Transcodering van H.264/AVC- en SVC-videostromen in het gecomprimeerde domein

Compressed-Domain Transcoding of H.264/AVC and SVC Video Streams

Jan De Cock

Promotoren: prof. dr. ir. R. Van de Walle, dr. P. Lambert
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. J. Van Campenhout
Faculteit Ingenieurswetenschappen
Academiejaar 2009 - 2010

UNIVERSITEIT GENT

ISBN 978-90-8578-308-4
NUR 965
Wettelijk depot: D/2009/10.500/66

Acknowledgements

Working on my Ph.D. has given me the opportunity to specialize in a challenging topic, to work in a stimulating research group, and to look beyond purely conceptual work by coming into contact with a large number of industrial partners and members of standardization bodies. Undoubtedly, the work presented in this text is a result of many influences and the help and tips of many people.

First of all, I would like to thank my advisor, prof. Rik Van de Walle, for allowing me to work on this topic within Multimedia Lab. I very much appreciated the environment in which I could not only perform theoretical research, but also validate my work in several projects. Furthermore, I would like to thank him for providing me with feedback on both high-level and low-level issues, in the midst of an often hectic schedule.

Multimedia Lab has been shaped by an inspiring group of people, performing research on a wide variety of topics. Even though the lab has grown rapidly over the last years, Multimedia Lab has been able to remain a close and motivated group, where there is more and more 'cross-fertilization', both within and across research topics. I would like to thank my colleagues for providing this enriching atmosphere, not only during working hours, but certainly also after 'business' hours.

A special word of thanks goes to Stijn Notebaert, who has shared my quest to low-level video (trans)coding from the very first steps. We laid many of the first stones of the work on requantization transcoding together. Having a friend and colleague to discuss work -at the most extreme hours of day-, share frustrations and successes, certainly helped to soften the toughest problems. Also, Rita Breems and Ellen Lammens deserve many thanks for taking care of the administrative burden related to my Ph.D.

I am grateful for having gotten the opportunity to attend several conferences and standardization meetings. The inspirational talks with people from other institutions and companies certainly stimulated my research. The standardization work on SVC was a direct incentive for my research on H.264/AVC-to-SVC transcoding. The work on next-generation video

coding within MPEG and VCEG has given me many new insights into video coding, and provided a constant challenge for searching for further coding improvements.
Furthermore, I thank everyone who reviewed my papers and the chapters of this Ph.D. text, and also the members of my jury, who gave me many helpful tips and suggestions to improve its final version.

Also, I would like to thank my friends for all the great times, and for being so patient with me - arranging a get-together wasn't always easy...

Finally, I would like to dedicate this work to the people who mean everything to me.

To my parents and family, who supported me during my education and the most stressful moments.
To Saar, who knows me like no other, taught me more than she can imagine, and pulled me through the most difficult time of my Ph.D. We have shared many amazing moments. I hope many more will follow.

Jan De Cock
September 2009

Summary

The importance and use of video in our society is increasing steadily. Not even fifteen years ago, the bandwidth of networks did not suffice for streaming video (at a decent quality). Nowadays, however, we are confronted daily with video fragments and movies on the Internet. The availability of video has thoroughly changed the content of web sites. Also, more and more video is being created by web site users, leading to the rise of completely new business models and new ways of sharing content. Furthermore, in the last few years, the introduction of digital video at home has lead to an increased availability of content and applications, and to an improved quality of the delivered signals and user experience.

The progress was made possible, to a large extent, by efficient video compression techniques. MPEG-2 Video, which was standardized in the early nineties, and the MPEG-4 Visual format (whose breakthrough was reinforced by the DivX and Xvid implementations) have fostered the proliferation of video fragments and digitized movies.

More recently, H.264/AVC has been standardized by the Joint Video Team of the ISO/IEC MPEG and ITU-T VCEG groups. The H.264/AVC standard further reduces the video bit rate at a given quality when compared to previous specifications, and can be considered as the reference when it comes to video compression. Logically, this was the standard of choice used in this work.

During encoding of video streams it is important to take into account the huge diversity of devices that will decode and play the video streams at the receiver side. Multiple devices like PCs, laptops, cell phones and PDAs are often used to play a single video file. Obviously, these devices have widely varying characteristics, which should be considered when sending video. As an example, sending a high-resolution video stream (720p or 1080p) over a limited-bandwidth network to a device with reduced display capabilities (such as PDAs or GSMs) will unnecessarily burden the network, and lead to high decoding complexity for the device. In such scenarios, a reduction of the bit rate or spatial resolution improves efficiency. This adaptation step can be per-

formed, for example, at an intermediary network node.

Additionally, the diversity of coding standards and formats used in production environments, distribution networks, and broadcast channels (just to name a few, MPEG-1, MPEG-2, H.263, MPEG-4 Visual, VC-1, and H.264/AVC) necessitates efficient conversion techniques. Depending on the targeted application, loss of quality can or cannot be tolerated. For archiving, quality loss is not acceptable, while for limited-bandwidth network distribution a small, albeit unnoticeable, quality loss can be permitted.

This diversity explains the necessity of video adaptation techniques. One way for easy stream adaptation is by using scalable video coding. Scalable coding allows creating a layered representation of the video stream during encoding. For these streams, the lower layer (or base layer) contains the lowest quality. Additional layers (enhancement layers) will contain refinements of the quality (SNR or quality scalability), the spatial resolution (spatial scalability), or frame rate (temporal scalability). Although provisions for scalable coding were already available for MPEG-2 and MPEG-4 Visual, they are rarely used in practice. Recently, SVC was standardized as an extension of H.264/AVC. SVC allows creating scalable streams with minimal quality loss for the same bit rate when compared to single-layer H.264/AVC. Despite these scalability tools, most of the video streams today are still created in a single-layer format. At the moment, it is unclear whether SVC will induce a breakthrough for scalable video coding.

The lack of scalable streams results in the necessity for developing alternative techniques to enable video adaptation. In this dissertation, *transcoding* is used for enabling efficient adaptation of H.264/AVC video streams. Its efficiency is obtained by reusing as much information as possible from the original bitstream, such as mode decisions and motion information. The ultimate goal is to perform the required adaptation process faster than the straightforward concatenation of decoder and encoder.

Chapter 2 deals with H.264/AVC bit rate reduction, also referred to as *transrating*. During encoding, the quality of the resulting stream is mainly determined by the quantization parameter (QP). A high QP indicates coarse quantization of the residual data, which implies a lower quality. Due to the coarser approximation of the residual coefficients, however, the resulting values can be represented in a more compact way and the bit rate of the resulting bit stream is reduced. Based on this knowledge, transrating techniques are developed. Starting from the coefficients in the input bitstream, the QP is increased and the residual coefficients are changed reflecting the new QP value. This technique is referred to as *requantization*. As a result, the bit rate of the outgoing bitstream is reduced and adapted, e.g., to the bandwidth of the

network.

Several complications occur, however, during the adaptation process. In video coding, the high compression ratio is obtained by exploiting different types of redundancy in video sequences. Not only spatial, but also temporal and statistical redundancy can be found in video streams. In order to exploit these, techniques are used such as intra prediction, motion-compensated prediction, variable-size block transforms, and entropy coding. As a result of these techniques, the prediction of frames will depend on prior coded (and decoded) frames or regions in the current frame. By changing the residual data in the current frame, dependent frames will also be affected, resulting in *temporal* drift in the transcoded sequence. In H.264/AVC, a *spatial* drift component will also be present when intra-coded macroblocks are transrated. In this dissertation, the importance of this drift component is demonstrated. The requantization errors will propagate throughout the frames and result in disturbing artefacts in the output video sequences. As opposed to temporal drift, which only results in visible drift after a number of frames, spatial drift will result in visible errors even within a single frame.

In order to reduce the effect of both drift components, this thesis presents architectures that highly improve rate-distortion (R-D) results of the adapted streams. In Chapter 2, architectures are proposed that add a spatial compensation loop to the transcoder, leading to strongly improved quality of the transcoded video streams. Also, attention is paid to *mixed* transrating architectures, that apply techniques based on individual picture and/or macroblock types. Macroblock types that are more susceptible to requantization errors can apply techniques with higher complexity (but with improved drift reduction), while other macroblocks can fall back to low-complexity techniques (e.g., open-loop transrating). In this manner, a trade-off is reached between quality and complexity of the overall transrating solution. The proposed solutions were partly developed within a bilateral project in cooperation with Cisco Systems. All techniques were implemented in a framework for transcoding which is fully compliant with the H.264/AVC standard and has been verified for a large variety of video streams.

Additionally, the architectures were extended in order to comply with H.264/AVC's *High profile*, which targets high-definition and high-resolution video coding. Similar problems can be identified for high-resolution video. The problem of intra-coded regions is further aggravated, given that the higher-resolution frames allow more propagation within individual images. *Selective* requantization can help to improve the quality of the transcoded frames by avoiding requantization for intra-coded macroblocks.

In order to further improve the rate-distortion performance, the possibility

was examined to refine the motion information in the transcoded H.264/AVC video streams, instead of only requantizing the residual coefficients. In this way, the motion information (motion vectors, partitioning information) is optimized for the characteristics of the outgoing bitstream. Since larger partition sizes will be preferred for lower bit rates and the benefits of smaller partition sizes can only be reaped for higher bit rates, tailoring this information to the targeted bit rate will further improve the quality of the output stream.

Chapter 3 deals with the conversion of non-scalable H.264/AVC-streams to SVC streams with multiple quality layers. By using intelligent transcoding techniques, a fast conversion of existing H.264/AVC files and archives to a scalable format becomes possible. In this thesis, it is shown that this conversion can be executed several times faster than by using a cascade of an H.264/AVC decoder and an SVC encoder. Different problems require closer attention during the conversion. In particular, intra-coded macroblocks give rise to issues. These macroblocks use inter-layer prediction mechanisms that are based on reconstructed pixel values from lower layers (*inter-layer intra prediction*). If the conversion is to be performed in the transform domain (without a complete decoding step), specific solutions need to be found for these macroblocks. In Chapter 3, different solutions for these problems are discussed and several intelligent architectures for H.264/AVC-to-SVC conversion are presented. Furthermore, the *bitstream rewriting* functionality in SVC is exploited during the conversion. Bitstream rewriting allows the conversion of SVC streams with multiple quality layers to an H.264/AVC stream with a single layer, i.e., the inverse operation, and was introduced for backward compatibility. This functionality can help the potential breakthrough of SVC by providing conversion nodes in networks, hereby avoiding that new decoders with SVC capability have to be provided to the end users. This thesis discusses how the rewriting functionality can be exploited to ease the conversion from H.264/AVC to SVC. Firstly, architectures based on this functionality retain their full quality after conversion. This means that after decoding, identical pictures are obtained when all layers are present in the SVC stream when compared to the original H.264/AVC single-layer stream. Secondly, more flexible architectures are obtained, and conditions are relaxed for intra-coded macroblocks.

Also, the refinement of motion information in the different layers of the created SVC stream is discussed. In previous solutions, the motion information is partitioned in the base layer of the output stream. For lower layers, however, tailored motion information can be beneficial for rate-distortion performance. Motion information is then refined in higher layers. Based on a multi-layer control mechanism, the desired layers are optimized. As a downside, the total

bit rate of the SVC stream will somewhat increase, due to the partitioning of the motion information in different layers.

In Chapter 4, the possibility of decoding streams at a reduced resolution is examined. This technique reduces the complexity of decoders, as well as the memory requirements for the reference pictures. In H.264/AVC, this is an important argument, given the use of multiple reference pictures. Low-resolution decoding can be achieved by using inverse transformations with shorter base functions or by using *frequency synthesis* techniques. Also, motion-compensated prediction is performed in the reduced-resolution domain.

Apart from low-resolution decoding, transcoding to a reduced-resolution stream is examined. Additional problems arise in this case, since the stream needs to be converted to a compliant output stream. Since not all block types can be mapped to reduced-size equivalents in H.264/AVC and the prediction signal needs to be updated, compensation techniques are required to avoid artefacts in the output video stream. Similar problems arise when multiple reference pictures are used. For these reasons, open-loop transcoding will fail for spatial resolution reduction transcoding. Intra-coded macroblocks further complicate the conversion, since a straightforward conversion to reduced-size equivalent blocks cannot be achieved.

From the results it is shown that spatial resolution reduction poses several challenges, and that it can quickly lead to error propagation in the output video stream. Different techniques have been studied that tackle error propagation, such as using 1/8-pixel interpolation during motion-compensated prediction. For spatial resolution reduction transcoding, an architecture is proposed which avoid artefacts by performing compensation in the reduced resolution. In this way, the shortcomings of open-loop architectures are avoided and transcoding leads to a compliant stream without visual artefacts.

Samenvatting

De beschikbaarheid en verspreiding van video kent de laatste jaren een steile groei. Waar nog geen vijftien jaar geleden de bandbreedte van netwerken ontoereikend was om streaming video (aan enige kwaliteit) mogelijk te maken, worden we tegenwoordig dagelijks geconfronteerd met videofragmenten en -films op het internet. De beschikbaarheid van videofragmenten heeft het uitzicht en de inhoud van websites danig veranderd. Steeds meer video wordt gecreëerd door gebruikers van websites zelf, wat heeft geleid tot het ontstaan van volledig nieuwe zakenmodellen en nieuwe manieren om content te delen. Ook de introductie van digitale video in de huiskamers heeft geleid tot een groter aanbod van content en toepassingen, alsook tot een betere kwaliteit van het afgeleverde televisiesignaal.

