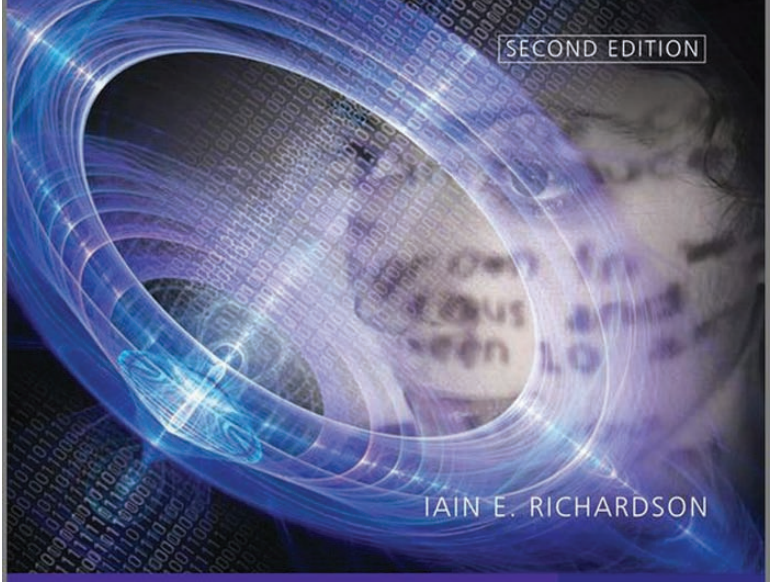
THE H.264 ADVANCED VIDEO COMPRESSION STANDARD

SECOND EDITION



IAIN E. RICHARDSON

WILEY

INTEL-1040
9,641,841

THE H.264 ADVANCED VIDEO COMPRESSION STANDARD

Second Edition

Iain E. Richardson
Vcodex Limited, UK

WILEY
A John Wiley and Sons, Ltd., Publication

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
Printed and bound in Great Britain by CPI Autony Rowe, Chippenham, Wiltshire

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

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

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

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

	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**

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

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)

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

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

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

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).

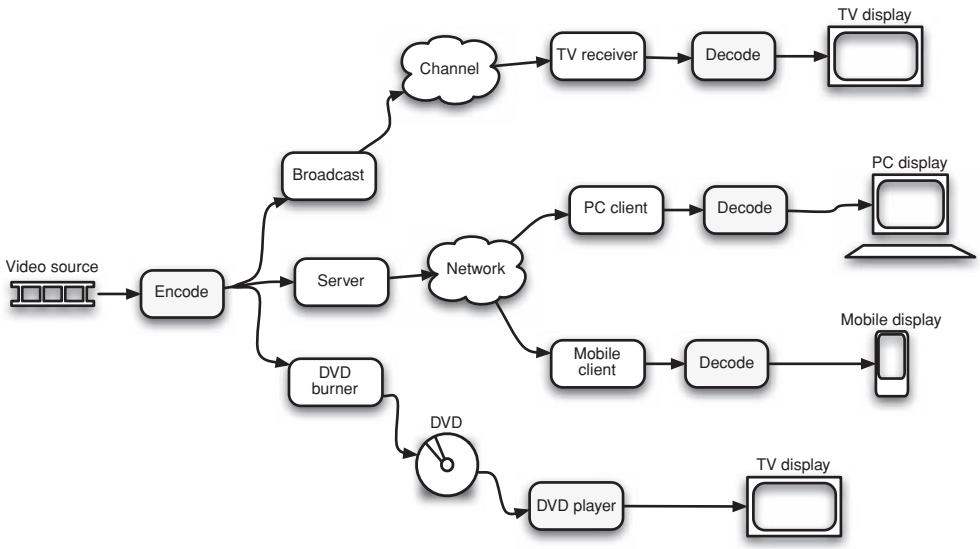


Figure 1.1 Video coding scenarios, one-way

- 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.

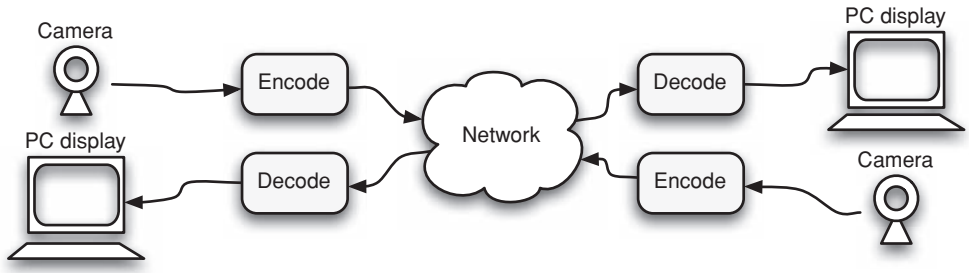


Figure 1.2 Video coding scenario, two-way

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:

'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

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

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

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

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

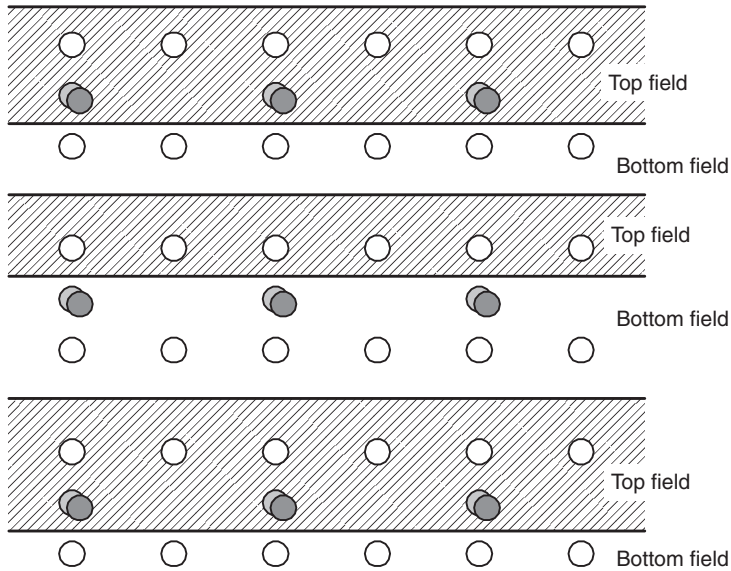


Figure 2.12 Allocation of 4:2:0 samples to top and bottom fields

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

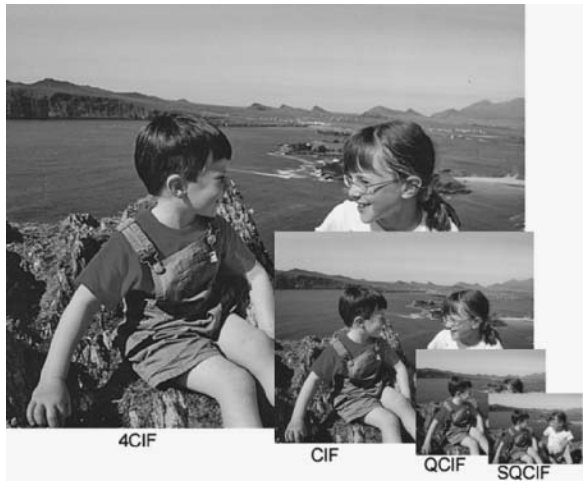


Figure 2.13 Video frame sampled at range of resolutions

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

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

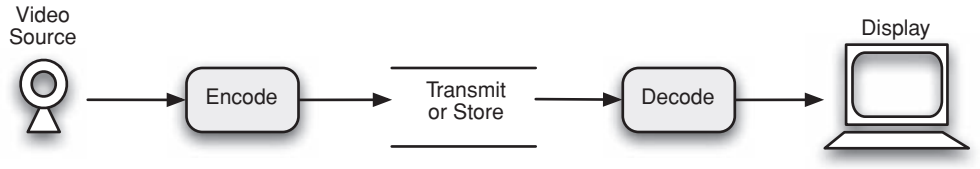


Figure 3.1 Encoder / Decoder

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

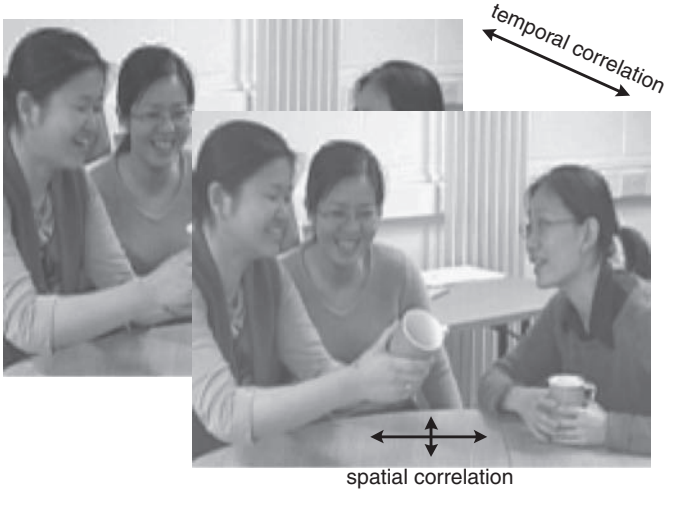


Figure 3.2 Spatial and temporal correlation in a video sequence

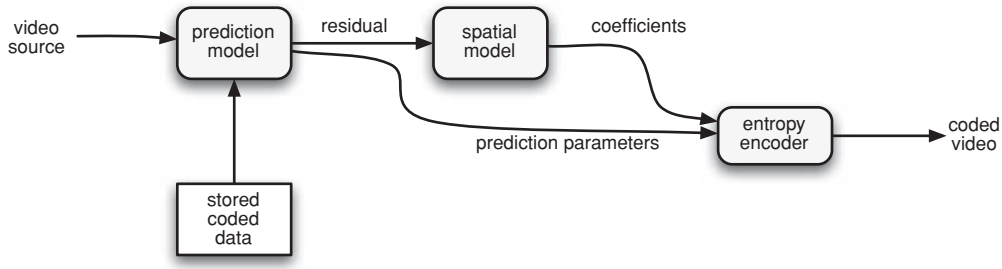


Figure 3.3 Video encoder block diagram

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

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

Figure 3.4 Frame 1

Figure 3.5 Frame 2

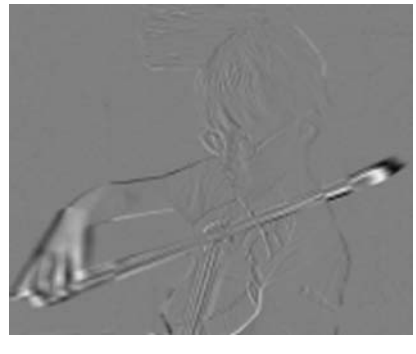


Figure 3.6 Difference

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,

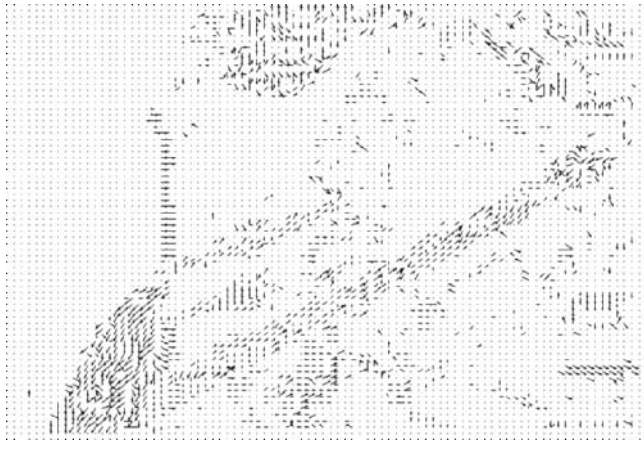


Figure 3.7 Optical flow

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

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

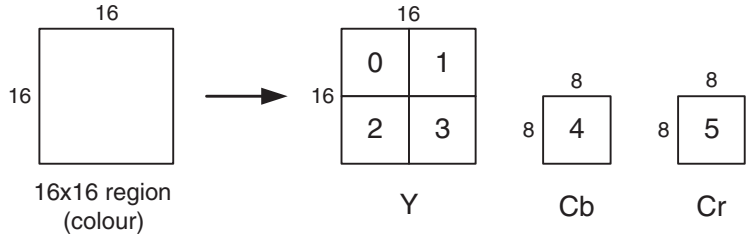


Figure 3.8 Macroblock (4:2:0)

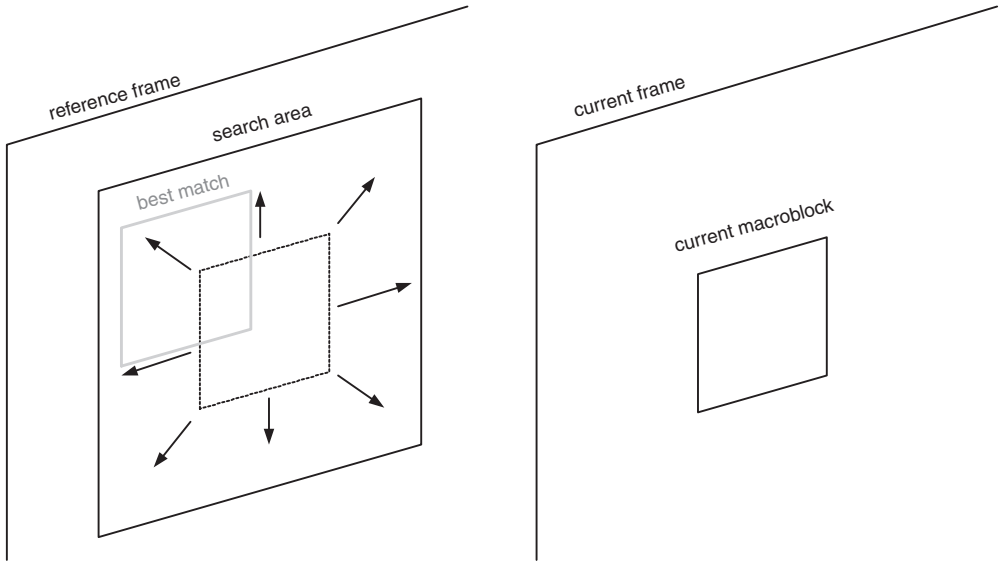


Figure 3.9 Motion estimation

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

Figure 3.10 Frame 1

Figure 3.11 Frame 2

Figure 3.12 Residual : no motion compensation

Figure 3.13 Residual : 16 × 16 block size

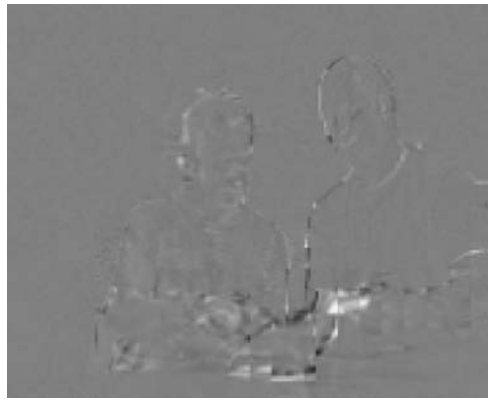


Figure 3.14 Residual : 8 × 8 block size

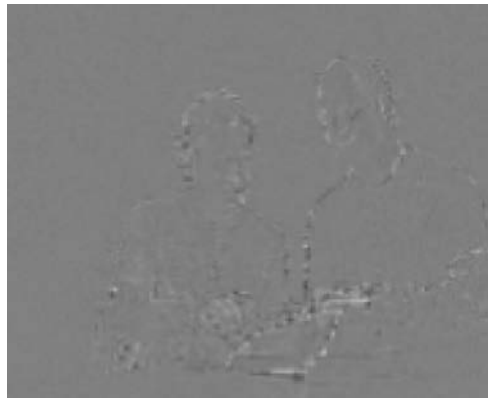


Figure 3.15 Residual : 4 × 4 block size

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

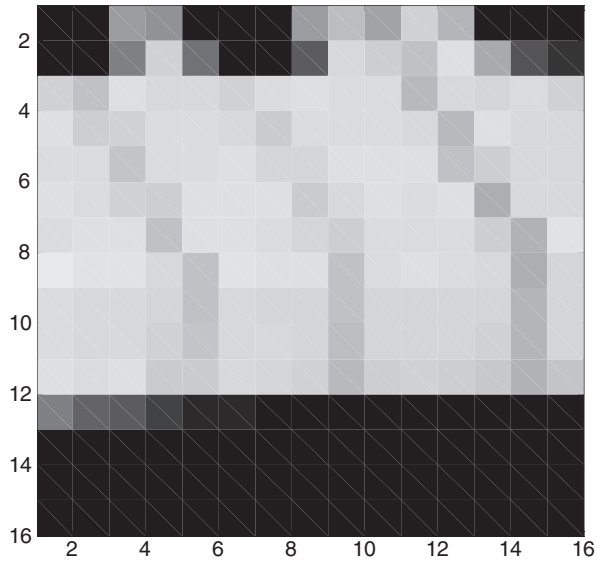


Figure 3.16 Close-up of reference region

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.

