

Iain E. G. Richardson

H.264 and MPEG-4

VIDEO COMPRESSION

 **WILEY**



H.264 and MPEG-4 Video Compression

H.264 and MPEG-4 Video Compression

Video Coding for Next-generation Multimedia

Iain E. G. Richardson

The Robert Gordon University, Aberdeen, UK



Copyright © 2003

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk

Visit our Home Page on www.wileyeurope.com or www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London W1T 4LP, UK, without the permission in writing of the Publisher.

Requests to the Publisher should be addressed to the Permissions Department, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, or emailed to permreq@wiley.co.uk, or faxed to (+44) 1243 770620.

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.

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

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

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

John Wiley & Sons Canada Ltd, 22 Worcester Road, Etobicoke, Ontario, Canada M9W 1L1

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

British Library Cataloguing in Publication Data

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

ISBN 0-470-84837-5

Typeset in 10/12pt Times roman by TechBooks, New Delhi, India

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

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

To Phyllis

Contents

About the Author	xiii
Foreword	xv
Preface	xix
Glossary	xxi
1 Introduction	1
1.1 The Scene	1
1.2 Video Compression	3
1.3 MPEG-4 and H.264	5
1.4 This Book	6
1.5 References	7
2 Video Formats and Quality	9
2.1 Introduction	9
2.2 Natural Video Scenes	9
2.3 Capture	10
2.3.1 Spatial Sampling	11
2.3.2 Temporal Sampling	11
2.3.3 Frames and Fields	13
2.4 Colour Spaces	13
2.4.1 RGB	14
2.4.2 YCbCr	15
2.4.3 YCbCr Sampling Formats	17
2.5 Video Formats	19
2.6 Quality	20
2.6.1 Subjective Quality Measurement	21
2.6.2 Objective Quality Measurement	22
2.7 Conclusions	24
2.8 References	24

3 Video Coding Concepts	27
3.1 Introduction	27
3.2 Video CODEC	28
3.3 Temporal Model	30
3.3.1 Prediction from the Previous Video Frame	30
3.3.2 Changes due to Motion	30
3.3.3 Block-based Motion Estimation and Compensation	32
3.3.4 Motion Compensated Prediction of a Macroblock	33
3.3.5 Motion Compensation Block Size	34
3.3.6 Sub-pixel Motion Compensation	37
3.3.7 Region-based Motion Compensation	41
3.4 Image model	42
3.4.1 Predictive Image Coding	44
3.4.2 Transform Coding	45
3.4.3 Quantisation	51
3.4.4 Reordering and Zero Encoding	56
3.5 Entropy Coder	61
3.5.1 Predictive Coding	61
3.5.2 Variable-length Coding	62
3.5.3 Arithmetic Coding	69
3.6 The Hybrid DPCM/DCT Video CODEC Model	72
3.7 Conclusions	82
3.8 References	83
 4 The MPEG-4 and H.264 Standards	 85
4.1 Introduction	85
4.2 Developing the Standards	85
4.2.1 ISO MPEG	86
4.2.2 ITU-T VCEG	87
4.2.3 JVT	87
4.2.4 Development History	88
4.2.5 Deciding the Content of the Standards	88
4.3 Using the Standards	89
4.3.1 What the Standards Cover	90
4.3.2 Decoding the Standards	90
4.3.3 Conforming to the Standards	91
4.4 Overview of MPEG-4 Visual/Part 2	92
4.5 Overview of H.264 / MPEG-4 Part 10	93
4.6 Comparison of MPEG-4 Visual and H.264	94
4.7 Related Standards	95
4.7.1 JPEG and JPEG2000	95
4.7.2 MPEG-1 and MPEG-2	95
4.7.3 H.261 and H.263	96
4.7.4 Other Parts of MPEG-4	97
4.8 Conclusions	97
4.9 References	98

5 MPEG-4 Visual	99
5.1 Introduction	99
5.2 Overview of MPEG-4 Visual (Natural Video Coding)	100
5.2.1 Features	100
5.2.2 Tools, Objects, Profiles and Levels	100
5.2.3 Video Objects	103
5.3 Coding Rectangular Frames	104
5.3.1 Input and Output Video Format	106
5.3.2 The Simple Profile	106
5.3.3 The Advanced Simple Profile	115
5.3.4 The Advanced Real Time Simple Profile	121
5.4 Coding Arbitrary-shaped Regions	122
5.4.1 The Core Profile	124
5.4.2 The Main Profile	133
5.4.3 The Advanced Coding Efficiency Profile	138
5.4.4 The N-bit Profile	141
5.5 Scalable Video Coding	142
5.5.1 Spatial Scalability	142
5.5.2 Temporal Scalability	144
5.5.3 Fine Granular Scalability	145
5.5.4 The Simple Scalable Profile	148
5.5.5 The Core Scalable Profile	148
5.5.6 The Fine Granular Scalability Profile	149
5.6 Texture Coding	149
5.6.1 The Scalable Texture Profile	152
5.6.2 The Advanced Scalable Texture Profile	152
5.7 Coding Studio-quality Video	153
5.7.1 The Simple Studio Profile	153
5.7.2 The Core Studio Profile	155
5.8 Coding Synthetic Visual Scenes	155
5.8.1 Animated 2D and 3D Mesh Coding	155
5.8.2 Face and Body Animation	156
5.9 Conclusions	156
5.10 References	156
 6 H.264/MPEG-4 Part 10	 159
6.1 Introduction	159
6.1.1 Terminology	159
6.2 The H.264 CODEC	160
6.3 H.264 structure	162
6.3.1 Profiles and Levels	162
6.3.2 Video Format	162
6.3.3 Coded Data Format	163
6.3.4 Reference Pictures	163
6.3.5 Slices	164
6.3.6 Macroblocks	164

6.4 The Baseline Profile	165
6.4.1 Overview	165
6.4.2 Reference Picture Management	166
6.4.3 Slices	167
6.4.4 Macroblock Prediction	169
6.4.5 Inter Prediction	170
6.4.6 Intra Prediction	177
6.4.7 Deblocking Filter	184
6.4.8 Transform and Quantisation	187
6.4.9 4×4 Luma DC Coefficient Transform and Quantisation (16×16 Intra-mode Only)	194
6.4.10 2×2 Chroma DC Coefficient Transform and Quantisation	195
6.4.11 The Complete Transform, Quantisation, Rescaling and Inverse Transform Process	196
6.4.12 Reordering	198
6.4.13 Entropy Coding	198
6.5 The Main Profile	207
6.5.1 B Slices	207
6.5.2 Weighted Prediction	211
6.5.3 Interlaced Video	212
6.5.4 Context-based Adaptive Binary Arithmetic Coding (CABAC)	212
6.6 The Extended Profile	216
6.6.1 SP and SI slices	216
6.6.2 Data Partitioned Slices	220
6.7 Transport of H.264	220
6.8 Conclusions	222
6.9 References	222
7 Design and Performance	225
7.1 Introduction	225
7.2 Functional Design	225
7.2.1 Segmentation	226
7.2.2 Motion Estimation	226
7.2.3 DCT/IDCT	234
7.2.4 Wavelet Transform	238
7.2.5 Quantise/Rescale	238
7.2.6 Entropy Coding	238
7.3 Input and Output	241
7.3.1 Interfacing	241
7.3.2 Pre-processing	242
7.3.3 Post-processing	243
7.4 Performance	246
7.4.1 Criteria	246
7.4.2 Subjective Performance	247
7.4.3 Rate-distortion Performance	251

7.4.4 Computational Performance	254
7.4.5 Performance Optimisation	255
7.5 Rate control	256
7.6 Transport and Storage	262
7.6.1 Transport Mechanisms	262
7.6.2 File Formats	263
7.6.3 Coding and Transport Issues	264
7.7 Conclusions	265
7.8 References	265
8 Applications and Directions	269
8.1 Introduction	269
8.2 Applications	269
8.3 Platforms	270
8.4 Choosing a CODEC	270
8.5 Commercial issues	272
8.5.1 Open Standards?	273
8.5.2 Licensing MPEG-4 Visual and H.264	274
8.5.3 Capturing the Market	274
8.6 Future Directions	275
8.7 Conclusions	276
8.8 References	276
Bibliography	277
Index	279

About the Author

Iain Richardson is a lecturer and researcher at The Robert Gordon University, Aberdeen, Scotland. He was awarded the degrees of MEng (Heriot-Watt University) and PhD (The Robert Gordon University) in 1990 and 1999 respectively. He has been actively involved in research and development of video compression systems since 1993 and is the author of over 40 journal and conference papers and two previous books. He leads the Image Communication Technology Research Group at The Robert Gordon University and advises a number of companies on video compression technology issues.

Foreword

Work on the emerging “Advanced Video Coding” standard now known as ITU-T Recommendation H.264 and as ISO/IEC 14496 (MPEG-4) Part 10 has dominated the video coding standardization community for roughly the past three years. The work has been stimulating, intense, dynamic, and all consuming for those of us most deeply involved in its design. The time has arrived to see what has been accomplished.

Although not a direct participant, Dr Richardson was able to develop a high-quality, up-to-date, introductory description and analysis of the new standard. The timeliness of this book is remarkable, as the standard itself has only just been completed.

The new H.264/AVC standard is designed to provide a technical solution appropriate for a broad range of applications, including:

- Broadcast over cable, satellite, cable modem, DSL, terrestrial.
- Interactive or serial storage on optical and magnetic storage devices, DVD, etc.
- Conversational services over ISDN, Ethernet, LAN, DSL, wireless and mobile networks, modems.
- Video-on-demand or multimedia streaming services over cable modem, DSL, ISDN, LAN, wireless networks.
- Multimedia messaging services over DSL, ISDN.

The range of bit rates and picture sizes supported by H.264/AVC is correspondingly broad, addressing video coding capabilities ranging from very low bit rate, low frame rate, “postage stamp” resolution video for mobile and dial-up devices, through to entertainment-quality standard-definition television services, HDTV, and beyond. A flexible system interface for the coded video is specified to enable the adaptation of video content for use over this full variety of network and channel-type environments. However, at the same time, the technical design is highly focused on providing the two limited goals of high coding efficiency and robustness to network environments for conventional rectangular-picture camera-view video content. Some potentially-interesting (but currently non-mainstream) features were deliberately left out (at least from the first version of the standard) because of that focus (such as support of arbitrarily-shaped video objects, some forms of bit rate scalability, 4:2:2 and 4:4:4 chroma formats, and color sampling accuracies exceeding eight bits per color component).

In the work on the new H.264/AVC standard, a number of relatively new technical developments have been adopted. For increased coding efficiency, these include improved prediction design aspects as follows:

- Variable block-size motion compensation with small block sizes,
- Quarter-sample accuracy for motion compensation,
- Motion vectors over picture boundaries,
- Multiple reference picture motion compensation,
- Decoupling of referencing order from display order,
- Decoupling of picture representation methods from the ability to use a picture for reference,
- Weighted prediction,
- Improved “skipped” and “direct” motion inference,
- Directional spatial prediction for intra coding, and
- In-the-loop deblocking filtering.

In addition to improved prediction methods, other aspects of the design were also enhanced for improved coding efficiency, including:

- Small block-size transform,
- Hierarchical block transform,
- Short word-length transform,
- Exact-match transform,
- Arithmetic entropy coding, and
- Context-adaptive entropy coding.

And for robustness to data errors/losses and flexibility for operation over a variety of network environments, some key design aspects include:

- Parameter set structure,
- NAL unit syntax structure,
- Flexible slice size,
- Flexible macroblock ordering,
- Arbitrary slice ordering,
- Redundant pictures,
- Data partitioning, and
- SP/SI synchronization switching pictures.

Prior to the H.264/AVC project, the big recent video coding activity was the MPEG-4 Part 2 (Visual) coding standard. That specification introduced a new degree of creativity and flexibility to the capabilities of the representation of digital visual content, especially with its coding of video “objects”, its scalability features, extended N-bit sample precision and 4:4:4 color format capabilities, and its handling of synthetic visual scenes. It introduced a number of design variations (called “profiles” and currently numbering 19 in all) for a wide variety of applications. The H.264/AVC project (with only 3 profiles) returns to the narrower and more traditional focus on efficient compression of generic camera-shot rectangular video pictures with robustness to network losses – making no attempt to cover the ambitious breadth of MPEG-4 Visual. MPEG-4 Visual, while not quite as “hot off the press”, establishes a landmark in recent technology development, and its capabilities are yet to be fully explored.

Most people first learn about a standard in publications other than the standard itself. My personal belief is that if you want to know about a standard, you should also obtain a copy of it, read it, and refer to that document alone as the ultimate authority on its content, its boundaries, and its capabilities. No tutorial or overview presentation will provide all of the insights that can be obtained from careful analysis of the standard itself.

At the same time, no standardized specification document (at least for video coding), can be a complete substitute for a good technical book on the subject. Standards specifications are written primarily to be precise, consistent, complete, and correct and not to be particularly readable. Standards tend to leave out information that is not *absolutely* necessary to comply with them. Many people find it surprising, for example, that video coding standards say almost nothing about how an encoder works or how one should be designed. In fact an encoder is essentially allowed to do anything that produces bits that can be correctly decoded, regardless of what picture quality comes out of that decoding process. People, however, can usually only understand the principles of video coding if they think from the perspective of the encoder, and nearly all textbooks (including this one) approach the subject from the encoding perspective. A good book, such as this one, will tell you why a design is the way it is and how to make use of that design, while a good standard may only tell you exactly what it is and abruptly (deliberately) stop right there.

In the case of H.264/AVC or MPEG-4 Visual, it is highly advisable for those new to the subject to read some introductory overviews such as this one, and even to get a copy of an older and simpler standard such as H.261 or MPEG-1 and try to understand that first. The principles of digital video codec design are not too complicated, and haven't really changed much over the years – but those basic principles have been wrapped in layer-upon-layer of technical enhancements to the point that the simple and straightforward concepts that lie at their core can become obscured. The entire H.261 specification was only 25 pages long, and only 17 of those pages were actually required to fully specify the technology that now lies at the heart of all subsequent video coding standards. In contrast, the H.264/AVC and MPEG-4 Visual and specifications are more than 250 and 500 pages long, respectively, with a high density of technical detail (despite completely leaving out key information such as how to encode video using their formats). They each contain areas that are difficult even for experts to fully comprehend and appreciate.

Dr Richardson's book is not a completely exhaustive treatment of the subject. However, his approach is highly informative and provides a good initial understanding of the key concepts, and his approach is conceptually superior (and in some aspects more objective) to other treatments of video coding publications. This and the remarkable timeliness of the subject matter make this book a strong contribution to the technical literature of our community.

Gary J. Sullivan

Biography of Gary J. Sullivan, PhD

Gary J. Sullivan is the chairman of the Joint Video Team (JVT) for the development of the latest international video coding standard known as H.264/AVC, which was recently completed as a joint project between the ITU-T video coding experts group (VCEG) and the ISO/IEC moving picture experts group (MPEG).

He is also the Rapporteur of Advanced Video Coding in the ITU-T, where he has led VCEG (ITU-T Q.6/SG16) for about seven years. He is also the ITU-T video liaison representative to MPEG and served as MPEG's (ISO/IEC JTC1/SC29/WG11) video chairman from March of 2001 to May of 2002.

He is currently a program manager of video standards and technologies in the eHome A/V platforms group of Microsoft Corporation. At Microsoft he designed and remains active in the extension of DirectX[®] Video Acceleration API/DDI feature of the Microsoft Windows[®] operating system platform.

Preface

With the widespread adoption of technologies such as digital television, Internet streaming video and DVD-Video, video compression has become an essential component of broadcast and entertainment media. The success of digital TV and DVD-Video is based upon the 10-year-old MPEG-2 standard, a technology that has proved its effectiveness but is now looking distinctly old-fashioned. It is clear that the time is right to replace MPEG-2 video compression with a more effective and efficient technology that can take advantage of recent progress in processing power. For some time there has been a running debate about which technology should take up MPEG-2's mantle. The leading contenders are the International Standards known as MPEG-4 Visual and H.264.

This book aims to provide a clear, practical and unbiased guide to these two standards to enable developers, engineers, researchers and students to understand and apply them effectively. Video and image compression is a complex and extensive subject and this book keeps an unapologetically limited focus, concentrating on the standards themselves (and in the case of MPEG-4 Visual, on the elements of the standard that support coding of 'real world' video material) and on video coding concepts that directly underpin the standards. The book takes an application-based approach and places particular emphasis on tools and features that are helpful in practical applications, in order to provide practical and useful assistance to developers and adopters of these standards.

I am grateful to a number of people who helped to shape the content of this book. I received many helpful comments and requests from readers of my book *Video Codec Design*. Particular thanks are due to Gary Sullivan for taking the time to provide helpful and detailed comments, corrections and advice and for kindly agreeing to write a Foreword; to Harvey Hanna (Impact Labs Inc), Yafan Zhao (The Robert Gordon University) and Aitor Garay for reading and commenting on sections of this book during its development; to members of the Joint Video Team for clarifying many of the details of H.264; to the editorial team at John Wiley & Sons (and especially to the ever-helpful, patient and supportive Kathryn Sharples); to Phyllis for her constant support; and finally to Freya and Hugh for patiently waiting for the long-promised trip to Storybook Glen!

I very much hope that you will find this book enjoyable, readable and above all useful. Further resources and links are available at my website, <http://www.vcodex.com/>. I always appreciate feedback, comments and suggestions from readers and you will find contact details at this website.