Deze vooruitgang werd in niet onbelangrijke mate mogelijk gemaakt door het efficiënt comprimeren van deze videodata. Het MPEG-2-videoformaat, dat begin jaren '90 werd gestandaardiseerd, en het MPEG-4 Visual formaat (en diens bekendere implementaties DivX en Xvid) hebben de verspreiding van video (zowel korte fragmenten als volledige films) in grote mate vooruit geholpen.

Recenter werd de H.264/AVC-standaard ontwikkeld door het Joint Video Team van ISO/IEC MPEG en ITU-T VCEG. Deze standaard levert een verdere reductie van de bitsnelheid bij eenzelfde kwaliteit, en is op dit moment het referentiepunt wat betreft videocompressie. Deze standaard werd dan ook gebruikt als uitgangspunt in dit doctoraal proefschrift.

Bij het gebruik en encoderen van video moet steeds meer rekening gehouden worden met de grote diversiteit aan toestellen die deze video kunnen ontvangen. PC's, laptops, GSM's en PDA's worden vaak aangewend om eenzelfde videofragment af te spelen. Deze toestellen beschikken echter over zeer uiteenlopende specificaties. Het versturen van een fragment in hoge definitie (bv. 720p of 1080p video) over een netwerk met lage bandbreedte naar een toestel met beperkte weergavemogelijkheden (zoals PDA of GSM) zal zowel het netwerk als het toestel overmatig belasten. In een dergelijk scenario is

een reductie van de bitsnelheid of spatiale resolutie van de videostroom een oplossing om de efficiëntie te verhogen.

Bovendien leiden de verscheidenheid aan standaarden en formaten gebruikt in bestaande productieomgevingen, distributienetwerken en omroepkanalen tot de noodzaak om efficiënte conversie tussen deze formaten mogelijk te maken. Afhankelijk van het beoogde doel zal dit al dan niet zonder verlies aan kwaliteit mogen gebeuren. Voor archiveringsdoeleinden bijvoorbeeld zal een kwaliteitsverlies niet getolereerd worden, terwijl bij distributie over netwerken met beperkte bandbreedte een klein (doch bij voorkeur niet merkbaar) verlies toegelaten kan worden.

De diversiteit aan standaarden en formaten enerzijds, en de uiteenlopende karakteristieken van toestellen anderzijds, verklaren de noodzaak aan technieken voor adaptatie van videostromen. Eén manier om snelle aanpassing van videostromen mogelijk te maken, is door te gebruikmaken van schaalbare videocodering. Schaalbare videocodering laat toe een gelaagde representatie van de videostroom te vormen tijdens het encoderen, waarbij de onderste laag (de 'basislaag') van de stroom telkens de laagste kwaliteit bevat. Bijkomende lagen (verfijningslagen) zullen een verbetering inhouden van de kwaliteit (kwaliteitsschaalbaarheid), spatiale resolutie (spatiale schaalbaarheid) of beeldsnelheid (temporele schaalbaarheid) van de video. Hoewel voorzieningen voor schaalbare codering reeds voorhanden waren bij MPEG-2 en MPEG-4 Visual, zijn deze in de praktijk zelden of nooit gebruikt. Recent werd als uitbreiding op H.264/AVC een nieuwe standaard voor schaalbare videocodering (SVC) gedefinieerd. SVC laat toe schaalbare stromen te creëren met minimaal verlies in *rate-distortion* (R-D) prestatie vergeleken met H.264/AVC. Ondanks deze schaalbare voorzieningen wordt de overgrote meerderheid van videostromen nog steeds geëncodeerd met één enkele laag. Het is nog de vraag of SVC een kentering kan teweegbrengen en een doorbraak kan betekenen voor schaalbare video.

Het gebrek aan schaalbare stromen brengt met zich mee dat alternatieve technieken ontwikkeld moeten worden om adaptatie van video te realiseren. In dit proefschrift wordt *transcodering* gebruikt als techniek om efficiënte adaptatie van H.264/AVC-videostromen mogelijk te maken. Transcodering behandelt de adaptatie van video en bereikt zijn efficiëntie door informatie uit de originele bitstroom te hergebruiken. In het bijzonder zal transcodering trachten de adaptatie sneller te bewerkstelligen dan de voor de hand liggende combinatie van decoder en encoder.

Een eerste belangrijk deel in dit proefschrift behandelt de reductie van de bitsnelheid van H.264/AVC-videostromen. De kwaliteit van videostromen wordt voornamelijk bepaald door de gebruikte quantisatieparameter (QP) tij-

dens het encoderen. Een hoge QP duidt op grove quantisatie van residuele data en leidt bijgevolg tot een lagere kwaliteit. Door de ruwere benadering van de getransformeerde coëfficiënten kunnen de resulterende waarden compacter worden voorgesteld, en zal de bitsnelheid van de stroom verkleind worden. Hierop kan ingespeeld worden tijdens transcodering: vertrekkende van de coëfficiënten in de oorspronkelijke bitstroom, kan de QP verhoogd worden, en zullen de coëfficiënten aangepast worden aan deze nieuwe QP. Deze techniek wordt *herquantisatie* genoemd. Bijgevolg zal de bitsnelheid van de uitgaande bitstroom gereduceerd worden, en bijvoorbeeld aangepast worden aan de bandbreedte van het netwerk. Verschillende complicaties treden echter op bij deze aanpassing. Standaarden voor videocodering halen hun hoge compressieratio door het uitbuiten van redundantie in de videosequenties. Zowel spatiale, temporele als statistische redundantie kunnen teruggevonden worden in videostromen. Om deze uit te buiten zullen technieken gebruikt worden als intrapredictie, bewegingsvoorspelling, transformatie met variabele blokgrootte en entropiecodering. Door deze technieken zal de voorspelling van beelden afhangen van eerder gecodeerde (en gedecodeerde) beelden of regio's in het huidige beeld. Door het aanpassen van de residuele data in het huidige beeld zullen bijgevolg ook afhankelijke beelden aangetast worden, resulterend in *temporele drift* in de getranscodeerde sequentie. In H.264/AVC zal door toedoen van intravoorspelling echter ook een *spatiale* driftcomponent voorkomen, wanneer intragecodeerde macroblokken worden getranscodeerd. In dit proefschrift wordt het belang van deze tweede driftcomponent aangetoond. Storende artefacten zullen het gevolg zijn van de foutdrift die zich propageert doorheen het beeld. Waar temporele drift slechts over verschillende beelden leidt tot zichtbare fouten, is dit niet langer het geval voor spatiale drift, en zal deze vorm van drift zelfs binnen één beeld leiden tot zichtbare fouten.

Om het effect van beide drifttermen te reduceren worden in dit werk verschillende architecturen geïntroduceerd die leiden tot verbeterde R-D-resultaten. Het toevoegen van een spatiale compensatielus in transcodeerarchitecturen wordt in dit werk voorgesteld, wat leidt tot een sterk verhoogde kwaliteit van de getranscodeerde videostromen. In het bijzonder wordt aandacht besteed aan zogenaamde *gemengde* architecturen, die technieken toepassen gebaseerd op individuele beeld- en/of bloktypes. Bloktypes die gevoeliger zijn voor herquantisatie kunnen technieken gebruiken met hogere complexiteit, terwijl andere blokken lage-complexiteitstechnieken toelaten. Op deze manier wordt tevens een compromis bereikt worden tussen kwaliteit en complexiteit van de transcodeeroplossing.

De voorgestelde technieken werden deels ontwikkeld binnen een bilateraal project in samenwerking met Cisco Systems (voorheen Scientific Atlanta).

Alle technieken werden geïmplementeerd binnen een eigen ontwikkeld raamwerk voor transcodering dat volledig voldoet aan de H.264/AVC-standaard, en werden uitvoerig getest op een grote verscheidenheid aan videostromen.

De architecturen werden voorts uitgebreid om ook te voldoen aan het *High profile* van H.264/AVC, dat zich richt op het coderen van hoge-definitie- en hoge-resolutievideo. Gelijkaardige problemen werden gevonden voor hoge-resolutievideo, waarbij het probleem van intragecodeerde regio's verder toeneemt, gezien de hogere resolutiebeelden meer propagatie toelaten binnen individuele beelden. De resultaten tonen aan dat *selectieve* herquantisatie hier verder de kwaliteit van de getranscodeerde beelden verbetert door herquantisatie te vermijden voor intragecodeerde blokken.

Om de R-D-prestaties verder te bevorderen, werd de mogelijkheid onderzocht om niet enkel de residuele data, maar ook de bewegingsinformatie van H.264/AVC-stromen te verfijnen tijdens transcodering. Op die manier wordt deze informatie (zoals bewegingsvectoren en partitioneringsinformatie) geoptimaliseerd voor de karakteristieken van de uitgaande bitstroom. Kleinere blokgroottes zullen immers meer voordeel bieden voor hogere bitsnelheden, terwijl bij lagere bitsnelheden grote blokpartitites zullen geprefereerd worden. Zoals aangetoond verbetert het aanpassen van deze informatie aan de uitgaande bitsnelheid verder de kwaliteit van de videostroom.

Een tweede deel in dit proefschrift behandelt de omzetting van niet-schaalbare H.264/AVC-stromen naar SVC-stromen met meerdere kwaliteitslagen. Transcodering maakt een snelle omzetting mogelijk van bestaande H.264/AVC-bestanden en -archieven naar een schaalbaar formaat. In hoofdstuk 3 wordt aangetoond dat deze omzetting vele malen sneller gebeurt d.m.v. transcodering dan gebruikmakende van een combinatie van H.264/AVC-decoder en SVC-encoder. Verschillende problemen duiken op bij de conversie van H.264/AVC naar SVC. In het bijzonder veroorzaken intragecodeerde macroblokken specifieke problemen. Deze macroblokken gebruiken predictiemechanismen die zich baseren op gereconstrueerde pixelwaarden uit onderliggende lagen. Indien de conversie gebeurent in het getransformeerde domein, zonder een volledige decodeerstap, moeten specifieke oplossingen gevonden worden voor deze macroblokken. In dit deel worden verschillende oplossingen voor deze problemen besproken en worden verschillende architecturen voor het transcoderen van H.264/AVC naar SVC voorgesteld. Voorts wordt beroep gedaan op de *bitstream rewriting* functionaliteit die voorzien is in SVC. Bitstream rewriting laat toe SVC-stromen met meerdere kwaliteitslagen om te zetten in een H.264/AVC-stroom met één enkele laag. Op die manier wordt achterwaartse compatibiliteit voorzien. Deze functionaliteit kan een belangrijke rol spelen in de doorbraak van SVC, gezien op deze manier geen migratie

naar SVC-decoders moet worden gemaakt. Dit proefschrift bespreekt hoe deze functionaliteit kan worden uitgebuit om de conversie naar SVC te vergemakkelijken. Enerzijds kunnen de architecturen deze functionaliteit gebruiken om de volledige kwaliteit na conversie te behouden. Dit betekent dat de resulterende SVC-stromen de originele kwaliteit hebben wanneer alle lagen aanwezig zijn bij decodering. Anderzijds leidt het tot flexibelere architecturen, in het bijzonder voor intragecodeerde macroblokken.

Verder wordt besproken hoe de bewegingsinformatie kan verfijnd worden in de verschillende lagen van de gecreëerde SVC-stroom. Bij voorgaande oplossingen werd de bewegingsinformatie telkens onderverdeeld in de basislaag van de uitgaande stroom. Voor onderliggende lagen, echter, kan het voordelig zijn andere bewegingsinformatie te gebruiken. Bovenliggende lagen verfijnen vervolgens de bewegingsinformatie. Aan de hand van een meerlaagsoptimalisatiemodel wordt de gewenste laag geoptimaliseerd. Dit gaat ten koste van de totale bitsnelheid van de SVC-stroom; deze zal licht stijgen als een gevolg van de spreiding van de bewegingsinformatie over de verschillende lagen.

In hoofdstuk 4 van dit proefschrift werd de mogelijkheid bestudeerd om H.264/AVC-videostromen te decoderen aan en te transcoderen naar een verlaagde resolutie. Het decoderen aan verlaagde resolutie verlaagt de complexiteit van decoders evenals de geheugenvereisten voor referentiebeelden. Het lage-resolutiedecoderen maakt enerzijds gebruik van een inverse transformatie met kortere basisfuncties of van *frequentiesynthese*, en anderzijds van bewegingscompensatie in de lagere resolutie.

Extra problemen duiken op wanneer de originele stroom dient geconverteerd te worden naar een geldige H.264/AVC-stroom. Gezien niet alle blokgroottes rechtstreeks kunnen geconverteerd worden naar gereduceerde blokgroottes (van de submacrobloktypes bestaat geen gereduceerd equivalent), zal compensatie nodig zijn om artefacten te vermijden in de uitgaande videostroom. Hetzelfde geldt voor het gebruik van meerdere referentiebeelden. Omwille van deze reden zal open-loop transcodering falen voor reductie van de spatiale resolutie. Voorts zullen intragecodeerde macroblokken problemen veroorzaken, gezien zij niet kunnen geconverteerd worden naar een equivalent intragecodeerd type in het gereduceerde domein.

Uit de resultaten blijkt dat de reductie van de spatiale resolutie geen evidente omzetting betekent en snel kan leiden tot foutdrift in de uitgaande videostroom. Verschillende technieken worden bestudeerd die deze foutpropagatie kunnen reduceren, zoals gebruikmaken van 1/8-pixel interpolatie om fijnere compensatie mogelijk te maken in het gereduceerde domein.