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:

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

and the inverse DCT (IDCT) by:

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

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:

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

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

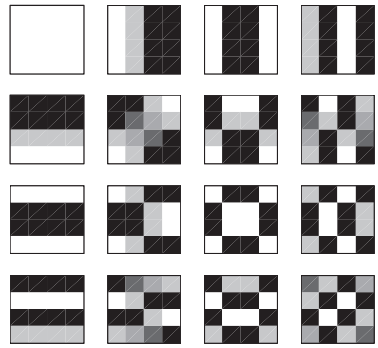


Figure 3.28 4 × 4 DCT basis patterns

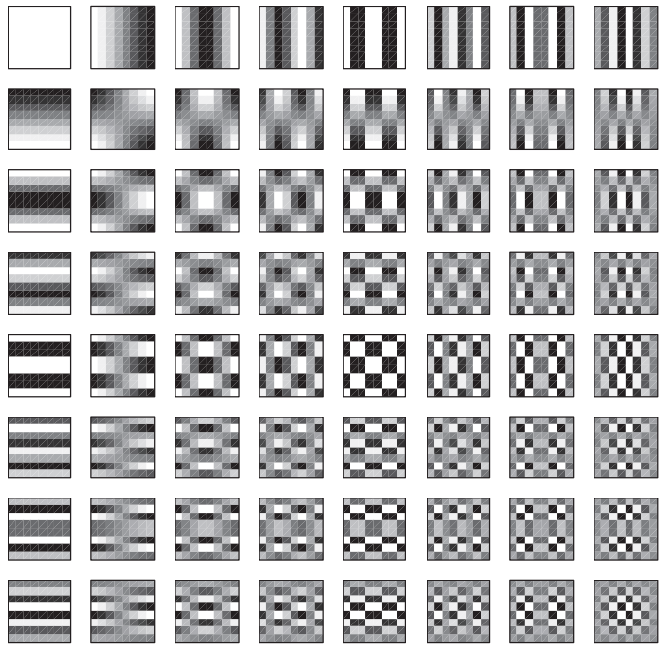


Figure 3.29 8 × 8 DCT basis patterns

Example 1 Calculating the DCT of a 4 × 4 block

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

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

The Forward DCT of **X** is given by: $\mathbf{Y} = \mathbf{AXA^T}$. 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, $\mathbf{Y} = \mathbf{Y}'\mathbf{A^T}$, 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;

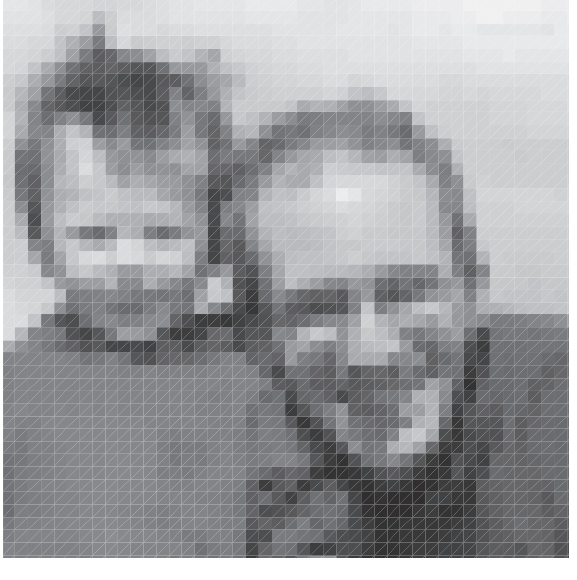


Figure 3.30 Image section showing 4 × 4 block

126	159	178	181
98	151	181	181
80	137	176	156
75	114	88	68

Original block

537.2	-76.0	-54.8	-7.8
-106.1	35.0	-12.7	-6.1
-42.7	46.5	10.3	-9.8
-20.2	12.9	3.9	-8.5

DCT coefficients

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

134	134	134	134
134	134	134	134
134	134	134	134
134	134	134	134

1 coefficient

169	169	169	169
149	149	149	149
120	120	120	120
100	100	100	100

2 coefficients

144	159	179	194
124	138	159	173
95	110	130	145
75	89	110	124

3 coefficients

146	179	187	165
95	146	179	175
66	117	150	146
76	109	117	96

5 coefficients

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

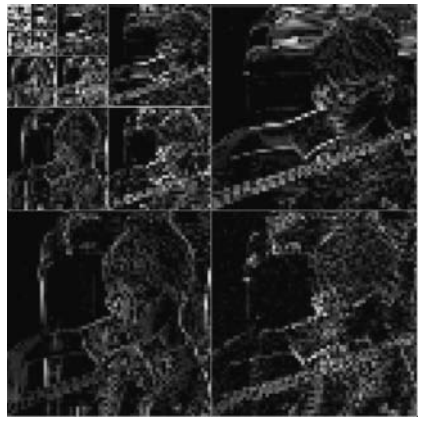


Figure 3.36 Five-stage wavelet decomposition of image

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:

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

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.

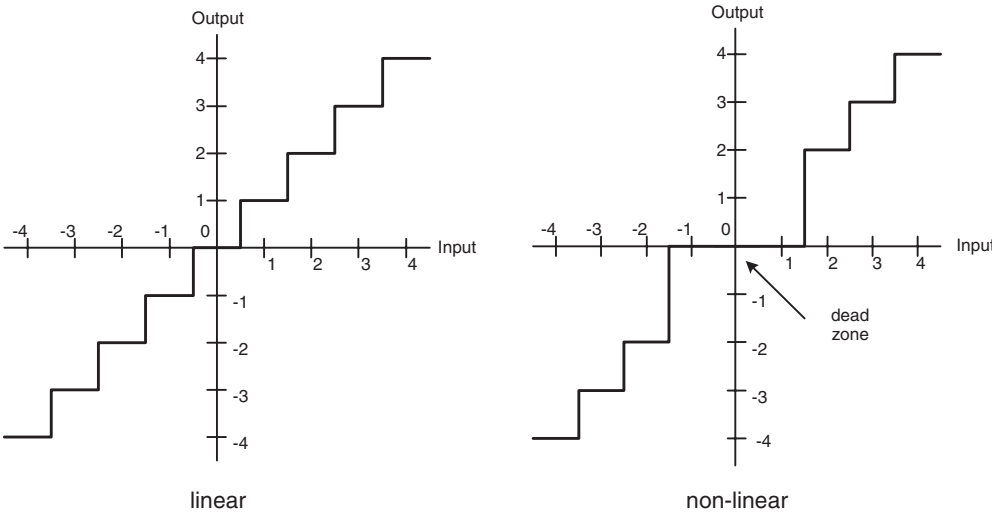


Figure 3.37 Scalar quantizers: linear; non-linear with dead zone

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.

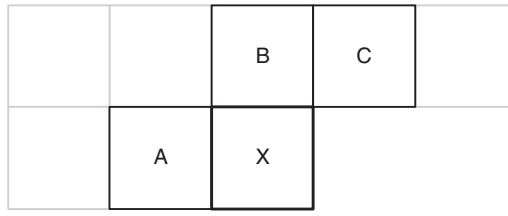


Figure 3.45 Motion vector prediction candidates

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.

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

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.