Iain Richardson

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
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
BAB	Binary Alpha Block, indicates the boundaries of a region (MPEG-4 Visual)
BAP	Body Animation Parameters
Block	Region of macroblock (8×8 or 4×4) for transform purposes
block matching	Motion estimation carried out on rectangular picture areas
blocking	Square or rectangular distortion areas in an image
B-picture (slice)	Coded picture (slice) predicted using bidirectional motion compensation
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAE	Context-based Arithmetic Encoding
CAVLC	Context Adaptive Variable Length Coding
chrominance	Colour difference component
CIF	Common Intermediate Format, a colour image format
CODEC	<i>CO</i> der / <i>DE</i> Coder pair
colour space	Method of representing colour images
DCT	Discrete Cosine Transform
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	Exponential Golomb variable length codes
FAP	Facial Animation Parameters
FBA	Face and Body Animation
FGS	Fine Granular Scalability
field	Odd- or even-numbered lines from an interlaced video sequence
flowgraph	Pictorial representation of a transform algorithm (or the algorithm itself)
FMO	Flexible Macroblock Order, in which macroblocks may be coded out of raster sequence
Full Search	A motion estimation algorithm
GMC	Global Motion Compensation, motion compensation applied to a complete coded object (MPEG-4 Visual)
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
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-picture (slice)	Picture (or 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)
JPEG2000	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
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 (H.264)
Macroblock sub-partition	Region of macroblock with its own motion vector (H.264)
media processor	Processor with features specific to multimedia coding and processing
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
NAL	Network Abstraction Layer
objective quality	Visual quality measured by algorithm(s)
OBCM	Overlapped Block Motion Compensation
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
quantise	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
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
RVLC	Reversible Variable Length Code
scalable coding	Coding a signal into a number of layers
SI slice	Intra-coded slice used for switching between coded bitstreams (H.264)
slice	A region of a coded picture
SNHC	Synthetic Natural Hybrid Coding
SP slice	Inter-coded slice used for switching between coded bitstreams (H.264)
sprite	Texture region that may be incorporated in a series of decoded frames (MPEG-4 Visual)
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 (H.264)

TSS	Three Step Search, a motion estimation algorithm
VCEG	Video Coding Experts Group, a committee of ITU
VCL	Video Coding Layer
video packet	Coded unit suitable for packetisation
VLC	Variable Length Code
VLD	Variable Length Decoder
VLE	Variable Length Encoder
VLSI	Very Large Scale Integrated circuit
VO	Video Object
VOP	Video Object Plane
VQEG	Video Quality Experts Group
VQEG	Video Quality Experts Group
Weighted prediction	Motion compensation in which the prediction samples from two references are scaled
YCbCr	Luminance, Blue chrominance, Red chrominance colour space
YUV	A colour space (see YCbCr)

1

Introduction

1.1 THE SCENE

Scene 1: Your avatar (a realistic 3D model with your appearance and voice) walks through a sophisticated virtual world populated by other avatars, product advertisements and video walls. On one virtual video screen is a news broadcast from your favourite channel; you want to see more about the current financial situation and so you interact with the broadcast and pull up the latest stock market figures. On another screen you call up a videoconference link with three friends. The video images of the other participants, neatly segmented from their backgrounds, are presented against yet another virtual backdrop.

Scene 2: Your new 3G vidphone rings; you flip the lid open and answer the call. The face of your friend appears on the screen and you greet each other. Each sees a small, clear image of the other on the phone's screen, without any of the obvious 'blockiness' of older-model video phones. After the call has ended, you call up a live video feed from a football match. The quality of the basic-rate stream isn't too great and you switch seamlessly to the higher-quality (but more expensive) 'premium' stream. For a brief moment the radio signal starts to break up but all you notice is a slight, temporary distortion in the video picture.

These two scenarios illustrate different visions of the next generation of multimedia applications. The first is a vision of MPEG-4 Visual: a rich, interactive on-line world bringing together synthetic, natural, video, image, 2D and 3D 'objects'. The second is a vision of H.264/AVC: highly efficient and reliable video communications, supporting two-way, 'streaming' and broadcast applications and robust to channel transmission problems. The two standards, each with their advantages and disadvantages and each with their supporters and critics, are contenders in the race to provide video compression for next-generation communication applications.

Turn on the television and surf through tens or hundreds of digital channels. Play your favourite movies on the DVD player and breathe a sigh of relief that you can throw out your antiquated VHS tapes. Tune in to a foreign TV news broadcast on the web (still just a postage-stamp video window but the choice and reliability of video streams is growing all the time). Chat to your friends and family by PC videophone. These activities are now commonplace and unremarkable, demonstrating that digital video is well on the way to becoming a ubiquitous

and essential component of the entertainment, computing, broadcasting and communications industries.

Pervasive, seamless, high-quality digital video has been the goal of companies, researchers and standards bodies over the last two decades. In some areas (for example broadcast television and consumer video storage), digital video has clearly captured the market, whilst in others (videoconferencing, video email, mobile video), market success is perhaps still too early to judge. However, there is no doubt that digital video is a globally important industry which will continue to pervade businesses, networks and homes. The continuous evolution of the digital video industry is being driven by commercial and technical forces. The commercial drive comes from the huge revenue potential of persuading consumers and businesses (a) to replace analogue technology and older digital technology with new, efficient, high-quality digital video products and (b) to adopt new communication and entertainment products that have been made possible by the move to digital video. The technical drive comes from continuing improvements in processing performance, the availability of higher-capacity storage and transmission mechanisms and research and development of video and image processing technology.

Getting digital video from its source (a camera or a stored clip) to its destination (a display) involves a chain of components or processes. Key to this chain are the processes of compression (encoding) and decompression (decoding), in which bandwidth-intensive ‘raw’ digital video is reduced to a manageable size for transmission or storage, then reconstructed for display. Getting the compression and decompression processes ‘right’ can give a significant technical and commercial edge to a product, by providing better image quality, greater reliability and/or more flexibility than competing solutions. There is therefore a keen interest in the continuing development and improvement of video compression and decompression methods and systems. The interested parties include entertainment, communication and broadcasting companies, software and hardware developers, researchers and holders of potentially lucrative patents on new compression algorithms.

The early successes in the digital video industry (notably broadcast digital television and DVD-Video) were underpinned by international standard ISO/IEC 13818 [1], popularly known as ‘MPEG-2’ (after the working group that developed the standard, the Moving Picture Experts Group). Anticipation of a need for better compression tools has led to the development of two further standards for video compression, known as ISO/IEC 14496 Part 2 (‘MPEG-4 Visual’) [2] and ITU-T Recommendation H.264/ISO/IEC 14496 Part 10 (‘H.264’) [3]. MPEG-4 Visual and H.264 share the same ancestry and some common features (they both draw on well-proven techniques from earlier standards) but have notably different visions, seeking to improve upon the older standards in different ways. The vision of MPEG-4 Visual is to move away from a restrictive reliance on rectangular video images and to provide an open, flexible framework for visual communications that uses the best features of efficient video compression and object-oriented processing. In contrast, H.264 has a more pragmatic vision, aiming to do what previous standards did (provide a mechanism for the compression of rectangular video images) but to do it in a more efficient, robust and practical way, supporting the types of applications that are becoming widespread in the marketplace (such as broadcast, storage and streaming).

At the present time there is a lively debate about which (if either) of these standards will come to dominate the market. MPEG-4 Visual is the more mature of the two new standards (its first Edition was published in 1999, whereas H.264 became an International

Standard/Recommendation in 2003). There is no doubt that H.264 can out-perform MPEG-4 Visual in compression efficiency but it does not have the older standard's bewildering flexibility. The licensing situation with regard to MPEG-4 Visual is clear (and not popular with some parts of the industry) but the cost of licensing H.264 remains to be agreed. This book is about these two important new standards and examines the background to the standards, the core concepts and technical details of each standard and the factors that will determine the answer to the question 'MPEG-4 Visual or H.264?'.

1.2 VIDEO COMPRESSION

Network bitrates continue to increase (dramatically in the local area and somewhat less so in the wider area), high bitrate connections to the home are commonplace and the storage capacity of hard disks, flash memories and optical media is greater than ever before. With the price per transmitted or stored bit continually falling, it is perhaps not immediately obvious why video compression is necessary (and why there is such a significant effort to make it better). Video compression has two important benefits. First, it makes it possible to use digital video in transmission and storage environments that would not support uncompressed ('raw') video. For example, current Internet throughput rates are insufficient to handle uncompressed video in real time (even at low frame rates and/or small frame size). A Digital Versatile Disk (DVD) can only store a few seconds of raw video at television-quality resolution and frame rate and so DVD-Video storage would not be practical without video and audio compression. Second, video compression enables more efficient use of transmission and storage resources. If a high bitrate transmission channel is available, then it is a more attractive proposition to send high-resolution compressed video or multiple compressed video channels than to send a single, low-resolution, uncompressed stream. Even with constant advances in storage and transmission capacity, compression is likely to be an essential component of multimedia services for many years to come.

An information-carrying signal may be compressed by removing redundancy from the signal. In a lossless compression system statistical redundancy is removed so that the original signal can be perfectly reconstructed at the receiver. Unfortunately, at the present time lossless methods can only achieve a modest amount of compression of image and video signals. Most practical video compression techniques are based on lossy compression, in which greater compression is achieved with the penalty that the decoded signal is not identical to the original. The goal of a video compression algorithm is to achieve efficient compression whilst minimising the distortion introduced by the compression process.

Video compression algorithms operate by removing redundancy in the temporal, spatial and/or frequency domains. Figure 1.1 shows an example of a single video frame. Within the highlighted regions, there is little variation in the content of the image and hence there is significant spatial redundancy. Figure 1.2 shows the same frame after the background region has been low-pass filtered (smoothed), removing some of the higher-frequency content. The human eye and brain (Human Visual System) are more sensitive to lower frequencies and so the image is still recognisable despite the fact that much of the 'information' has been removed. Figure 1.3 shows the next frame in the video sequence. The sequence was captured from a camera at 25 frames per second and so there is little change between the two frames in the short interval of $1/25$ of a second. There is clearly significant temporal redundancy, i.e. most



Figure 1.1 Video frame (showing examples of homogeneous regions)



Figure 1.2 Video frame (low-pass filtered background)



Figure 1.3 Video frame 2

of the image remains unchanged between successive frames. By removing different types of redundancy (spatial, frequency and/or temporal) it is possible to compress the data significantly at the expense of a certain amount of information loss (distortion). Further compression can be achieved by encoding the processed data using an entropy coding scheme such as Huffman coding or Arithmetic coding.

Image and video compression has been a very active field of research and development for over 20 years and many different systems and algorithms for compression and decompression have been proposed and developed. In order to encourage interworking, competition and increased choice, it has been necessary to define standard methods of compression encoding and decoding to allow products from different manufacturers to communicate effectively. This has led to the development of a number of key International Standards for image and video compression, including the JPEG, MPEG and H.26x series of standards.

1.3 MPEG-4 AND H.264

MPEG-4 Visual and H.264 (also known as Advanced Video Coding) are standards for the coded representation of visual information. Each standard is a document that primarily defines two things, a coded representation (or syntax) that describes visual data in a compressed form and a method of decoding the syntax to reconstruct visual information. Each standard aims to ensure that compliant encoders and decoders can successfully interwork with each other, whilst allowing manufacturers the freedom to develop competitive and innovative products. The standards specifically do not define an encoder; rather, they define the output that an encoder should

produce. A decoding method is defined in each standard but manufacturers are free to develop alternative decoders as long as they achieve the same result as the method in the standard.

MPEG-4 Visual (Part 2 of the MPEG-4 group of standards) was developed by the Moving Picture Experts Group (MPEG), a working group of the International Organisation for Standardisation (ISO). This group of several hundred technical experts (drawn from industry and research organisations) meet at 2–3 month intervals to develop the MPEG series of standards. MPEG-4 (a multi-part standard covering audio coding, systems issues and related aspects of audio/visual communication) was first conceived in 1993 and Part 2 was standardised in 1999. The H.264 standardisation effort was initiated by the Video Coding Experts Group (VCEG), a working group of the International Telecommunication Union (ITU-T) that operates in a similar way to MPEG and has been responsible for a series of visual telecommunication standards. The final stages of developing the H.264 standard have been carried out by the Joint Video Team, a collaborative effort of both VCEG and MPEG, making it possible to publish the final standard under the joint auspices of ISO/IEC (as MPEG-4 Part 10) and ITU-T (as Recommendation H.264) in 2003.

MPEG-4 Visual and H.264 have related but significantly different visions. Both are concerned with compression of visual data but MPEG-4 Visual emphasises flexibility whilst H.264's emphasis is on efficiency and reliability. MPEG-4 Visual provides a highly flexible toolkit of coding techniques and resources, making it possible to deal with a wide range of types of visual data including rectangular frames ('traditional' video material), video objects (arbitrary-shaped regions of a visual scene), still images and hybrids of natural (real-world) and synthetic (computer-generated) visual information. MPEG-4 Visual provides its functionality through a set of coding tools, organised into 'profiles', recommended groupings of tools suitable for certain applications. Classes of profiles include 'simple' profiles (coding of rectangular video frames), object-based profiles (coding of arbitrary-shaped visual objects), still texture profiles (coding of still images or 'texture'), scalable profiles (coding at multiple resolutions or quality levels) and studio profiles (coding for high-quality studio applications).

In contrast with the highly flexible approach of MPEG-4 Visual, H.264 concentrates specifically on efficient compression of video frames. Key features of the standard include compression efficiency (providing significantly better compression than any previous standard), transmission efficiency (with a number of built-in features to support reliable, robust transmission over a range of channels and networks) and a focus on popular applications of video compression. Only three profiles are currently supported (in contrast to nearly 20 in MPEG-4 Visual), each targeted at a class of popular video communication applications. The Baseline profile may be particularly useful for "conversational" applications such as video-conferencing, the Extended profile adds extra tools that are likely to be useful for video streaming across networks and the Main profile includes tools that may be suitable for consumer applications such as video broadcast and storage.

1.4 THIS BOOK

The aim of this book is to provide a technically-oriented guide to the MPEG-4 Visual and H.264/AVC standards, with an emphasis on practical issues. Other works cover the details of the other parts of the MPEG-4 standard [4–6] and this book concentrates on the application of MPEG-4 Visual and H.264 to the coding of natural video. Most practical applications of

MPEG-4 (and emerging applications of H.264) make use of a subset of the tools provided by each standard (a 'profile') and so the treatment of each standard in this book is organised according to profile, starting with the most basic profiles and then introducing the extra tools supported by more advanced profiles.

Chapters 2 and 3 cover essential background material that is required for an understanding of both MPEG-4 Visual and H.264. Chapter 2 introduces the basic concepts of digital video including capture and representation of video in digital form, colour-spaces, formats and quality measurement. Chapter 3 covers the fundamentals of video compression, concentrating on aspects of the compression process that are common to both standards and introducing the transform-based CODEC 'model' that is at the heart of all of the major video coding standards.

Chapter 4 looks at the standards themselves and examines the way that the standards have been shaped and developed, discussing the composition and procedures of the VCEG and MPEG standardisation groups. The chapter summarises the content of the standards and gives practical advice on how to approach and interpret the standards and ensure conformance. Related image and video coding standards are briefly discussed.

Chapters 5 and 6 focus on the technical features of MPEG-4 Visual and H.264. The approach is based on the structure of the Profiles of each standard (important conformance points for CODEC developers). The Simple Profile (and related Profiles) have shown themselves to be by far the most popular features of MPEG-4 Visual to date and so Chapter 5 concentrates first on the compression tools supported by these Profiles, followed by the remaining (less commercially popular) Profiles supporting coding of video objects, still texture, scalable objects and so on. Because this book is primarily about compression of natural (real-world) video information, MPEG-4 Visual's synthetic visual tools are covered only briefly. H.264's Baseline Profile is covered first in Chapter 6, followed by the extra tools included in the Main and Extended Profiles. Chapters 5 and 6 make extensive reference back to Chapter 3 (Video Coding Concepts). H.264 is dealt with in greater technical detail than MPEG-4 Visual because of the limited availability of reference material on the newer standard.

Practical issues related to the design and performance of video CODECs are discussed in Chapter 7. The design requirements of each of the main functional modules required in a practical encoder or decoder are addressed, from motion estimation through to entropy coding. The chapter examines interface requirements and practical approaches to pre- and post-processing of video to improve compression efficiency and/or visual quality. The compression and computational performance of the two standards is compared and rate control (matching the encoder output to practical transmission or storage mechanisms) and issues faced in transporting and storing of compressed video are discussed.

Chapter 8 examines the requirements of some current and emerging applications, lists some currently-available CODECs and implementation platforms and discusses the important implications of commercial factors such as patent licenses. Finally, some predictions are made about the next steps in the standardisation process and emerging research issues that may influence the development of future video coding standards.

1.5 REFERENCES

1. ISO/IEC 13818, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information, 2000.

2. ISO/IEC 14496-2, Coding of Audio-Visual Objects – Part 2:Visual, 2001.
3. ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, 2003.
4. F. Pereira and T. Ebrahimi (eds), *The MPEG-4 Book*, IMSC Press, 2002.
5. A. Walsh and M. Bourges-Sévenier (eds), *MPEG-4 Jump Start*, Prentice-Hall, 2002.
6. ISO/IEC JTC1/SC29/WG11 N4668, MPEG-4 Overview, <http://www.m4if.org/resources/Overview.pdf>, March 2002.

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 that are helpful to an understanding of video coding. Digital video is a representation of a natural (real-world) visual scene, sampled spatially and temporally. A scene is sampled at a point in time to produce a frame (a representation of the complete visual scene at that point in time) or a field (consisting 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 sets of samples (components) are typically required to represent a scene in colour. Popular formats for representing video in digital form include the ITU-R 601 standard and the 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 (automatic) measurements are easier to implement but as yet do not accurately match the opinion of a 'real' human.

2.2 NATURAL VIDEO SCENES

A typical 'real world' or 'natural' video scene is 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 ('continuous tone'). Characteristics of a typical natural video scene (Figure 2.1) that are relevant for video processing and compression include spatial characteristics (texture variation within scene, number and shape of objects, colour, etc.) and temporal characteristics (object motion, changes in illumination, movement of the camera or viewpoint and so on).



