency	Delay through a communication system
Level	A set of conformance parameters (applied to a Profile)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000015

loop filter	Spatial filter placed within encoding or decoding feedback loop
luminance or luma	Monochrome or brightness component
Macroblock	Region of frame coded as a unit (usually 16 × 16 pixels in the original frame)
Macroblock partition	Region of macroblock with its own motion vector
Macroblock sub-partition	Region of macroblock with its own motion vector
motion compensation	Prediction of a video frame with modelling of motion
motion estimation	Estimation of relative motion between two or more video frames
motion vector	Vector indicating a displaced block or region to be used for motion compensation
MPEG	Motion Picture Experts Group, a committee of ISO/IEC
MPEG-1	A multimedia coding standard
MPEG-2	A multimedia coding standard
MPEG-4	A multimedia coding standard
MVC	Multiview Video Coding, in which multiple views of a scene may be jointly coded
NAL	Network Abstraction Layer
objective quality	Visual quality measured by algorithm(s)
Picture (coded)	Coded (compressed) video frame
P-picture (slice)	Coded picture (or slice) using motion-compensated prediction from one reference frame
profile	A set of functional capabilities (of a video CODEC)
progressive (video)	Video data represented as a series of complete frames
PSNR	Peak Signal to Noise Ratio, an objective quality measure
QCIF	Quarter Common Intermediate Format
quantize	Reduce the precision of a scalar or vector quantity
rate control	Control of bit rate of encoded video signal
rate-distortion	Measure of CODEC performance (distortion at a range of coded bit rates)
RBSP	Raw Byte Sequence Payload
RVC	Reconfigurable Video Coding, a framework for video coding in which a decoder may be constructed from pre-defined Functional Units.
RGB	Red/Green/Blue colour space
ringing (artefacts)	'Ripple'-like artefacts around sharp edges in a decoded image
RTP	Real Time Protocol, a transport protocol for real-time data

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000016

scalable coding	Coding a signal into a number of layers
SVC	Scalable Video Coding
SI slice	Intra-coded slice used for switching between coded bitstreams (H.264)
slice	A region of a coded picture
SP slice	Inter-coded slice used for switching between coded bitstreams
statistical redundancy	Redundancy due to the statistical distribution of data
studio quality	Lossless or near-lossless video quality
subjective quality	Visual quality as perceived by human observer(s)
subjective redundancy	Redundancy due to components of the data that are subjectively insignificant
sub-pixel (motion compensation)	Motion-compensated prediction from a reference area that may be formed by interpolating between integer-valued pixel positions
test model	A software model and document that describe a reference implementation of a video coding standard
texture	Image or residual data
tree-structured motion compensation	Motion compensation featuring a flexible hierarchy of partition sizes
VCEG	Video Coding Experts Group, a committee of ITU
VCL	Video Coding Layer
video packet	Coded unit suitable for packetization
VLC	Variable Length Code
VLD	Variable Length Decoder
VLE	Variable Length Encoder
VLSI	Very Large Scale Integrated circuit
VQEG	Video Quality Experts Group
weighted prediction	Motion compensation in which the prediction samples from two references are scaled
YCrCb	Luminance/Red chrominance/Blue chrominance colour space

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000017

List of Figures

1.1	Video coding scenarios, one-way	3
1.2	Video coding scenario, two-way	3
2.1	Still image from natural video scene	8
2.2	Spatial and temporal sampling of a video sequence	8
2.3	Image with two sampling grids	9
2.4	Image sampled at coarse resolution (black sampling grid)	10
2.5	Image sampled at finer resolution (grey sampling grid)	10
2.6	Interlaced video sequence	11
2.7	Top field	12
2.8	Bottom field	13
2.9	Red, Green and Blue components of colour image	13
2.10	Cr, Cg and Cb components	14
2.11	4:2:0, 4:2:2 and 4:4:4 sampling patterns (progressive)	15
2.12	Allocation of 4:2:0 samples to top and bottom fields	17
2.13	Video frame sampled at range of resolutions	18
2.14	SD and HD formats	19
2.15	DSCQS testing system	21
2.16	PSNR examples: (a) Original; (b) 30.6dB; (c) 28.3dB	22
2.17	Image with blurred background (PSNR = 27.7dB)	22
3.1	Encoder/Decoder	26
3.2	Spatial and temporal correlation in a video sequence	26
3.3	Video encoder block diagram	27
3.4	Frame 1	29
3.5	Frame 2	29
3.6	Difference	29
3.7	Optical flow	30
3.8	Macroblock (4:2:0)	31
3.9	Motion estimation	32
3.10	Frame 1	33
3.11	Frame 2	33
3.12	Residual : no motion compensation	33
3.13	Residual : 16 × 16 block size	34
3.14	Residual : 8 × 8 block size	34
3.15	Residual : 4 × 4 block size	34

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000018

3.16	Close-up of reference region	35
3.17	Reference region interpolated to half-pixel positions	36
3.18	Integer, half-pixel and quarter-pixel motion estimation	36
3.19	Residual : 4 × 4 blocks, 1/2-pixel compensation	37
3.20	Residual : 4 × 4 blocks, 1/4-pixel compensation	37
3.21	Motion vector map : 16 × 16 blocks, integer vectors	38
3.22	Motion vector map : 4 × 4 blocks, 1/4-pixel vectors	39
3.23	Intra prediction: available samples	39
3.24	Intra prediction: spatial extrapolation	39
3.25	2D autocorrelation function of image	40
3.26	2D autocorrelation function of residual	41
3.27	Spatial prediction (DPCM)	41
3.28	4 × 4 DCT basis patterns	45
3.29	8 × 8 DCT basis patterns	45
3.30	Image section showing 4 × 4 block	46
3.31	Close-up of 4 × 4 block; DCT coefficients	47
3.32	Block reconstructed from (a) 1, (b) 2, (c) 3, (d) 5 coefficients	47
3.33	Two-dimensional wavelet decomposition process	48
3.34	Image after one level of decomposition	49
3.35	Two-stage wavelet decomposition of image	49
3.36	Five-stage wavelet decomposition of image	50
3.37	Scalar quantizers: linear; non-linear with dead zone	52
3.38	Vector quantization	53
3.39	8 × 8 DCT coefficient distribution (frame)	53
3.40	Residual field picture	54
3.41	8 × 8 DCT coefficient distribution (field)	54
3.42	Zigzag scan example : frame block	55
3.43	Zigzag scan example : field block	55
3.44	Wavelet coefficient and 'children'	57
3.45	Motion vector prediction candidates	58
3.46	Generating the Huffman code tree: Sequence 1 motion vectors	60
3.47	Huffman tree for sequence 2 motion vectors	61
3.48	MPEG4 TCOEF VLCs (partial)	64
3.49	Sub-range example	66
3.50	Arithmetic coding example	67
3.51	DPCM/DCT video encoder	69
3.52	DPCM/DCT video decoder	69
3.53	Input frame F_n	71
3.54	Reconstructed reference frame F'_{n-1}	71
3.55	Residual F_n-F'_{n-1} : no motion compensation	72
3.56	16 × 16 motion vectors superimposed on frame	73
3.57	Motion compensated reference frame	73
3.58	Motion compensated residual frame	74
3.59	Original macroblock : luminance	74
3.60	Residual macroblock : luminance	75
3.61	DCT coefficient magnitudes : top-right 8 × 8 block	76

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000019

3.62	Comparison of original and decoded residual blocks	78
3.63	Decoded frame F'$_n$	79
4.1	The H.264 video coding and decoding process	82
4.2	Video coding: source frames, encoded bitstream, decoded frames	83
4.3	Video codec: high level view	84
4.4	Typical H.264 encoder	84
4.5	Typical H.264 decoder	85
4.6	Prediction: flow diagram	86
4.7	Intra prediction	86
4.8	Original macroblock, intra prediction and residual	86
4.9	Inter prediction	87
4.10	Original macroblock, inter prediction and residual	87
4.11	Forward transform	88
4.12	Quantization example	88
4.13	Example: Block, transform coefficients, quantized coefficients	88
4.14	Rescaling example	89
4.15	Inverse transform: combining weighted basis patterns to create a 4 × 4 image block	90
4.16	Rescaled coefficients and inverse transform output	90
4.17	Reconstruction flow diagram	91
4.18	Profiles and Levels: example	93
4.19	H.264 syntax : overview	94
4.20	Syntax example: P-macroblock	95
4.21	P-macroblock decoding process	96
4.22	Residual luma and chroma coefficient blocks	97
4.23	A video frame compressed at the same bitrate using MPEG-2 (left), MPEG-4 Visual (centre) and H.264 compression (right)	97
5.1	Syntax overview	100
5.2	Picture handling in H.264, overview	103
5.3	Decoded Picture Buffer and picture orders	104
5.4	Display order: Type 0 example	105
5.5	Display order: Type 1 example	106
5.6	List 0 and List 1 ordering: example	108
5.7	Default reference pictures: example	108
5.8	Reference picture re-ordering syntax, simplified overview	109
5.9	Picture Adaptive Frame Field Coding example	112
5.10	Frame with macroblock pairs	113
5.11	MB pair coded using Macroblock Adaptive Frame Field Coding	113
5.12	Example: Sequence and Picture Parameter Sets	116
5.13	Macroblock layer syntax overview	119
5.14	mb_pred and sub_mb_pred syntax overview	123
5.15	Residual data syntax overview	125
5.16	Block scanning order, 4 × 4 transform, 4:2:0 sampling	125
5.17	Block scanning order, 8 × 8 luma transform, 4:2:0 sampling	126
5.18	Residual CAVLC block syntax overview	126
5.19	Residual CABAC block syntax overview	127

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000020

5.20	P macroblock, example 2	129
5.21	P macroblock example 4	131
5.22	B macroblock, example 5	133
6.1	Example of macroblock types and prediction sources	138
6.2	Intra prediction: adjacent blocks example	139
6.3	Intra prediction source samples, 4 × 4 or 8 × 8 luma blocks	140
6.4	Intra prediction source samples, chroma or 16 × 16 luma blocks	140
6.5	Example of intra block size choices, CIF, Baseline Profile. Reproduced by permission of Elecard.	141
6.6	QCIF frame with highlighted macroblock	142
6.7	Predicted luma frame formed using H.264 intra prediction	142
6.8	Residual after subtracting intra prediction	143
6.9	4 × 4 luma block to be predicted	144
6.10	Labelling of prediction samples, 4 × 4 prediction	144
6.11	4 × 4 intra prediction modes	144
6.12	Prediction blocks, 4 × 4 modes 0–8	145
6.13	Intra 16 × 16 prediction modes	146
6.14	16 × 16 macroblock	146
6.15	Prediction blocks, intra 16 × 16 modes 0–3	147
6.16	Intra mode prediction example	148
6.17	P macroblock prediction example	150
6.18	Example of integer and sub-pixel prediction	152
6.19	Current region	153
6.20	4 × 4 block to be predicted	153
6.21	Reference region	153
6.22	Prediction from integer samples	154
6.23	Reference region, half-pixel interpolated	154
6.24	Prediction from interpolated samples	154
6.25	Interpolation of luma half-pel positions	155
6.26	Interpolation of luma quarter-pel positions	156
6.27	Luma region interpolated to quarter-pel positions	156
6.28	Interpolation of chroma eighth-pel positions	157
6.29	Macroblock partitions and sub-macroblock partitions	158
6.30	Current and neighbouring partitions : same partition sizes	159
6.31	Current and neighbouring partitions : different partition sizes	159
6.32	Scaled motion vector example	160
6.33	Temporal direct motion vector example	161
6.34	Forming a motion compensated prediction	162
6.35	Biprediction example	163
6.36	MBAFF: prediction from corresponding field	164
6.37	P slice showing partition choices. Reproduced by permission of Elecard	165
6.38	B slice showing macroblock modes. Light-shaded circles are skipped macroblocks. Reproduced by permission of Elecard.	166
6.39	Inter prediction example, P slice	167
6.40	Inter prediction example, B slice	168
6.41	Low delay prediction structure	169

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000021

6.42	'Classic' Group of Pictures prediction structure	169
6.43	IPPP... with multiple reference pictures	170
6.44	Hierarchical GOP structure	171
6.45	Edge filtering order in a macroblock	172
6.46	Pixels adjacent to vertical and horizontal boundaries	172
6.47	16 × 16 luma macroblock showing block edges	173
6.48	Original frame, 'violin' frame 2	174
6.49	Reconstructed, QP = 36, no filter	175
6.50	Reconstructed, QP = 36, with filter	175
6.51	Reconstructed, QP = 32, no filter	176
6.52	Reconstructed, QP = 32, with filter	176
7.1	Re-scaling and inverse transform	181
7.2	Forward transform and quantization	181
7.3	Luma forward transform: default	182
7.4	Luma inverse transform: default	182
7.5	Luma forward transform: Intra 16 × 16 mode	183
7.6	Luma inverse transform: Intra 16 × 16 mode	183
7.7	Luma forward transform: 8 × 8 transform	183
7.8	Luma inverse transform: 8 × 8 transform	183
7.9	Chroma forward transform: 4:2:0 macroblock	184
7.10	Chroma inverse transform: 4:2:0 macroblock	184
7.11	Chroma forward transform: 4:2:2 macroblock	184
7.12	Chroma inverse transform: 4:2:2 macroblock	184
7.13	Development of the forward transform and quantization process	185
7.14	Development of the rescaling and inverse transform process	186
7.15	Quantization parameter QP vs. effective quantizer step size, logarithmic y-axis	192
7.16	Frequency dependent quantization, 4 × 4 block	204
7.17	Progressive scan orders for 4 × 4 and 8 × 8 blocks	206
7.18	Field scan orders for 4 × 4 and 8 × 8 blocks	207
7.19	CAVLC encoder overview	211
7.20	CABAC coding process overview	218
8.1	Baseline, Constrained Baseline, Extended and Main Profiles	225
8.2	Main and High Profiles	226
8.3	Main and Intra Profiles	227
8.4	Selected Level constraints	228
8.5	Selected display resolutions	229
8.6	H.264 encoder and decoder buffers	230
8.7	Hypothetical Reference Decoder (HRD)	230
8.8	HRD example 1: encoder buffer	232
8.9	HRD example 1: decoder CPB	232
8.10	HRD Example 2: encoder buffer	234
8.11	HRD Example 2: decoder CPB	234
8.12	HRD Example 3: encoder buffer	235
8.13	HRD Example 3: decoder CPB	236
8.14	Bitstream conformance testing	236

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000022

8.15	Decoder conformance testing	237
8.16	Arbitrary Slice Order: Example	238
8.17	FMO: Interleaved map, QCIF, 3 slice groups	239
8.18	FMO: Dispersed macroblock map, QCIF, 4 slice groups	239
8.19	FMO: Foreground and Background map, 4 slice groups	240
8.20	FMO: Box-out, Raster and Wipe maps	240
8.21	Switching streams using I-slices	241
8.22	Switching streams using SP-slices	242
8.23	Encoding SP-slice A_2 (simplified)	242
8.24	Encoding SP-slice B_2 (simplified)	242
8.25	Decoding SP-slice A_2 (simplified)	243
8.26	Encoding SP-slice AB_2 (simplified)	243
8.27	Decoding SP-slice AB_2	244
8.28	Fast-forward using SP-slices	244
8.29	Encapsulation of H.264 syntax elements	245
8.30	MPEG-2 Transport Stream	246
8.31	RTP packet structure (simplified)	246
8.32	ISO Media File	247
8.33	Block diagram from US patent 3679821 (redrawn)	251
8.34	Issued US patents including the terms 'video coding' or 'video compression', 1990–2007. Source: USPTO patent database.	251
9.1	JM software operation	256
9.2	Planar YCbCr file format, 4:2:0 sampling	257
9.3	JM encoder configuration file	258
9.4	JM encoder output display	258
9.5	Original, reconstructed and decoded frames, container.qcif, QP = 32	259
9.6	Section of coded frame, JM encoder (left), ×264 encoder (right)	262
9.7	Screenshot of an H.264 stream analyzer: Baseline Profile frame. Reproduced by permission of Elecard	263
9.8	Screenshot of stream analyzer: Main Profile frame. Courtesy of Elecard	264
9.9	Foreman/QCIF/Basic complexity	266
9.10	Foreman/QCIF/Basic complexity and options	267
9.11	Foreman/QCIF/Basic and Medium Complexity	268
9.12	Foreman/QCIF/Medium complexity with rate control	269
9.13	Foreman, QCIF sequences: coding time	269
9.14	QCIF sequences: rate vs. PSNR	270
9.15	CIF sequences: rate vs. PSNR	270
9.16	CIF sequences: coding time	271
9.17	Rate-distortion comparison of prediction structures	271
9.18	Sample frames from sequences using different prediction structures, coded at 280kbps	272
9.19	Carphone, QCIF: H.264 vs. MPEG-4 Visual	273
9.20	Frame from 'Foreman' sequence showing macroblock sizes	274
9.21	Encoder with rate feedback	275
9.22	Bitrate allocation for rate control	275
9.23	Foreman, QCIF, 100 frames: coded bitrate	278

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000023

9.24	Foreman, QCIF, 100 frames: QP per frame	278
9.25	Foreman, QCIF, 100 frames: Luma PSNR per frame	279
9.26	Available macroblock prediction modes	280
9.27	Rate and MSE costs for different coding options	281
9.28	Rate, quality and complexity	283
10.1	Multiple streams/simulcast	288
10.2	Multiple streams/scalable	289
10.3	Overview of scalability types	291
10.4	Temporally scalable sequence, 3 layers	292
10.5	Decoding a temporally scalable sequence	293
10.6	Hierarchical prediction structure	293
10.7	Quality Scalability	295
10.8	Spatial scalability, two layers	296
10.9	Medium Grain Quality Scalability	298
10.10	Combined Spatial, Temporal and Quality scalability	299
10.11	Simulcast vs. scalable bitrates	300
10.12	Quality scalability at CIF resolution	301
10.13	Spatial + temporal scalability, CIF → 4CIF resolution	301
10.14	Spatial scalability, 720p → 1080p resolution	302
10.15	Three views of the same scene	302
10.16	Multiview video: view examples	303
10.17	Multiview video: views and frames	303
10.18	Interview prediction of key frames	304
10.19	Inter-view prediction of all frames	305
10.20	Overview of configurable video codec	306
10.21	Reconfigurable Video Coding scenario	307
10.22	Fully Configurable Video Coding framework	309

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000024

List of Tables

2.1	Video frame formats	17
2.2	ITU-R BT.601-5 Parameters	18
2.3	HD display formats	19
3.1	SAE of residual frame after motion compensation, 16 × 16 block size	37
3.2	Probability of occurrence of motion vectors in sequence 1	59
3.3	Huffman codes for sequence 1 motion vectors	60
3.4	Probability of occurrence of motion vectors in sequence 2	61
3.5	Huffman codes for sequence 2 motion vectors	62
3.6	MPEG-4 Visual Transform Coefficient (TCOEF) VLCs : partial, all codes < 9 bits	63
3.7	MPEG4 Motion Vector Difference (MVD) VLCs	65
3.8	Motion vectors, sequence 1: probabilities and sub-ranges	66
3.9	Residual luminance samples : top-right 8 × 8 block	75
3.10	DCT coefficients	75
3.11	Quantized coefficients	76
3.12	Variable length coding example	77
3.13	Rescaled coefficients	77
3.14	Decoded residual luminance samples	78
4.1	Overview of the H.264 standard document	92
5.1	Syntax examples: format	99
5.2	H.264 Syntax Sections	102
5.3	Selecting a reference picture to re-map	110
5.4	Initial reference picture list	110
5.5	Final reference picture list	111
5.6	Selected NAL unit types	114
5.7	Sequence Parameter Set example	115
5.8	Picture Parameter Set example	116
5.9	Slice types in H.264	117
5.10	Slice Header, IDR/Intra, Frame 0	118
5.11	Slice Header, Inter, Frame 1	118
5.12	Macroblock types	120
5.13	coded_block_pattern examples.	121
5.14	Type and prediction elements for B macroblocks, excluding Direct or 8 × 8 partitions	124

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000025

5.15	I macroblock, example 1	128
5.16	P macroblock, example 2	129
5.17	P macroblock, example 3	130
5.18	P macroblock, example 4	131
5.19	B macroblock, example 5	132
5.20	B macroblock, example 6	134
5.21	B macroblock, example 7	134
5.22	B macroblock, example 8	135
6.1	Intra prediction types	139
6.2	Choice of prediction mode, most probable mode = 1	149
6.3	Reference picture sources	151
6.4	Reference frames and motion vectors for P and B macroblocks	158
6.5	Layers in hierarchical GOP example	171
7.1	V_{i4} values, 4×4 blocks	190
7.2	Table v defined in H.264 standard	190
7.3	Estimated Q_{step} ($4\times$blocks) $= \mathbf{V}_{i4} / \mathbf{Si} \cdot 2^6$), element-by-element division	191
7.4	Tables v and m	194
7.5	8-point Integer Transform Basis C_{t8}	199
7.6	8-point Integer Transform Basis C_{18}	200
7.7	Estimated Q_{step} (8×8 blocks)	202
7.8	Scaling matrices	205
7.9	Exp-Golomb Codewords	208
7.10	Mappings to code_num	210
7.11	Choice of look-up table for coeff_token	212
7.12	Thresholds for determining whether to increment *suffixLength*	213
7.13	Binarization of MVD magnitude	219
7.14	Context models for bin 1	219
7.15	Context models	220
8.1	Selected formats, frame rates and levels	229
8.2	HRD example 1: access unit sizes	231
8.3	HRD example 2: frame sizes	233
8.4	HRD example 3: frame sizes	235
8.5	Macroblock allocation map types	239
8.6	Switching from stream A to stream B using SP-slices	242
8.7	SEI messages	249
8.8	Video Usability Information: selected parameters	250
9.1	JM software directory structure	257
9.2	Selected parameter sections in the JM encoder configuration file	260
9.3	Section of trace file showing Sequence Parameter Set	262
9.4	'Low Complexity' and 'Basic' configurations	266
9.5	'Basic' plus options	267
9.6	'Basic', 'Best Baseline', 'Medium'	268
9.7	Bitrate and delay constraints	274
10.1	Example: Quality Scalability	291
10.2	Example: Spatial + Temporal Scalability	292

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000026

1

Introduction

1.1 A change of scene

2000:

Most viewers receive analogue television via terrestrial, cable or satellite transmission.

VHS video tapes are the principal medium for recording and playing TV programs, movies, etc.

Cell phones are cell phones, i.e. a mobile handset can only be used to make calls or send SMS messages.

Internet connections are slow, primarily over telephone modems for home users.

Web pages are web pages, with static text, graphics and photos and not much else.

Video calling requires dedicated videoconferencing terminals and expensive leased lines. Video calling over the internet is possible but slow, unreliable and difficult to set up.

Consumer video cameras, camcorders, use tape media, principally analogue tape. Home-made videos generally stay within the home.

2010:

Most viewers receive digital television via terrestrial, cable, satellite or internet, with benefits such as a greater choice of channels, electronic programme guides and high definition services. Analogue TV has been switched off in many countries. Many TV programmes can be watched via the internet.

DVDs are the principal medium for playing pre-recorded movies and TV programs. Many alternatives exist, most of them digital, including internet movie downloading (legal and not-so-legal), hard-disk recording and playback and a

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000027

variety of digital media formats. High definition DVDs, Blu-Ray Disks, are increasing in popularity.

Cell phones function as cameras, web browsers, email clients, navigation systems, organizers and social networking devices. Occasionally they are used to make calls.

Home internet access speeds continue to get faster via broadband and mobile connections, enabling widespread use of video-based web applications.

Web pages are applications, movie players, games, shopping carts, bank tellers, social networks, etc, with content that changes dynamically.

Video calling over the internet is commonplace with applications such as Skype and iChat. Quality is still variable but continues to improve.

Consumer video cameras use hard disk or flash memory card media. Editing, uploading and internet sharing of home videos is widespread.

A whole range of illegal activities has been born – DVD piracy, movie sharing via the internet, recording and sharing of assaults, etc.

Video footage of breaking news items such as the Chilean earthquake is more likely to come from a cell phone than a TV camera.

All these changes in a ten-year period signify a small revolution in the way we create, share and watch moving images. Many factors have contributed to the shift towards digital video – commercial factors, legislation, social changes and technological advances. From the technology viewpoint, these factors include better communications infrastructure, with widespread, relatively inexpensive access to broadband networks, 3G mobile networks, cheap and effective wireless local networks and higher-capacity carrier transmission systems; increasingly sophisticated devices, with a bewildering array of capabilities packed into a lightweight cellular handset; and the development of easy-to-use applications for recording, editing, sharing and viewing video material. This book will focus on one technical aspect that is key to the widespread adoption of digital video technology – video compression.

Video compression or video encoding is the process of reducing the amount of data required to represent a digital video signal, prior to transmission or storage. The complementary operation, decompression or decoding, recovers a digital video signal from a compressed representation, prior to display. Digital video data tends to take up a large amount of storage or transmission capacity and so video encoding and decoding, or **video coding**, is essential for any application in which storage capacity or transmission bandwidth is constrained. Almost all consumer applications for digital video fall into this category, for example:

- Digital television broadcasting: TV programmes are coded prior to transmission over a limited-bandwidth terrestrial, satellite or cable channel (Figure 1.1).
- Internet video streaming: Video is coded and stored on a server. The coded video is transmitted (streamed) over the internet, decoded on a client and displayed (Figure 1.1).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000028

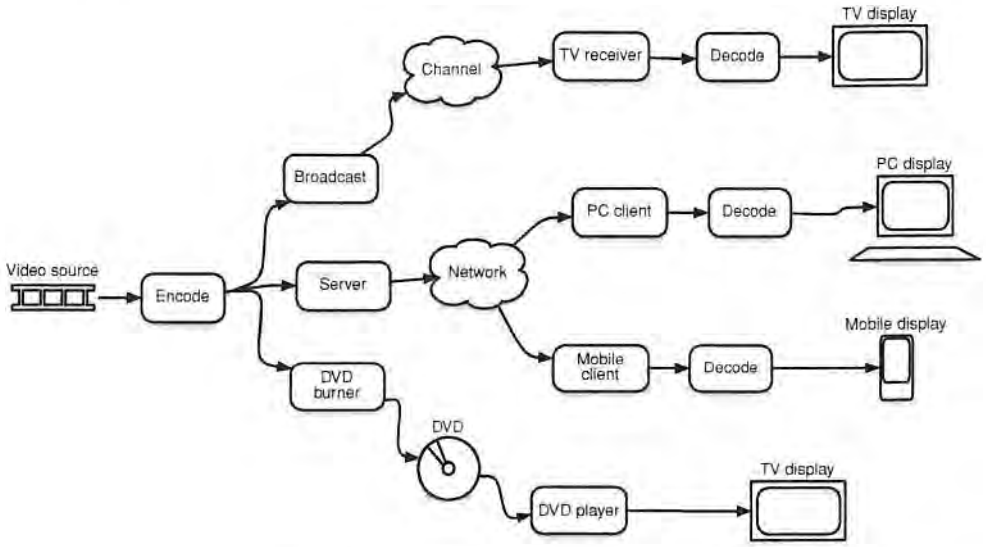


Figure 1.1 Video coding scenarios, one-way

- Mobile video streaming: As above, but the coded video is transmitted over a mobile network such as GPRS or 3G (Figure 1.1).
- DVD video: Source video is coded and stored on a DVD or other storage medium. A DVD player reads the disk and decodes video for display (Figure 1.1).
- Video calling: Each participant includes an encoder and a decoder (Figure 1.2). Video from a camera is encoded and transmitted across a network, decoded and displayed. This occurs in two directions simultaneously.

Each of these examples includes an encoder, which compresses or encodes an input video signal into a coded bitstream, and a decoder, which decompresses or decodes the coded bitstream to produce an output video signal. The encoder or decoder is often built in to a device such as a video camera or a DVD player.

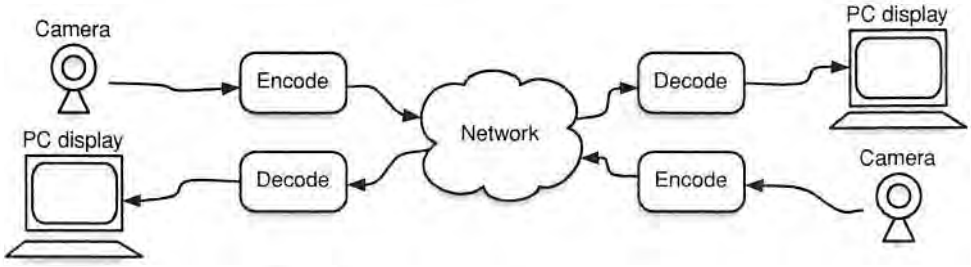


Figure 1.2 Video coding scenario, two-way

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000029

1.2 Driving the change

The consumer applications discussed above represent very large markets. The revenues involved in digital TV broadcasting and DVD distribution are substantial. Effective video coding is an essential component of these applications and can make the difference between the success or failure of a business model. A TV broadcasting company that can pack a larger number of high-quality TV channels into the available transmission bandwidth has a market edge over its competitors. Consumers are increasingly discerning about the quality and performance of video-based products and there is therefore a strong incentive for continuous improvement in video coding technology. Even though processor speeds and network bandwidths continue to increase, a better video codec results in a better product and therefore a more competitive product. This drive to improve video compression technology has led to significant investment in video coding research and development over the last 15-20 years and to rapid, continuous advances in the state of the art.

1.3 The role of standards

Many different techniques for video coding have been proposed and researched. Hundreds of research papers are published each year describing new and innovative compression techniques. In contrast to this wide range of innovations, commercial video coding applications tend to use a limited number of standardized techniques for video compression. Standardized video coding formats have a number of potential benefits compared with non-standard, proprietary formats:

- Standards simplify inter-operability between encoders and decoders from different manufacturers. This is important in applications where each 'end' of the system may be produced by a different company, e.g. the company that records a DVD is typically not the same as the company that manufactures a DVD player.
- Standards make it possible to build platforms that incorporate video, in which many different applications such as video codecs, audio codecs, transport protocols, security and rights management, interact in well-defined and consistent ways.
- Many video coding techniques are patented and therefore there is a risk that a particular video codec implementation may infringe patent(s). The techniques and algorithms required to implement a standard are well-defined and the cost of licensing patents that cover these techniques, i.e. licensing the right to use the technology embodied in the patents, can be clearly defined.

Despite recent debates about the benefits of royalty-free codecs versus industry standard video codecs [i], video coding standards are very important to a number of major industries. With the ubiquitous presence of technologies such as DVD/Blu-Ray, digital television, internet video and mobile video, the dominance of video coding standards is set to continue for some time to come.

1.4 Why H.264 Advanced Video Coding is important

This book is about a standard, jointly published by the International Telecommunications Union (ITU) and the International Standards Organisation (ISO) and known by several names:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000030

'H.264', 'MPEG-4 Part 10' and 'Advanced Video Coding'. The standard itself is a document over 550 pages long and filled with highly technical definitions and descriptions. Developed by a team consisting of hundreds of video compression experts, the Joint Video Team, a collaborative effort between the Moving Picture Experts Group (MPEG) and the Video Coding Experts Group (VCEG), this document is the culmination of many man-years' work. It is almost impossible to read and understand without an in-depth knowledge of video coding.

Why write a book about this document? Whilst the standard itself is arguably only accessible to an insider expert, H.264/AVC has huge significance to the broadcast, internet, consumer electronics, mobile and security industries, amongst others. H.264/AVC is the latest in a series of standards published by the ITU and ISO. It describes and defines a method of coding video that can give better performance than any of the preceding standards. H.264 makes it possible to compress video into a smaller space, which means that a compressed video clip takes up less transmission bandwidth and/or less storage space compared to older codecs. A combination of market expansion, technology advances and increased user expectation is driving demand for better, higher quality digital video. For example:

- TV companies are delivering more content in High Definition. Most new television sets can display HD pictures. Customers who pay a premium for High Definition content expect correspondingly high image quality.
- An ever-increasing army of users are uploading and downloading videos using sites such as YouTube. Viewers expect rapid download times and high resolution.
- Recording and sharing videos using mobile handsets is increasingly commonplace.
- Internet video calls, whilst still variable in quality, are easier to make and more widely used than ever.
- The original DVD-Video format, capable of supporting only a single movie in Standard Definition seems increasingly limited.

In each case, better video compression is the key to delivering more, higher-quality video in a cost effective way. H.264 compression makes it possible to transmit HD television over a limited-capacity broadcast channel, to record hours of video on a Flash memory card and to deliver massive numbers of video streams over an already busy internet.

The benefits of H.264/AVC come at a price. The standard is complex and therefore challenging to the engineer or designer who has to develop, program or interface with an H.264 codec. H.264 has more options and parameters – more 'control knobs' – than any previous standard codec. Getting the controls and parameters 'right' for a particular application is not an easy task. Get it right and H.264 will deliver high compression performance; get it wrong and the result is poor-quality pictures and/or poor bandwidth efficiency. Computationally expensive, an H.264 coder can lead to slow coding and decoding times or rapid battery drain on handheld devices. Finally, H.264/AVC, whilst a published industry standard, is not free to use. Commercial implementations are subject to licence fees and the intellectual property position in itself is complicated.

1.5 About this book

The aim of this book is to de-mystify H.264 and its complexities. H.264/AVC will be a key component of the digital media industry for some time to come. A better understanding of

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000031

the technology behind the standard and of the inter-relationships of its many component parts should make it possible to get the most out of this powerful tool.

This book is organized as follows.

Chapter 2 explains the concepts of digital video and covers source formats and visual quality measures.

Chapter 3 introduces video compression and the functions found in a typical video codec, such as H.264/AVC and other block-based video compression codecs.

Chapter 4 gives a high-level overview of H.264/AVC at a relatively non-technical level.

Chapters 5, 6 and 7 cover the standard itself in detail. **Chapter** 5 deals with the H.264/AVC syntax, i.e. the construction of an H.264 bitstream) including picture formats and picture management. **Chapter 6** describes the prediction methods supported by the standard, intra and inter prediction. **Chapter 7** explains the residual coding processes, i.e. transform and quantization and symbol coding.

Chapter 8 deals with issues closely related to the main standard – storage and network transport of H.264 data, conformance or how to ensure compatibility with H.264 and licensing, including the background and details of the intellectual property licence associated with H.264 implementations.

Chapter 9 examines the implementation and performance of H.264. It explains how to experiment with H.264, the effect of H.264 parameters on performance, implementation challenges and performance optimization.

Chapter 10 covers extensions to H.264/AVC, in particular the Scalable and Multiview Video Coding extensions that have been published since the completion of the H.264 standard. It examines possible future developments, including Reconfigurable Video Coding, a more flexible way of specifying and implementing video codecs, and possible successors to H.264, currently being examined by the standards groups.

Readers of my earlier book, "H.264 and MPEG-4 Video Compression", may be interested to know that Chapters 4–10 are largely or completely new material.

1.6 Reference

i. Ian Hickson, 'Codecs for <audio> and <video>', HTML5 specification discussion, http://lists.whatwg.org/htdig.cgi/whatwg-whatwg.org/2009-June/020620.html, accessed August 2009.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000032

2

Video formats and quality

2.1 Introduction

Video coding is the process of compressing and decompressing a digital video signal. This chapter examines the structure and characteristics of digital images and video signals and introduces concepts such as sampling formats and quality metrics. Digital video is a representation of a natural or real-world visual scene, sampled spatially and temporally. A scene is typically sampled at a point in time to produce a frame, which represents the complete visual scene at that point in time, or a field, which typically consists of odd- or even-numbered lines of spatial samples. Sampling is repeated at intervals (e.g. 1/25 or 1/30 second intervals) to produce a moving video signal. Three components or sets of samples are typically required to represent a scene in colour. Popular formats for representing video in digital form include the ITU-R 601 standard, High Definition formats and a set of 'intermediate formats'. The accuracy of a reproduction of a visual scene must be measured to determine the performance of a visual communication system, a notoriously difficult and inexact process. Subjective measurements are time consuming and prone to variations in the response of human viewers. Objective or automatic measurements are easier to implement but as yet do not accurately match the behaviour of a human observer.

2.2 Natural video scenes

A 'real world' or natural video scene is typically composed of multiple objects each with their own characteristic shape, depth, texture and illumination. The colour and brightness of a natural video scene changes with varying degrees of smoothness throughout the scene, i.e. it has continuous tone. Characteristics of a typical natural video scene (Figure 2.1) that are relevant for video processing and compression include spatial characteristics such as texture variation within scene, number and shape of objects, colour, etc. and temporal characteristics such as object motion, changes in illumination and movement of the camera or viewpoint.

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000033



Figure 2.1 Still image from natural video scene

2.3 Capture

A natural visual scene is spatially and temporally continuous. Representing a visual scene in digital form involves sampling the real scene spatially, usually on a rectangular grid in the video image plane, and temporally, as a series of still frames or components of frames sampled at regular intervals in time (Figure 2.2). Digital video is the representation of a sampled video scene in digital form. Each spatio-temporal sample, a picture element or pixel, is represented as one or more numbers that describes the brightness or luminance and the colour of the sample.

To obtain a 2-D sampled image, a camera focuses a 2-D projection of the video scene onto a sensor, such as an array of Charge Coupled Devices (CCDs). In the case of colour image capture, each colour component is separately filtered and projected onto a CCD array (see section 2.4).

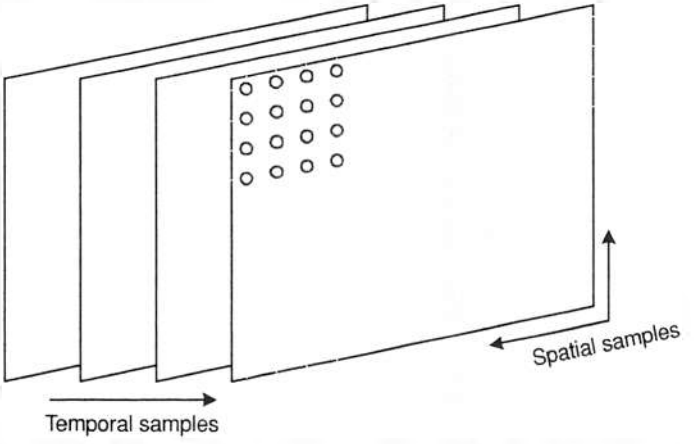


Figure 2.2 Spatial and temporal sampling of a video sequence

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000034



Figure 2.3 Image with two sampling grids

2.3.1 *Spatial sampling*

The output of a CCD array is an analogue video signal, a varying electrical signal that represents a video image. Sampling the signal at a point in time produces a sampled image or frame that has defined values at a set of sampling points. The most common format for a sampled image is a rectangle with the sampling points positioned on a square or rectangular grid. Figure 2.3 shows a continuous-tone frame with two different sampling grids superimposed upon it. Sampling occurs at each of the intersection points on the grid and the sampled image may be reconstructed by representing each sample as a square picture element or pixel. The number of sampling points influences the visual quality of the image. Choosing a 'coarse' sampling grid, the black grid in Figure 2.3, produces a low-resolution sampled image (Figure 2.4) whilst increasing the number of sampling points slightly, the grey grid in Figure 2.3, increases the resolution of the sampled image (Figure 2.5).

2.3.2 *Temporal sampling*

A moving video image is formed by taking a rectangular 'snapshot' of the signal at periodic time intervals. Playing back the series of snapshots or frames produces the appearance of motion. A higher temporal sampling rate or frame rate gives apparently smoother motion in the video scene but requires more samples to be captured and stored. Frame rates below 10 frames per second may be used for very low bit-rate video communications, because the amount of data is relatively small, but motion is clearly jerky and unnatural at this rate. Between 10–20 frames per second is more typical for low bit-rate video communications; the image is smoother but jerky motion may be visible in fast-moving parts of the sequence. Temporal

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000035

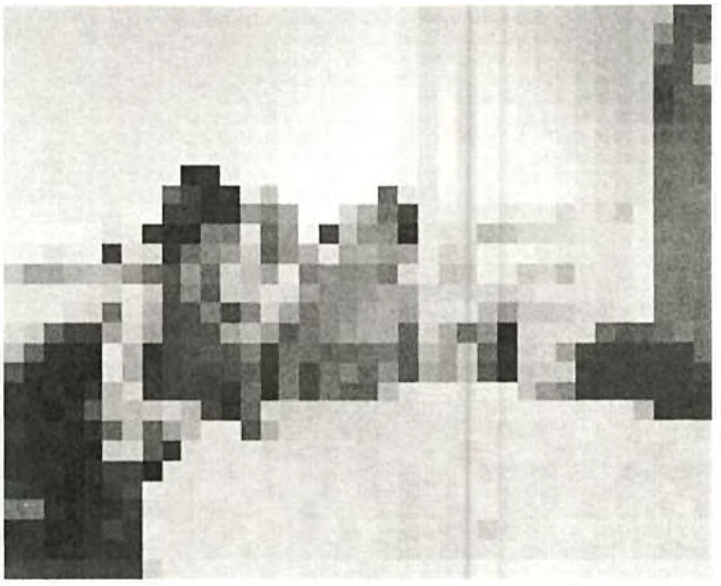


Figure 2.4 Image sampled at coarse resolution (black sampling grid)

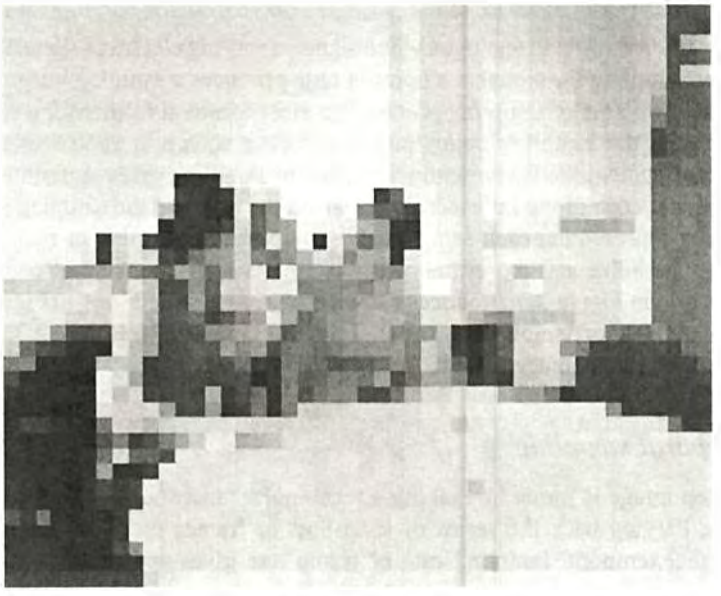


Figure 2.5 Image sampled at finer resolution (grey sampling grid)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000036

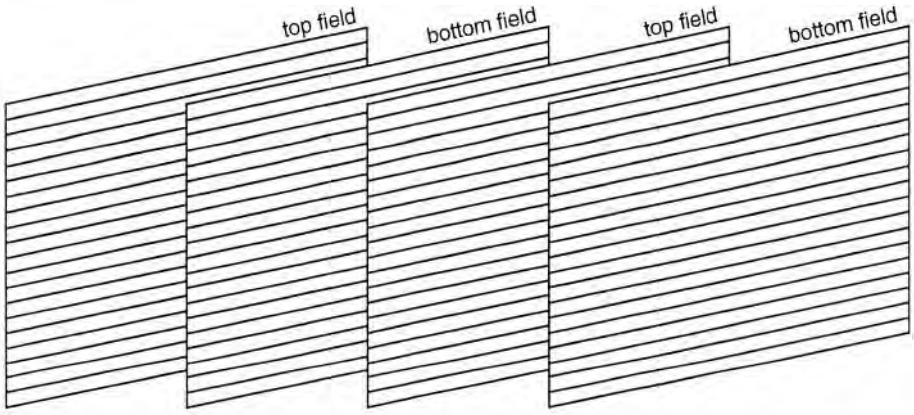


Figure 2.6 Interlaced video sequence

sampling at 25 or 30 complete frames per second is the norm for Standard Definition television pictures, with interlacing to improve the appearance of motion, see below; 50 or 60 frames per second produces very smooth apparent motion at the expense of a very high data rate.

