

Final Details for Order #112-0461028-3520257

[Print this page for your records.](#)

Order Placed: February 14, 2025

Amazon.com order number: 112-0461028-3520257

Order Total: \$109.89

Shipped on February 14, 2025

Items Ordered

1 of: *Coding Video: A Practical Guide to HEVC and Beyond*, Richardson, Iain E.

Sold by: Amazon.com Services, Inc

Supplied by: Other

Condition: New

Price

\$101.99

Shipping Address:

Seung Hurt

SAN DIEGO, CA 92129-3059

United States

Shipping Speed:

FREE Prime Delivery

Payment information

Payment Method:

Prime Visa ending in 9412

Earns 5% back

Billing address

Seung Hurt

SAN DIEGO, CA 92129-3059

United States

Credit Card transactions

Item(s) Subtotal: \$101.99

Shipping & Handling: \$0.00

Total before tax: \$101.99

Estimated tax to be collected: \$7.90

Grand Total: \$109.89

Visa ending in 9412: February 14, 2025: \$109.89

To view the status of your order, return to [Order Summary](#).

[Conditions of Use](#) | [Privacy Notice](#) © 1996-2025, Amazon.com, Inc. or its affiliates

[Back to top](#)

English

United States

Help

[Conditions of Use](#) [Privacy Notice](#) [Consumer Health Data Privacy Disclosure](#) [Your Ads Privacy Choices](#)

© 1996-2025, Amazon.com, Inc. or its affiliates

PETITIONER'S

Exhibit No.:

1019

26-Feb-25

[Klara Miller]

— IAIN E. RICHARDSON —

CODING VIDEO

A PRACTICAL GUIDE TO HEVC AND BEYOND



WILEY

A fully up-to-date guide to transformative consumer technologies

Video compression – or video coding – has been at the centre of a revolution in the way video is produced, delivered, and consumed. It has made the switch from analogue to digital video possible and has enabled fundamental shifts in the way we now watch video. New video compression standards, together with adaptive streaming protocols, are used to deliver high-quality video to homes and workplaces around the world.

Coding Video provides a practical and comprehensive guide to the new landscape of video coding and video streaming. This book explains the core technologies with a wealth of practical examples and illustrations, covers key standards such as H.265/HEVC and includes an introduction to the new H.266/VVC standard. *Coding Video* will appeal to engineers, application developers, product designers and digital video professionals, as well as to graduate students and researchers in Engineering, Computer Science and related subjects.

Iain E. Richardson is the founder of Vcodex BV and was formerly Professor of Video Coding at the Robert Gordon University, Aberdeen, Scotland. He is the author of four other books on video coding, which include two widely-cited books on the H.264 Advanced Video Coding standard.



A book companion website is available at
www.wiley.com/go/richardson/codingvideo1

Cover Design: Wiley
Cover Image: © Aleksandra Konoplia/Getty Images

www.wiley.com

WILEY



Also available
as an e-book



Coding Video

Coding Video

A Practical Guide to HEVC and Beyond

Iain E. Richardson
Director, Vcodex Ltd
Delft
The Netherlands

WILEY

This edition first published 2024
© 2024 John Wiley & Sons Ltd

All rights reserved, including rights for text and data mining and training of artificial technologies or similar technologies. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by law. Advice on how to obtain permission to reuse material from this title is available at <http://www.wiley.com/go/permissions>.

The right of Iain E. Richardson to be identified as the author of this work has been asserted in accordance with law.

Registered Offices

John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Limit of Liability/Disclaimer of Warranty

While the publisher and authors have used their best efforts in preparing this work, they make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives, written sales materials or promotional statements for this work. This work is sold with the understanding that the publisher is not engaged in rendering professional services. The advice and strategies contained herein may not be suitable for your situation. You should consult with a specialist where appropriate. The fact that an organization, website, or product is referred to in this work as a citation and/or potential source of further information does not mean that the publisher and authors endorse the information or services the organization, website, or product may provide or recommendations it may make. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Library of Congress Cataloging-in-Publication Data applied for

Hardback ISBN 9781118711781

Cover Design: Wiley
Cover Image: © Aleksandra Konoplia/Getty Images

Set in 9.5/12.5pt STIXTwoText by Straive, Pondicherry, India

SKY10082067_081524

To Naomi

al technologies or
rieval system, or
ecording or otherwise,
rom this title is available

asserted in accordance

8SQ, UK

about Wiley products

mand. Some content
ormats.

John Wiley & Sons,
ed without written
Wiley & Sons, Inc. is

they make no
ntents of this work and
ties of merchantability
representatives, written
nderstanding that the
s contained herein
ppropriate. The fact
or potential source of
mation or services the
rther, readers should
1 when this work was
loss of profit or any other
, or other damages.

To Naomi

Contents

About the Author	<i>xvii</i>
Acknowledgements	<i>xix</i>
About the Companion Website	<i>xxi</i>

1	Introduction	1
1.1	Why Write This Book?	1
1.2	What Is in the Book?	2
1.3	How Should You Use This Book?	3
	References	4
2	Video Coding and Video Quality	5
2.1	Introduction	5
2.2	An Overview of Video Coding	6
2.2.1	Just How Much Data Are We Talking About?	6
2.2.2	Bandwidth Is Increasing Every Year, But...	7
2.3	Inputs and Outputs	9
2.3.1	Digital Video	10
2.3.2	Uncompressed Video Formats	10
2.3.2.1	Frames, Fields and Pixels	10
2.3.2.2	Colour and Colour Spaces	11
2.3.3	Side Information	15
2.3.4	Coded Video	15
2.4	Structural Elements	15
2.5	Prediction	17
2.5.1	Prediction in a Video Encoder	17
2.5.2	Prediction in a Video Decoder	18
2.5.3	Prediction Example	18
2.5.4	Choosing a Prediction	21
2.5.5	Inter and Intra Prediction	23
2.6	Transform and Quantisation	25
2.6.1	Transform	25
2.6.2	Quantisation	26
2.7	Bitstream Coding	27
2.7.1	Pre-processing	28