Figure 2.1 Still image from natural video scene

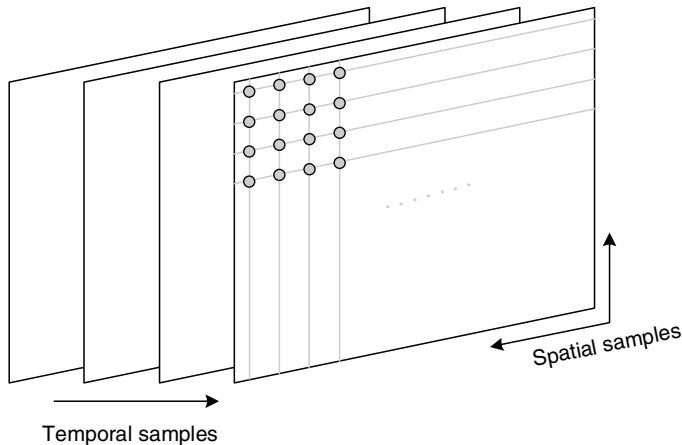


Figure 2.2 Spatial and temporal sampling of a video sequence

2.3 CAPTURE

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

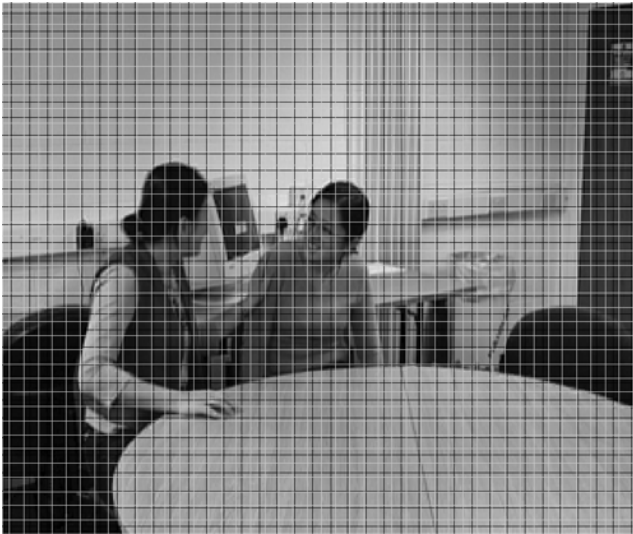


Figure 2.3 Image with 2 sampling grids

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

2.3.1 Spatial Sampling

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

2.3.2 Temporal Sampling

A moving video image is captured by taking a rectangular ‘snapshot’ of the signal at periodic time intervals. Playing back the series of frames produces the appearance of motion. A higher temporal sampling rate (frame rate) gives apparently smoother motion in the video scene but requires more samples to be captured and stored. Frame rates below 10 frames per second are sometimes used for very low bit-rate video communications (because the amount of data



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



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

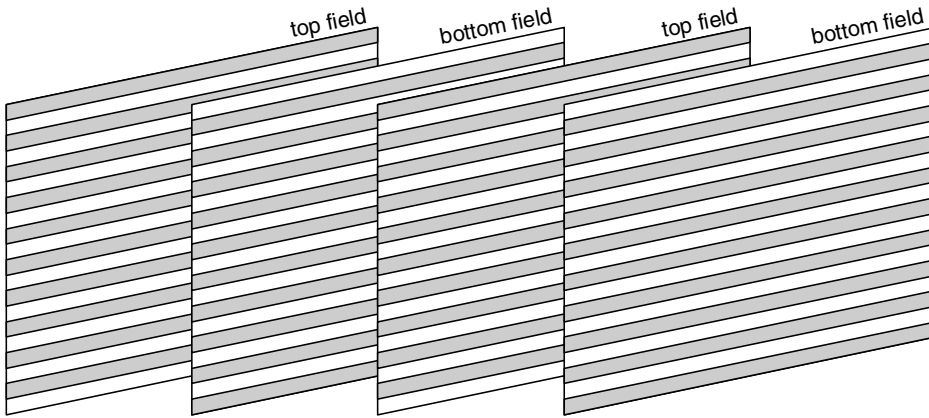


Figure 2.6 Interlaced video sequence

is relatively small) but motion is clearly jerky and unnatural at this rate. Between 10 and 20 frames per second is more typical for low bit-rate video communications; the image is smoother but jerky motion may be visible in fast-moving parts of the sequence. Sampling at 25 or 30 complete frames per second is standard for television pictures (with interlacing to improve the appearance of motion, see below); 50 or 60 frames per second produces smooth apparent motion (at the expense of a very high data rate).

2.3.3 Frames and Fields

A video signal may be sampled as a series of complete frames (*progressive* sampling) or as a sequence of interlaced fields (*interlaced* sampling). In an interlaced video sequence, half of the data in a frame (one field) is sampled at each temporal sampling interval. A field consists of either the odd-numbered or even-numbered lines within a complete video frame and an interlaced video sequence (Figure 2.6) contains a series of fields, each representing half of the information in a complete video frame (e.g. Figure 2.7 and Figure 2.8). The 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 can appear smoother than in an equivalent progressive video sequence containing 25 frames per second.

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 (e.g. 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 represent colour accurately. The method chosen to represent brightness (luminance or luma) and colour is described as a colour space.



Figure 2.7 Top field



Figure 2.8 Bottom field

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). Any colour can be created by combining red, green and blue in varying proportions. 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 Cathode Ray Tubes (CRTs) and Liquid Crystal Displays (LCDs) display 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 YCbCr

The human visual system (HVS) is less sensitive to colour than to luminance (brightness). 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 YCbCr colour space and its variations (sometimes referred to as YUV) is a popular way of efficiently representing colour images. Y is the luminance (luma) component and can be calculated as a weighted average of R, G and B:

$$Y = k_r R + k_g G + k_b B \quad (2.1)$$

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 :

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

The complete description of a colour image is given by Y (the luminance component) and three colour differences Cb , Cr and Cg that represent the difference between the colour intensity and the mean luminance of each image sample. Figure 2.10 shows the chroma components (red, green and blue) 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



Figure 2.10 Cr, Cg and Cb components

difference between the 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, $Cb + Cr + Cg$ is a constant and so only two of the three chroma components need to be stored or transmitted since the third component can always be calculated from the other two. In the YCbCr colour space, only the luma (Y) and blue and red chroma (Cb , Cr) are transmitted. YCbCr has an important advantage over RGB, that is 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 YCbCr 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 YCbCr 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 YCbCr colour space and vice versa are given in Equation 2.3 and Equation 2.4¹. Note that there is no need to specify a separate factor k_g (because $k_b + k_r + k_g = 1$) and that G can be extracted from the YCbCr representation by subtracting Cr and Cb from Y , demonstrating that it is not necessary to store or transmit a Cg component.

$$\begin{aligned}
 Y &= k_r R + (1 - k_b - k_r)G + k_b B \\
 Cb &= \frac{0.5}{1 - k_b}(B - Y) \\
 Cr &= \frac{0.5}{1 - k_r}(R - Y)
 \end{aligned} \tag{2.3}$$

$$\begin{aligned}
 R &= Y + \frac{1 - k_r}{0.5}Cr \\
 G &= Y - \frac{2k_b(1 - k_b)}{1 - k_b - k_r}Cb - \frac{2k_r(1 - k_r)}{1 - k_b - k_r}Cr \\
 B &= Y + \frac{1 - k_b}{0.5}Cb
 \end{aligned} \tag{2.4}$$

¹ Thanks to Gary Sullivan for suggesting the form of Equations 2.3 and 2.4

ITU-R recommendation BT.601 [1] defines $k_b = 0.114$ and $k_r = 0.299$. Substituting into the above equations gives the following widely-used conversion equations:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.564(B - Y) \\ Cr &= 0.713(R - Y) \end{aligned} \quad (2.5)$$

$$\begin{aligned} R &= Y + 1.402Cr \\ G &= Y - 0.344Cb - 0.714Cr \\ B &= Y + 1.772Cb \end{aligned} \quad (2.6)$$

2.4.3 YCbCr Sampling Formats

Figure 2.11 shows three sampling patterns for Y, Cb and Cr that are supported by MPEG-4 Visual and H.264. 4:4:4 sampling means that the three components (Y, Cb and Cr) have the same resolution and hence a sample of each component exists at every pixel position. The numbers indicate the relative sampling rate of each component in the *horizontal* direction, i.e. for every four luminance samples there are four Cb and four Cr samples. 4:4:4 sampling preserves the full fidelity of the chrominance components. In 4:2:2 sampling (sometimes referred to as YUY2), the chrominance components have the same vertical resolution as the luma but half the horizontal resolution (the numbers 4:2:2 mean that for every four luminance samples in the horizontal direction there are two Cb and two Cr samples). 4:2:2 video is used for high-quality colour reproduction.

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

Example

Image resolution: 720×576 pixels

Y resolution: 720×576 samples, each represented with eight bits

4:4:4 Cb, Cr resolution: 720×576 samples, each eight bits

Total number of bits: $720 \times 576 \times 8 \times 3 = 9\,953\,280$ bits

4:2:0 Cb, Cr resolution: 360×288 samples, each eight bits

Total number of bits: $(720 \times 576 \times 8) + (360 \times 288 \times 8 \times 2) = 4\,976\,640$ bits

The 4:2:0 version requires half as many bits as the 4:4:4 version.

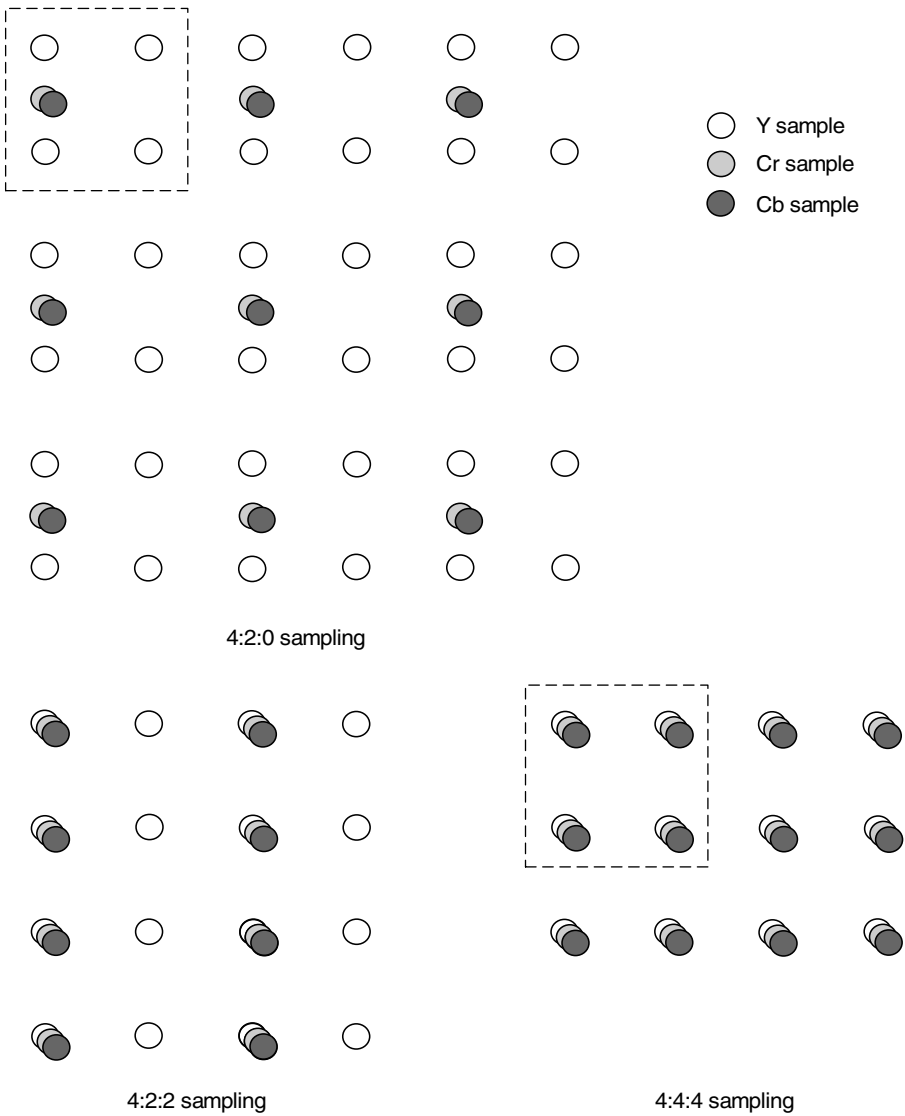


Figure 2.11 4:2:0, 4:2:2 and 4:4:4 sampling patterns (progressive)

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

In a 4:2:0 interlaced video sequence, the Y, Cb and Cr samples corresponding to a complete video frame are allocated to two fields. Figure 2.12 shows the method of allocating

Table 2.1 Video frame formats

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



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

Y, Cb and Cr samples to a pair of interlaced fields adopted in MPEG-4 Visual and H.264. It is clear from this figure that the total number of samples in a pair of fields is the same as the number of samples in an equivalent progressive frame.

2.5 VIDEO FORMATS

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

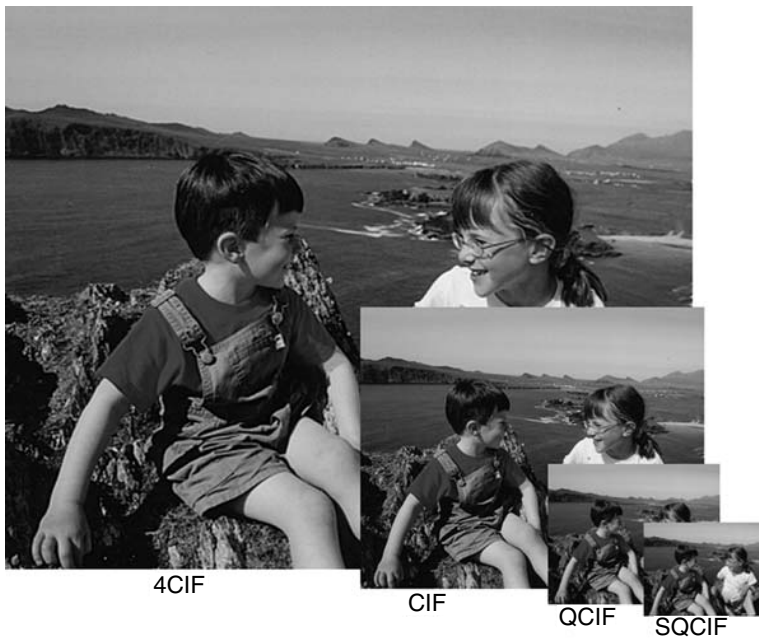


Figure 2.13 Video frame sampled at range of resolutions

are popular for videoconferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications where the display resolution and the bitrate are limited. Table 2.1 lists the number of bits required to represent one uncompressed frame in each format (assuming 4:2:0 sampling and 8 bits per luma and chroma sample).

A widely-used format for digitally coding video signals for television production is ITU-R Recommendation BT.601-5 [1] (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 13.5 MHz and the chrominance at 6.75 MHz to produce a 4:2:2 Y:Cb:Cr component signal. The parameters of the sampled digital signal depend on the video frame rate (30 Hz for an NTSC signal and 25 Hz for a PAL/SECAM signal) and are shown in Table 2.2. The higher 30 Hz frame rate of NTSC is compensated for by a lower spatial resolution so that the total bit rate is the same in each case (216 Mbps). The actual area shown 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 synchronisation and the active luminance signal is restricted to a range of 16 (black) to 235 (white).

2.6 QUALITY

In order to specify, evaluate and compare video communication systems it is necessary to determine the quality of the video images displayed to the viewer. Measuring visual quality is

Table 2.2 ITU-R BT.601-5 Parameters

	30 Hz frame rate	25 Hz 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

a difficult and often imprecise art because there are so many factors that can affect the results. Visual quality is inherently *subjective* and is influenced by many factors that make it difficult to obtain a completely accurate measure of quality. For example, a viewer’s opinion of visual quality can depend very much on the task at hand, such as passively watching a DVD movie, actively participating in a videoconference, communicating using sign language or trying to identify a person in a surveillance video scene. Measuring visual quality using *objective* criteria gives accurate, repeatable results but as yet there are no objective measurement systems that completely reproduce the subjective experience of a human observer watching a video display.

2.6.1 Subjective Quality Measurement

2.6.1.1 Factors Influencing Subjective Quality

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

Other important influences on perceived quality include visual attention (an observer perceives a scene by fixating on a sequence of points in the image rather than by taking in everything simultaneously) and the so-called ‘recency effect’ (our opinion of a visual sequence is more heavily influenced by recently-viewed material than older video material) [2, 3]. All of these factors make it very difficult to measure visual quality accurately and quantitatively.

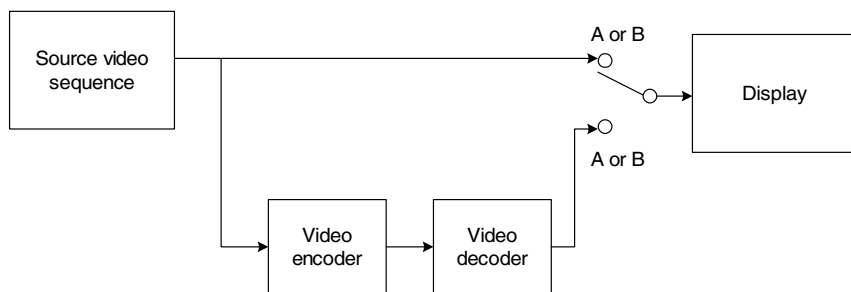


Figure 2.14 DSCQS testing system

2.6.1.2 ITU-R 500

Several test procedures for subjective quality evaluation are defined in ITU-R Recommendation BT.500-11 [4]. A commonly-used procedure from the standard is the Double Stimulus Continuous Quality Scale (DSCQS) method in which an assessor is presented with a pair of images or short video sequences A and B, one after the other, and is asked to give A and B a ‘quality score’ by marking on a continuous line with five intervals ranging from ‘Excellent’ to ‘Bad’. In a typical test session, the assessor is shown a series of pairs of sequences and is asked to grade each pair. Within each pair of sequences, one is an unimpaired “reference” sequence and the other is the same sequence, modified by a system or process under test. Figure 2.14 shows an experimental set-up appropriate for the testing of a video CODEC in which the original sequence is compared with the same sequence after encoding and decoding. The selection of which sequence is ‘A’ and which is ‘B’ is randomised.

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

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

2.6.2 Objective Quality Measurement

The complexity and cost of subjective quality measurement make it attractive to be able to measure quality automatically using an algorithm. Developers of video compression and video



Figure 2.15 PSNR examples: (a) original; (b) 30.6 dB; (c) 28.3 dB



Figure 2.16 Image with blurred background (PSNR = 27.7 dB)

processing systems rely heavily on so-called objective (algorithmic) quality measures. The most widely used measure is Peak Signal to Noise Ratio (PSNR) but the limitations of this metric have led to many efforts to develop more sophisticated measures that approximate the response of ‘real’ human observers.

2.6.2.1 PSNR

Peak Signal to Noise Ratio (PSNR) (Equation 2.7) is measured on a logarithmic scale and depends on the mean squared error (MSE) of between an original and an impaired image or video frame, relative to $(2^n - 1)^2$ (the square of the highest-possible signal value in the image, where n is the number of bits per image sample).

$$PSNR_{dB} = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \quad (2.7)$$

PSNR can be calculated easily and quickly and is therefore a very popular quality measure, widely used to compare the ‘quality’ of compressed and decompressed video images. Figure 2.15 shows a close-up of 3 images: the first image (a) is the original and (b) and (c) are degraded (blurred) versions of the original image. Image (b) has a measured PSNR of 30.6 dB whilst image (c) has a PSNR of 28.3 dB (reflecting the poorer image quality).