Voor het geval van spatiale-resolutietranscodering wordt een architectuur voorgesteld die artefacten vermijdt door compensatie uit te voeren in de gere-

duceerde resolutie. Op deze manier worden de tekortkomingen van open-loop architecturen vermeden en kan transcodering leiden tot compliant bitstromen zonder visuele artefacten.

List of abbreviations

AC	Alternating Current
AVC	Advanced Video Coding
BL	Base Layer
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CBR	Constant Bit Rate
CGS	Coarse-Grain quality Scalability
CIF	Common Intermediate Format
CPDT	Cascaded Pixel-Domain Transcoder
DC	Direct Current
DCT	Discrete Cosine Transform
DF	Deblocking Filter
DFD	Displaced Frame Difference
DM	Directly Mappable
DRS	Dynamic Rate Shaping
FCD	Final Committee Draft
FFT	Fast Fourier Transform
FGS	Fine-Grain quality Scalability
FMO	Flexible Macroblock Ordering
FPDT	Fast Pixel-Domain Transcoder
fps	frames per second
FR	Full Resolution
FRExt	Fidelity Range Extensions
GOP	Group Of Pictures
GSM	Global System for Mobile communications
IBBT	Interdisciplinary Institute for Broadband Technology
IDCT	Inverse Discrete Cosine Transform
IDR	Instantaneous Decoder Refresh
IEC	International Electrotechnical Commission
INTERMEDIA	Interactive Media with Personal Networked Devices
IP	Intra Prediction
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector

JM	Joint Model
JPEG	Joint Photographic Experts Group
JSVM	Joint Scalable Video Model
JTC	Joint Technical Committee
JVT	Joint Video Team
kbps	kilobits per second
MB	Macroblock
Mbps	Megabits per second
MCP	Motion-Compensated Prediction
MGS	Medium-Grain quality Scalability
MPEG	Moving Picture Experts Group
MRA	Mixed Requantization Architecture
MV	Motion Vector
NDM	Non Directly Mappable
NoE	Network of Excellence
NTSC	National Television System Committee
OL	Open Loop
PAL	Phase Alternating Line
PC	Personal Computer
PDA	Personal Digital Assistant
PR	Progressive Refinement
PSNR	peak signal-to-noise ratio
QCIF	Quarter CIF
QoE	Quality of Experience
QoS	Quality of Service
QP	Quantization Parameter
R-D	Rate-Distortion
RR	Reduced Resolution
SAD	Sum of Absolute Differences
SC	Spatial Compensation
SNR	Signal-to-Noise Ratio
SSD	Sum of Squared Differences
SVC	Scalable Video Coding
SVT	Sveriges Television
TC	Temporal Compensation
UHDV	Ultra High Definition Video
VBR	Variable Bit Rate
VCEG	Video Coding Experts Group

Contents

Chapter 1

Introduction

Different video coding standards have been introduced in the last two decades, with varying success in deployment. Among the more popular, we find H.261, MPEG-1 Video, H.262/MPEG-2 Video[1], H.263, and MPEG-4 Part 2. More recently, H.264/AVC [1] was standardized, and is gaining widespread acceptance as the state of the art video coding standard, with superior compression performance over its competitors. As was the case for MPEG-2 [2], part of the success of H.264/AVC can be attributed to the collaboration of VCEG and MPEG for its development. The collaboration of ITU-T SG16/Q.6 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG), also known as the Joint Video Team (JVT), finalized (the first version of) the specification in 2003.

In a number of previous video coding standards, provisions have been made to allow creating scalable bitstreams. When scalability is provided during encoding, adaptation of the created video streams can be easily achieved by removing fragments (dropping packets) from the bitstream. After decoding the resulting stream (which consists of a subset of the original packets), a video sequence is obtained with a lower quality, spatial resolution, or frame rate. Hence, scalability allows easy adaptation of bitstreams to match the properties of networks (e.g., bandwidth) and display devices (e.g., spatial resolution). A number of scalability techniques were included in MPEG-2 and in H.263 (Annex O). Very few applications, however, have ever used these standardized scalability techniques. More recently, the JVT has been working on the scalable extension of H.264/AVC (SVC), providing scalability at a minor compression penalty when compared to single-layer H.264/AVC video cod-

[1] Although the MPEG-1 and MPEG-2 standards comprise much more than only video specifications (for example, MPEG-1 Audio, MPEG-2 Audio, and MPEG-2 Systems), for notational simplicity we will refer to MPEG-1 Video and H.262/MPEG-2 Video as 'MPEG-1' and 'MPEG-2', respectively.

ing. The resulting standard was finished in 2007. As of now, it is uncertain if this effort to standardize scalable video streams will be successful in practical and professional applications. Scalable bitstreams have the benefit of allowing straightforward adaptation, but suffer a loss in rate-distortion performance when compared to single-layer coding.

It is in this spirit that transcoding for digital video[2] was introduced about two decades ago. Since the overwhelming majority of video sequences were encoded in a non-scalable format, algorithms were required that allowed adaptation of, as an example, MPEG-1 video streams [4]. In the meantime, little has changed, and most video applications still produce non-scalable video streams. Often multicast solutions are preferred over scalable bitstreams in practical applications, witness the choice between low-, medium-, and high-quality versions of the same content many streaming servers on the internet provide. Hence, the need for efficient adaptation solutions for (non-scalable) video streams remains as large as it was twenty years ago.

Generally, transcoding can be defined as *the conversion of one coded signal to another* [5]. Transcoding will be used to change the characteristics of the video signal to match, for example, the limitations of transmission networks or display devices. Typically, transcoding operations will result in a reduction of the bit rate[3].

For video adaptation, a straightforward solution can be to simply decode and (re)encode the video signal. Since full decoding, and in particular, (re)-encoding is a computationally very demanding operation, this cascade of decoding and encoding can be very time-consuming. To overcome this problem, different alternative transcoding architectures have been introduced that try to 'shortcut' the adaptation process of the decoder-encoder cascade. Computational efficiency has been the major driving force behind the development of these architectures.

Efficient adaptation of video bitstreams can be performed by reusing as much information as possible from the incoming bitstream, and by only changing the required data in the bitstream. This means for example that the motion vectors will be reused, while changes will be made to the residual data (transform coefficients) of the bitstream[4]. Another way of reducing complexity is by avoiding algorithmic steps in the compression chain. An example is to

[2] Transcoding for analog television has been around since the early sixties, concerned among others with the conversion between different television standards in Europe and the US, e.g., between PAL and NTSC. Some of the major challenges encountered were related to the conversions required for Eurovision broadcast, as described for example in [3], or for the Olympic Games in Rome and Tokyo.

[3] Exceptions exist, e.g., transcoding from H.264/AVC to MPEG-2.

[4] This is typically the case for bit rate reduction (transrating).

work in the frequency domain, hereby avoiding inverse and forward transform operations [6].

In order to obtain higher compression performance, standardization committees have pushed the limits of coding algorithms in order to identify and exploit spatial, temporal, and statistical redundancy in the video stream. As a result, the amount of dependencies in the bitstream severely increased. This means that by changing one *syntax element* of the bitstream, several other elements can be harmed. Due to the resulting mismatch between the transcoder and decoder, drift can arise in the bitstream, and video quality can degrade. Because of this reason, a significant effort related to the development of transcoding algorithms was dedicated to assuring visual quality of the transcoded bitstreams. Ideally, the transcoded bitstream should have the quality of a stream encoded directly with the required parameters. The problem of drift, and compensation methods to stop degradation have been extensively studied in literature, in particular for MPEG-2. Problems related to new coding algorithms in H.264/AVC have, for example, been discussed in [7].

Taking into account the aforementioned, a less ambiguous definition of transcoding can be formulated.

Transcoding can be regarded as a process for efficient adaptation of video content, in order to match the properties and constraints of transmission networks and terminal devices, by efficiently (re)using information from the incoming bitstream, while at the same time minimizing the quality loss due to the adaptation.

This definition and its different aspects (computational efficiency, quality loss minimization, and reuse of information) will serve as a leitmotif throughout this dissertation, and will be investigated for different transcoding scenarios in each chapter.

Several properties and constraints can be the subject or reason for the adaptation process, such as the available network bandwidth, delay, packet loss, bit rate variation[5], buffer constraints, display screen resolution, battery life, etc. To cope with the variety of configurations, conditions, and capabilities, a myriad of transcoding operations can be applied.

Transrating Bit rate reduction transcoding, also referred to as SNR or fidelity transcoding, or simply *transrating*, deals with reduction of video stream bit rate while (typically) leaving the other video stream characteristics (such as the temporal and spatial resolution) intact. This is accomplished by reducing

[5] constant bit rate (CBR) or variable bit rate (VBR).

the accuracy of the residual data in the bitstream. Commonly, two approaches are used: (i) by increasing the quantization step size of the residual transform coefficients or (ii) by omitting transform coefficients from the bitstream. In Chapter 2, we focus on transrating based on the first-mentioned. Since changes to the residual data in one frame or macroblock will impact the reconstruction of other frames or macroblocks (due to the interdependencies caused by motion-compensated prediction or intra prediction), precautions have to be taken to avoid the propagation of requantization error drift. Transrating architectures with varying computational complexity and error restriction capabilities are investigated in Chapter 2.

Heterogeneous transcoding Transcoding between different video coding standards is often referred to as *heterogeneous transcoding*. Given the abundance of MPEG-2 video content and the benefits of efficient migration to H.264/AVC, it is hardly a suprise that manifold publications [8–13] have been published on transcoding between MPEG-2 and H.264/AVC since the introduction of the latter specification. Heterogeneous transcoding can also be applied for backward compatibility, as studied, e.g., in [14]. H.263 to H.264/AVC transcoding [15–18] and its reverse operation [19] also received attention in the latest years, despite the lower popularity of the former codec.

In Chapter 3, we study the conversion of single-layer H.264/AVC to SVC streams consisting of multiple quality layers. Although this operation is strictly speaking not a conversion between different video coding standards (SVC was standardized as Annex G of the H.264/AVC recommendation-standard), we classify it among heterogeneous transcoding operations given the non-triviality of the extension of SVC compared to single-layer H.264/AVC (concerning coding tools and algorithms available).

Spatial resolution reduction transcoding Spatial resolution downscaling is undoubtedly one of the most relevant, yet most challenging transcoding operations. When broadcasting video to a range of possible devices with divergent screen resolutions, a number of approaches can be followed to display the video on these devices:

- The original video stream is transmitted to the devices, after which decoding at the full resolution is performed. This full-resolution sequence can subsequently be downsized according to the screen resolution or the user's wishes. Straightforward pixel-domain downscaling techniques can be applied in this case.

- Instead of decoding at full resolution, reduced-complexity techniques can be designed that decode directly at the reduced resolution. Both computational complexity and memory requirements are reduced using this technique (due to a reduction of calculations related to motion-compensated prediction and reduced memory accesses). Dedicated decoding algorithms that allow this functionality are required, however, at the receiver devices. This approach was, for example, studied in [20,21].

- Both of the above methods have the disadvantage that a full-resolution video stream is transmitted all the way over the network down to the receiver devices, hereby unnecessary putting a load on the network. As an extreme example, transmitting a 1080p HDTV stream to a mobile device would pointlessly overload the transmission network. A universal solution (requiring no prior assumptions of the receiver devices) is to transcode the bitstreams at the server side or at intermediary nodes in the distribution chain. Different transcoding approaches can be followed in this case. Pixel-domain transcoding techniques first decode the stream after which the stream is downscaled in the pixel-domain and subsequently reencoded at the reduced resolution. The biggest challenge for this approach is to limit the computational complexity of the reencoding operation. Even though reencoding is performed at the reduced resolution, an exhaustive search for the rate-distortion optimal coding modes and motion vectors requires a tremendous amount of time. Fast techniques that reuse information from the input stream are encouraged, e.g., by deriving motion vectors based on a combination of motion vectors found in the original stream.

We investigate both reduced-resolution *decoding* and *transcoding* (the second and third approach) in Chapter 4.

Temporal resolution reduction transcoding Temporal resolution reduction deals with the adaptation of the video frame rate of encoded bitstreams. Conversion of frame (or field) rate is required in different scenarios due to technical constraints, such as the difference in frame rate between film (24 fps) and television systems (PAL @ 25 fps or NTSC @ 30 fps). 2:2 or 2:3 pulldown (Telecine) techniques are commonly used in these scenarios. Most introduced transcoding architectures, however, focus on frame rate reduction with the goal of further lowering the bit rate of the output video stream, e.g., by a reduction of the frame rate by two, three, or four. Several solutions for frame skipping have been introduced, such as in [22–25]. A downside is that the performance of these transcoders will heavily rely on the used GOP structure. In most cases,

a transcoder with pixel-domain reconstruction will be the only reliable solution (instead of faster transform-domain solutions that avoid a full reconstruction). Reduction in computational complexity for this type of transcoder is obtained by minimizing the time needed for motion reestimation, e.g., in the case of frame skipping by retracing motion vectors back to a valid reference frame (referred to as *forward dominant vector selection*).