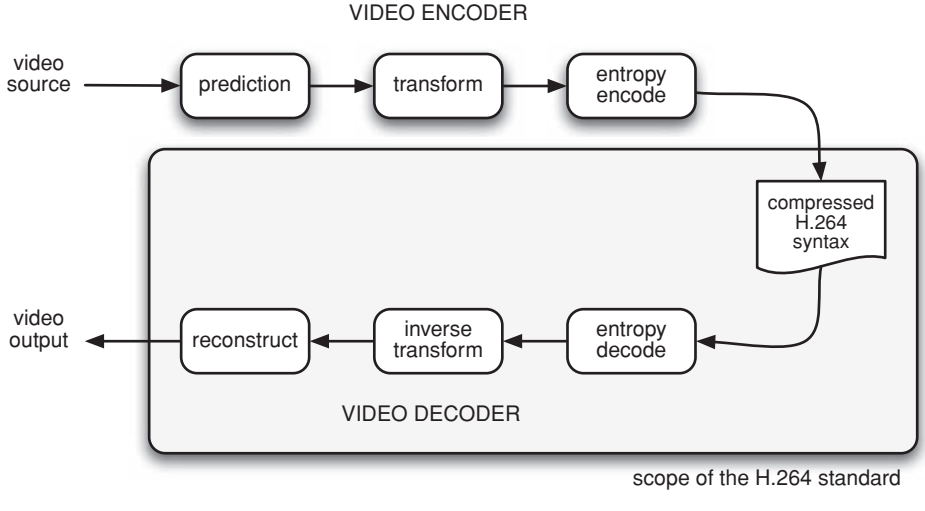


Figure 4.1 The H.264 video coding and decoding process

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

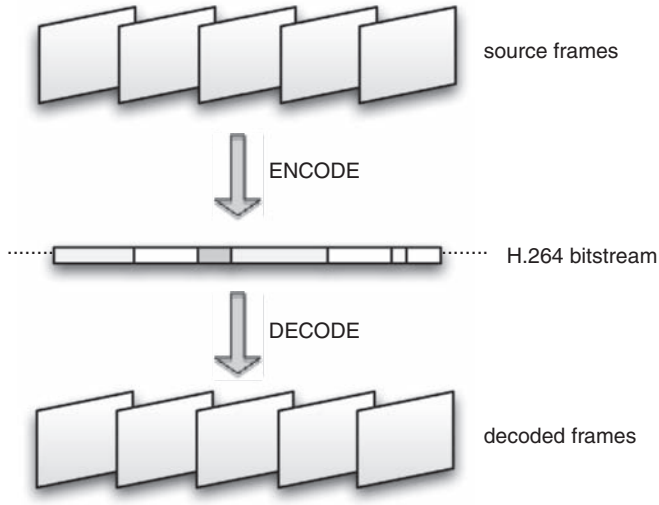


Figure 4.2 Video coding: source frames, encoded bitstream, decoded frames

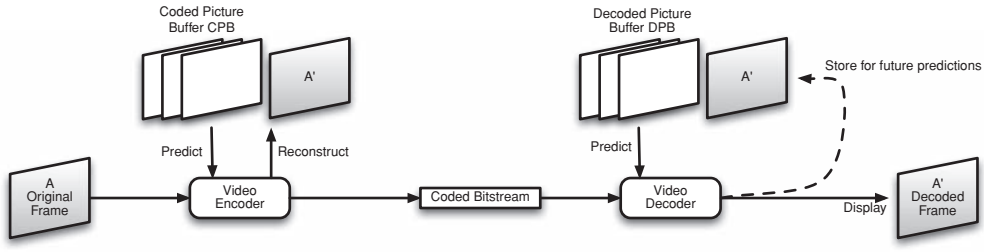


Figure 4.3 Video codec: high level view

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

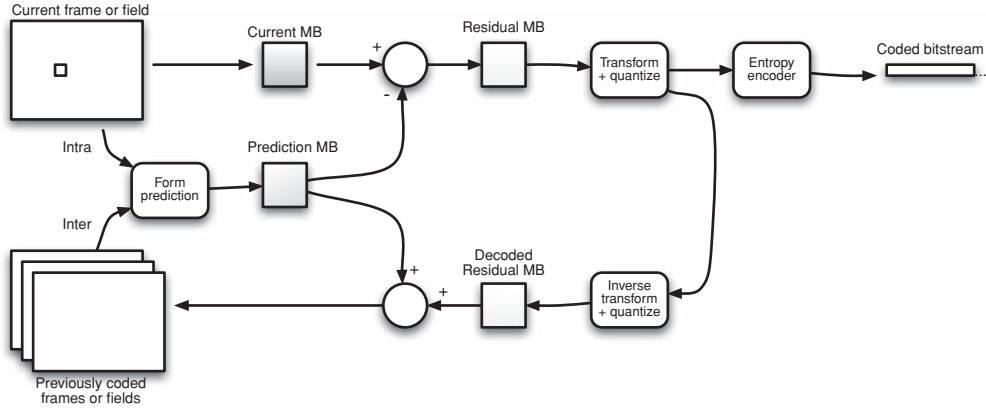


Figure 4.4 Typical H.264 encoder

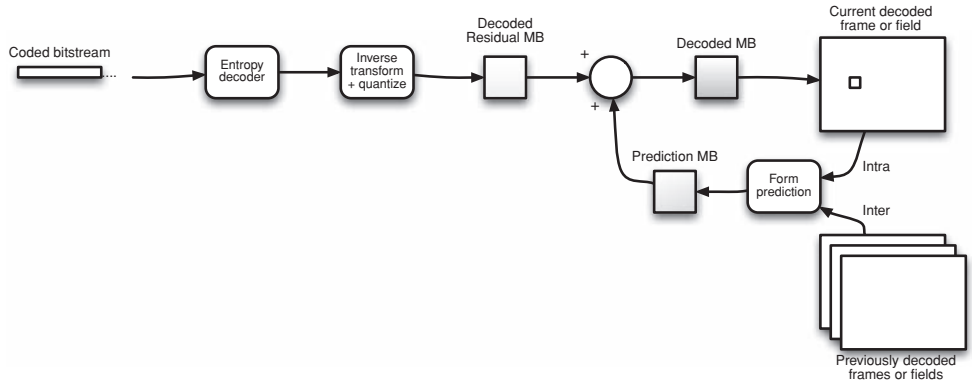


Figure 4.5 Typical H.264 decoder

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**.

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

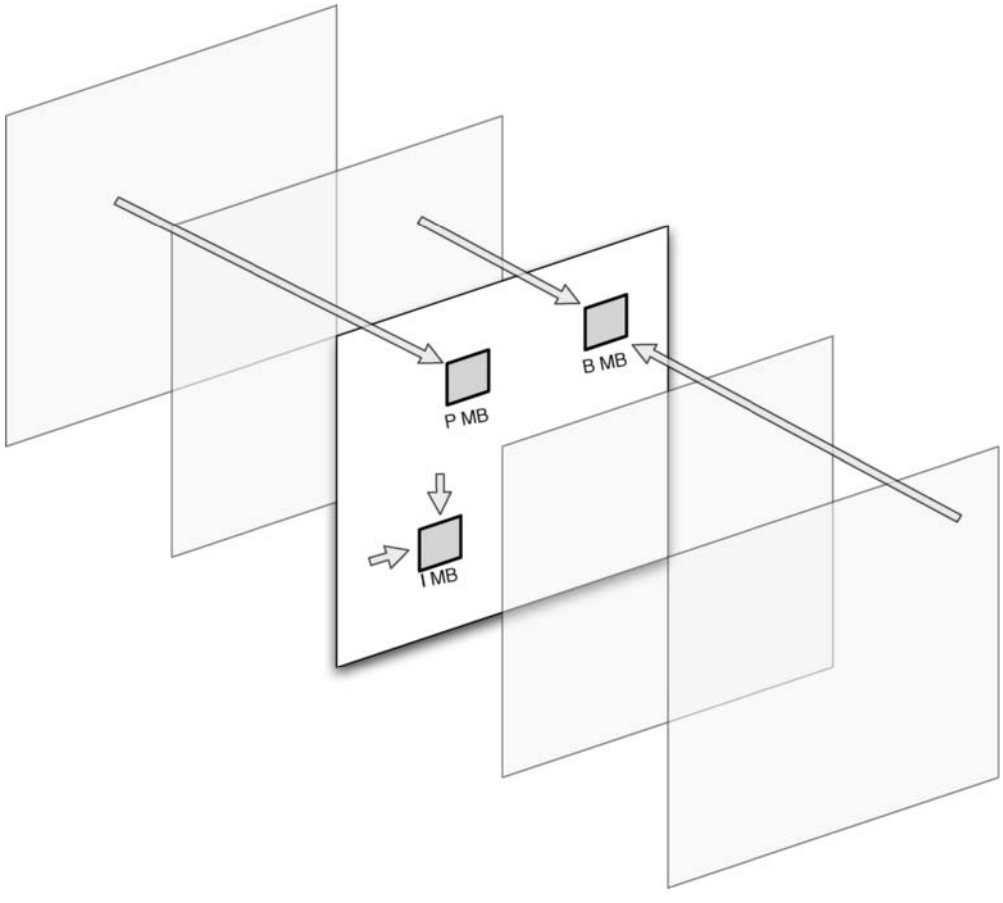


Figure 6.1 Example of macroblock types and prediction sources

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.