2.7.2	Entropy Encoding	29	3.3.8.2	VP8, VP9 and	
2.8	The Coded Bitstream	30	3.3.8.3	Essential Video Coding	
2.9	Storing and Transmitting the Coded Bitstream	30	3.4	Comparing	
2.10	The Decoder	32	3.5	Conclusions	
2.10.1	Entropy Decoding	32		References	
2.10.2	Rescaling and Inverse Transform	32	4	Structures	
2.10.3	Reconstruction	33	4.1	Introduction	
2.10.4	Decoder Output	33	4.1.1	How Does a	
2.11	The Video Codec Model	33	4.1.2	Separating a	
2.12	Video Codec Performance	35	4.1.3	Regular or Irregular	
2.12.1	Video Quality	36	4.1.4	Transmissions	
2.12.2	Subjective Quality Measurement	37	4.2	Coded Video	
2.12.2.1	Factors Influencing Subjective Quality	37	4.2.1	Coded Sequences	
2.12.2.2	Subjective Quality Assessment Methods	37	4.2.2	Coded Frames	
2.12.3	Objective Quality Measurement	38	4.2.3	Multi-picture	
2.12.3.1	PSNR	38	4.3	Coded Video	
2.12.3.2	Other Objective Quality Metrics	39	4.3.1	Basic Units	
2.13	Conclusion	40	4.3.2	Slices	76
	References	41	4.3.3	Tiles	78
3	A History of Video Coding and Video Coding Standards	43	4.4	Coded Video	
3.1	Introduction	43	4.4.1	Coding Structures	
3.2	The Foundations of Video Coding, 1950–1990	44	4.4.2	Prediction Structures	
3.2.1	Entropy Coding: 1952–1987	44	4.4.3	Transform Structures	
3.2.2	Frame Differencing: 1956–1972	46	4.5	HEVC Coding	
3.2.3	The Discrete Cosine Transform: 1974–1984	47	4.5.1	Coded HEVC	
3.2.4	The Development of the Video Codec Model: 1977–1985	48	4.5.2	Parameter Sets	
3.2.5	Early Advances in Motion-Compensated Prediction: 1989–1995	51	4.5.3	Profiles and	
3.2.5.1	Bidirectional Prediction	51	4.5.4	Reference Pictures	
3.2.5.2	Variable Block Size Motion Compensation	52	4.5.5	HEVC Pictures	
3.2.5.3	Multiple Reference Frames	52	4.5.5.1	Random Access	
3.2.5.4	Motion Compensation with Fractional Pixel Accuracy	52	4.5.5.2	Examples: I	
3.3	Video Coding Standards and Formats: 1990–2021	53	4.5.5.3	Random Access	
3.3.1	How Are the Standards Developed?	54	4.5.5.4	Bitstream Structures	
3.3.1.1	MPEG and VCEG	54	4.5.6	HEVC-Coded	
3.3.1.2	The Development Process	55	4.5.7	Slices and S	
3.3.1.3	Standards and Intellectual Property Rights	55	4.5.8	Tiles	97
3.3.2	H.120, H.261 and MPEG-1: Early Video Coding Standards	56	4.5.9	Combining	
3.3.3	MPEG-2: Digital TV and DVD-Video	58	4.5.10	HEVC Structures	
3.3.4	H.263 and MPEG-4 Part 2: Internet and Mobile Video	59	4.5.10.1	Parallel Decoding	
3.3.5	H.264: Multi-Purpose Video Coding	60	4.5.10.2	Parallel Decoding	
3.3.6	H.265/HEVC	61	4.5.10.3	Wavefront Parallel	
3.3.7	H.266/VVC	62	4.5.11	Coding Trees	
3.3.8	Other Standards and Formats	62	4.5.11.1	Quantisation	
3.3.8.1	VC-1	62			

3.3.8.2	VP8, VP9 and AV1	63
3.3.8.3	Essential Video Coding and Low-Complexity Enhancement Video Coding	64
3.4	Comparing Video Coding Standards	65
3.5	Conclusions	66
	References	66
4	Structures	69
4.1	Introduction	69
4.1.1	How Does a Video Codec Use Structures?	69
4.1.2	Separating a Sequence into Processing Units	70
4.1.3	Regular or Irregular Structures	70
4.1.4	Transmission and Access	72
4.2	Coded Video: Sequence to Picture	72
4.2.1	Coded Sequence	73
4.2.2	Coded Frame or Picture	74
4.2.3	Multi-picture Structures	74
4.3	Coded Video: Picture to Basic Unit	75
4.3.1	Basic Units	76
4.3.2	Slices	76
4.3.3	Tiles	78
4.4	Coded Video: Basic Unit to Block	80
4.4.1	Coding Structures	80
4.4.2	Prediction Structures	80
4.4.3	Transform Structures	81
4.5	HEVC Coding Structures	82
4.5.1	Coded HEVC Sequence	82
4.5.2	Parameter Sets	82
4.5.3	Profiles and Levels	85
4.5.4	Reference Picture Sets	86
4.5.5	HEVC Picture Types and Random Access	88
4.5.5.1	Random Access Picture	88
4.5.5.2	Examples: RAPs, Trailing and Leading Pictures	90
4.5.5.3	Random Access and Error Recovery	92
4.5.5.4	Bitstream Splicing Operations	92
4.5.6	HEVC-Coded Pictures	94
4.5.7	Slices and Slice Segments	95
4.5.8	Tiles	97
4.5.9	Combining Slices and Tiles in HEVC	99
4.5.10	HEVC Structures and Parallel Processing	101
4.5.10.1	Parallel Decoding Using Tiles	101
4.5.10.2	Parallel Decoding Using Slices	102
4.5.10.3	Wavefront Parallel Processing/Parallelising CTU Rows	102
4.5.11	Coding Tree Units and Coding Units	103
4.5.11.1	Quantisation Groups	104

48
1995 51

56

x Contents

4.5.12	Prediction Units and Transform Units	104	6.3.2	Forward I
4.5.12.1	Prediction Unit Structures	105	6.3.3	Backward
4.5.12.2	Transform Unit, Transform Block and Transform Sub-Block	106	6.3.4	Bipredicti
4.5.13	HEVC Structures Example	107	6.4	Inter Prec
4.6	Structures in Versatile Video Coding/H.266	111	6.5	Motion V
4.6.1	Sequence Down to Picture	111	6.5.1	How Doe
4.6.2	Picture Down to Coding Unit	111	6.6	Sub-Pixel
4.7	Conclusion	114	6.6.1	What Doe
	Reference	114	6.6.2	Interpolat
5	Intra Prediction	115	6.6.3	Trade-Off
5.1	Introduction	115	6.7	Reference
5.2	The Intra Prediction Process	118	6.7.1	Reference
5.3	Intra Prediction Modes	118	6.7.2	Reference
5.3.1	DC Prediction	120	6.7.2.1	Low-Dela
5.3.2	Planar Prediction	122	6.7.2.2	Random /
5.3.3	Directional Prediction	123	6.7.3	Reference
5.3.4	Sample Filtering	126	6.7.3.1	Adding Pi
5.4	Prediction Block Sizes	126	6.7.3.2	Removing
5.5	Signalling Intra Prediction Choices	128	6.7.3.3	Ordering
5.6	Choosing a Prediction	129	6.7.3.4	Communi
5.7	HEVC Intra Prediction	130	6.8	Signalling
5.7.1	HEVC Intra Prediction Modes	131	6.8.1	Inter Pred
5.7.2	HEVC Intra Prediction Filtering	132	6.8.2	Signalling
5.7.3	Signalling HEVC Intra Prediction Modes	133	6.8.3	Signalling
5.7.3.1	HEVC Luma Prediction Mode Signalling	133	6.8.4	Signalling
5.7.3.2	HEVC Chroma Prediction Mode Signalling	133	6.9	Skip Mode
5.7.4	HEVC Intra Prediction Examples	134	6.10	Loop Filte
5.8	VVC Intra Prediction	139	6.11	When Int
5.8.1	Intra Prediction Modes	139	6.12	HEVC Int
5.8.2	Intra Prediction of Rectangular Blocks	141	6.12.1	Overview
5.8.3	Intra Interpolation Filters	141	6.12.1.1	HEVC Int
5.8.4	Cross-Component Prediction	142	6.12.2	HEVC Int
5.8.5	Extra Reference Lines	142	6.12.2.1	Slice Type
5.8.6	Intra Sub-Partitions	142	6.12.2.2	Symmetri
5.8.7	VVC Intra-coding Performance	143	6.12.2.3	Asymmeti
5.9	Conclusions	144	6.12.2.4	CU Partiti
	References	144	6.12.3	HEVC Int
6	Inter Prediction	145	6.12.3.1	Skip or M
6.1	Introduction	145	6.12.3.2	Inter-PU C
6.2	Inter Prediction – the Basics	146	6.12.3.3	HEVC Int
6.2.1	Motion Vectors for Inter Prediction: The Terminology Problem	152	6.12.4	HEVC Int
6.3	Forward, Backward and Biprediction	155	6.12.5	HEVC Bip
6.3.1	Picture or Slice Types	155	6.12.6	HEVC Sul
			6.12.6.1	Luma Sarr
			6.12.6.2	Chroma S