In H.264/AVC, random frame skipping is complicated further due to the larger degree of freedom available in motion vector or reference picture selection. Hence, frame rate reduction transcoding becomes more complicated for H.264/AVC bitstreams when the GOP structure was not tailored to frame rate reduction applications at encoding time, due to the complicated dependency relations that can be introduced in the bitstream. On the bright side, the additional flexibility of H.264/AVC can also be used to allow arbitrary GOP structures that allow temporal scalability. Thanks to the reference picture management control in H.264/AVC, no changes were required in the syntax to allow hierarchical coding and temporal scalability. As it turns out, hierarchically coded GOP structures achieve excellent rate-distortion performance [26]. By increasing the GOP length (and hence the corresponding coding delay), rate-distortion performance can further be improved. Frame rate reduction can easily be achieved when hierarchical coding patterns are used, hereby allowing dyadic scalability.

In this dissertation, we elaborate on the first three important classes of transcoding operations, i.e., transrating, heterogeneous transcoding, and spatial resolution reduction transcoding. In all cases, the focus is on the H.264/AVC standard and its scalable extension SVC. For H.264/AVC-to-SVC transcoding, we evaluate the performance on hierarchically coded sequences, hereby allowing combined quality and temporal scalability for the output sequences. Although transcoding has been an important research topic for the last two decades, it is shown that solutions that were developed in the past, e.g., for MPEG-1 or MPEG-2, cannot be reused as such for use in H.264/AVC. Too many fundamental changes were made in the core design of the codec to carelessly adopt and apply previously developed solutions to H.264/AVC. Some of the major changes that will affect the design of transcoding solutions are the intertwined transform and quantizer in H.264/AVC, the addition of intra prediction and the possibility to include intra-coded macroblocks in P and B pictures, the quarter-pixel interpolation process for motion-compensated prediction, the in-loop deblocking filter, the use of submacroblock partitions, multiple reference indices, and variable prediction direction (forward, backward, or bipredictive) within a single macroblock.

In all stages of the research, the concepts described in this dissertation were evaluated and implemented into a fully operational framework for transcoding. This framework was written in C++ and contains a full implementation of all coding tools in the Main and High profiles of H.264/AVC (including interlaced coding), and of the inter-layer mechanisms needed to support quality scalability in SVC. In all cases, output of the transcoder is fully compliant to the H.264/AVC and (final) SVC standard, and can be decoded as such by all available H.264/AVC (Main/High profile) or SVC decoders. Branches of this implemented framework have been contributed to a number of research projects, such as IBBT projects MCDP, PoKuMOn, QoE, and the European FP6 IST Network of Excellence INTERMEDIA. Part of the research on requantization transcoding in Chapter 2 has been performed in cooperation with Scientific Atlanta (currently subsidiary of Cisco Systems) in light of the development of real-time hardware systems for transrating of H.264/AVC video streams.

The author's research has led to four SCI-indexed journal papers, two of which as first author (in *IEEE Transactions on Multimedia* and *Lecture Notes in Computer Science*), and two as co-author (in *Multimedia Tools and Applications* and *Journal of Visual Communication and Image Representation*). Additionally, three journal papers as first author and two as second author have been submitted for publication. Furthermore, the presented work has led to 26 conference publications listed in the ISI Proceedings database (nine as first author).

Chapter 2

Requantization transcoding

2.1 Rationale and related work

Video adaptation is required in order to meet network and user constraints in the multimedia content delivery chain. In many scenarios, reduction of the bit rate of video bitstreams is necessary. Often, the rate is altered without implying changes to the other video stream characteristics (such as spatial or temporal resolution). This case can also be referred to as rate transcoding, rate shaping, or *transrating*[1].

The reduction of the bit rate is typically achieved by decreasing the amount of residual data in the bitstream. Two frequently used techniques are *requantization*, i.e., a reduction of residual data by increasing the coarseness of the quantizer, and *dynamic rate shaping* (also known as selective transmission or thresholding), by dropping coefficients from the bitstream. The focus of the remainder of this chapter will be on requantization transcoding.

For MPEG-1/2 bitstreams, different solutions for transcoding have been investigated. In MPEG-1/2, variable length entropy codes allowed the use of fast coefficient clipping techniques, such as constrained or unconstrained dynamic rate shaping (DRS) [27, 28]. Apart from DRS techniques, the major part of the investigated architectures focused on requantization transcoding. In [4], a single-loop transcoder system was derived starting from a cascaded decoder-encoder solution, by merging common operations in the decoder and encoder loop. By doing this, a significant reduction in complexity is achieved. In [29], a number of reduced-complexity architectures, including an open-loop system, were investigated. Open-loop requantization transcoding proved to be a viable solution for MPEG-1/2, with a performance very

[1] Transrating can, e.g., also be used for decoder buffer control at splice points (for example in Digital Program Insertion applications).

similar to a decoder-encoder cascade [30]. Open-loop requantization transcoding is also the technique which is used in various commercial MPEG-1/2 rate shaping systems. Drift from open-loop requantization, however, may result in degraded performance when using longer GOP structures, due to temporal proliferation of requantization errors. In order to overcome this issue, drift-free solutions have been developed for MPEG-2 [6]. Owing to transform-domain operations, the complexity of these solutions was kept significantly lower than cascaded decoder-encoder architectures. In particular, solutions were sought for simplifications of the motion-compensated prediction (MCP) process during transcoding, resulting in efficient techniques for transform-domain MCP (MC-DCT) [6, 31].

The H.264/AVC specification introduces advanced coding techniques [32] in order to increase the coding efficiency compared with previous video coding standards, e.g., MPEG-1/2, H.263, and MPEG-4 Visual. This also results in more dependencies in the coded bitstream, which have to be taken into account during transcoding. Because of this, previously existing transcoding techniques are rendered obsolete. As an example, open-loop requantization can be applied to H.264/AVC bitstreams, but the visual quality of the transcoded bitstreams will be substantially affected. In particular, drift will be severe, both spatially and temporally. Hence, updated, intelligent techniques are required in order to reduce the bit rate.

Most of the research in literature on transcoding for H.264/AVC has been dedicated to cascaded decoder-encoder solutions. Different fast motion estimation algorithms have been proposed. Mode refinement for a cascaded decoder-encoder in requantization transcoding was for example discussed in [33, 34]. Solutions for spatial resolution reduction [35–37] or heterogeneous transcoding from or to H.264/AVC [18, 19] also focused on pixel-domain cascaded decoder-encoder solutions. In particular, the problem of fast motion estimation and mode decision using motion information from the incoming bitstream was studied.

In [7], an assessment was made of the performance of various existing requantization techniques, including a cascaded pixel-domain transcoder (CPDT) and a single-loop transcoder with temporal compensation (fast pixel-domain transcoder, FPDT). Another mixed requantization architecture (MRA) was examined, which uses the CPDT architecture for intra-coded pictures, and FPDT for predictive (P) and bidirectionally predicted (B) pictures. Although performance was significantly increased, MRA led to unpredictable results and rapidly degrading quality in GOP structures. A gap of about 3 to 4 dB in rate-distortion performance was found when compared to the CPDT solution. In this context, the problem of requantization of intra-coded regions in P and B

pictures was remarked, but not resolved.

Other research has shown that the selection of the requantization step size does not have monotonic effects on the rate-distortion curves for transcoding [38, 39]. In [38], the author explains how careful selection of the requantizer can result in improved performance for transcoding of H.264/AVC bitstreams.

In this chapter, requantization transcoding for H.264/AVC is discussed. In the following sections, we show that the use of intra-coded macroblocks poses a new challenge in transrating. In particular, the problem of spatial drift of requantization errors causes new problems in H.264/AVC video coding, and renders previously existing solutions useless. Results will show that open-loop requantization transcoding introduces severe quality degradation and therefore can not be used as a bit rate reduction technique for intra-coded pictures. This is to a large extent caused by spatial drift due to intra prediction. This chapter starts with a discussion of H.264/AVC quantization, which forms the basis for the design of requantization architectures in the remainder of this chapter. An analysis is provided of the spatial and temporal drift induced by requantization, by comparing the imperfect open-loop transcoder and the drift-free cascaded pixel-domain architecture. Intra prediction will be shown to be the major source of drift in H.264/AVC transcoding, leading to disturbing artifacts in the transcoded video streams. Based on this evaluation, we introduce single-loop transcoding techniques, which form the basis for the hybrid spatio-temporal transcoding architecture.

Most algorithms in literature focus on transrating for low resolution video (SDTV or lower). In Section 2.4, the performance of the proposed transcoding techniques is also investigated for high-resolution streams. We extend the presented techniques for use with the advanced coding tools of H.264/AVC High profile, and investigate whether the results still hold for high-resolution video material.

The extent to which the bit rate is reduced during transcoding depends on the application and on the (possible time-varying) parameters of the system. For broadcast applications, a typical reduction of 10-15% is desired, whereas for transmission to mobile networks and devices a larger reduction is often required. When such large reductions of the bit rate are targeted, it becomes beneficial to also investigate a change in motion parameters. By reducing the motion partitioning granularity and merging partitions, the motion data rate can be reduced, and an improvement in rate-distortion performance can be obtained. The addition of motion refinement to transrating is discussed in Section 2.5. Adjustment of the motion parameters to better match the properties and bit rate of the outgoing bitstream increases computational complexity, but can also improve rate-distortion performance. Apart from motion refinement

for pixel-domain architectures, we also investigate whether motion refinement can successfully be applied in the transform domain.

2.2 Requantization transcoding for H.264/AVC

2.2.1 Transform and quantization in H.264/AVC

This brief overview of the H.264/AVC specification provides a background for the open-loop requantization problem elaborated on in the next sections. More details regarding the transform and quantization in H.264/AVC can be found in [40].

The quantization design in H.264/AVC is different when compared to quantization schemes of previous video coding specifications, due to the fact that floating point arithmetic is avoided (16-bit integer operations are sufficient for 8-bit pixel data), the quantization step size increases exponentially in order to provide much broader range of possible bit rates, and pre- and post-scaling (normalization) operations of the forward and inverse integer transforms are incorporated in the quantization multiplier coefficients. Because of this intertwinement, and to get a clear understanding of the H.264/AVC quantization, we start by discussing the integer transform design of H.264/AVC.

A. H.264/AVC 4×4 integer transform

A regular type-II 4×4 DCT transform can be defined as follows:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^T \tag{2.1}$$

where

$$\boldsymbol{A} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \tag{2.2}$$

and a, b, and c are defined as $1/2$, $\sqrt{1/2}\cdot cos(\pi/8)$, and $\sqrt{1/2}\cdot cos(3\pi/8)$, respectively. Matrix $\boldsymbol{A}$ can be factorized as follows:

$$\boldsymbol{A} = \boldsymbol{B}\,\boldsymbol{C} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & b \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \tag{2.3}$$

with $d = c/b$, leading to:

$$\boldsymbol{Y} = \boldsymbol{B}\,\boldsymbol{C}\,\boldsymbol{X}\,\boldsymbol{C}^T\,\boldsymbol{B}\,. \tag{2.4}$$

Since $\boldsymbol{B}$ is a diagonal matrix, this can be rewritten as:

$$\boldsymbol{Y} = (\boldsymbol{C}\ \boldsymbol{X}\ \boldsymbol{C}^T) \circ \boldsymbol{E} \tag{2.5}$$

where $\circ$ indicates Hadamard matrix multiplication[2] , and

$$\boldsymbol{E} = \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} . \tag{2.6}$$

The value of d represents a non-rational value, and cannot be implemented using binary arithmetic. This is avoided by using the approximation $d = 1/2$ in matrix $\boldsymbol{C}$ (cf. [41]). As a result,

$$\boldsymbol{C} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix} . \tag{2.7}$$

In the forward transform, rounding errors due to integer arithmetic divisions are avoided by scaling the even rows of $\boldsymbol{C}$. The final multiplier matrix $\boldsymbol{C}_F$ is obtained:

$$\boldsymbol{C}_F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} . \tag{2.8}$$

By appropriate scaling, $\boldsymbol{E}_F$ follows:

$$\boldsymbol{E}_F = \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix} . \tag{2.9}$$

To maintain orthogonality (after the approximation of d as $1/2$), the value of b is updated accordingly:

$$b = \sqrt{2/5}\ . \tag{2.10}$$

The inverse transform is constructed in a similar way, as

$$\boldsymbol{X} = \boldsymbol{C}^T(\boldsymbol{Y} \circ \boldsymbol{E})\,\boldsymbol{C}\ . \tag{2.11}$$

[2]The Hadamard product, also known as entrywise product or Schur product, for two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times n}$ is formed as $(A \circ B)_{ij} = A_{ij}\ B_{ij}$, and $(A \circ B) \in \mathbb{R}^{m\times n}$.

For the inverse transform, the original matrices $\boldsymbol{C}$ and $\boldsymbol{E}$ are kept, given the higher dynamic range of the input coefficients at the inverse transform:

$$\boldsymbol{C}_I = \boldsymbol{C} \ , \tag{2.12}$$

and

$$\boldsymbol{E}_I = \boldsymbol{E} \ . \tag{2.13}$$

In H.264/AVC, the normalization multiplications are postponed to the quantization. From here on, we denote the *core* forward and inverse transforms as T and T^{-1}, respectively, i.e., $T(\boldsymbol{X}) = \boldsymbol{W} = \boldsymbol{C}_F \boldsymbol{X} \boldsymbol{C}_F^T$ and $T^{-1}(\widehat{\boldsymbol{W}}) = \boldsymbol{C}_I^T \widehat{\boldsymbol{W}} \boldsymbol{C}_I$. By Q and Q^{-1}, we indicate the forward and inverse H.264/AVC quantization processes, including post- and pre-scaling normalization operations E_F and E_I, as dicussed in the following section. The forward and inverse transform and quantization processes are illustrated in Figure 2.1, where Q_{tr} denotes 'traditional' quantization, without scaling. Quantization is performed for every element W_{ij} of $\boldsymbol{W}$, where the subscript $_{ij}$ indicates the position in the 4×4 matrix[3].