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.

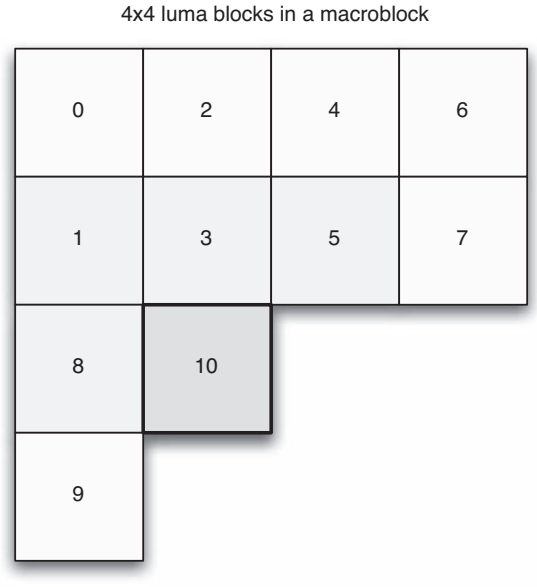


Figure 6.2 Intra prediction: adjacent blocks example

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.

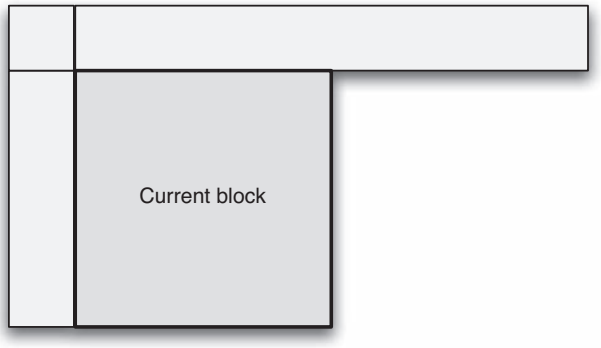


Figure 6.3 Intra prediction source samples, 4 × 4 or 8 × 8 luma blocks

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

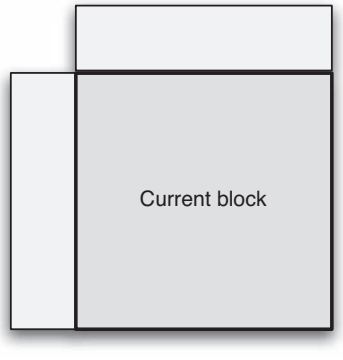


Figure 6.4 Intra prediction source samples, chroma or 16 × 16 luma blocks

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.

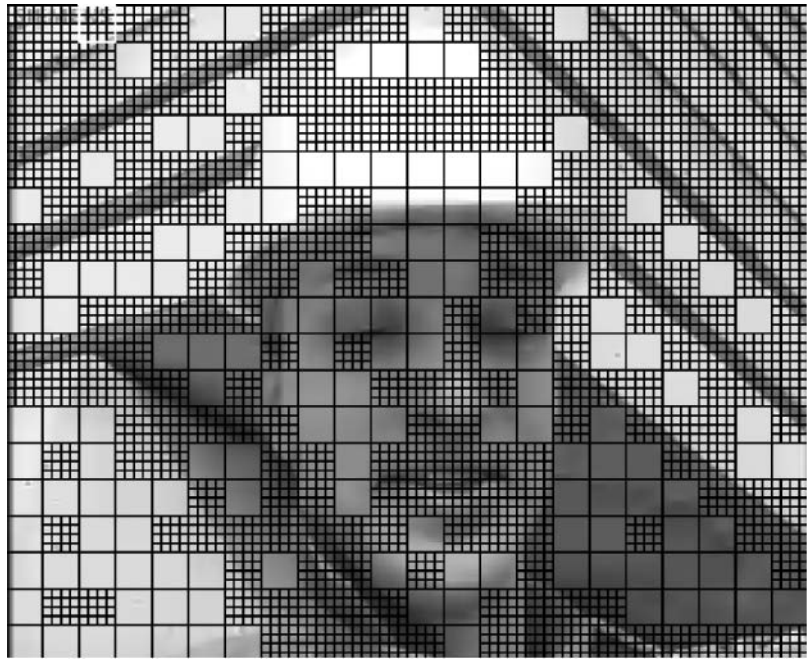


Figure 6.5 Example of intra block size choices, CIF, Baseline Profile. Reproduced by permission of Elecard.

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

Example

Blocks A and B were predicted using modes 3, diagonal down-left, and 1, horizontal, respectively. The most probable mode for block E is therefore 1, horizontal. *prev_intra4×4_pred_mode* is set to '0' and so *rem_intra4×4_pred_mode* is sent. Depending on the value of *rem_intra4×4_pred_mode*, one of the eight remaining prediction modes listed in Table 6.2 may be chosen.

Table 6.2 Choice of prediction mode, most probable mode = 1

rem_intra4×4_pred_mode	Prediction mode for block E
0	0
1	2
2	3
3	4
4	5
5	6
6	7
7	8

16 × 16 luma prediction or chroma prediction
The prediction mode is signalled as part of the macroblock syntax and predictive mode coding is not used.

6.4 Inter prediction

Inter prediction is the process of predicting a block of luma and chroma samples from a picture that has previously been coded and transmitted, a reference picture. This involves selecting a prediction region, generating a prediction block and subtracting this from the original block of samples to form a residual that is then coded and transmitted. The block of samples to be predicted, a macroblock partition or sub-macroblock partition, can range in size from a complete macroblock, i.e. 16 × 16 luma samples and corresponding chroma samples, down to a 4 × 4 block of luma samples and corresponding chroma samples.

The reference picture is chosen from a list of previously coded pictures, stored in a Decoded Picture Buffer, which may include pictures before and after the current picture in display order (Chapter 5). The offset between the position of the current partition and the prediction region in the reference picture is a motion vector. The motion vector may point to integer, half- or quarter-sample positions in the luma component of the reference picture. Half- or quarter-sample positions are generated by interpolating the samples of the reference picture. Each motion vector is differentially coded from the motion vectors of neighbouring blocks.

The prediction block may be generated from a single prediction region in a reference picture, for a P or B macroblock, or from two prediction regions in reference pictures, for a B macroblock [i]. Optionally, the prediction block may be weighted according to the temporal distance between the current and reference picture(s), known as weighted prediction. In a B macroblock, a block may be predicted in direct mode, in which case no residual samples or motion vectors are sent and the decoder infers the motion vector from previously received vectors.