k 106

6.3.2	Forward Prediction from Past Frames	155
6.3.3	Backward Prediction from Future Frames	156
6.3.4	Biprediction from Two Frames	158
6.4	Inter Prediction Block Sizes	159
6.5	Motion Vectors	162
6.5.1	How Does the Encoder Choose Each Motion Vector?	167
6.6	Sub-Pixel Interpolation	167
6.6.1	What Does a Sub-Pixel Interpolated Reference Frame Look Like?	170
6.6.2	Interpolation Filters	171
6.6.3	Trade-Offs in Sub-Pixel Interpolation	173
6.7	Reference Pictures	176
6.7.1	Reference Pictures for Inter Prediction	176
6.7.2	Reference Picture Structures	181
6.7.2.1	Low-Delay Picture Structures	181
6.7.2.2	Random Access Picture Structures	182
6.7.3	Reference Picture Buffers	183
6.7.3.1	Adding Pictures to the Reference Picture Buffer	184
6.7.3.2	Removing Pictures from the Reference Picture Buffer	184
6.7.3.3	Ordering Reference Pictures	184
6.7.3.4	Communicating Reference Picture Structures	186
6.8	Signalling Inter Prediction Choices	186
6.8.1	Inter Prediction Parameters	186
6.8.2	Signalling the Base Motion Vector Parameters	189
6.8.3	Signalling the Delta Motion Vector Parameters	191
6.8.4	Signalling Reference Picture Choices	191
6.9	Skip Mode	192
6.10	Loop Filter	193
6.11	When Inter Prediction Does Not Find a Good Match	194
6.12	HEVC Inter Prediction	197
6.12.1	Overview	197
6.12.1.1	HEVC Inter Prediction Example	198
6.12.2	HEVC Inter prediction Unit Partitioning	200
6.12.2.1	Slice Types and PU Types	200
6.12.2.2	Symmetric PU Partitions	200
6.12.2.3	Asymmetric PU Partitions	201
6.12.2.4	CU Partitionings into Inter-PUs: Examples	202
6.12.3	HEVC Inter-PU Coding	204
6.12.3.1	Skip or Merge Coding	205
6.12.3.2	Inter-PU Coding: Advanced Motion Vector Prediction	208
6.12.3.3	HEVC Inter-PU Coding: Examples	210
6.12.4	HEVC Inter Prediction Sources	212
6.12.5	HEVC Biprediction and Weighted Prediction	213
6.12.6	HEVC Sub-Pixel Interpolation	213
6.12.6.1	Luma Sample Interpolation	213
6.12.6.2	Chroma Sample Interpolation	215

oblem 152

6.12.7	HEVC Reference Pictures	219
6.13	Inter Prediction in VVC	219
6.13.1	Inter Prediction Partitions	219
6.13.2	Extended Merge Mode	220
6.13.3	Affine Motion-Compensated Prediction	220
6.13.4	Biprediction Enhancements	221
6.13.5	Sub-Pixel Interpolated Motion Compensation	221
6.14	Conclusions	222
	References	222

7	Transform and Quantisation	223
7.1	Introduction	223
7.1.1	Transform: An Overview	224
7.1.2	Quantisation: An Overview	225
7.2	Residual Blocks	226
7.3	Block Transforms	228
7.3.1	What Is the Purpose of a Block Transform?	228
7.3.2	Spatial Frequencies and Basis Patterns	230
7.3.3	What Does the Transform Do?	231
7.3.4	Types of Block Transform	233
7.3.4.1	The Karhunen-Loeve Transform	233
7.3.5	The Discrete Cosine Transform	233
7.3.6	The Hadamard Transform	234
7.3.7	Discrete Sine Transform	234
7.3.8	The Discrete Cosine Transform in Detail	234
7.3.9	The Discrete Cosine Transform: What It Means	240
7.3.10	The Discrete Cosine Transform: Significant Coefficients	244
7.4	Quantisation	246
7.4.1	Combining Transform and Quantisation	249
7.4.2	Designing the Quantiser	251
7.5	Transform and Quantisation in Practice	253
7.6	HEVC Transform and Quantisation	254
7.6.1	Overview of HEVC Transform and Quantisation	254
7.6.2	The HEVC Transforms	256
7.6.2.1	DCT-Based Transforms	256
7.6.2.2	DST-Based Transform	258
7.6.2.3	The HEVC Forward and Inverse Transform and Quantisation Processes	259
7.6.3	HEVC Quantisation and Rescaling	260
7.6.3.1	HEVC QP and Qstep	261
7.6.3.2	HEVC Forward Quantiser	264
7.6.3.3	HEVC Rescaling	264
7.6.3.4	HEVC Quantisation Matrices	265
7.6.4	Example: The Complete HEVC Transform and Quantise Processes	267
7.6.4.1	Forward Transform and Quantise	268
7.6.4.2	Rescale and Inverse Transform	269

7.6.5	Bypassing Tr
7.6.5.1	Transform By
7.6.5.2	Transform an
7.6.5.3	PCM Mode
7.7	Transform an
7.7.1	Transforms in
7.7.2	Quantisation
7.8	Conclusions
	References

8	Entropy Codin
8.1	Introduction
8.2	Entropy Codin
8.2.1	Entropy Codin
8.2.2	Types of Entrc
8.2.2.1	Fixed-Length
8.2.2.2	Variable-Leng
8.2.2.3	Arithmetic Co
8.2.2.4	Binary Arithm
8.2.3	Designing an l
8.2.3.1	Compression I
8.2.3.2	Computational
8.2.3.3	Synchronisati
8.3	Pre-processing
8.3.1	Pre-processing
8.4	Probability Mo
8.5	Variable-Lengt
8.5.1	Unary Coding
8.5.2	Exponential G
8.5.3	Huffman Codin
8.5.4	Precalculated V
8.5.5	Context-Adapti
8.6	Arithmetic Coc
8.7	Binary Arithme
8.7.1	Binarisation
8.7.2	Encoding a Bin
8.7.3	Decoding a Bin
8.7.4	BAC Encoding
8.7.4.1	All the Possible
8.7.5	Binary Fraction
8.7.6	Sending Bits: B
8.7.6.1	Renormalisati
8.7.7	Receiving Bits a
8.7.8	Binary Arithme
8.8	Context-Adapti