2.3.3 Frames and fields

A video signal may be sampled as a series of complete frames, **progressive** sampling, or as a sequence of interlaced fields, **interlaced** sampling. In an interlaced video sequence, half of the data in a frame, one field, is typically sampled at each temporal sampling interval. A field may consist of either the odd-numbered or even-numbered lines within a complete video frame and an interlaced video sequence (Figure 2.6) typically contains a series of fields, each representing half of the information in a complete video frame, illustrated in Figure 2.7 and Figure 2.8. The



Figure 2.7 Top field

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000037



Figure 2.8 Bottom field

advantage of this sampling method is that it is possible to send twice as many fields per second as the number of frames in an equivalent progressive sequence with the same data rate, giving the appearance of smoother motion. For example, a PAL video sequence consists of 50 fields per second and when played back, motion appears smoother than in an equivalent progressive video sequence containing 25 frames per second. Increasingly, video content may be captured and/or displayed in progressive format. When video is captured in one format (e.g. interlaced) and displayed in another (e.g. progressive), it is necessary to convert between formats.

2.4 Colour spaces

Most digital video applications rely on the display of colour video and so need a mechanism to capture and represent colour information. A monochrome image (Figure 2.1) requires just one number to indicate the brightness or luminance of each spatial sample. Colour images, on the other hand, require at least three numbers per pixel position to accurately represent colour. The method chosen to represent brightness, luminance or luma and colour is described as a **colour space**.

2.4.1 RGB

In the RGB colour space, a colour image sample is represented with three numbers that indicate the relative proportions of Red, Green and Blue, the three additive primary colours of light. Combining red, green and blue in varying proportions can create any colour. Figure 2.9 shows the red, green and blue components of a colour image: the red component consists of all the red samples, the green component contains all the green samples and the blue component contains the blue samples. The person on the right is wearing a blue sweater and so this appears 'brighter' in the blue component, whereas the red waistcoat of the figure on the left

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000038



Figure 2.9 Red. Green and Blue components of colour image

appears brighter in the red component. The RGB colour space is well suited to capture and display of colour images. Capturing an RGB image involves filtering out the red, green and blue components of the scene and capturing each with a separate sensor array. Colour displays show an RGB image by separately illuminating the red, green and blue components of each pixel according to the intensity of each component. From a normal viewing distance, the separate components merge to give the appearance of 'true' colour.

2.4.2 *YCrCb*

The human visual system (HVS) is less sensitive to colour than to luminance. In the RGB colour space the three colours are equally important and so are usually all stored at the same resolution but it is possible to represent a colour image more efficiently by separating the luminance from the colour information and representing luma with a higher resolution than colour.

The Y:Cr:Cb colour space is a popular way of efficiently representing colour images. Y is the luminance component and can be calculated as a weighted average of R, G and B:

$$Y = k_r R + k_g G + k_b B \tag{2.1}$$

where k are weighting factors.

The colour information can be represented as **colour difference** (chrominance or chroma) components, where each chrominance component is the difference between R, G or B and the luminance Y:

$$\begin{aligned} Cr &= R - Y \\ Cb &= B - Y \\ Cg &= G - Y \end{aligned} \tag{2.2}$$

The complete description of a colour image is given by Y, the luminance component, and three colour differences Cr, Cb and Cg that represent the difference between the colour intensity and the mean luminance of each image sample. Figure 2.10 shows the red, green and blue chroma components corresponding to the RGB components of Figure 2.9. Here, mid-grey is zero difference, light grey is a positive difference and dark grey is a negative difference. The chroma components only have significant values where there is a large difference between the

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000039



Figure 2.10 Cr, Cg and Cb components

colour component and the luma image (Figure 2.1). Note the strong blue and red difference components.

So far, this representation has little obvious merit since we now have four components instead of the three in RGB. However, Cr+Cb+Cg is a constant and so only two of the three chrominance components need to be stored or transmitted since the third component can always be calculated from the other two. In the Y:Cr:Cb colour space, only the luma (Y) and red and blue chroma (Cr, Cb) are transmitted. Y:Cr:Cb has an important advantage over RGB, in that the Cr and Cb components may be represented with a **lower resolution** than Y because the HVS is less sensitive to colour than luminance. This reduces the amount of data required to represent the chrominance components without having an obvious effect on visual quality. To the casual observer, there is no obvious difference between an RGB image and a Y:Cr:Cb image with reduced chrominance resolution. Representing chroma with a lower resolution than luma in this way is a simple but effective form of image compression.

An RGB image may be converted to Y:Cr:Cb after capture in order to reduce storage and/or transmission requirements. Before displaying the image, it is usually necessary to convert back to RGB. The equations for converting an RGB image to and from Y:Cr:Cb colour space and vice versa are given in (2.3 and 2.4). Note that G can be extracted from the Y:Cr:Cb representation by subtracting Cr and Cb from Y, demonstrating that it is not necessary to store or transmit a Cg component.

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.564(B - Y) \\ Cr &= 0.713(R - Y) \end{aligned} \tag{2.3}$$

$$\begin{aligned} R &= Y + 1.402Cr \\ G &= Y - 0.344Cb - 0.714Cr \\ B &= Y + 1.772Cb \end{aligned} \tag{2.4}$$

2.4.3 *YCrCb sampling formats*

Figure 2.11 shows three sampling patterns for Y, Cr and Cb that are supported by H.264/AVC. 4:4:4sampling means that the three components (Y:Cr:Cb) have the same resolution and hence

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000040

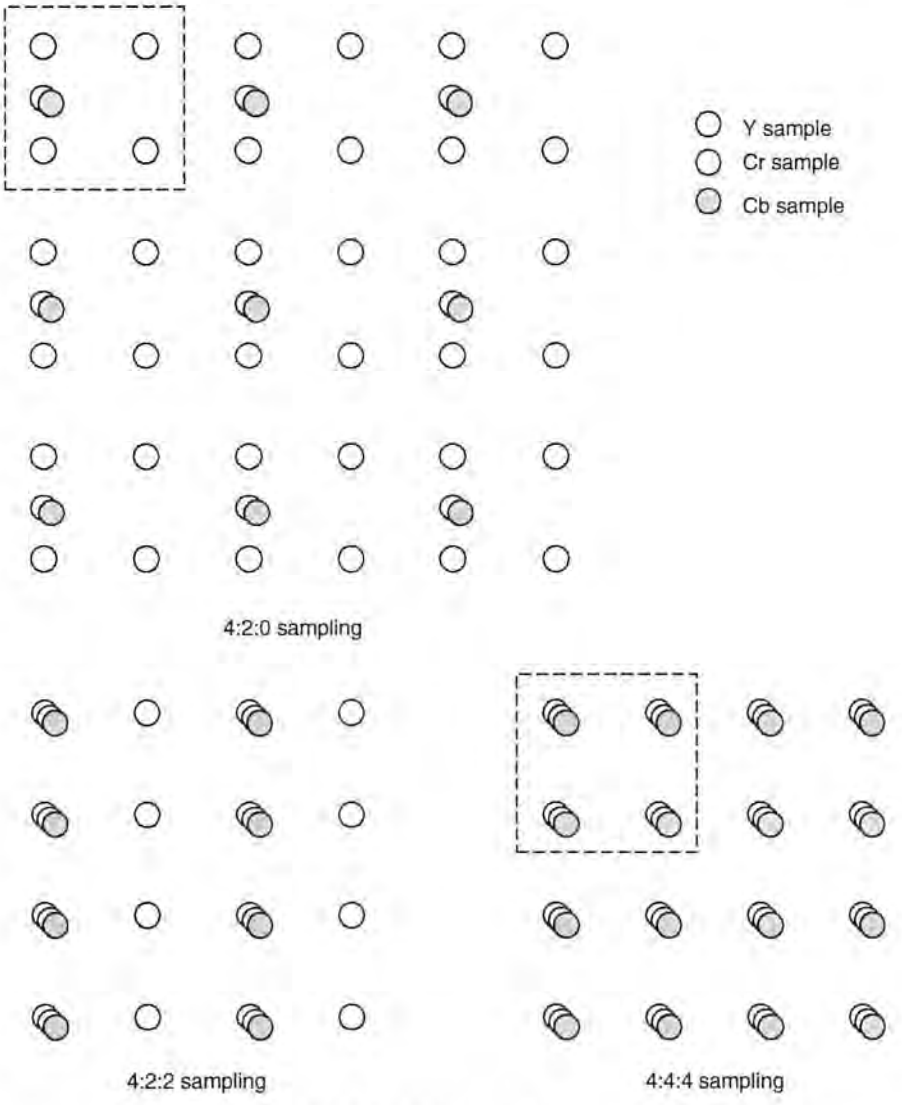


Figure 2.11 4:2:0, 4:2:2 and 4:4:4 sampling patterns (progressive)

a sample of each component exists at every pixel position. The numbers indicate the relative sampling rate of each component in the **horizontal** direction, i.e. for every 4 luminance samples there are 4 Cr and 4 Cb samples. 4:4:4 sampling preserves the full fidelity of the chrominance components. In 4:2:2 sampling, sometimes referred to as YUY2, the chrominance components have the same vertical resolution as the luma but half the horizontal resolution. The numbers 4:2:2 mean that for every 4 luminance samples in the horizontal direction there are 2 Cr and 2 Cb samples. 4:2:2 video is used for high-quality colour reproduction.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000041

In the popular 4:2:0 sampling format ('YV12'), Cr and Cb each have half the horizontal and vertical resolution of Y. The term '4:2:0' is rather confusing because the numbers do not actually have a logical interpretation and appear to have been chosen historically as a 'code' to identify this particular sampling pattern and to differentiate it from 4:4:4 and 4:2:2. 4:2:0 sampling is widely used for consumer applications such as video conferencing, digital television and digital versatile disk (DVD) storage. Because each colour difference component contains ¼ of the number of samples in the Y component, 4:2:0 Y:Cr:Cb video requires exactly ½ as many samples as 4:4:4 or R:G:B video.

Example

Image resolution: 720 × 576 pixels
Y resolution: 720 × 576 samples, each represented with 8 bits
4:4:4 Cr, Cb resolution: 720 × 576 samples, each 8 bits
Total number of bits: 720 × 576 × 8 × 3 = 9953280 bits
4:2:0 Cr, Cb resolution: 360 × 288 samples, each 8 bits
Total number of bits: (720 × 576 × 8) + (360 × 288 × 8 × 2) = 4976640 bits
The 4:2:0 version requires half as many bits as the 4:4:4 version.

4:2:0 sampling is sometimes described as '12 bits per pixel'. The reason for this can be seen by examining a group of 4 pixels, enclosed in dotted lines in Figure 2.11. Using 4:4:4 sampling, a total of 12 samples are required, 4 each of Y, Cr and Cb, requiring a total of 12 × 8 = 96 bits, an average of 96/4 = 24 bits per pixel. Using 4:2:0 sampling, only 6 samples are required, 4 Y and one each of Cr, Cb, requiring a total of 6 × 8 = 48 bits, an average of 48/4 = 12 bits per pixel.

In a 4:2:0 interlaced video sequence, the Y, Cr and Cb samples corresponding to a complete video frame are allocated to two fields.

Figure 2.12 shows the method of allocating Y, Cr and Cb samples to a pair of fields adopted in H.264. It is clear from this figure that the total number of samples in a pair of fields is the same as the number of samples in an equivalent progressive frame.

2.5 Video formats

2.5.1 Intermediate formats

The video compression algorithms described in this book can compress a wide variety of video frame formats. In practice, it is common to capture or convert to one of a set of 'intermediate formats' prior to compression and transmission. The Common Intermediate Format (CIF) is the basis for a popular set of formats listed in Table 2.1. Figure 2.13 shows the luma component of a video frame sampled at a range of resolutions, from 4CIF down to Sub-QCIF. The choice of frame resolution depends on the application and available storage or transmission capacity. For example, 4CIF is appropriate for standard-definition television and DVD-video; CIF and QCIF are popular for videoconferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications where the display resolution and the bitrate are limited.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000042

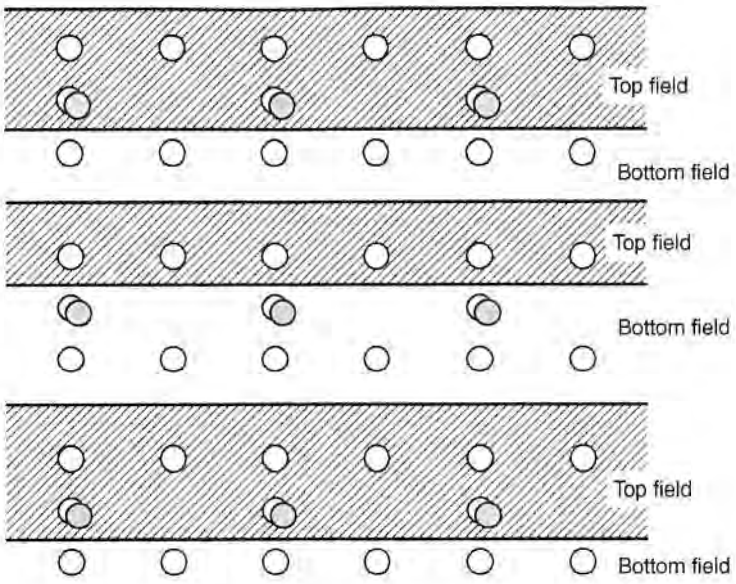


Figure 2.12 Allocation of 4:2:0 samples to top and bottom fields

Table 2.1 lists the number of bits required to represent 1 uncompressed frame in each format, assuming 4:2:0 sampling and 8 bits per luma and chroma sample.

2.5.2 *Standard Definition*

A widely-used format for digitally coding video signals for television production is ITU-R Recommendation BT.601-5 [i]. Note that the term 'coding' in the Recommendation title means conversion to digital format and does not imply compression. The luminance component of the video signal is sampled at 13.5MHz and the chrominance at 6.75MHz to produce a 4:2:2 Y:Cr:Cb component signal. The parameters of the sampled digital signal depend on the video frame rate, 30Hz for an NTSC signal and 25Hz for a PAL/SECAM signal, and are shown in Table 2.2. The higher 30Hz frame rate of NTSC is compensated for by a lower spatial resolution so that the total bit rate is the same in each case, 216Mbps. The actual area shown

Table 2.1 Video frame formats

Format	Luminance resolution (horiz. × vert.)	Bits per frame (4:2:0, 8 bits per sample)
Sub-QCIF	128 × 96	147456
Quarter CIF (QCIF)	176 × 144	304128
CIF	352 × 288	1216512
4CIF	704 × 576	4866048

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000043



Figure 2.13 Video frame sampled at range of resolutions

on the display, the **active area**, is smaller than the total because it excludes horizontal and vertical blanking intervals that exist 'outside' the edges of the frame.

Each sample has a possible range of 0 to 255. Levels of 0 and 255 are reserved for synchronization and the active luminance signal is restricted to a range of 16 (black) to 235 (white).

2.5.3 High Definition

Several High Definition (HD) video formats exist [ii]. The most widely used television display formats are listed in Table 2.3 and shown graphically in Figure 2.14. Note that equivalent versions are defined for base frame rates of 30Hz, rather than the European 25Hz base frame

Table 2.2 ITU-R BT.601-5 Parameters

	30Hz frame rate	25Hz frame rate
Fields per second	60	50
Lines per complete frame	525	625
Luminance samples per line	858	864
Chrominance samples per line	429	432
Bits per sample	8	8
Total bit rate	216 Mbps	216 Mbps
Active lines per frame	480	576
Active samples per line (Y)	720	720
Active samples per line (Cr,Cb)	360	360

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000044

Table 2.3 HD display formats

Format	Progressive or Interlaced	Horizontal pixels	Vertical pixels	Frames or fields per second
720p	Progressive	1280	720	25 frames
1080i	Interlaced	1920	1080	50 fields
1080p	Progressive	1920	1080	25 frames

rate. It is clear that HD formats require an even larger uncompressed storage or transmission rate than SD formats. SD video has $(720 \times 576 \times 25) = 10368000$ displayed pixels per second. 720p HD video has $(1280 \times 720 \times 25) = 23040000$ displayed pixels per second and at the top end, 1080p HD has $(1920 \times 1080 \times 25) = 51840000$ displayed pixels per second. The very large storage or transmission capacity required for uncompressed video at these resolutions means it is essential to compress video for practical applications.

2.6 Quality

In order to specify, evaluate and compare video communication systems it is necessary to determine the quality of the video images displayed to the viewer. Measuring visual quality is a difficult and often imprecise art because there are so many factors that can affect the results. Visual quality is inherently **subjective** and is therefore influenced by many subjective factors that make it difficult to obtain a completely accurate measure of quality. For example, a viewer's opinion of visual quality can depend very much on the task at hand, such as passively watching a DVD movie, actively participating in a videoconference or trying to identify a person in

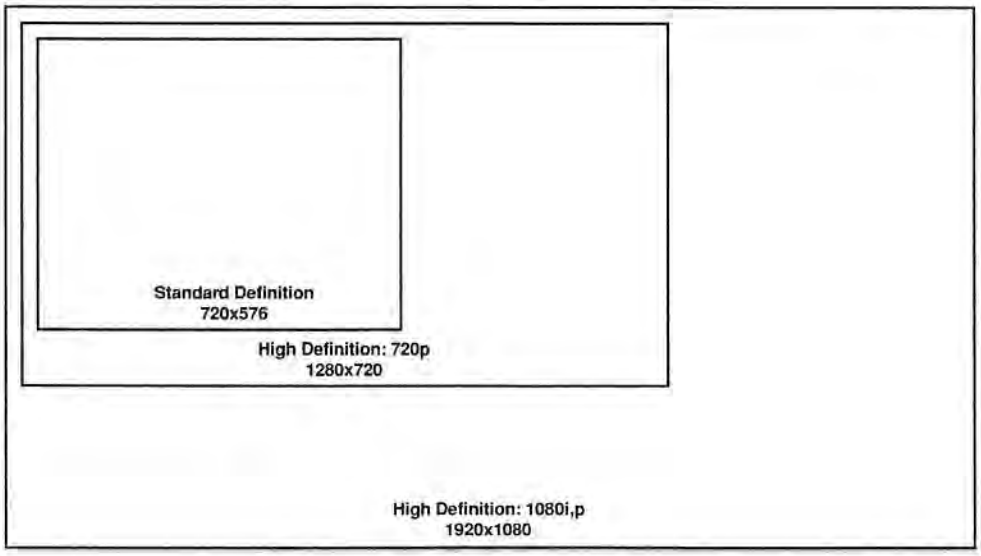


Figure 2.14 SD and HD formats

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000045

a surveillance video scene. Measuring visual quality using **objective** criteria gives accurate, repeatable results but as yet there are no objective measurement systems that completely reproduce the subjective experience of a human observer watching a video display.

2.6.1 Subjective quality measurement

2.6.1.1 Factors influencing subjective quality

Our perception of a visual scene is formed by a complex interaction between the components of the Human Visual System (HVS), the eye and the brain. The perception of visual quality is influenced by spatial fidelity, i.e. how clearly parts of the scene can be seen, whether there is any obvious distortion, and temporal fidelity, i.e. whether motion appears natural and 'smooth'. However, a viewer's opinion of 'quality' is also affected by other factors such as the viewing environment, the observer's state of mind and the extent to which the observer interacts with the visual scene. A user carrying out a specific task that requires concentration on part of a visual scene will have a quite different requirement for quality than a user who is passively watching a movie. It has been shown that a viewer's opinion of visual quality is measurably higher if the viewing environment is comfortable and non-distracting, regardless of the quality of the visual image itself.

Other important influences on perceived quality include visual attention, i.e. the way an observer perceives a scene by fixating on a sequence of points in the image rather than by taking in everything simultaneously [iii], and the so-called 'recency effect', i.e. our opinion of a visual sequence is more heavily influenced by recently-viewed material than older video material [iv, v]. All of these factors make it very difficult to accurately and quantitatively measure visual quality.

2.6.1.2 ITU-R 500

Several test procedures for subjective quality evaluation are defined in ITU-R Recommendation BT.500-11 [vi]. A widely-used procedure from the standard is the Double Stimulus Continuous Quality Scale (DSCQS) method in which an assessor is presented with a pair of images or short video sequences A and B, one after the other, and is asked to give A and B a 'quality score' by marking on a continuous line with five intervals ranging from 'Excellent' to 'Bad'. In a typical test session, the assessor is shown a series of pairs of sequences and is asked to grade each pair. Within each pair of sequences, one is an unimpaired 'reference' sequence and the other is the same sequence, modified by a system or process under test. Figure 2.15 shows an experimental set-up appropriate for the testing of a video CODEC in which the original sequence is compared with the same sequence after encoding and decoding. The selection of which sequence is 'A' and which is 'B' is randomized.

The order of the two sequences, original and 'impaired', is randomized during the test session so that the assessor does not know which is the original and which is the impaired sequence. This helps prevent the assessor from pre-judging the impaired sequence compared with the reference sequence. At the end of the session, the scores are converted to a normalized range and the end result is a score, sometimes described as a 'mean opinion score' (MOS) that indicates the **relative** quality of the impaired and reference sequences.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000046

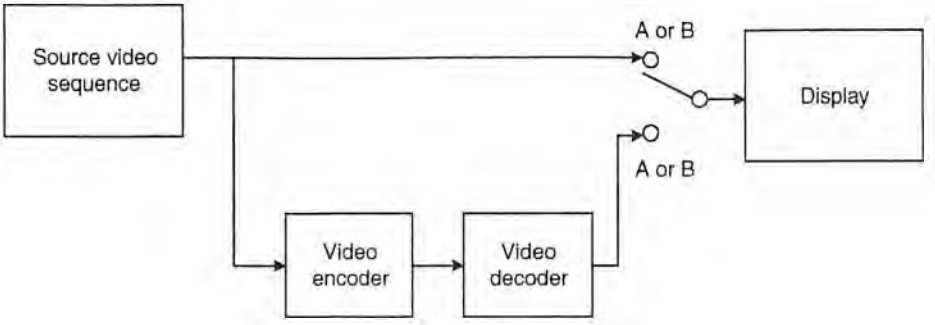


Figure 2.15 DSCQS testing system

Tests such as DSCQS are accepted as realistic measures of subjective visual quality. However, this type of test suffers from practical problems. The results can vary significantly depending on the assessor and also on the video sequence under test. This variation is compensated for by repeating the test with several sequences and several assessors. An 'expert' assessor who is familiar with the nature of video compression distortions or 'artefacts' may give a biased score and it is recommended to use 'non-expert' assessors. This means that a large pool of assessors is required because a non-expert assessor will quickly learn to recognize characteristic artefacts in the video sequences and so will become 'expert'. These factors make it expensive and time consuming to carry out the DSCQS tests thoroughly.

2.6.2 *Objective quality measurement*

The complexity and cost of subjective quality measurement make it attractive to be able to measure quality automatically using an algorithm. Developers of video compression and video processing systems rely heavily on so-called objective (algorithmic) quality measures. The most widely used measure is Peak Signal to Noise Ratio (PSNR) but the limitations of this metric have led to many efforts to develop more sophisticated measures that approximate the response of 'real' human observers.

2.6.2.1 PSNR

Peak Signal to Noise Ratio (PSNR) (2.5) is measured on a logarithmic scale and depends on the mean squared error (MSE) between an original and an impaired image or video frame, relative to $(2^n - 1)^2$, the square of the highest-possible signal value in the image, where n is the number of bits per image sample.

$$PSNR_{dB} = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \tag{2.5}$$

PSNR can be calculated easily and quickly and is therefore a very popular quality measure, widely used to compare the 'quality' of compressed and decompressed video images. Figure 2.16 shows a close-up of three images: the first image (a) is the original and (b) and (c) are degraded (blurred) versions of the original image. Image (b) has a measured PSNR of 30.6dB whilst image (c) has a PSNR of 28.3dB, reflecting the poorer image quality.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000047



Figure 2.16 PSNR examples: (a) Original; (b) 30.6dB; (c) 28.3dB

The PSNR metric suffers from a number of limitations. PSNR requires an unimpaired original image for comparison but this may not be available in every case and it may not be easy to verify that an 'original' image has perfect fidelity. PSNR does not correlate well with subjective video quality measures such as ITU-R 500. For a given image or image sequence, high PSNR usually indicates high quality and low PSNR usually indicates low quality. However, a particular value of PSNR does not necessarily equate to an 'absolute' subjective quality. For example, Figure 2.17 shows a distorted version of the original image from Figure 2.16 in which only the background of the image has been blurred. This image has a PSNR of 27.7dB relative to the original. Most viewers would rate this image as significantly better than image (c) in Figure 2.16 because the face is clearer, in opposition to the PSNR rating. This example shows that PSNR ratings do not necessarily correlate with 'true' subjective



Figure 2.17 Image with blurred background (PSNR = 27.7dB)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000048

quality. In this case, a human observer gives a higher importance to the face region and so is particularly sensitive to distortion in this area.

2.6.2.2 Other objective quality metrics

Because of the limitations of crude metrics such as PSNR, there has been a lot of work in recent years to try to develop a more sophisticated objective test that more closely approaches subjective test results. Many different approaches have been proposed. It has proved difficult to establish reliable objective video quality metrics that accurately predict the results of subjective tests. Recent proposals have included Just Noticeable Difference (JND) [vii], Digital Video Quality (DVQ) [viii], Structual SIMilarity index (SSIM) [ix], PSNRplus [x] and Predicted Mean Opinion Score (MOSp) [xi]. These metrics have varying degrees of success in predicting subjective test scores, with reported correlations of between 70 per cent and 90 per cent between each objective metric and measured subjective quality scores.

The ITU-T Video Quality Experts Group (VQEG) aims to develop industry standards related to video and multimedia quality assessment. VQEG has developed Recommendation J.247, which covers 'full reference' video quality measurement, i.e. quality metrics that require access to an original, uncompressed version of a video signal [xii]. The Recommendation lists four objective quality metrics that 'can be recommended by ITU-T at this time':

A: NTT Full Reference Method.

B: OPTICOM Perceptual Video Quality Method.

C: Psytechnics Full Reference Method.

D: Yonsei Full Reference Method.

The general approach of these methods is as follows. First, the original (reference) and test (degraded) video sequences are aligned spatially and temporally. Next, a series of degradation parameters are calculated. Each of the four metrics calculates a different set of parameters such as blurring, edges introduced by compression, blockiness, etc. Finally, these parameters are combined to produce a single number that is an estimate of subjective quality. The correlation between subjective and estimated quality is reported as ranging from 77 per cent to 84 per cent in the Recommendation, indicating that there is still scope for developing better Full Reference objective quality metrics.

An even more challenging task is to measure or estimate quality when a full reference, an unimpaired copy of the original video, is not available. This is the case in many practical applications. For example, the original source may not be available in the case of user-generated video content, or it may be desirable to monitor quality at the receiver, for example in a customer's digital television receiver, where there is no access to the original video. In these situations, No Reference (NR) or Reduced Reference (RR) quality estimation is required. No Reference metrics attempt to estimate subjective quality based only on characteristics of the decoded video clip. This is a difficult task but some success has been reported using methods such as modelling typical image/video compression artefacts [xiii, xiv]. Reduced Reference metrics calculate a quality 'signature', typically a low-bitrate side signal, which is

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000049

communicated to the decoder. A quality estimate is formed by processing the decoded video clip together with the side information [xv].

2.7 Summary

Sampling analogue video produces a digital video signal, which has the advantages of accuracy, quality and compatibility with digital media and transmission but which typically occupies a prohibitively large bitrate. Issues inherent in digital video systems include spatial and temporal resolution, colour representation and the measurement of visual quality. The next chapter introduces the basic concepts of video compression, necessary to accommodate digital video signals on practical storage and transmission media.

2.8 References

i. ITU-R Recommendation BT.601-6, 'Studio encoding parameters of digital television for standard 4:3 and widescreen 16:9 aspect ratios', ITU-R, 2007.

ii. 'High Definition Image Formats for Television Broadcasting', European Broadcasting Union (EBU) Technical Report 3299, Geneva, 2004.

iii. J. Findlay and I. Gilchrist, *Active Vision: the psychology of looking and seeing*. Oxford University Press, 2003.

iv. N. Wade and M. Swanston, (2001) *Visual Perception: An Introduction*. 2nd edition. London: Psychology Press.

v. R. Aldridge, J. Davidoff, D. Hands, M. Ghanbari and D. E. Pearson, 'Recency effect in the subjective assessment of digitally coded television pictures', proc. Fifth International Conference on Image Processing and its applications, Heriot-Watt University, Edinburgh, UK (July 1995).

vi. ITU-T Recommendation BT.500-11, 'Methodology for the subjective assessment of the quality of television pictures', ITU-T, 2002.

vii. J. Lubin and D. Fibush, 'Sarnoff JND vision model', T1A1.5 Working group Document, T1 Standards Committee, 1997.

viii. A.B. Watson, J. Hu and J.F. McGowan III, 'Digital video quality metric based on human vision', *Journal of Electronic imaging*, vol. 10, no.1, Jan 2001, pp. 20–29.

ix. Z. Wang and A.C. Bovik, 'A universal image quality index', IEEE Signal Proc. Letters, vol. 9, no. 3, Mar. 2002, pp. 81–84.

x. T. Oelbaum, K. Diepold and W. Zia, 'A generic method to increase the prediction accuracy of visual quality metrics', PCS 2007.

xi. A. Bhat, I.E. Richardson and C.S. Kannangara, 'A Novel Perceptual Quality Metric for Video Compression,' Proc. International Picture Coding Symposium 2009, Chicago, May 2009.

xii. ITU-T Recommendation J.247, 'Objective perceptual multimedia video quality measurement in the presence of a full reference', ITU-T, August 2008.

xiii. Z. Wang, H. Sheikh and A. Bovik, 'No-reference perceptual quality assessment of JPEG compressed images', Proc. International Conference on Image Processing, 2002.

xiv. R. Dosselmann and X. Yang, 'A prototype no-reference video quality system', Proc. Canadian Conference on Computer and Robot Vision, 2007.

xv. Z. Wang and E. Simoncelli, 'Reduced-reference image quality assessment using a wavelet-domain natural image statistic model', Proc. SPIE Human Vision and Electronic Imaging, Vol. 5666, 2005.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000050

3

Video coding concepts

3.1 Introduction

compress *vb.*: to squeeze together or compact into less space; condense

compression *noun*: the act of compression or the condition of being compressed

Compression is the act or process of compacting data into a smaller number of bits. Video compression (video coding) is the process of converting digital video into a format suitable for transmission or storage, whilst typically reducing the number of bits. 'Raw' or uncompressed digital video typically requires a large bitrate, approximately 216Mbits for 1 second of uncompressed Standard Definition video, see Chapter 2, and compression is necessary for practical storage and transmission of digital video.

Compression involves a complementary pair of systems, a compressor (encoder) and a decompressor (decoder). The encoder converts the source data into a compressed form occupying a reduced number of bits, prior to transmission or storage, and the decoder converts the compressed form back into a representation of the original video data. The encoder/decoder pair is often described as a **CODEC** (en**CO**der/**DEC**oder) (Figure 3.1).

Data compression is achieved by removing **redundancy,** i.e. components that are not necessary for faithful reproduction of the data. Many types of data contain **statistical** redundancy and can be effectively compressed using **lossless** compression, so that the reconstructed data at the output of the decoder is a perfect copy of the original data. Unfortunately, lossless compression of image and video information gives only a moderate amount of compression. The best that can be achieved with lossless image compression standards such as JPEG-LS [i] is a compression ratio of around 3–4 times. **Lossy** compression is necessary to achieve higher compression. In a lossy compression system, the decompressed data is not identical to the source data and much higher compression ratios can be achieved at the expense of a loss of visual quality. Lossy video compression systems are based on the principle of removing **subjective** redundancy, elements of the image or video sequence that can be removed without significantly affecting the viewer's perception of visual quality.

Most video coding methods exploit both **temporal** and **spatial** redundancy to achieve compression. In the temporal domain, there is usually a high correlation or similarity between

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000051

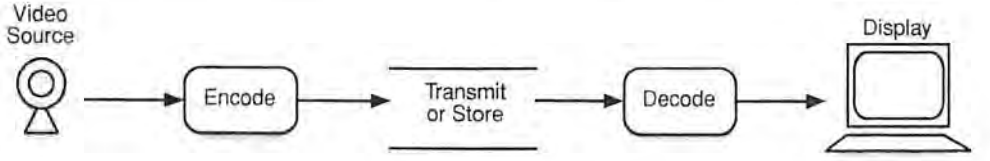


Figure 3.1 Encoder / Decoder

frames of video that were captured at around the same time. Temporally adjacent frames, i.e. successive frames in time order, are often highly correlated, especially if the temporal sampling rate or frame rate is high. In the spatial domain, there is usually a high correlation between pixels (samples) that are close to each other, i.e. the values of neighbouring samples are often very similar (Figure 3.2).

The H.264 Advanced Video Coding standard shares a number of common features with other popular compression formats such as MPEG-2 Video, MPEG-4 Visual, H.263, VC-1, etc. Each of these formats is based upon a CODEC 'model' that uses prediction and/or block-based motion compensation, transform, quantization and entropy coding. In this chapter we examine the main components of this model, starting with the prediction model, including intra prediction, motion estimation and compensation, and continuing with image transforms, quantization, predictive coding and entropy coding. The chapter concludes with a 'walk-through' of the basic model, following through the process of encoding and decoding a block of image samples.

3.2 Video CODEC

A video CODEC (Figure 3.3) encodes a source image or video sequence into a compressed form and decodes this to produce a copy or approximation of the source sequence. If the



Figure 3.2 Spatial and temporal correlation in a video sequence

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000052

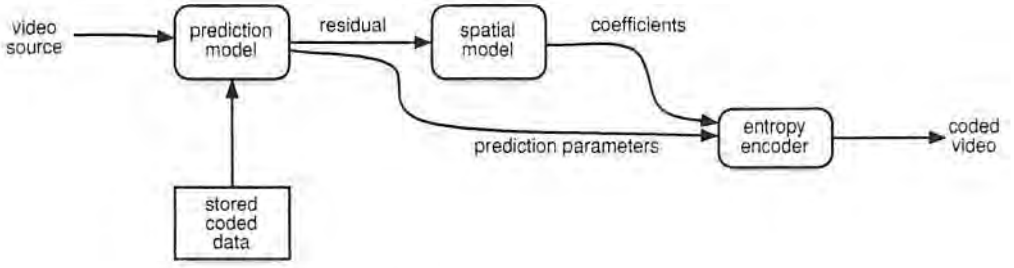


Figure 3.3 Video encoder block diagram

decoded video sequence is identical to the original, then the coding process is lossless; if the decoded sequence differs from the original, the process is lossy.

The CODEC represents the original video sequence by a **model**, an efficient coded representation that can be used to reconstruct an approximation of the video data. Ideally, the model should represent the sequence using as few bits as possible and with as high a fidelity as possible. These two goals, compression efficiency and high quality, are usually conflicting, i.e. a lower compressed bit rate typically produces reduced image quality at the decoder.

A video encoder (Figure 3.3) consists of three main functional units: a **prediction model**, a **spatial model** and an **entropy encoder**. The input to the prediction model is an uncompressed 'raw' video sequence. The prediction model attempts to reduce redundancy by exploiting the similarities between neighbouring video frames and/or neighbouring image samples, typically by constructing a prediction of the current video frame or block of video data. In H.264/AVC, the prediction is formed from data in the current frame or in one or more previous and/or future frames. It is created by spatial extrapolation from neighbouring image samples, **intra prediction**, or by compensating for differences between the frames, **inter** or **motion compensated prediction**. The output of the prediction model is a residual frame, created by subtracting the prediction from the actual current frame, and a set of model parameters indicating the intra prediction type or describing how the motion was compensated.

The residual frame forms the input to the spatial model which makes use of similarities between local samples in the residual frame to reduce spatial redundancy. In H.264/AVC this is carried out by applying a transform to the residual samples and quantizing the results. The transform converts the samples into another domain in which they are represented by transform coefficients. The coefficients are quantized to remove insignificant values, leaving a small number of significant coefficients that provide a more compact representation of the residual frame. The output of the spatial model is a set of quantized transform coefficients.

The parameters of the prediction model, i.e. intra prediction mode(s) or inter prediction mode(s) and motion vectors, and the spatial model, i.e. coefficients, are compressed by the entropy encoder. This removes statistical redundancy in the data, for example representing commonly occurring vectors and coefficients by short binary codes. The entropy encoder produces a compressed bit stream or file that may be transmitted and/or stored. A compressed sequence consists of coded prediction parameters, coded residual coefficients and header information.

The video decoder reconstructs a video frame from the compressed bit stream. The coefficients and prediction parameters are decoded by an entropy decoder after which the spatial model is decoded to reconstruct a version of the residual frame. The decoder uses the prediction

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000053

parameters, together with previously decoded image pixels, to create a prediction of the current frame and the frame itself is reconstructed by adding the residual frame to this prediction.

3.3 Prediction model

The data to be processed are a set of image samples in the current frame or field. The goal of the prediction model is to reduce redundancy by forming a prediction of the data and subtracting this prediction from the current data. The prediction may be formed from previously coded frames (a **temporal prediction**) or from previously coded image samples in the same frame (a **spatial prediction**). The output of this process is a set of residual or difference samples and the more accurate the prediction process, the less energy is contained in the residual. The residual is encoded and sent to the decoder which re-creates the same prediction so that it can add the decoded residual and reconstruct the current frame. In order that the decoder can create an identical prediction, it is essential that the encoder forms the prediction using only data available to the decoder, i.e. data that has already been coded and transmitted.

3.3.1 Temporal prediction

The predicted frame is created from one or more past or future frames known as **reference frames**. The accuracy of the prediction can usually be improved by compensating for motion between the reference frame(s) and the current frame.

3.3.1.1 Prediction from the previous video frame

The simplest method of temporal prediction is to use the previous frame as the predictor for the current frame. Two successive frames from a video sequence are shown in Figure 3.4 and Figure 3.5. Frame 1 is used as a predictor for frame 2 and the residual formed by subtracting the predictor (frame 1) from the current frame (frame 2) is shown in Figure 3.6. In this image, mid-grey represents a difference of zero and light or dark greys correspond to positive and negative differences respectively. The obvious problem with this simple prediction is that a lot of energy remains in the residual frame (indicated by the light and dark areas) and this means that there is still a significant amount of information to compress after temporal prediction. Much of the residual energy is due to object movements between the two frames and a better prediction may be formed by **compensating** for motion between the two frames.

3.3.1.2 Changes due to motion

Causes of changes between video frames include motion, uncovered regions and lighting changes. Types of motion include rigid object motion, for example a moving car, deformable object motion, for example a person speaking, and camera motion such as panning, tilt, zoom and rotation. An uncovered region may be a portion of the scene background uncovered by a moving object. With the exception of uncovered regions and lighting changes, these differences correspond to pixel movements between frames. It is possible to estimate the trajectory of each pixel between successive video frames, producing a field of pixel trajectories known as the

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000054



Figure 3.4 Frame 1



Figure 3.5 Frame 2



Figure 3.6 Difference

optical flow or optic flow [ii]. Figure 3.7 shows the optical flow field for the frames of Figure 3.4 and Figure 3.5. The complete field contains a flow vector for every pixel position but for clarity, the field is sub-sampled so that only the vector for every 2nd pixel is shown.

If the optical flow field is accurately known, it should be possible to form an accurate prediction of most of the pixels of the current frame by moving each pixel from the reference frame along its optical flow vector. However, this is not a practical method of motion compensation for several reasons. An accurate calculation of optical flow is very computationally intensive,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000055



Figure 3.7 Optical flow

since the more accurate methods use an iterative procedure for every pixel, and for the decoder to re-create the prediction frame it would be necessary to send the optical flow vector for every pixel to the decoder resulting in a large amount of transmitted data and negating the advantage of a small residual.

3.3.1.3 Block-based motion estimation and compensation

A practical and widely used method of motion compensation is to compensate for movement of rectangular sections or 'blocks' of the current frame. The following procedure is carried out for each block of MxN samples in the current frame:

- Search an area in the reference frame, a past or future frame, to find a similar MxN-sample region. This search may be carried out by comparing the MxN block in the current frame with some or all of the possible MxN regions in a search area, e.g. a region centred on the current block position, and finding the region that gives the 'best' match. A popular matching criterion is the energy in the residual formed by subtracting the candidate region from the current MxN block, so that the candidate region that minimises the residual energy is chosen as the best match. This process of finding the best match is known as **motion estimation**.
- The chosen candidate region becomes the predictor for the current MxN block (a motion compensated prediction) and is subtracted from the current block to form a residual MxN block (**motion compensation**).
- The residual block is encoded and transmitted and the offset between the current block and the position of the candidate region (**motion vector**) is also transmitted.

The decoder uses the received motion vector to re-create the predictor region. It decodes the residual block, adds it to the predictor and reconstructs a version of the original block.

Block-based motion compensation is popular for a number of reasons. It is relatively straightforward and computationally tractable, it fits well with rectangular video frames and with block-based image transforms such as the Discrete Cosine Transform and it provides a

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000056

reasonably effective temporal model for many video sequences. There are however a number of disadvantages. For example, 'real' objects rarely have neat edges that match rectangular boundaries, objects often move by a fractional number of pixel positions between frames and many types of object motion are hard to compensate for using block-based methods, e.g. deformable objects, rotation, warping and complex motion such as a cloud of smoke. Despite these disadvantages, block-based motion compensation is the basis of the temporal prediction model used by all current video coding standards.

3.3.1.4 Motion compensated prediction of a macroblock

The **macroblock**, corresponding to a 16 × 16-pixel region of a frame, is the basic unit for motion compensated prediction in a number of important visual coding standards including MPEG-1, MPEG-2, MPEG-4 Visual, H.261, H.263 and H.264. For source video material in the popular 4:2:0 format (Chapter 2), a macroblock is organised as shown in Figure 3.8. A 16 × 16-pixel region of the source frame is represented by 256 luminance samples arranged in four 8 × 8-sample blocks, 64 red chrominance samples in one 8 × 8 block and 64 blue chrominance samples in one 8 × 8 block, giving a total of six 8 × 8 blocks. An H.264/AVC codec processes each video frame in units of a macroblock.

Motion estimation:
Motion estimation of a macroblock involves finding a 16 × 16-sample region in a reference frame that closely matches the current macroblock. The reference frame is a previously encoded frame from the sequence and may be before or after the current frame in display order. A search area in the reference frame centred on the current macroblock position is searched and the 16 × 16 region within the search area that minimizes a matching criterion is chosen as the 'best match' (Figure 3.9).

Motion compensation:
The luma and chroma samples of the selected 'best' matching region in the reference frame is subtracted from the current macroblock to produce a residual macroblock that is encoded and transmitted together with a motion vector describing the position of the best matching region relative to the current macroblock position.