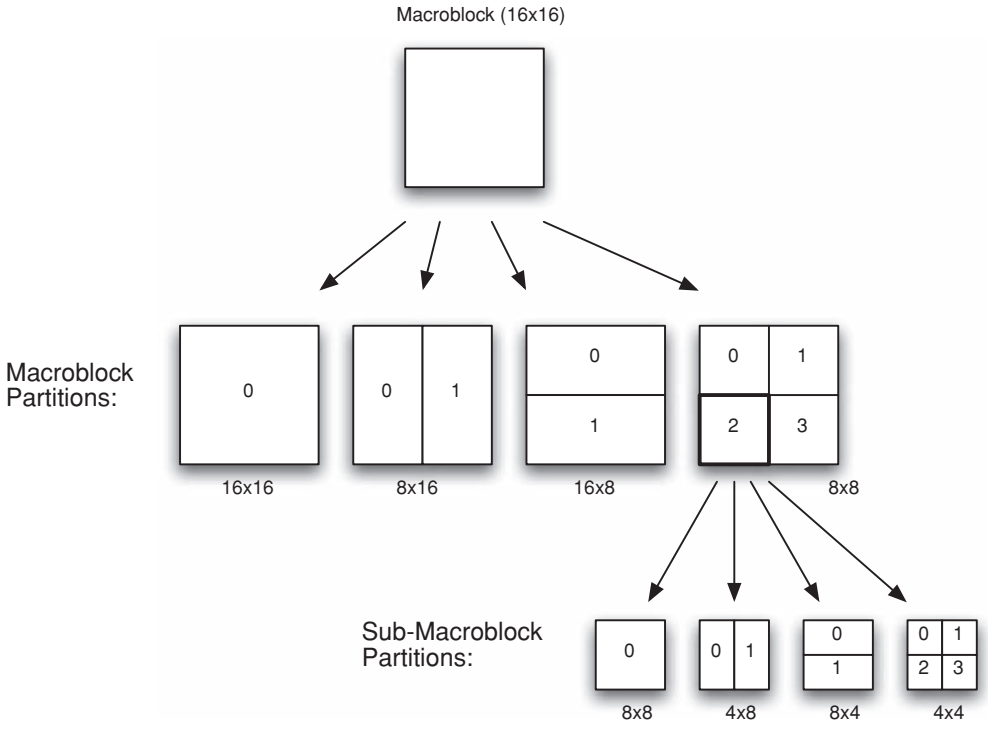


Figure 6.29 Macroblock partitions and sub-macroblock partitions

However, the sub-macroblock partitions within an 8 × 8 sub-macroblock share the same reference frame(s). Table 6.4 summarizes the information sent with each macroblock.

6.4.4 Motion vector prediction

Encoding a motion vector for each partition can cost a significant number of bits, especially if small partition sizes are chosen. Motion vectors for neighbouring partitions are often highly correlated and so each motion vector is predicted from vectors of nearby, previously coded partitions. A predicted vector, MVp, is formed based on previously calculated motion vectors and MVD, the difference between the current vector and the predicted vector, is encoded and

Table 6.4 Reference frames and motion vectors for P and B macroblocks

Macroblock type:	P	B
Reference frame indices:	One per macroblock partition, from list 0	One or two per macroblock partition, from list 0, list 1 or both lists.
Motion vector pairs (x,y):	One per macroblock partition or sub-macroblock partition	One or two per macroblock or sub-macroblock partition

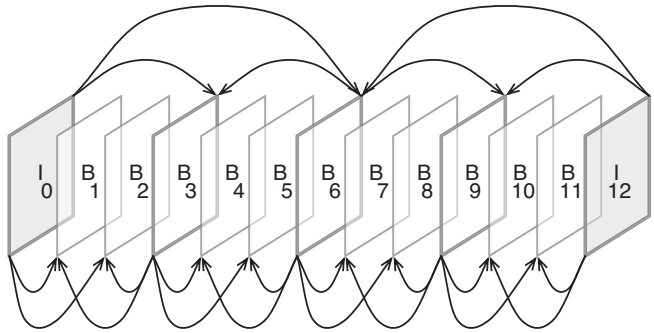


Figure 6.44 Hierarchical GOP structure

6.5 Loop filter

A filter is applied to every decoded macroblock to reduce blocking distortion [iv]. The deblocking filter is applied after the inverse transform in the encoder before reconstructing and storing the macroblock for future predictions and in the decoder before reconstructing and displaying the macroblock. The filter smooths block edges, improving the appearance of decoded frames. The filtered image is used for motion-compensated of future frames and this generally improves compression performance because the filtered image is a more faithful reproduction of the original frame than a blocky, unfiltered image.[1]

Filtering is applied to vertical or horizontal edges of 4×4 blocks in a macroblock excluding edges on slice boundaries, in the following order:

1. Filter 4 vertical boundaries of the luma component in order a,b,c,d in Figure 6.45
2. Filter 4 horizontal boundaries of the luma component in order e,f,g,h, Figure 6.45
3. Filter 2 vertical boundaries of each chroma component (i,j)
4. Filter 2 horizontal boundaries of each chroma component (k,l)

Each filtering operation affects up to **three** pixels on either side of the boundary. Figure 6.46 shows four pixels on either side of a vertical or horizontal boundary in adjacent blocks p and q, p0,p1,p2,p3 and q0,q1,q2,q3. The 'strength' of the filter, i.e. the amount

Table 6.5 Layers in hierarchical GOP example

Layer	Slices
0	I0, I12
1	B6
2	B3, B9
3	B1, B2, B4, B5, B7, B8, B10, B11

[1] Intra-coded macroblocks are filtered, but intra prediction (section 6.3) is carried out using **unfiltered** reconstructed macroblocks to form the prediction.

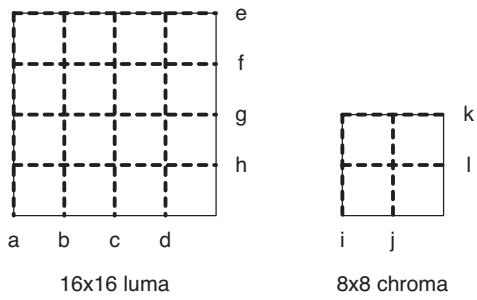


Figure 6.45 Edge filtering order in a macroblock

of filtering, depends on the current quantizer parameter QP, the coding modes of neighbouring blocks and the gradient of image samples across the boundary.

6.5.1 Boundary strength

The choice of filtering outcome depends on the **boundary strength** and on the **gradient** of image samples across the boundary. The boundary strength parameter **Bs** is chosen according to the following rules:

p or q is intra coded **and** boundary is a macroblock boundary	Bs = 4, strongest filtering
p or q is intra coded and boundary is **not** a macroblock boundary	Bs = 3
neither p or q is intra coded; p or q contain coded coefficients	Bs = 2
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have different reference frames **or** a different number of reference frames **or** different motion vector values	Bs = 1
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have same reference frame and identical motion vectors	Bs = 0, no filtering

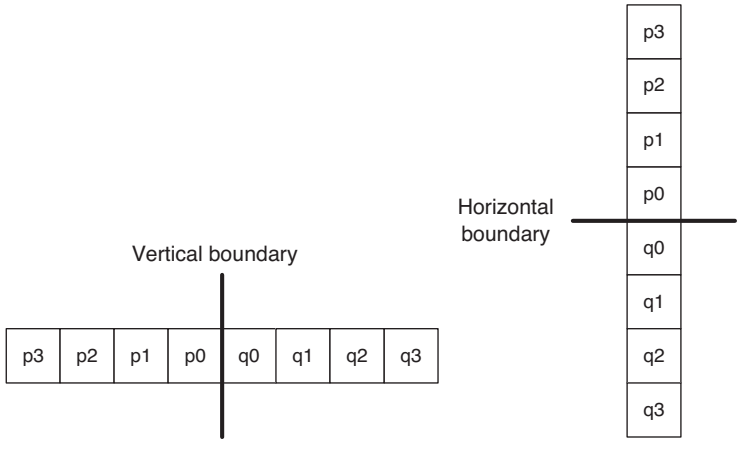


Figure 6.46 Pixels adjacent to vertical and horizontal boundaries

The filter is stronger at places where there is likely to be significant blocking distortion, such as the boundary of an intra coded macroblock or a boundary between blocks that contain coded coefficients.

6.5.2 Filter decision

A group of samples from the set (p2,p1,p0,q0,q1,q2) is filtered only if:

(a) Bs > 0 and
(b) |p0-q0|, |p1-p0| and |q1-q0| are each **less** than a threshold α or β, where α and β are defined in the standard.