7.6.5	Bypassing Transform and/or Quantisation in HEVC	271
7.6.5.1	Transform Bypass	271
7.6.5.2	Transform and Quantisation Bypass	271
7.6.5.3	PCM Mode	272
7.7	Transform and Quantise in H.266 Versatile Video Coding	272
7.7.1	Transforms in H.266	272
7.7.2	Quantisation in H.266	272
7.8	Conclusions	273
	References	273
8	Entropy Coding	275
8.1	Introduction	275
8.2	Entropy Coding for Video Compression	278
8.2.1	Entropy Coding to Maximise Compression Efficiency	278
8.2.2	Types of Entropy Coding	280
8.2.2.1	Fixed-Length Coding	280
8.2.2.2	Variable-Length Coding	281
8.2.2.3	Arithmetic Coding	281
8.2.2.4	Binary Arithmetic Coding	281
8.2.3	Designing an Entropy Coder	282
8.2.3.1	Compression Efficiency	282
8.2.3.2	Computational Complexity	282
8.2.3.3	Synchronisation	282
8.3	Pre-processing	283
8.3.1	Pre-processing Examples	283
8.4	Probability Models and Context Adaptation	285
8.5	Variable-Length Coding	286
8.5.1	Unary Coding	287
8.5.2	Exponential Golomb Coding	287
8.5.3	Huffman Coding	289
8.5.4	Precalculated VLC Tables	290
8.5.5	Context-Adaptive VLC	290
8.6	Arithmetic Coding	291
8.7	Binary Arithmetic Coding	294
8.7.1	Binarisation	295
8.7.2	Encoding a Bin	296
8.7.3	Decoding a Bin	297
8.7.4	BAC Encoding and Decoding: Example	298
8.7.4.1	All the Possible Coding Outcomes	299
8.7.5	Binary Fractions	302
8.7.6	Sending Bits: BAC Encoder	303
8.7.6.1	Renormalisation and Outputting Bits at the Encoder	304
8.7.7	Receiving Bits at the BAC Decoder	307
8.7.8	Binary Arithmetic Coding: Putting It All Together	308
8.8	Context-Adaptive Binary Arithmetic Coding (CABAC)	309

244

tion Processes 259

Processes 267

8.8.1	The CABAC Encoder	310	9.6	Concluding Reference
8.8.2	The CABAC Decoder	311	10	Storing and Retrieving
8.8.3	CABAC Context Models	311	10.1	Introduction
8.8.4	CABAC Context Adaptation	312	10.2	Storing and Retrieving
8.9	Entropy Coding in HEVC	314	10.3	Coded Video
8.9.1	Pre-processing in HEVC	315	10.3.1	The MPEG-4
8.9.1.1	HEVC Transform Blocks and Sub-blocks	315	10.4	Transport
8.9.1.2	Coding Transform Sub-blocks	318	10.4.1	Broadcast
8.9.1.3	Pre-processing HEVC Prediction Parameters	321	10.4.2	Video Coding
8.9.1.4	Pre-processing HEVC Quantisation Parameters	321	10.4.3	Video Streaming
8.9.2	HEVC Binarisation	322	10.4.3.1	Progressive
8.9.3	The HEVC Binary Arithmetic Coder	322	10.4.3.2	Streaming
8.9.3.1	HEVC BAC Encoding	322	10.4.3.3	DASH and
8.9.3.2	HEVC BAC Decoding	325	10.4.3.4	Content
8.9.3.3	HEVC BAC Coding Example	327	10.5	Video Rate
8.9.4	HEVC Context Modelling	329	10.6	Error Handling
8.9.5	HEVC Context Models and Initialisation	331	10.7	Concluding Reference
8.10	Entropy Coding in H.266/VVC	334	11	Implementing
8.11	Conclusion	334	11.1	Introduction
	References	335	11.2	Implementing
9	Coded Video Filtering	337	11.3	Software Implementation
9.1	Introduction	337	11.3.1	Comparing
9.2	Filtering and Video Coding	339	11.4	Hardware
9.3	Detecting and Correcting Video Coding Artefacts	341	11.5	Video Coding
9.3.1	Blocking Artefacts	341	11.5.1	Measuring
9.3.1.1	Characteristics of Blocking Artefacts	341	11.5.2	Measuring
9.3.1.2	Causes of Block Artefacts	342	11.5.3	Optimising
9.3.1.3	Detecting and Correcting Block Artefacts	343	11.5.3.1	Computational
9.3.2	Ringing Artefacts	344	11.5.3.2	Rate-Distortion
9.3.2.1	Characteristics and Causes of Ringing Artefacts	344	11.5.3.3	Software Complexity
9.3.2.2	Detecting and Correcting Ringing Artefacts	345	11.6	Getting Started
9.3.3	Magnitude Offset Artefacts	345	11.6.1	Installing
9.3.3.1	Characteristics and Causes of Magnitude Offset Artefacts	345	11.6.2	Source Video
9.3.3.2	Detecting and Correcting Magnitude Offset Artefacts	347	11.6.3	Adjusting
9.4	HEVC In-Loop Filtering	347	11.6.4	Encode, Decode
9.4.1	Deblocking Filter	348	11.6.5	Measure Rate
9.4.1.1	HEVC Deblocking Filter Example	350	11.6.6	Analysing
9.4.2	Sample Adaptive Offset (SAO) Filter	351	11.7	Concluding Reference
9.4.2.1	SAO Edge Offset	352		
9.4.2.2	SAO Band Offset	352		
9.5	VVC Filtering	354		

9.6	Conclusions	355
	References	355
10	Storing and Transporting Coded Video	357
10.1	Introduction	357
10.2	Storing and Delivering Coded Video	359
10.3	Coded Video File Formats	361
10.3.1	The MPEG-4 File Format	363
10.4	Transport of Coded Video	365
10.4.1	Broadcast	365
10.4.2	Video Calling	366
10.4.3	Video Streaming	367
10.4.3.1	Progressive Download	367
10.4.3.2	Streaming Servers	369
10.4.3.3	DASH and HLS: HTTP Adaptive Bitrate Streaming	370
10.4.3.4	Content Delivery Networks	373
10.5	Video Rate Control	374
10.6	Error Handling	379
10.7	Conclusions	381
	References	382
11	Implementation and Performance	383
11.1	Introduction	383
11.2	Implementing Video Codecs	384
11.3	Software Implementation	384
11.3.1	Comparing HM and x265 Encoding	387
11.4	Hardware Implementation	388
11.5	Video Codec Performance	390
11.5.1	Measuring Codec Performance	390
11.5.2	Measuring Rate-Distortion Performance	392
11.5.3	Optimising Video Codec Performance	394
11.5.3.1	Computational Optimisation	394
11.5.3.2	Rate-Distortion Optimisation	394
11.5.3.3	Software Optimisation	397
11.6	Getting Started with Experiments	397
11.6.1	Installing Software	398
11.6.2	Source Video Material	398
11.6.3	Adjusting Source Video Clips	399
11.6.4	Encode, Decode and Play Video Clips	399
11.6.5	Measure Rate-Distortion Performance	400
11.6.6	Analysing Coded Bitstreams	402
11.7	Conclusion	405
	References	405

12	Conclusions	407
12.1	What This Book Has and Has Not Covered	407
12.2	Where Is Video Coding Going Next?	408
12.3	Where Should You Go Next?	409
	References	409
	Glossary	411
	Index	415

About the Author

Iain E. Richardson is a video communication include two widely ci thirty years, he has c communications, as, e and as an independe and delivers lectures c ness in litigation cases

About the Author

Iain E. Richardson is an internationally recognised expert on video compression and digital video communications. He is the author of four other books about video coding, which include two widely cited books on the H.264 Advanced Video Coding standard. For over thirty years, he has carried out research in the field of video compression and video communications, as a Professor at the Robert Gordon University in Aberdeen, Scotland, and as an independent consultant with his own company, Vcodex. He advises companies and delivers lectures on video compression technology and is sought after as an expert witness in litigation cases involving video coding.