There are many variations on the basic motion estimation and compensation process. The reference frame may be a previous frame in temporal order, a future frame or a combination

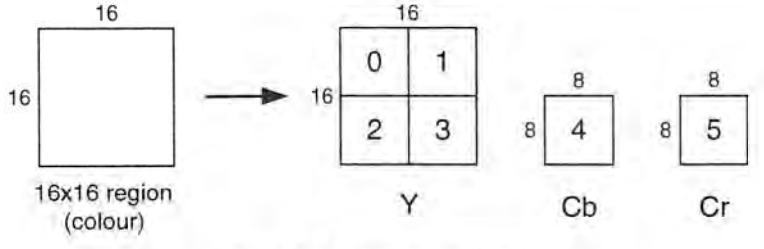


Figure 3.8 Macroblock (4:2:0)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000057

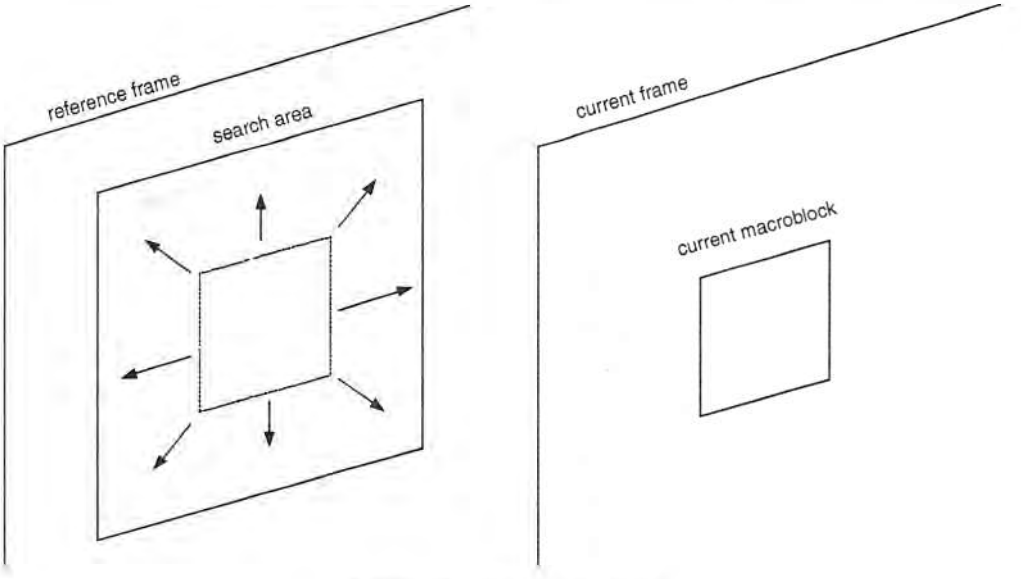


Figure 3.9 Motion estimation

of predictions from two or more previously encoded frames. If a future frame is chosen as the reference, it is necessary to encode this frame before the current frame, i.e. frames must be encoded out of order. Where there is a significant change between the reference and current frames, for example a scene change or an uncovered area, it may be more efficient to encode the macroblock without motion compensation and so an encoder may choose **intra** mode encoding using intra prediction or **inter** mode encoding with motion compensated prediction for each macroblock. Moving objects in a video scene rarely follow 'neat' 16 × 16-pixel boundaries and so it may be more efficient to use a variable block size for motion estimation and compensation. Objects may move by a fractional number of pixels between frames, e.g. 2.78 pixels rather than 2.0 pixels in the horizontal direction, and a better prediction may be formed by interpolating the reference frame to sub-pixel positions before searching these positions for the best match.

3.3.1.5 Motion compensation block size

Two successive frames of a video sequence are shown in Figure 3.10 and Figure 3.11. Frame 1 is subtracted from frame 2 without motion compensation to produce a residual frame (Figure 3.12). The energy in the residual is reduced by motion compensating each 16 × 16 macroblock (Figure 3.13). Motion compensating each 8 × 8 block instead of each 16 × 16 macroblock reduces the residual energy further (Figure 3.14) and motion compensating each 4 × 4 block gives the smallest residual energy of all (Figure 3.15). These examples show that smaller motion compensation block sizes can produce better motion compensation results. However, a smaller block size leads to increased complexity, with more search operations to be carried out, and an increase in the number of motion vectors that need to be transmitted. Sending each

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000058



Figure 3.10 Frame 1



Figure 3.11 Frame 2



Figure 3.12 Residual : no motion compensation

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000059

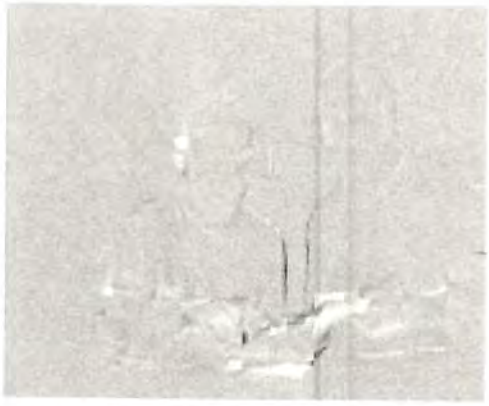


Figure 3.13 Residual : 16 × 16 block size

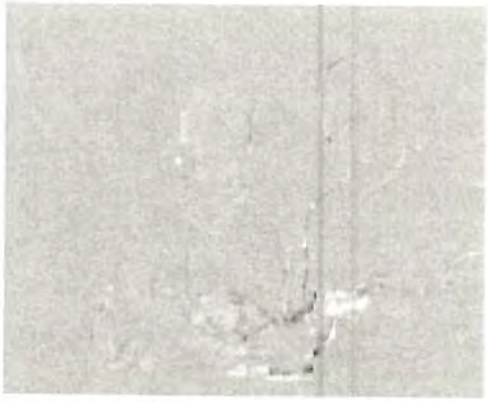


Figure 3.14 Residual : 8 × 8 block size



Figure 3.15 Residual : 4 × 4 block size

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000060

motion vector requires bits to be transmitted and the extra overhead for vectors may outweigh the benefit of reduced residual energy. An effective compromise is to adapt the block size to the picture characteristics, for example choosing a large block size in flat, homogeneous regions of a frame and choosing a small block size around areas of high detail and complex motion (Chapter 6).

3.3.1.6 Sub-pixel motion compensation

Figure 3.16 shows a close-up view of part of a reference frame. In some cases, predicting from interpolated sample positions in the reference frame may form a better motion compensated prediction. In Figure 3.17, the reference region pixels are interpolated to half-pixel positions and it may be possible to find a better match for the current macroblock by searching the interpolated samples. Sub-pixel motion estimation and compensation involves searching sub-pixel interpolated positions as well as integer-pixel positions and choosing the position that gives the best match and minimizes the residual energy. Figure 3.18 shows the concept of quarter-pixel motion estimation. In the first stage, motion estimation finds the best match on the integer pixel grid (circles). The encoder searches the half-pixel positions immediately next to this best match (squares) to see whether the match can be improved and if required, the quarter-pixel positions next to the best half-pixel position (triangles) are then searched. The final match, at an integer, half-pixel or quarter-pixel position, is subtracted from the current block or macroblock.

The residual in Figure 3.19 is produced using a block size of 4 × 4 pixels using half-pixel interpolation and has less residual energy than Figure 3.15. This approach may be extended further by interpolation onto a $^1/_4$-pixel grid to give a still smaller residual (Figure 3.20). In

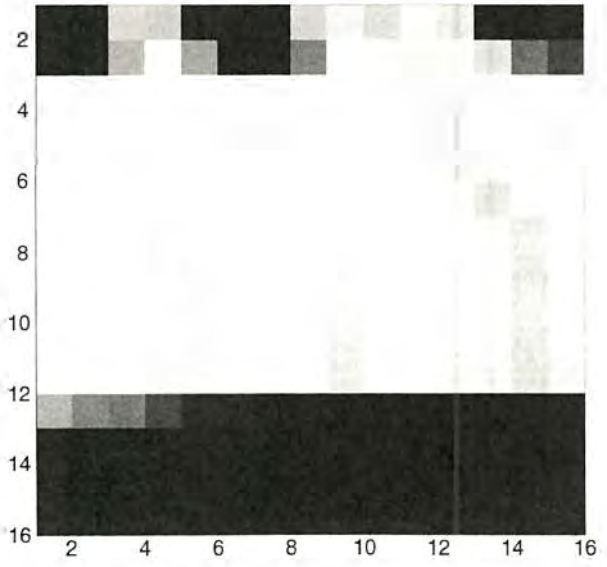


Figure 3.16 Close-up of reference region

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000061

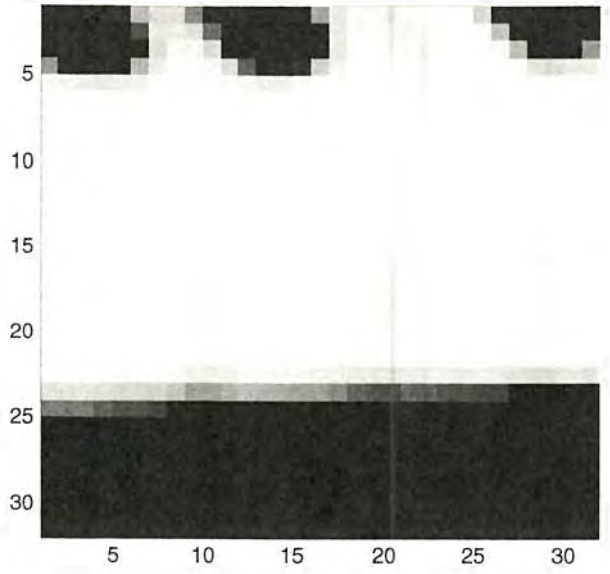


Figure 3.17 Reference region interpolated to half-pixel positions

general, 'finer' interpolation provides better motion compensation performance, producing a smaller residual at the expense of increased complexity. The performance gain tends to diminish as the interpolation steps increase. Half-pixel interpolation gives a significant gain over integer-pixel motion compensation, quarter-pixel interpolation gives a moderate further improvement, eighth-pixel interpolation gives a small further improvement again and so on.

Some examples of the performance achieved by sub-pixel interpolation are given in Table 3.1. A motion-compensated reference frame, the previous frame in the sequence, is subtracted from the current frame and the energy of the residual, approximated by the Sum of Absolute Errors (SAE), is listed in the table. A lower SAE indicates better motion compensation performance. In each case, sub-pixel motion compensation gives improved performance compared with integer-pixel compensation. The improvement from integer to half-pixel is

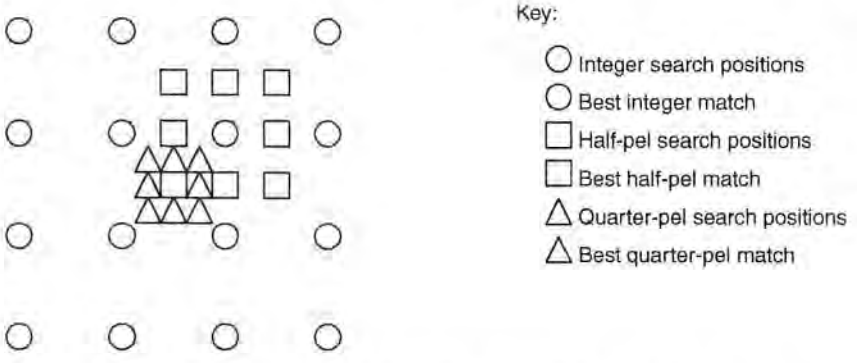


Figure 3.18 Integer, half-pixel and quarter-pixel motion estimation

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000062



Figure 3.19 Residual : 4 × 4 blocks, 1/2-pixel compensation



Figure 3.20 Residual : 4 × 4 blocks, 1/4-pixel compensation

Table 3.1 SAE of residual frame after motion compensation, 16 × 16 block size

Sequence	No motion compensation	Integer-pel	Half-pel	Quarter-pel
'Violin', QCIF	171945	153475	128320	113744
'Grasses', QCIF	248316	245784	228952	215585
'Carphone', QCIF	102418	73952	56492	47780

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000063

more significant than the further improvement from half- to quarter-pixel. The sequence 'Grasses' has highly complex motion and is particularly difficult to motion-compensate, hence the large SAE; 'Violin' and 'Carphone' are less complex and motion compensation produces smaller SAE values.

Searching for matching 4 × 4 blocks with ¼-pixel interpolation is considerably more complex than searching for 16 × 16 blocks with no interpolation. In addition to the extra complexity, there is a coding penalty since the vector for every block must be encoded and transmitted to the receiver in order to correctly reconstruct the image. As the block size is reduced, the number of vectors that have to be transmitted increases. More bits are required to represent ½ or ¼-pixel vectors because the fractional part of the vector, e.g. 0.25 or 0.5, must be encoded as well as the integer part. Figure 3.21 plots the integer motion vectors that are transmitted along with the residual of Figure 3.13. The motion vectors required for the residual of Figure 3.20 are plotted in Figure 3.22, in which there are 16 times as many vectors, each represented by two fractional numbers DX and DY with ¼-pixel accuracy. There is therefore a trade-off in compression efficiency associated with more complex motion compensation schemes, since more accurate motion compensation requires more bits to encode the vector field, but fewer bits to encode the residual whereas less accurate motion compensation requires fewer bits for the vector field but more bits for the residual. The efficiency of sub-pixel interpolation schemes can be improved by using sophisticated interpolation filters [iii].

3.3.2 *Spatial model: intra prediction*

The prediction for the current block of image samples is created from previously-coded samples in the same frame. Figure 3.23 shows a block that is to be predicted, centre, in the current frame. Assuming that the blocks of image samples are coded in raster-scan order, which is not always the case, the upper/left shaded blocks are available for intra prediction. These blocks have already been coded and placed in the output bitstream. When the decoder processes the current block, the shaded upper/left blocks are already decoded and can be used to re-create the prediction.

Many different approaches to intra prediction have been proposed. H.264/AVC uses spatial extrapolation to create an intra prediction for a block or macroblock. Figure 3.24 shows the

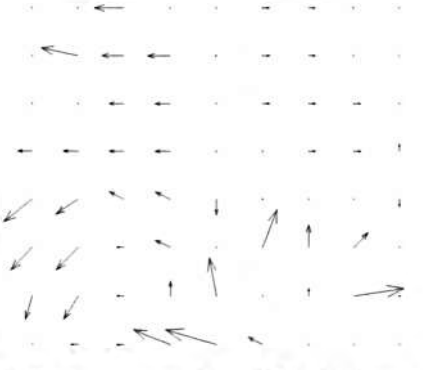


Figure 3.21 Motion vector map : 16 × 16 blocks, integer vectors

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000064



Figure 3.22 Motion vector map : 4 × 4 blocks, 1/4-pixel vectors

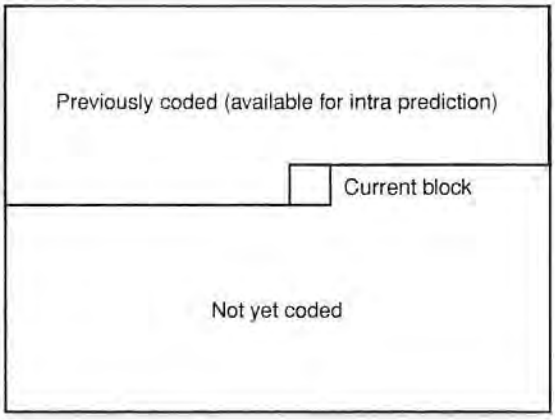


Figure 3.23 Intra prediction: available samples

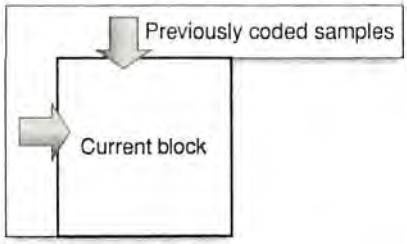


Figure 3.24 Intra prediction: spatial extrapolation

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000065

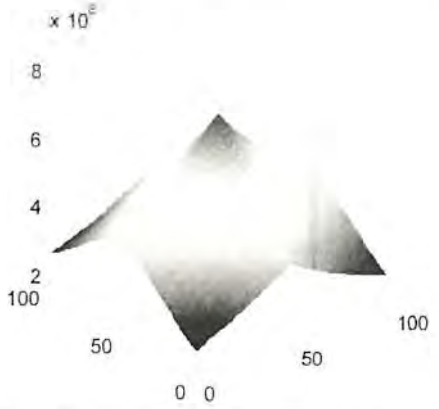


Figure 3.25 2D autocorrelation function of image

general concept. One or more prediction(s) are formed by extrapolating samples from the top and/or left sides of the current block. In general, the nearest samples are most likely to be highly correlated with the samples in the current block (Figure 3.25) and so only the pixels along the top and/or left edges are used to create the prediction block. Once the prediction has been generated, it is subtracted from the current block to form a residual in a similar way to inter prediction. The residual is transformed and encoded, together with an indication of how the prediction was generated. Intra prediction is described in detail in Chapter 6.

3.4 Image model

A natural video image consists of a grid of sample values. Natural images are often difficult to compress in their original form because of the high correlation between neighbouring image samples. Figure 3.25 shows the two-dimensional autocorrelation function of a natural video image (Figure 3.4) in which the height of the graph at each position indicates the similarity between the original image and a spatially-shifted copy of itself. The peak at the centre of the figure corresponds to zero shift. As the spatially-shifted copy is moved away from the original image in any direction, the function drops off as shown in the figure, with the gradual slope indicating that image samples within a local neighbourhood are highly correlated.

A motion-compensated residual image such as Figure 3.20 has an autocorrelation function (Figure 3.26) that drops off rapidly as the spatial shift increases, indicating that neighbouring samples are weakly correlated. Efficient motion compensation or intra prediction reduces local correlation in the residual making it easier to compress than the original video frame. The function of the **image model** is to further decorrelate image or residual data and to convert it into a form that can be efficiently compressed using an entropy coder. Practical image models typically have three main components, transformation to decorrelate and compact the data, quantization to reduce the precision of the transformed data and reordering to arrange the data to group together significant values.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000066

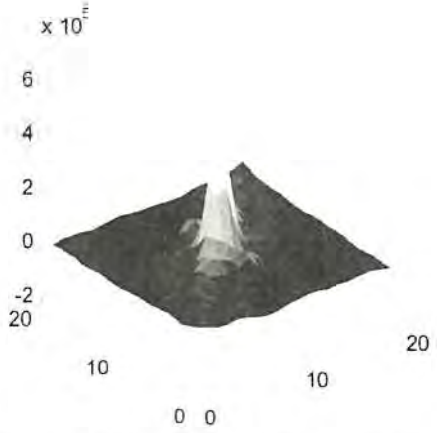


Figure 3.26 2D autocorrelation function of residual

3.4.1 Predictive image coding

Motion compensation is an example of predictive coding in which an encoder creates a prediction of a region of the current frame based on a previous or future frame and subtracts this prediction from the current region to form a residual. If the prediction is successful, the energy in the residual is lower than in the original frame and the residual can be represented with fewer bits.

Predictive coding was used as the basis for early image compression algorithms and is an important component of H.264 Intra coding, see section 3.3.2 and Chapter 6. Spatial prediction involves predicting an image sample or region from previously-transmitted samples in the same image or frame and is sometimes described as 'Differential Pulse Code Modulation' (DPCM), a term borrowed from a method of differentially encoding PCM samples in telecommunication systems.

Figure 3.27 shows a pixel X that is to be encoded. If the frame is processed in raster order, then pixels A, B and C in the current and previous rows are available in both the encoder

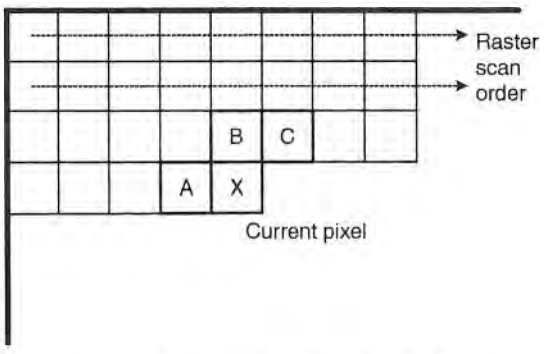


Figure 3.27 Spatial prediction (DPCM)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000067

and the decoder since these should already have been decoded before X. The encoder forms a prediction for X based on some combination of previously coded pixels, subtracts this prediction from X and encodes the residual, the result of the subtraction. The decoder forms the same prediction and adds the decoded residual to reconstruct the pixel.

Example

Encoder prediction $P(X) = (2A + B + C)/4$

Residual $R(X) = X - P(X)$ *is encoded and transmitted.*

Decoder decodes $R(X)$ *and forms the same prediction:*

$$P(X) = (2A + B + C)/4$$

Reconstructed pixel $X = R(X) + P(X)$

If the encoding process is lossy, i.e. if the residual is quantized – see section 3.4.3, then the decoded pixels A′, B′ and C′ may not be identical to the original A, B and C due to losses during encoding and so the above process could lead to a cumulative mismatch or 'drift' between the encoder and decoder. To avoid this, the encoder should itself decode the residual R'(X) and reconstruct each pixel. Hence the encoder uses decoded pixels A′, B′ and C′ to form the prediction, i.e. $P(X) = (2A' + B' + C')/4$ in the above example. In this way, both encoder and decoder use the same prediction P(X) and drift is avoided.

The compression efficiency of this approach depends on the accuracy of the prediction P(X). If the prediction is accurate, i.e. P(X) is a close approximation of X, then the residual energy will be small. However, it is usually not possible to choose a predictor that works well for all areas of a complex image and better performance may be obtained by adapting the predictor depending on the local statistics of the image for example, using different predictors for areas of flat texture, strong vertical texture, strong horizontal texture, etc. It is necessary for the encoder to indicate the choice of predictor to the decoder and so there is a tradeoff between efficient prediction and the extra bits required to signal the choice of predictor.

3.4.2 Transform coding

3.4.2.1 Overview

The purpose of the transform stage in an image or video CODEC is to convert image or motion-compensated residual data into another domain, the transform domain. The choice of transform depends on a number of criteria:

1. Data in the transform domain should be decorrelated, i.e. separated into components within minimal inter-dependence, and compact, i.e. most of the energy in the transformed data should be concentrated into a small number of values.
2. The transform should be reversible.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000068

3. The transform should be computationally tractable, e.g. low memory requirement, achievable using limited-precision arithmetic, low number of arithmetic operations, etc.

Many transforms have been proposed for image and video compression and the most popular transforms tend to fall into two categories, block-based and image-based. Examples of block-based transforms include the Karhunen-Loeve Transform (KLT), Singular Value Decomposition (SVD) and the ever-popular Discrete Cosine Transform (DCT) and its approximations [iv]. Each of these operates on blocks of N × N image or residual samples and hence the image is processed in units of a block. Block transforms have low memory requirements and are well suited to compression of block-based motion compensation residuals but tend to suffer from artefacts at block edges ('blockiness'). Image-based transforms operate on an entire image or frame or a large section of the image known as a 'tile'. The most popular image transform is the Discrete Wavelet Transform, DWT or just 'wavelet'. Image transforms such as the DWT have been shown to out-perform block transforms for still image compression but they tend to have higher memory requirements because the whole image or tile is processed as a unit and they do not necessarily 'fit' well with block-based motion compensation. The DCT and the DWT both feature in MPEG-4 Visual, with approximations to the DCT incorporated in H.264, and are discussed further in the following sections.

3.4.2.2 DCT

The Discrete Cosine Transform (DCT) operates on **X**, a block of N × N samples, typically image samples or residual values after prediction, to create **Y**, an N × N block of coefficients. The action of the DCT and its inverse, the IDCT can be described in terms of a transform matrix **A**. The forward DCT (FDCT) of an N × N sample block is given by:

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^{\mathrm{T}} \tag{3.1}$$

and the inverse DCT (IDCT) by:

$$\mathbf{X} = \mathbf{A}^{\mathrm{T}}\mathbf{Y}\mathbf{A} \tag{3.2}$$

where **X** is a matrix of samples, **Y** is a matrix of coefficients and **A** is an N × N transform matrix. The elements of **A** are:

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N} \quad \text{where } C_i = \sqrt{\frac{1}{N}}\ (\mathrm{i}=0),\ C_i = \sqrt{\frac{2}{N}}\ (\mathrm{i}>0) \tag{3.3}$$

(3.1) and (3.2) may be written in summation form:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{ij} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N} \quad \text{(3.4 2–D FDCT)}$$

$$X_{ij} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N} \quad \text{(3.5 2–D IDCT)}$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000069

Example: N = 4

The transform matrix **A** for a 4 × 4 DCT is:

$$A = \begin{bmatrix} \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{5\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{7\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{2\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{6\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{10\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{14\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{9\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{15\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \tag{3.6}$$

The cosine function is symmetrical and repeats after 2π radians and hence **A** can be simplified to:

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) \end{bmatrix} \tag{3.7}$$

or

$$A = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad \text{where} \quad \begin{aligned} a &= \frac{1}{2} \\ b &= \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) \\ c &= \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) \end{aligned} \tag{3.8}$$

Evaluating the cosines gives:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.653 & 0.271 & -0.271 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.271 & -0.653 & 0.653 & -0.271 \end{bmatrix}$$

The output of a 2-dimensional FDCT is a set of N × N coefficients representing the image block data in the DCT domain which can be considered as 'weights' of a set of standard **basis patterns**. The basis patterns for the 4 × 4 and 8 × 8 DCTs are shown in Figure 3.28 and Figure 3.29 respectively and are composed of combinations of horizontal and vertical cosine functions. Any image block may be reconstructed by combining all N × N basis patterns, with each basis multiplied by the appropriate weighting factor (coefficient).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000070

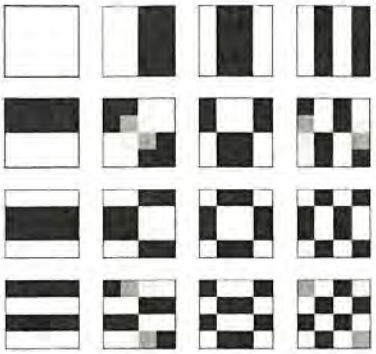


Figure 3.28 4 × 4 DCT basis patterns

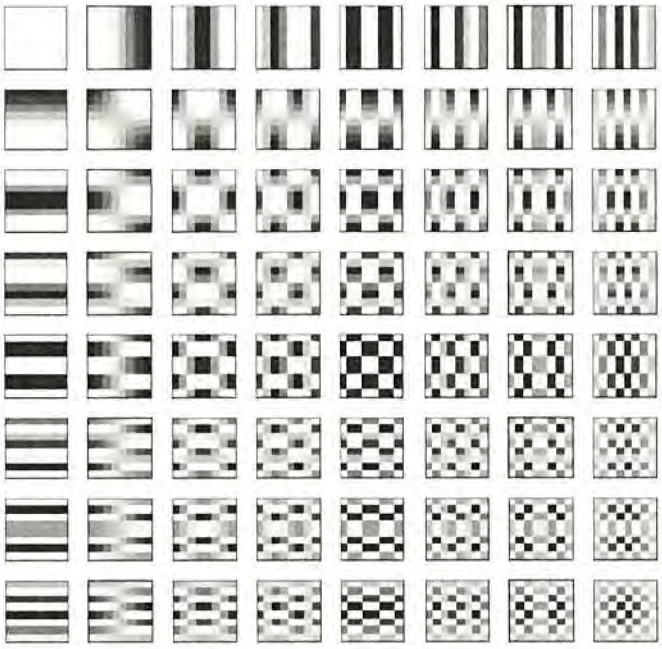


Figure 3.29 8 × 8 DCT basis patterns

Example 1 Calculating the DCT of a 4 × 4 block

Let **X** be a 4 × 4 block of samples from an image:

	J = 0	1	2	3
i= 0	5	11	8	10
1	9	8	4	12
2	1	10	11	4
3	19	6	15	7

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000071

The Forward DCT of **X** is given by: $\mathbf{Y} = \mathbf{AXA}^{\mathbf{T}}$. The first matrix multiplication, $\mathbf{Y}' = \mathbf{AX}$, corresponds to calculating the 1-dimensional DCT of each **column** of **X**. For example, Y'_{00} is calculated as follows:

$$Y'_{00} = A_{00}X_{00} + A_{01}X_{10} + A_{02}X_{20} + A_{03}X_{30} =$$
$$(0.5 * 5) + (0.5 * 9) + (0.5 * 1) + (0.5 * 19) = 17.0$$

The complete result of the column calculations is:

$$Y' = AX = \begin{bmatrix} 17 & 17.5 & 19 & 16.5 \\ -6.981 & 2.725 & -6.467 & 4.125 \\ 7 & -0.5 & 4 & 0.5 \\ -9.015 & 2.660 & 2.679 & -4.414 \end{bmatrix}$$

Carrying out the second matrix multiplication, $\mathbf{Y} = \mathbf{Y}'\mathbf{A}^{\mathbf{T}}$, is equivalent to carrying out a 1-D DCT on each **row** of Y':

$$Y = AXA^T = \begin{bmatrix} 35.0 & -0.079 & -1.5 & 1.115 \\ -3.299 & -4.768 & 0.443 & -9.010 \\ 5.5 & 3.029 & 2.0 & 4.699 \\ -4.045 & -3.010 & -9.384 & -1.232 \end{bmatrix}$$

Note that the order of the row and column calculations does not affect the final result.

Example 2 Image block and DCT coefficients

Figure 3.30 shows an image with a 4 × 4 block selected and Figure 3.31 shows the block in close-up, together with the DCT coefficients. The advantage of representing the block in the DCT domain is not immediately obvious since there is no reduction in the amount of data;



Figure 3.30 Image section showing 4 × 4 block

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000072

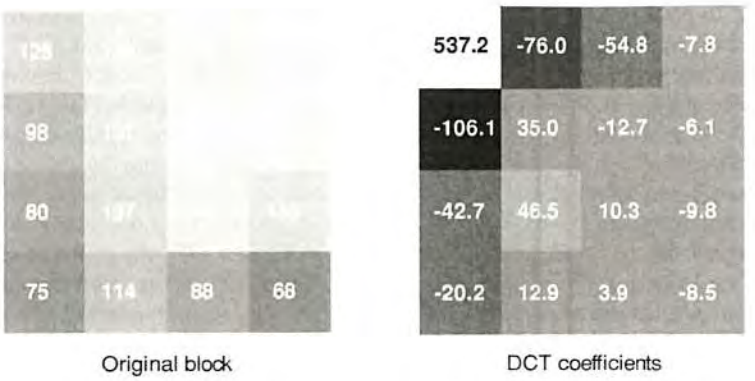


Figure 3.31 Close-up of 4 × 4 block; DCT coefficients

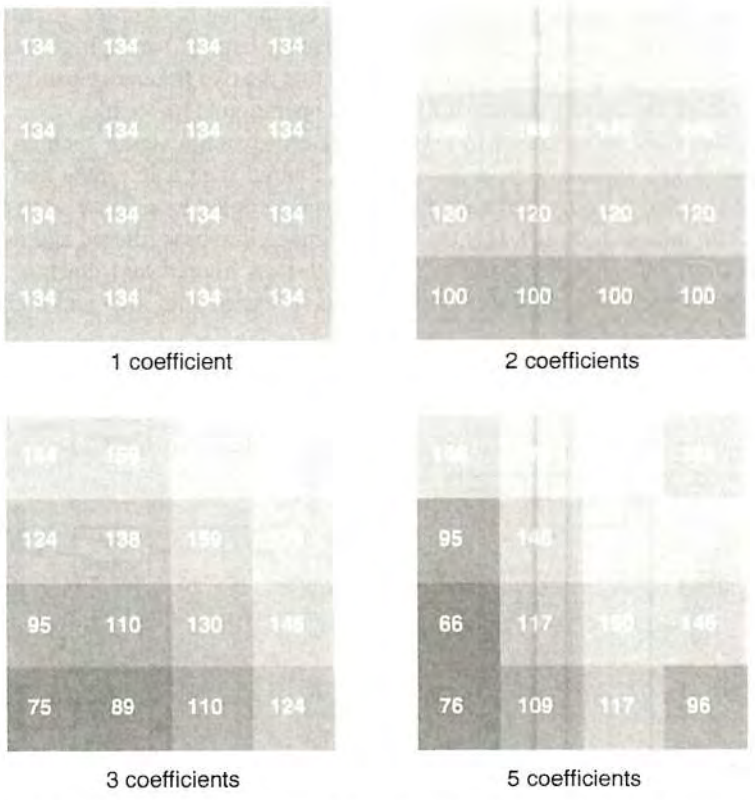


Figure 3.32 Block reconstructed from (a) 1, (b) 2, (c) 3, (d) 5 coefficients

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000073

instead of 16 pixel values, we need to store 16 DCT coefficients. The usefulness of the DCT becomes clear when the block is reconstructed from a subset of the coefficients.

Setting all the coefficients to zero except the most significant, coefficient (0,0) described as the 'DC' coefficient, and performing the IDCT gives the output block shown in Figure 3.32 (a), the mean of the original pixel values. Calculating the IDCT of the two most significant coefficients gives the block shown in Figure 3.32 (b). Adding more coefficients before calculating the IDCT produces a progressively more accurate reconstruction of the original block and by the time five coefficients are included (Figure 3.32 (d)), the reconstructed block is a reasonably close match to the original. Hence it is possible to reconstruct an approximate copy of the block from a subset of the 16 DCT coefficients. Removing the coefficients with insignificant magnitudes, for example by quantization, see section 3.4.3, enables image data to be represented with a reduced number of coefficient values at the expense of some loss of quality.

3.4.2.3 Wavelet

The 'wavelet transform' that is popular in image compression is based on sets of filters with coefficients that are equivalent to discrete wavelet functions [v]. The basic operation of the transform is as follows, applied to a discrete signal containing N samples. A pair of filters is applied to the signal to decompose it into a low frequency band (L) and a high frequency band (H). Each band is subsampled by a factor of two, so that the two frequency bands each contain N/2 samples. With the correct choice of filters, this operation is reversible.

This approach may be extended to apply to a 2-dimensional signal such as an intensity image (Figure 3.33). Each row of a 2D image is filtered with a low-pass and a high-pass filter (L_x and H_x) and the output of each filter is down-sampled by a factor of two to produce the intermediate images L and H. L is the original image low-pass filtered and downsampled in the x-direction and H is the original image high-pass filtered and downsampled in the x-direction. Next, each column of these new images is filtered with low- and high-pass filters

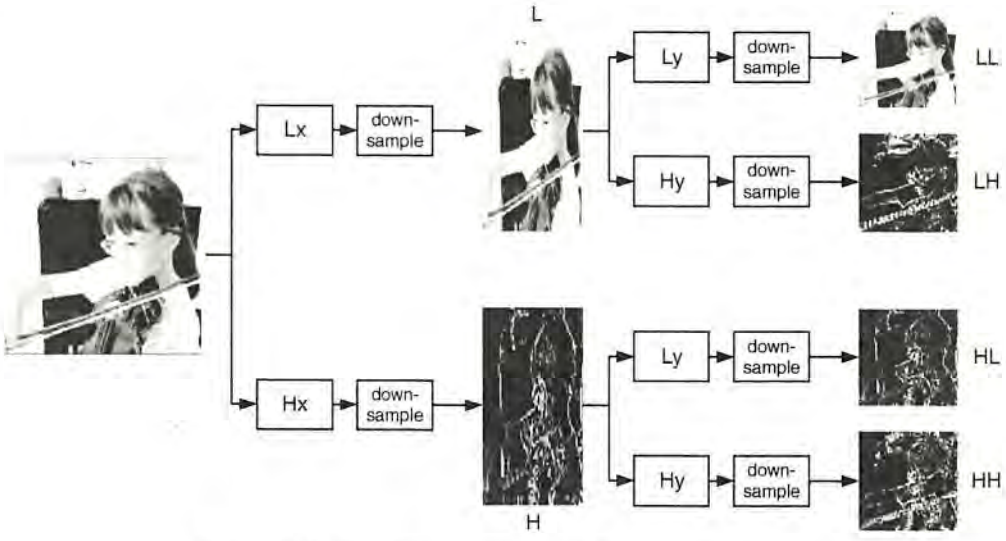


Figure 3.33 Two-dimensional wavelet decomposition process

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000074



Figure 3.34 Image after one level of decomposition

L_y and H_y and down-sampled by a factor of two to produce four sub-images LL, LH, HL and HH. These four 'sub-band' images can be combined to create an output image with the same number of samples as the original (Figure 3.34). 'LL' is the original image, low-pass filtered in horizontal and vertical directions and subsampled by a factor of two. 'HL' is high-pass filtered in the vertical direction and contains residual vertical frequencies, 'LH' is high-pass filtered in the horizontal direction and contains residual horizontal frequencies and 'HH' is high-pass filtered in both horizontal and vertical directions. Between them, the four sub-band images contain all of the information present in the original image but the sparse nature of the LH, HL and HH sub-bands makes them amenable to compression.

In an image compression application, the 2-dimensional wavelet decomposition described above is applied again to the 'LL' image, forming four new sub-band images. The resulting low-pass image, always the top-left sub-band image, is iteratively filtered to create a tree of sub-band images. Figure 3.35 shows the result of two stages of this decomposition and



Figure 3.35 Two-stage wavelet decomposition of image

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000075



Figure 3.36 Five-stage wavelet decomposition of image

Figure 3.36 shows the result of five stages of decomposition. Many of the samples (coefficients) in the higher-frequency sub-band images are close to zero, shown here as near-black, and it is possible to achieve compression by removing these insignificant coefficients prior to transmission. At the decoder, the original image is reconstructed by repeated up-sampling, filtering and addition, reversing the order of operations shown in Figure 3.33.

3.4.3 Quantization

A quantizer maps a signal with a range of values X to a quantized signal with a reduced range of values Y. It should be possible to represent the quantized signal with fewer bits than the original since the range of possible values is smaller. A **scalar quantizer** maps one sample of the input signal to one quantized output value and a **vector quantizer** maps a group of input samples, a 'vector', to a group of quantized values.

3.4.3.1 Scalar quantization

A simple example of scalar quantization is the process of rounding a fractional number to the nearest integer, i.e. the mapping is from R to Z. The process is lossy (not reversible) since it is not possible to determine the exact value of the original fractional number from the rounded integer.

A more general example of a uniform quantizer is:

$$\begin{aligned} FQ &= round\left(\frac{X}{QP}\right) \\ Y &= FQ.QP \end{aligned} \tag{3.9}$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000076

where QP is a quantization 'step size'. The quantized output levels are spaced at uniform intervals of QP as shown in the following example.

Example Y = QP.round(X/QP)

	Y			
X	QP=1	QP=2	QP=3	QP=5
−4	−4	−4	−3	−5
−3	−3	−2	−3	−5
−2	−2	−2	−3	0
−1	−1	0	0	0
0	0	0	0	0
1	1	0	0	0
2	2	2	3	0
3	3	2	3	5
4	4	4	3	5
5	5	4	6	5
6	6	6	6	5
7	7	6	6	5
8	8	8	9	10
9	9	8	9	10
10	10	10	9	10
11	11	10	12	10
.....				

Figure 3.37 shows two examples of scalar quantizers, a linear quantizer with a uniform mapping between input and output values and a non-linear quantizer that has a 'dead zone' about zero, in which small-valued inputs are mapped to zero.

In image and video compression CODECs, the quantization operation is usually made up of two parts, a forward quantizer FQ in the encoder and an 'inverse quantizer' or 'rescaler' (IQ) in the decoder. A critical parameter is the **step size** QP between successive re-scaled values. If the step size is large, the range of quantized values is small and can therefore be efficiently represented and hence highly compressed during transmission, but the re-scaled values are a crude approximation to the original signal. If the step size is small, the re-scaled values match the original signal more closely but the larger range of quantized values reduces compression efficiency.

Quantization may be used to reduce the precision of image data after applying a transform such as the DCT or wavelet transform and to remove insignificant values such as near-zero DCT or wavelet coefficients. The forward quantizer in an image or video encoder is designed to map insignificant coefficient values to zero whilst retaining a small number of significant, non-zero coefficients. The output of a forward quantizer is therefore typically a 'sparse' array of quantized coefficients, mainly containing zeros.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000077

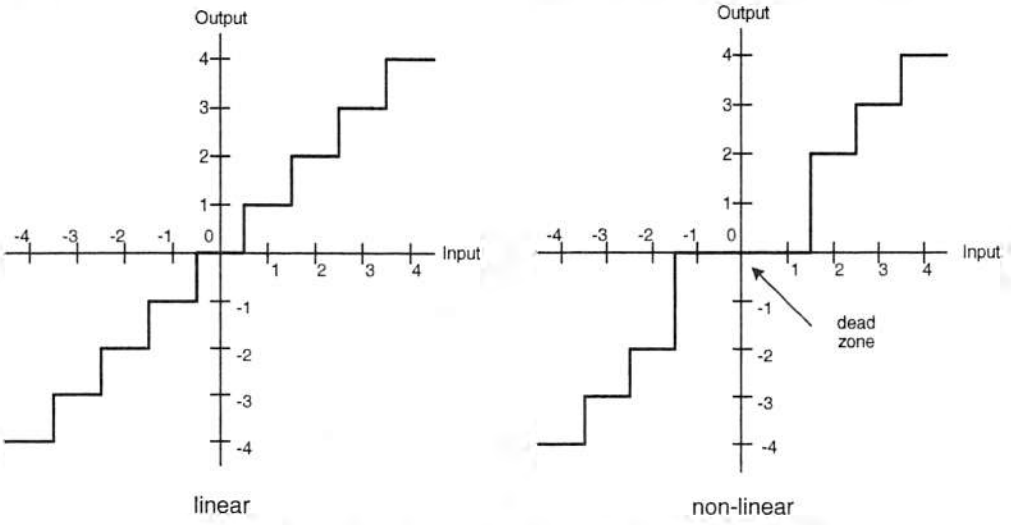


Figure 3.37 Scalar quantizers: linear; non-linear with dead zone

3.4.3.2 Vector quantization

A vector quantizer maps a set of input data such as a block of image samples to a single value (codeword) and at the decoder, each codeword maps to an approximation to the original set of input data, a 'vector'. The set of vectors are stored at the encoder and decoder in a codebook. A typical application of vector quantization to image compression [vi] is as follows:

1. Partition the original image into regions such as N × N pixel blocks.
2. Choose a vector from the codebook that matches the current region as closely as possible.
3. Transmit an index that identifies the chosen vector to the decoder.
4. At the decoder, reconstruct an approximate copy of the region using the selected vector.

A basic system is illustrated in Figure 3.38. Here, quantization is applied in the image (spatial) domain, i.e. groups of image samples are quantized as vectors, but it can equally be applied to motion compensated and/or transformed data. Key issues in vector quantizer design include the design of the codebook and efficient searching of the codebook to find the optimal vector.

3.4.4 Reordering and zero encoding

Quantized transform coefficients are required to be encoded as compactly as possible prior to storage and transmission. In a transform-based image or video encoder, the output of the quantizer is a sparse array containing a few non-zero coefficients and a large number of zero-valued coefficients. Re-ordering to group together non-zero coefficients and efficient encoding of zero coefficients are applied prior to entropy encoding.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000078

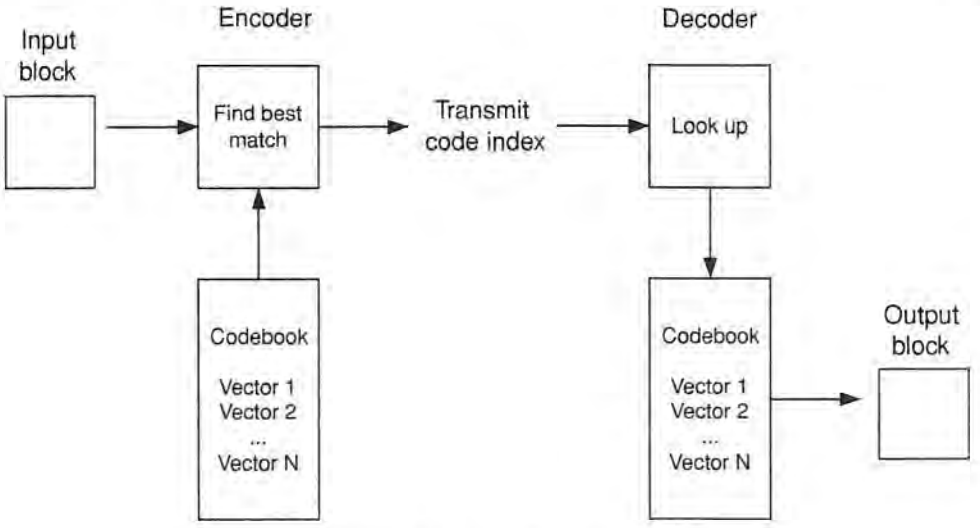


Figure 3.38 Vector quantization

3.4.4.1 DCT

Coefficient distribution

The significant DCT coefficients of a block of image or residual samples are typically the 'low frequency' positions around the DC (0,0) coefficient.

Figure 3.39 plots the probability of non-zero DCT coefficients at each position in an 8 × 8 block in a QCIF residual frame (Figure 3.6). The non-zero DCT coefficients are clustered around the top-left (DC) coefficient and the distribution is roughly symmetrical in the horizontal and vertical directions.