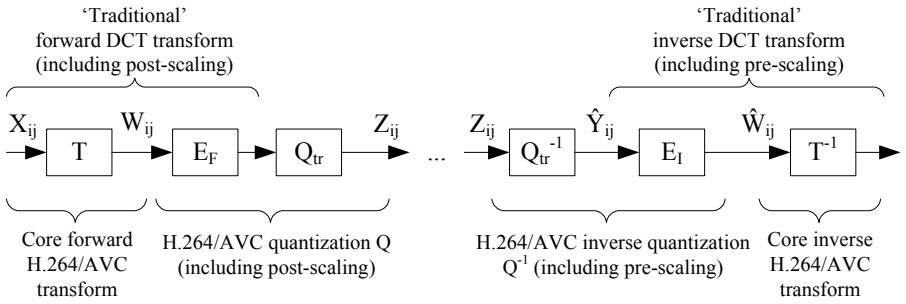


Figure 2.1: Transform, scaling, and quantization in H.264/AVC.

B. Quantization

The H.264/AVC specification defines a scalar quantizer with 52 predefined quantization step (Q_{step}) sizes, corresponding to a quantization parameter (QP) ranging from 0 to 51. The quantization step sizes are listed in Table 2.1. Note that the quantization step size is doubled for an increase of the quantization parameter by 6.

[3]The subscript $_{ij}$ is used throughout the text to indicate the position of elements in the corresponding matrices.

Table 2.1: H.264/AVC quantization step sizes.

	Q_{step}					
$QP\%6$	$\lfloor \frac{QP}{6} \rfloor = 0$	$\lfloor \frac{QP}{6} \rfloor = 1$	$\lfloor \frac{QP}{6} \rfloor = 2$	$\lfloor \frac{QP}{6} \rfloor = 3$	$\lfloor \frac{QP}{6} \rfloor = 4$	...
0	.625	1.25	2.5	5	10	...
1	.6875	1.375	2.75	5.5	11	...
2	.8125	1.625	3.25	6.5	13	...
3	.875	1.75	3.5	7	14	...
4	1	2	4	8	16	...
5	1.125	2.25	4.5	9	18	...

After the core forward transform is applied, the quantization can be performed, which incorporates the post-scaling operation with matrix $\boldsymbol{E}_F$ (as obtained in Equation 2.9). Note that the H.264/AVC standard does not specify the *forward* quantization process. Instead, the following formula can be regarded as a preferred way of performing quantization, given the reconstruction formulas in the standard:

$$\begin{cases} |Z_{ij}| = (|W_{ij}| \ M_{ij} + \epsilon \ 2^{qbits}) \gg qbits \\ \text{sign}(Z_{ij}) = \text{sign}(W_{ij}) \end{cases} \tag{2.14}$$

Here, the parameter ϵ controls the dead zone size of the quantizer characteristic, and $\gg$ denotes the right shift operation. The H.264/AVC Joint Model reference software [42] uses values of $\epsilon = 1/3$ for intra coding and $\epsilon = 1/6$ for inter coding. The parameters $qbits = 15 + \lfloor QP/6 \rfloor$ and M_{ij} both define the coarseness of the quantizer characteristic. The M_{ij} values implement the division by Q_{step} and the multiplication with post-scaling values $E_{F,ij}$. After upscaling by 2^{qbits}, the result is rounded to the nearest integer value:

$$M_{ij} = \text{round}\Big((2^{qbits} \ E_{F,ij}) / Q_{\text{step}} \Big) \ . \tag{2.15}$$

As mentioned, the post-scaling factors $E_{F,ij}$ represent the normalization values for the forward DCT transform. Since the odd and even base functions of the core forward transform have different norms, these values become matrix position-dependent. By including these values in the quantization, different quantizer multiplier coefficients are obtained depending on the position in the 4×4 block of transform coefficients. Hence, the forward multiplier factors M_{ij} depend not only on the quantization parameter (see Table 2.2), but they also become position dependent in the 4×4 block according to the following partitioning (corresponding to a combination of two even, two odd, or an even

and an odd base function in the transform, respectively):

$$r = \begin{cases} 0, & (i,j) \in \{(0,0),(0,2),(2,0),(2,2)\} \\ 1, & (i,j) \in \{(1,1),(1,3),(3,1),(3,3)\} \\ 2, & \text{otherwise} \end{cases} . \tag{2.16}$$

At decoder side, the inverse quantization reconstructs the transform coefficients $\widehat{W}_{ij}$ using the quantized values Z_{ij}:

$$\widehat{W}_{ij} = Z_{ij}\ V_{ij}\ 2^{\lfloor QP/6 \rfloor} \tag{2.17}$$

where the inverse multiplier coefficients V_{ij} are derived as follows:

$$V_{ij} = \text{round}\Big(\frac{64\ Q_{\text{step}}\ E_{I,ij}}{2^{\lfloor QP/6 \rfloor}}\Big). \tag{2.18}$$

As can be seen from the definition, the inverse quantization incorporates the pre-scaling operation with matrix $\boldsymbol{E}_I$ of the inverse transform $\boldsymbol{C}_I^T(\widehat{\boldsymbol{Y}} \circ \boldsymbol{E}_I)\boldsymbol{C}_I$. The scaling factors $E_{I,ij}$ are derived as normalization values for the inverse transform, hence these factors are matrix position-dependent. The resulting inverse multiplier coefficients V_{ij} can be found for the different matrix positions in Table 2.2.

Table 2.2: Original multiplication factors M_{ij} and V_{ij}.

	M_{ij}			V_{ij}		
QP % 6	$r=0$	$r=1$	$r=2$	$r=0$	$r=1$	$r=2$
0	13107	5243	8066	10	16	13
1	11916	4660	7490	11	18	14
2	10082	4194	4194	13	20	16
3	9362	3647	3647	14	23	18
4	8192	3355	3355	16	25	20
5	7282	2893	2893	18	29	23

2.2.2 Open-loop requantization for H.264/AVC

In this paragraph, we examine H.264/AVC requantization for the most straightforward requantization technique, i.e., open-loop requantization. In traditional open-loop systems, the transcoder is a simple concatenation of an inverse and forward quantization step. This system is shown in Figure 2.2. Other syntax elements (macroblock partitioning, mode decisions, motion vectors, etc.) are bypassed to the output bitstream.

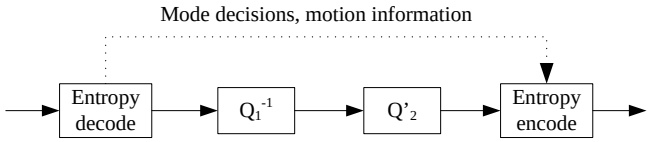


Figure 2.2: Open-loop transcoder.

For H.264/AVC requantization, a number of elements in the design further complicate this system. As noted in the previous section, the intertwined transform and quantization lead to the incorporation of normalization values in the quantization, as was illustrated in Figure 2.1.

Derivation of normalized multiplier coefficients for requantization

Because the normalization values $E_{F,ij}$ and $E_{I,ij}$ are already included in the encoder and decoder, respectively, care has to be taken not to repeat the scaling operation during transcoding. If not, the requantization process would produce values which are upscaled by a factor $64\ E_{F,ij}\ E_{I,ij}$, which equals 4 ($r = 0$), 3.2 ($r = 1$), or 2.56 ($r = 2$). Therefore, we propose the use of position-independent requantization multiplier coefficients M' and V', which are obtained by eliminating the factors $E_{F,ij}$ and $E_{I,ij}$ from the formulas above. Besides resulting in position-independent values, this has the advantage that rounding errors are avoided, since for $r = 1$ and $r = 2$ the values of V_{ij} are approximations of the actual values that would be obtained by Equation (2.18). The position-independent values M' and V' can be constructed as follows. As for the original multiplier coefficients, working with fractional values should be avoided. For V_{ij}, this is achieved by multipling the quantization step size by a factor of 16, resulting in integer multiplier coefficients (any lower power of 2 would still result in fractional values). This can also be seen from Table 2.1. Hence, the values for V' are calculated as follows:

$$V' = \frac{16\ Q_{\text{step}}}{2^{\lfloor QP/6 \rfloor}}\ . \tag{2.19}$$

In this way, $V' = V_{ij|r=0}$ (i.e., the value which is among others used for the DC coefficient), since $E_{I,ij|r=0} = 1/4$. As mentioned, unnecessary up- and downscaling operations are to be avoided, i.e., so that

$$M'\ V' \gg 15 = 1 \tag{2.20}$$

(compare to $M_{ij}\ V_{ij} \gg 15 = 64\ E_{F,ij}\ E_{I,ij}$). In order to achieve this, the forward multiplier coefficients are updated as follows, by incorporating the

factor of 16 used for construction of V':

$$M' = 2^{11+\lfloor QP/6 \rfloor} \frac{1}{Q_{\text{step}}} . \tag{2.21}$$

From the quantization step sizes in Table 2.1, we derived the updated position-independent quantizer coefficients as given in Table 2.3. By using these updated values, redundant scaling operations are eliminated in the requantization process, leading to more accurate requantized coefficients. As opposed to the formulas for M_{ij} (Equation 2.15) and V_{ij} (Equation 2.18), no rounding is needed to obtain the values for M' and V'.

In the remainder of this dissertation, the updated values M' and V' will be used for open-loop requantization. Requantization using these updated values will be denoted as Q'.

Table 2.3: Modified (position-independent) multiplier coefficients M' and V' for requantization.

$QP\%6$	M'	V'
0	3277	10
1	2979	11
2	2521	13
3	2341	14
4	2048	16
5	1821	18

2.2.3 Requantization error

A. Successive quantization error

When compared to *direct encoding* using QP_2, successive quantization may result in requantization errors since the second generation quantizer only has access to the first generation quantized transform coefficients $Z_{ij,1}$ instead of the original transform coefficients W_{ij} [43].

The direct encoding process applies coarse quantization to the original transform coefficients, resulting in the following formulas for forward and inverse quantization:

$$|Z_{ij}| = (|W_{ij}| \; M_{ij} + \epsilon \; 2^{qbits}) \gg qbits \tag{2.22}$$

$$qbits = 15 + \lfloor QP_2/6 \rfloor \tag{2.23}$$

$$\widehat{W}_{ij} = Z_{ij} \; V_{ij} \; 2^{\lfloor QP_2/6 \rfloor} \tag{2.24}$$

Successive quantization consists of a first generation and a second generation quantization. During the first generation quantization, the original transform coefficients W_{ij} are requantized resulting in the values $Z_{ij,1}$. After inverse quantization, the reconstructed coefficients $\widehat{W}_{ij,1}$ are obtained. The second generation quantization takes these values as input and results in the values $Z_{ij,2}$. The final reconstructed values are indicated as $\widehat{W}_{ij,2}$. Different values of the dead zone control parameter can be used in the first and second quantizer; this is reflected in the notation ϵ_1 and ϵ_2. The formulas for forward and inverse quantization are as follows:

$$\begin{aligned} |Z_{ij,1}| &= (|W_{ij}|\ M_{ij} + \epsilon_1\ 2^{qbits_1}) \gg qbits_1 \\ qbits_1 &= 15 + \lfloor QP_1/6 \rfloor \\ \widehat{W}_{ij,1} &= Z_{ij,1}\ V'_1\ 2^{\lfloor QP_1/6 \rfloor} \end{aligned} \tag{2.25}$$

and

$$\begin{aligned} |Z_{ij,2}| &= (\left|\widehat{W}_{ij,1}\right|\ M'_2 + \epsilon_2\ 2^{qbits_2}) \gg qbits_2 \\ qbits_2 &= 15 + \lfloor QP_2/6 \rfloor \\ \widehat{W}_{ij,2} &= Z_{ij,2}\ V_{ij,2}\ 2^{\lfloor QP_2/6 \rfloor}\ . \end{aligned} \tag{2.26}$$

Ideally, successive quantization should be equivalent to direct encoding: $\widehat{W}_{ij,2} = \widehat{W}_{ij}$. However, in many cases requantization errors are introduced since direct encoding is different from successive quantization.

This is illustrated in Figure 2.3. In this figure, the top half shows the quantizer bins for the first quantizer (with quantization step size $Q_{\text{step},1}$ and dead zone ϵ_1), while the bottom half shows the quantizer bins for the second quantizer (with quantization step size $Q_{\text{step},2}$ and dead zone ϵ_2). A quantized transform coefficient with value i, which was initially located in the quantizer bin with boundaries

$$[(i - \epsilon_1)\, Q_{\text{step},1}, (i + 1 - \epsilon_1)\, Q_{\text{step},1}[\tag{2.27}$$

will be reconstructed to value

$$r_{1,i} = i\, Q_{\text{step},1}\ . \tag{2.28}$$

Requantization maps this value to a larger bin ($QP_2 > QP_1$) with index j:

$$[(j - \epsilon_2)\, Q_{\text{step},2}, (j + 1 - \epsilon_2)\, Q_{\text{step},2}[\ , \tag{2.29}$$

and results in value

$$r_{2,j} = j\, Q_{\text{step},2} \tag{2.30}$$

after inverse quantization. In the case of open-loop transrating, $r_{1,i}$ is comprised in the latter interval and $j \leq i$.