The thresholds α and β increase with the average quantizer parameter QP of the two blocks p and q. The purpose of the filter decision is to 'switch off' the filter when there is a significant change or gradient across the block boundary in the original image. When QP is small, anything other than a very small gradient across the boundary is likely to be due to image features that should be preserved and not due to blocking effects and so the thresholds α and β are low. When QP is larger, blocking distortion is likely to be more significant and α, β are higher so that more boundary pixels are filtered.

Example

Figure 6.47 shows the 16 × 16 luma component of a macroblock without any blocking distortion with four 4 × 4 blocks a,b,c and d highlighted. Assuming a medium to large value of QP, the block boundary between a and b is likely to be filtered because the gradient across this boundary is small. There are no significant image features to preserve and blocking distortion will be quite obvious on this boundary. However, there is a significant change in luminance across the boundary between c and d due to a horizontal image feature and so the filter is switched off to preserve this strong filter.

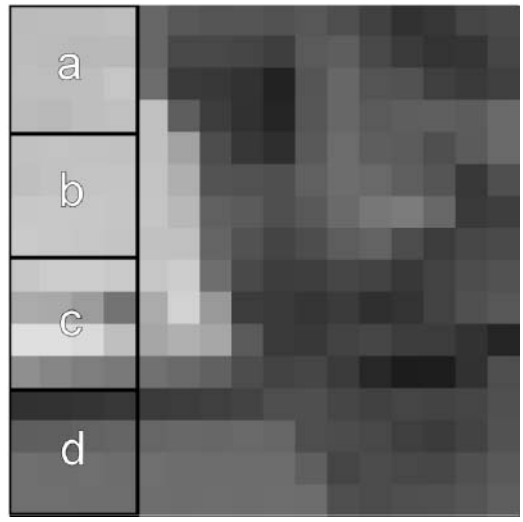


Figure 6.47 16 × 16 luma macroblock showing block edges

6.5.3 Filter implementation

(a) Bs ∈ {1,2,3}:
A 4-tap linear filter is applied with inputs p1, p0, q0 and q1, producing filtered outputs P0 and Q0 (0<Bs<4). If |p2-p0| is less than a threshold β, another 4-tap linear filter is applied with inputs p2, p1, p0 and q0, producing filtered output P1. If |q2-q0| is less than the threshold β, a 4-tap linear filter is applied with inputs q2, q1, q0 and p0, producing filtered output Q1. p1 and q1 are never filtered for chroma, only for luma data.

(b) Bs = 4:
If |p2-p0|<β and |p0-q0|<round(α/4):
P0 is produced by 5-tap filtering of p2, p1, p0, q0 and q1
P1 is produced by 4-tap filtering of p2, p1, p0 and q0
Luma only: P2 is produced by 5-tap filtering of p3, p2, p1, p0 and q0.
else:
P0 is produced by 3-tap filtering of p1, p0 and q1.

If |q2-q0|<β and |p0-q0|<round(α/4):
Q0 is produced by 5-tap filtering of q2, q1, q0, p0 and p1
Q1 is produced by 4-tap filtering of q2, q1, q0 and p0
Luma only: Q2 is produced by 5-tap filtering of q3, q2, q1, q0 and p0.
else:
Q0 is produced by 3-tap filtering of q1, q0 and p1.

6.5.4 Loop filter example

A video clip is encoded with a fixed Quantization Parameter of 36, i.e. relatively high quantization. Figure 6.48 shows an original frame from the clip and Figure 6.49 shows the same

Figure 6.48 Original frame, 'violin' frame 2

Figure 6.49 Reconstructed, QP = 36, no filter

frame after inter coding and decoding, with the loop filter disabled. Note the obvious blocking artefacts and note also the effect of varying motion-compensation block sizes for example, 16 × 16 blocks in the background to the left of the picture; 4 × 4 blocks around the arm. With the loop filter enabled (Figure 6.50) there is still some obvious distortion but most of the block edges have disappeared or faded. Note that sharp contrast boundaries such as the line of the arm against the dark piano are preserved by the filter whilst block edges in smoother regions of the picture such as the background to the left are smoothed. In this example the

Figure 6.50 Reconstructed, QP = 36, with filter

Figure 6.51 Reconstructed, QP = 32, no filter

loop filter makes only a small contribution to compression efficiency: the encoded bitrate is around 1.5 per cent smaller and the PSNR around 1 per cent larger for the sequence with the filter. However, the subjective quality of the filtered sequence is significantly better. The coding performance gain provided by the filter depends on the bitrate and sequence content.

Figure 6.51 and Figure 6.52 show the un-filtered and filtered frame respectively, this time with a lower quantizer parameter (QP = 32).

Figure 6.52 Reconstructed, QP = 32, with filter

8

H.264 conformance, transport and licensing

8.1 Introduction

Chapters 4,5,6 and 7 covered the basic concepts of H.264 Advanced Video Compression and the key algorithms that enable an H.264 codec to efficiently code and decode video. The purpose of an industry standard is to enable interoperability between encoders, bitstreams and decoders; i.e. the standard makes it possible for a bitstream encoded by one manufacturer's encoder to be decoded by a different manufacturer's decoder. H.264/AVC defines Profiles and Levels to place operational limits on (a) the particular coding tools and (b) the computational capacity and storage required to decode a sequence. Conformance is verified using a theoretical 'model', a Hypothetical Reference Decoder.

Practical applications of H.264/AVC involve transmitting and/or storing coded video information. The standard includes a number of features designed to support efficient, robust transport of the coded bitstream, including Parameter Sets and NAL Units, described in Chapter 5, and specific transport tools, described in this chapter. To help support an increasingly diverse range of video content and display types, 'side' information including Supplemental Enhancement Information and Video Usability Information may be transmitted along with the coded video data.

Video coding is big business, with many worldwide industries relying on video compression to enable digital media products and services. With many thousands of published patents in the field of video coding, licensing of H.264/AVC implementations is an important issue for commercial digital video applications.

8.2 Conforming to the Standard

H.264/AVC specifies many syntax options and decoding algorithms [i] which cover a wide range of potential video coding scenarios. The standard is designed to support video coding for applications such as small hand-held devices with limited display resolution and minimal computational capacity, through to high-definition decoders with large amounts of memory

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

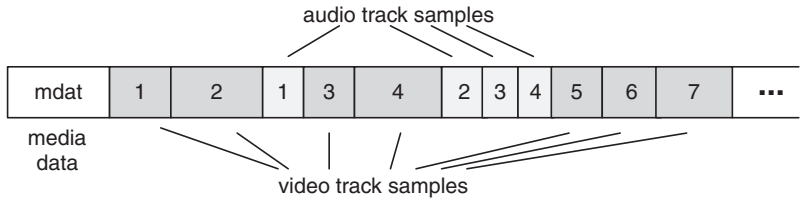


Figure 8.32 ISO Media File

of H.264 (section 8.4.1) has been designed with efficient packetisation in mind, since each NAL unit can be placed in its own RTP packet.

8.4.3 File formats

It is common for single compressed video sequences to be stored in files, simply by mapping the encoded stream to a sequence of bytes in a file. However, storing and playing back combined audio-visual data requires a more sophisticated file structure, especially when, for example, the stored data is to be streamed across a network or when the file is required to store multiple audio-visual objects. The H.264 File Format is designed to store H.264 Video data. It is derived from the ISO Base Media File Format which in turn is based on Apple Computer's QuickTime format.

In the ISO Media File Format, a coded stream such as an H.264 video sequence or an audio stream is stored as a **track**, representing a sequence of coded data items or **samples**, e.g. a coded VOP or coded slice, with time stamps (Figure 8.32). The file formats deal with issues such as synchronisation between tracks, random access indices and carriage of the file on a network transport mechanism.

8.4.4 Coding and transport issues

It has long been recognized that it is necessary to take into account practical transport issues in a video communication system and a number of tools in each standard are specifically designed to address these issues.

Scaling a delivered video stream to support decoders with different capabilities and/or delivery bitrates is addressed by the Scalable Video Coding extension of H.264/AVC (Chapter 10).

Latency is a particular issue for two-way real time applications such as videoconferencing. Tools such as B-slices, coded pictures that use motion-compensated prediction from earlier and later pictures in temporal order, can improve compression efficiency but typically introduce a delay of several frame periods into the coding and decoding 'chain' which may be unacceptable for low-latency two way applications. Latency requirements also have an influence on rate control algorithms since post-encoder and pre-decoder buffers, useful for smoothing out rate variations, lead to increased latency.