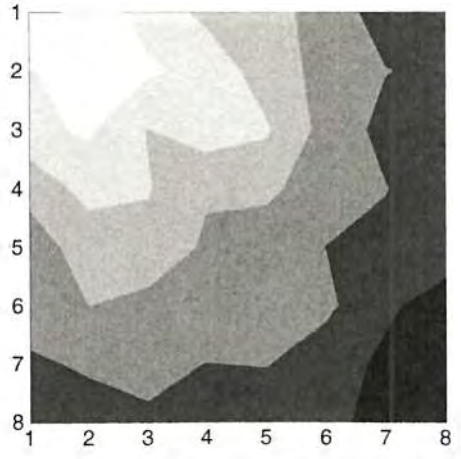


Figure 3.39 8 × 8 DCT coefficient distribution (frame)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000079



Figure 3.40 Residual field picture

Figure 3.41 plots the probability of non-zero DCT coefficients for a residual field (Figure 3.40): here, the coefficients are clustered around the DC position but are 'skewed', i.e. more non-zero coefficients occur along the left-hand edge of the plot. This is because a field picture may have a stronger high-frequency component in the vertical axis due to the subsampling in the vertical direction, resulting in larger DCT coefficients corresponding to vertical frequencies (Figure 3.28).

Scan

After quantization, the DCT coefficients for a block are reordered to group together non-zero coefficients, enabling efficient representation of the remaining zero-valued quantized coefficients. The optimum re-ordering path or scan order depends on the distribution of non-zero DCT coefficients. For a typical frame block with a distribution similar to Figure 3.39, a suitable scan order is a zigzag starting from the DC or top-left coefficient. Starting with the DC coefficient, each quantized coefficient is copied into a one-dimensional array in the order shown in Figure 3.42. Non-zero coefficients tend to be grouped together at the start of the re-ordered array, followed by long sequences of zeros.

The zig-zag scan may not be ideal for a field block because of the skewed coefficient distribution, Figure 3.41, and a modified scan order such as Figure 3.43 may be more effective for some field blocks, in which coefficients on the left hand side of the block are scanned before the right hand side.

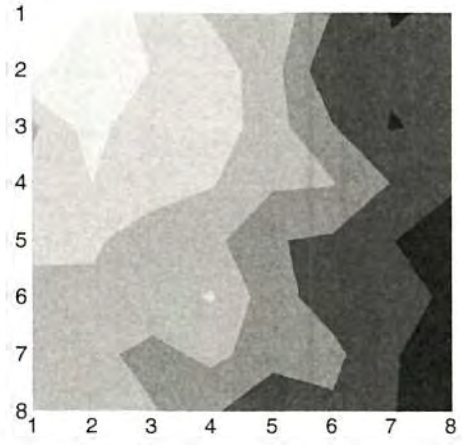


Figure 3.41 8 × 8 DCT coefficient distribution (field)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000080

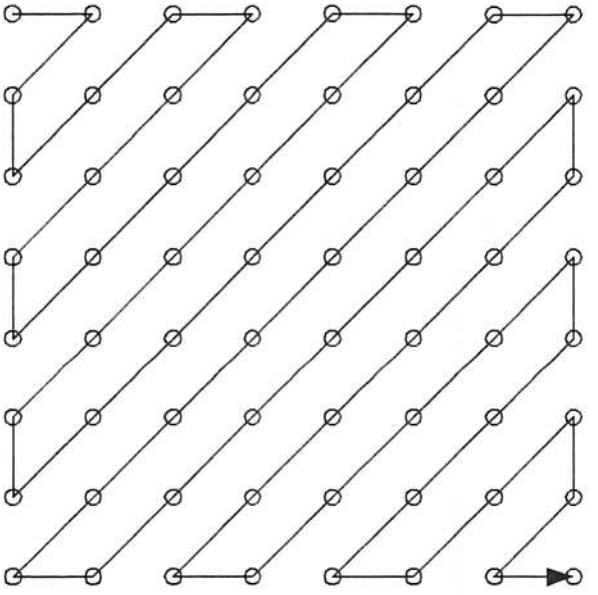


Figure 3.42 Zigzag scan example : frame block

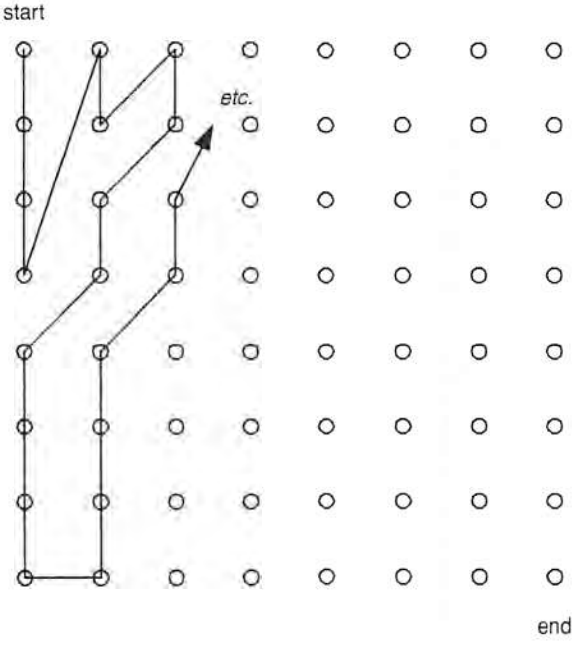


Figure 3.43 Zigzag scan example : field block

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000081

Run-Level Encoding

The output of the re-ordering process is an array that typically contains one or more clusters of non-zero coefficients near the start, followed by strings of zero coefficients. The large number of zero values may be encoded to represent them more compactly. The array of re-ordered coefficients are represented as (run,level) pairs where **run** indicates the number of zeros preceding a non-zero coefficient and **level** indicates the magnitude of the non-zero coefficient.

Example

1. Input array: 16,0,0,−3,5,6,0,0,0,0,−7,...
2. Output values: (0,16),(2,−3),(0,5),(0,6),(4,−7)...
3. Each of these output values (a run-level pair) is encoded as a separate symbol by the entropy encoder.

Higher-frequency DCT coefficients are very often quantized to zero and so a reordered block will usually end in a run of zeros. A special case is required to indicate the final non-zero coefficient in a block. If 'two-dimensional' run-level encoding is used, each run-level pair is encoded as above and a separate code symbol, '**last**', indicates the end of the non-zero values. If 'three-dimensional' run-level encoding is used, each symbol encodes three quantities, **run**, **level** and **last**. In the example above, if −7 is the final non-zero coefficient, the 3-D values are:

(0, 16, 0), (2, −3, 0), (0, 5, 0), (0, 6, 0), (4, −7, 1)

The 1 in the final code indicates that this is the last non-zero coefficient in the block.

3.4.4.2 Wavelet

Coefficient distribution

Figure 3.36 shows a typical distribution of 2D wavelet coefficients. Many coefficients in higher sub-bands, towards the bottom-right of the figure, are near zero and may be quantized to zero without significant loss of image quality. Non-zero coefficients tend to be related to structures in the image; for example, the violin bow appears as a clear horizontal structure in all the horizontal and diagonal sub-bands. When a coefficient in a lower-frequency sub-band is non-zero, there is a strong probability that coefficients in the corresponding position in higher-frequency sub-bands will also be non-zero. We may consider a 'tree' of non-zero quantized coefficients, starting with a 'root' in a low-frequency sub-band. Figure 3.44 illustrates this concept. A single coefficient in the LL band of layer 1 has one corresponding coefficient in each of the other bands of layer 1, i.e. these four coefficients correspond to the same region in the original image. The layer 1 coefficient position maps to four corresponding child coefficient positions in each sub-band at layer 2. Recall that the layer 2 sub-bands have twice the horizontal and vertical resolution of the layer 1 sub-bands.

Zerotree encoding

It is desirable to encode the non-zero wavelet coefficients as compactly as possible prior to entropy coding [vii]. An efficient way of achieving this is to encode each tree of non-zero coefficients starting from the lowest or root level of the decomposition. A coefficient at the

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000082

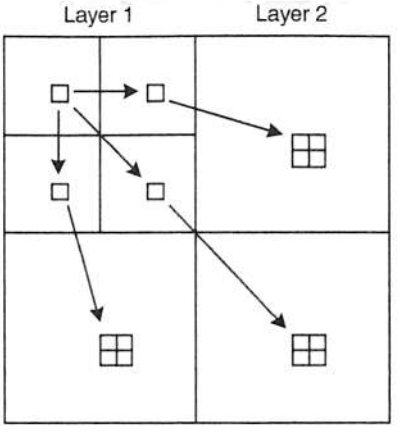


Figure 3.44 Wavelet coefficient and 'children'

lowest layer is encoded, followed by its child coefficients at the next layer up, and so on. The encoding process continues until the tree reaches a zero-valued coefficient. Further children of a zero-valued coefficient are likely to be zero themselves and so the remaining children are represented by a single code that identifies a tree of zeros (**zerotree**). The decoder reconstructs the coefficient map starting from the root of each tree; non-zero coefficients are decoded and reconstructed and when a zerotree code is reached, all remaining 'children' are set to zero. This is the basis of the **embedded zero tree** (EZW) method of encoding wavelet coefficients. An extra possibility is included in the encoding process, where a zero coefficient may be followed by (a) a zero tree, as before or (b) a non-zero child coefficient. Case (b) does not occur very often but reconstructed image quality is slightly improved by catering for the occasional occurrences of case (b).

3.5 Entropy coder

The entropy encoder converts a series of symbols representing elements of the video sequence into a compressed bitstream suitable for transmission or storage. Input symbols may include quantized transform coefficients, run-level or zerotree encoded as described in section 3.4.4, motion vectors with integer or sub-pixel resolution, marker codes that indicate a resynchronization point in the sequence, macroblock headers, picture headers, sequence headers etc and supplementary information, 'side' information that is not essential for correct decoding.

3.5.1 Predictive coding

Certain symbols are highly correlated in local regions of the picture. For example, the average or DC value of neighbouring intra-coded blocks of pixels may be very similar, neighbouring motion vectors may have similar x and y displacements and so on. Coding efficiency can be improved by predicting elements of the current block or macroblock from previously-encoded data and encoding the difference between the prediction and the actual value.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000083

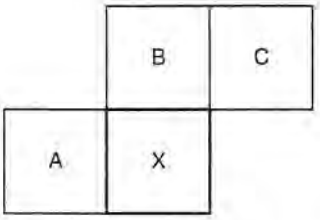


Figure 3.45 Motion vector prediction candidates

The motion vector for a block or macroblock indicates the offset to a prediction reference in a previously encoded frame. Vectors for neighbouring blocks or macroblocks are often correlated because object motion may extend across large regions of a frame. This is especially true for small block sizes, e.g. 4 × 4 block vectors (Figure 3.22) and/or for large moving objects. Compression of the motion vector field may be improved by predicting each motion vector from previously encoded vectors. A simple prediction for the vector of the current macroblock X is the horizontally adjacent macroblock A (Figure 3.45); alternatively three or more previously-coded vectors may be used to predict the vector at macroblock X, e.g. A, B and C in Figure 3.45. The difference between the predicted and actual motion vector, the Motion Vector Difference or MVD, is encoded and transmitted.

The quantization parameter or quantizer step size controls the trade-off between compression efficiency and image quality. In a real-time video CODEC it may be necessary to modify the quantization within an encoded frame, for example to change the compression ratio in order to match the coded bit rate to a transmission channel rate. It is usually sufficient to change the parameter by a small amount between successive coded macroblocks. The modified quantization parameter must be signalled to the decoder and instead of sending a new quantization parameter value, it may be preferable to send a delta or difference value, e.g. +/−1 or +/−2, indicating the change required. Fewer bits are required to encode a small delta value than to encode a completely new quantization parameter.

3.5.2 Variable-length coding

A variable-length encoder maps input symbols to a series of codewords, variable length codes or VLCs. Each symbol maps to a codeword and codewords may have varying length but must each contain an integral number of bits. Frequently-occurring symbols are represented with short VLCs whilst less common symbols are represented with long VLCs. Over a sufficiently large number of encoded symbols this leads to compression of the data.

3.5.2.1 Huffman coding

Huffman coding assigns a VLC to each symbol based on the probability of occurrence of different symbols. According to the original scheme proposed by Huffman in 1952 [viii], it is necessary to calculate the probability of occurrence of each symbol and to construct a set of variable length codewords. This process will be illustrated by two examples.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000084

Table 3.2 Probability of occurrence of motion vectors in sequence 1

Vector	Probability p	$\log_2(1/p)$
−2	0.1	3.32
−1	0.2	2.32
0	0.4	1.32
1	0.2	2.32
2	0.1	3.32

Example 1: Huffman coding, Sequence 1 motion vectors

The motion vector difference data (MVD) for video sequence 1 is required to be encoded. Table 3.2 lists the probabilities of the most commonly-occurring motion vectors in the encoded sequence and their **information content**, $\log_2(1/p)$. To achieve optimum compression, each value should be represented with exactly $\log_2(1/p)$ bits. '0' is the most common value and the probability drops for larger motion vectors. This distribution is representative of a sequence containing moderate motion.

1. Generating the Huffman code tree

To generate a Huffman code table for this set of data, the following iterative procedure is carried out:

1. Order the list of data in increasing order of probability.
2. Combine the two lowest-probability data items into a 'node' and assign the joint probability of the data items to this node.
3. Re-order the remaining data items and node(s) in increasing order of probability and repeat step 2.

The procedure is repeated until there is a single 'root' node that contains all other nodes and data items listed 'beneath' it. This procedure is illustrated in Figure 3.46.

Original list:	The data items are shown as square boxes. Vectors (−2) and (+2) have the lowest probability and these are the first candidates for merging to form node 'A'.
Stage 1:	The newly-created node 'A', shown as a circle, has a probability of 0.2, from the combined probabilities of (−2) and (2). There are now three items with probability 0.2. Choose vectors (−1) and (1) and merge to form node 'B'.
Stage 2:	A now has the lowest probability (0.2) followed by B and the vector 0; choose A and B as the next candidates for merging to form 'C'.
Stage 3:	Node C and vector (0) are merged to form 'D'.
Final tree:	The data items have all been incorporated into a binary 'tree' containing five data values and four nodes. Each data item is a 'leaf' of the tree.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000085

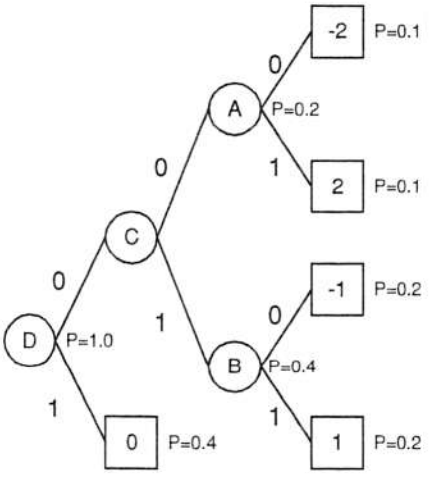


Figure 3.46 Generating the Huffman code tree: Sequence 1 motion vectors

2. Encoding

Each 'leaf' of the binary tree is mapped to a variable-length code. To find this code, the tree is traversed from the root node, D in this case, to the leaf or data item. For every branch, a 0 or 1 is appended to the code, 0 for an upper branch, 1 for a lower branch, shown in the final tree of Figure 3.46, giving the following set of codes (Table 3.3).

Encoding is achieved by transmitting the appropriate code for each data item. Note that once the tree has been generated, the codes may be stored in a look-up table.

High probability data items are assigned short codes, e.g. 1 bit for the most common vector '0'. However, the vectors (−2, 2, −1, 1) are each assigned 3-bit codes despite the fact that –1 and 1 have higher probabilities than –2 and 2. The lengths of the Huffman codes, each an integral number of bits, do not match the ideal lengths given by $\log_2(1/p)$. No code contains any other code as a prefix, which means that, reading from the left-hand bit, each code is uniquely decodable.

For example, the series of vectors (1, 0, −2) would be transmitted as the binary sequence 0111000.

Table 3.3 Huffman codes for sequence 1 motion vectors

Vector	Code	Bits (actual)	Bits (ideal)
0	1	1	1.32
1	011	3	2.32
−1	010	3	2.32
2	001	3	3.32
−2	000	3	3.32

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000086

Table 3.4 Probability of occurrence of motion vectors in sequence 2

Vector	Probability	$\log_2(1/p)$
−2	0.02	5.64
−1	0.07	3.84
0	0.8	0.32
1	0.08	3.64
2	0.03	5.06

3. Decoding

In order to decode the data, the decoder must have a local copy of the Huffman code tree or look-up table. This may be achieved by transmitting the look-up table itself or by sending the list of data and probabilities prior to sending the coded data. Each uniquely-decodeable code is converted back to the original data, for example:

1. 011 is decoded as (1)
2. 1 is decoded as (0)
3. 000 is decoded as (−2).

Example 2: Huffman coding, sequence 2 motion vectors

Repeating the process described above for a second sequence with a different distribution of motion vector probabilities gives a different result. The probabilities are listed in Table 3.4 and note that the zero vector is much more likely to occur in this example, representative of a sequence with little movement.

The corresponding Huffman tree is given in Figure 3.47. The 'shape' of the tree has changed because of the distribution of probabilities and this gives a different set of Huffman codes, shown in Table 3.5. There are still four nodes in the tree, one less than the number of data items (5), as is always the case with Huffman coding.

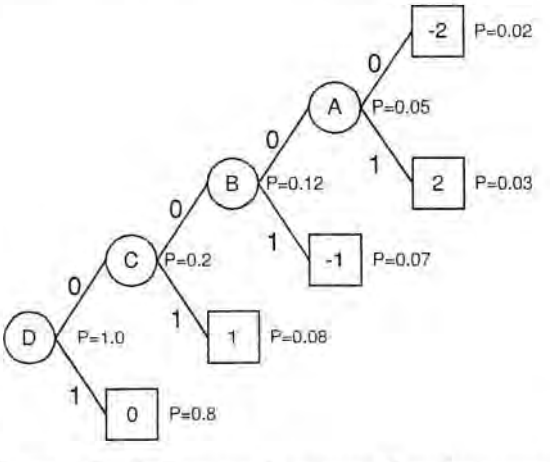


Figure 3.47 Huffman tree for sequence 2 motion vectors

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000087

Table 3.5 Huffman codes for sequence 2 motion vectors

Vector	Code	Bits (actual)	Bits (ideal)
0	1	1	0.32
1	01	2	3.64
−1	001	3	3.84
2	0001	4	5.06
−2	0000	4	5.64

If the probability distributions are accurate, Huffman coding provides a relatively compact representation of the original data. In these examples, the frequently occurring (0) vector is represented efficiently as a single bit. However, to achieve optimum compression, a separate code table is required for each of the two sequences because of their different probability distributions. The loss of potential compression efficiency due to the requirement for integral-length codes is very clear for vector '0' in sequence 2, since the optimum number of bits (information content) is 0.32 but the best that can be achieved with Huffman coding is 1 bit.

3.5.2.2 Pre-calculated Huffman-based coding

The Huffman coding process has two disadvantages for a practical video CODEC. First, the decoder must use the same codeword set as the encoder. This means that the encoder needs to transmit the information contained in the probability table before the decoder can decode the bit stream and this extra overhead reduces compression efficiency, particularly for shorter video sequences. Second, the probability table for a large video sequence, required to generate the Huffman tree, cannot be calculated until after the video data is encoded which may introduce an unacceptable delay into the encoding process. For these reasons, image and video coding standards define sets of codewords based on the probability distributions of 'generic' video material. The following two examples of pre-calculated VLC tables are taken from MPEG-4 Visual (Simple Profile).

Transform Coefficients (TCOEF)

MPEG-4 Visual uses 3-D coding of quantized coefficients in which each codeword represents a combination of (run, level, last). A total of 102 specific combinations of (run, level, last) have VLCs assigned to them and 26 of these codes are shown in Table 3.6.

A further 76 VLCs are defined, each up to 13 bits long. The last bit of each codeword is the sign bit 's', indicating the sign of the decoded coefficient, where 0=positive and 1=negative. Any (run, level, last) combination that is not listed in the table is coded using an escape sequence, a special ESCAPE code (0000011) followed by a 13-bit fixed length code describing the values of run, level and last.

Some of the codes shown in Table 3.6 are represented in 'tree' form in Figure 3.48. A codeword containing a run of more than eight zeros is not valid, hence any codeword starting with 000000000... indicates an error in the bitstream or possibly a start code, which begins with a long sequence of zeros, occurring at an unexpected position in the sequence. All other sequences of bits can be decoded as valid codes. Note that the smallest codes are allocated to

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000088

Table 3.6 MPEG-4 Visual Transform Coefficient (TCOEF) VLCs : partial, all codes < 9 bits

Last	Run	Level	Code
0	0	1	10s
0	1	1	110s
0	2	1	1110s
0	0	2	1111s
1	0	1	0111s
0	3	1	01101s
0	4	1	01100s
0	5	1	01011s
0	0	3	010101s
0	1	2	010100s
0	6	1	010011s
0	7	1	010010s
0	8	1	010001s
0	9	1	010000s
1	1	1	001111s
1	2	1	001110s
1	3	1	001101s
1	4	1	001100s
0	0	4	0010111s
0	10	1	0010110s
0	11	1	0010101s
0	12	1	0010100s
1	5	1	0010011s
1	6	1	0010010s
1	7	1	0010001s
1	8	1	0010000s
ESCAPE			0000011s
...	...	...	...

short runs and small levels since these occur most frequently, e.g. code '10' represents a run of 0 and a level of +/−1.

Motion Vector Difference (MVD)

Differentially coded motion vectors (MVD) are each encoded as a pair of VLCs, one for the x-component and one for the y-component. Part of the table of VLCs is shown in Table 3.7. A further 49 codes, 8–13 bits long, are not shown here. Note that the shortest codes represent small motion vector differences, e.g. MVD=0 is represented by a single bit code '1'.

These code tables are clearly similar to 'true' Huffman codes since each symbol is assigned a unique codeword, common symbols are assigned shorter codewords and, within a table, no codeword is the prefix of any other codeword. The main differences from 'true' Huffman coding are (a) the codewords are pre-calculated based on 'generic' probability distributions and (b) in the case of TCOEF, only 102 commonly-occurring symbols have defined codewords with any other symbol encoded using a fixed-length code.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000089

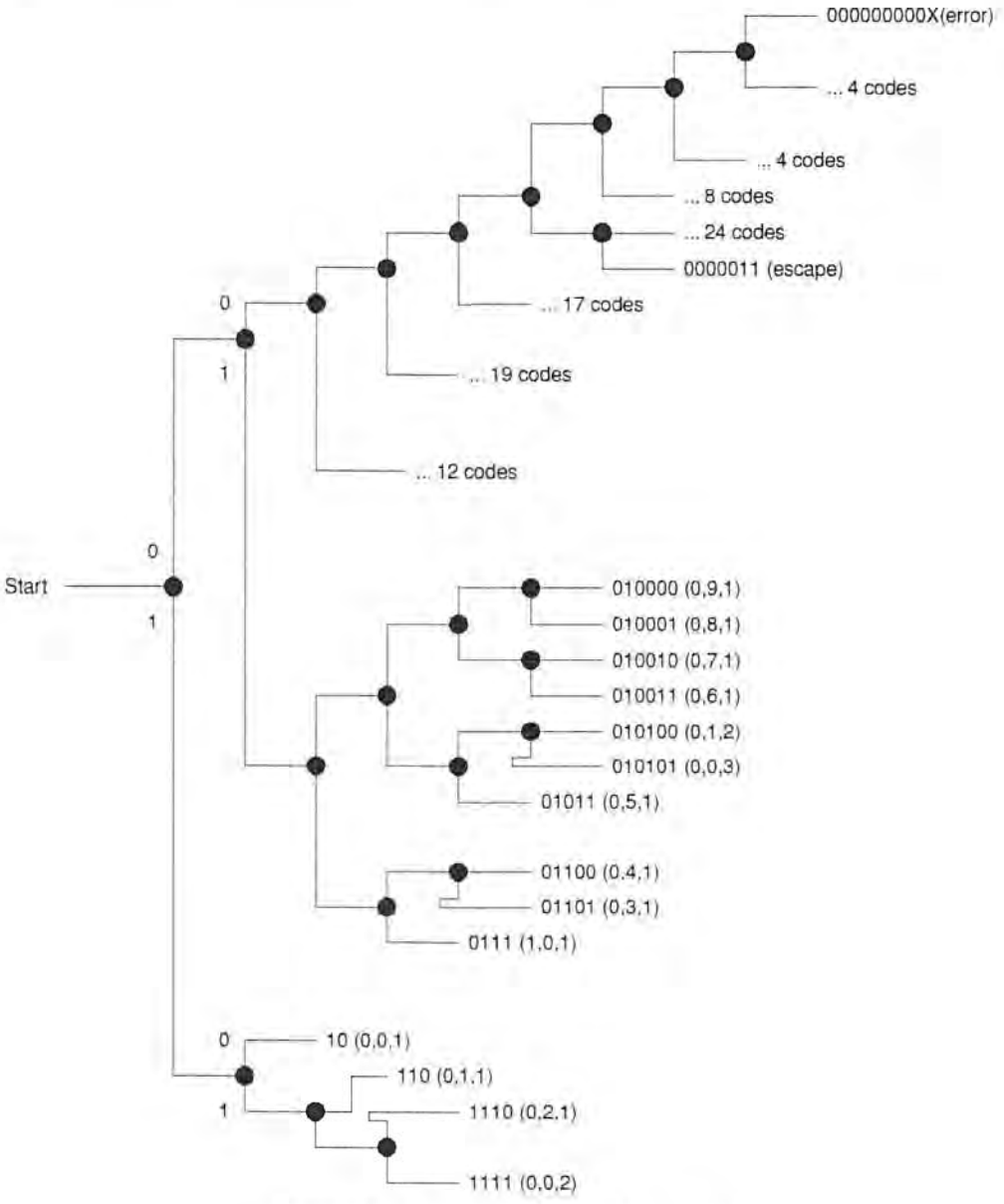


Figure 3.48 MPEG4 TCOEF VLCs (partial)

3.5.2.3 Other variable-length codes

As well as Huffman and Huffman-based codes, a number of other families of VLCs are of interest in video coding applications. One serious disadvantage of Huffman-based codes for transmission of coded data is that they are sensitive to transmission errors. An error in a sequence of VLCs may cause a decoder to lose synchronization and fail to decode subsequent codes correctly, leading to spreading or propagation of an error in a decoded

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000090

Table 3.7 MPEG4 Motion Vector Difference (MVD) VLCs

MVD	Code
0	1
+0.5	010
−0.5	011
+1	0010
−1	0011
+1.5	00010
−1.5	00011
+2	0000110
−2	0000111
+2.5	00001010
−2.5	00001011
+3	00001000
−3	00001001
+3.5	00000110
−3.5	00000111
...	...

sequence. Reversible VLCs (RVLCs) that can be successfully decoded in either a forward or a backward direction can dramatically improve decoding performance when errors occur. A drawback of pre-defined code tables such as Table 3.6 and Table 3.7 is that both encoder and decoder must store the table in some form. An alternative approach is to use codes that can be generated automatically 'on the fly' if the input symbol is known. Exponential Golomb codes (Exp-Golomb) fall into this category and are described in Chapter 7.

3.5.3 Arithmetic coding

The variable length coding schemes described in section 3.5.2 share the fundamental disadvantage that assigning a codeword containing an integral number of bits to each symbol is sub-optimal, since the optimal number of bits for a symbol depends on the information content and is usually a fractional number. Compression efficiency of variable length codes is particularly poor for symbols with probabilities greater than 0.5 as the best that can be achieved is to represent these symbols with a single-bit code.

Arithmetic coding provides a practical alternative to Huffman coding that can more closely approach theoretical maximum compression ratios [ix]. An arithmetic encoder converts a sequence of data symbols into a single fractional number and can approach the optimal fractional number of bits required to represent each symbol.

Example

Table 3.8 lists the five motion vector values (−2, −1, 0, 1, 2) and their probabilities from Example 1 in section 3.5.2.1. Each vector is assigned a **sub-range** within the range 0.0 to 1.0, depending on its probability of occurrence. In this example, (−2) has a probability of 0.1 and is given the subrange 0–0.1, i.e. the first 10 per cent of the total range 0 to 1.0. (−1) has a probability of 0.2 and is given the next 20 per cent of the total range, i.e. the sub-range 0.1–0.3.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000091

After assigning a sub-range to each vector, the total range 0–1.0 has been divided amongst the data symbols (the vectors) according to their probabilities (Figure 3.49).

Table 3.8 Motion vectors, sequence 1: probabilities and sub-ranges

Vector	Probability	$\log_2(1/P)$	Sub-range
−2	0.1	3.32	0–0.1
−1	0.2	2.32	0.1–0.3
0	0.4	1.32	0.3–0.7
1	0.2	2.32	0.7–0.9
2	0.1	3.32	0.9–1.0

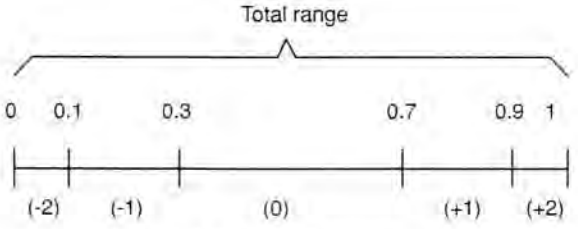


Figure 3.49 Sub-range example

Encoding procedure for vector sequence (0, −1, 0, 2).

Encoding procedure	Range (L → H)	Symbol	Sub-range (L → H)	Notes
1. Set the initial range	0 → 1.0			
2. For the first data symbol, find the corresponding sub-range (Low to High).		(0)	0.3 → 0.7	
3. Set the new range (1) to this sub-range	0.3 → 0.7			
4. For the next data symbol, find the sub-range L to H		(−1)	0.1 → 0.3	This is the sub-range within the interval 0 - 1
5. Set the new range (2) to this sub-range within the previous range	0.34 → 0.42			0.34 is 10% of the range; 0.42 is 30% of the range
6. Find the next sub-range		(0)	0.3 → 0.7	
7. Set the new range (3) within the previous range	0.364 → 0.396			0.364 is 30% of the range; 0.396 is 70% of the range
8. Find the next sub-range		(2)	0.9 → 1.0	
9. Set the new range (4) within the previous range	0.3928 → 0.396			0.3928 is 90% of the range; 0.396 is 100% of the range

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000092

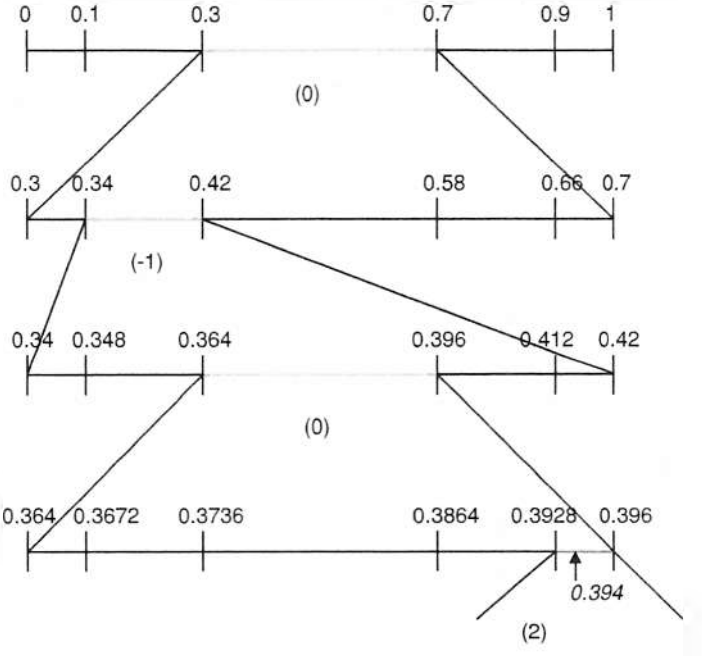


Figure 3.50 Arithmetic coding example

Each time a symbol is encoded, the range (L to H) becomes progressively smaller. At the end of the encoding process, four steps in this example, we are left with a final range (L to H). The entire sequence of data symbols can be represented by transmitting any fractional number that lies within this final range. In the example above, we could send any number in the range 0.3928 to 0.396: for example, 0.394. Figure 3.50 shows how the initial range (0, 1) is progressively partitioned into smaller ranges as each data symbol is processed. After encoding the first symbol, vector 0, the new range is (0.3, 0.7). The next symbol (vector −1) selects the sub-range (0.34, 0.42) which becomes the new range, and so on. The final symbol, vector +2, selects the sub-range (0.3928, 0.396) and the number 0.394 falling within this range is transmitted. 0.394 can be represented as a fixed-point fractional number using 9 bits, so our data sequence (0, −1, 0, 2) is compressed to a 9-bit quantity.

Decoding procedure

Decoding procedure	Range	Sub-range	Decoded symbol
1. Set the initial range	0 → 1		
2. Find the sub-range in which the received number falls. This indicates the first data symbol.		0.3 → 0.7	(0)
3. Set the new range (1) to this sub-range	0.3 → 0.7		

(*Continued*)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000093

Decoding procedure	Range	Sub-range	Decoded symbol
4. Find the sub-range **of the new range** in which the received number falls. This indicates the second data symbol.		0.34 → 0.42	(−1)
5. Set the new range (2) to this sub-range within the previous range	0.34 → 0.42		
6. Find the sub-range in which the received number falls and decode the third data symbol.		0.364 → 0.396	(0)
7. Set the new range (3) to this sub-range within the previous range	0.364 → 0.396		
8. Find the sub-range in which the received number falls and decode the fourth data symbol.		0.3928 → 0.396	

The principal advantage of arithmetic coding is that the transmitted number, 0.394 in this case, which may be represented as a fixed-point number with sufficient accuracy using 9 bits, is not constrained to an integral number of bits for each transmitted data symbol. To achieve optimal compression, the sequence of data symbols should be represented with:

$$\log_2(1/P_0) + \log_2(1/P_{-1}) + \log_2(1/P_0) + \log_2(1/P_2) \text{ bits} = 8.28 \text{ bits}$$

In this example, arithmetic coding achieves 9 bits, which is close to optimum. A scheme using an integral number of bits for each data symbol such as Huffman coding is unlikely to come so close to the optimum number of bits and in general, arithmetic coding can out-perform Huffman coding.

3.5.3.1 Context-based Arithmetic Coding

Successful entropy coding depends on accurate models of symbol probability. Context-based Arithmetic Encoding (CAE) uses local spatial and/or temporal characteristics to estimate the probability of a symbol to be encoded. CAE is used in the JBIG standard for bi-level image compression [x] and has been adopted for coding binary shape 'masks' in MPEG-4 Visual and entropy coding in certain Profiles of H.264 (Chapter 7).

3.6 The hybrid DPCM/DCT video CODEC model

The major video coding standards released since the early 1990s have been based on the same generic design or model of a video CODEC that incorporates a motion estimation and compensation first stage, sometimes described as DPCM, a transform stage and an entropy encoder. The model is often described as a hybrid DPCM/DCT CODEC. Any CODEC that is compatible with H.261, H.263, MPEG-1, MPEG-2, MPEG-4 Visual, H.264/AVC or VC-1 has to implement a similar set of basic coding and decoding functions, although there are many differences of detail between the standards and between implementations.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000094

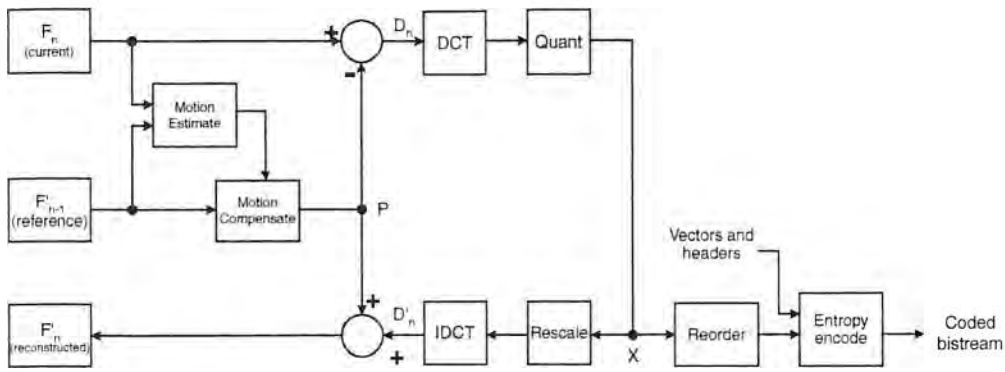


Figure 3.51 DPCM/DCT video encoder

Figure 3.51 and Figure 3.52 show the basic DPCM/DCT hybrid encoder and decoder. In the encoder, video frame n (F_n) is processed to produce a coded or compressed bitstream. In the decoder, the compressed bitstream, shown at the right of the figure, is decoded to produce a reconstructed video frame F'_n. The reconstructed output frame is not usually identical to the source frame. The figures have been deliberately drawn to highlight the common elements within encoder and decoder. Most of the functions of the decoder are actually contained within the encoder, the reason for which will be explained later.

Encoder data flow

There are two main data flow paths in the encoder, left to right (encoding) and right to left (reconstruction). The encoding flow is as follows:

1. An input video frame F_n is presented for encoding and is processed in units of a macroblock, corresponding to a 16 × 16 pixel region of the video image.
2. F_n is compared with a **reference** frame, for example the previous encoded frame (F'_{n-1}). A motion estimation function finds a 16 × 16 region in F'_{n-1} that 'matches' the current macroblock in F_n. The offset between the current macroblock position and the chosen reference region is a motion vector MV.
3. Based on the chosen motion vector MV, a motion compensated prediction P is generated, the 16 × 16 region selected by the motion estimator. P may consist of interpolated sub-pixel data.
4. P is subtracted from the current macroblock to produce a residual or difference macroblock D.

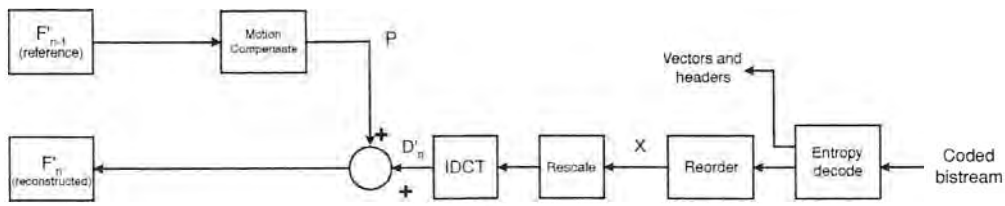


Figure 3.52 DPCM/DCT video decoder

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000095

5. D is transformed using the DCT. Typically, D is split into 8 × 8 or 4 × 4 sub-blocks and each sub-block is transformed separately.
6. Each sub-block is quantized (X).
7. The DCT coefficients of each sub-block are reordered and run-level coded.
8. Finally, the coefficients, motion vector and associated header information for each macroblock are entropy encoded to produce the compressed bitstream.

The reconstruction data flow is as follows:

1. Each quantized macroblock X is rescaled and inverse transformed to produce a decoded residual D'. Note that the non-reversible quantization process means that D' is not identical to D, i.e. distortion has been introduced.
2. The motion compensated prediction P is added to the residual D' to produce a reconstructed macroblock. The reconstructed macroblocks are combined to produce reconstructed frame F'_n.

After encoding a complete frame, the reconstructed frame F'_n may be used as a reference frame for the next encoded frame F_{n+1}.

Decoder data flow

1. A compressed bitstream is entropy decoded to extract coefficients, motion vector and header for each macroblock.
2. Run-level coding and reordering are reversed to produce a quantized, transformed macroblock X.
3. X is rescaled and inverse transformed to produce a decoded residual D'.
4. The decoded motion vector is used to locate a 16 × 16 region in the decoder's copy of the previous (reference) frame F'_{n-1}. This region becomes the motion compensated prediction P.
5. P is added to D' to produce a reconstructed macroblock. The reconstructed macroblocks are saved to produce decoded frame F'_n.

After a complete frame is decoded, F'_n is ready to be displayed and may also be stored as a reference frame for the next decoded frame F'_{n+1}.

It is clear from the figures and from the above explanation that the encoder includes a decoding path : rescale, IDCT, reconstruct. This is necessary to ensure that the encoder and decoder use identical reference frames F'_{n-1} for motion compensated prediction.

Example

A 25-Hz video sequence in CIF format, with 352 × 288 luminance samples and 176 × 144 red/blue chrominance samples per frame, is encoded and decoded using a DPCM/DCT CODEC. Figure 3.53 shows a CIF video frame (F_n) that is to be encoded and Figure 3.54 shows the reconstructed previous frame F'_{n-1}. Note that F'_{n-1} has been encoded and decoded and shows some distortion. The difference between F_n and F'_{n-1} **without** motion compensation (Figure 3.55) clearly still contains significant energy, especially around the edges of moving areas.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000096



Figure 3.53 Input frame F_n



Figure 3.54 Reconstructed reference frame F'_{n-1}

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000097



Figure 3.55 Residual F_n-F'_{n-1} : no motion compensation

Motion estimation is carried out with a 16 × 16 block size and half-pixel accuracy, producing the set of vectors shown in Figure 3.56, superimposed on the current frame for clarity. Many of the vectors are zero and are shown as dots, which means that the best match for the current macroblock is in the same position in the reference frame. Around moving areas, the vectors point in the direction that blocks have moved **from**. E.g. the man on the left is walking to the left; the vectors therefore point to the **right**, i.e. where he has come from. Some of the vectors do not appear to correspond to 'real' movement, e.g. on the surface of the table, but indicate simply that the best match is not at the same position in the reference frame. 'Noisy' vectors like these often occur in homogeneous regions of the picture, where there are no clear object features in the reference frame.

The motion-compensated reference frame (Figure 3.57) is the reference frame 'reorganized' according to the motion vectors. For example, note that the walking person has been moved to the left to provide a better match for the same person in the current frame and that the hand of the left-most person has been moved down to provide an improved match. Subtracting the motion compensated reference frame from the current frame gives the motion-compensated residual in Figure 3.58 in which the energy has clearly been reduced, particularly around moving areas.