In this figure, three regions are indicated which result in different quantized coefficients when successive quantization is applied, when compared to direct quantization using the second quantizer.

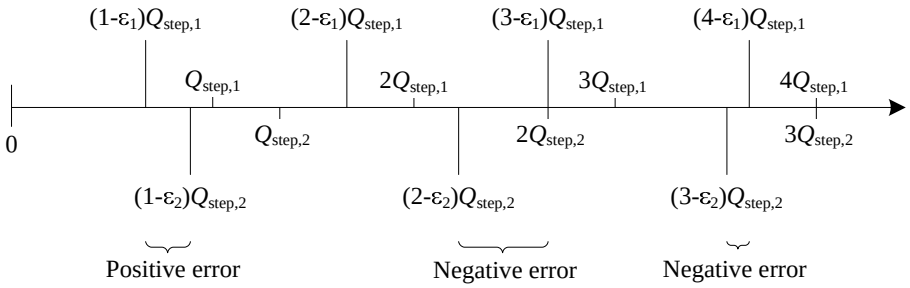


Figure 2.3: Illustration of positive and negative errors due to successive requantization.

Depending on whether a positive or negative error occurs, the requantized coefficient can suffer a deviation of ± 1, leading to a difference of $\pm V_{ij,2}\, 2^{\lfloor QP_2/6 \rfloor}$ after dequantization. The result of the inverse transform, $(\boldsymbol{C}_I^T \widehat{\boldsymbol{W}}_2 \boldsymbol{C}_I)$, is likely to differ from that obtained through direct quantization, $(\boldsymbol{C}_I^T \widehat{\boldsymbol{W}} \boldsymbol{C}_I)$, resulting in reconstruction errors which can propagate spatially and/or temporally.

In [44], Shen made an analytical study of the requantization error based on the analysis of quantizer bins. This study determines upper and lower bounds on the requantization error when the dead zone size is equivalent with the quantization step size ($\epsilon = 1/2$). However, other dead zone sizes are more appropriate for image and video compression schemes [45].

In [38], a requantization theorem for uniform scalar quantizers was derived. It showed that perfect requantization can be achieved if the following requirements are fulfilled, corresponding to the properties of an embedded quantizer:

- The boundary of quantizer Q_2 should be aligned with a boundary of a quantizer bin of Q_1.
- The quantizer step size of Q_2 should be an integer multiple of that of Q_1.

From these requirements, a number of dead zone combinations (ϵ_1, ϵ_2) were derived that allow for perfect requantization. The number of combinations

allowed for perfect requantization, however, severely reduces the flexibility of a transcoding system. For example, for a (typical) dead zone combination $(\epsilon_1, \epsilon_2) = (1/3, 1/3)$, the ratio $\frac{Q_{\text{step},2}}{Q_{\text{step},1}}$ has to equal $3n + 1$, $n \in \mathbb{Z}^+$, i.e., the smallest requantization Q_{step} ratio equals 4 (corresponding to a $\Delta QP = QP_2 - QP_1 = 12$), while finer perfect requantization is not possible.

The practical applicability of an open-loop *perfect* requantization system for H.264/AVC was mentioned in [38]. The problem of drift due to the energy loss and the absence of error compensation was mentioned, but its impact was not studied. In the remainder of this chapter, we will show that open-loop requantization leads to significant drift; this also applies to the case of 'perfect' requantization.

B. Reconstruction error

The error introduced above results from successive quantization, with the effect of possible positive or negative errors when compared to direct encoding. In certain cases, perfect requantization can be achieved, i.e., the output coefficient after transcoding will be identical to the one obtained by direct encoding. Nonetheless, the loss of information caused by coarser quantization with QP_2 will accumulate and propagate due to dependency coding.

Requantization leads to a loss in accuracy, which results in different reconstruction values $r_{1,i}$ and $r_{2,j}$ (as introduced in Equation 2.28 and Equation 2.30). The difference

$$|r_{1,i} - r_{2,j}| = \left| i\, Q_{\text{step},1} - j\, Q_{\text{step},2} \right| \tag{2.31}$$

leads to different reconstructed pixels before and after transcoding, and will cause drift at the receiving decoder.

In MPEG-2, error propagation was caused by motion-compensated prediction. In H.264/AVC, however, the source of drift will no longer be restricted to MCP alone. In fact, both spatial and temporal drift will be found because of intra prediction and MCP, respectively.

2.2.4 Requantization error drift in H.264/AVC

Even in the case of perfect requantization, open-loop requantization will suffer greatly due to propagation of the loss term in Equation (2.31). The individual errors will accumulate and spread due to MCP, or in the case of H.264/AVC, also due to intra prediction. In this section, the drift terms are investigated that constitute the difference between open-loop and drift-free transcoding. Both MCP and intra prediction play an important role in the development of drift in H.264/AVC.

An efficient version of intra prediction was adopted by the H.264/AVC specification. The efficiency is obtained by exploiting the spatial redundancy between the pixels in the current (4×4, 8×8, or 16×16) block and the surrounding (reconstructed) pixels of the neighboring (macro)blocks. H.264/AVC intra prediction results in improved compression efficiency, but also introduces new dependencies in the video bitstream. When requantization is applied to intra-coded macroblocks in H.264/AVC bitstreams (which can be present in all picture types), spatial drift propagation can be noticed. In order to avoid spatial drift, algorithms for requantization transcoding need to be (re)assessed.

Different requantization transcoding solutions have been proposed [5, 46], in both pixel and compressed domain. The most straightforward solution for transcoding is the decoder-encoder cascade. In this case, the incoming bitstream is fully decoded, and reencoded using the given parameters (such as a predefined bit rate). Due to its computational complexity, however, this solution is in many cases not feasible. Different reduced-complexity alternatives are available, two of which are the cascaded-pixel domain transcoder (CPDT) and the open-loop transcoder (OL).

In the remainder of this section, we study the drift as found in H.264/AVC transcoding. When compared to MPEG-2, the introduction of intra prediction and the intertwined transform and quantization lead to a different analysis. In the following discussion, the superscript 1 indicates a decoder-side signal and the superscript 2 denotes an encoder-side signal. Lowercase variables represent pixel-domain signals, while uppercase variables denote the equivalent signal in the transform domain. All notations are presented in Table 2.4 ($k \in \{1, 2\}$).

Table 2.4: List of notations.

Symbol	Description
C_n^k	quantized and transformed signal at decoder/encoder side
E_n^k	transformed error signal at decoder/encoder side
e_n^k	error signal at decoder/encoder side
x_n^k	reconstructed signal at decoder/encoder side
y_n^k	reference signal at decoder/encoder side
$T(\cdot)$	integer transform
$Q_i(\cdot)$	quantization (quantization parameter QP_i)
$I_m(\cdot)$	intra prediction (prediction mode m)
$M_{\boldsymbol{v}}(\cdot)$	motion compensation (motion vector $\boldsymbol{v}$)

A. Cascaded pixel-domain transcoding

The CPDT architecture can be considered as the reference model for requantization transcoding. This closed-loop architecture is by definition drift-free, but requires more processing power compared to compressed-domain solutions, due to the double prediction loop. Instead of performing motion estimation for the output video bitstream, the mode decisions and motion information of the input video bitstream are reused. In this manner, the computational complexity is significantly reduced. Note that if desired, the MVs and mode information can be refined to better reflect the characteristics of the output video signal, at the cost of increased computational complexity, as will be discussed further on in this chapter. The CPDT architecture is depicted in Figure 2.4. In the figure, a distinction is made between the buffer for the current picture, which is used to store reconstructed values for intra prediction, and the reference picture buffers (lists L0 and L1), which are used for MCP. In H.264/AVC, only pictures from reference picture list L0 are used for P-type macroblocks, while for B-type macroblocks, a choice can be made between reference pictures from list L0, L1, or a (weighted) combination of pictures from lists L0 and L1. After reconstruction of the current picture, the deblocking filter (DF) is applied, after which the picture can be stored for future reference.

For an individual block, the decoded pixels at decoder side and the corresponding prediction error at encoder side are given by the following expressions in case of intra prediction and MCP, respectively:

$$x_n^1 = \begin{cases} I_m(y_n^1) + e_n^1 \\ M_{\boldsymbol{v}}(y_{\text{Lx}}^1) + e_n^1 \end{cases} \tag{2.32}$$

and

$$e_n^2 = \begin{cases} x_n^2 - I_m(y_n^2) \\ x_n^2 - M_{\boldsymbol{v}}(y_{\text{Lx}}^2) \end{cases} , \tag{2.33}$$

where $I_m(\cdot)$ denotes the intra prediction operator with prediction mode m and $M_{\boldsymbol{v}}(\cdot)$ represents the motion compensation operator with motion vector $\boldsymbol{v} = (v_x, v_y)$. For simplicity, we omit the second motion vector in case of bidirectionally predicted blocks in B pictures. Since $x_n^1 = x_n^2$, Equation (2.32) can be substituted in Equation (2.33):

$$e_n^2 = \begin{cases} I_m(y_n^1) + e_n^1 - I_m(y_n^2) \\ M_{\boldsymbol{v}}(y_{\text{Lx}}^1) + e_n^1 - M_{\boldsymbol{v}}(y_{\text{Lx}}^2) \end{cases} . \tag{2.34}$$

The output prediction error equals the addition of the input prediction error and the difference between the prediction signal in decoder and encoder loop. This holds for both intra prediction and MCP.

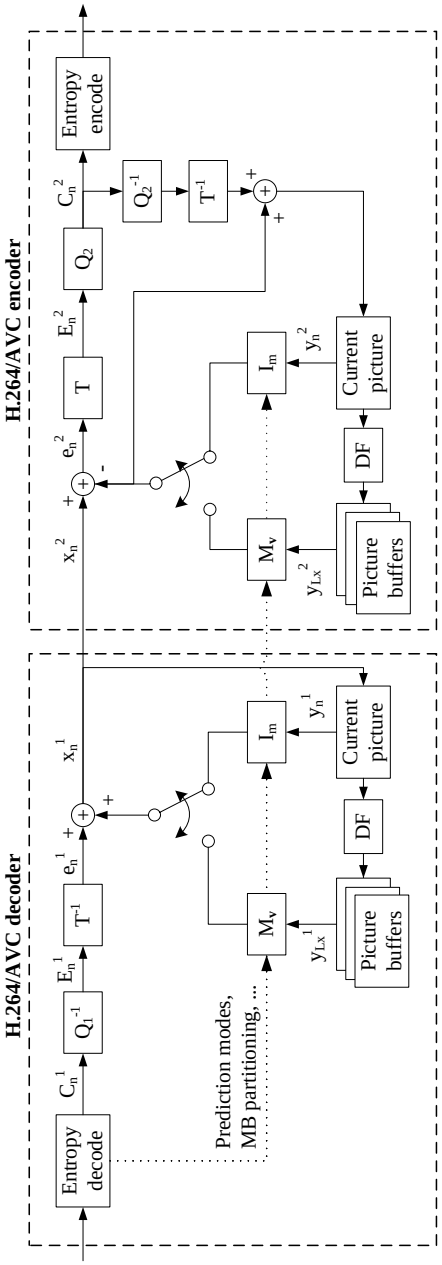


Figure 2.4: Cascaded pixel-domain transcoder architecture (CPDT).

The input and output prediction errors of frame n are related to input and output quantized coefficients indicated by C_n^1 and C_n^2 respectively:

$$e_n^1 = T^{-1}(Q_1^{-1}(C_n^1)) \tag{2.35}$$

$$C_n^2 = Q_2(T(e_n^2)) \ . \tag{2.36}$$

In this way, the output coefficients C_n^2 before entropy coding can be written as:

$$C_n^2 = \begin{cases} Q_2[T(I_m(y_n^1) + T^{-1}(Q_1^{-1}(C_n^1)) - I_m(y_n^2))] \\ Q_2[T(M_v(y_{\text{Lx}}^1) + T^{-1}(Q_1^{-1}(C_n^1)) - M_{\boldsymbol{v}}(y_{\text{Lx}}^2))] \end{cases} . \tag{2.37}$$

Due to linearity of the core forward integer transform:

$$C_n^2 = \begin{cases} Q_2[T(T^{-1}(Q_1^{-1}(C_n^1))) + T(I_m(y_n^1) - I_m(y_n^2))] \\ Q_2[T(T^{-1}(Q_1^{-1}(C_n^1))) + T(M_{\boldsymbol{v}}(y_{\text{Lx}}^1) - M_{\boldsymbol{v}}(y_{\text{Lx}}^2))] \end{cases} . \tag{2.38}$$

In case all coding parameters, i.e., mode decisions and motion information, remain the same for the output video bitstream, the above expressions can be simplified by merging the input and output coding loops. This, however, assumes linearity of the intra prediction and MCP processes, which is not the case in general (due to non-linear operations, as will be discussed further on):

$$C_n^2 \approx \begin{cases} Q_2[T(T^{-1}(Q_1^{-1}(C_n^1))) + T(I_m(y_n^1 - y_n^2))] \\ Q_2[T(T^{-1}(Q_1^{-1}(C_n^1))) + T(M_{\boldsymbol{v}}(y_{\text{Lx}}^1 - y_{\text{Lx}}^2))] \end{cases} . \tag{2.39}$$

B. Open-loop transcoding

The OL transcoder is the simplest solution for requantization, but it is also characterized by severe drift propagation. The residual information is dequantized with QP_1 and requantized with QP_2 (typically, $QP_1 < QP_2$). The OL transcoder is shown in Figure 2.5. Note that the modified multiplication coefficients M' are reflected in the notation Q_2' for the requantization step (as derived in Section 2.2.2).