The PSNR measure suffers from a number of limitations. PSNR requires an unimpaired original image for comparison but this may not be available in every case and it may not be easy to verify that an 'original' image has perfect fidelity. PSNR does not correlate well with subjective video quality measures such as those defined in ITU-R 500. For a given image or image sequence, high PSNR usually indicates high quality and low PSNR usually indicates low quality. However, a particular value of PSNR does not necessarily equate to an 'absolute' subjective quality. For example, Figure 2.16 shows a distorted version of the original image from Figure 2.15 in which only the background of the image has been blurred. This image has a PSNR of 27.7 dB relative to the original. Most viewers would rate this image as significantly better than image (c) in Figure 2.15 because the face is clearer, contradicting the PSNR rating. This example shows that PSNR ratings do not necessarily correlate with 'true' subjective quality. In this case, a human observer gives a higher importance to the face region and so is particularly sensitive to distortion in this area.

2.6.2.2 Other Objective Quality Metrics

Because of the limitations of crude metrics such as PSNR, there has been a lot of work in recent years to try to develop a more sophisticated objective test that more closely approaches subjective test results. Many different approaches have been proposed [5, 6, 7] but none of these has emerged as a clear alternative to subjective tests. As yet there is no standardised, accurate system for objective ('automatic') quality measurement that is suitable for digitally coded video. In recognition of this, the ITU-T Video Quality Experts Group (VQEG) aim to develop standards for objective video quality evaluation [8]. The first step in this process was to test and compare potential models for objective evaluation. In March 2000, VQEG reported on the first round of tests in which ten competing systems were tested under identical conditions. Unfortunately, none of the ten proposals was considered suitable for standardisation and VQEG are completing a second round of evaluations in 2003. Unless there is a significant breakthrough in automatic quality assessment, the problem of accurate objective quality measurement is likely to remain for some time to come.

2.7 CONCLUSIONS

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

2.8 REFERENCES

1. Recommendation ITU-R BT.601-5, Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, ITU-T, 1995.

REFERENCES

2. N. Wade and M. Swanston, *Visual Perception: An Introduction*, 2nd edition, Psychology Press, London, 2001.
3. R. Aldridge, J. Davidoff, D. Hands, M. Ghanbari and D. E. Pearson, Recency effect in the subjective assessment of digitally coded television pictures, *Proc. Fifth International Conference on Image Processing and its Applications*, Heriot-Watt University, Edinburgh, UK, July 1995.
4. Recommendation ITU-T BT.500-11, Methodology for the subjective assessment of the quality of television pictures, ITU-T, 2002.
5. C. J. van den Branden Lambrecht and O. Verscheure, Perceptual quality measure using a spatio-temporal model of the Human Visual System, Digital Video Compression Algorithms and Technologies, *Proc. SPIE*, **2668**, San Jose, 1996.
6. H. Wu, Z. Yu, S. Winkler and T. Chen, Impairment metrics for MC/DPCM/DCT encoded digital video, *Proc. PCS01*, Seoul, April 2001.
7. K. T. Tan and M. Ghanbari, A multi-metric objective picture quality measurement model for MPEG video, *IEEE Trans. Circuits and Systems for Video Technology*, **10** (7), October 2000.
8. <http://www.vqeg.org/> (Video Quality Experts Group).

5

MPEG-4 Visual

5.1 INTRODUCTION

ISO/IEC Standard 14496 Part 2 [1] (MPEG-4 Visual) improves on the popular MPEG-2 standard both in terms of compression efficiency (better compression for the same visual quality) and flexibility (enabling a much wider range of applications). It achieves this in two main ways, by making use of more advanced compression algorithms and by providing an extensive set of ‘tools’ for coding and manipulating digital media. MPEG-4 Visual consists of a ‘core’ video encoder/decoder model together with a number of additional coding tools. The core model is based on the well-known hybrid DPCM/DCT coding model (see Chapter 3) and the basic function of the core is extended by tools supporting (among other things) enhanced compression efficiency, reliable transmission, coding of separate shapes or ‘objects’ in a visual scene, mesh-based compression and animation of face or body models.

It is unlikely that any single application would require all of the tools available in the MPEG-4 Visual framework and so the standard describes a series of *profiles*, recommended sets or groupings of tools for particular types of application. Examples of profiles include Simple (a minimal set of tools for low-complexity applications), Core and Main (with tools for coding multiple arbitrarily-shaped video objects), Advanced Real Time Simple (with tools for error-resilient transmission with low delay) and Advanced Simple (providing improved compression at the expense of increased complexity).

MPEG-4 Visual is embodied in ISO/IEC 14496-2, a highly detailed document running to over 500 pages. Version 1 was released in 1998 and further tools and profiles were added in two Amendments to the standard culminating in Version 2 in late 2001. More tools and profiles are planned for future Amendments or Versions but the ‘toolkit’ structure of MPEG-4 means that any later versions of 14496-2 should remain backwards compatible with Version 1.

This chapter is a guide to the tools and features of MPEG-4 Visual. Practical implementations of MPEG-4 Visual are based on one or more of the profiles defined in the standard and so this chapter is organised according to profiles. After an overview of the standard and its approach and features, the profiles for coding rectangular video frames are discussed (Simple,

Advanced Simple and Advanced Real-Time Simple profiles). These are by far the most popular profiles in use at the present time and so they are covered in some detail. Tools and profiles for coding of arbitrary-shaped objects are discussed next (the Core, Main and related profiles), followed by profiles for scalable coding, still texture coding and high-quality ('studio') coding of video.

In addition to tools for coding of 'natural' (real-world) video material, MPEG-4 Visual defines a set of profiles for coding of 'synthetic' (computer-generated) visual objects such as 2D and 3D meshes and animated face and body models. The focus of this book is very much on coding of natural video and so these profiles are introduced only briefly. Coding tools in the MPEG-4 Visual standard that are not included in any Profile (such as Overlapped Block Motion Compensation, OBMC) are (perhaps contentiously!) not covered in this chapter.

5.2 OVERVIEW OF MPEG-4 VISUAL (NATURAL VIDEO CODING)

5.2.1 Features

MPEG-4 Visual attempts to satisfy the requirements of a wide range of visual communication applications through a toolkit-based approach to coding of visual information. Some of the key features that distinguish MPEG-4 Visual from previous visual coding standards include:

- Efficient compression of progressive and interlaced 'natural' video sequences (compression of sequences of rectangular video frames). The core compression tools are based on the ITU-T H.263 standard and can out-perform MPEG-1 and MPEG-2 video compression. Optional additional tools further improve compression efficiency.
- Coding of video objects (irregular-shaped regions of a video scene). This is a new concept for standard-based video coding and enables (for example) independent coding of foreground and background objects in a video scene.
- Support for effective transmission over practical networks. Error resilience tools help a decoder to recover from transmission errors and maintain a successful video connection in an error-prone network environment and scalable coding tools can help to support flexible transmission at a range of coded bitrates.
- Coding of still 'texture' (image data). This means, for example, that still images can be coded and transmitted within the same framework as moving video sequences. Texture coding tools may also be useful in conjunction with animation-based rendering.
- Coding of animated visual objects such as 2D and 3D polygonal meshes, animated faces and animated human bodies.
- Coding for specialist applications such as 'studio' quality video. In this type of application, visual quality is perhaps more important than high compression.

5.2.2 Tools, Objects, Profiles and Levels

MPEG-4 Visual provides its coding functions through a combination of *tools*, *objects* and *profiles*. A *tool* is a subset of coding functions to support a specific feature (for example, basic

Table 5.1 MPEG-4 Visual profiles for coding natural video

MPEG-4 Visual profile	Main features
Simple	Low-complexity coding of rectangular video frames
Advanced Simple	Coding rectangular frames with improved efficiency and support for interlaced video
Advanced Real-Time Simple	Coding rectangular frames for real-time streaming
Core	Basic coding of arbitrary-shaped video objects
Main	Feature-rich coding of video objects
Advanced Coding Efficiency	Highly efficient coding of video objects
N-Bit	Coding of video objects with sample resolutions other than 8 bits
Simple Scalable	Scalable coding of rectangular video frames
Fine Granular Scalability	Advanced scalable coding of rectangular frames
Core Scalable	Scalable coding of video objects
Scalable Texture	Scalable coding of still texture
Advanced Scalable Texture	Scalable still texture with improved efficiency and object-based features
Advanced Core	Combines features of Simple, Core and Advanced Scalable Texture Profiles
Simple Studio	Object-based coding of high quality video sequences
Core Studio	Object-based coding of high quality video with improved compression efficiency.

Table 5.2 MPEG-4 Visual profiles for coding synthetic or hybrid video

MPEG-4 Visual profile	Main features
Basic Animated Texture	2D mesh coding with still texture
Simple Face Animation	Animated human face models
Simple Face and Body Animation	Animated face and body models
Hybrid	Combines features of Simple, Core, Basic Animated Texture and Simple Face Animation profiles

video coding, interlaced video, coding object shapes, etc.). An *object* is a video element (e.g. a sequence of rectangular frames, a sequence of arbitrary-shaped regions, a still image) that is coded using one or more tools. For example, a simple video object is coded using a limited subset of tools for rectangular video frame sequences, a core video object is coded using tools for arbitrarily-shaped objects and so on. A *profile* is a set of object types that a CODEC is expected to be capable of handling.

The MPEG-4 Visual profiles for coding ‘natural’ video scenes are listed in Table 5.1 and these range from Simple Profile (coding of rectangular video frames) through profiles for arbitrary-shaped and scalable object coding to profiles for coding of studio-quality video. Table 5.2 lists the profiles for coding ‘synthetic’ video (animated meshes or face/body models) and the hybrid profile (incorporates features from synthetic and natural video coding). These profiles are not (at present) used for natural video compression and so are not covered in detail in this book.

Profile	Object types															
	Simple	Advanced Simple	Advanced Real-Time Simple	Core	Main	Advanced Coding Efficiency	N-bit	Simple Scalable	Fine Granular Scalability	Core Scalable	Scalable Texture	Advanced Scalable Texture	Simple Studio	Core Studio	Simple Face Animation	Simple Face and Body Animation
Simple	✓															
Advanced Simple	✓	✓														
Advanced Real-Time Simple	✓		✓													
Core	✓			✓												
Advanced Core	✓			✓								✓				
Main	✓			✓	✓						✓					
Advanced Coding Efficiency	✓			✓		✓										
N-bit	✓			✓			✓									
Simple Scalable	✓							✓								
Fine Granular Scalability	✓	✓							✓							
Core Scalable	✓			✓				✓		✓						
Scalable Texture											✓					
Advanced Scalable Texture												✓				
Simple Studio													✓			
Core Studio													✓	✓		
Basic Animated Texture											✓				✓	
Simple Face Animation															✓	
Simple FBA																✓
Hybrid	✓			✓							✓			✓		✓

Figure 5.1 MPEG-4 Visual profiles and objects

Figure 5.1 lists each of the MPEG-4 Visual profiles (left-hand column) and visual object types (top row). The table entries indicate which object types are contained within each profile. For example, a CODEC compatible with Simple Profile must be capable of coding and decoding Simple objects and a Core Profile CODEC must be capable of coding and decoding Simple *and* Core objects.

Profiles are an important mechanism for encouraging interoperability between CODECs from different manufacturers. The MPEG-4 Visual standard describes a diverse range of coding tools and it is unlikely that any commercial CODEC would require the implementation of all the tools. Instead, a CODEC designer chooses a profile that contains adequate tools for the target application. For example, a basic CODEC implemented on a low-power processor may use Simple profile, a CODEC for streaming video applications may choose Advanced Real Time Simple and so on. To date, some profiles have had more of an impact on the marketplace than others. The Simple and Advanced Simple profiles are particularly popular with manufacturers and users whereas the profiles for the coding of arbitrary-shaped objects have had very limited commercial impact (see Chapter 8 for further discussion of the commercial impact of MPEG-4 Profiles).

Profiles define a subset of coding tools and *Levels* define constraints on the parameters of the bitstream. Table 5.3 lists the Levels for the popular Simple-based profiles (Simple,

Table 5.3 Levels for Simple-based profiles

Profile	Level	Typical resolution	Max. bitrate	Max. objects
Simple	L0	176 × 144	64 kbps	1 simple
	L1	176 × 144	64 kbps	4 simple
	L2	352 × 288	128 kbps	4 simple
	L3	352 × 288	384 kbps	4 simple
Advanced Simple (AS)	L0	176 × 144	128 kbps	1 AS or simple
	L1	176 × 144	128 kbps	4 AS or simple
	L2	352 × 288	384 kbps	4 AS or simple
	L3	352 × 288	768 kbps	4 AS or simple
	L4	352 × 576	3 Mbps	4 AS or simple
	L5	720 × 576	8 Mbps	4 AS or simple
Advanced Real-Time Simple (ARTS)	L1	176 × 144	64 kbps	4 ARTS or simple
	L2	352 × 288	128 kbps	4 ARTS or simple
	L3	352 × 288	384 kbps	4 ARTS or simple
	L4	352 × 288	2 Mbps	16 ARTS or simple

Advanced Simple and Advanced Real Time Simple). Each Level places constraints on the maximum performance required to decode an MPEG-4 coded sequence. For example, a multimedia terminal with limited processing capabilities and a small amount of memory may only support Simple Profile @ Level 0 bitstream decoding. The Level definitions place restrictions on the amount of buffer memory, the decoded frame size and processing rate (in macroblocks per second) and the number of video objects (one in this case, a single rectangular frame). A terminal that can cope with these parameters is guaranteed to be capable of successfully decoding any conforming Simple Profile @ Level 0 bitstream. Higher Levels of Simple Profile require a decoder to handle up to four Simple Profile video objects (for example, up to four rectangular objects covering the QCIF or CIF display resolution).

5.2.3 Video Objects

One of the key contributions of MPEG-4 Visual is a move away from the ‘traditional’ view of a video sequence as being merely a collection of rectangular frames of video. Instead, MPEG-4 Visual treats a video sequence as a collection of one or more *video objects*. MPEG-4 Visual defines a video object as a flexible ‘entity that a user is allowed to access (seek, browse) and manipulate (cut and paste)’ [1]. A video object (VO) is an area of the video scene that may occupy an arbitrarily-shaped region and may exist for an arbitrary length of time. An instance of a VO at a particular point in time is a *video object plane* (VOP).

This definition encompasses the traditional approach of coding complete frames, in which each VOP is a single frame of video and a sequence of frames forms a VO (for example, Figure 5.2 shows a VO consisting of three rectangular VOPs). However, the introduction of the VO concept allows more flexible options for coding video. Figure 5.3 shows a VO that consists of three irregular-shaped VOPs, each one existing within a frame and each coded separately (*object-based coding*).

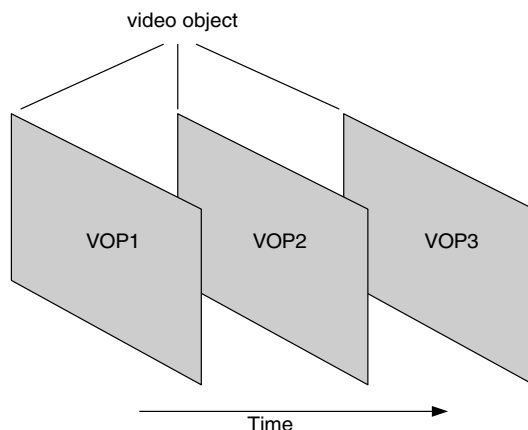


Figure 5.2 VOPs and VO (rectangular)

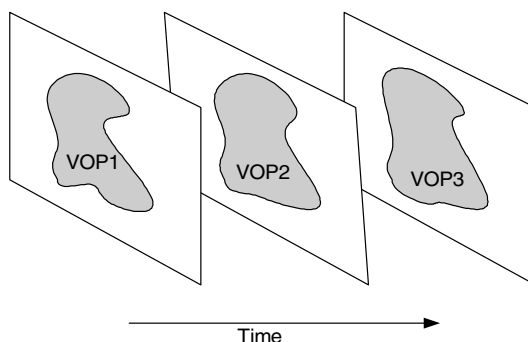


Figure 5.3 VOPs and VO (arbitrary shape)

A video scene (e.g. Figure 5.4) may be made up of a background object (VO3 in this example) and a number of separate foreground objects (VO1, VO2). This approach is potentially much more flexible than the fixed, rectangular frame structure of earlier standards. The separate objects may be coded with different visual qualities and temporal resolutions to reflect their ‘importance’ to the final scene, objects from multiple sources (including synthetic and ‘natural’ objects) may be combined in a single scene and the composition and behaviour of the scene may be manipulated by an end-user in highly interactive applications. Figure 5.5 shows a new video scene formed by adding VO1 from Figure 5.4, a new VO2 and a new background VO. Each object is coded separately using MPEG-4 Visual (the compositing of visual and audio objects is assumed to be handled separately, for example by MPEG-4 Systems [2]).