Figure 3.59 shows a macroblock from the original frame, taken from around the head of the figure on the right, and Figure 3.60 the luminance residual after motion compensation. Applying a 2D DCT to the top-right 8 × 8 block of luminance samples (Table 3.9) produces the DCT coefficients listed in Table 3.10. The magnitude of each coefficient is plotted in Figure 3.61; note that the larger coefficients are clustered around the top-left (DC) coefficient.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000098



Figure 3.56 16 × 16 motion vectors superimposed on frame



Figure 3.57 Motion compensated reference frame

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000099



Figure 3.58 Motion compensated residual frame

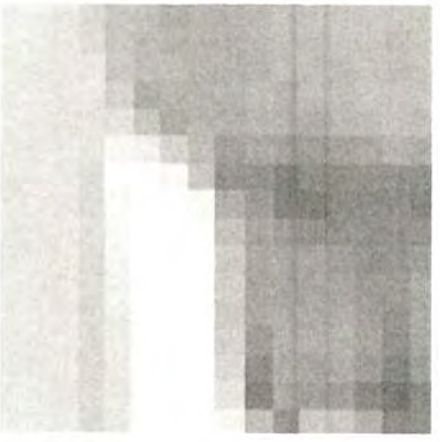


Figure 3.59 Original macroblock : luminance

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000100



Figure 3.60 Residual macroblock : luminance

Table 3.9 Residual luminance samples : top-right 8 × 8 block

−4	−4	−1	0	1	1	0	−2
1	2	3	2	−1	−3	−6	−3
6	6	4	−4	−9	−5	−6	−5
10	8	−1	−4	−6	−1	2	4
7	9	−5	−9	−3	0	8	13
0	3	−9	−12	−8	−9	−4	1
−1	4	−9	−13	−8	−16	−18	−13
14	13	−1	−6	3	−5	−12	−7

Table 3.10 DCT coefficients

−13.50	20.47	20.20	2.14	−0.50	−10.48	−3.50	−0.62
10.93	−11.58	−10.29	−5.17	−2.96	10.44	4.96	−1.26
−8.75	9.22	−17.19	2.26	3.83	−2.45	1.77	1.89
−7.10	−17.54	1.24	−0.91	0.47	−0.37	−3.55	0.88
19.00	−7.20	4.08	5.31	0.50	0.18	−0.61	0.40
−13.06	3.12	−2.04	−0.17	−1.19	1.57	−0.08	−0.51
1.73	−0.69	1.77	0.78	−1.86	1.47	1.19	0.42
−1.99	−0.05	1.24	−0.48	−1.86	−1.17	−0.21	0.92

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000101

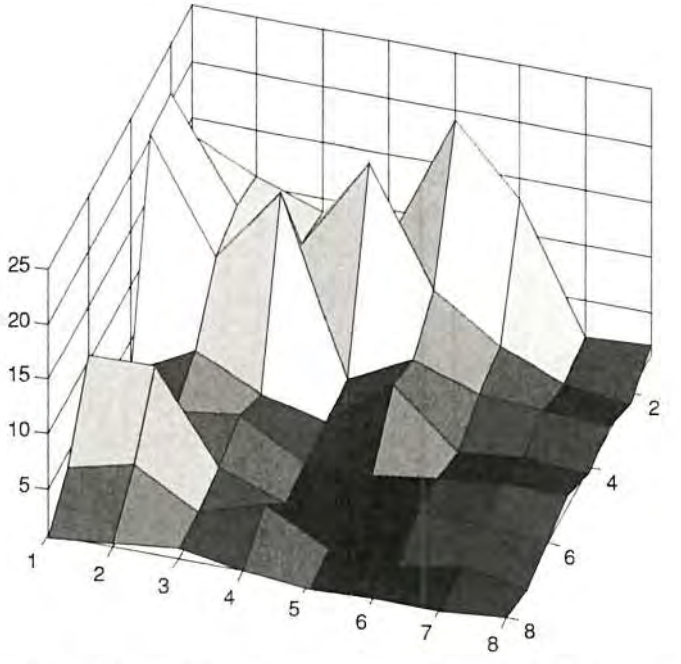


Figure 3.61 DCT coefficient magnitudes : top-right 8×8 block

A simple forward quantizer is applied:

$$\text{Qcoeff} = \text{round}(\text{coeff}/\text{Qstep})$$

where Qstep is the quantizer step size, 12 in this example. Small-valued coefficients become zero in the quantized block (Table 3.11) and the non-zero outputs are clustered around the top-left (DC) coefficient.

The quantized block is re-ordered in a zigzag scan starting at the top-left to produce a linear array:

$$-1,\ 2,\ 1,\ -1,\ -1,\ 2,\ 0,\ -1,\ 1,\ -1,\ 2,\ -1,\ -1,\ 0,$$
$$0,\ -1,\ 0,\ 0,\ 0,\ -1,\ -1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ \ldots,$$

Table 3.11 Quantized coefficients

−1	2	2	0	0	−1	0	0
1	−1	−1	0	0	1	0	0
−1	1	−1	0	0	0	0	0
−1	−1	0	0	0	0	0	0
2	−1	0	0	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

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000102

Table 3.12 Variable length coding example

Run, Level, Last	VLC including sign bit
(0,−1, 0)	101
(0,2, 0)	11100
(0,1, 0)	100
(0,−1, 0)	101
(0,−1, 0)	101
(0,2, 0)	11100
(1,−1, 0)	1101
...	...
(5,1, 1)	00100110

This array is processed to produce a series of (zero run, level) pairs:

$$(0, -1)(0, 2)(0, 1)(0, -1)(0, -1)(0, 2)(1, -1)(0, 1)(0, -1)$$
$$(0, 2)(0, -1)(0, -1)(2, -1)(3, -1)(0, -1)(5, 1)(\text{EOB})$$

'EOB' (End Of Block) indicates that the remainder of the coefficients are zero.

Each (run, level) pair is encoded as a VLC. Using the MPEG-4 Visual TCOEF table (Table 3.6), the VLCs shown in Table 3.12 are produced.

The final VLC signals that LAST=1, indicating that this is the end of the block. The motion vector for this macroblock is (0, 1), i.e. the vector points downwards. The predicted vector based on neighbouring macroblocks is (0,0) and so the motion vector difference values are MVDx=0, MVDy=+1. Using the MPEG4 MVD table (Table 3.7), these are coded as (1) and (0010) respectively.

The macroblock is transmitted as a series of VLCs, including a macroblock header, motion vector difference (X, Y) and transform coefficients (TCOEF) for each 8 × 8 block.

At the decoder, the VLC sequence is decoded to extract header parameters, MVDx and MVDy and (run,level) pairs for each block. The 64-element array of reordered coefficients is reconstructed by inserting (run) zeros before every (level). The array is then ordered to produce an 8 × 8 block identical to Table 3.11. The quantized coefficients are rescaled using:

$$\text{Rcoeff} = \text{Qstep}.\text{Qcoeff}$$

Where Qstep=12 as before, to produce the block of coefficients shown in Table 3.13. Note that the block is significantly different from the original DCT coefficients (Table 3.10) due to

Table 3.13 Rescaled coefficients

−12	24	24	0	0	−12	0	0
12	−12	−12	0	0	12	0	0
−12	12	−12	0	0	0	0	0
−12	−12	0	0	0	0	0	0
24	−12	0	0	0	0	0	0
−12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000103

Table 3.14 Decoded residual luminance samples

−3	−3	−1	1	−1	−1	−1	−3
5	3	2	0	−3	−4	−5	−6
9	6	1	−3	−5	−6	−5	−4
9	8	1	−4	−1	1	4	10
7	8	−1	−6	−1	2	5	14
2	3	−8	−15	−11	−11	−11	−2
2	5	−7	−17	−13	−16	−20	−11
12	16	3	−6	−1	−6	−11	−3

the quantization process. An Inverse DCT is applied to create a decoded residual block (Table 3.14) which is similar but not identical to the original residual block (Table 3.9). The original and decoded residual blocks are plotted side by side in Figure 3.62 and it is clear that the decoded block has less high-frequency variation because of the loss of high-frequency DCT coefficients through quantization.

The decoder forms its own predicted motion vector based on previously decoded vectors and recreates the original motion vector (0, 1). Using this vector, together with its own copy of the previously decoded frame F'_{n-1}, the decoder reconstructs the macroblock. The complete decoded frame is shown in Figure 3.63. Because of the quantization process, some distortion has been introduced, for example around detailed areas such as the faces and the equations on the whiteboard and there are some obvious edges along 8 × 8 block boundaries. The complete sequence was compressed by around 300 times, i.e. the coded sequence occupies less than 1/300 the size of the uncompressed video, and so significant compression was achieved at the expense of relatively poor image quality.

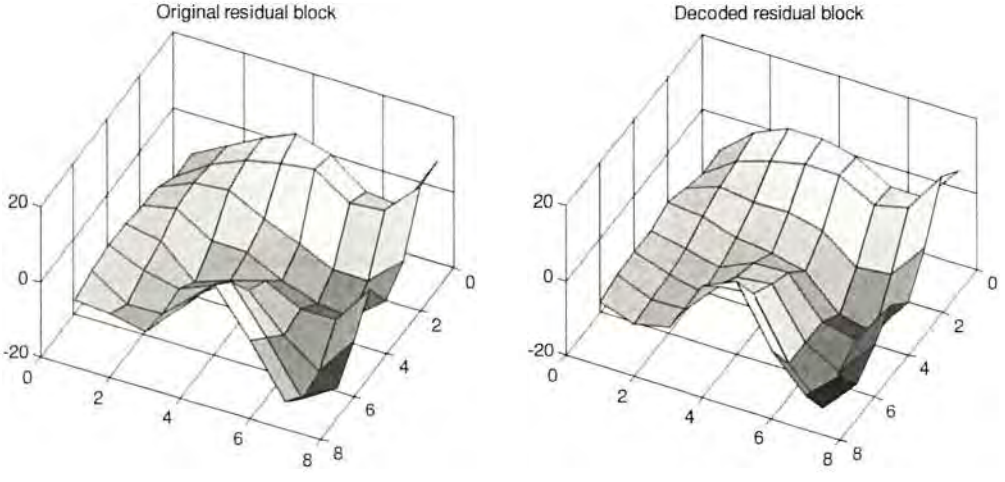


Figure 3.62 Comparison of original and decoded residual blocks

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000104



Figure 3.63 Decoded frame F'_n

3.7 Summary

The video coding tools described in this chapter, motion compensated prediction, transform coding, quantization and entropy coding, form the basis of the reliable and effective coding model that has dominated the field of video compression for over 10 years. This coding model is at the heart of the H.264/AVC standard. The next chapter introduces the main features of H.264/AVC and the standard is discussed in detail in following chapters.

3.8 References

i. Information technology – lossless and near-lossless compression of continuous-tone still images: Baseline, ISO/IEC 14495-1:2000 ('JPEG-LS').

ii. B. Horn and B. G. Schunk, 'Determining Optical Flow', *Artificial Intelligence*, 17:185–203, 1981.

iii. T. Wedi, 'Adaptive Interpolation Filters and High Resolution Displacements for Video Coding', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp 484–491, April 2006.

iv. K. R. Rao and P. Yip, *'Discrete Cosine Transform*. Academic Press, 1990.

v. S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

vi. N. Nasrabadi and R. King, 'Image coding using vector quantization: a review', *IEEE Trans. Communications*, vol. 36, no. 8, August 1988.

vii. W. A. Pearlman, 'Trends of tree-based, set-partitioned compression techniques in still and moving image systems', Proc. International Picture Coding Symposium, Seoul, April 2001.

viii. D. Huffman, 'A method for the construction of minimum redundancy codes', Proceedings of the IRE, vol. 40, pp.1098–1101, 1952.

ix. I. Witten, R. Neal and J. Cleary, 'Arithmetic coding for data compression', Communications of the ACM, 30(6), June 1987.

x. Information technology – coded representation of picture and audio information – progressive bi-level image compression, ITU-T Rec. T.82 ('JBIG').

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000105

4

What is H.264?

4.1 Introduction

This chapter introduces H.264 from a number of perspectives. First, we address the question – what is H.264? H.264 can mean different things from different viewpoints. It is an industry standard; it defines a format for compressed video data; it provides a set of tools that can be used in a variety of ways to compress and communicate visual information; it is a stage in an evolving series of standardized methods for video compression. We look at the basic operations carried out within an H.264 encoder (compressor) and decoder (decompressor). We look in more depth inside H.264, examining what is contained within the H.264 standard, what are the main sets of tools and components, presenting an outline of the H.264 compression syntax or format and looking in more detail at the flow of data through a typical H.264 codec. Finally we discuss some of the practical issues – just how good is H.264 at doing its job; how is H.264 used in 'real' applications.

4.2 What is H.264?

H.264 Advanced Video Coding is an industry standard for video coding, but it is also a popular format for coded video, a set of tools for video compression and a stage in a continuously evolving digital video communication landscape. This section introduces some of these 'views' of H.264/AVC.

4.2.1 A video compression format

H.264 is a method and format for video compression, the process of converting digital video into a format that takes up less capacity when it is stored or transmitted. Video compression or video coding is an essential technology for applications such as digital television, DVD-Video, mobile TV, videoconferencing and internet video streaming. Standardizing video compression makes it possible for products from different manufacturers such as encoders, decoders and storage media to inter-operate. An **encoder** converts video into a compressed format and a **decoder** converts compressed video back into an uncompressed format.

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000106

In a typical application of H.264 such as remote surveillance, video from a camera is encoded or compressed using H.264 to produce an H.264 bitstream. This is sent across a network to a decoder which reconstructs a version of the source video, Figure 1.1. Figure 1.1 shows how source video, uncompressed video material, is encoded by an encoder and stored on a server. The compressed video, a sequence of bits, is transmitted to one or more clients. This example shows clients with widely varying display capabilities from a full-sized TV display to a mobile screen and this highlights a current challenge in video coding – how to cater for receivers with different capabilities. In a two-way 'conversational' video application such as videoconferencing, Apple iChat, Skype Video, etc, both ends of the system require an encoder to compress video from the local camera and a decoder to decompress video from the remote camera, Figure 1.2.

4.2.2 *An industry standard*

Recommendation H.264: Advanced Video Coding is a document co-published by two international standards bodies, the ITU-T (International Telecommunication Union) and the ISO/IEC (International Organisation for Standardisation/International Electrotechnical Commission) [i]. It defines a format or syntax for compressed video and a method for decoding this syntax to produce a displayable video sequence. The standard document does not actually specify how to encode digital video – this is left to the manufacturer of a video encoder – but in practice the encoder is likely to mirror the steps of the decoding process. Figure 4.1 shows the encoding and decoding processes and highlights the parts that are covered by the H.264 standard.

The H.264/AVC standard was first published in 2003, with several revisions and updates published since then. It builds on the concepts of earlier standards such as MPEG-2 and MPEG-4 Visual and offers the potential for better compression efficiency, i.e. better-quality compressed video, and greater flexibility in compressing, transmitting and storing video.

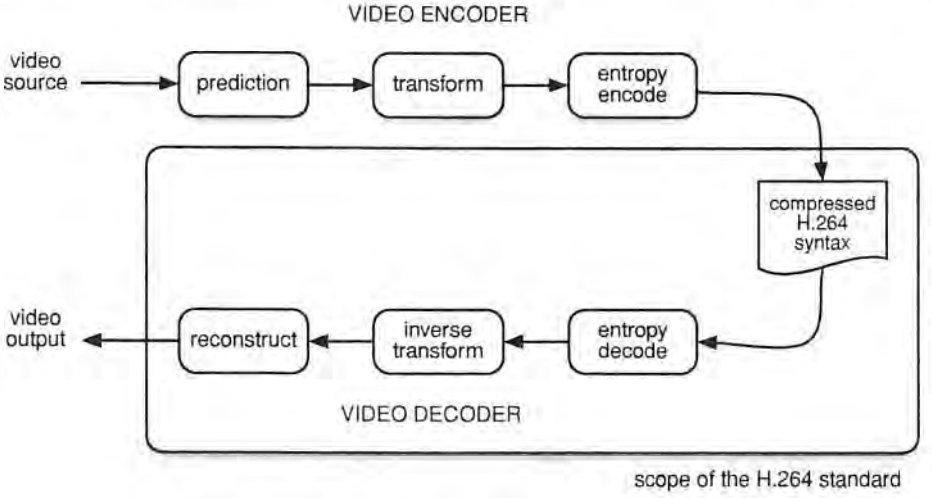


Figure 4.1 The H.264 video coding and decoding process

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000107

4.2.3 A toolkit for video compression

H.264/AVC describes a set of tools or methods for video compression. The standard specifies how video coded with these tools should be represented and decoded. A video encoder may choose which tools to use and how to apply them to the current video sequence, with some constraints. An H.264-compliant decoder must be capable of using a defined sub-set of tools, known as a **profile**.

4.2.4 Better video compression

One of the most important drivers for the standardization of H.264 and its subsequent adoption by industry is its improved performance compared with earlier standards. The benchmark for mass-market applications such as digital TV and consumer video storage on DVD-Video is the earlier MPEG-2 standard [ii]. H.264 offers significantly better compression performance than MPEG-2 Visual. Using H.264 it is possible to compress video into a much smaller number of bits than using MPEG-2, for the same video resolution and image quality. This means, for example, that much more video material can be stored on a disk or transmitted over a broadcast channel by using the H.264 format.

4.3 How does an H.264 codec work?

An H.264 video encoder carries out prediction, transforming and encoding processes (Figure 4.1) to produce a compressed H.264 bitstream. An H.264 video decoder carries out the complementary processes of decoding, inverse transform and reconstruction to produce a decoded video sequence. As Figure 4.2 shows, a sequence of original video frames or fields is

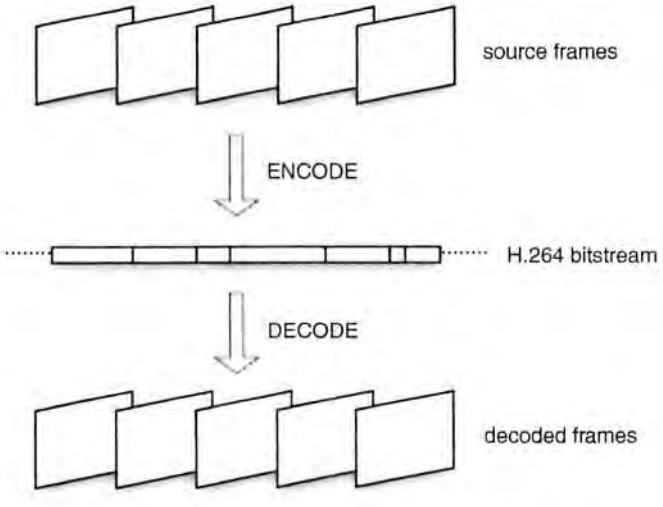


Figure 4.2 Video coding: source frames, encoded bitstream, decoded frames

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000108

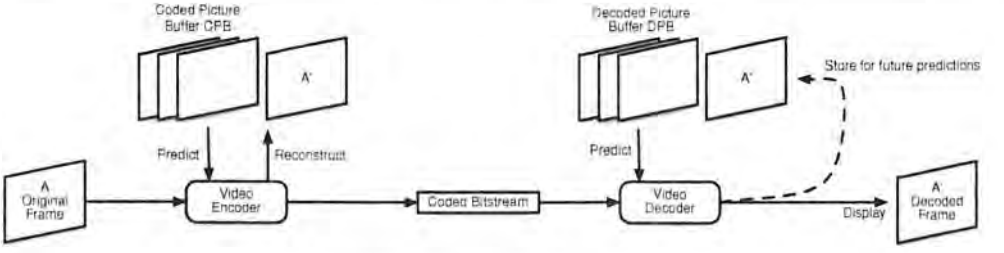


Figure 4.3 Video codec: high level view

encoded into the H.264 format, a series of bits that represents the video in compressed form. This compressed bitstream is stored or transmitted and can be decoded to reconstruct the video sequence. The decoded version is, in general, not identical to the original sequence because H.264 is a lossy compression format, i.e. some picture quality is lost during compression.

A frame or field to be coded, e.g. Frame A in Figure 4.3, is processed by an H.264-compatible video encoder. As well as coding and sending the frame as part of the coded bitstream or coded file, the encoder reconstructs the frame, i.e. creates a copy of the decoded frame A' that will eventually be produced by the decoder. This reconstructed copy may be stored in a coded picture buffer, CPB, and used during the encoding of further frames. The decoder receives the coded bitstream and decodes frame A' for display or further processing. At the same time, the decoder may store a copy of frame A' in a decoded picture buffer, DPB, to be used during the decoding of further frames.

The structure of a typical H.264 codec is shown in Figure 4.4 (encoder) and Figure 4.5 (decoder) in slightly more detail. Data is processed in units of a **macroblock** (MB) corresponding to 16 × 16 displayed pixels. In the encoder, a prediction macroblock is generated and subtracted from the current macroblock to form a residual macroblock; this is transformed, quantized and encoded. In parallel, the quantized data are re-scaled and inverse transformed and added to the prediction macroblock to reconstruct a coded version of the frame which is stored for later predictions. In the decoder, a macroblock is decoded, re-scaled and inverse transformed to form a decoded residual macroblock. The decoder generates the same

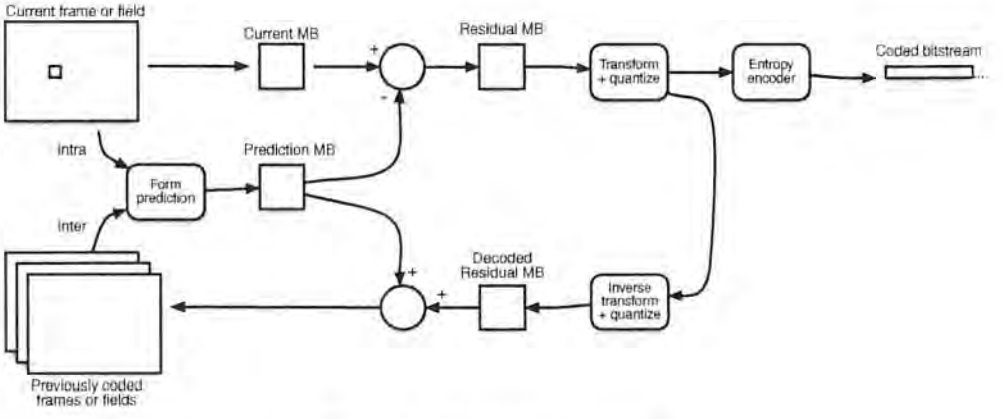


Figure 4.4 Typical H.264 encoder

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000109

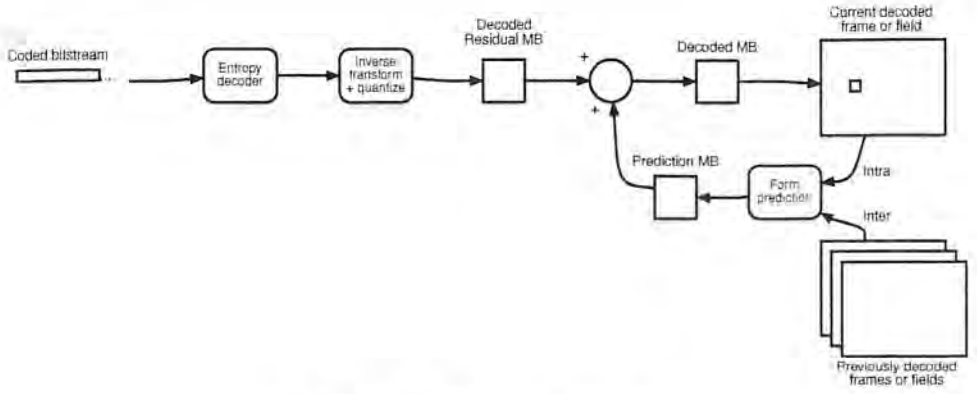


Figure 4.5 Typical H.264 decoder

prediction that was created at the encoder and adds this to the residual to produce a decoded macroblock.

4.3.1 Encoder processes

4.3.1.1 Prediction

The encoder forms a prediction of the current macroblock based on previously-coded data, either from the current frame using **intra** prediction or from other frames that have already been coded and transmitted using **inter** prediction. The encoder subtracts the prediction from the current macroblock to form a **residual**[1] (Figure 4.6).

The prediction methods supported by H.264 are more flexible than those in previous standards, enabling accurate predictions and hence efficient video compression. Intra prediction uses 16 × 16 and 4 × 4 block sizes to predict the macroblock from surrounding, previously coded pixels within the same frame (Figure 4.7).

The values of the previously-coded neighbouring pixels are extrapolated to form a prediction of the current macroblock. Figure 4.8 shows an example. A 16 × 16 prediction block is formed, an approximation of the original macroblock. Subtracting the prediction from the original macroblock produces a residual block (also containing 16 × 16 samples).

Inter prediction uses a range of block sizes from 16 × 16 down to 4 × 4 to predict pixels in the current frame from similar regions in previously coded frames (Figure 4.9). These previously coded frames may occur before or after the current frame in display order. In the example shown in Figure 4.9, macroblock 1 (MB1) in the current frame is predicted from a 16 × 16 region in the most recent 'past' frame. MB2 is predicted from two previously coded frames. The upper 8 × 16 block of samples, a 'partition', is predicted from a past frame and the lower 8 × 16 partition is predicted from a future frame.

[1] Finding a suitable inter prediction is often described as **motion estimation**. Subtracting an inter prediction from the current macroblock is **motion compensation**.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000110

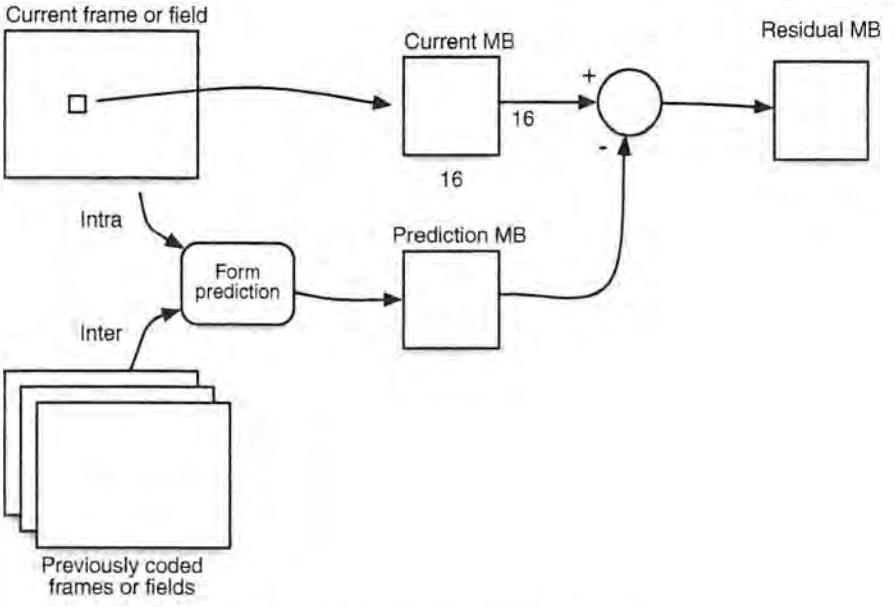


Figure 4.6 Prediction: flow diagram

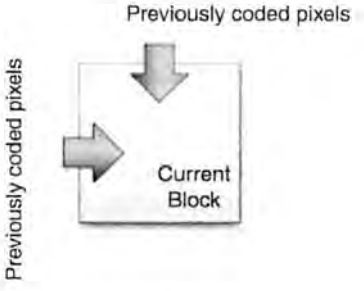


Figure 4.7 Intra prediction

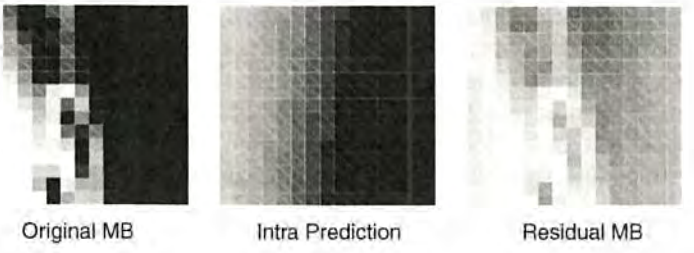


Figure 4.8 Original macroblock, intra prediction and residual

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000111

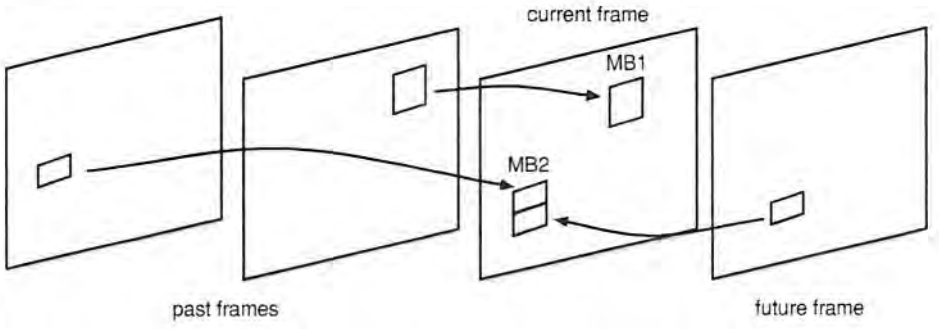


Figure 4.9 Inter prediction

Figure 4.10 shows an example of an original macroblock and its prediction from the previous frame. In this example, the 16 × 16 prediction block is a good match for the current macroblock. The values in the residual macroblock are mostly near zero.

4.3.1.2 Transform and quantization

A block of residual samples is transformed using a 4 × 4 or 8 × 8 **integer transform**, an approximate form of the Discrete Cosine Transform (**DCT**). The transform outputs a set of **coefficients**, each of which is a weighting value for a standard basis pattern. When combined, the weighted basis patterns re-create the block of residual samples (Figure 4.11).

The output of the transform, a block of transform coefficients, is **quantized**, i.e. each coefficient is divided by an integer value. Quantization reduces the precision of the transform coefficients according to a quantization parameter (QP). For example, the original coefficient values in Figure 4.12 are divided by a QP or step size of 8 and rounded to the nearest integer. Typically, the result is a block in which most or all of the coefficients are zero, with a few non-zero coefficients. Setting QP to a high value means that more coefficients are set to zero, resulting in high compression at the expense of poor decoded image quality. Setting QP to a low value means that more non-zero coefficients remain after quantization, resulting in better image quality at the decoder but also in lower compression (Figure 4.13).

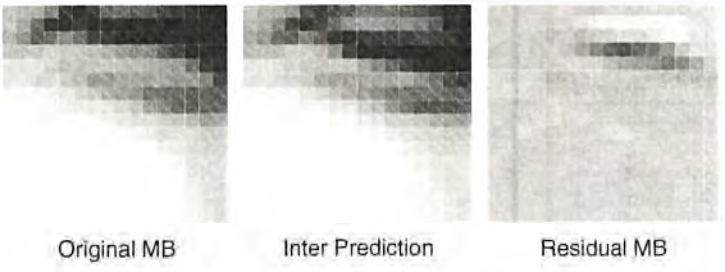


Figure 4.10 Original macroblock, inter prediction and residual

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000112

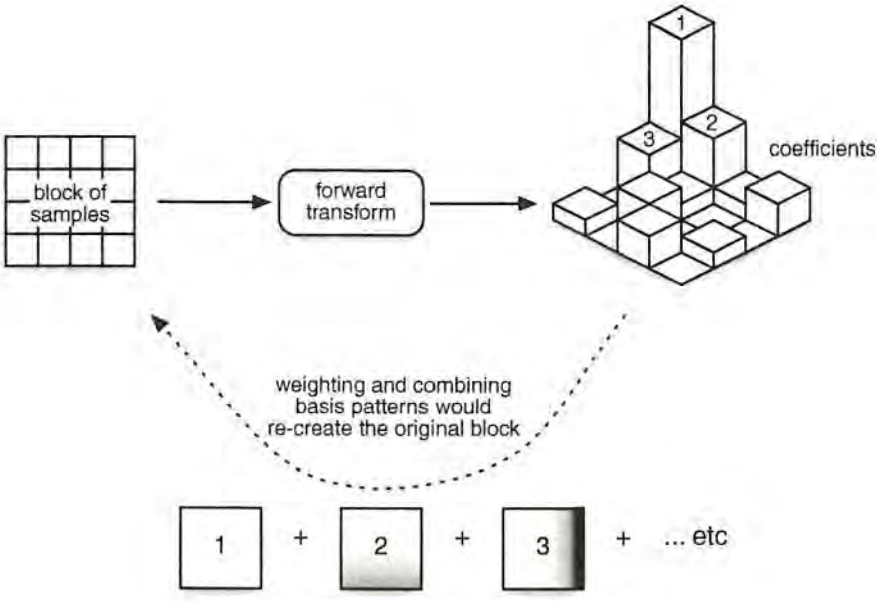


Figure 4.11 Forward transform

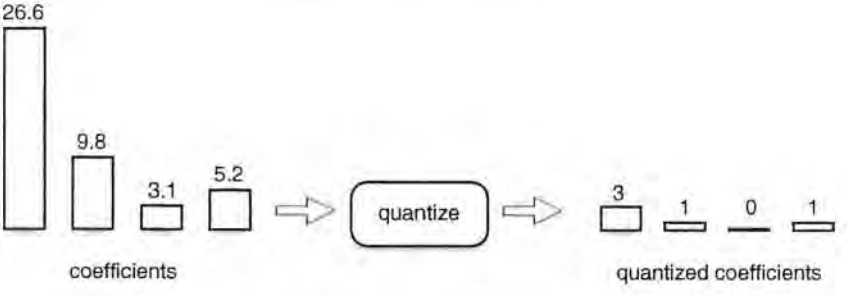


Figure 4.12 Quantization example

73 87 64 13 40 63 23 2 36 24 68 26 29 98 67 12 **Original block (4x4)**	181.3 47.4 −65.8 4.0 5.7 29.6 16.4 −2.2 40.3 18.3 −28.8 −13.8 13.1 −20.2 13.8 33.9 **Forward transform coefficients**
18 5 −7 0 1 3 2 0 4 2 −3 −1 1 −2 1 3 **Quantized coefficients (step size = 10)**	9 2 −3 0 0 1 1 0 2 1 −1 −1 1 −1 1 2 **Quantized coefficients (step size = 20)**

Figure 4.13 Example: Block, transform coefficients, quantized coefficients

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000113

4.3.1.3 Bitstream encoding

The video coding process produces a number of values that must be **encoded** to form the compressed bitstream. These values include:

- Quantized transform coefficients
- Information to enable the decoder to re-create the prediction
- Information about the structure of the compressed data and the compression tools used during encoding
- Information about the complete video sequence.

These values and parameters, **syntax elements**, are converted into binary codes using **variable length coding** and/or **arithmetic coding**. Each of these encoding methods produces an efficient, compact binary representation of the information. The encoded bitstream can then be stored and/or transmitted.

4.3.2 Decoder processes

4.3.2.1 Bitstream decoding

A video decoder receives the compressed H.264 bitstream, decodes each of the syntax elements and extracts the information described above, i.e. quantized transform coefficients, prediction information, etc. This information is then used to reverse the coding process and recreate a sequence of video images.

4.3.2.2 Rescaling and inverse transform

The quantized transform coefficients are **re-scaled**. Each coefficient is multiplied by an integer value to restore its original scale.[2] In the example of Figure 4.14, the quantized coefficients are each multiplied by a QP or step size of 8. The re-scaled coefficients are similar but not identical to the originals (Figure 4.12).

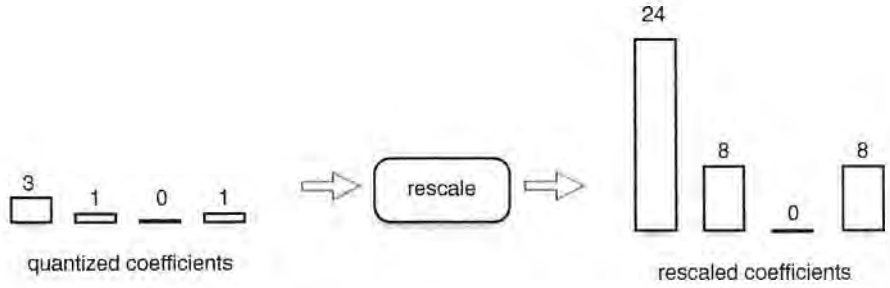


Figure 4.14 Rescaling example

[2] This is often described as **inverse quantization** but it is important to note that quantization is not a fully reversible process. Information removed during quantization cannot be restored during re-scaling.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000114

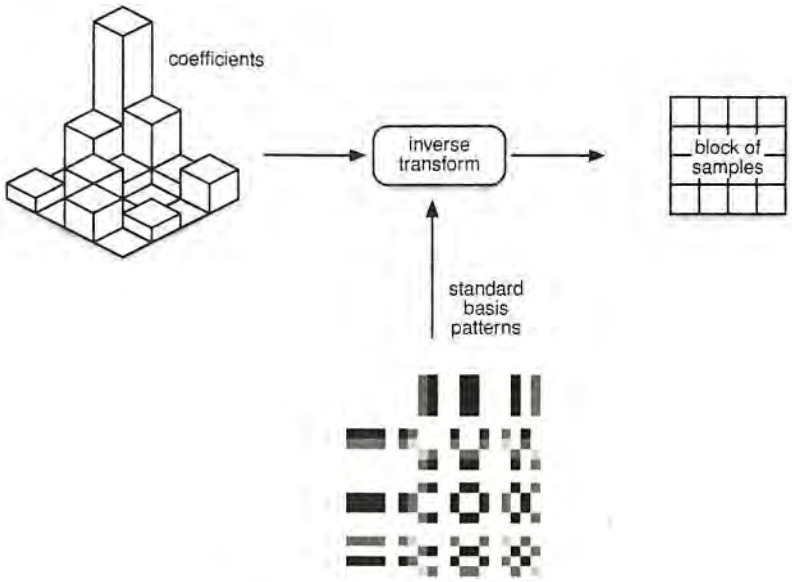


Figure 4.15 Inverse transform: combining weighted basis patterns to create a 4 × 4 image block

An inverse transform combines the standard basis patterns, weighted by the re-scaled coefficients, to re-create each block of residual data. Figure 4.15 shows how the inverse DCT or integer transform creates an image block by weighting each basis pattern according to a coefficient value and combining the weighted basis patterns. These blocks are combined together to form a residual macroblock. In Figure 4.16, the quantized coefficients of Figure 4.13 are rescaled using a quantization step size of 10 or 20 and inverse transformed. The reconstructed blocks are similar but not identical to the original block of Figure 4.13. The difference or loss is due to the forward quantization process. A larger quantization step size tends to produce a larger difference between original and reconstructed blocks.

180.0	50.0	−70.0	0.0	180.0	40.0	−60.0	0.0
10.0	30.0	20.0	0.0	0.0	20.0	20.0	0.0
40.0	20.0	−30.0	−10.0	40.0	20.0	−20.0	−20.0
10.0	−20.0	10.0	30.0	20.0	−20.0	20.0	40.0
Rescaled coefficients (QStep=10)				**Rescaled coefficients (QStep=20)**			
74	89	65	11	71	78	59	24
42	64	23	4	29	61	23	2
30	29	64	24	40	23	73	29
27	99	67	8	30	103	60	15
Inverse transform output (QStep=10)				**Inverse transform output (QStep=20)**			

Figure 4.16 Rescaled coefficients and inverse transform output

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000115

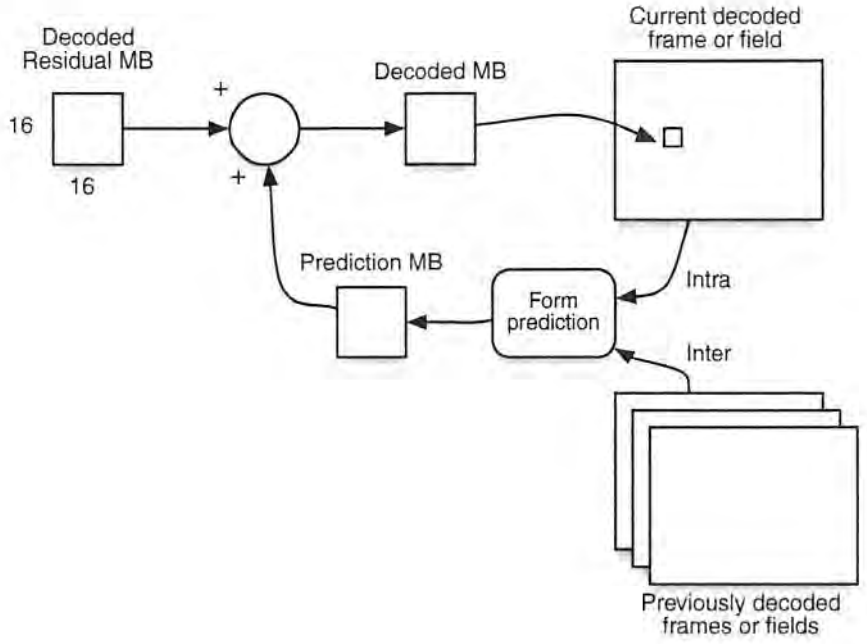


Figure 4.17 Reconstruction flow diagram

4.3.2.3 Reconstruction

For each macroblock, the decoder forms an identical prediction to the one created by the encoder using inter prediction from previously-decoded frames or intra prediction from previously-decoded samples in the current frame. The decoder adds the prediction to the decoded residual to **reconstruct** a decoded macroblock which can then be displayed as part of a video frame (Figure 4.17).

4.4 The H.264/AVC Standard

The current version of Recommendation H.264 [i] is a document of over 550 pages. It contains normative content, essential instructions that must be observed by H.264 codecs, and informative content, guidance that is not mandatory, and is organised as shown in Table 4.1. Note that as amendments or updated versions of the standard are published, further sections or Annexes may be added.

An H.264-compliant bitstream, a coded video sequence, should conform to the syntax definition, with syntax elements constructed according to the semantics described in Chapter 7 of the standard. An H.264 decoder should follow the parsing process in Chapter 9 of the standard to extract syntax elements from the bitstream and these should be decoded according to the decoding processes described in Chapter 8 of the standard in order to create a decoded video sequence.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000116

Table 4.1 Overview of the H.264 standard document

Section	What it describes
0. Introduction	A brief overview of the purpose and main features of the standard.
1–5. Scope, References, Definitions, Abbreviations, Conventions	How H.264 fits with other published standards; terminology and conventions used throughout the document.
6. Data formats and relationships	Defines the basic formats assumed for video and coded data; principles of deriving relationships between coded units.
7. Syntax and semantics	A series of tables describing the syntax or bitstream format and an explanation of the meaning and allowed values of each syntax element (semantics)
8. Decoding process	Details of all the stages of processing required to decode a video sequence from H.264 syntax elements.
9. Parsing process	The processes required to extract syntax elements from a coded H.264 bitstream.
A. Profiles and levels	Profiles define subsets of video coding tools; levels define limits of decoder capabilities.
B. Byte stream format	Syntax and semantics of a stream of coded NAL units.
C. Hypothetical reference decoder	A hypothetical decoder 'model' that is used to determine performance limits.
D. Supplemental enhancement information	Information that may be sent in an H.264 bitstream that is not essential for decoding. Examples include user-generated data, buffering control, etc.
E. Video usability information	Non-essential information about display of the coded video.
G. Scalable Video Coding	A self-contained extension to the H.264/AVC standard that supports Scalable Video Coding (SVC). SVC supports efficient transmission of video at multiple spatial resolutions, frame rates and/or quality levels.

4.5 H.264 Profiles and Levels

H.264 supports a portfolio of 'tools'. These are algorithms or processes for coding and decoding video and related operations. These include tools essential to any implementation of H.264 such as the basic 4×4 transform, optional or interchangeable tools such as CABAC or CAVLC entropy coding, tools for specific applications or scenarios such as the SI/SP tools which can support streaming video and tools not directly related to video coding such as VUI and SEI messages.

An encoder has considerable flexibility in choosing and using tools defined in the standard. A decoder may be capable of operating with a very limited set of tools, for example if it is only required to decode bitstreams that use a subset of tools. H.264 **profiles** each define a specific subset of tools. An H.264 bitstream that conforms to a particular profile can only contain video coded using some or all of the tools within the profile; a profile-compliant decoder must be capable of decoding with every tool in the profile. The profiles therefore act as constraints on

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000117

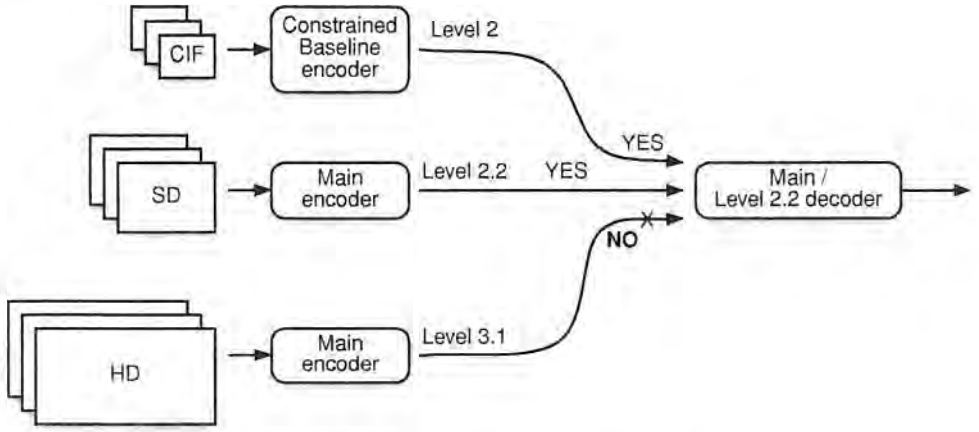


Figure 4.18 Profiles and Levels: example

the capabilities required by a decoder. The Main Profile is perhaps the most widely used at the present time, with tools that offer a good compromise between compression performance and computational complexity. The Constrained Baseline Profile is a subset of the Main Profile that is popular for low-complexity, low-delay applications such as mobile video.