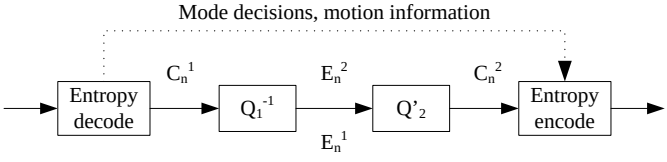


Figure 2.5: Open-loop transcoder.

The inverse quantization can be described as

$$E_n^1 = Q_1^{-1}(C_n^1) \ , \tag{2.40}$$

while the requantization can be formulated as

$$C_n^2 = Q_2'(E_n^2) \ . \tag{2.41}$$

Since $E_n^1 = E_n^2$ for the OL transcoder, Equation (2.40) and Equation (2.41) can be combined:

$$C_n^2 = Q_2'(Q_1^{-1}(C_n^1)) \ . \tag{2.42}$$

Apart from losses in the inverse transform, the following expression applies, where the adapted multiplication factors for the requantization process are used for requantization Q_2':

$$Q_2(T(T^{-1}(Q_1^{-1}(\cdot)))) \approx Q_2'(Q_1^{-1}(\cdot))) \ . \tag{2.43}$$

In this way, the above expression (Equation (2.42)) can be rewritten as follows, by reintroducing the inverse and forward transform processes in order to obtain the relation between the input and output coefficients in the OL transcoding solution:

$$C_n^2 \approx Q_2\big[T(T^{-1}(Q_1^{-1}(C_n^1)))\big] \ . \tag{2.44}$$

C. Drift components

By comparing Equation (2.39) to Equation (2.44), two drift components can be identified. These two terms constitute the difference in E_n^2 between the CPDT and OL solutions. In the case of intra prediction, a spatial drift term D_s is obtained (before quantization Q_2):

$$D_s = T(I_m(y_n^1 - y_n^2)) \ . \tag{2.45}$$

For motion-compensated prediction, a temporal drift term D_t is identified:

$$D_t = T(M_{\boldsymbol{v}}(y_{\text{Lx}}^1 - y_{\text{Lx}}^2)) \ . \tag{2.46}$$

The two components reinforce each other, since intra-predicted blocks can be used as reference for MCP, and vice versa.

Temporal drift component The temporal drift component results from requantization errors that propagate in the motion compensation loop. This type of drift also had to be dealt with in transcoded MPEG-2 streams. In H.264/AVC, multiple (long-term) reference picture MCP [47] is applied, which allows errors to propagate beyond the boundaries of the current GOP. In this way, the temporal drift accumulates and propagates from (reference) frame to frame until an Instantaneous Decoding Refresh (IDR) picture is processed, which clears the reference picture buffer.

Spatial drift component Since spatial dependency coding is not present in MPEG-2, this type of drift did not occur in MPEG-2 transcoding. Spatial drift accumulates and propagates from block to block according to the intra prediction modes. Since intra-predicted macroblocks can occur in I, P, and B pictures, this type of drift has a significant impact in all picture types. In particular, in high-motion regions in P and B pictures, intra macroblocks are often inserted as the rate-distortion-optimal choice by the encoder. Also, when the distance between reference pictures increases, intra prediction often becomes the optimal choice during encoding [48]. This is for example the case when using hierarchical GOP structures in the lowest temporal layers (as for example in SVC [49]).

The importance of the spatial drift term can be seen in Figure 2.6(a), where the PSNR values of the Stefan sequence (CIF resolution) are shown frame per frame after transcoding. A long IBBP GOP structure was used (298 frames, only the first frame is intra-coded) to also visualize the temporal drift effect. Drift is avoided in the first (intra-coded) picture by using the double-loop CPDT architecture for this frame.

The top curve illustrates the case where intra macroblocks in P and B pictures are not requantized, while the P and B macroblocks are requantized open-loop. An increase of the quantization parameter by 4 is used (from QP 22 to QP 26). Due to the high motion content of the Stefan sequence, temporal drift can be noticed. By additionally requantizing the intra-coded macroblocks for the same sequence, the bottom curve is obtained (in this case, all macroblocks in P and B pictures in the stream are open-loop transcoded). For this curve, the impact of spatial drift becomes visible. The variations in the PSNR values are related to the amount of intra-coded macroblocks as shown in Figure 2.6(b). This graph shows the amount of MBs that were coded as intra macroblock in the input video stream (out of a total of 396 macroblocks per frame). It can be seen that when the amount of intra MBs increases in a frame, the quality for the lower curve will decrease significantly when compared to the top curve. Because of spatial drift, quality can decrease within a single frame, while for

temporal drift, quality slowly deteriorates over a number of frames. This can be seen intuitively, given the high number of dependencies in intra-coded macroblocks [40].

In contrast, for the top curve, more texture information is preserved by leaving the intra-coded macroblocks unchanged, leading to higher PSNR values when large intra-coded regions are present in the frames (for these MBs the initial quality is kept, with $QP_2 = QP_1 = 22$). This effect becomes visible in the second half of the sequence.

In the proposed architectures in the following sections, temporal and spatial compensation loops are introduced to tackle both drift terms. Requantization with spatial compensation will also be applied to intra-coded macroblocks to obtain a uniform quality in the stream (hereby avoiding the peaks as shown in the second half of Figure 2.6(a).

Since spatial drift propagation results in annoying block artifacts in the decoded frames, precautions should be taken in order to avoid this kind of drift. Typical artifacts are visualized in Figure 2.7. Figure 2.7(a) shows a frame from the *Stefan* sequence, transcoded using the CPDT architecture. Figure 2.7(b) shows the same frame, transcoded open-loop. Drift is clearly visible, and propagates throughout the intra-coded frame.

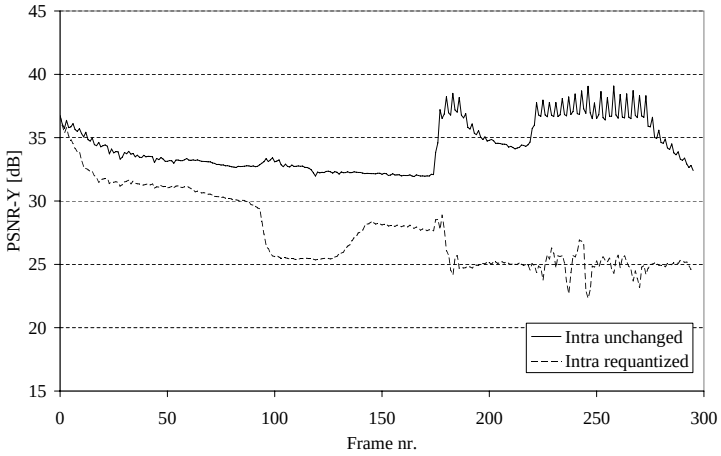
Spatial drift is not limited to intra-coded pictures, however, since intra-coded macroblocks can also be inserted in P and B pictures. This is visualized in Figure 2.7(c) and Figure 2.7(d). In the latter, requantization errors propagate in intra-coded regions, as can be seen for example around Stefan Edberg's head, legs, and tennis racket. When these macroblocks are used as reference for MCP, these drift errors propagate to depending frames. This results in severe drift errors, with worsening artifacts towards the end of GOPs.

D. Non-linear operations

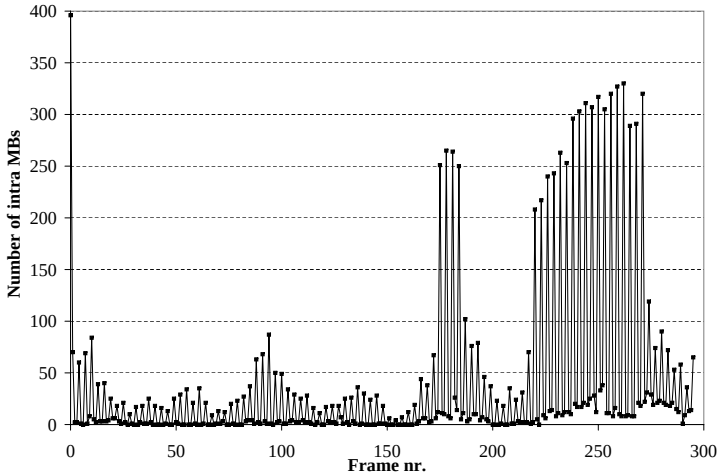
A number of non-linear operations prevent the formulas derived above from being exact, resulting in the approximation of the drift components in Equation (2.45) and Equation (2.46).

- **Intra prediction** The H.264/AVC specification provides nine intra 4×4 and four intra 16×16 prediction modes for the luma component, and four modes for the chroma components. Besides the horizontal and vertical prediction modes, a DC prediction mode and diagonal prediction modes are provided[4]. The latter modes make use of divisions in order to calculate their prediction values. These integer divisions (more precisely, the

[4] An overview of the intra prediction modes is given in Appendix A.



(a) Sequence without and with spatial drift (Stefan, CIF resolution).



(b) Number of intra MBs per frame (Stefan, CIF resolution).

Figure 2.6: Comparison of the quality of frames in the Stefan sequence without and with requantization of intra-coded macroblocks ($QP_1 = 22$, $QP_2 = 26$).

rounding operation after division) are non-linear operations, which may result in arithmetic rounding errors.

- **Inverse transform** The inverse integer transform in the H.264/AVC specification operates on values which are multiplied by a factor of 64 in order to prevent precision losses during the transform (see Equation (2.18)). After the inverse transform, the values have to be down-

(a) Driftless transcoded intra-coded picture (*Stefan*, frame 15, transcoded using CPDT, $QP_1 = 22$, $QP_2 = 28$)

(b) Spatial drift in intra-coded picture (*Stefan*, frame 15, transcoded using OL, $QP_1 = 22$, $QP_2 = 28$)

(c) Driftless transcoded P picture (detail of *Stefan*, frame 183, transcoded using CPDT, $QP_1 = 22$, $QP_2 = 28$)

(d) Spatial drift in intra MBs in P picture (detail of *Stefan*, frame 183, transcoded using OL, $QP_1 = 22$, $QP_2 = 28$)

Figure 2.7: Spatial drift propagation in intra-coded pictures (2.7(a) vs. 2.7(b)) and intra-coded MBs in P pictures (2.7(c) vs. 2.7(d)).

scaled again in order to correspond to the magnitude of the original values. This downscaling process results in rounding errors as a consequence of the division (shift operation).

- **Sub-pixel interpolation** When motion vectors in H.264/AVC refer to sub-pixel displacement positions, the sub-pixel position values are obtained by interpolation. For half-pixel displacements, a 6-tap filter is applied with filter coefficients $(1, -5, 20, 20, -5, 1)/32$. For quarter-pixel accuracy displacements, the half-pixel values are additionally interpolated using a bilinear filter. When applying the interpolation formulas to the requantization difference values in the reference frames, rounding errors may arise caused by integer arithmetic.

- **Bit depth clipping operations** A cascaded decoder-encoder solution will completely decode the video sequence, resulting in pixel values in the range of 0 to 255 (for a bit depth of 8 bits). In certain cases, these boundaries may be crossed during compensation (intra prediction or motion compensation), and clipping is necessary to the boundaries of the range. Since reconstructed values are not available in the open-loop transcoder, clipping cannot be performed, and may result in drift.

- **Deblocking filter** In the CPDT architecture, H.264/AVC in-loop deblocking filtering [50] is applied, resulting in altered values in the reconstructed pictures. This operation is not performed in the open-loop transcoder. In order to overcome blocking artifacts in the transcoded sequence, techniques for deblocking in the transform domain have been developed, as in [51–54]. These techniques, however, are only applicable in the absence of dependency coding (such as for JPEG images or MPEG-1/2 intra-coded pictures). For strongly dependent coding, such as for H.264/AVC, a full reconstruction of the pictures is required for deblocking.

It is clear that these individual effects are not easily quantifiable and depend highly on the transcoded sequence. The global effect of these non-linear operations, and hence of the approximation of the drift terms, will become clear in the results section.

2.3 Single-loop architectures with spatial and/or temporal compensation

In this section, we investigate algorithms that reduce the temporal and spatial drift components, while maintaining low complexity of the transcoder. Otherwise stated, architectures are examined that improve rate-distortion performance of the open-loop transcoder, at complexity lower than the CPDT transcoder. Architectures that refine mode information and motion vectors, hence having higher complexity than the CPDT transcoder, will be discussed in Section 2.5.

2.3.1 Basic single-loop requantization transcoder with temporal compensation

One approach for simplification of the CPDT transcoder is to identify and combine the decoder and encoder loop and to merge common modules based on the

assumed linearity of the transform and motion compensation [55]. In this way, a simplified 'single-loop' architecture is obtained. The traditional single-loop transcoder architecture is shown in Figure 2.8.

When compared to the open-loop transcoder, a compensation loop is added where the difference between the original coefficients and the requantized coefficients is used for compensation, corresponding to the temporal drift term D_t in Equation (2.46). The compensation prevents that errors, which result from the requantization process, propagate to depending blocks. This architecture was used for MPEG-1/2, for temporal compensation of requantization errors. The depicted architecture performs motion compensation in the pixel domain, hence the need for an inverse transform. A transform-domain approach was also used for MPEG-2, with a DCT-domain motion compensation module. This was discussed for example in [6].