5.3 CODING RECTANGULAR FRAMES

Notwithstanding the potential flexibility offered by object-based coding, the most popular application of MPEG-4 Visual is to encode complete frames of video. The tools required

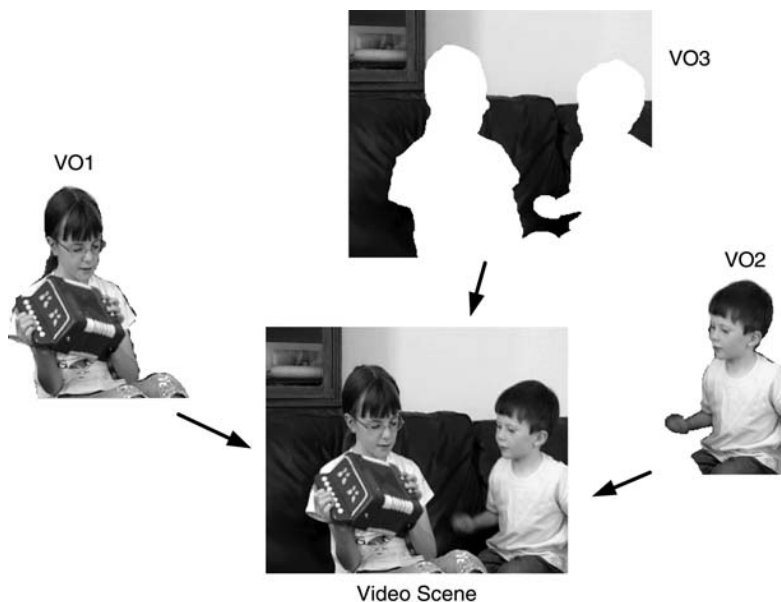


Figure 5.4 Video scene consisting of three VOs



Figure 5.5 Video scene composed of VOs from separate sources

to handle rectangular VOPs (typically complete video frames) are grouped together in the so-called *simple* profiles. The tools and objects for coding rectangular frames are shown in Figure 5.6. The basic tools are similar to those adopted by previous video coding standards, DCT-based coding of macroblocks with motion compensated prediction. The Simple profile is based around the well-known hybrid DPCM/DCT model (see Chapter 3, Section 3.6) with

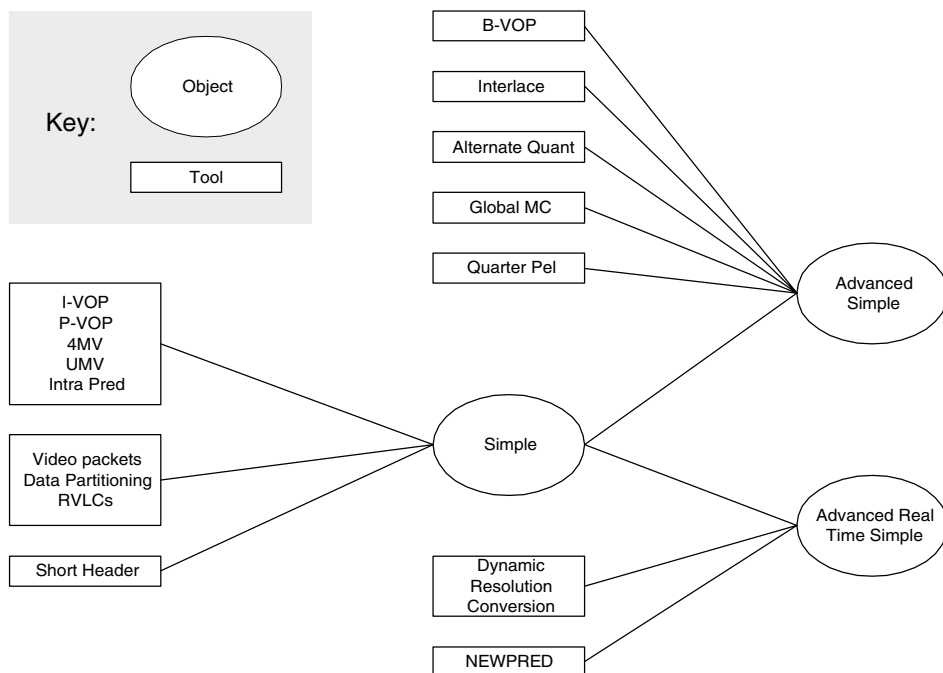


Figure 5.6 Tools and objects for coding rectangular frames

additional tools to improve coding efficiency and transmission efficiency. Because of the widespread popularity of Simple profile, enhanced profiles for rectangular VOPs have been developed. The Advanced Simple profile improves further coding efficiency and adds support for interlaced video and the Advanced Real-Time Simple profile adds tools that are useful for real-time video streaming applications.

5.3.1 Input and output video format

The input to an MPEG-4 Visual encoder and the output of a decoder is a video sequence in 4:2:0, 4:2:2 or 4:4:4 progressive or interlaced format (see Chapter 2). MPEG-4 Visual uses the sampling arrangement shown in Figure 2.11 for progressive sampled frames and the method shown in Figure 2.12 for allocating luma and chroma samples to each pair of fields in an interlaced sequence.

5.3.2 The Simple Profile

A CODEC that is compatible with Simple Profile should be capable of encoding and decoding Simple Video Objects using the following tools:

- I-VOP (Intra-coded rectangular VOP, progressive video format);
- P-VOP (Inter-coded rectangular VOP, progressive video format);

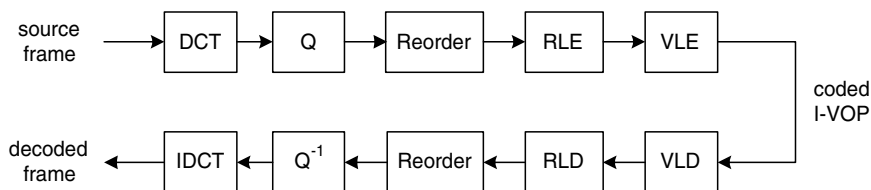


Figure 5.7 I-VOP encoding and decoding stages

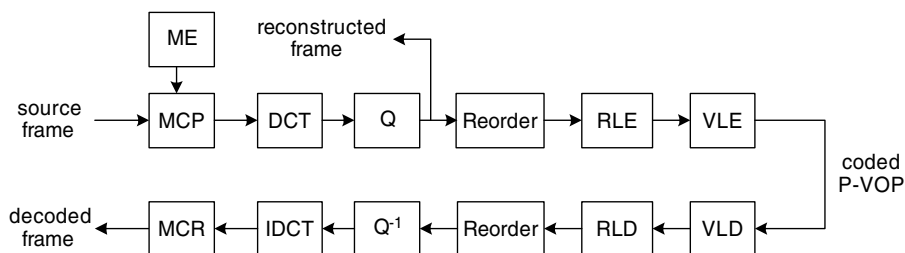


Figure 5.8 P-VOP encoding and decoding stages

- short header (mode for compatibility with H.263 CODECs);
- compression efficiency tools (four motion vectors per macroblock, unrestricted motion vectors, Intra prediction);
- transmission efficiency tools (video packets, Data Partitioning, Reversible Variable Length Codes).

5.3.2.1 The Very Low Bit Rate Video Core

The Simple Profile of MPEG-4 Visual uses a CODEC model known as the Very Low Bit Rate Video (VLBV) Core (the hybrid DPCM/DCT model described in Chapter 3). In common with other standards, the architecture of the encoder and decoder are not specified in MPEG-4 Visual but a practical implementation will require to carry out the functions shown in Figure 5.7 (coding of Intra VOPs) and Figure 5.8 (coding of Inter VOPs). The basic tools required to encode and decode rectangular I-VOPs and P-VOPs are described in the next section (Section 3.6 of Chapter 3 provides a more detailed ‘walk-through’ of the encoding and decoding process). The tools in the VLBV Core are based on the H.263 standard and the ‘short header’ mode enables direct compatibility (at the frame level) between an MPEG-4 Simple Profile CODEC and an H.263 Baseline CODEC.

5.3.2.2 Basic coding tools

I-VOP

A rectangular I-VOP is a frame of video encoded in Intra mode (without prediction from any other coded VOP). The encoding and decoding stages are shown in Figure 5.7.

Table 5.4 Values of *dc_scaler* parameter depending on QP range

Block type	$QP \leq 4$	$5 \leq QP \leq 8$	$9 \leq QP \leq 24$	$25 \leq QP$
Luma	8	$2 \times QP$	$QP + 8$	$(2 \times QP) - 16$
Chroma	8	$(QP + 13)/2$	$(QP + 13)/2$	$QP - 6$

DCT and IDCT: Blocks of luma and chroma samples are transformed using an 8×8 Forward DCT during encoding and an 8×8 Inverse DCT during decoding (see Section 3.4).

Quantisation: The MPEG-4 Visual standard specifies the method of rescaling ('inverse quantising') quantised transform coefficients in a decoder. Rescaling is controlled by a quantiser scale parameter, QP , which can take values from 1 to 31 (larger values of QP produce a larger quantiser step size and therefore higher compression and distortion). Two methods of rescaling are described in the standard: 'method 2' (basic method) and 'method 1' (more flexible but also more complex). Method 2 inverse quantisation operates as follows. The DC coefficient in an Intra-coded macroblock is rescaled by:

$$DC = DC_Q \cdot dc_scaler \quad (5.1)$$

DC_Q is the quantised coefficient, DC is the rescaled coefficient and *dc_scaler* is a parameter defined in the standard. In short header mode (see below), *dc_scaler* is 8 (i.e. all Intra DC coefficients are rescaled by a factor of 8), otherwise *dc_scaler* is calculated according to the value of QP (Table 5.4). All other transform coefficients (including AC and Inter DC) are rescaled as follows:

$$\begin{aligned} |F| &= QP \cdot (2 \cdot |F_Q| + 1) && \text{(if } QP \text{ is odd and } F_Q \neq 0) \\ |F| &= QP \cdot (2 \cdot |F_Q| + 1) - 1 && \text{(if } QP \text{ is even and } F_Q \neq 0) \\ F &= 0 && \text{(if } F_Q = 0) \end{aligned} \quad (5.2)$$

F_Q is the quantised coefficient and F is the rescaled coefficient. The sign of F is made the same as the sign of F_Q . Forward quantisation is not defined by the standard.

Zig-zag scan: Quantised DCT coefficients are reordered in a zig-zag scan prior to encoding (see Section 3.4).

Last-Run-Level coding: The array of reordered coefficients corresponding to each block is encoded to represent the zero coefficients efficiently. Each nonzero coefficient is encoded as a triplet of (last, run, level), where 'last' indicates whether this is the final nonzero coefficient in the block, 'run' signals the number of preceding zero coefficients and 'level' indicates the coefficient sign and magnitude.

Entropy coding: Header information and (last, run, level) triplets (see Section 3.5) are represented by variable-length codes (VLCs). These codes are similar to Huffman codes and are defined in the standard, based on pre-calculated coefficient probabilities

A coded I-VOP consists of a VOP header, optional video packet headers and coded macroblocks. Each macroblock is coded with a header (defining the macroblock type, identifying which blocks in the macroblock contain coded coefficients, signalling changes in quantisation parameter, etc.) followed by coded coefficients for each 8×8 block.

In the decoder, the sequence of VLCs are decoded to extract the quantised transform coefficients which are re-scaled and transformed by an 8×8 IDCT to reconstruct the decoded I-VOP (Figure 5.7).

P-VOP

A P-VOP is coded with Inter prediction from a previously encoded I- or P-VOP (a reference VOP). The encoding and decoding stages are shown in Figure 5.8.

Motion estimation and compensation: The basic motion compensation scheme is block-based compensation of 16×16 pixel macroblocks (see Chapter 3). The offset between the current macroblock and the compensation region in the reference picture (the motion vector) may have half-pixel resolution. Predicted samples at sub-pixel positions are calculated using bilinear interpolation between samples at integer-pixel positions. The method of motion estimation (choosing the ‘best’ motion vector) is left to the designer’s discretion. The matching region (or prediction) is subtracted from the current macroblock to produce a residual macroblock (Motion-Compensated Prediction, MCP in Figure 5.8).

After motion compensation, the residual data is transformed with the DCT, quantised, reordered, run-level coded and entropy coded. The quantised residual is rescaled and inverse transformed in the encoder in order to reconstruct a local copy of the decoded MB (for further motion compensated prediction). A coded P-VOP consists of VOP header, optional video packet headers and coded macroblocks each containing a header (this time including differentially-encoded motion vectors) and coded residual coefficients for every 8×8 block.

The decoder forms the same motion-compensated prediction based on the received motion vector and its own local copy of the reference VOP. The decoded residual data is added to the prediction to reconstruct a decoded macroblock (Motion-Compensated Reconstruction, MCR in Figure 5.8).

Macroblocks within a P-VOP may be coded in Inter mode (with motion compensated prediction from the reference VOP) or Intra mode (no motion compensated prediction). Inter mode will normally give the best coding efficiency but Intra mode may be useful in regions where there is not a good match in a previous VOP, such as a newly-uncovered region.

Short Header

The ‘short header’ tool provides compatibility between MPEG-4 Visual and the ITU-T H.263 video coding standard. An I- or P-VOP encoded in ‘short header’ mode has identical syntax to an I-picture or P-picture coded in the baseline mode of H.263. This means that an MPEG-4 I-VOP or P-VOP should be decodeable by an H.263 decoder and vice versa.

In short header mode, the macroblocks within a VOP are organised in Groups of Blocks (GOBs), each consisting of one or more complete rows of macroblocks. Each GOB may (optionally) start with a resynchronisation marker (a fixed-length binary code that enables a decoder to resynchronise when an error is encountered, see Section 5.3.2.4).

5.3.2.3 Coding Efficiency Tools

The following tools, part of the Simple profile, can improve compression efficiency. They are only used when short header mode is not enabled.

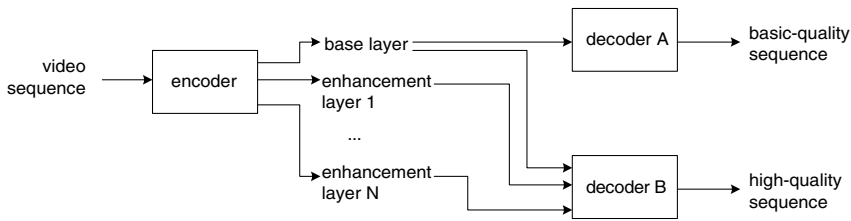


Figure 5.57 Scalable coding: general concept

5.5 SCALABLE VIDEO CODING

Scalable encoding of video data enables a decoder to decode selectively only part of the coded bitstream. The coded stream is arranged in a number of *layers*, including a ‘base’ layer and one or more ‘enhancement’ layers (Figure 5.57). In this figure, decoder A receives only the base layer and can decode a ‘basic’ quality version of the video scene, whereas decoder B receives all layers and decodes a high quality version of the scene. This has a number of applications, for example, a low-complexity decoder may only be capable of decoding the base layer; a low-rate bitstream may be extracted for transmission over a network segment with limited capacity; and an error-sensitive base layer may be transmitted with higher priority than enhancement layers.

MPEG-4 Visual supports a number of scalable coding modes. *Spatial scalability* enables a (rectangular) VOP to be coded at a hierarchy of spatial resolutions. Decoding the base layer produces a low-resolution version of the VOP and decoding successive enhancement layers produces a progressively higher-resolution image. *Temporal scalability* provides a low frame-rate base layer and enhancement layer(s) that build up to a higher frame rate. The standard also supports *quality scalability*, in which the enhancement layers improve the visual quality of the VOP and *complexity scalability*, in which the successive layers are progressively more complex to decode. *Fine Grain Scalability* (FGS) enables the quality of the sequence to be increased in small steps. An application for FGS is streaming video across a network connection, in which it may be useful to scale the coded video stream to match the available bit rate as closely as possible.

5.5.1 Spatial Scalability

The base layer contains a reduced-resolution version of each coded frame. Decoding the base layer alone produces a low-resolution output sequence and decoding the base layer with enhancement layer(s) produces a higher-resolution output. The following steps are required to *encode* a video sequence into two spatial layers:

1. Subsample each input video frame (Figure 5.58) (or video object) horizontally and vertically (Figure 5.59).
2. Encode the reduced-resolution frame to form the base layer.
3. Decode the base layer and up-sample to the original resolution to form a prediction frame (Figure 5.60).
4. Subtract the full-resolution frame from this prediction frame (Figure 5.61).
5. Encode the difference (residual) to form the enhancement layer.



Figure 5.58 Original video frame



Figure 5.59 Sub-sampled frame to be encoded as base layer



Figure 5.60 Base layer frame (decoded and upsampled)



Figure 5.61 Residual to be encoded as enhancement layer

A single-layer decoder decodes only the base layer to produce a reduced-resolution output sequence. A two-layer decoder can reconstruct a full-resolution sequence as follows:

1. Decode the base layer and up-sample to the original resolution.
2. Decode the enhancement layer.
3. Add the decoded residual from the enhancement layer to the decoded base layer to form the output frame.

An **I-VOP** in an enhancement layer is encoded without any spatial prediction, i.e. as a complete frame or object at the enhancement resolution. In an enhancement layer **P-VOP**, the decoded, up-sampled base layer VOP (at the same position in time) is used as a prediction *without* any motion compensation. The difference between this prediction and the input frame is encoded using the texture coding tools, i.e. no motion vectors are transmitted for an enhancement P-VOP. An enhancement layer **B-VOP** is predicted from two directions. The backward prediction is formed by the decoded, up-sampled base layer VOP (at the same position in time), without any motion compensation (and hence without any MVs). The forward prediction is formed by the previous VOP in the enhancement layer (even if this is itself a B-VOP), with motion-compensated prediction (and hence MVs).

If the VOP has arbitrary (binary) shape, a base layer and enhancement layer BAB is required for each MB. The base layer BAB is encoded as usual, based on the shape and size of the base layer object. A BAB in a P-VOP enhancement layer is coded using prediction from an up-sampled version of the base layer BAB. A BAB in a B-VOP enhancement layer may be coded in the same way, or using forward prediction from the previous enhancement VOP (as described in Section 5.4.1.1).

5.5.2 Temporal Scalability

The base layer of a temporal scalable sequence is encoded at a low video frame rate and a temporal enhancement layer consists of I-, P- and/or B-VOPs that can be decoded together with the base layer to provide an increased video frame rate. Enhancement layer VOPs are predicted using motion-compensated prediction according to the following rules.

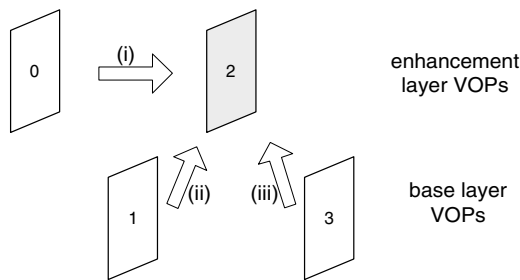


Figure 5.62 Temporal enhancement P-VOP prediction options

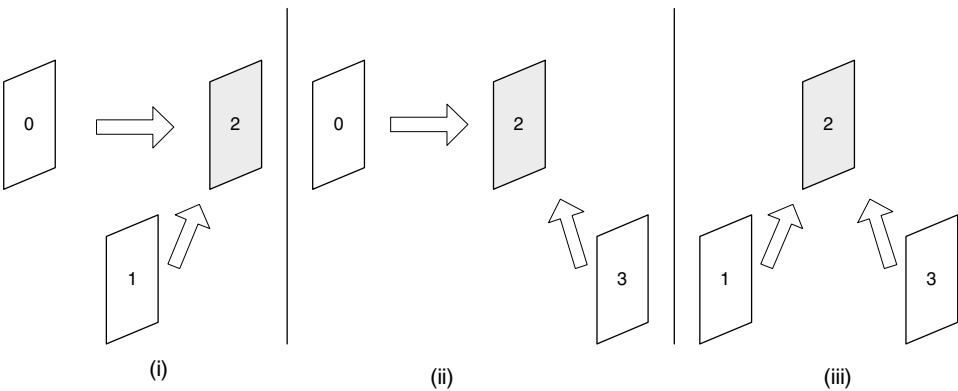


Figure 5.63 Temporal enhancement B-VOP prediction options

An enhancement *I-VOP* is encoded without any prediction. An enhancement *P-VOP* is predicted from (i) the previous enhancement VOP, (ii) the previous base layer VOP or (iii) the next base layer VOP (Figure 5.62). An enhancement *B-VOP* is predicted from (i) the previous enhancement and previous base layer VOPs, (ii) the previous enhancement and next base layer VOPs or (iii) the previous and next base layer VOPs (Figure 5.63).