A second constraint is the amount of data that a decoder can handle, which is likely to be limited by the processing speed, memory capacity and display size at the decoder. An H.264/AVC **level** specifies an upper limit on the frame size, processing rate (number of frames or blocks can be decoded per second) and working memory required to decode a video sequence. A particular decoder can only decode H.264 bitstreams up to a certain combination of Profile and Level.

Example

The decoder shown in Figure 4.18 can decode Main Profile bitstreams, up to Level 2.2 which corresponds to Standard Definition video coded at bitrates of 4Mbps or less. The figure shows three potential input bitstreams. The first is coded at CIF resolution with a maximum bitrate of 2Mbps (Level 2) using the tools in the Constrained Baseline Profile. Constrained Baseline is a subset of Main Profile and Level 2 is lower than Level 2.2, so the Main Profile/Level 2.2 decoder **can** decode this bitstream.

The second is coded at Standard Definition resolution, 720 × 576 displayed pixels, at a bitrate of less than 4Mbps (Level 2.2) by a Main Profile encoder. The decoder **can** decode this bitstream.

The third is coded at 720p High Definition resolution, 1280 × 720 displayed pixels, and is coded with a Main Profile encoder at Level 3.1. This is higher than the maximum Level supported by the decoder and so it **cannot** be decoded.

This example shows that Profiles and Levels are a useful mechanism for ensuring that a decoder's capabilities are not exceeded. The Profile and Level of an H.264 bitstream are signalled in the Sequence Parameter Set (Chapters 5 and 8).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000118

4.6 The H.264 Syntax

H.264 provides a clearly defined format or **syntax** for representing compressed video and related information. Figure 4.19 shows an overview of the structure of the H.264 syntax.

At the top level, an H.264 **sequence** consists of a series of 'packets' or Network Adaptation Layer Units, NAL Units or NALUs. These can include **parameter sets** containing key parameters that are used by the decoder to correctly decode the video data and **slices**, which are coded video frames or parts of video frames.

At the next level, a **slice** represents all or part of a coded video frame and consists of a number of coded **macroblocks**, each containing compressed data corresponding to a 16 × 16 block of displayed pixels in a video frame. At the lowest level of Figure 4.19, a macroblock contains type information describing the particular choice of methods used to code the macroblock,

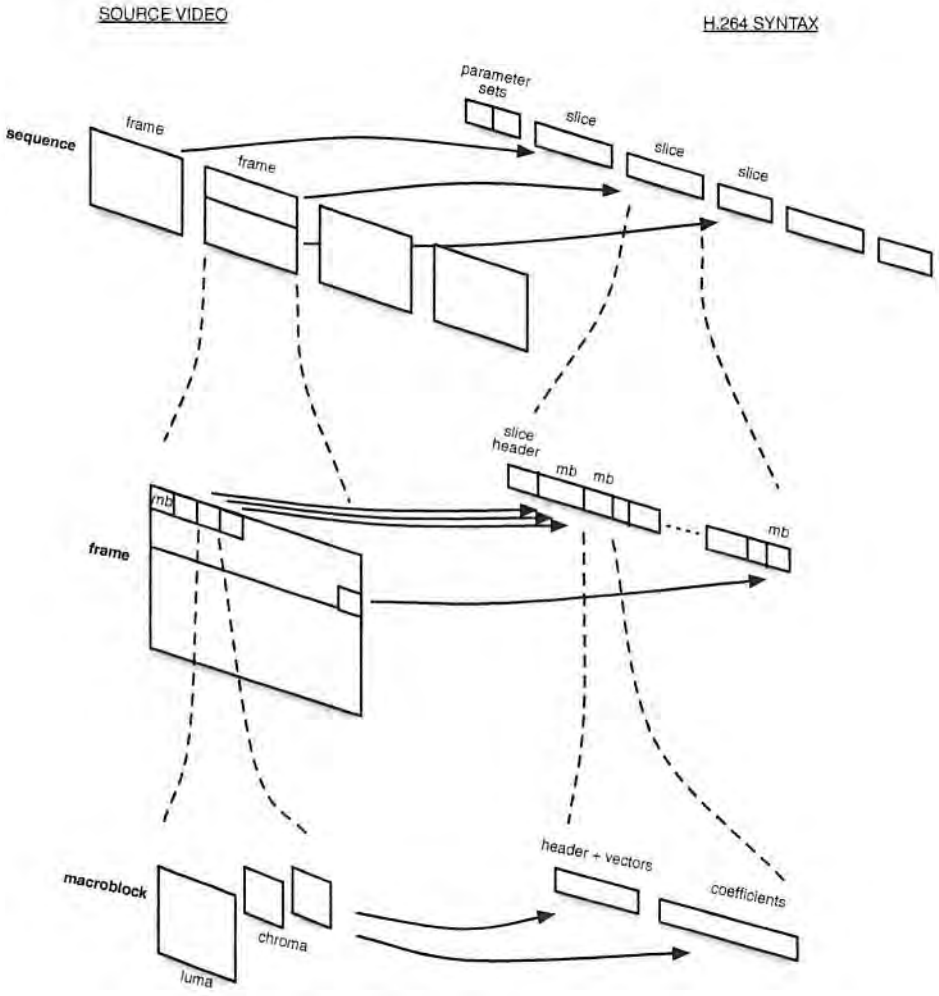


Figure 4.19 H.264 syntax: overview

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000119

prediction information such as coded motion vectors or intra prediction mode information and coded residual data.

Example

An H.264 decoder receives the following series of bits, describing an inter-coded macroblock or P MB coded using context-adaptive VLCs:

1101111100011100011100 10 (alternate syntax elements highlighted).

Figure 4.20 shows the syntax elements in this sequence of bits. The header elements, the first eight bits and six syntax elements, indicate the macroblock type (P, one motion vector),

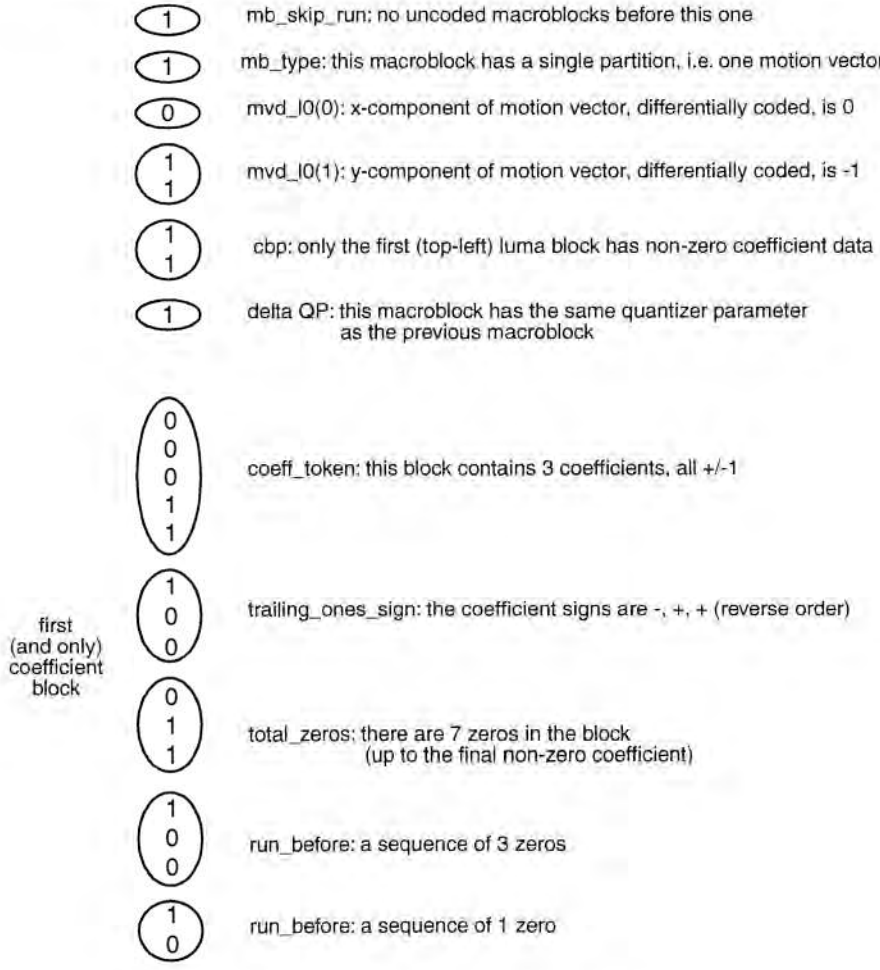


Figure 4.20 Syntax example: P-macroblock

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000120

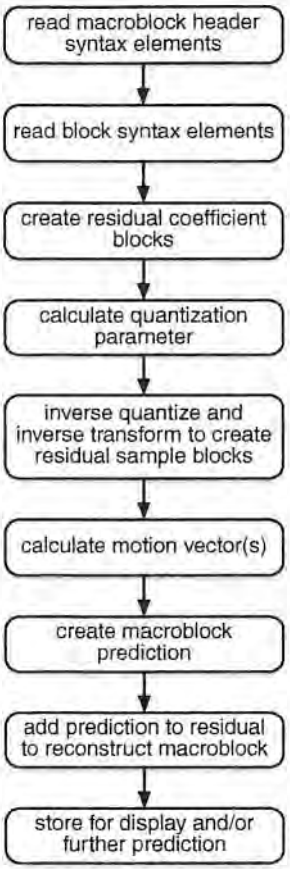


Figure 4.21 P-macroblock decoding process

the prediction parameters, in this case the x and y components of the motion vector, coded differentially from previously-received vectors, the coded block pattern which indicates that only one of the 4 × 4 blocks in the MB actually contains residual data and the quantization parameter, coded differentially from the previous macroblock. The next 16 bits represent 5 syntax elements that specify the contents of the first and only non-zero coefficient block. Decoding the macroblock proceeds as follows (Figure 4.21):

- Extract the residual luma and chroma coefficient blocks (Figure 4.22, also Chapter 7)
- Calculate the actual quantization parameter QP
- Carry out the inverse quantization and inverse transform processes to produce the residual sample blocks (Chapter 7)
- Calculate the actual motion vector
- Produce a prediction macroblock (Chapter 6)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000121

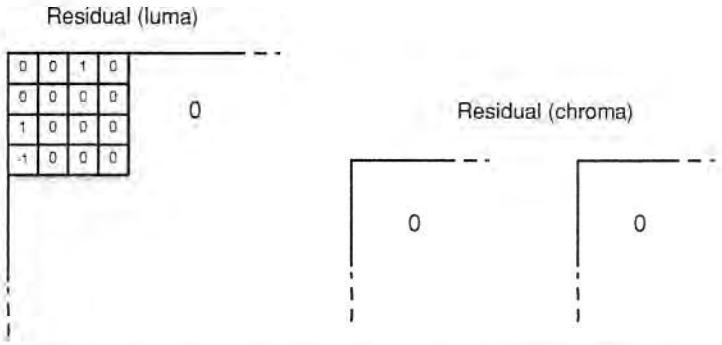


Figure 4.22 Residual luma and chroma coefficient blocks

- Add the prediction macroblock to the residual samples to reconstruct the decoded macroblock
- Store as part of the decoded frame for display and/or for further prediction (Chapters 5 and 6).

4.7 H.264 in practice

4.7.1 Performance

Perhaps the biggest advantage of H.264 over previous standards is its compression performance. Compared with standards such as MPEG-2 and MPEG-4 Visual, H.264 can deliver:

- Better image quality at the same compressed bitrate, or
- A lower compressed bitrate for the same image quality.

For example, a single-layer DVD can store a movie of around two hours' length in MPEG-2 format. Using H.264, it should be possible to store four hours or more of movie-quality video on the same disk, i.e. a lower bitrate for the same quality. Alternatively, the H.264 compression format can deliver better quality at the same bitrate compared with MPEG-2 and MPEG-4 Visual (Figure 4.23).



Figure 4.23 A video frame compressed at the same bitrate using MPEG-2 (left), MPEG-4 Visual (centre) and H.264 compression (right)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000122

The improved compression performance of H.264 comes at the price of greater computational cost. H.264 is more sophisticated than earlier compression methods and this means that it can take significantly more processing power to compress and decompress H.264 video.

4.7.2 Applications

As well as its improved compression performance, H.264 offers greater flexibility in terms of compression options and transmission support. An H.264 encoder can select from a wide variety of compression tools, making it suitable for applications ranging from low-bitrate, low-delay mobile transmission through high definition consumer TV to professional television production. The standard provides integrated support for transmission or storage, including a packetized compressed format and features that help to minimize the effect of transmission errors.

H.264/AVC has been adopted for an increasing range of applications, including:

- High Definition DVDs (Blu-Ray)
- High Definition TV broadcasting in Europe
- Apple products including iTunes video downloads, iPod video and MacOS
- NATO and US DoD video applications
- Mobile TV broadcasting
- Many mobile video services
- Many internet video services
- Videoconferencing
- Consumer camcorders.

4.8 Summary

H.264 Advanced Video Compression is an industry standard for video coding that defines a format or syntax for compressed video and a method of decoding this syntax. It provides a set of tools or algorithms that can be used to deliver efficient, flexible and robust video compression for a wide range of applications, from low-complexity, low bitrate mobile video applications to high-definition broadcast services. The following chapters examine H.264's tools, features and performance in detail.

4.9 References

i. Recommendation ITU-T H.264 | ISO/IEC 14496-10:2009, 'Advanced Video Coding for generic audio-visual services', March 2009.
ii. ISO/IEC 14496-2:2000, 'Information technology – Generic coding of moving pictures and associated audio information: Video', 2000.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000123

5

H.264 syntax

5.1 Introduction

An 'H.264 video' is a video sequence that is represented in a particular format, the H.264/AVC syntax. This syntax is defined in the H.264 industry standard and specifies the exact structure of an H.264-compliant video sequence, in terms of syntax elements, parameters describing different aspects of the coded sequence, and the way in which each element is represented as a binary code. The syntax is hierarchical, from the highest level, the video sequence level, down through individual coded frames or fields (access units) and subsets of access units (slices) to macroblocks and blocks. Control parameters are stored as separate syntax sections such as Parameter Sets or as part of sections such as the macroblock layer. This chapter examines the H.264/AVC syntax in detail, beginning with an overview of frame and picture structures and handling, followed by each of the main layers of the syntax hierarchy.

5.1.1 A note about syntax examples

The various syntax sections in the H.264/AVC standard are illustrated later in the chapter with extracts from coded video sequences. These were generated using the Joint Model software video codec [i] using the 'Trace' mode which is described further in Chapter 9. Each syntax example is presented in the following way, Table 5.1:

Table 5.1 Syntax examples: format

Parameter	Binary code	Symbol	Discussion
profile_idc	*1000010*	*66*	*Baseline Profile*
↑ The name of the syntax parameter, listed in the H.264/AVC standard	↑ The binary code	↑ The value decoded from this code	↑ What the value indicates.

Selected sections of the H.264 syntax are presented as syntax diagrams such as Figure 5.13.

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000124

It should be emphasized that the syntax examples, diagrams and explanations in this chapter are for illustration only. Complete and accurate explanations of the syntax and the semantics or meaning of each syntax element are given in the H.264/AVC standard.

5.2 H.264 syntax

The hierarchical organization of the H.264/AVC syntax can be seen in Figure 5.1 [ii]. The **Network Abstraction Layer** (NAL) consists of a series of **NAL Units** (section 5.4). Sequence Parameter Sets (SPS) and Picture Parameter Sets (PPS) are NAL units that signal certain common control parameters to the decoder (section 5.5). Coded video data is communicated in **Video Coding Layer** (VCL) NAL units, known as coded **slices**. An **access unit**, a coded

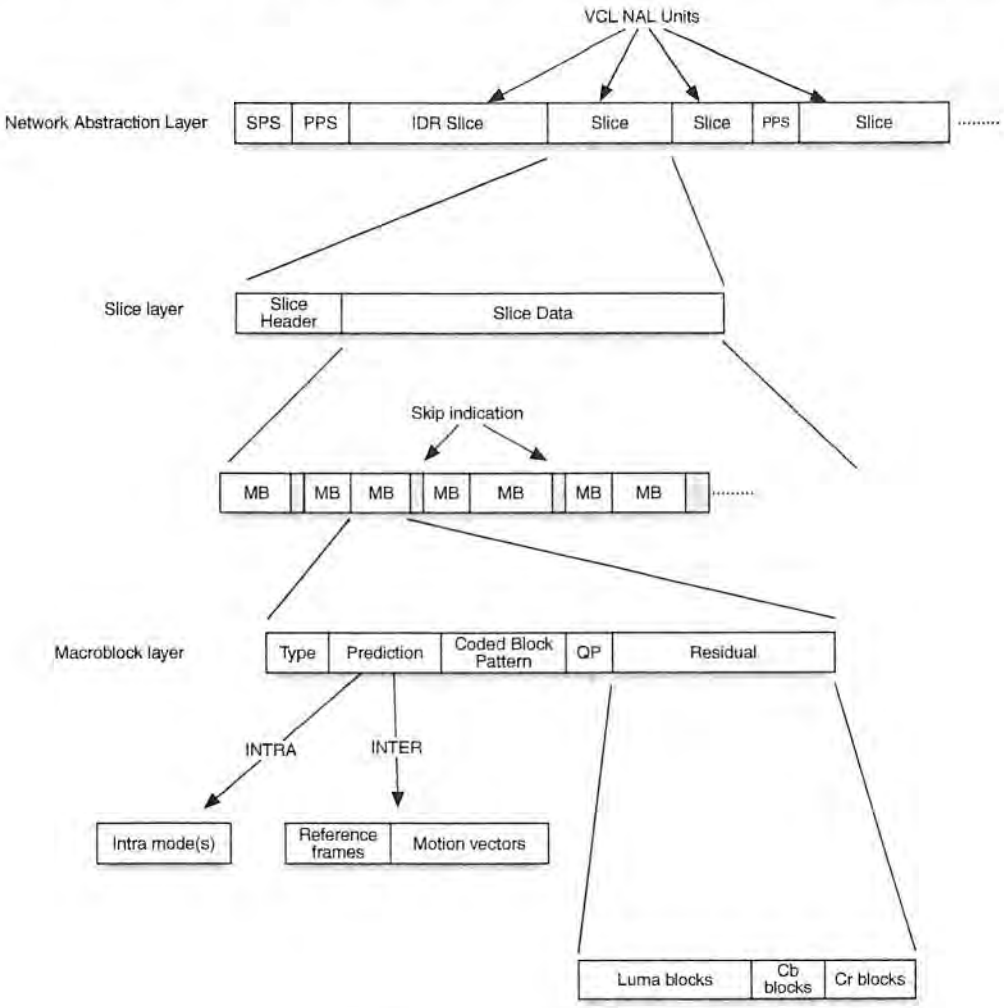


Figure 5.1 Syntax overview

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000125

frame or field, is made up of one or more slices. At the **slice layer** (section 5.6), each slice consists of a Slice Header and Slice Data. The Slice Data is a series of coded **macroblocks** (MB) and skip macroblock indicators which signal that certain macroblock positions contain no data. Each coded macroblock (section 5.7) contains the following syntax elements:

- MB type : I/intra coded, P/inter coded from one reference frame, B/inter coded from one or two reference frames.
- Prediction information : prediction mode(s) for an I macroblock, choice of reference frame(s) and motion vectors for a P or B macroblock.
- Coded Block Pattern CBP : indicates which luma and chroma blocks contain non-zero residual coefficients.
- Quantization Parameter QP, for macroblocks with CBP $\neq 0$.
- Residual data, for blocks containing non-zero residual coefficients.

A **coded video sequence** begins with an Instantaneous Decoder Refresh (**IDR**) Access Unit, made up of one or more IDR slices, a special type of Intra coded slice. Subsequent video frames or fields, described as Access Units, are coded as slices. The video sequence ends when a new IDR slice is received, signalling a new coded sequence, or when the transmission is complete.

A list of all the main sections of the syntax is given in Table 5.2. A NAL unit contains a **Raw Byte Sequence Payload (RBSP)**, a sequence of bytes containing syntax elements. H.264 syntax elements are binary codes with varying length and so a sequence of syntax elements within a NAL unit will not necessarily fit into an integral number of bytes. Zero-valued bits are added to the end of the RBSP contents in order to ensure that the length is an integral number of bytes (**RBSP trailing bits**). An RBSP syntax element can be transmitted or stored as a complete packet. Some syntax sections contain sub-sections, e.g. the Sequence Parameter Set may contain the sections *scaling_list* and *vui_parameters* and always contains the section *rbsp_trailing_bits*. Other sections, e.g. *scaling_list*, do not contain any sub-sections.

Before discussing the syntax sections in detail, it is necessary to examine the way in which H.264/AVC handles frames, fields and pictures.

5.3 Frames, fields and pictures

Chapter 2 introduced the concept of digital video and typical formats for storing and handling digital video information. An H.264 encoder converts video frames into compressed or coded picures. H.264/AVC defines a **frame** as an array of luma samples and two corresponding arrays of chroma samples. The two **fields**, top field and bottom field, that make up a frame may be sampled at the same time instant, progressive scan, or at different time instants, interlaced scan. The term **picture** refers collectively to a frame or a field. Figure 5.2 is an overview of the way pictures are handled during H.264 encoding and decoding. Frames or fields are encoded to form coded pictures, each of which is composed of one or more slices. Slices are decoded to produce decoded pictures which are stored in a Decoded Picture Buffer (DPB). Pictures in the DPB may be used for inter prediction of further coded pictures and/or output for display.

It is important to distinguish between three different orders of pictures: decoding order, the order in which pictures are decoded from the bitstream, display order, the order in which pictures are output for display, and reference order, the order in which pictures are arranged for inter prediction of other pictures (Figure 5.3).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000126

Table 5.2 H.264 Syntax Sections

Syntax section	Description	Contains section(s)
NAL unit	Network abstraction layer unit.	Raw Byte Sequence Payload (RBSP)
Sequence Parameter Set (RBSP)	Parameters common to a video sequence.	Scaling List (optional) VUI Parameters (optional) RBSP trailing bits
Scaling List	Encoder-specified scaling matrix for inverse quantization.	None
SPS extension (RBSP)	Auxiliary picture information for alpha blending.	RBSP trailing bits
Picture Parameter Set (RBSP)	Parameters common to one or more coded pictures.	Scaling List (optional) RBSP trailing bits
Supplement Enhancement Information (RBSP)	Container for SEI message, see below.	SEI message RBSP trailing bits
Supplemental Enhancement Information Message	SEI messages contain information that may assist decoding or display but is not essential for constructing decoded video frames.	SEI payload
Access Unit Delimiter (RBSP)	Optional delimiter that indicates the slice type in the next coded picure.	RBSP trailing bits
End of Sequence (RBSP)	Optional delimiter that indicates the next slice is an IDR.	None
End of Stream (RBSP)	Optional delimiter that indicates the end of the coded bitstream.	None
Filler Data (RBSP)	Optional series of filler bytes.	RBSP trailing bits
Slice layer without partitioning (RBSP)	Coded slice that does not use data partitioning.	Slice header Slice data RBSP slice trailing bits
Slice data partition A layer (RBSP)	Partition A of a Data Partitioned Slice, see Chapter 8.	Slice header Slice data RBSP slice trailing bits
Slice data partition B layer (RBSP)	Partition B of a Data Partitioned Slice, see Chapter 8.	Slice header Slice data RBSP slice trailing bits
Slice data partition C layer (RBSP)	Partition C of a Data Partitioned Slice, see Chapter 8.	Slice header Slice data RBSP slice trailing bits
RBSP slice trailing bits	Contains RBSP trailing bits and occurs at end of coded slice or slice data partition.	RBSP trailing bits
RBSP trailing bits	Padding bits to ensure RBSP is byte aligned.	None
Slice header	Parameters common to a coded slice.	Reference picture list reordering Prediction weight table (optional) Decoded reference picture marking (optional)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000127

Table 5.2 (*Continued*)

Syntax section	Description	Contains section(s)
Reference picture list reordering	Series of commands to change default reference picture list order(s).	None
Prediction weight table	Luma and chroma weight offsets to change effect of motion compensated prediction.	None
Decoded reference picture marking	Series of commands to mark reference pictures as long term references, etc.	None
Slice data	Contains series of coded macroblocks.	Macroblock layer
Macroblock layer	PCM samples or macroblock headers, prediction and transform coefficients.	Macroblock prediction (optional) Sub-macroblock prediction (optional) Residual data (optional)
Macroblock prediction	Intra prediction modes or reference indices and motion vectors.	None
Sub-macroblock prediction	Reference indices and motion vectors.	None
Residual data	Contains series of residual blocks, as indicated by Coded Block Pattern.	Residual block CAVLC or residual block CABAC
Residual block CAVLC	Block of transform coefficients coded using CAVLC.	None
Residual block CABAC	Block of transform coefficients coded using CABAC.	None

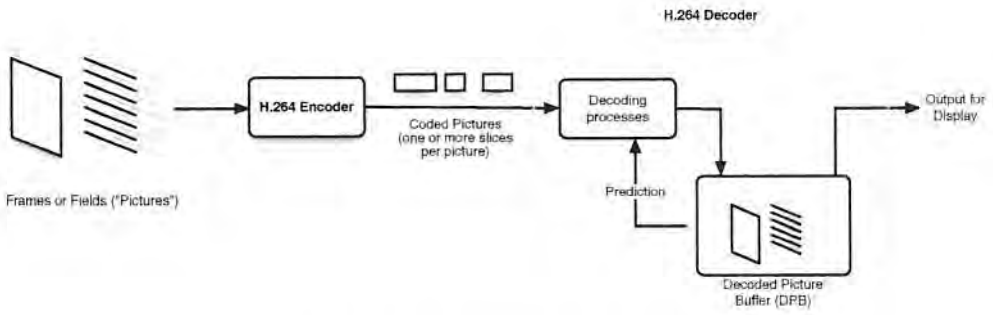


Figure 5.2 Picture handling in H.264, overview

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000128

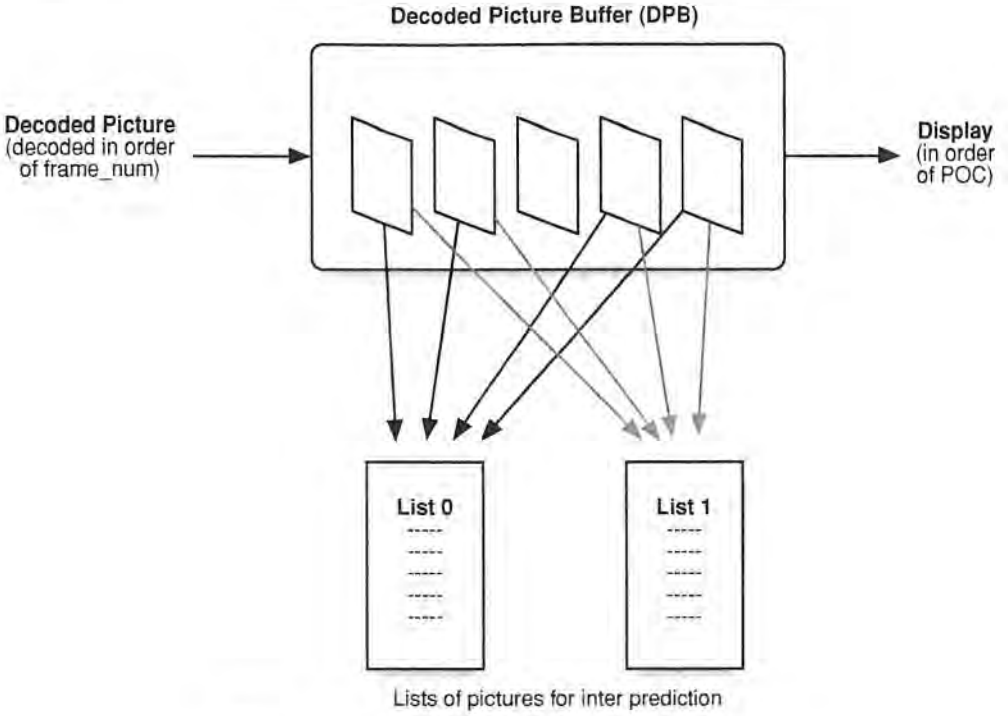


Figure 5.3 Decoded Picture Buffer and picture orders

As shown in Figure 5.3, the **decoding order** of pictures, i.e. the order in which a video decoder should process the coded pictures, is indicated by the parameter *frame_num*.

The **display order** of pictures is determined by the parameters *TopFieldOrderCount* and *BottomFieldOrderCount*, collectively described as Picture Order Count, POC.

The **reference order** of pictures is determined by one or two lists, each of which is an ordered list of all the available decoded pictures. A P slice uses a single list, list0, and a B slice uses two lists, list0 and list1, each containing available reference pictures in different orders.

5.3.1 *Decoding order*

The parameter *frame_num*, decoded from the slice header, determines the decoding order of pictures, coded frames or fields. Except in certain special cases, *frame_num* for each decoded picture increases by one compared with the previous **reference** frame in decoding order.

5.3.2 *Display order*

Display order is determined by the POC parameters *TopFieldOrderCount* and *BottomFieldOrderCount* which are derived from the slice header using one of three methods:

- Type 0: The least significant bits of POC are sent in every slice header. This allows maximum flexibility but typically requires more bits than the other methods.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000129

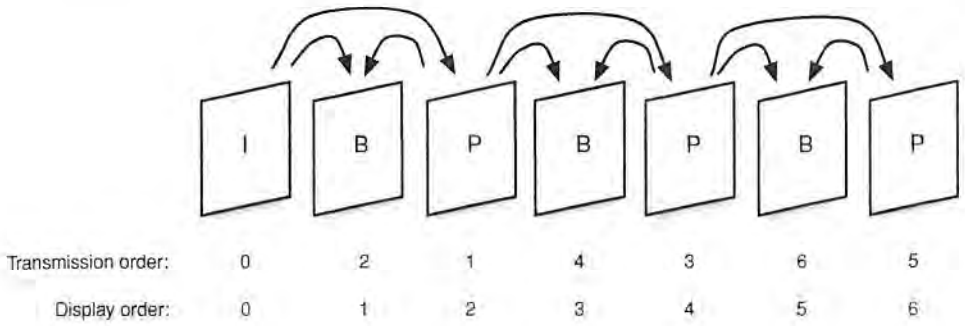


Figure 5.4 Display order: Type 0 example

- Type 1: A 'cycle' of POC increments is set up in the sequence parameter set and POC changes according to this cycle unless otherwise signalled in the slice header using a Delta offset. The cycle defines the interval between frames used for reference, plus a POC offset to frames not used for reference.
- Type 2: POC is derived directly from *frame_num* and display order is the same as decoding order.

Display Order Example, Type 0:

Frame pictures, one frame per slice, display order IBPBPBPBPB... (Figure 5.4).

In this example, the B slices, bipredicted slices in which each macroblock or partition can be predicted from up to two reference pictures, are not used for reference prediction of any other pictures. POC increments by two for every complete frame, i.e. every two fields. Note that *frame_num* increments **after** each reference picture is transmitted. In this example, only the I and P pictures are reference pictures. Increments in *frame_num* are indicated in bold type.

Slice	Type	Used for reference	*frame_num*	POC LSBs	Display order
1st	I	Yes	**0**	0	0
2nd	P	Yes	**1**	4	2
3rd	B	No	**2**	2	1
4th	P	Yes	2	8	4
5th	B	No	**3**	6	3
6th	P	Yes	3	12	6
7th	B	No	**4**	10	5
8th	P	Yes	4	16	8
...	...				

Display Order Example, Type 1:

Frame pictures, one frame per slice, display order: IBBPBBPBBPB... (Figure 5.4)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000130

In this example, the B slices are not used for reference. The POC cycle consists of one reference frame, with an offset from the previous reference frame to the next reference frame of +6 and an offset from the previous reference frame to the next non-reference frame of −4.

Slice	Type	Used for reference	*frame_num*	Delta POC	Offset	POC	Display order
1st	I	Yes	**0**	0	0	0	0
2nd	P	Yes	**1**	0	+6	6	3
3rd	B	No	**2**	0	−4	2	1
4th	B	No	2	2	−4+2	4	2
5th	P	Yes	2	0	+6	12	6
6th	B	No	**3**	0	−4	8	4
7th	B	No	3	2	−4+2	10	5
8th	P	Yes	3	0	+6	18	9
...	...						

The 4th and 7th slices have a delta POC, signalled in the slice header, of +2. The POC for these slices is calculated as: POC = expected POC + 2.

5.3.3 *Reference picture lists*

A picture that is coded and available for reference is stored in the Decoded Picture Buffer (DPB) and marked as one of the following:

(a) a short term reference picture, indexed according to *frame_num* or POC or,
(b) a long term reference picture, indexed according to *LongTermPicNum*, a reference number derived from the parameter *LongTermFrameIdx* which is assigned when a picture is marked as a long term reference picture.

Short term reference pictures may be assigned a *LongTermFrameIdx*, i.e. changed to a long term reference picture, at a later time.

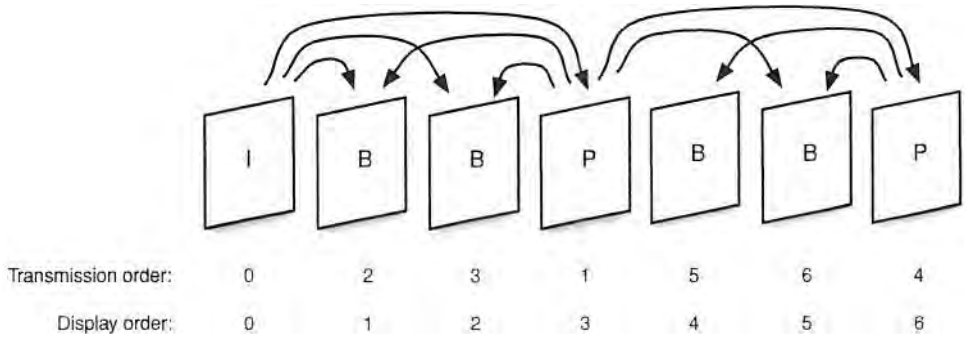


Figure 5.5 Display order: Type 1 example

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000131

Short term reference pictures are removed from the DPB (a) by an explicit command in the bitstream or (b) when the DPB is full and an automatic mode of DPB handling is in use, in which case the oldest short term picture is removed. Long term pictures are removed by an explicit command in the bitstream.

5.3.3.1 Default reference picture list order

Reference pictures are ordered in one or two lists prior to encoding or decoding a slice. A P slice uses a single list of reference pictures, list0 and a B slice uses two lists, list0 and list1. In each list, short term reference pictures are listed first followed by long term reference pictures, in increasing order of *LongTermPicNum*. The default short term reference picture order depends on **decoding order** when the current slice is a P slice and on **display order** when the current slice is a B slice. The list orders are important, since indices to reference pictures earlier in the list require fewer bits to signal. Hence the default orders are organized so that reference pictures temporally 'nearer' to the current picture occur earlier in the list, since these are likely to contain the best prediction match for the current picture.

List0 (P slice) : The default order is in decreasing order of *PicNum*. *PicNum* is a version of *frame_num*, 'wrapped around' modulo *MaxFrameNum*.

List0 (B slice) : The default order is (1) in decreasing order of POC, for pictures with POC earlier than current picture, then (2) in increasing order of POC, for pictures with POC later than the current picture.

List1 (B slice) : The default order is (1) in increasing order of POC, for pictures with POC later than the current picture, then (2) in decreasing order of POC, for pictures with POC earlier than current picture.

Reference Picture Order Example: P slices (list0)

The reference picture list is initially empty. The current *frame_num* is 150 and the maximum size of the DPB is 5 frames. Italics indicate a *LongTermPicNum*. Notice that by default, the first entry in list0 is the most recently coded frame. This is likely to be the 'best' prediction reference for the majority of macroblocks.

Operation	list0(0)	list0(1)	list0(2)	list0(3)	list0(4)
Initial state	-	-	-	-	-
Encode frame 150	150	-	-	-	-
Encode 151	151	150	-	-	-
Encode 152	152	151	150	-	-
Encode 153	153	152	151	150	-
Encode 154	154	153	152	151	150
Encode 155	155	154	153	152	151
Assign 154 to LongTermPicNum *3*	155	153	152	151	*3*
Encode 156 and mark it as LongTermPicNum *1*	155	153	152	*1*	*3*
Encode 157	157	155	153	*1*	*3*
.....					

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000132

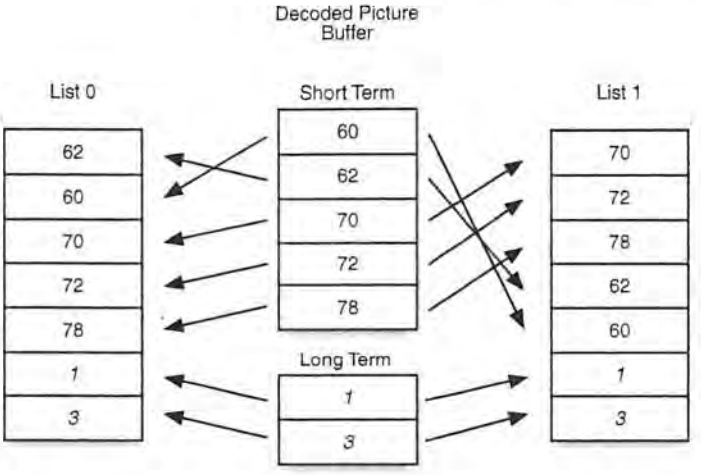


Figure 5.6 List 0 and List 1 ordering: example

Reference Picture Order Example: B slice (list0 and list1)

The DPB contains short term reference pictures with POC 60, 62, 70, 72 and 78 and long term reference pictures with LongTermFrameIdx *1* and *3*. The current POC is 68. The default order of list0 is:

62, 60, 70, 72, 78, *1*, *3*

The default order of list1 is:

70, 72, 78, 62, 60, *1*, *3*

Figure 5.6 shows how the two lists are created from the sequence of pictures in the DPB. List0 defaults to prediction from 'past' pictures in display order whilst List1 defaults to 'future' pictures in display order. This is shown graphically in Figure 5.7. Note that the first entry in

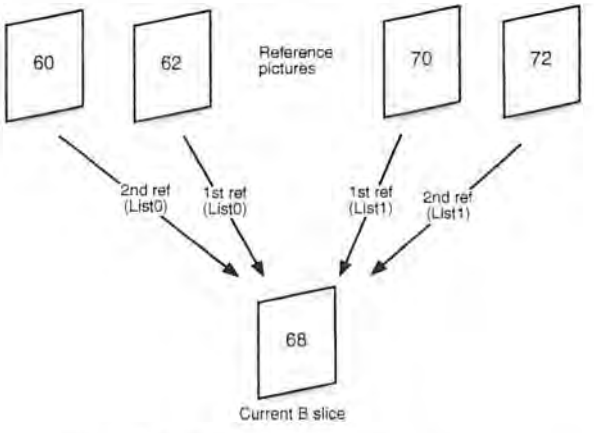


Figure 5.7 Default reference pictures: example

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000133

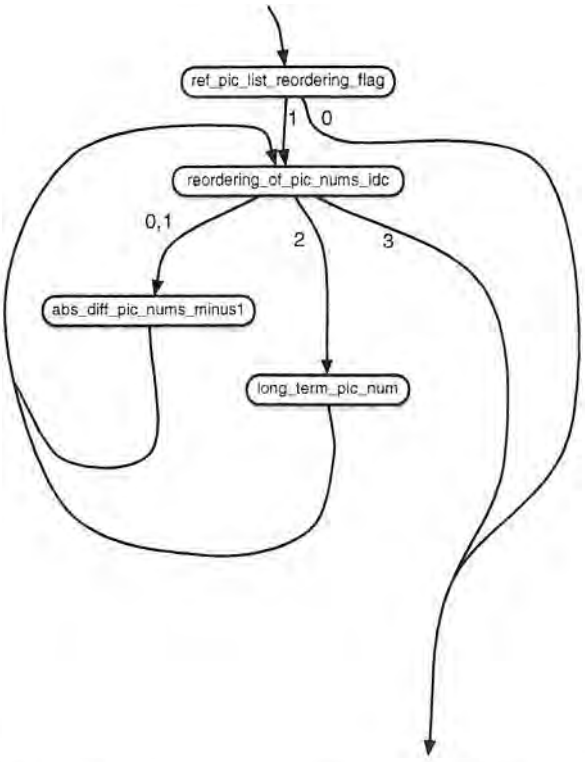


Figure 5.8 Reference picture re-ordering syntax, simplified overview

list0 is the most recent **past** picture (62) and the first entry in list1 is the nearest **future** picture (70). These are likely to be the best 'past' and 'future' prediction references respectively.

5.3.3.2 Changing the reference picture list order

The default order of list0, and of list1 in the case of a B slice, may be modified by explicit commands in the slice header. These commands are initiated by the syntax element *ref_pic_list_reordering_flag* and change the list order **for the current slice only**. This may be used, for example, to enable an encoder to place a particularly useful reference frame earlier in the list than its usual position, since earlier reference indices cost fewer bits to signal.

The reference picture list re-ordering process is illustrated in Figure 5.8 and proceeds as follows for list 0. This is a simplified overview.

- Initialize a pointer (refIdxL0) to point to the first reference picture index in the list (0)
- While *reordering_of_pic_nums_idc* $\neq 3$
  - Select a reference picture, either short term, indicated by *abs_diff_pic_num_minus1*, or long term, indicated by *long_term_pic_num*
  - Move all other pictures from position refIdxL0 onwards in the list to one position later
  - Put this picture in the list at the position indicated by refIdxL0
  - Increment pointer refIdxL0.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000134

Table 5.3 Selecting a reference picture to re-map

reordering_of_pic_nums_idc	remappedPicNum is calculated as:
0	predictedPicNum $-$ (*abs_diff_pic_num_minus_1* $+1$)
1	predictedPicNum $+$ (*abs_diff_pic_num_minus_1* $+1$)
2	*long_term_pic_num*
3	(Exit the re-ordering process)

The value *abs_diff_pic_num_minus1*$+1$ signals a positive or negative offset from a **predicted** reference picture number. For the first reordering or remapping instruction, the predicted picture number predictedPicNum is the current picture number. For subsequent reordering instructions, the predicted picture number is the picture number of the last remapped picture. The picture to be remapped or moved, remappedPicNum, is selected according to the following rules shown in Table 5.3.

Reference Picture Re-Ordering Example:
The current slice is a P slice. The DPB contains five reference frames, indexed in list0. Three of these are short-term frames with PicNum equal to 153, 155, and 157, respectively. The other two are long-term frames with LongTermPicNum equal to *1* and *3*, respectively. The current *frame_num* is 158. The default reference list order (list0) is as follows (Table 5.4).

Table 5.4 Initial reference picture list

Index	0	1	2	3	4
Picture	157	155	153	*1*	*3*

The following series of four reference picture reordering commands are received.
Initial predictedPicNum = 158, the current picture; initial refIdxL0 = 0

1. *reordering_of_pic_nums_idc* = *0*, *abs_diff_pic_num_minus_1* = *4*
   remappedPicNum = 158 − 5 = 153
   Put picture 153 at position refIdxL0=0 (the start of the list).
   New list: **153**, 157, 155, *1*, *3*
   New predictedPicNum = 153; new refIdxL0 = 1.
2. *reordering_of_pic_nums_idc* = *1*, *abs_diff_pic_num_minus_1* = *1*
   remappedPicNum = 153 + 2 = 155
   Put picture 155 at position refIdxL0=1, second in the list.
   New list: 153, **155**, 157, *1*, *3*
   New predictedPicNum = 155; new refIdxL0 = 2.
3. *reordering_of_pic_nums_idc* = *2*, *long_term_pic_num* = 3
   remappedPicNum= *3*
   Put long term picture *3* at position refIdxL0=2, third in the list.
   New list: 153, 155, **3**, 157, *1*
4. *reordering_of_pic_nums_idc* = 3
   End of reordering process.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000135

The final list0 order is as follows (Table 5.5).