About the Companion Website

This book is accompanied by a companion website

www.wiley.com/go/richardson/codingvideo1



I hope that at least
out and concentrate
, which will give an
m or model. From
asic concepts of the
apter then explains
th a particular focus

i here. For example,
nicated in H.265 or
pics further, such as
which has published
a special section on
y about a particular
cument itself should

if a topic. Chapter 11
ndard-based codecs.
s to understand how
visualise the way the
v.codex.com/coding-

ideo, starting with an
to compress.

HEVC) standard and
logy 22 (12): 1646–1646.

1 to the special section
Video Technology
2.

2

Video Coding and Video Quality

2.1 Introduction

Video coding or video compression bridges the gap between displayable or uncompressed video and practical methods of storing or transmitting data. Digital video is made up of a series of frames or pictures, each of which contains hundreds of thousands or millions of picture elements or pixels. Storing or transmitting video in its original, uncompressed form can require an impractically large amount of storage capacity or bandwidth. A video encoder compresses video into a smaller and more compact form known as coded video, suitable for storage or transmission. A video decoder reverses this process, extracting displayable video from coded video (see Figure 2.1). The complementary pair of an encoder and a decoder is a video codec.

In this chapter, I will introduce each of the following aspects of a video codec, before covering them in further detail throughout the book:

Inputs and outputs: The data inputs and outputs for a video codec. The input to an encoder and the output of a decoder are uncompressed or raw video data, i.e. video that can be displayed. The output of an encoder and the input to a decoder are an encoded, compressed bitstream that is suitable for storage or transmission.

Data structures: A video codec processes data in a hierarchy from a complete video sequence, e.g. a video scene or programme, through groups of coded frames and individual coded frames, down to basic coding units (Coding Tree Units [CTUs], Macroblocks [MBs] or Superblocks) and sub-blocks.

Prediction: Some or all of the data to be coded are predicted from previously coded data. We can predict a block of video data from neighbouring data in the same video frame. This is known as intra prediction. Inter prediction uses video data from previously coded frames to predict a block of video data. Once the prediction is created, it is subtracted from the original block to form a residual block, which typically contains less information than the original block.

Transform and quantisation: Blocks of video data or residual data are transformed into another form, such as a frequency domain representation, and then quantised to reduce the data precision. This has the effect of removing or discarding much of the information in a block, at the expense of a loss of visual quality.

Coding Video: A Practical Guide to HEVC and Beyond, First Edition. Iain E. Richardson.
© 2024 John Wiley & Sons Ltd. Published 2024 by John Wiley & Sons Ltd.
Companion website: www.wiley.com/go/richardson/codingvideo1

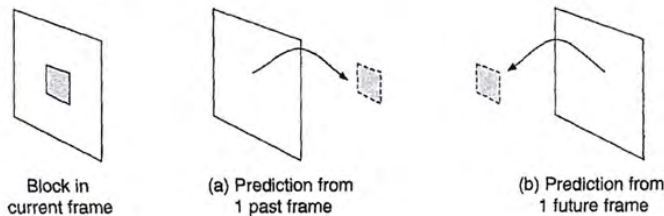


Figure 6.24 Prediction from a past or future frame

$N + 1$, and so the encoder predicts Block 2 from the future frame, $N + 1$. The reference frame for Block 1 is frame $N - 1$, and the reference frame for Block 2 is frame $N + 1$.

Figure 6.24 illustrates the two options of creating a prediction from a past frame – forward prediction, and creating a prediction from a future frame – backward prediction.

6.3.4 Bipediction from Two Frames

A third option is to create an inter prediction by combining pixels in two reference frames. Figure 6.25 shows current frame N , which is part-way through a dissolve or fade between two scenes, a crowd scene shown in Frame $N - 1$ and a city scene shown in Frame $N + 1$. The highlighted block contains elements of both scenes – it is a composite of a girl's shoulder from the crowd scene and a building from the cityscape. It may not be possible to create an accurate prediction from either the past frame $N - 1$ or the future frame $N + 1$. However, the encoder might be able to create a good prediction by combining or averaging pixels from the past frame and the future frame. This is known as bipediction.

A bipediction block is created by averaging each pixel of two source blocks, each from a different frame, as shown in Figure 6.26. The encoder identifies a block in the first frame, Source 1, and a block in the second frame, Source 2. The bipediction block is formed by averaging each pair of pixels from Source 1 and Source 2. This bipediction block is then subtracted from the original block to create a prediction residual. Bipediction, predicting a block from two reference frames, can often give a better prediction than uni-directional prediction, which predicts from a single reference frame. However, the encoder needs to

Figure 6.25 Bipediction example: create block in frame N as average of past and future regions

Figure 6.26 Crea

Figure 6.27
future frame

send offsets to two pr
bipediction may requ

Bipediction has bee
MPEG-2 video coding
was restricted to crea
frame (Figure 6.27). R
ibility so that a bipre
from two past frames
more prediction optio
three or more frames.
pictures are handled,

6.4 Inter Predi

In the previous sectic
frame from a region i
of this block?

Figure 6.26 Creating a bipredicted block

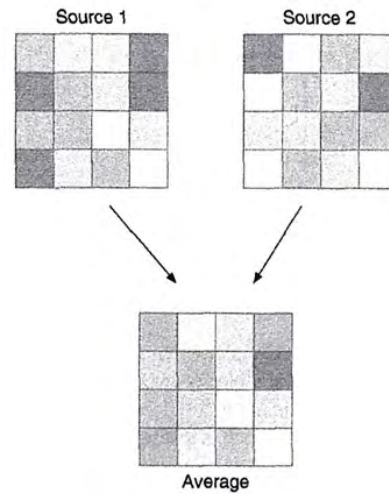
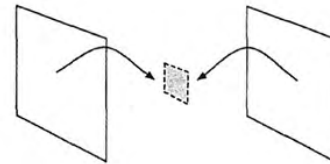


Figure 6.27 Biprediction: one past and one future frame



send offsets to two prediction source blocks, e.g. Source 1 and Source 2 in Figure 6.26, so biprediction may require more bits to signal the choice of prediction source.

Biprediction has been a feature in video coding since the publication of the MPEG-1 and MPEG-2 video coding standards in the early 1990s. In these early standards, the encoder was restricted to creating a biprediction using a specific past frame and a specific future frame (Figure 6.27). Recent standards such as H.264 and HEVC give the encoder more flexibility so that a bipredicted block can be created from one past and one future frame, or from two past frames, or from two future frames, as illustrated in Figure 6.28. In theory, more prediction options are possible, such as creating a prediction by combining data from three or more frames. Again, biprediction has implications for the way in which reference pictures are handled, which we will discuss further in Section 6.7.

6.4 Inter Prediction Block Sizes

In the previous sections, we saw how inter prediction predicts a block within the current frame from a region in a previously coded frame. How should the encoder choose the size of this block?

on from
frame

-1. The reference frame
is $N + 1$.
is a past frame – forward
and prediction.

ixels in two reference
through a dissolve or
nd a city scene shown
h scenes – it is a com-
from the cityscape. It
he past frame $N - 1$ or
eate a good prediction
future frame. This is

rice blocks, each from a
block in the first frame,
tion block is formed by
rediction block is then
Biprediction, predicting
on than uni-directional
r, the encoder needs to

Frame $N + 1$

past and future regions

reference frame in a position two pixels left and one pixel down. If we want to predict the block containing the ball, we can use a motion vector $(-2, -1)$, and we will create an accurate prediction.