5.5.3 Fine Granular Scalability

Fine Granular Scalability (FGS) [5] is a method of encoding a sequence as a base layer and enhancement layer. The enhancement layer can be truncated during or after encoding (reducing the bitrate and the decoded quality) to give highly flexible control over the transmitted bitrate. FGS may be useful for video streaming applications, in which the available transmission bandwidth may not be known in advance. In a typical scenario, a sequence is coded as a base layer and a high-quality enhancement layer. Upon receiving a request to send the sequence at a particular bitrate, the streaming server transmits the base layer and a truncated version of the enhancement layer. The amount of truncation is chosen to match the available transmission bitrate, hence maximising the quality of the decoded sequence without the need to re-encode the video clip.

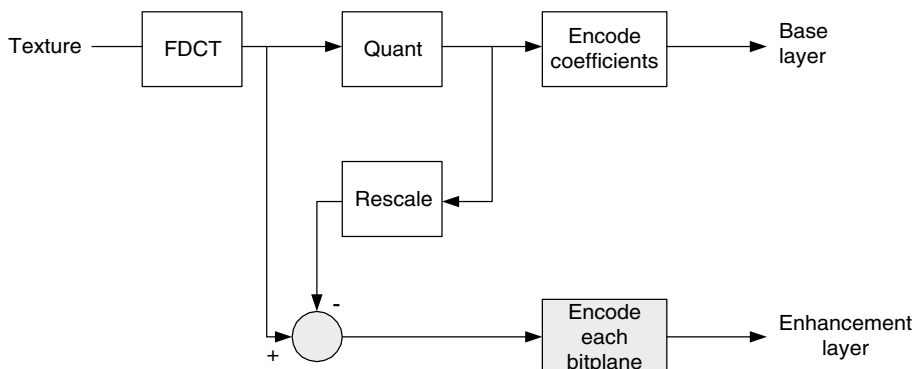


Figure 5.64 FGS encoder block diagram (simplified)

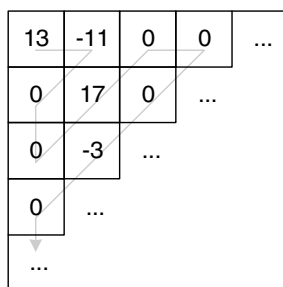


Figure 5.65 Block of residual coefficients (top-left corner)

Encoding

Figure 5.64 shows a simplified block diagram of an FGS encoder (motion compensation is not shown). In the Base Layer, the texture (after motion compensation) is transformed with the forward DCT, quantised and encoded. The quantised coefficients are re-scaled ('inverse quantised') and these re-scaled coefficients are subtracted from the unquantised DCT coefficients to give a set of difference coefficients. The difference coefficients for each block are encoded as a series of *bitplanes*. First, the residual coefficients are reordered using a zigzag scan. The highest-order bits of each coefficient (zeros or ones) are encoded first (the MS bit-plane) followed by the next highest-order bits and so on until the LS bits have been encoded.

Example

A block of residual coefficients is shown in Figure 5.65 (coefficients not shown are zero). The coefficients are reordered in a zigzag scan to produce the following list:

$$+13, -11, 0, 0, +17, 0, 0, 0, -3, 0, 0, \dots$$

The bitplanes corresponding to the magnitude of each residual coefficient are shown in Table 5.6. In this case, the highest plane containing nonzero bits is plane 4 (because the highest magnitude is 17).

Table 5.6 Residual coefficient bitplanes (magnitude)

Value	+13	-11	0	0	+17	0	0	0	-3	0...
Plane 4 (MSB)	0	0	0	0	1	0	0	0	0	0...
Plane 3	1	1	0	0	0	0	0	0	0	0...
Plane 2	1	1	0	0	0	0	0	0	0	0...
Plane 1	0	0	0	0	0	0	0	0	1	0...
Plane 0 (LSB)	1	1	0	0	1	0	0	0	1	0...

Table 5.7 Encoded values

Plane	Encoded values
4	(4, EOP) (+)
3	(0) (+) (0, EOP) (-)
2	(0, EOP)
1	(1) (6, EOP) (-)
0	(0) (0) (2) (3, EOP)

Each bitplane contains a series of zeros and ones. The ones are encoded as (run, EOP) where 'EOP' indicates 'end of bitplane' and each (run, EOP) pair is transmitted as a variable-length code. Whenever the MS bit of a coefficient is encoded, it is immediately followed in the bitstream by a sign bit. Table 5.7 lists the encoded values for each bitplane. Bitplane 4 contains four zeros, followed by a 1. This is the last nonzero bit and so is encoded as (4, EOP). This also the MS bit of the coefficient '+17' and so the sign of this coefficient is encoded.

This example illustrates the processing of one block. The encoding procedure for a complete frame is as follows:

1. Find the highest bit position of any difference coefficient in the frame (the MSB).
2. Encode each bitplane as described above, starting with the plane containing the MSB.

Each complete encoded bitplane is preceded by a start code, making it straightforward to truncate the bitstream by sending only a limited number of encoded bitplanes.

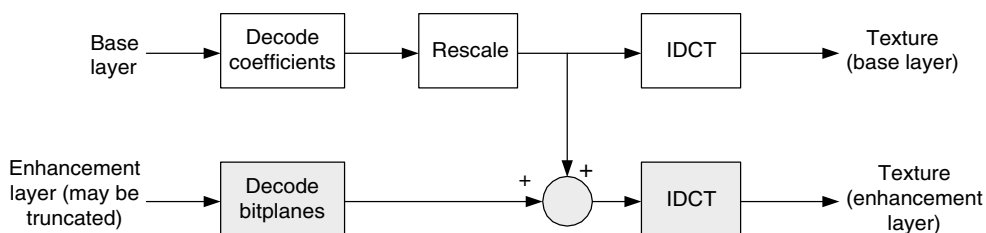
Decoding

The decoder decodes the base layer and enhancement layer (which may be truncated). The difference coefficients are reconstructed from the decoded bitplanes, added to the base layer coefficients and inverse transformed to produce the decoded enhancement sequence (Figure 5.66).

If the enhancement layer has been truncated, then the accuracy of the difference coefficients is reduced. For example, assume that the enhancement layer described in the above example is truncated after bitplane 3. The MS bits (and the sign) of the first three nonzero coefficients are decoded (Table 5.8); if the remaining (undecoded) bitplanes are filled with

Table 5.8 Decoded values (truncated after plane 3)

Plane 4 (MSB)	0	0	0	0	1	0	0	0	0	0...
Plane 3	1	1	0	0	0	0	0	0	0	0...
Plane 2	0	0	0	0	0	0	0	0	0	0...
Plane 1	0	0	0	0	0	0	0	0	0	0...
Plane 0 (LSB)	0	0	0	0	0	0	0	0	0	0...
Decoded value	+8	-8	0	0	+16	0	0	0	0	0...

**Figure 5.66** FGS decoder block diagram (simplified)

zeros then the list of output values becomes:

+8, -8, 0, 0, +16, 0 . . .

Optional enhancements to FGS coding include selective enhancement (in which bit planes of selected MBs are bit-shifted up prior to encoding, in order to give them a higher priority and a higher probability of being included in a truncated bitstream) and frequency weighting (in which visually-significant low frequency DCT coefficients are shifted up prior to encoding, again in order to give them higher priority in a truncated bitstream).

5.5.4 The Simple Scalable Profile

The Simple Scalable profile supports Simple and Simple Scalable objects. The Simple Scalable object contains the following tools:

- I-VOP, P-VOP, 4MV, unrestricted MV and Intra Prediction;
- Video packets, Data Partitioning and Reversible VLCs;
- B-VOP;
- Rectangular Temporal Scalability (1 enhancement layer) (Section 5.5.2);
- Rectangular Spatial Scalability (1 enhancement layer) (Section 5.5.1).

The last two tools support scalable coding of rectangular VOs.

5.5.5 The Core Scalable Profile

The Core Scalable profile includes Simple, Simple Scalable and Core objects, plus the Core Scalable object which features the following tools, in each case with up to two enhancement layers per object:

- Rectangular Temporal Scalability (Section 5.5.2);
- Rectangular Spatial Scalability (Section 5.5.1);
- Object-based Spatial Scalability (Section 5.5.1).

5.5.6 The Fine Granular Scalability Profile

The FGS profile includes Simple and Advanced Simple objects plus the FGS object which includes these tools:

- B-VOP, Interlace and Alternate Quantiser tools;
- FGS Spatial Scalability;
- FGS Temporal Scalability.

FGS ‘Spatial Scalability’ uses the encoding and decoding techniques described in Section 5.5.3 to encode each frame as a base layer and an FGS enhancement layer. FGS ‘Temporal Scalability’ combines FGS (Section 5.5.3) with temporal scalability (Section 5.5.2). An enhancement-layer frame is encoded using forward or bidirectional prediction from *base layer* frame(s) only. The DCT coefficients of the enhancement-layer frame are encoded in bitplanes using the FGS technique.

5.6 TEXTURE CODING

The applications targeted by the developers of MPEG4 include scenarios where it is necessary to transmit still texture (i.e. still images). Whilst block transforms such as the DCT are widely considered to be the best practical solution for motion-compensated video coding, the Discrete Wavelet Transform (DWT) is particularly effective for coding still images (see Chapter 3) and MPEG-4 Visual uses the DWT as the basis for tools to compress still texture. Applications include the coding of rectangular texture objects (such as complete image frames), coding of arbitrary-shaped texture regions and coding of texture to be mapped onto animated 2D or 3D meshes (see Section 5.8).

The basic structure of a still texture encoder is shown in Figure 5.68. A 2D DWT is applied to the texture object, producing a DC component (low-frequency subband) and a number of AC (high-frequency) subbands (see Chapter 3). The DC subband is quantised, predictively encoded (using a form of DPCM) and entropy encoded using an arithmetic encoder. The AC subbands are quantised and reordered (‘scanned’), zero-tree encoded and entropy encoded.

Discrete Wavelet Transform

The DWT adopted for MPEG-4 Still Texture coding is the Daubechies (9,3)-tap biorthogonal filter [6]. This is essentially a matched pair of filters, one low pass (with three filter coefficients or ‘taps’) and one high pass (with nine filter taps).

Quantisation

The DC subband is quantised using a scalar quantiser (see Chapter 3). The AC subbands may be quantised in one of three ways:

6

H.264/MPEG4 Part 10

6.1 INTRODUCTION

The Moving Picture Experts Group and the Video Coding Experts Group (MPEG and VCEG) have developed a new standard that promises to outperform the earlier MPEG-4 and H.263 standards, providing better compression of video images. The new standard is entitled 'Advanced Video Coding' (AVC) and is published jointly as Part 10 of MPEG-4 and ITU-T Recommendation H.264 [1, 3].

6.1.1 Terminology

Some of the important terminology adopted in the H.264 standard is as follows (the details of these concepts are explained in later sections):

A field (of interlaced video) or a frame (of progressive or interlaced video) is encoded to produce a *coded picture*. A coded frame has a *frame number* (signalled in the bitstream), which is not necessarily related to decoding order, and each coded field of a progressive or interlaced frame has an associated *picture order count*, which defines the decoding order of fields. Previously coded pictures (*reference pictures*) may be used for inter prediction of further coded pictures. Reference pictures are organised into one or two lists (sets of numbers corresponding to reference pictures), described as *list 0* and *list 1*.

A coded picture consists of a number of *macroblocks*, each containing 16×16 luma samples and associated chroma samples (8×8 Cb and 8×8 Cr samples in the current standard). Within each picture, macroblocks are arranged in *slices*, where a slice is a set of macroblocks in raster scan order (but not necessarily contiguous – see section 6.4.3). An *I slice* may contain only I macroblock types (see below), a *P slice* may contain P and I macroblock types and a *B slice* may contain B and I macroblock types. (There are two further slice types, SI and SP, discussed in section 6.6.1).

I macroblocks are predicted using intra prediction from decoded samples in the current slice. A prediction is formed either (a) for the complete macroblock or (b) for each 4×4 block

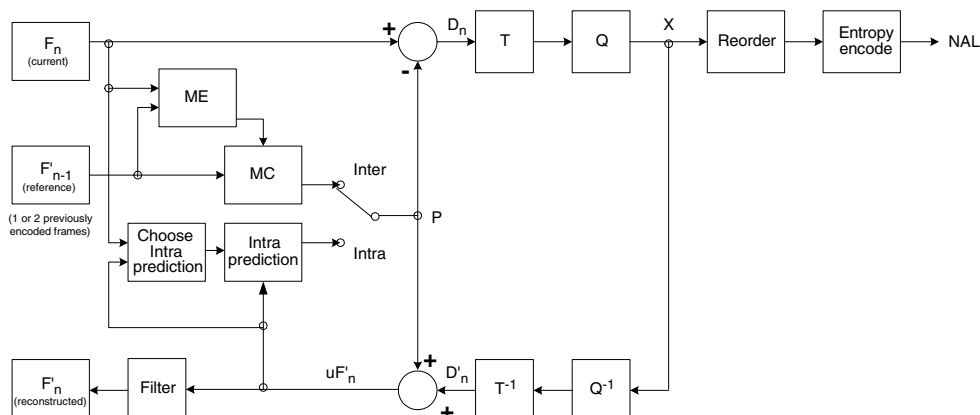


Figure 6.1 H.264 Encoder

of luma samples (and associated chroma samples) in the macroblock. (An alternative to intra prediction, I_PCM mode, is described in section 6.4.6).

P macroblocks are predicted using inter prediction from reference picture(s). An inter coded macroblock may be divided into *macroblock partitions*, i.e. blocks of size 16×16 , 16×8 , 8×16 or 8×8 luma samples (and associated chroma samples). If the 8×8 partition size is chosen, each 8×8 *sub-macroblock* may be further divided into *sub-macroblock partitions* of size 8×8 , 8×4 , 4×8 or 4×4 luma samples (and associated chroma samples). Each macroblock partition may be predicted from one picture in list 0. If present, every sub-macroblock partition in a sub-macroblock is predicted from the same picture in list 0.

B macroblocks are predicted using inter prediction from reference picture(s). Each macroblock partition may be predicted from one or two reference pictures, one picture in list 0 and/or one picture in list 1. If present, every sub-macroblock partition in a sub-macroblock is predicted from (the same) one or two reference pictures, one picture in list 0 and/or one picture in list 1.

6.2 THE H.264 CODEC

In common with earlier coding standards, H.264 does not explicitly define a CODEC (enCOder / DECOder pair) but rather defines the syntax of an encoded video bitstream together with the method of decoding this bitstream. In practice, a compliant encoder and decoder are likely to include the functional elements shown in Figure 6.1 and Figure 6.2. With the exception of the deblocking filter, most of the basic functional elements (prediction, transform, quantisation, entropy encoding) are present in previous standards (MPEG-1, MPEG-2, MPEG-4, H.261, H.263) but the important changes in H.264 occur in the details of each functional block.

The Encoder (Figure 6.1) includes two dataflow paths, a ‘forward’ path (left to right) and a ‘reconstruction’ path (right to left). The dataflow path in the Decoder (Figure 6.2) is shown from right to left to illustrate the similarities between Encoder and Decoder. Before examining the detail of H.264, we will describe the main steps in encoding and decoding a frame (or field)

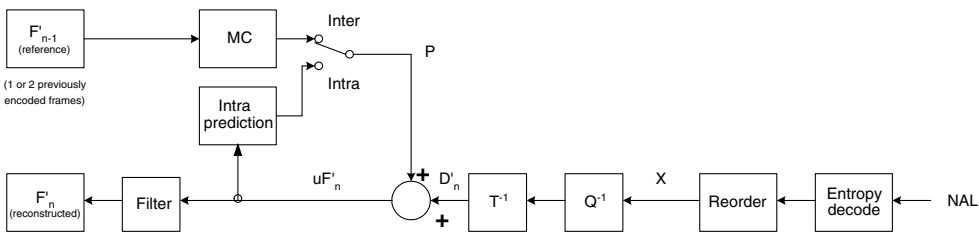


Figure 6.2 H.264 Decoder

of video. The following description is simplified in order to provide an overview of encoding and decoding. The term “block” is used to denote a macroblock partition or sub-macroblock partition (inter coding) or a 16×16 or 4×4 block of luma samples and associated chroma samples (intra coding).

Encoder (forward Path)

An input frame or field F_n is processed in units of a macroblock. Each macroblock is encoded in intra or inter mode and, for each block in the macroblock, a prediction PRED (marked ‘P’ in Figure 6.1) is formed based on reconstructed picture samples. In Intra mode, PRED is formed from samples in the current slice that have previously encoded, decoded and reconstructed (uF'_n in the figures; note that *unfiltered* samples are used to form PRED). In Inter mode, PRED is formed by motion-compensated prediction from one or two reference picture(s) selected from the set of list 0 and/or list 1 reference pictures. In the figures, the reference picture is shown as the previous encoded picture F'_{n-1} but the prediction reference for each macroblock partition (in inter mode) may be chosen from a selection of past or future pictures (in display order) that have already been encoded, reconstructed and filtered.

The prediction PRED is subtracted from the current block to produce a residual (difference) block D_n that is transformed (using a block transform) and quantised to give X , a set of quantised transform coefficients which are reordered and entropy encoded. The entropy-encoded coefficients, together with side information required to decode each block within the macroblock (prediction modes, quantiser parameter, motion vector information, etc.) form the compressed bitstream which is passed to a Network Abstraction Layer (NAL) for transmission or storage.

Encoder (Reconstruction Path)

As well as encoding and transmitting each block in a macroblock, the encoder decodes (reconstructs) it to provide a reference for further predictions. The coefficients X are scaled (Q^{-1}) and inverse transformed (T^{-1}) to produce a difference block D'_n . The prediction block PRED is added to D'_n to create a reconstructed block uF'_n (a decoded version of the original block; u indicates that it is unfiltered). A filter is applied to reduce the effects of blocking distortion and the reconstructed reference picture is created from a series of blocks F'_n .