Table 5.5 Final reference picture list

Index	0	1	2	3	4
Picture	153	155	*3*	157	*1*

5.3.4 Frame and field coding

A video sequence may be coded in Frame mode only, with no special coding of fields or interlaced video, or in Frame/Field mode, using special coding tools for interlaced video, signalled by the syntax element *frame_mbs_only_flag* in the Sequence Parameter Set (section 5.5). If this flag is set to zero, special coding of fields or interlaced video is enabled.

5.3.4.1 Coding pictures in frame or field mode

If frame/field coding is enabled, each frame of video, a pair of fields, may be coded as a complete frame, i.e. one picture = one frame, or as two separate fields, i.e. one picture = one field. The syntax element *field_pic_flag* is signalled in each Slice Header (section 5.6.2) and indicates whether the current coded picture is a frame or a field. This gives rise to the following two cases:

1. Picture is coded as a complete frame: The complete frame, which happens to include two fields, is coded as one or more slices, each containing an integral number of macroblocks. Reference picture list(s), used for motion compensated prediction of P- or B-macroblocks, are constructed from previously coded **frames**, i.e. each reference picture is a complete coded frame. Block coefficients are scanned in a zig-zag order (Chapter 7). Macroblocks are optionally coded as macroblock pairs in Macroblock Adaptive Frame Field Mode.
2. Picture contains one field: The current top or bottom field is coded as one or more slices, each containing a number of macroblocks. Reference picture list(s) are constructed from previously coded **fields**, i.e. each reference picture is a coded field. Each field of a stored reference frame is identified as a separate reference picture with a unique index[1]. An inter-coded macroblock is predicted from region(s) in previously coded field(s). Block coefficients are coded in a modified or field scan order (Chapter 7).

The coded frames in the Decoded Picture Buffer (DPB) may therefore be accessed as complete frames, containing top + bottom field, or as individual fields, depending on whether the current picture is coded in frame or field mode.

An encoder may choose to switch adaptively between frame and field coding during a sequence, for example to improve the compression performance for interlaced video or for progressive video that was converted from an interlaced source. For an interlaced video

[1] When decoding a field, there are effectively twice as many reference pictures available as there would be when decoding a frame at the same position.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000136

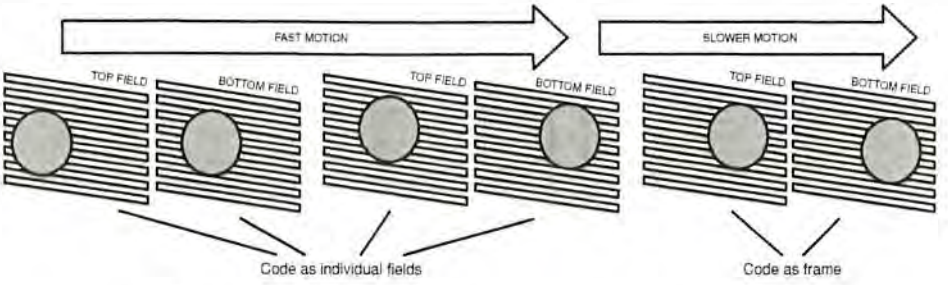


Figure 5.9 Picture Adaptive Frame Field Coding example

sequence, field coding tends to be more efficient when there is significant motion in the scene which tends to give larger changes between successive fields, whereas frame coding tends to be more efficient in more static areas of the scene where there is less change between successive fields. Switching between frame and field coding at the picture level is known as Picture Adaptive Frame Field Coding (PAFF).

Example:

Figure 5.9 shows a sequence of six fields from an interlaced sequence or from a progressive sequence that was converted from interlaced content. The first four contain rapid motion, a fast-moving ball, and so the encoder chooses to code each field as an individual picture using field coding mode. Each field is coded as a unit; reference picture list(s) consist of previously-coded fields; block scanning uses the modified field scan described in Chapter 7. The last two fields contain slower motion and the encoder chooses to code these as a complete frame. The combined frame is coded as a single picture; reference picture list(s) consist of previously-coded frames; block scanning uses the zig-zag progressive scan described in Chapter 7.

5.3.4.2 Coding macroblocks in frame or field mode (MBAFF)

Macroblock Adaptive Frame Field Coding (MBAFF) [iii] is a coding mode that is enabled in the Sequence Parameter Set (section 5.5) and is available thereafter in any slice coded in Frame mode. MBAFF makes it possible to adaptively switch between frame and field coding within a single frame, for example to maximise coding efficiency. Macroblocks are handled in pairs consisting of two vertically adjacent macroblocks (Figure 5.10).

For each macroblock pair, *mb_field_decoding_flag*, signaled in the header of the upper macroblock, indicates whether the pair is coded as a frame macroblock pair or a field macroblock pair (Figure 5.11). If it is a **frame MB pair**, then the top and bottom macroblocks are coded separately as frame macroblocks. Blocks are scanned using zig-zag scan (Chapter 7). If it is a **field MB pair**, alternate lines of the MB pair are mapped to top and bottom field macroblocks as shown in the figure. Each field macroblock is predicted from the corresponding field in reference picture(s) (Chapter 6). Blocks in a field MB are scanned using field scan (Chapter 7).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000137

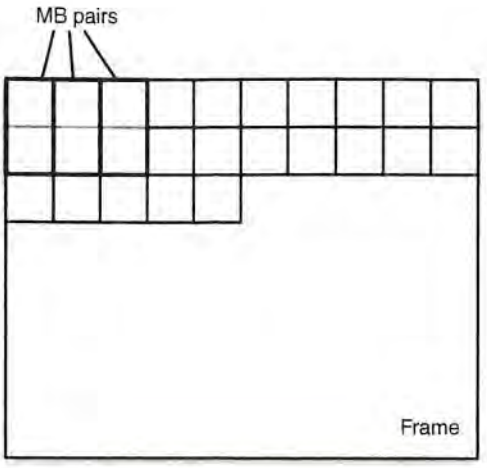


Figure 5.10 Frame with macroblock pairs

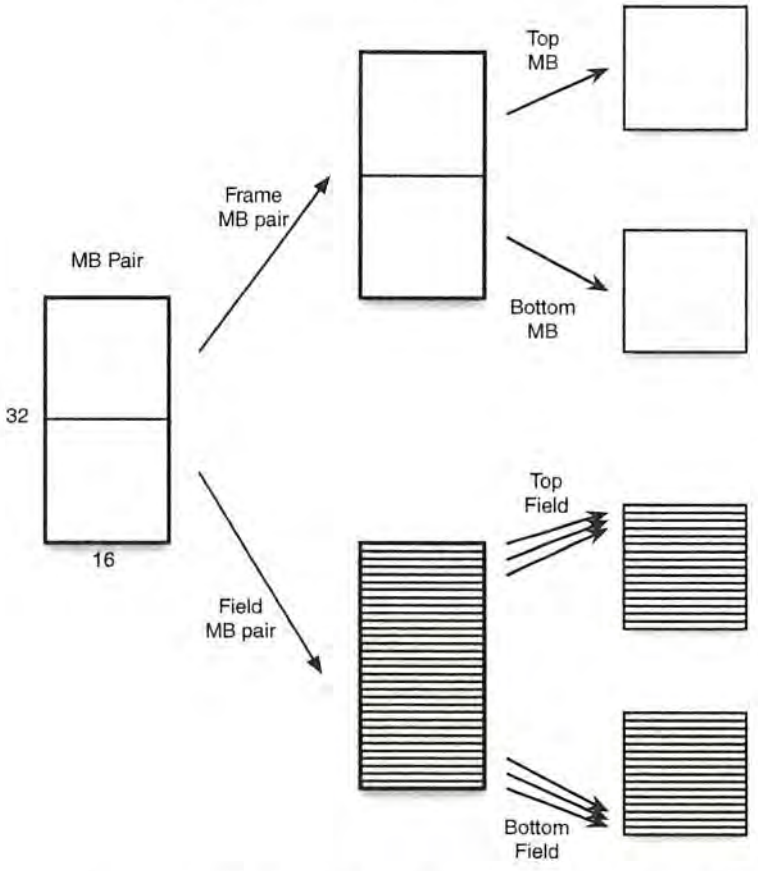


Figure 5.11 MB pair coded using Macroblock Adaptive Frame Field Coding

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000138

5.4 NAL unit

Coded H.264 data is stored or transmitted as a series of packets known as Network Abstraction Layer Units, NAL Units or NALUs. Each NAL Unit consists of a 1-byte NALU header followed by a byte stream containing control information or coded video data. The header indicates the NALU type, some of which are listed in Table 5.6, and the 'importance' of the NALU. Parameter Sets and slices that are used for reference, i.e. used to predict further frames, are considered important or high priority, since their loss could make it difficult to decode subsequent coded slices. Non-reference slices are considered to be less important to the decoder, since their loss will not affect any further decoding. This information can optionally be used to prioritise certain NALUs during transmission (Chapter 8).

Coded slices are described as Video Coding Layer (VCL) NAL units. A coded sequence begins with an Instantaneous Decoder Refresh (IDR) Access Unit, made up of one or more IDR slices, each of which is an Intra coded slice. This is followed by the default slice type, i.e. a non-IDR coded slice, and/or by Data Partitioned slices. Data Partitioned slices carry different components of coded video data in separate NAL units, which may be useful in error-prone transport situations. Non-VCL NAL units include Parameter Sets, Supplemental Enhancement Information parameters that may be useful for decoding and displaying video data, but are not essential for correct decoding, and delimiters that define boundaries between coded sections.

Each NALU may be transported in one or more network packets, streamed sequentially as a byte stream or encapsulated in a media file (Chapter 8).

Table 5.6 Selected NAL unit types

NAL Unit	Description	VCL
Coded slice, non-IDR	A typical coded slice, see section 5.6	Yes
Coded slice data partition A	Part of a data partitioned slice, see Chapter 8	Yes
Coded slice data partition B	Part of a data partitioned slice, see Chapter 8	Yes
Coded slice data partition C	Part of a data partitioned slice, see Chapter 8	Yes
Coded slice, IDR	Part of the Access Unit at the start of a coded video sequence	Yes
SEI	Supplemental Enhancement Information, see Chapter 8	No
SPS	Sequence Parameter Set, one per sequence, see section 5.5	No
PPS	Picture Parameter Set, see section 5.5	No
Access unit delimiter	Indicates type of the slices in the next coded picture (optional)	No
End of sequence	Indicates that the next access unit is IDR, i.e. the start of a new coded video sequence (optional)	No
End of stream	Indicates the end of the coded bitstream (optional)	No
Filler	Filler bytes (optional), e.g. to prevent buffer under-run, see Chapter 8.	No
.....		

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000139

5.5 Parameter Sets

Parameter Sets are NAL units that carry decoding parameters common to a number of coded slices. Sending these parameters independently of coded slices can improve efficiency, since common parameters need only be sent once. Parameter Sets are essential to the correct decoding of a sequence. In a lossy transmission scenario, where bits or packets may be lost or corrupted during transmission, Parameter Sets may be sent with a higher quality of service using e.g. Forward Error Correction or a priority mechanism.

A Sequence Parameter Set (SPS) contains parameters common to an entire video sequence or programme, such as the Profile and Level (Chapter 8), the size of a video frame and certain decoder constraints such as the maximum number of reference frames (section 5.3.3). Table 5.7 shows an example of a SPS for a Baseline Profile, QCIF coded sequence. Note that each SPS has a unique identifier, *seq_parameter_set_id*, which is 0 in this case.

A Picture Parameter Set (PPS) contains common parameters that may apply to a sequence or subset of coded frames, such as entropy coding type, number of active reference pictures and initialization parameters. See Table 5.8 for an example. The PPS has its own identifier, *pic_parameter_set_id*, and 'points' to a SPS identifier. In this example, the PPS 'inherits' the parameters of SPS 0. Each coded slice 'points' to a PPS and inherits its parameters. In this way, there is considerable flexibility to set up common coding parameters via SPS and PPS NAL units and then to refer to these in subsequent coded slices.

Table 5.7 Sequence Parameter Set example

Parameter	Binary code	Symbol	Discussion
profile_idc	1000010	66	Baseline Profile
constrained_set0_flag	0	0	Bistream might not obey all the constraints of the Baseline Profile
constrained_set1_flag	0	0	As above, Main Profile
constrained_set2_flag	0	0	As above, Extended Profile
constrained_set3_flag	0	0	Used to specify the special case of Level 1b
reserved_zero_4bits	0	0	Not used
level_idc	11110	30	Level 3
seq_parameter_set_id	1	0	Sequence Parameter Set 0
log2_max_frame_num_minus4	1	0	frame_num will not exceed 16.
pic_order_cnt_type	1	0	Default POC
log2_max_pic_order_cnt_lsb_minus4	1	0	LSB of POC will not exceed 16.
num_ref_frames	1011	10	Up to 10 reference frames.
gaps_in_frame_num_value_allowed_flag	0	0	No gaps in frame_num.
pic_width_in_mbs_minus1	1011	10	11 macroblocks wide = QCIF
pic_height_in_map_units_minus1	1001	8	9 MBs high = QCIF
frame_mbs_only_flag	1	1	No field slices or field MBs
direct_8 × 8_inference_flag	1	1	Specifies how certain B macroblock motion vectors are derived
frame_cropping_flag	0	0	Frames are not cropped
vui_parameters_present_flag	0	0	VUI parameters not present

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000140

Table 5.8 Picture Parameter Set example

Parameter	Binary code	Symbol	Discussion
pic_parameter_set_id	1	0	Picture Parameter Set 0
seq_parameter_set_id	1	0	Use SPS 0
entropy_coding_mode_flag	0	0	CAVLC entropy coding
pic_order_present_flag	0	0	POC not present
num_slice_groups_minus1	1	0	One slice group
num_ref_idx_l0_active_minus1	1010	9	10 reference pictures in list0
num_ref_idx_l1_active_minus1	1010	9	10 reference pictures in list1
weighted_pred_flag	0	0	Weighted prediction not used
weighted_bipred_idc	0	0	Weighted biprediction not used
pic_init_qp_minus26	1	0	Initial QP (luma) = 26
pic_init_qs_minus26	1	0	Initial SI/SP QP=26
chroma_qp_index_offset	1	0	No chroma QP offset
deblocking_filter_control_present_flag	0	0	Use default filter parameters
constrained_intra_pred_flag	0	0	Intra prediction is not constrained
redundant_pic_cnt_present_flag	0	0	Redundant picture count parameter is not used

Activation of Parameter Sets

A Parameter Set is 'inactive' until it is activated, i.e. its parameters are not used by the decoder. A PPS, previously transmitted to the decoder, is activated when it is referred to in a slice header and remains active until a different PPS is activated. A SPS is activated when a PPS that refers to it, e.g. the PPS in Table 5.8, is activated. A single SPS remains active for an entire coded video sequence, which must start with an IDR Access Unit, hence a SPS is effectively activated by an IDR slice. Each coded slice in a video sequence may refer to one of a number of Picture Parameter Sets.

Using parameter sets: Example

SPS0 is sent at the start of a sequence, followed by PPS0 and PPS1, both of which 'inherit' SPS0 (Figure 5.12). PPS0 is activated by IDR slice 0, which means that SPS0 becomes active at the same time. Slices 1 and 2 use the parameters of PPS0 and SPS0. PPS1 is activated by slice 3, making PPS0 inactive, hence slice 3 uses the parameters of PPS1 and SPS0; PPS0 is activated again by slice 4, making PPS1 inactive.

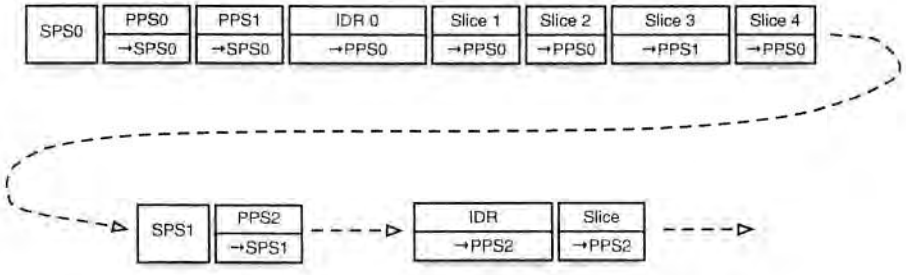


Figure 5.12 Example: Sequence and Picture Parameter Sets

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000141

At a later point, SPS1 and PPS2 are sent: PPS2 inherits SPS1. Both SPS1 and PPS2 are inactive until a new IDR slice is sent, activating both SPS1 and PPS2 and signalling the start of a new coded video sequence. Subsequent slices use the parameters of SPS1 and PPS2.

5.6 Slice layer

5.6.1 Slice types

Each coded picture, a coded frame or field, is composed of one or more slices, each containing a **slice header** followed an integral number of macroblocks. The number of macroblocks in a slice need not be constant. There are minimal data inter-dependencies between coded slices, which can help to limit the propagation of errors between slices. Possible scenarios for choosing slice sizes include:

- One slice per coded picture, common practice in many H.264 encoding applications.
- N slices per picture, each containing M macroblocks, N and M integer. The number of bytes in each coded slice will tend to vary depending on the amount of motion and detail in the picture area.
- N slices per picture, containing a varying number of macroblocks, chosen to keep the number of bytes per slice roughly constant. This may be useful if, for example, each slice is mapped to a fixed-size network packet.

The available slice types are listed in Table 5.9, together with the macroblock types that may be present in each slice. Note that a B slice (for example) may contain B, P and/or I macroblock types and an encoder may choose between these types depending on the content of the sequence. Macroblock types are explained further in section 5.7.1.

5.6.2 Slice header

The slice header conveys information common to all macroblocks in the slice, such as the slice type which determines which macroblock types are allowed, the frame number that the slice corresponds to, reference picture settings and default quantization parameter (QP). Table 5.10

Table 5.9 Slice types in H.264

Slice type	Contains macroblock types	Notes
I (including IDR)	I only	Intra prediction only.
P	I and/or P	Intra prediction (I) and/or prediction from one reference per macroblock partition (P).
B	I, P and/or B	Intra prediction (I), prediction from one reference (P) or biprediction, i.e. prediction from two references (B)
SP	P and/or I	Switching P slice, see Chapter 8
SI	SI	Switching I slice, see Chapter 8

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000142

Table 5.10 Slice Header, IDR/Intra, Frame 0

Parameter	Binary code	Symbol	Discussion
first_mb_in_slice	1	0	First MB is at position 0, the top-left position in the slice.
slice_type	1000	7	I slice, contains only I MBs
pic_parameter_set_id	1	0	Use PPS 0
frame_num	0	0	Slice is in frame 0
idr_pic_id	1	0	IDR #0 : only present in IDR picture
pic_order_cnt_lsb	0	0	Picture order count = 0
no_output_of_prior_pics_flag	0	0	Not used
long_term_reference_flag	0	0	Not used
slice_qp_delta	1000	4	QP = initial QP + 4 = 30

shows an example of a slice header from an I slice with frame_mbs_only_flag set, i.e. no field coding. This is the first I slice and is therefore an IDR slice. The default QP is set to the initial sequence value (26) plus 4.

The header of a P slice is shown in Table 5.11. This is frame 1; the picture order count (POC) is 2. Note that POC increases by 2 for every complete frame, see section 5.3.2. There is one active reference picture which happens to be frame 0.

5.6.3 Slice data

The slice data section consists of a series of macroblocks that make up a slice. A macroblock containing no data, a skip macroblock, is a very common occurrence in many coded sequences. It is signalled by the parameters mb_skip_run, a count of a sequence of skipped macroblocks, used with CAVLC entropy coding mode, or mb_skip_flag, indicating a single skipped macroblock, used with CABAC entropy coding mode. A skip macroblock can only

Table 5.11 Slice Header, Inter, Frame 1

Parameter	Binary code	Symbol	Discussion
first_mb_in_slice	1	0	First MB at position 0
slice_type	110	5	P slice, can contain I or P MBs
pic_parameter_set_id	1	0	Use PPS 0
frame_num	1	1	Frame 1
pic_order_cnt_lsb	10	2	Picture 2
num_ref_idx_active_override_flag	1	1	Default number of reference pictures overridden by following parameter
num_ref_idx_l0_active_minus1	1	0	One reference picture in List 0
ref_pic_list_reordering_flag_l0	0	0	No re-ordering of reference pictures
adaptive_ref_pic_buffering_flag	0	0	Reference pictures handled as 'first in / first out' (default)
slice_qp_delta	1000	4	QP = 26 + 4 = 30

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000143

occur in a P, SP or B slice. A complete slice data section consists of coded and skipped macroblocks that comprise a single coded slice (Figure 5.1).

5.7 Macroblock layer

5.7.1 Overview

The Macroblock Layer contains all the syntax elements necessary to decode a single macroblock (Figure 5.13).

mb_type indicates the macroblock coding type, I, SI, P or B, and further information about macroblock prediction and coding. An I macroblock (Intra) can occur in any slice type and is coded without reference to any other slices. An SIntra MB (SI) occurs only in switching or SI

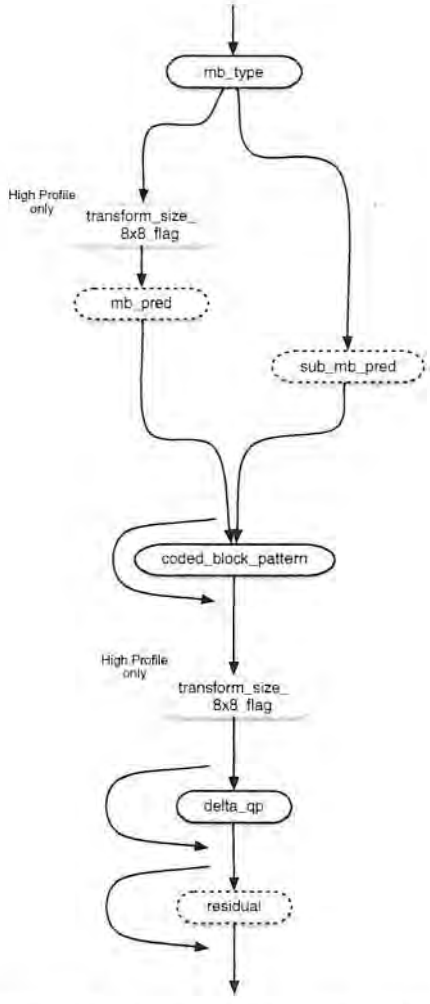


Figure 5.13 Macroblock layer syntax overview

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000144

Table 5.12 Macroblock types

Macroblock coding type	Further information signalled in mb_type codeword
I	4 × 4 : indicates 4 × 4 Intra prediction mode.
	8 × 8 : indicates 8 × 8 Intra prediction mode.
	16 × 16 : indicates 16 × 16 Intra prediction mode; luma and chroma coded block pattern is also indicated by mb_type.
	PCM: indicates that the special PCM mode is used (section 5.7.2).
SI	None.
P	16 × 16, 16 × 8, 8 × 16: indicates partition size.
	8 × 8: indicates 8 × 8 partition size, a further syntax element sub_mb_type and optionally a reference index are also transmitted.
	P_Skip (inferred): indicates Skip mode, no further data transmitted.
B	16 × 16: indicates 16 × 16 partition size. Prediction mode may be Direct, no motion vectors or references transmitted, Pred, prediction from one reference picture in list 0 or list 1, or BiPred, prediction from two references, one from list 0 and one from list 1.
	16 × 8: indicates 16 × 8 partition size plus prediction modes of each partition, either Pred from L0, Pred from L1 or BiPred.
	8 × 16: indicates 8 × 16 partition size plus prediction modes of each partition, Pred from L0 or L1, or BiPred.
	8 × 8: indicates 8 × 8 partition size. A further syntax element sub_mb_type is sent for each 8 × 8 partition.
	B_Skip (inferred): indicates Skip mode, no further data transmitted.

slices (Chapter 8). A P macroblock (Predicted) can occur in P or SP slices and is inter-coded with one prediction reference. A B macroblock (BiPredicted) can occur in B slices and is inter-coded with one or two prediction references. Table 5.12 summarizes the information signalled by the codeword mb_type. Note that mb_type is not actually transmitted for skip MBs in P or B slices – these macroblock types are inferred whenever mb_skip is signalled in the slice data layer.

transform_size_8 × 8_flag is only present in High Profile bitstreams. It occurs in one of two places (Figure 5.13) depending on the macroblock type and does not occur in an Intra 16 × 16 macroblock. This flag indicates that an optional 8 × 8 integer transform is used for luma blocks, instead of the default 4 × 4 integer transform (Chapter 7).

mb_pred, for all macroblock types except P/B with 8 × 8 partition size, or **sub_mb_pred**, for P/B macroblocks with 8 × 8 partition size, indicates the type of intra or inter prediction used for the macroblock (section 5.7.3 and Chapter 6).

coded_block_pattern is a separate syntax element for macroblocks other than Intra_16 × 16_ types. It can take values between 0 and 47 and is constructed as follows:

(i) The four LSBs of coded_block_pattern, $b_3b_2b_1b_0$, indicate whether there are one or more non-zero transform coefficients within each of the four 8 × 8 luma blocks. A 1 = non-zero coefficient(s) present, 0 = no coefficients present.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000145

Table 5.13 coded_block_pattern examples.

Pattern : ✓ = coefficients MAY BE present, 0 = no coefficients	Coded_block_pattern
Luma 8x8 blocks: ✓ ✓ / 0 ✓ — Chroma DC: ✓ — Chroma AC: 0 0	**01**1011$_2$ = 27$_{10}$
Luma 8x8 blocks: 0 0 / 0 ✓ — Chroma DC: 0 — Chroma AC: 0 0	**00**1000$_2$ = 8$_{10}$
Luma 8x8 blocks: ✓ ✓ / ✓ ✓ — Chroma DC: ✓ — Chroma AC: ✓ ✓	**10**1111$_2$ = 47$_{10}$

(ii) The two MSBs, b_5b_4, can take the values 00_2, indicating no chroma coefficients present, 01_2, indicating chroma DC coefficients are present, no chroma AC coefficients, or 10_2, indicating chroma DC coefficients may be present, chroma AC coefficients are present.

Table 5.13 shows three examples; a tick indicates that coefficients may be present in the block, a 0 indicates that no non-zero coefficients are present.

delta_qp indicates a change in quantization parameter (QP), either positive or negative, from its previous value. If there is no change in QP from the previous value, delta_qp = 0.

If residual coefficient data is present, i.e. if there are non-zero coefficients in the macroblock, as indicated by a non-zero CBP, **residual_data** is then transmitted (section 5.7.4).

5.7.2 The Intra PCM mode

The Intra PCM (Pulse Code Modulation) mode is an optional mode of coding a macroblock, signalled by mb_type I_PCM. If I_PCM mode is chosen, the usual prediction, transform and

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000146

coding processes are bypassed. Instead, each luma and chroma sample is transmitted directly as follows:

1. Send zero bits until the current position in the bitstream is byte aligned.
2. Send every luma sample, 256 samples in total, as an individual BitDepth_Y-bit value.
3. Send every chroma sample, 64 samples for 4:2:0 format, as an individual BitDepth_C-bit value.

BitDepth_Y and BitDepth_C are the number of bits used to represent an uncompressed luma or chroma sample respectively. Each BitDepth parameter defaults to 8, i.e. 1 byte per sample, but higher bit depths may be set in the SPS for certain Profiles (Chapter 8).

Clearly, I_PCM mode does not provide compression, as the 'raw' luma and chroma samples are simply inserted into the H.264/AVC bitstream. However, there are situations when I_PCM mode may be useful, for example:

1. When video is coded at a very high perceived quality / very low QP setting, for example, for applications such as content archiving or distribution where high quality is required, there may occasionally be macroblocks which, when coded using the usual processes of prediction, transform and entropy coding, generate **more** bits than the original 'raw' data.
2. If I_PCM mode is selected for every macroblock, the H.264/AVC bitstream becomes a 'container' for uncompressed video. This makes it possible for an application to make use of all the other features of H.264/AVC such as packet handling, frame numbering and transport support, whilst retaining the original video samples.

5.7.3 Macroblock prediction

The macroblock prediction syntax elements indicate how intra or inter prediction is carried out for the current macroblock. Sub_mb_pred is used for P or B macroblocks coded with 8×8 partition size and mb_pred is used for all other cases (Figure 5.14). For a detailed description of intra and inter prediction, see Chapter 6.

I macroblock: If the prediction type is 4×4 or 8×8, the prediction modes for each 4×4 or 8×8 luma block are signalled. Note that if the prediction type is 16×16, the mode has already been signalled as part of mb_type (section 5.7.1). The chroma prediction mode is signalled. Section 5.2.5 explains how these modes are coded.

B macroblock, Direct 16×16 mode: No further prediction information is sent: List0 and List1 reference frame indices and List 0 and List 1 motion vectors are derived from previously coded macroblocks (Chapter 6).

P or B macroblock, 8×8 partition size: For each of the four 8×8 partitions or sub-macroblocks, sub_mb_type is sent indicating for the sub-macroblock:

(a) Direct mode or
(b) Sub-macroblock partition size, 8×8, 8×4, 4×8 or 4×4, and prediction sources, List 0, List 1 and/or BiPredicted. This is followed by some or all of the following elements, depending on sub_mb_type:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000147

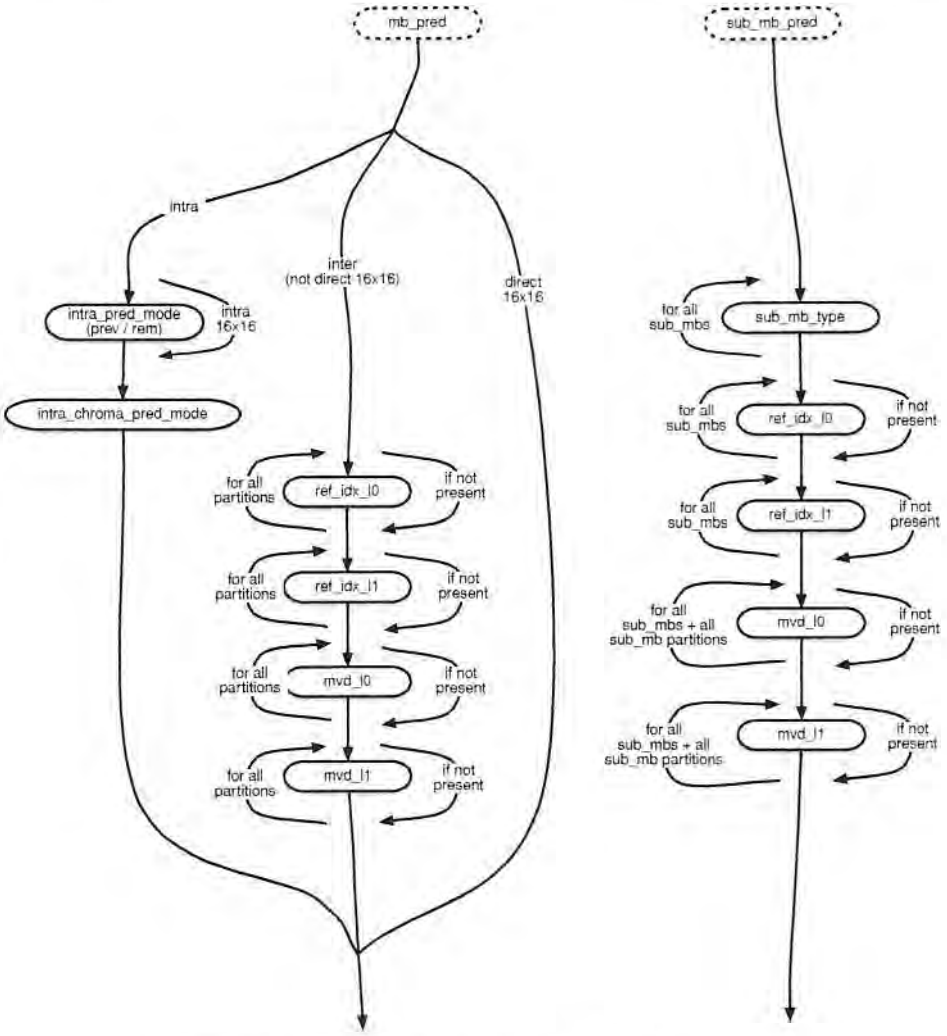


Figure 5.14 mb_pred and sub_mb_pred syntax overview

(i) Reference picture index/indices for List 0 and/or List 1, one index or one pair of indices for each sub-macroblock
(ii) Motion vector differences (mvd) for List 0 and/or List 1.

Note that all sub-macroblock partitions in a sub-macroblock share the same reference picture(s), whereas each sub-macroblock partition has its own motion vector (x,y) pair.

All other P or B macroblocks: Some or all of the following are sent for each macroblock partition, i.e. one 16×16 partition, two 8×16 partitions or two 16×8 partitions:

(a) Reference picture index/indices for List 0 and/or List 1
(b) Motion vector differences for List 0 and/or List 1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000148

Table 5.14 Type and prediction elements for B macroblocks, excluding Direct or 8 × 8 partitions

Partition	Type (part 0)	Type (part 1)	Number of reference indices	Number of mvd (x,y) pairs
16 × 16	L0 or L1	N/A	1	1
16 × 16	Bipred	N/A	2 : one for each List	2
16 × 8	L0 or L1	L0 or L1	2 : one for each partition	2
"	L0 or L1	Bipred	3 : one for partition 0, two for partition 1	3
"	Bipred	L0 or L1	3 : two for partition 0, one for partition 1	3
"	Bipred	Bipred	4 : two for each partition	4
8 × 16	Same options as for 16 × 8			

The motion vector differences (mvd) are added to a predicted motion vector to create the x and y components of each motion vector (Chapter 6).

As an example, Table 5.14 lists the type and prediction elements, reference indices and motion vector difference (mvd) pairs, for B macroblocks excluding the special cases of Direct and 8 × 8 partition types. Each partition has one or two reference frames, one from List 0, one from List 1 or one from each List, i.e. biprediction, and a corresponding number of mvd(x,y) pairs signalled in the bitstream.

Note that each syntax element is only sent if it is needed. For example, if there is only one reference frame in a List, it is not necessary to send a reference index for that List and so the decoder does not expect to receive one.

5.7.4 Residual data

Residual data for a complete macroblock, if present, is sent according to the syntax summarised in Figure 5.15. Figure 5.16 shows the sequence in which residual blocks are coded. First, a 4 × 4 block of transformed luma DC coefficients is sent if Intra 16 × 16 mode was chosen for this macroblock. This optional first block is labelled '-1' in the Figure 5.16. Then, the luma component is sent in 8 × 8 block order. If coded_block_pattern indicates the 8 × 8 block contains no coefficients (section 5.7.1), that block is bypassed. Each 8 × 8 luma block is processed using four 4 × 4 transforms (Figure 5.16) or one 8 × 8 transform (Figure 5.17, High Profiles only).

After all the coded luma transform coefficients are sent, two blocks of transformed chroma DC coefficients are transmitted, if coded_block_pattern indicates they are present, e.g. for blocks 16 and 17 in Figure 5.16, blocks 4 and 5 in Figure 5.17. Finally, the chroma AC blocks are transmitted if present. If the sampling format is 4:2:0 (Chapter 2), each chroma DC block contains 2 × 2 samples and each chroma AC block contains 8 × 8 samples. These sizes change if 4:2:2 or 4:4:4 sampling is employed (Chapter 7).

Each residual block is coded using CAVLC or CABAC. The CAVLC and CABAC block syntax is summarized in Figure 5.18 and Figure 5.19 and described in detail in Chapter 7.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000149

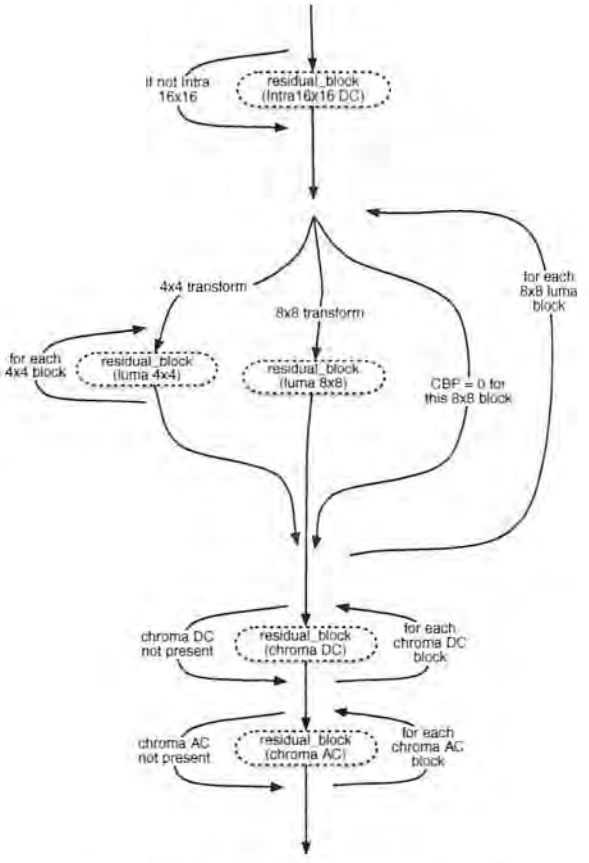


Figure 5.15 Residual data syntax overview

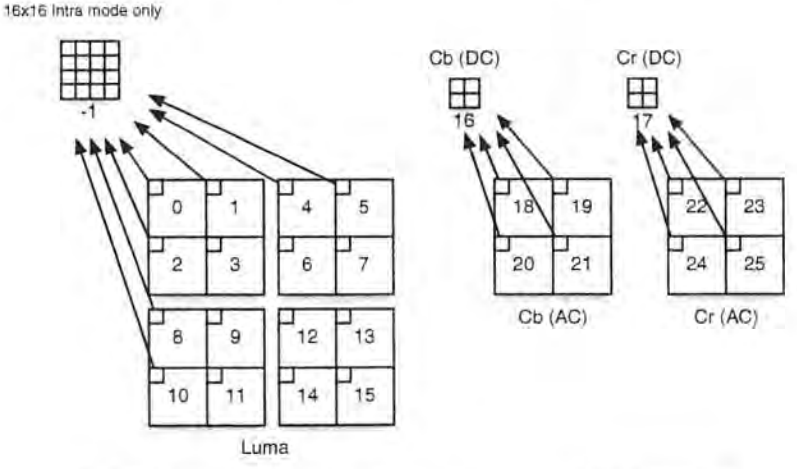


Figure 5.16 Block scanning order, 4 × 4 transform, 4:2:0 sampling

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000150

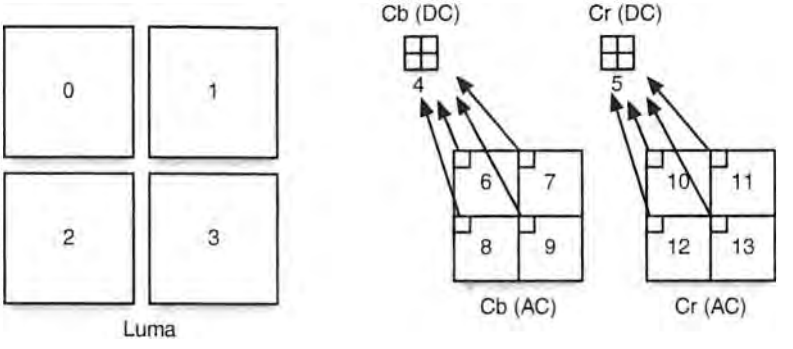


Figure 5.17 Block scanning order, 8 × 8 luma transform, 4:2:0 sampling

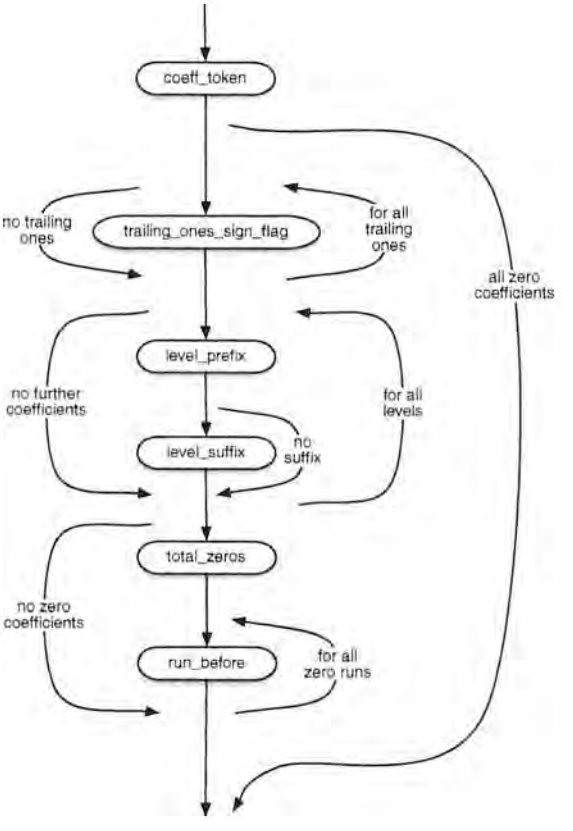


Figure 5.18 Residual CAVLC block syntax overview

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000151

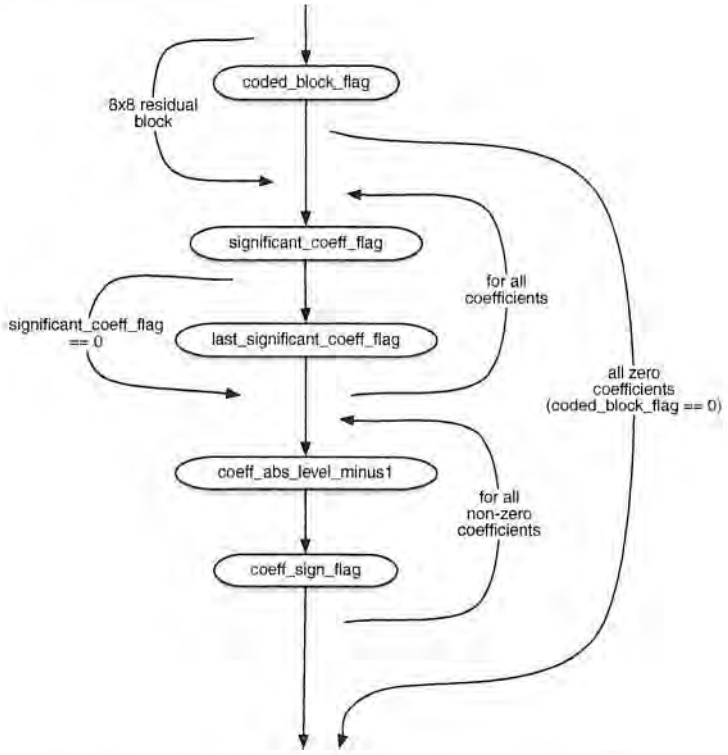


Figure 5.19 Residual CABAC block syntax overview

5.7.5 *Macroblock syntax examples*

The following examples are taken from trace files generated with the JM H.264 reference software codec [i].