Unfortunately, objects in a scene rarely move in neat, integer steps. In Figure 6.40, the ball in the current frame is in the same position as the previous example. However, the ball in the reference frame can be found 1.5 pixels to the left and 0.5 pixels down, i.e. the ball has moved by a non-integer number of pixels from one frame to the next. In a natural video scene, objects will tend to move by arbitrary amounts, for example, by 0.87 pixels, -14.92 pixels, etc.

We can create a better prediction in the example of Figure 6.40 if we can make it possible to send a non-integer motion vector, in this case $(-1.5, +0.5)$. We can do this by re-sampling the reference frame to a higher resolution, for example, by re-sampling at half-pixel positions as shown in Figure 6.41 and choosing a motion vector that points to pixels in the re-sampled reference frame.

You may have noticed that the above examples are simplifications of the real situation. In reality, the ball would appear much more pixelated than shown. Let us look at a more realistic example. Figure 6.42 shows a section of a current frame with a dark vertical line that is two pixels wide. The same line exists in the reference frame shown on the left of the figure, but because of fractional-pixel movement from one frame to the next, the line appears smeared across three horizontal pixels. If the encoder tries to predict a 4×4 block in the top left of the region, Figure 6.43, right, it will fail to find a good match at an integer-offset position, Figure 6.43, left.

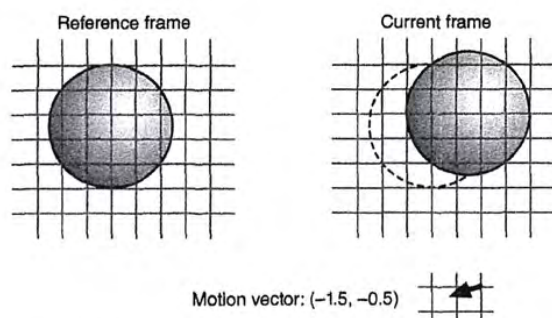


Figure 6.40 Object moving by a non-integer number of pixels

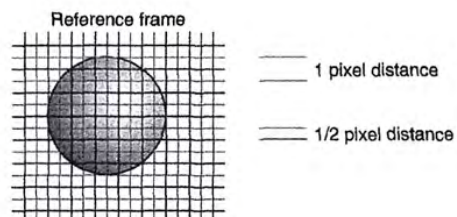


Figure 6.41 Reference frame sampled to $1/2$ -pixel positions

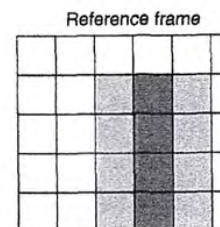


Figure 6.42 Reference fr

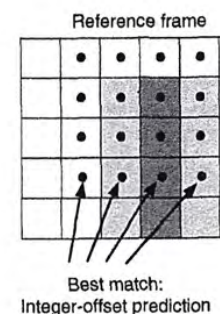


Figure 6.43 Block to be

Figure 6.44 shows the
nal, integer pixel positio
are interpolated from n
the number of samples

The encoder can now
We can find a better m

Figure 6.44 Reference fr
interpolated to half-pixel

we want to predict the
we will create an accu-

ps. In Figure 6.40, the
nple. However, the ball
xels down, i.e. the ball
next. In a natural video
, by 0.87 pixels, -14.92

we can make it possible
do this by re-sampling
pling at half-pixel posi-
points to pixels in the

ns of the real situation.
1. Let us look at a more
with a dark vertical line
shown on the left of the
ie to the next, the line
s to predict a 4x4 block
od match at an integer-

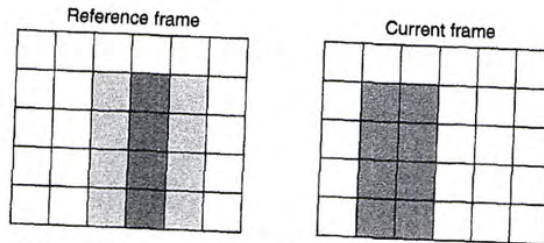


Figure 6.42 Reference frame and current frame, integer pixel positions

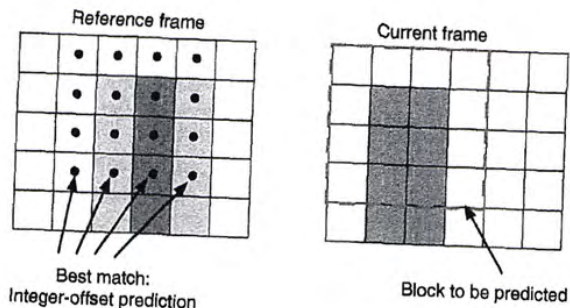
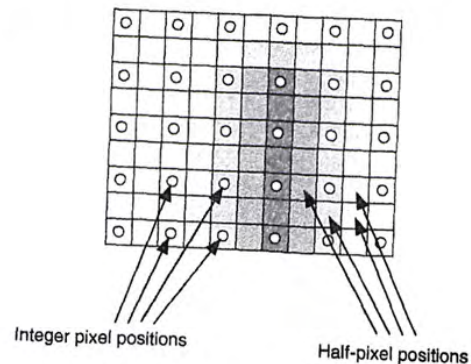


Figure 6.43 Block to be predicted, best match, integer motion vector

Figure 6.44 shows the reference frame area interpolated to half-pixel accuracy. The original, integer pixel positions are marked with white circles. The missing half-pixel positions are interpolated from neighbouring integer pixel positions. We have interpolated to twice the number of samples in the horizontal and vertical positions.

The encoder can now search for a 4x4 block within this expanded, re-sampled region. We can find a better match in the position shown in Figure 6.45. The motion vector that

Figure 6.44 Reference frame interpolated to half-pixel positions



reference frame sampled to
ions

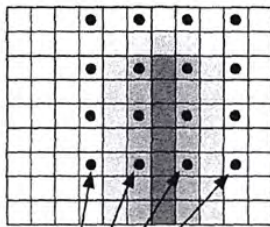


Figure 6.45 Best match in re-sampled reference frame

Best match:
Half-pixel prediction

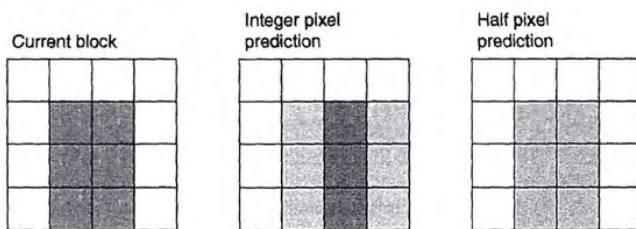


Figure 6.46 Result of integer and half-pixel prediction

points to this block is $(+1.5, 0)$. This is not a perfect match because the re-sampled pixel values are not identical to the current frame pixel values, but it is a better match than the integer position shown in Figure 6.43.