Decoder

The decoder receives a compressed bitstream from the NAL and entropy decodes the data elements to produce a set of quantised coefficients X . These are scaled and inverse transformed

to give D'_n (identical to the D'_n shown in the Encoder). Using the header information decoded from the bitstream, the decoder creates a prediction block PRED, identical to the original prediction PRED formed in the encoder. PRED is added to D'_n to produce uF'_n which is filtered to create each decoded block F'_n .

6.3 H.264 STRUCTURE

6.3.1 Profiles and Levels

H.264 defines a set of three *Profiles*, each supporting a particular set of coding functions and each specifying what is required of an encoder or decoder that complies with the Profile. The *Baseline Profile* supports intra and inter-coding (using I-slices and P-slices) and entropy coding with context-adaptive variable-length codes (CAVLC). The *Main Profile* includes support for interlaced video, inter-coding using B-slices, inter coding using weighted prediction and entropy coding using context-based arithmetic coding (CABAC). The *Extended Profile* does not support interlaced video or CABAC but adds modes to enable efficient switching between coded bitstreams (SP- and SI-slices) and improved error resilience (Data Partitioning). Potential applications of the Baseline Profile include videotelephony, videoconferencing and wireless communications; potential applications of the Main Profile include television broadcasting and video storage; and the Extended Profile may be particularly useful for streaming media applications. However, each Profile has sufficient flexibility to support a wide range of applications and so these examples of applications should not be considered definitive.

Figure 6.3 shows the relationship between the three Profiles and the coding tools supported by the standard. It is clear from this figure that the Baseline Profile is a subset of the Extended Profile, but not of the Main Profile. The details of each coding tool are described in Sections 6.4, 6.5 and 6.6 (starting with the Baseline Profile tools).

Performance limits for CODECs are defined by a set of Levels, each placing limits on parameters such as sample processing rate, picture size, coded bitrate and memory requirements.

6.3.2 Video Format

H.264 supports coding and decoding of 4:2:0 progressive or interlaced video¹ and the default sampling format for 4:2:0 progressive frames is shown in Figure 2.11 (other sampling formats may be signalled as Video Usability Information parameters). In the default sampling format, chroma (Cb and Cr) samples are aligned horizontally with every 2nd luma sample and are located vertically between two luma samples. An interlaced frame consists of two fields (a top field and a bottom field) separated in time and with the default sampling format shown in Figure 2.12.

¹ An extension to H.264 to support alternative colour sampling structures and higher sample accuracy is currently under development.

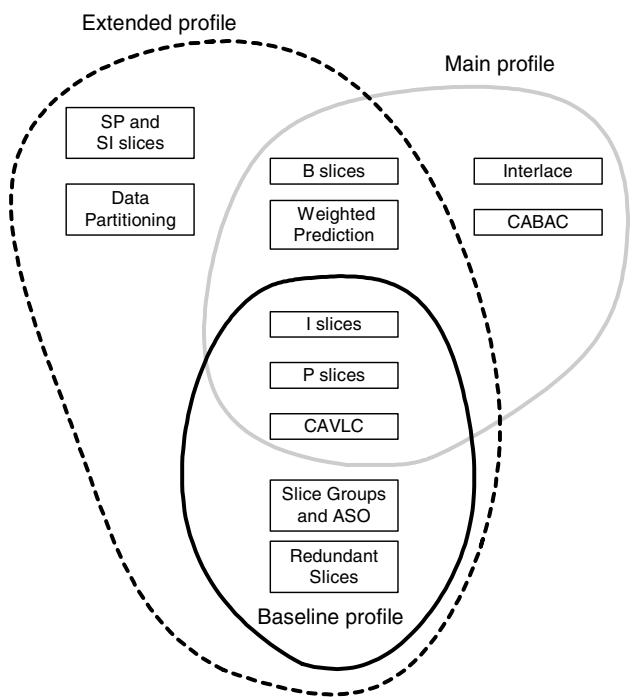


Figure 6.3 H.264 Baseline, Main and Extended profiles



Figure 6.4 Sequence of NAL units

6.3.3 Coded Data Format

H.264 makes a distinction between a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL). The output of the encoding process is VCL data (a sequence of bits representing the coded video data) which are mapped to NAL units prior to transmission or storage. Each NAL unit contains a Raw Byte Sequence Payload (RBSP), a set of data corresponding to coded video data or header information. A coded video sequence is represented by a sequence of NAL units (Figure 6.4) that can be transmitted over a packet-based network or a bitstream transmission link or stored in a file. The purpose of separately specifying the VCL and NAL is to distinguish between coding-specific features (at the VCL) and transport-specific features (at the NAL). Section 6.7 describes the NAL and transport mechanisms in more detail.

6.3.4 Reference Pictures

An H.264 encoder may use one or two of a number of previously encoded pictures as a reference for motion-compensated prediction of each inter coded macroblock or macroblock

Table 6.1 H.264 slice modes

Slice type	Description	Profile(s)
I (Intra)	Contains only I macroblocks (each block or macroblock is predicted from previously coded data within the same slice).	All
P (Predicted)	Contains P macroblocks (each macroblock or macroblock partition is predicted from one list 0 reference picture) and/or I macroblocks.	All
B (Bi-predictive)	Contains B macroblocks (each macroblock or macroblock partition is predicted from a list 0 and/or a list 1 reference picture) and/or I macroblocks.	Extended and Main
SP (Switching P)	Facilitates switching between coded streams; contains P and/or I macroblocks.	Extended
SI (Switching I)	Facilitates switching between coded streams; contains SI macroblocks (a special type of intra coded macroblock).	Extended

partition. This enables the encoder to search for the best ‘match’ for the current macroblock partition from a wider set of pictures than just (say) the previously encoded picture.

The encoder and decoder each maintain one or two lists of reference pictures, containing pictures that have previously been encoded and decoded (occurring before and/or after the current picture in display order). Inter coded macroblocks and macroblock partitions in P slices (see below) are predicted from pictures in a single list, **list 0**. Inter coded macroblocks and macroblock partitions in a B slice (see below) may be predicted from two lists, **list 0** and **list 1**.

6.3.5 Slices

A video picture is coded as one or more slices, each containing an integral number of macroblocks from 1 (1 MB per slice) to the total number of macroblocks in a picture (1 slice per picture). The number of macroblocks per slice need not be constant within a picture. There is minimal inter-dependency between coded slices which can help to limit the propagation of errors. There are five types of coded slice (Table 6.1) and a coded picture may be composed of different types of slices. For example, a Baseline Profile coded picture may contain a mixture of I and P slices and a Main or Extended Profile picture may contain a mixture of I, P and B slices.

Figure 6.5 shows a simplified illustration of the syntax of a coded slice. The slice header defines (among other things) the slice type and the coded picture that the slice ‘belongs’ to and may contain instructions related to reference picture management (see Section 6.4.2). The slice data consists of a series of coded macroblocks and/or an indication of skipped (not coded) macroblocks. Each MB contains a series of header elements (see Table 6.2) and coded residual data.

6.3.6 Macroblocks

A macroblock contains coded data corresponding to a 16×16 sample region of the video frame (16×16 luma samples, 8×8 Cb and 8×8 Cr samples) and contains the syntax elements described in Table 6.2. Macroblocks are numbered (addressed) in raster scan order within a frame.

Table 6.2 Macroblock syntax elements

mb_type	Determines whether the macroblock is coded in intra or inter (P or B) mode; determines macroblock partition size (see Section 6.4.2).
mb_pred	Determines intra prediction modes (intra macroblocks); determines list 0 and/or list 1 references and differentially coded motion vectors for each macroblock partition (inter macroblocks, except for inter MBs with 8×8 macroblock partition size).
sub_mb_pred	(Inter MBs with 8×8 macroblock partition size only) Determines sub-macroblock partition size for each sub-macroblock; list 0 and/or list 1 references for each macroblock partition; differentially coded motion vectors for each macroblock sub-partition.
coded_block_pattern	Identifies which 8×8 blocks (luma and chroma) contain coded transform coefficients.
mb_qp_delta	Changes the quantiser parameter (see Section 6.4.8).
residual	Coded transform coefficients corresponding to the residual image samples after prediction (see Section 6.4.8).

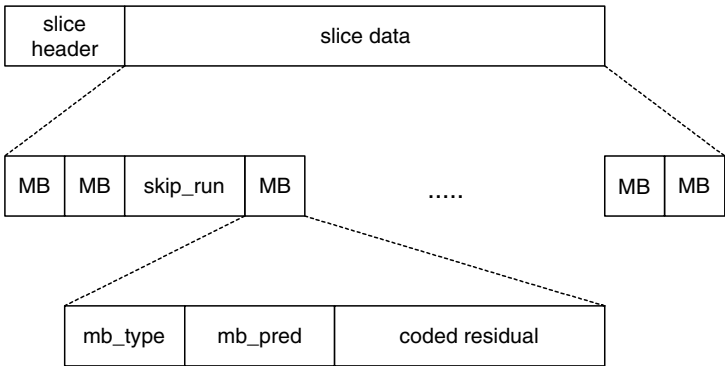


Figure 6.5 Slice syntax

6.4 THE BASELINE PROFILE

6.4.1 Overview

The Baseline Profile supports coded sequences containing I- and P-slices. I-slices contain intra-coded macroblocks in which each 16×16 or 4×4 luma region and each 8×8 chroma region is predicted from previously-coded samples in the same slice. P-slices may contain intra-coded, inter-coded or skipped MBs. Inter-coded MBs in a P slice are predicted from a number of previously coded pictures, using motion compensation with quarter-sample (luma) motion vector accuracy.

After prediction, the residual data for each MB is transformed using a 4×4 integer transform (based on the DCT) and quantised. Quantised transform coefficients are reordered and the syntax elements are entropy coded. In the Baseline Profile, transform coefficients are entropy coded using a context-adaptive variable length coding scheme (CAVLC) and all other

syntax elements are coded using fixed-length or Exponential-Golomb Variable Length Codes. Quantised coefficients are scaled, inverse transformed, reconstructed (added to the prediction formed during encoding) and filtered with a de-blocking filter before (optionally) being stored for possible use in reference pictures for further intra- and inter-coded macroblocks.

6.4.2 Reference Picture Management

Pictures that have previously been encoded are stored in a reference buffer (the decoded picture buffer, DPB) in both the encoder and the decoder. The encoder and decoder maintain a list of previously coded pictures, reference picture list 0, for use in motion-compensated prediction of inter macroblocks in P slices. For P slice prediction, list 0 can contain pictures before and after the current picture in display order and may contain both *short term* and *long term* reference pictures. By default, an encoded picture is reconstructed by the encoder and marked as a short term picture, a recently-coded picture that is available for prediction. Short term pictures are identified by their frame number. Long term pictures are (typically) older pictures that may also be used for prediction and are identified by a variable LongTermPicNum. Long term pictures remain in the DPB until explicitly removed or replaced.

When a picture is encoded and reconstructed (in the encoder) or decoded (in the decoder), it is placed in the decoded picture buffer and is either (a) marked as ‘unused for reference’ (and hence not used for any further prediction), (b) marked as a short term picture, (c) marked as a long term picture or (d) simply output to the display. By default, short term pictures in list 0 are ordered from the highest to the lowest PicNum (a variable derived from the frame number) and long term pictures are ordered from the lowest to the highest LongTermPicNum. The encoder may signal a change to the default reference picture list order. As each new picture is added to the short term list at position 0, the indices of the remaining short-term pictures are incremented. If the number of short term and long term pictures is equal to the maximum number of reference frames, the oldest short-term picture (with the highest index) is removed from the buffer (known as *sliding window* memory control). The effect that this process is that the encoder and decoder each maintain a ‘window’ of N short-term reference pictures, including the current picture and $(N - 1)$ previously encoded pictures.

Adaptive memory control commands, sent by the encoder, manage the short and long term picture indexes. Using these commands, a short term picture may be assigned a long term frame index, or any short term or long term picture may be marked as ‘unused for reference’.

The encoder chooses a reference picture from list 0 for encoding each macroblock partition in an inter-coded macroblock. The choice of reference picture is signalled by an index number, where index 0 corresponds to the first frame in the short term section and the indices of the long term frames start after the last short term frame (as shown in the following example).

Example: Reference buffer management (P-slice)

Current frame number = 250

Number of reference frames = 5

Operation	Reference picture list				
	0	1	2	3	4
Initial state	–	–	–	–	–
Encode frame 250	250	–	–	–	–
Encode 251	251	250	–	–	–
Encode 252	252	251	250	–	–
Encode 253	253	252	251	250	–
Assign 251 to LongTermPicNum 0	253	252	250	0	–
Encode 254	254	253	252	250	0
Assign 253 to LongTermPicNum 4	254	252	250	0	4
Encode 255	255	254	252	0	4
Assign 255 to LongTermPicNum 3	254	252	0	3	4
Encode 256	256	254	0	3	4

(Note that in the above example, 0, 3 and 4 correspond to the decoded frames 251, 255 and 253 respectively).

Instantaneous Decoder Refresh Picture

An encoder sends an IDR (Instantaneous Decoder Refresh) coded picture (made up of I- or SI-slices) to clear the contents of the reference picture buffer. On receiving an IDR coded picture, the decoder marks all pictures in the reference buffer as ‘unused for reference’. All subsequent transmitted slices can be decoded without reference to any frame decoded prior to the IDR picture. The first picture in a coded video sequence is always an IDR picture.

6.4.3 Slices

A bitstream conforming to the the Baseline Profile contains coded I and/or P slices. An I slice contains only intra-coded macroblocks (predicted from previously coded samples in the same slice, see Section 6.4.6) and a P slice can contain inter coded macroblocks (predicted from samples in previously coded pictures, see Section 6.4.5), intra coded macroblocks or Skipped macroblocks. When a Skipped macroblock is signalled in the bitstream, no further data is sent for that macroblock. The decoder calculates a vector for the skipped macroblock (see Section 6.4.5.3) and reconstructs the macroblock using motion-compensated prediction from the first reference picture in list 0.

An H.264 encoder may optionally insert a picture delimiter RBSP unit at the boundary between coded pictures. This indicates the start of a new coded picture and indicates which slice types are allowed in the following coded picture. If the picture delimiter is not used, the decoder is expected to detect the occurrence of a new picture based on the header of the first slice in the new picture.

Redundant coded picture

A picture marked as ‘redundant’ contains a redundant representation of part or all of a coded picture. In normal operation, the decoder reconstructs the frame from ‘primary’

Table 6.3 Macroblock to slice group map types

Type	Name	Description
0	Interleaved	run_length MBs are assigned to each slice group in turn (Figure 6.6).
1	Dispersed	MBs in each slice group are dispersed throughout the picture (Figure 6.7).
2	Foreground and background	All but the last slice group are defined as rectangular regions within the picture. The last slice group contains all MBs not contained in any other slice group (the 'background'). In the example in Figure 6.8, group 1 overlaps group 0 and so MBs not already allocated to group 0 are allocated to group 1.
3	Box-out	A 'box' is created starting from the centre of the frame (with the size controlled by encoder parameters) and containing group 0; all other MBs are in group 1 (Figure 6.9).
4	Raster scan	Group 0 contains MBs in raster scan order from the top-left and all other MBs are in group 1 (Figure 6.9).
5	Wipe	Group 0 contains MBs in vertical scan order from the top-left and all other MBs are in group 1 (Figure 6.9).
6	Explicit	A parameter, slice_group_id, is sent for each MB to indicate its slice group (i.e. the macroblock map is entirely user-defined).

(nonredundant)' pictures and discards any redundant pictures. However, if a primary coded picture is damaged (e.g. due to a transmission error), the decoder may replace the damaged area with decoded data from a redundant picture if available.

Arbitrary Slice Order (ASO)

The Baseline Profile supports Arbitrary Slice Order which means that slices in a coded frame may follow any decoding order. ASO is defined to be in use if the first macroblock in any slice in a decoded frame has a smaller macroblock address than the first macroblock in a *previously* decoded slice in the same picture.

Slice Groups

A *slice group* is a subset of the macroblocks in a coded picture and may contain one or more slices. Within each slice in a slice group, MBs are coded in raster order. If only one slice group is used per picture, then all macroblocks in the picture are coded in raster order (unless ASO is in use, see above). Multiple slice groups (described in previous versions of the draft standard as Flexible Macroblock Ordering or FMO) make it possible to map the sequence of coded MBs to the decoded picture in a number of flexible ways. The allocation of macroblocks is determined by a *macroblock to slice group map* that indicates which slice group each MB belongs to. Table 6.3 lists the different types of macroblock to slice group maps.

Example: 3 slice groups are used and the map type is 'interleaved' (Figure 6.6). The coded picture consists of first, all of the macroblocks in slice group 0 (filling every 3rd row of macroblocks); second, all of the macroblocks in slice group 1; and third, all of the macroblocks in slice group 0. Applications of multiple slice groups include error resilience, for example if one of the slice groups in the dispersed map shown in Figure 6.7 is 'lost' due to errors, the missing data may be concealed by interpolation from the remaining slice groups.

6.6 THE EXTENDED PROFILE

The Extended Profile (known as the X Profile in earlier versions of the draft H.264 standard) may be particularly useful for applications such as video streaming. It includes all of the features of the Baseline Profile (i.e. it is a superset of the Baseline Profile, unlike Main Profile), together with B-slices (Section 6.5.1), Weighted Prediction (Section 6.5.2) and additional features to support efficient streaming over networks such as the Internet. SP and SI slices facilitate switching between different coded streams and 'VCR-like' functionality and Data Partitioned slices can provide improved performance in error-prone transmission environments.

6.6.1 SP and SI slices

SP and SI slices are specially-coded slices that enable (among other things) efficient switching between video streams and efficient random access for video decoders [10]. A common requirement in a streaming application is for a video decoder to switch between one of several encoded streams. For example, the same video material is coded at multiple bitrates for transmission across the Internet and a decoder attempts to decode the highest-bitrate stream it can receive but may require switching automatically to a lower-bitrate stream if the data throughput drops.