1. Intra coded macroblock
   Table 5.15 shows the first part of a trace for an Intra coded macroblock. The first element, mb_type, indicates that this is an Intra macroblock using 4×4 prediction. The prediction modes of each 4×4 block are signalled, followed by the prediction mode of the chroma blocks. See Chapter 6 for a detailed description of how the intra mode is signalled using Most Probable and Rem parameters. Coded_block_pattern is $31_{10} = \mathbf{01}1111_2$, i.e. all luma blocks and the chroma DC blocks contain non-zero coefficients, but there are no coefficients present for chroma AC.
2. P macroblock, one reference frame
   In this example, illustrated in Table 5.16 and Figure 5.20, a P macroblock is coded using 16×16 mode, i.e. one motion vector per macroblock. There is only one reference frame and so the choice of reference is not signalled. MVDx = 0 and MVDy = −1, indicating

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000152

Table 5.15 I macroblock, example 1

Parameter	Binary code	Symbol	Discussion
mb_type (I_SLICE) (0, 1) = 9	1	0	Intra coding, 4 × 4 prediction of each luma block.
Intra mode = 3 0	0011	0,3	Rem = 3
Intra mode = −1 1	1	1	Most probable (MP)
Intra mode = 2 2	0010	0,2	Rem = 2
Intra mode = −1 3	1	1	MP
Intra mode = −1 4	1	1	MP
Intra mode = 2 5	0010	0,2	Rem = 2
Intra mode = 5 6	0101	0,5	Rem = 5
Intra mode = 5 7	0101	0,5	Rem = 5
Intra mode = 5 8	0101	0,5	Rem = 5
Intra mode = −1 9	1	1	MP
Intra mode = −1 10	1	1	MP
Intra mode = 1 11	0001	0,1	Rem = 1
Intra mode = −1 12	1	1	MP
Intra mode = 2 13	0010	0,2	Rem = 2
Intra mode = −1 14	1	1	MP
Intra mode = 3 15	0011	0,3	Rem = 3
Chroma intra pred mode	01	0	Mode 0
CBP (0, 1) = 31	10	31	Coefficients present for all luma blocks and for chroma DC blocks.
Delta QP (0, 1) = 0	1	0	No change in QP
Luma # c & tr.1s(0,0) vlc=2 #c=8 #t1=3	1101	8	Coefficient token (see Chapter 7)
Luma trailing ones sign (0,0)	0	0	Trailing ones
Luma lev (0,0) k=4 vlc=0 lev= 1	1	1	Level
Luma lev (0,0) k=3 vlc=1 lev= −1	11	−1	Level
Luma lev (0,0) k=2 vlc=1 lev= −2	11	−2	Level
Luma lev (0,0) k=1 vlc=1 lev= −4	11	−4	Level
Luma lev (0,0) k=0 vlc=2 lev= 3	100	3	Level
Luma totalrun (0,0) vlc=7 totzeros= 1	1	1	Total number of zeros
Luma run (0,0) k=7 vlc=0 run= 0	1	0	Zero run
Luma run (0,0) k=6 vlc=0 run= 1	0	1	Zero run
... (etc)			

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000153

Table 5.16 P macroblock, example 2

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No preceding skip
mb_type (P_SLICE) (0, 2) = 1	1	0	16 × 16 mode
mvd_l0 (0) = 0 (org_mv 0 pred_mv 0)	1	0	X vector difference
mvd_l0 (1) = −1 (org_mv 0 pred_mv 1)	11	−1	Y vector difference
CBP (0, 2) = 1	11	1	000001_2: Only first block has coefficient data
Delta QP (0, 2) = 0	1	0	No change in QP
Luma # c & tr.1s(0,0) vlc=0 #c=3 #t1=2	101	3	Coefficient token
Luma trailing ones sign (0,0)	10	2	Trailing ones
Luma lev (0,0) k=0 vlc=0 lev= −2	1	−1	Level
Luma totalrun (0,0) vlc=2 totzeros= 2	110	2	Total zeros
Luma run (0,0) k=2 vlc=1 run= 1	1	1	Zero run
Luma run (0,0) k=1 vlc=0 run= 0	1	0	Zero run
Luma # c & tr.1s(1,0) vlc=1 #c=0 #t1=0	11	0	Zero 4 × 4 block
Luma # c & tr.1s(0,1) vlc=1 #c=0 #t1=0	11	0	"
Luma # c & tr.1s(1,1) vlc=0 #c=0 #t1=0	1	0	"

an offset of $-1/4$ sample in the y direction from the predicted motion vector (Chapter 6). The predicted vector is $(0,+1/4)$ so the actual motion vector is (0, 0). Only one 8 × 8 block, the top-left, contains non-zero coefficients. Within this 8 × 8 block, only the top-left 4 × 4 block contains non-zero coefficients. CBP indicates that coefficient information is only sent for the first 8 × 8 luma quadrant; non-zero coefficients are signalled for the first 4 × 4 block in this quadrant. The decoder fills the remaining 4 × 4 blocks with zeros.

3. P macroblock, one reference frame
   Another P macroblock example is shown in Table 5.17. The two preceding macroblocks are skipped. The macroblock uses 8 × 16 prediction mode with one reference frame. The MVD for the first or left-hand partition is $(0, 1/4)$ and MVD for the second or right-hand

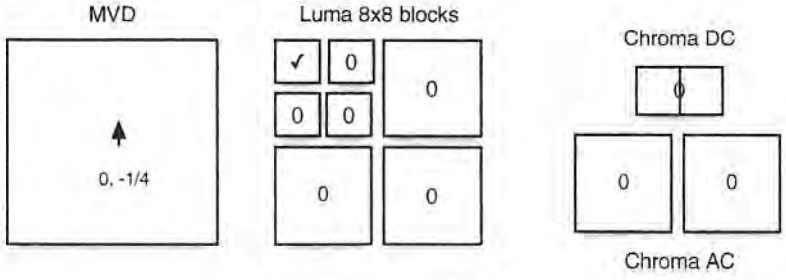


Figure 5.20 P macroblock, example 2

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000154

Table 5.17 P macroblock, example 3

Parameter	Binary code	Symbol	Discussion
mb_skip_run	11	2	Two previous MBs are skipped
mb_type (P_SLICE) (0, 3) = 3	11	2	8×16 mode
mvd_l0 (0) = 0	1	0	Partition 0 MVDx
mvd_l0 (1) = 1	10	1	Partition 0 MVDy
mvd_l0 (0) = −1	11	−1	Partition 1 MVDx
mvd_l0 (1) = 0	1	0	Partition 1 MVDy
CBP (0, 3) = 1	11	1	000001_2: Only first 8×8 block non zero
Delta QP (0, 3) = 0	1	0	No change in QP
Luma # c & tr.1s(0,0) vlc=0 #c=0 #t1=0	1	0	Zero 4×4 block
Luma # c & tr.1s(1,0) vlc=0 #c=1 #t1=1	1	1	Coefficient token
Luma trailing ones sign (1,0)	1	1	Trailing ones
Luma totalrun (1,0) vlc=0 totzeros= 1	11	1	Total zeros
Luma # c & tr.1s(0,1) vlc=0 #c=2 #t1=2	1	2	Coefficient token
Luma trailing ones sign (0,1)	10	2	Trailing ones
Luma totalrun (0,1) vlc=1 totzeros= 3	100	3	Total zeros
Luma run (0,1) k=1 vlc=2 run= 1	10	1	Zero run
Luma # c & tr.1s(1,1) vlc=1 #c=0 #t1=0	11	0	Zero 4×4 block

partition is $(-1/4, 0)$. Only the first, top-left 8×8 luma quadrant contains non-zero coefficients and the second and third 4×4 blocks within this quadrant contain non-zero coefficients.

4. P macroblock, multiple reference frames
   In this example (Table 5.18, Figure 5.21) the macroblock is coded using 8×8 partitions. A further syntax element sub_mb_type is sent for each of the four 8×8 luma partitions to indicate the sub-macroblock partition types. Next, the reference frame for each macroblock partition is signalled. Note that all sub-macroblock partitions in a partition share the same reference frame. Partition 0 uses reference index 2, three frames in the past, partition 1 uses reference index 1, two frames in the past, and partitions 2 and 3 use reference index 0, one frame in the past, the default case. MVDx and MVDy are sent for each partition or sub-macroblock partition; the only non-zero values are (−3, 0) and (−3, 1). Note that these are signalled as (−12,0) and (−12, 4) respectively, in units of $1/4$ sample. CBP indicates that the only non-zero coefficients occur in the Chroma DC blocks.

5. B macroblock
   This B macroblock is coded using two 8×16 partitions (Table 5.19, Figure 5.22). Partition 0 is predicted from the first picture in List 1, the next frame in display order, and partition 1 is bi-predicted from the first pictures in List 0 and List 1, the previous

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000155

Table 5.18 P macroblock, example 4

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No preceding skipped MBs
mb_type (P_SLICE) (5, 2) = 8	100	3	8×8 partitions with multiple reference frames
sub_mb_type(0)	1	0	Partition 0 mode = 8×8
sub_mb_type(1)	1	0	Partition 1 mode = 8×8
sub_mb_type(2)	10	1	Partition 2 mode = 8×4
sub_mb_type(3)	10	1	Partition 3 mode = 8×4
ref_idx_l0 = 2	11	2	Part 0 L0 ref = 2 (3 prev)
ref_idx_l0 = 1	10	1	Part 1 L0 ref = 1 (2 prev)
ref_idx_l0 = 0	1	0	Part 2 L0 ref = 0 (prev)
ref_idx_l0 = 0	1	0	Part 3 L0 ref = 0 (prev)
mvd_l0 (0) = 0	1	0	Part 0 MVDx
mvd_l0 (1) = 0	1	0	Part 0 MVDy
mvd_l0 (0) = 0	1	0	Part 1 MVDx
mvd_l0 (1) = 0	1	0	Part 1 MVDy
mvd_l0 (0) = −12	11001	−12	Part 2 sub 0 MVDx
mvd_l0 (1) = 0	1	0	Part 2 sub 0 MVDy
mvd_l0 (0) = 0	1	0	Part 2 sub 1 MVDx
mvd_l0 (1) = 0	1	0	Part 2 sub 1 MVDy
mvd_l0 (0) = 0	1	0	Part 3 sub 0 MVDx
mvd_l0 (1) = 0	1	0	Part 3 sub 0 MVDy
mvd_l0 (0) = −12	11001	−12	Part 3 sub 1 MVDx
mvd_l0 (1) = 4	1000	4	Part 3 sub 1 MVDy
CBP (5, 2) = 16	10	16	010000_2 : Only Chroma DC present
Delta QP (5, 2) = 0	1	0	No change in QP
ChrDC # c & tr.1s(0,0) vlc=0 #c=1 #t1=1	1	1	Cb DC coefficients
ChrDC trailing ones sign (0,0)	0	0	Just a single trailing one
ChrDC totalrun (0,0) vlc=0 totzeros= 2	1	2	Zero run
ChrDC # c & tr.1s(1,1) vlc=0 #c=0 #t1=0	1	0	No Cr DC coefficients

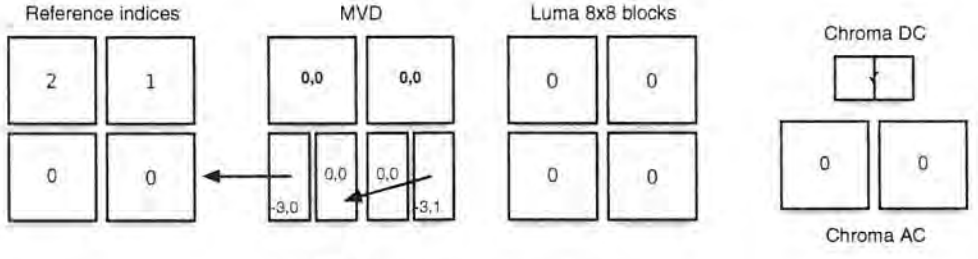


Figure 5.21 P macroblock example 4

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000156

Table 5.19 B macroblock, example 5

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No preceding skipped MBs
mb_type (B_SLICE) (5, 4) = 3	10000	15	B_L1_Bi_8 × 16 ; partition 0 from L1, partition 1 is bipred
ref_idx_l0 = 0	1	0	Part 1 L0 ref, order: all L0s then all L1s in partition order
ref_idx_l1 = 0	1	0	Part 0 L1 ref
ref_idx_l1 = 0	1	0	Part 1 L1 ref
mvd_l0 (0) = 11	10110	11	Part 1 L0 MVDx
mvd_l0 (1) = 2	100	2	Part 1 L0 MVDy
mvd_l1 (0) = −3	111	−3	Part 0 L1 MVDx
mvd_l1 (1) = 0	1	0	Part 0 L1 MVDy
mvd_l1 (0) = 1	10	1	Part 1 L1 MVDx
mvd_l1 (1) = −2	101	−2	Part 1 L1 MVDy
CBP (5, 4) = 0	1	0	000000_2: No coefficient data

+ next frames in display order. Partition 0 has MVD = ($-^3/_4$, 0) which, when added to the predicted vector, indicates an offset of ($7^3/_4$, $-1^1/_4$) from the reference frame, the next frame in display order. Partition 1 has two MVD pairs, ($2^3/_4$,$^1/_2$) with respect to the previous frame and ($^1/_4$,$-^1/_2$) with respect to the next frame in display order. The final motion vectors and reference regions are shown in Figure 5.22. There are no coded coefficients; this is quite common for B macroblocks, especially at lower bitrates, because inter prediction in a B macroblock tends to be very efficient, leading to minimal or zero energy in the quantized residual.

6. B macroblock
   Another B macroblock example is shown in Table 5.20. Similarly to the previous example, there are two 8 × 16 partitions. This time, the first is predicted from L0, the previous frame, and the second is bipredicted using L0, the previous frame, and L1, the next frame. Compare the order of the reference indices and motion vector differences with the previous example.

7. B macroblock
   This macroblock (Table 5.21) has a single 16 × 16 partition, bipredicted from the previous (L0) and next (L1) reference frames. One MVD pair (MVDx, MVDy) is sent for each prediction reference.

8. B macroblock
   This example is a macroblock coded using four 8 × 8 macroblock partitions (Table 5.22). Partitions 0 and 2 are predicted using Direct Prediction (Chapter 6) and so no MVD or reference index is sent for these partitions. Partition 1 is a single 8 × 8 sub-macroblock partition, predicted from L1, the next reference frame. Partition 3 contains four 4 × 4 sub-macroblock partitions, all predicted from L1. Each of these 4 × 4 sub MB partitions has a separate MVDx, MVDy pair.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000157

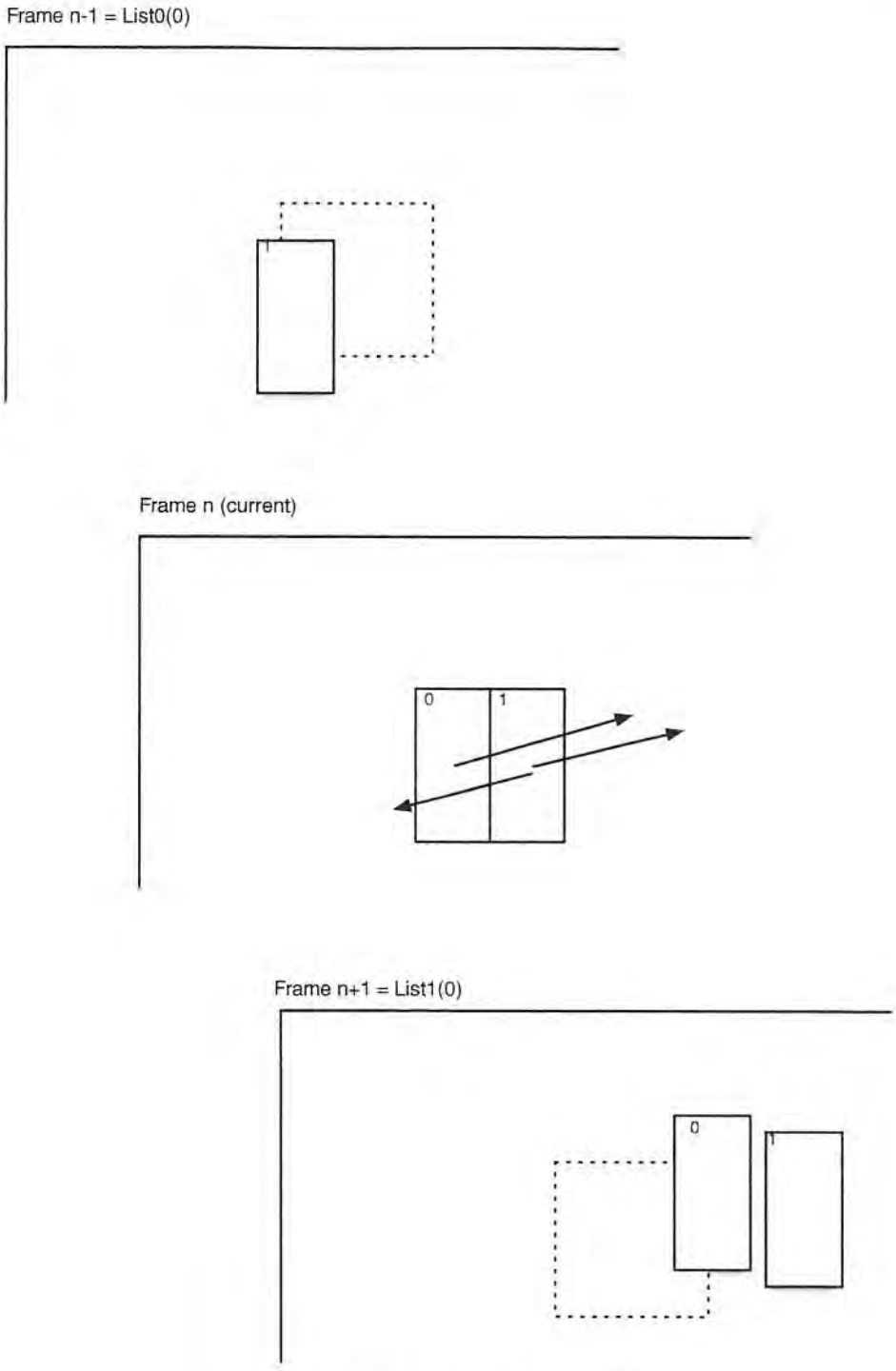


Figure 5.22 B macroblock, example 5

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000158

Table 5.20 B macroblock, example 6

Parameter	Binary code	Symbol	Discussion
mb_skip_run	1	0	No previous skip
mb_type (B_SLICE) (7, 2) = 3	1110	13	B_L0_Bi_8 × 16 : part 0 from L0, part 1 is bipred
ref_idx_l0 = 0	1	0	Part 0, L0 reference
ref_idx_l0 = 0	1	0	Part 1, L0 reference
ref_idx_l1 = 0	1	0	Part 1, L1 reference
mvd_l0 (0) = −22	101101	−22	Part 0, L0 MVDx
mvd_l0 (1) = 4	1000	4	Part 0, L0 MVDy
mvd_l0 (0) = 10	10100	10	Part 1, L0 MVDx
mvd_l0 (1) = 0	1	0	Part 1, L0 MVDy
mvd_l1 (0) = 11	10110	11	Part 1, L1 MVDx
mvd_l1 (1) = −1	11	−1	Part 1, L1 MVDy
CBP (7, 2) = 0	1	0	000000_2 : No coefficient data

5.8 Summary

At the heart of the H.264/AVC standard is the syntax, a specific format for representing a coded video sequence, together with ancillary information. A decoder parses the syntax, extracts the parameters and data elements and can then proceed to decode and display video. The syntax is organized hierarchically, from a complete video sequence at the highest level, down to coded macroblocks and blocks. A key feature of H.264/AVC is its highly flexible handling of pictures and predictions. Picture management procedures make it possible to construct and use a very large variety of prediction structures for intra and inter prediction. This flexibility makes it possible to accurately predict or estimate the current section of an image, which in turn leads to high compression efficiency. In the following chapters we will examine the prediction of macroblocks, transform and quantization of residuals and coding of syntax elements.

Table 5.21 B macroblock, example 7

Parameter	Binary code	Symbol	Discussion
mb_skip_run	110	5	5 previous MBs are skipped.
mb_type (B_SLICE) (7, 2) = 1	100	3	B_Bi_16 × 16 : one partition, bipredicted
ref_idx_l0 = 0	1	0	L0 reference
ref_idx_l1 = 0	1	0	L1 reference
mvd_l0 (0) = −1	11	−1	L0 MVDx
mvd_l0 (1) = 0	1	0	L0 MVDy
mvd_l1 (0) = −1	11	−1	L1 MVDx
mvd_l1 (1) = −1	11	−1	L1 MVDy
CBP (7, 2) = 0	1	0	No coefficient data

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000159

Table 5.22 B macroblock, example 8

Parameter	Binary code	Symbol	Discussion
mb_skip_run	10	1	1 previous MB is skipped
mb_type (B_SLICE) (6, 6) = 8	10111	22	B_8 × 8
sub_mb_pred(0)	1	0	Direct, no MVD or ref_ix
sub_mb_pred(1)	11	2	B_L1_8 × 8, 8 × 8 predicted from L1
sub_mb_pred(2)	1	0	Direct, no MVD or ref_ix
sub_mb_pred(3)	1100	11	B_L1_4 × 4, 4 × 4 predicted from L1
ref_idx_l1 = 0	1	0	L1 reference for 8 × 8 block 1
ref_idx_l1 = 0	1	0	L1 reference for 8 × 8 block 3
mvd_l1 (0) = −6	1101	−6	Block 1 MVDx
mvd_l1 (1) = 1	10	1	Block 1 MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 0, MVDx
mvd_l1 (1) = 0	1	0	Block 3, subblock 0, MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 1, MVDx
mvd_l1 (1) = 0	1	0	Block 3, subblock 1, MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 2, MVDx
mvd_l1 (1) = −3	111	−3	Block 3, subblock 2, MVDy
mvd_l1 (0) = 0	1	0	Block 3, subblock 3, MVDx
mvd_l1 (1) = 0	1	0	Block 3, subblock 3, MVDy
CBP (6, 6) = 22	101101	22	010110_2 : Luma coefficients in 8 × 8 blocks 1 and 2. Chroma DC coefficients present, no chroma AC coefficients.
Delta QP (6, 6) = 0	1	0	No change in QP.
Luma # c & tr.1s(2,0) vlc=0 #c=0 #t1=0 ....further data for luma 8 × 8 blocks 1 and 2 and chroma DC coefficients.	1	0	First 4 × 4 block of luma 8 × 8 block 1...

5.9 References

i. JM reference software version 16.0, http://iphome.hhi.de/suehring/tml/, July 2009.

ii. T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, 'Overview of the H.264/AVC video coding standard', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, (4 August 2003), pp. 560–576.

iii. L. Wang, R. Gandhi, K. Panusopone, Y. Yu and A. Luthra, 'MB-Level Adaptive Frame/Field Coding for JVT', Joint Video Team Document JVT-B106, January 2002.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000160

6

H.264 Prediction

6.1 Introduction

Perhaps the most important reason for the widespread adoption of H.264/AVC is its compression performance. An H.264 codec, if properly designed and used, can out-perform many other video codecs in terms of compression ratio for a given image quality. Much of the performance gain compared with previous standards is due to H.264/AVC's efficient prediction methods. For every macroblock, a prediction is created, an attempt to duplicate the information contained in the macroblock using previously coded data, and subtracted from the macroblock to form a residual. The efficiency or accuracy of this prediction process has a significant impact on compression performance. An accurate prediction means that the residual contains very little data and this in turn leads to good compression performance.

H.264/AVC supports a wide range of prediction options – intra prediction using data within the current frame, inter prediction using motion compensated prediction from previously coded frames, multiple prediction block sizes, multiple reference frames and special modes such as Direct and Weighted prediction. Together with sub-pixel interpolation and a built-in filter to reduce compression artefacts, these features give an H.264 encoder a great deal of flexibility in the prediction process. By selecting the best prediction options for an individual macroblock, an encoder can minimize the residual size to produce a highly compressed bitstream.

We start with an overview of macroblock prediction, followed by a detailed look at intra and inter prediction in H.264/AVC.

6.2 Macroblock prediction

Figure 6.1 shows the prediction sources for three macroblocks, an I Macroblock, a P Macroblock and a B Macroblock. An I Macroblock (I MB) is predicted using intra prediction from neighbouring samples in the current frame. A P Macroblock (P MB) is predicted from samples in a previously-coded frame which may be before or after the current picture in display order, i.e. a 'past' or a 'future' frame. Different rectangular sections (**partitions**) in a P MB may

The H.264 Advanced Video Compression Standard, Second Edition Iain E. Richardson
© 2010 John Wiley & Sons, Ltd

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000162

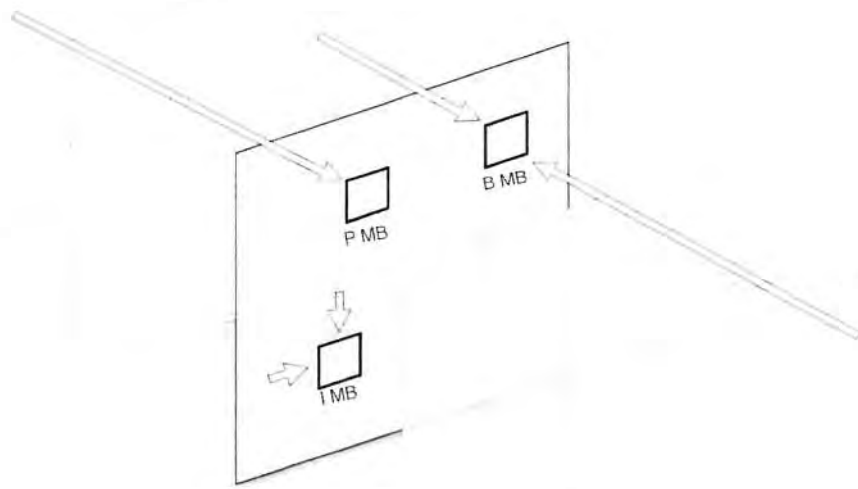


Figure 6.1 Example of macroblock types and prediction sources

be predicted from different reference frames. Each partition in a B Macroblock (B MB) is predicted from samples in one or two previously-coded frames, for example, one 'past' and one 'future' as shown in the figure.

6.3 Intra prediction

An intra (I) macroblock is coded without referring to any data outside the current slice. I macroblocks may occur in any slice type. Every macroblock in an I slice is an I macroblock. I macroblocks are coded using intra prediction, i.e. prediction from previously-coded data in the same slice. For a typical block of luma or chroma samples, there is a relatively high correlation between samples in the block and samples that are immediately adjacent to the block. Intra prediction therefore uses samples from adjacent, previously coded blocks to predict the values in the current block.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000163

Example

Figure 6.2 shows 4 × 4 blocks in the luma component of a macroblock. The current block is number 10 in the figure. Blocks 0-9 have already been coded and transmitted and are therefore available to the decoder by the time block 10 is decoded. This means that any of the samples in blocks 0 to 9 are potential candidates for intra prediction. However, an H.264 encoder may only choose certain samples in blocks **1, 3, 5** and **8** to generate a 4 × 4 intra prediction.

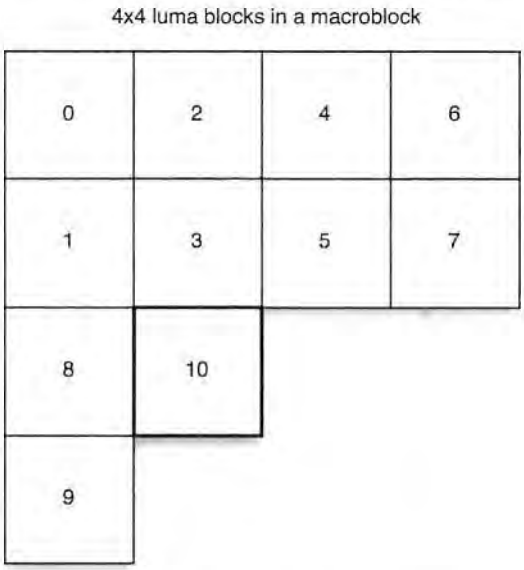


Figure 6.2 Intra prediction: adjacent blocks example

In an intra macroblock, there are three choices of intra prediction block size for the luma component, namely 16 × 16, 8 × 8 or 4 × 4. A single prediction block is generated for each chroma component. Each prediction block is generated using one of a number of possible prediction modes (Table 6.1).

Table 6.1 Intra prediction types

Intra prediction block size	Notes
16 × 16 (luma)	A single 16 × 16 prediction block P is generated. Four possible prediction modes.
8 × 8 (luma)	An 8 × 8 prediction block P is generated for each 8 × 8 luma block. Nine possible prediction modes. 'High' Profiles only.
4 × 4 (luma)	A 4 × 4 prediction block P is generated for each 4 × 4 luma block. Nine possible prediction modes.
Chroma	One prediction block P is generated for each chroma component. Four possible prediction modes. The same prediction mode is used for both chroma components.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000164

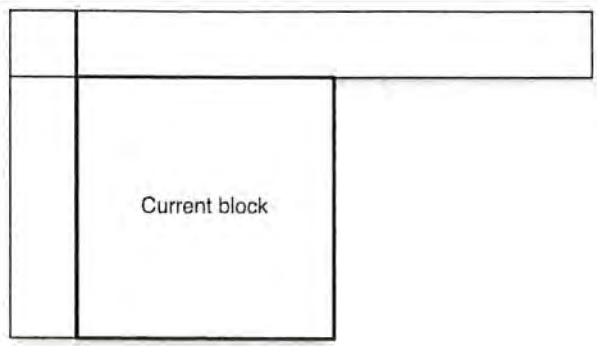


Figure 6.3 Intra prediction source samples, 4 × 4 or 8 × 8 luma blocks

When 4 × 4 or 8 × 8 block size is chosen for the luma component, the intra prediction is created from samples directly above the current block; directly to the left; above and to the left; above and to the right; or a combination of these depending on the current prediction mode and on whether the required samples have already been coded (Figure 6.3). For a 16 × 16 luma block or a chroma block, the prediction is created from samples directly to the left or above the current block or from a combination of these (Figure 6.4).

Only those samples that are actually available may be used to form a prediction. For example, a block on the left margin of the picture or slice has no neighbouring samples on the left side and so certain intra prediction modes are not available to the encoder. The encoder chooses an intra mode for the current block from the available prediction modes. The choice of intra mode is communicated to the decoder as part of the coded macroblock.

The choice of intra prediction block size for the luma component, 16 × 16, 4 × 4 or 8 × 8 when available, tends to be a trade-off between (i) prediction efficiency and (ii) cost of signalling the prediction mode.

(a) Smaller blocks: A smaller prediction block size (4 × 4) tends to give a more accurate prediction, i.e. the intra prediction for each block is a good match to the actual data in

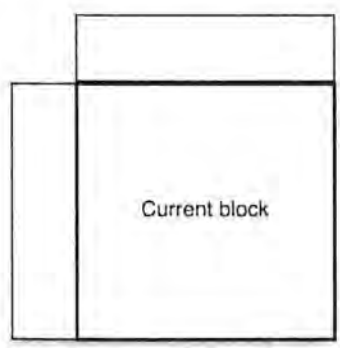


Figure 6.4 Intra prediction source samples, chroma or 16 × 16 luma blocks

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000165

the block. This in turn means a smaller coded residual, so that fewer bits are required to code the quantized transform coefficients for the residual blocks. However, the choice of prediction for every 4 × 4 block must be signalled to the decoder, which means that more bits tend to be required to code the prediction choices.

(b) Larger blocks: A larger prediction block size (16 × 16) tends to give a less accurate prediction, hence more residual data, but fewer bits are required to code the prediction choice itself.

An encoder will typically choose the appropriate intra prediction mode to minimize the total number of bits in the prediction and the residual.

Example 1:

Figure 6.5 shows a typical I picture from a CIF sequence coded using the Baseline profile, with the prediction block sizes superimposed. In homogeneous regions of the frame, where the texture is largely uniform, 16 × 16 prediction mode tends to be more efficient since the prediction is reasonably accurate and the prediction mode overhead is low. In more complex regions of the frame, 4 × 4 mode is often selected because the increased rate required to signal the prediction mode is offset by the reduced residual size.

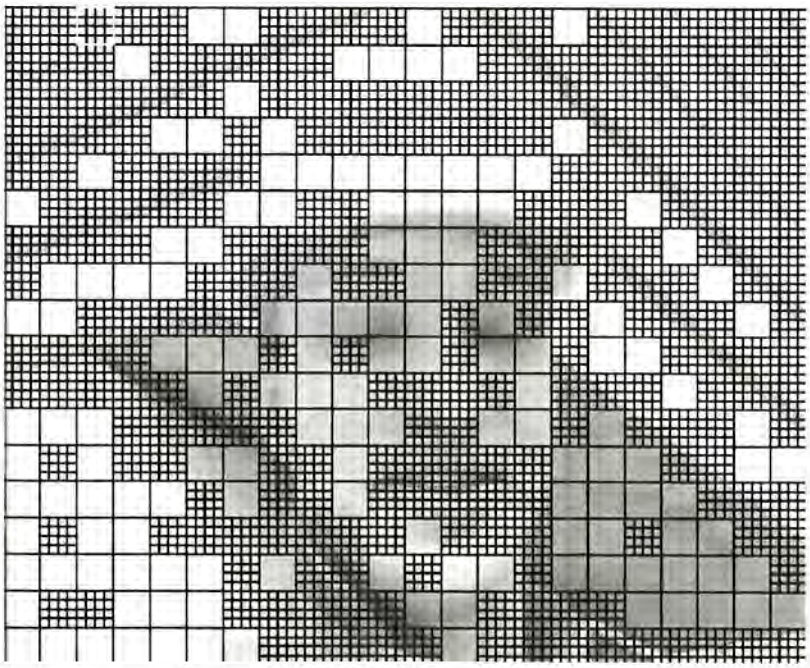


Figure 6.5 Example of intra block size choices. CIF, Baseline Profile. Reproduced by permission of Elecard.

Example 2:

A QCIF video frame (Figure 6.6) is encoded in intra mode and each block or macroblock is predicted from neighbouring, previously-encoded samples. Figure 6.7 shows the predicted

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000166



Figure 6.6 QCIF frame with highlighted macroblock



Figure 6.7 Predicted luma frame formed using H.264 intra prediction

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000167

luma frame P formed by choosing the best 4×4 or 16×16 prediction mode for each region. The predicted macroblocks (Figure 6.7) are not an accurate match for the original macroblocks (Figure 6.6). However, the predicted frame provides a rough approximation of the original. When the prediction is subtracted from the original, the residual (Figure 6.8) contains less information than the original frame and is therefore 'easier' to compress.

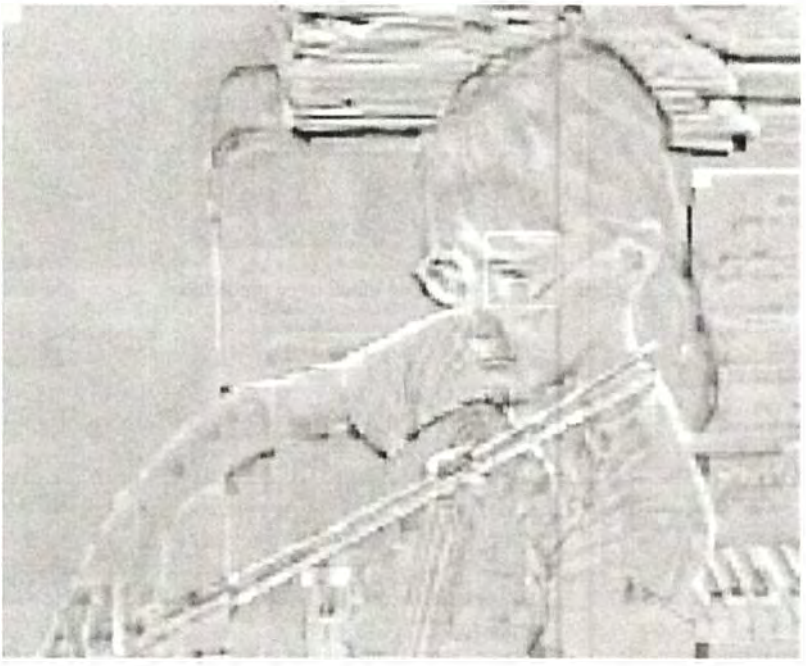


Figure 6.8 Residual after subtracting intra prediction

6.3.1 4 × 4 luma prediction modes

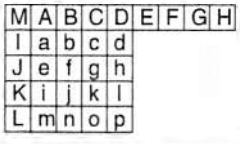
Figure 6.9 shows a 4×4 luma block, part of the highlighted macroblock in Figure 6.6, that is required to be predicted. The samples above and to the left, labelled A-M in Figure 6.10, have previously been encoded and reconstructed and are therefore available in the encoder and decoder to form a prediction reference. The samples a, b, c, ..., p of the prediction block P (Figure 6.10) are calculated based on the samples A-M as follows. The method of forming a DC prediction, mode 2, is modified depending on which samples A-M are available and each of the other modes may only be used if all of the required prediction samples are available. However, note that if samples E, F, G and H are not available, the value of sample D is copied to these positions and they are marked as 'available'.

The arrows in Figure 6.11 indicate the direction of prediction in each mode. For modes 3-8, the predicted samples are formed from a weighted average of the prediction samples A-M. For example, if mode 4 is selected, the top-right sample of P, labelled 'd' in Figure 6.10, is predicted by: d = round(B/4 + C/2 + D/4).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000168



Figure 6.9 4 × 4 luma block to be predicted



Mode 0 (Vertical)	The upper samples A,B,C,D are extrapolated vertically.
Mode 1 (Horizontal)	The left samples I,J,K,L are extrapolated horizontally.
Mode 2 (DC)	All samples in P are predicted by the mean of samples A..D and I..L.
Mode 3 (Diagonal Down-Left)	The samples are interpolated at a 45° angle between lower-left and upper-right.
Mode 4 (Diagonal Down-Right)	The samples are extrapolated at a 45° angle down and to the right.
Mode 5 (Vertical-Left)	Extrapolation at an angle of approximately 26.6° to the left of vertical, i.e. width/height = $^1/_2$.
Mode 6 (Horizontal-Down)	Extrapolation at an angle of approximately 26.6° below horizontal.
Mode 7 (Vertical-Right)	Extrapolation or interpolation at an angle of approximately 26.6° to the right of vertical.
Mode 8 (Horizontal-Up)	Interpolation at an angle of approximately 26.6° above horizontal.

Figure 6.10 Labelling of prediction samples. 4 × 4 prediction

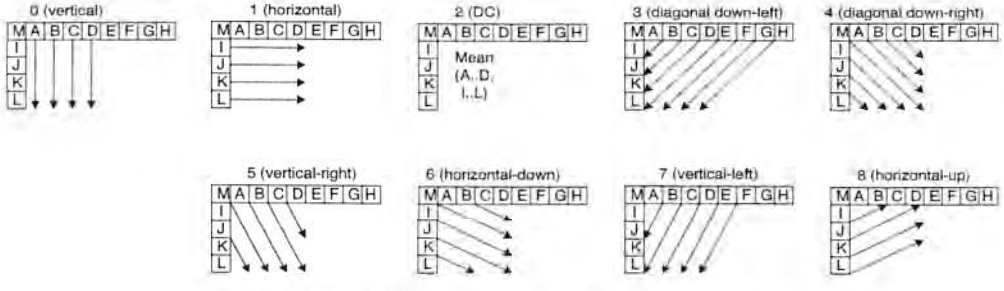


Figure 6.11 4 × 4 intra prediction modes

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000169

Example

The 9 prediction modes 0-8 are calculated for the 4 × 4 block shown in Figure 6.9 and the resulting prediction blocks P are shown in Figure 6.12. The Sum of Absolute Errors (SAE) for each prediction indicates the magnitude of the prediction error. In this case, the best match to

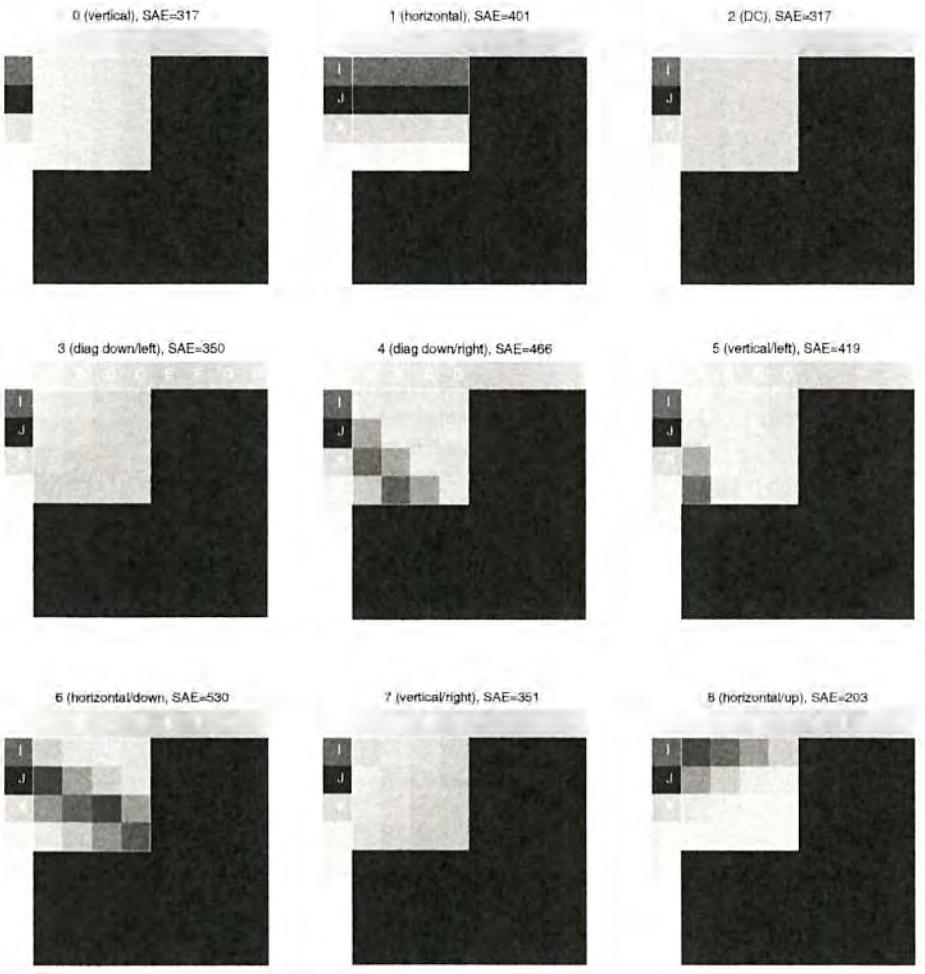


Figure 6.12 Prediction blocks, 4 × 4 modes 0–8

the actual current block is given by **mode 8**, horizontal-up, because this mode gives the smallest SAE. A visual comparison shows that the mode 8 P block appears quite similar to the original 4 × 4 block.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000170

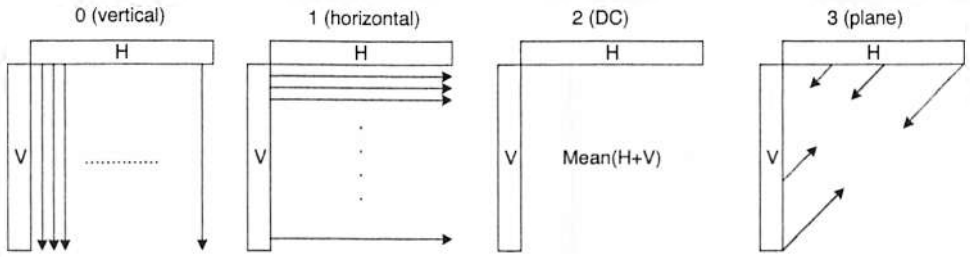


Figure 6.13 Intra 16 × 16 prediction modes

6.3.2 16 × 16 luma prediction modes

As an alternative to the 4 × 4 luma modes described above, the entire 16 × 16 luma component of a macroblock may be predicted in one operation. Four modes are available, shown in diagram form in Figure 6.13:

Mode 0 (vertical)	Extrapolation from upper samples (H)
Mode 1 (horizontal)	Extrapolation from left samples (V)
Mode 2 (DC)	Mean of upper and left-hand samples (H+V).
Mode 4 (Plane)	A linear 'plane' function is fitted to the upper and left-hand samples H and V. This works well in areas of smoothly-varying luminance.

Example

Figure 6.14 shows a luma macroblock with previously-encoded samples at the upper and left-hand edges. The results of the four prediction modes, shown in Figure 6.15, indicate that the best match is given by mode 3 which in this case produces a plane with a luminance gradient



Figure 6.14 16 × 16 macroblock

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1028, Page 000171