Figure 6.46 shows the current 4×4 block and the results of creating an integer-pixel prediction and a half-pixel prediction. The integer prediction is clearly not a good match for the original block. The half-pixel prediction shows the 'best match' sample values selected from the half-pixel interpolated grid, i.e. the black dots in Figure 6.45. The half-pixel prediction is a better match, but it is not an exact match. The brightness or luma levels of the interpolated pixel positions are not identical to the original pixels, because these interpolated values are created by averaging integer pixel values, as shown in Figure 6.44. In a sense, the actual pixel values we are searching for do not exist in the reference frame. At most, sub-pixel prediction can create interpolated samples that are closer to the pixel values in the current frame, as this example shows.

6.6.1 What Does a Sub-Pixel Interpolated Reference Frame Look Like?

On the left-hand side of Figure 6.47 is a close-up of a Quarter Common Intermediate Format (QCIF) frame, resolution 176×144 pixels, without any interpolation. Interpolating the frame to half-pixel positions, shown as the middle image, creates four times the number of pixels. This interpolated image looks smoother, but there is no new information in it. In



Close-up of QCIF fra

Figure 6.47 QCIF frame

contrast, a higher-res actual detail. For exam clearer and more deta though there is the whole frame.

6.6.2 Interpolation

Interpolated sub-pixe cally using filtering. T effect on the quality pixel close-up of a QC parison, Figure 6.49 s Note that the QCIF fra

Figure 6.48 QCIF
Foreman sequence

1 reference frame

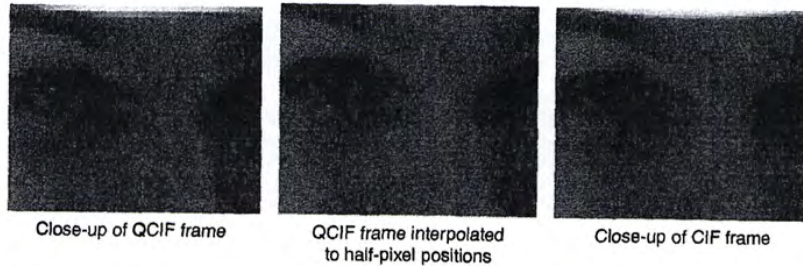


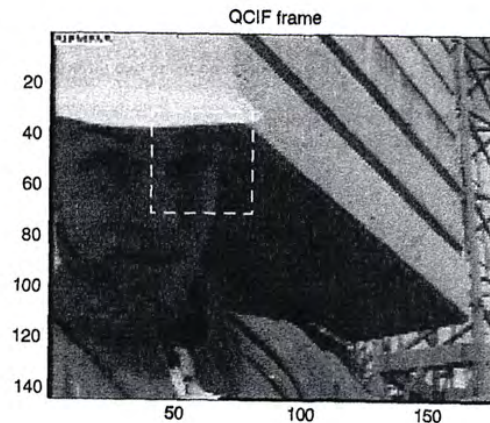
Figure 6.47 QCIF frame close-up, interpolated version, CIF version

contrast, a higher-resolution version of the original frame, shown at the right, has more actual detail. For example, the eye in an actual higher-resolution CIF frame on the right is clearer and more detailed than the eye in the interpolated QCIF frame in the middle, even though there is the same number of pixels in both images, i.e. 352×288 pixels for the whole frame.

6.6.2 Interpolation Filters

Interpolated sub-pixel positions are created by combining neighbouring pixel values, typically using filtering. The way in which neighbouring values are combined or filtered has an effect on the quality of the interpolation. For example, on the left of Figure 6.49 is a 40×40 pixel close-up of a QCIF frame from the *Foreman* sequence, shown in Figure 6.48. For comparison, Figure 6.49 shows the same region from the CIF version of the frame on the right. Note that the QCIF frame is significantly less detailed than the CIF frame, as we would expect.

Figure 6.48 QCIF frame from *Foreman* sequence



se the re-sampled pixel
a better match than the

reating an integer-pixel
early not a good match
match' sample values
1 Figure 6.45. The half-
the brightness or luma
original pixels, because
d values, as shown in
for do not exist in the
olated samples that are
ows.

ook Like?

Common Intermediate
rpolation. Interpolating
s four times the number
ew information in it. In

Notice that even though the step size is the same, Quantiser (b) produces more zero-valued coefficients than Quantiser (a). We would therefore expect Quantiser (b) to compress video more and to introduce more distortion in the decoded video frames than Quantiser (a).

Instead of quantising every coefficient in the same way, it can be useful to weight the quantisation step size so that lower-frequency coefficients are quantised less and higher-frequency coefficients are quantised more. An example of frequency-weighted quantisation is shown in Figure 7.40. The quantiser step size increases towards the bottom right of an 8×8 block, where the coefficient frequencies are higher. Lower-frequency coefficients, towards the top left of the block, tend to be more important for visual quality, and a weighted quantiser can retain more information in the important lower frequencies whilst removing more information from the higher frequencies.

16	16	16	16	17	18	20	24
16	16	16	17	18	20	24	25
16	16	17	18	20	24	25	28
16	17	18	20	24	25	28	33
17	18	20	24	25	28	33	41
18	20	24	25	28	33	41	54
20	24	25	28	33	41	54	71
24	25	28	38	41	54	71	91

Figure 7.40 Frequency-weighted quantisation

7.5 Transform and Quantisation in Practice

The transform and quantisation stages in a video codec are closely connected. The transform does not provide compression by itself but is designed to work with the subsequent quantisation stage, which quantises each transform coefficient.

In a practical codec, the transform and quantiser should be designed with certain goals in mind, including:

- Maximising rate-distortion performance – maximum quality and minimum compressed bitrate.
- Minimising computational complexity.
- Ensuring compatibility between encoders and decoders.
- Providing flexibility, for example, in the selection of transform size and quantiser step size.

In recently developed video coding standards, these goals are satisfied by approaches such as the following:

- Specify a transform that can be implemented using a minimal number of basic arithmetic operations such as additions, subtractions and binary shifts, with as few multiplications as possible since these tend to be more computationally expensive than additions and shifts.
- Specify a transform that can be efficiently implemented in different ways to maximise efficiency on different software and hardware platforms.
- Specify a particular approximation to transform weighting factors such as the irrational numbers in the 4×4 DCT, as shown in Eq. (7.7).

and rescaled to zero; to +8; and so on. a different mapping in Figure 7.39). With then rescaled to zero. rescaled to +8. Original to -8, and so on. This 7.38 since more coefficients be quantised to -1 of values -7 to +7 that tised to 0 and rescaled

- Avoid divisions and fractional arithmetic if possible.
- Limit the number of bits of precision required during computation. For example, if the input samples are 8 bits, what is the maximum dynamic range required during the computation of the transform and quantisation stages?
- Share aspects of the design across multiple transform sizes where possible.
- Specify the quantiser to give an effective level of control of the trade-off between quantiser step size, bitrate and quality.

We will examine the transform and quantiser design in the HEVC/H.265 standard and see how these goals are achieved in practice.

7.6 HEVC Transform and Quantisation

We saw in Chapter 4 how each HEVC coding unit (CU) is partitioned into one or more transform units (TUs). In a colour video sequence, each TU comprises a luma transform block (TB) and two chroma TBs. Luma TBs can range from 4×4 to 32×32 samples in size.

Each CU or TB can be processed in one of the following ways, as illustrated in Figure 7.41:

- 1) Transform the TB using a block transform and quantise the resulting coefficients. Code the coefficients using transform sub-block (TSB) coding.
- 2) Bypass the transform and quantise the TB samples directly. Code the coefficients, 4×4 at a time, using TSB coding, see Chapter 8.
- 3) Bypass both the transform and the quantisation stages and pass the TB samples directly for TSB coding.
- 4) In pulse code modulation (PCM) mode, directly code the CU samples in the bitstream.