Example

A decoder is decoding Stream A and wants to switch to decoding Stream B (Figure 6.48). For simplicity, assume that each frame is encoded as a single slice and predicted from one reference (the previous decoded frame). After decoding P-slices A_0 and A_1 , the decoder wants to switch to Stream B and decode B_2 , B_3 and so on. If all the slices in Stream B are coded as P-slices, then the decoder will not have the correct decoded reference frame(s) required to reconstruct B_2 (since B_2 is predicted from the decoded picture B_1 which does not exist in stream A). One solution is to code frame B_2 as an I-slice. Because it is coded without prediction from any other frame, it can be decoded independently of preceding frames in stream B and the decoder can therefore switch between stream A and stream B as shown in Figure 6.49. Switching can be accommodated by inserting an I-slice at regular intervals in the coded sequence to create 'switching points'. However, an I-slice is likely to contain much more coded data than a P-slice and the result is an undesirable peak in the coded bitrate at each switching point.

SP-slices are designed to support switching between similar coded sequences (for example, the same source sequence encoded at various bitrates) without the increased bitrate penalty of I-slices (Figure 6.49). At the switching point (frame 2 in each sequence), there are three SP-slices, each coded using motion compensated prediction (making them more efficient than I-slices). SP-slice A_2 can be decoded using reference picture A_1 and SP-slice B_2 can be decoded using reference picture B_1 . The key to the switching process is SP-slice AB_2 (known as a *switching SP-slice*), created in such a way that it can be decoded using motion-compensated reference picture A_1 , to produce decoded frame B_2 (i.e. the decoder output frame B_2 is identical whether decoding B_1 followed by B_2 or A_1 followed by AB_2). An extra SP-slice is required at each switching point (and in fact another SP-slice, BA_2 , would be required to switch in the other direction) but this is likely to be more efficient than encoding frames A_2

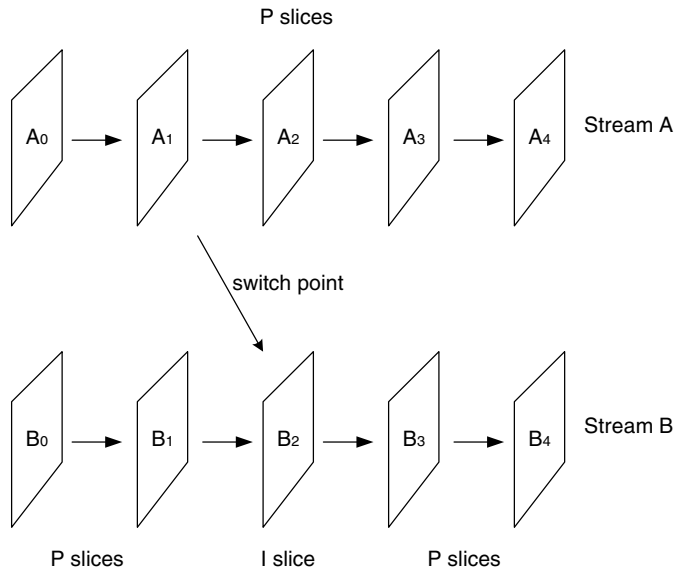


Figure 6.48 Switching streams using I-slices

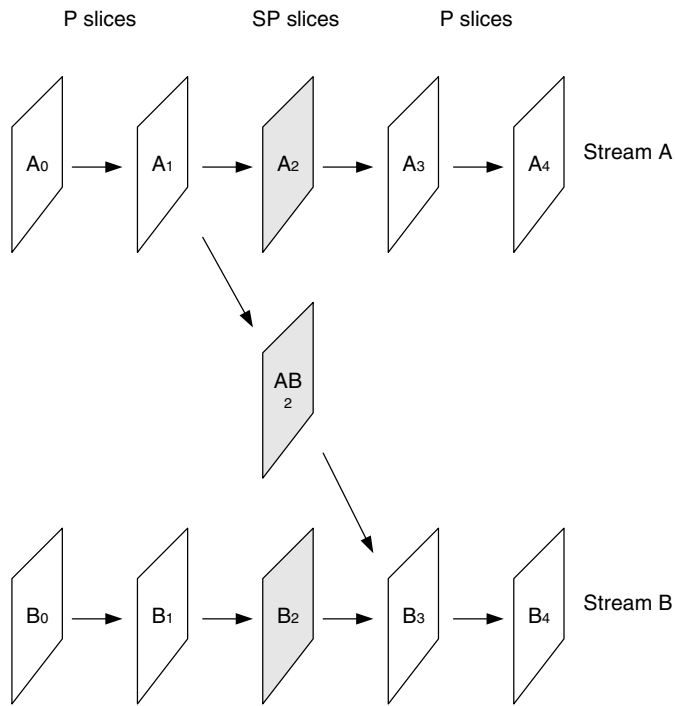


Figure 6.49 Switching streams using SP-slices

Table 6.18 Switching from stream A to stream B using SP-slices

Input to decoder	MC reference	Output of decoder
P-slice A_0	[earlier frame]	Decoded frame A_0
P-slice A_1	Decoded frame A_0	Decoded frame A_1
SP-slice A_2	Decoded frame A_1	Decoded frame B_2
P-slice B_3	Decoded frame B_2	Decoded frame B_3
...	...	...

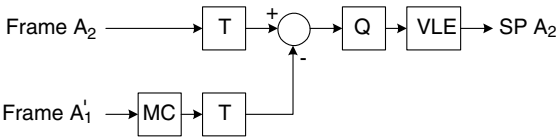


Figure 6.50 Encoding SP-slice A_2 (simplified)

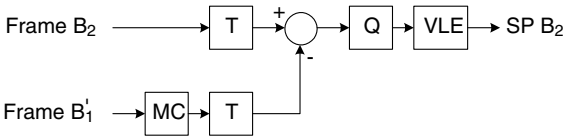


Figure 6.51 Encoding SP-slice B_2 (simplified)

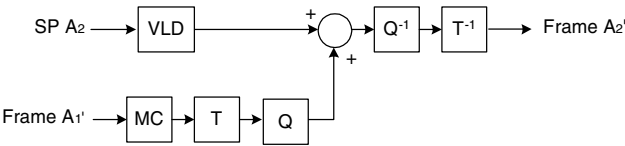


Figure 6.52 Decoding SP-slice A_2 (simplified)

and B_2 as I-slices. Table 6.18 lists the steps involved when a decoder switches from stream A to stream B.

Figure 6.50 shows a simplified diagram of the encoding process for SP-slice A_2 , produced by subtracting a motion-compensated version of A'_1 (decoded frame A_1) from frame A_2 and then coding the residual. Unlike a ‘normal’ P-slice, the subtraction occurs in the transform domain (after the block transform). SP-slice B_2 is encoded in the same way (Figure 6.51). A decoder that has previously decoded frame A_1 can decode SP-slice A_2 as shown in Figure 6.52. Note that these diagrams are simplified; in practice further quantisation and rescaling steps are required to avoid mismatch between encoder and decoder and a more detailed treatment of the process can be found in [11].

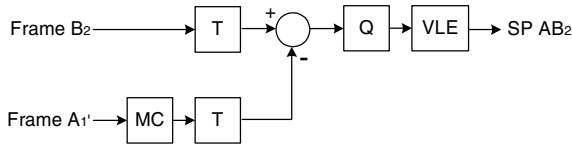


Figure 6.53 Encoding SP-slice AB_2 (simplified)

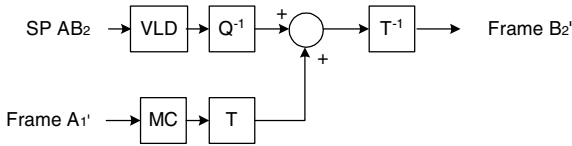


Figure 6.54 Decoding SP-slice AB_2 (simplified)

SP-slice AB_2 is encoded as shown in Figure 6.53 (simplified). Frame B_2 (the frame we are switching to) is transformed and a motion-compensated prediction is formed from A'_1 (the decoded frame from which we are switching). The ‘MC’ block in this diagram attempts to find the best match for each MB of frame B_2 using *decoded picture A_1 as a reference*. The motion-compensated prediction is transformed, then subtracted from the transformed B_2 (i.e. in the case of a switching SP slice, subtraction takes place in the transform domain). The residual (after subtraction) is quantized, encoded and transmitted.

A decoder that has previously decoded A'_1 can decode SP-slice AB_2 to produce B'_2 (Figure 6.54). A'_1 is motion compensated (using the motion vector data encoded as part of AB_2), transformed and added to the decoded and scaled (inverse quantized) residual, then the result is inverse transformed to produce B'_2 .

If streams A and B are versions of the same original sequence coded at different bitrates, the motion-compensated prediction of B_2 from A'_1 (SP-slice AB_2) should be quite efficient. Results show that using SP-slices to switch between different versions of the same sequence is significantly more efficient than inserting I-slices at switching points. Another application of SP-slices is to provide random access and ‘VCR-like’ functionalities. For example, an SP-slice and a switching SP-slice are placed at the position of frame 10 (Figure 6.55). A decoder can fast-forward from A_0 directly to A_{10} by (a) decoding A_0 , then (b) decoding switching SP-slice A_{0-10} to produce A_{10} by prediction from A_0 .

A further type of switching slice, the SI-slice, is supported by the Extended Profile. This is used in a similar way to a switching SP-slice, except that the prediction is formed using the 4×4 Intra Prediction modes (see Section 6.4.6.1) from previously-decoded samples of the reconstructed frame. This slice mode may be used (for example) to switch from one sequence to a completely different sequence (in which case it will not be efficient to use motion compensated prediction because there is no correlation between the two sequences).

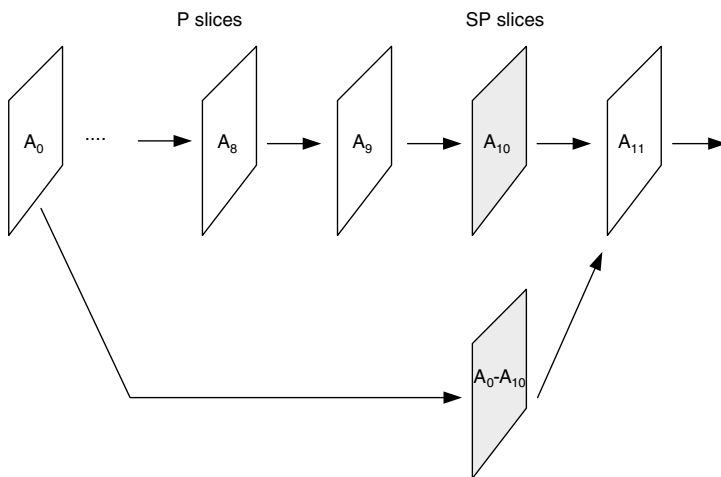


Figure 6.55 Fast-forward using SP-slices

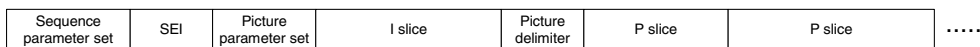


Figure 6.56 Example sequence of RBSP elements

6.6.2 Data Partitioned Slices

The coded data that makes up a slice is placed in three separate Data Partitions (A, B and C), each containing a subset of the coded slice. Partition A contains the slice header and header data for each macroblock in the slice, Partition B contains coded residual data for Intra and SI slice macroblocks and Partition C contains coded residual data for inter coded macroblocks (forward and bi-directional). Each Partition can be placed in a separate NAL unit and may therefore be transported separately.

If Partition A data is lost, it is likely to be difficult or impossible to reconstruct the slice, hence Partition A is highly sensitive to transmission errors. Partitions B and C can (with careful choice of coding parameters) be made to be independently decodeable and so a decoder may (for example) decode A and B only, or A and C only, lending flexibility in an error-prone environment.

6.7 TRANSPORT OF H.264

A coded H.264 video sequence consists of a series of NAL units, each containing an RBSP (Table 6.19). Coded slices (including Data Partitioned slices and IDR slices) and the End of Sequence RBSP are defined as VCL NAL units whilst all other elements are just NAL units.

An example of a typical sequence of RBSP units is shown in Figure 6.56. Each of these units is transmitted in a separate NAL unit. The header of the NAL unit (one byte) signals the type of RBSP unit and the RBSP data makes up the rest of the NAL unit.

Table 6.19

RBSP type	Description
Parameter Set	‘Global’ parameters for a sequence such as picture dimensions, video format, macroblock allocation map (see Section 6.4.3).
Supplemental Enhancement Information	Side messages that are not essential for correct decoding of the video sequence.
Picture Delimiter	Boundary between video pictures (optional). If not present, the decoder infers the boundary based on the frame number contained within each slice header.
Coded slice Data Partition A, B or C	Header and data for a slice; this RBSP unit contains actual coded video data. Three units containing Data Partitioned slice layer data (useful for error resilient decoding). Partition A contains header data for all MBs in the slice, Partition B contains intra coded data and partition C contains inter coded data.
End of sequence	Indicates that the next picture (in decoding order) is an IDR picture (see Section 6.4.2). (Not essential for correct decoding of the sequence).
End of stream	Indicates that there are no further pictures in the bitstream. (Not essential for correct decoding of the sequence).
Filler data	Contains ‘dummy’ data (may be used to increase the number of bytes in the sequence). (Not essential for correct decoding of the sequence).

Parameter sets

H.264 introduces the concept of *parameter sets*, each containing information that can be applied to a large number of coded pictures. A *sequence parameter set* contains parameters to be applied to a complete video sequence (a set of consecutive coded pictures). Parameters in the sequence parameter set include an identifier (seq_parameter_set_id), limits on frame numbers and picture order count, the number of reference frames that may be used in decoding (including short and long term reference frames), the decoded picture width and height and the choice of progressive or interlaced (frame or frame / field) coding. A *picture parameter set* contains parameters which are applied to one or more decoded pictures within a sequence. Each picture parameter set includes (among other parameters) an identifier (pic_parameter_set_id), a selected seq_parameter_set_id, a flag to select VLC or CABAC entropy coding, the number of slice groups in use (and a definition of the type of slice group map), the number of reference pictures in list 0 and list 1 that may be used for prediction, initial quantizer parameters and a flag indicating whether the default deblocking filter parameters are to be modified.

Typically, one or more sequence parameter set(s) and picture parameter set(s) are sent to the decoder prior to decoding of slice headers and slice data. A coded slice header refers to a pic_parameter_set_id and this ‘activates’ that particular picture parameter set. The ‘activated’ picture parameter set then remains active until a different picture parameter set is activated by being referred to in another slice header. In a similar way, a picture parameter set refers to a seq_parameter_set_id which ‘activates’ that sequence parameter set. The activated set remains in force (i.e. its parameters are applied to all consecutive coded pictures) until a different sequence parameter set is activated.

The parameter set mechanism enables an encoder to signal important, infrequently-changing sequence and picture parameters separately from the coded slices themselves. The parameter sets may be sent well ahead of the slices that refer to them, or by another transport

mechanism (e.g. over a reliable transmission channel or even ‘hard wired’ in a decoder implementation). Each coded slice may ‘call up’ the relevant picture and sequence parameters using a single VLC (`pic_parameter_set_id`) in the slice header.

Transmission and Storage of NAL units

The method of transmitting NAL units is not specified in the standard but some distinction is made between transmission over packet-based transport mechanisms (e.g. packet networks) and transmission in a continuous data stream (e.g. circuit-switched channels). In a packet-based network, each NAL unit may be carried in a separate packet and should be organised into the correct sequence prior to decoding. In a circuit-switched transport environment, a start code prefix (a uniquely-identifiable delimiter code) is placed before each NAL unit to make a *byte stream* prior to transmission. This enables a decoder to search the stream to find a start code prefix identifying the start of a NAL unit.

In a typical application, coded video is required to be transmitted or stored together with associated audio track(s) and side information. It is possible to use a range of transport mechanisms to achieve this, such as the Real Time Protocol and User Datagram Protocol (RTP/UDP). An Amendment to MPEG-2 Systems specifies a mechanism for transporting H.264 video (see Chapter 7) and ITU-T Recommendation H.241 defines procedures for using H.264 in conjunction with H.32× multimedia terminals. Many applications require storage of multiplexed video, audio and side information (e.g. streaming media playback, DVD playback). A forthcoming Amendment to MPEG-4 Systems (Part 1) specifies how H.264 coded data and associated media streams can be stored in the ISO Media File Format (see Chapter 7).

6.8 CONCLUSIONS

H.264 provides mechanisms for coding video that are optimised for compression efficiency and aim to meet the needs of practical multimedia communication applications. The range of available coding tools is more restricted than MPEG-4 Visual (due to the narrower focus of H.264) but there are still many possible choices of coding parameters and strategies. The success of a practical implementation of H.264 (or MPEG-4 Visual) depends on careful design of the CODEC and effective choices of coding parameters. The next chapter examines design issues for each of the main functional blocks of a video CODEC and compares the performance of MPEG-4 Visual and H.264.

6.9 REFERENCES

1. ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, 2003.
2. T. Wiegand, G. Sullivan, G. Bjontegaard and A. Luthra, Overview of the H.264 / AVC Video Coding Standard, IEEE Transactions on Circuits and Systems for Video Technology, to be published in 2003.
3. A. Hallapuro, M. Karczewicz and H. Malvar, Low Complexity Transform and Quantization – Part I: Basic Implementation, JVT document JVT-B038, Geneva, February 2002.
4. H.264 Reference Software Version JM6.1d, <http://bs.hhi.de/~suehring/tml/>, March 2003.

5. S. W. Golomb, Run-length encoding, *IEEE Trans. on Inf. Theory*, **IT-12**, pp. 399–401, 1966.
6. G. Bjøntegaard and K. Lillevold, Context-adaptive VLC coding of coefficients, JVT document JVT-C028, Fairfax, May 2002.
7. D. Marpe, G. Blättermann and T. Wiegand, Adaptive codes for H.26L, ITU-T SG16/6 document VCEG-L13, Eibsee, Germany, January 2001.
8. H. Schwarz, D. Marpe and T. Wiegand, CABAC and slices, JVT document JVT-D020, Klagenfurt, Austria, July 2002.
9. D. Marpe, H. Schwarz and T. Wiegand, Context-Based Adaptive Binary Arithmetic Coding in the H.264 / AVC Video Compression Standard, *IEEE Transactions on Circuits and Systems for Video Technology*, to be published in 2003.
10. M. Karczewicz and R. Kurceren, A proposal for SP-frames, ITU-T SG16/6 document VCEG-L27, Eibsee, Germany, January 2001.
11. M. Karczewicz and R. Kurceren, The SP and SI Frames Design for H.264/AVC, *IEEE Transactions on Circuits and Systems for Video Technology*, to be published in 2003.