Each standard includes a number of features to aid the handling of transmission errors. Bit errors are a characteristic of circuit-switched channels; packet-switched networks tend to suffer from packet losses, since a bit error in a packet typically results in the packet being

Table 9.1 JM software directory structure

Directory	Description
bin or build	Executable files lencod, ldecod
doc	Documentation
lcommon	Source (C) and object code files common to encoder and decoder
ldecod	Decoder source and object code files
lencod	Encoder source and object code files

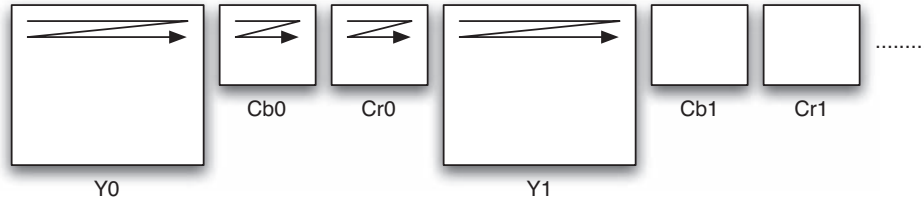


Figure 9.2 Planar YCbCr file format, 4:2:0 sampling

9.2.1.2 File formats

Source, reconstructed and decoded video files are in 'raw' YCbCr format, in which the luma, Cb and Cr samples are stored in the video file with no header or other information. Various sample orders are supported but the default is planar order, with each component of a frame stored in raster scan order, starting with frame 0 (Figure 9.2).

9.2.1.3 Basic operation

The JM software 'unpacks' and compiles into the directory structure shown in Table 9.1.

Example

Use the JM software to code 60 frames of a QCIF video sequence.

1. Download and unpack the JM software. Follow the instructions in *readme.txt* to compile lencod and ldecod.
2. Copy a QCIF source file into the *bin* directory. Various test video files are widely available: for example, *container.qcif*.
3. Create a configuration file. Start with one of the example files in the *bin* directory, e.g. encoder_baseline.cfg. Copy it and give it a new name, e.g. encoder_1.cfg.
4. Edit the new configuration file encoder_1.cfg (Figure 9.3). Changes are shaded; change the input and output file names, the number of frames to be encoded and set the quantizer parameter to 32.
5. Open a command prompt in the *bin* directory.
6. Run the encoder by typing:

```
lencod –d encoder_1.cfg
```

```
################################################
# Files
################################################
InputFile              = "container.qcif "       # Input sequence
InputHeaderLength      = 0      # If the inputfile has a header, state…
StartFrame             = 0      # Start frame for encoding. (0-N)
FramesToBeEncoded      = 60     # Number of frames to be coded
FrameRate              = 30.0   # Frame Rate per second (0.1-100.0)
SourceWidth            = 176    # Source frame width
SourceHeight           = 144    # Source frame height
SourceResize           = 0      # Resize source size for output
OutputWidth            = 176    # Output frame width
OutputHeight           = 144    # Output frame height
TraceFile              = "trace_enc.txt"      # Trace file
ReconFile              = "container_rec.qcif"
OutputFile             = "container.264"
StatsFile              = "stats.dat"          # Coding statistics file
################################################
# Encoder Control
################################################
ProfileIDC             = 66  # Profile IDC (66=baseline, 77=main, 88=extended…
IntraProfile           = 0   # Activate Intra Profile for FRExt (0: false, 1: true)
LevelIDC               = 40  # Level IDC   (e.g. 20 = level 2.0)
IntraPeriod            = 0   # Period of I-pictures   (0=only first)
IDRPeriod              = 0   # Period of IDR pictures (0=only first)
AdaptiveIntraPeriod    = 1   # Adaptive intra period
AdaptiveIDRPeriod      = 0   # Adaptive IDR period
IntraDelay             = 0   # Intra (IDR) picture delay (i.e. coding structure of PPIPPP... )
EnableIDRGOP           = 0   # Support for IDR closed GOPs (0: disabled, 1: enabled)
EnableOpenGOP          = 0   # Support for open GOPs (0: disabled, 1: enabled)
QPISlice               = 32  # Quant. param for I Slices (0-51)
QPPSlice               = 32  # Quant. param for P Slices (0-51)
FrameSkip              = 0   # Number of frames to be skipped in input
…
…
```

Figure 9.3 JM encoder configuration file

```
------------------------------ JM 16.0 (FRExt) ------------------------------
 Input YUV file                    : container.qcif
 Output H.264 bitstream            : container1.264
 Output YUV file                   : container_rec.qcif
 YUV Format                        : YUV 4:2:0
 Frames to be encoded I-P/B        : 60/0
 Freq. for encoded bitstream       : 30.00
…
…
-------------------------------------------------------------------------------
Frame     Bit/pic    QP   SnrY    SnrU    SnrV    Time(ms) MET(ms) Frm/Fld Ref
-------------------------------------------------------------------------------
00000(NVB)     160
00000(IDR)   17896   32  34.357  39.507  39.009        26       0    FRM    1
00001( P )     304   32  34.123  39.594  39.083        46      21    FRM    1
00002( P )     512   32  34.218  39.565  39.116        63      36    FRM    1
00003( P )     240   32  34.023  39.566  39.108        52      25    FRM    1
00004( P )     480   32  34.052  39.632  39.120        53      27    FRM    1
…
…
------------------ Average data all frames  ----------------------------------

 Total encoding time for the seq.  :   2.953 sec (20.32 fps)
 Total ME time for sequence        :   1.383 sec

 Y { PSNR (dB), cSNR (dB), MSE }   : {  33.578,  33.573,  28.56274 }
 U { PSNR (dB), cSNR (dB), MSE }   : {  39.404,  39.402,   7.46286 }
 V { PSNR (dB), cSNR (dB), MSE }   : {  38.910,  38.908,   8.36233 }

 Total bits                        : 50712 (I 17896, P 32656, NVB 160)
 Bit rate (kbit/s)  @ 30.00 Hz     : 25.36
 Bits to avoid Startcode Emulation : 0
 Bits for parameter sets           : 160
 Bits for filler data              : 0

-------------------------------------------------------------------------------
Exit JM 16 (FRExt) encoder ver 16.0
```

Figure 9.4 JM encoder output display

This will produce an output listing similar to that shown in Figure 9.4. Only selected lines are shown.

For each coded frame, the frame type, IDR/I or P in this case, number of coded bits, quantization parameter (QP), PSNR or 'Snr' for the components Y, U and V or Y, Cb, Cr and coding time are listed. Average PSNR and bitrate are listed at the end of the coded sequence. This is a relatively low-bitrate coded sequence at 21.85 kbps.

7. Run the decoder by typing:

```
ldecod –i container1.264 –o container_dec.qcif –r container.qcif
```

Input is container1.264; output is container_dec.qcif; use the original container.qcif as a reference for PSNR calculations.

The visual quality of the decoded QCIF file can be examined using a YUV viewer, a program that plays back a YUV file. A number of YUV viewers are available for download. Note that the Reconstructed file, container_rec.qcif in this case, and the Decoded file container_dec.qcif are identical. Figure 9.5 shows frame 56 from each QCIF file. Note that:

(i) The decoded/reconstructed frames have less detail due to quantization during encoding. A lower QP would improve decoded quality at the expense of a higher bitrate.
(ii) The decoded and reconstructed frames are identical.

Hence it is **not** actually necessary to run the decoder (ldecod) to view the decoded quality, since the encoder generates a decoded video file as well as all the necessary statistics such as bitrate and PSNR required to analyze coding performance.

9.2.1.4 Advanced operation

The large number of optional parameters in *encoder.cfg* give the user detailed control over the operation of the JM encoder. The main sections and their effect on encoder operation are as follows (Table 9.2). Note that some sections e.g. B slice parameters are not present in the Baseline configuration file and that the JM software continues to be developed and so these sections and their content may change.

Figure 9.5 Original, reconstructed and decoded frames, container.qcif, QP=32