The first option is probably the most common approach to coding a TB. We will discuss this in detail next.

7.6.1 Overview of HEVC Transform and Quantisation

The classic block-based transform and quantisation process is shown at the top of Figure 7.42. A block of samples X is transformed, for example, using a two-dimensional DCT, into a block of coefficients Y . The coefficients are quantised, for example, by dividing by a quantiser step and rounding, to produce a block of quantised levels L .

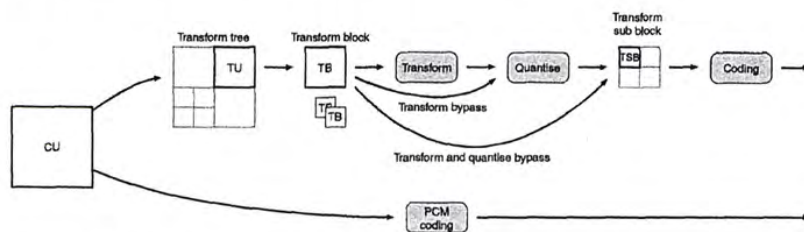


Figure 7.41 CU and TB processing in HEVC

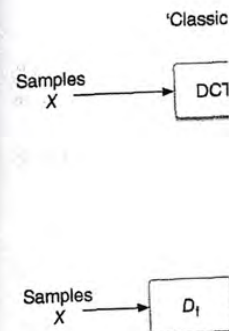


Figure 7.42 HEVC forward transform

The equivalent process for a block of samples is transformed into a block of coefficients Y . The coefficients are quantised, for example, by dividing by a quantiser step and rounding, to produce a block of quantised levels L .

A classic inverse process is rescaled, for example, by multiplying the coefficients Y' . The coefficients are then transformed back into a block of samples X using an inverse DCT, to produce the reconstructed samples X' .

The equivalent process for a block of samples is rescaled, for example, by multiplying the coefficients Y' . The coefficients are then transformed back into a block of samples X using an inverse discrete cosine transform (IDCT), to produce the reconstructed samples X' .

We will consider how the 'classic' transform and quantisation process is implemented in the HEVC standard.

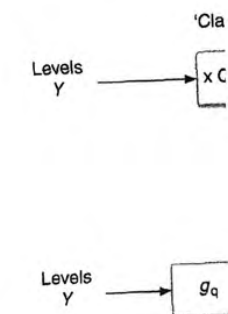


Figure 7.43 HEVC residual processing

on. For example, if the
quired during the com-

: possible.

ade-off between quan-

/C/H.265 standard and

oned into one or more
rises a luma transform
32 × 32 samples in size.
ways, as illustrated in

ilting coefficients. Code

ie the coefficients, 4 × 4

the TB samples directly

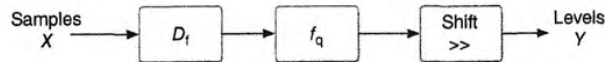
mples in the bitstream.

ig a TB. We will discuss

s shown at the top of
sing a two-dimensional
for example, by dividing
levels L .



'Classic' forward DCT and Quantise



HEVC implementation

Figure 7.42 HEVC forward transform and quantise

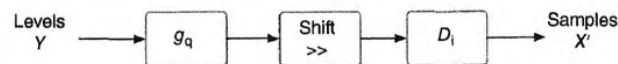
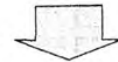
The equivalent process in an HEVC encoder is shown at the bottom of Figure 7.42. The block of samples is transformed by D_t , a scaled approximate DCT. At the output of the transform, each coefficient is multiplied by a factor f_q and then bit-shifted down to the correct dynamic range to produce output level block L .

A classic inverse process is shown at the top of Figure 7.43. Each quantised level in block L is rescaled, for example, by multiplying with quantiser step Q_{step} , to produce a block of coefficients Y' . The coefficient block is inverse-transformed, using a two-dimensional inverse DCT, to produce an output block X' .

The equivalent process in an HEVC decoder, shown at the bottom of Figure 7.43, applies a rescaling factor g_q followed by a bit shift to rescale the coefficients. A scaled approximate inverse discrete cosine transform (IDCT) D_i processes an output block X' .

We will consider how the HEVC forward and inverse processes are developed from the 'classic' transform and quantise processes.

'Classic' rescale and inverse DCT



HEVC implementation

Figure 7.43 HEVC rescale and inverse transform

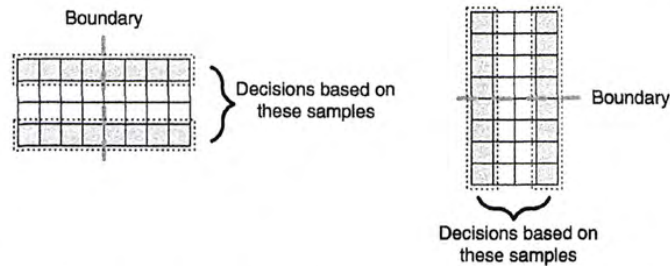
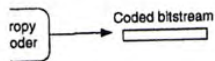


Figure 9.20 H.265 deblocking decision samples

set filters

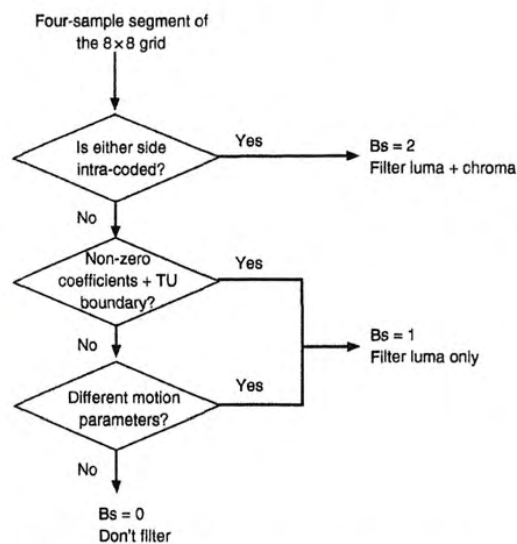


Figure 9.21 H.265 deblocking filter: Boundary Strength Bs

boundaries

four-sample segment and a horizontal four-sample segment. Filtering decisions are made based on the values of 16 samples perpendicular to the four-sample boundary, as shown in Figure 9.20.

A boundary strength parameter Bs is calculated, ranging from 0 to 2 (see Figure 9.21). If either adjacent 4×4 region is intra-coded, Bs is set to 2, and both luma and chroma samples are filtered. Otherwise, if the two adjacent regions have different motion parameters, or if either region has non-zero coefficients and the boundary is a transform block boundary, then Bs is set to 1 and only luma samples are filtered. If none of these conditions are true, Bs is set to 0, and no filtering takes place.

A further decision process (see Figure 9.22) determines whether a strong filter, a weak filter or no filter is applied to the samples on either side of the grid boundary. If Bs is 0 or if the segment is not actually on a TU or PU boundary, no filtering is applied.

king artefacts adjacent to
nstructed video frame [1].
its (PUs) and/or transform
9 shows PUs in an H.265
8×8 grid may be filtered.
ed with x – are not filtered.
dotted lines in Figure 9.19,
re 9.20 shows a vertical