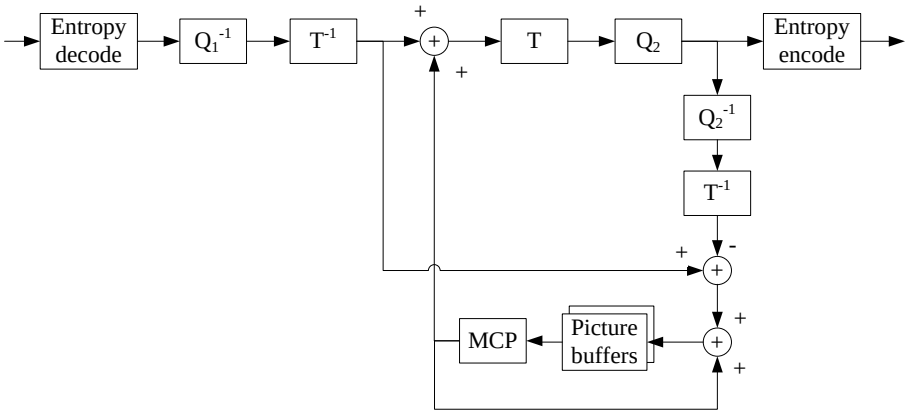


Figure 2.8: General architecture of the 'traditional' single-loop (pixel-domain) requantization transcoder with temporal compensation.

2.3.2 Single-loop architecture with spatial compensation

In literature, fast architectures for H.264/AVC transcoding are mostly based on the CPDT architecture, while single-loop architectures have been mentioned as not to be useful in practical applications [7,33,34]. These results were based on the observation of uncontrollable drift in intra-coded macroblocks and did not take into account the spatial drift term D_s in Equation (2.45). Here we present a single-loop architecture which does take into account spatial drift propagation. In this manner, single-loop architectures become practically viable, leading to a new class of transcoders available for H.264/AVC transcoding.

As demonstrated in the previous sections, compensation of spatially propagating errors is indispensable. This leads us to introduce an H.264/AVC requantization transcoder with spatial compensation, as shown in Figure 2.9.

In this architecture, a spatial compensation loop is used, by merging the decoder and encoder loop (similar to the single-loop architecture with temporal compensation). After merging, operations are performed on the accumulated difference signal[5] δ_n, instead of on the decoded signal y_n^k for the CPDT architecture:

$$\delta_n = e_n^1 + \phi_{n,m} - \hat{e}_n^2 = e_n^2 - \hat{e}_n^2 \ . \tag{2.47}$$

Here, $\hat{e}_n^2$ represents the approximation of e_2 after forward and inverse transform and quantization[6]. For every block, the accumulated requantization errors (δ_n) are stored in the current picture buffer.
The compensation signal $\phi_{n,m}$ is constructed by applying intra prediction with mode m to the accumulated errors of the neighboring prediction blocks:

$$\phi_{n,m} = I_m(\delta_A, \delta_B, \delta_C, \delta_D) \ . \tag{2.48}$$

Here, the notation of blocks A, B, C, D corresponds to the notation used in the H.264/AVC specification, as recapitulated in Appendix A.

A single-frame buffer is maintained for the current picture, which is used for compensation of intra-predicted macroblocks. The buffer stores the accumulated requantization errors $\delta_{n,ij}$ (where $i, j = 0, \ldots, 3$). The error values from neighboring blocks are used to form a spatial compensation signal for the current block, according to the used intra prediction modes. This technique can be applied to both 4×4 and 16×16 prediction modes.

For 4×4 prediction mode 0 (vertical prediction), the prediction is formed based on the four prediction pixels of the 4×4 block above. By applying the prediction formulas to the relevant requantization difference values, i.e., for mode $m = 0$, the bottom row of pixels $\delta_{B,3,j}$ of block B (with j ranging from 0 to 3), the following compensation matrix is obtained (the notation n is dropped for notational simplicity for the compensation matrices):

$$\boldsymbol{\phi}_0 = \begin{bmatrix} \delta_{B,3,0} & \delta_{B,3,1} & \delta_{B,3,2} & \delta_{B,3,3} \\ \delta_{B,3,0} & \delta_{B,3,1} & \delta_{B,3,2} & \delta_{B,3,3} \\ \delta_{B,3,0} & \delta_{B,3,1} & \delta_{B,3,2} & \delta_{B,3,3} \\ \delta_{B,3,0} & \delta_{B,3,1} & \delta_{B,3,2} & \delta_{B,3,3} \end{bmatrix} \ . \tag{2.49}$$

[5] In the context of spatial compensation, n is regarded as the index of the current prediction block.

[6] Obviously, for single-loop architectures, the signals with notation ^{1} and ^{2} can no longer be strictly regarded as being 'decoder-side' or 'encoder-side'.

The compensation matrices for the other prediction modes are formed in a similar manner by applying the prediction formulas in Appendix A to the requantization differences contained in the surrounding blocks. Note that for 4×4 prediction, seven values at most (out of 16) are used for prediction. Hence, only these values need to be stored in memory.

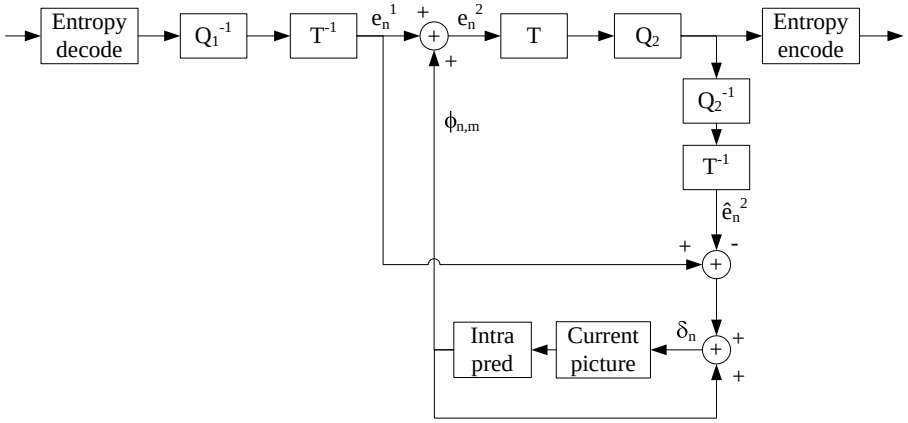


Figure 2.9: H.264/AVC requantization transcoder with spatial compensation.

2.3.3 Hybrid architecture with spatial and temporal compensation

To tackle both spatial and temporal drift terms, this architecture can be extended to the hybrid requantization transcoding architecture as shown in Figure 2.10. We proposed this technique, which can be applied to P and B pictures containing both MCP and intra-predicted macroblocks, in [56]. Depending on the macroblock type, a different type of compensation is used. Values of δ_n are firstly stored in the current frame buffer, and can be used to form the spatial drift compensation signal. After processing of each frame, the current frame buffer is added to the MCP reference frame buffer (in case the frame is used for future prediction). The same values can then be used to form the temporal drift compensation signal. For motion vector $\boldsymbol{v}$, this results in signal $\phi_{n,\boldsymbol{v}}$. Given that multiple reference frames can be used in H.264/AVC, the reference frame buffer is constructed in the same way as at the decoder side, containing several pictures with requantization errors. Since residual error coefficients are used, deblocking is not performed in the transcoder[7].

[7]Deblocking requires knowledge of the severity of block edges in the picture, which demands a full reconstruction [50].

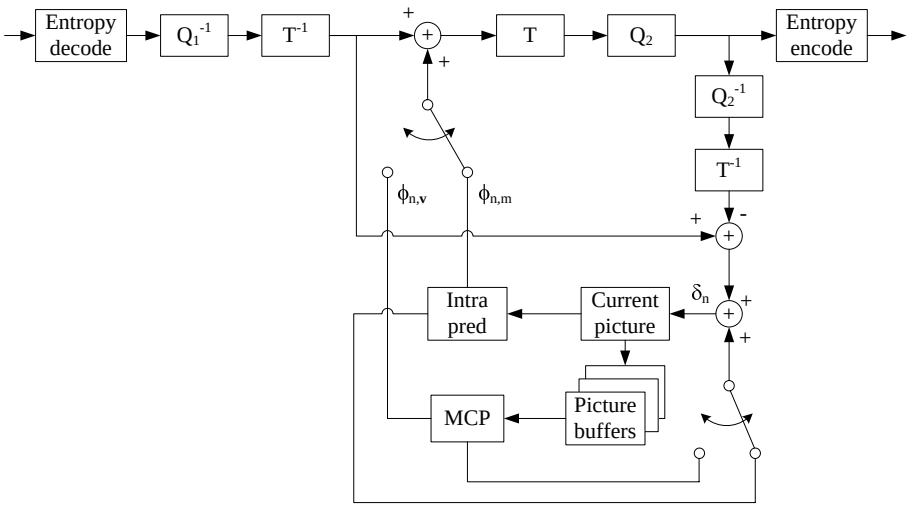


Figure 2.10: H.264/AVC hybrid requantization transcoder (spatial and temporal compensation).

2.3.4 Transform-domain optimizations

The presented architectures were further optimized by shifting operations to the transform domain. Transform-domain solutions that were applicable to MPEG-2, however, cannot be applied as such in H.264/AVC.

The possibility of using a pure transform-domain solution for intra prediction was discussed in [57]. This, unfortunately, resulted in formulas requiring a significant amount of floating-point operations, with high complexity for most of the intra prediction modes (in particular 4×4 modes 3 to 8). What's more, because of rounding errors, transform-domain intra prediction turns out not to be useful in practical transcoding situations.

For the same reasons (rounding errors and complexity), motion compensation is still performed in the pixel domain in the proposed solution, whereas for MPEG-2, MC-DCT can be used to speed up transcoding. In [8], an algorithm was discussed for MCP in the transform domain, used in the context of MPEG-2 to H.264/AVC transcoding. The authors, however, did not take into account the sub-pixel interpolation process. In this way, the algorithm could only be used for full-pixel accuracy. Adjusting the algorithm to quarter-pixel accuracy and the introduced dependencies would greatly increase the computational complexity. Together with the introduced rounding errors, this would render the transform-domain approach useless.

Another solution is to use a partial transform-domain architecture, as

shown in Figure 2.11 for the hybrid architecture. One inverse transform is still used, and MCP is applied in the pixel domain. A number of simplifications can be implemented, depending on the type of intra prediction applied.

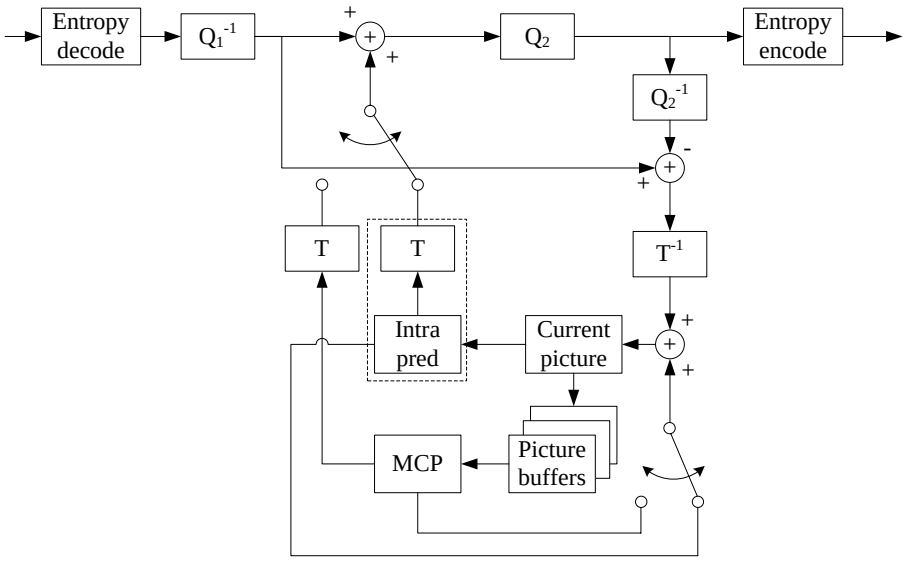


Figure 2.11: Partial transform-domain hybrid requantization transcoder.

A. Intra 4×4 prediction

For intra prediction, efficient compensation formulas can be obtained with lower complexity than the combination of intra prediction and forward transform. In particular, intra 4×4 modes 0 to 2 (vertical, horizontal, and DC prediction) benefit from this approach. Also, for modes 3 and 4 (diagonal modes), a lot of symmetry can found in the formulas. For these modes, the combination of intra prediction and transform can be used advantageously, as indicated by the dashed line in Figure 2.11.

For horizontal prediction (prediction mode 1), the prediction matrix ϕ_1 is constructed as follows, with $\delta_{A,i,3}$, $i = 0,\ldots,3$ being the 4 accumulated requantization error values in the rightmost column of the 4×4 block to the left of the current block (referred to as block A):

$$\boldsymbol{\phi}_1 = \begin{bmatrix} \delta_{A,0,3} & \delta_{A,0,3} & \delta_{A,0,3} & \delta_{A,0,3} \\ \delta_{A,1,3} & \delta_{A,1,3} & \delta_{A,1,3} & \delta_{A,1,3} \\ \delta_{A,2,3} & \delta_{A,2,3} & \delta_{A,2,3} & \delta_{A,2,3} \\ \delta_{A,3,3} & \delta_{A,3,3} & \delta_{A,3,3} & \delta_{A,3,3} \end{bmatrix} . \tag{2.50}$$

In order to obtain the transform-domain compensation matrix $\boldsymbol{\Phi}_1$, the H.264/AVC kernel integer transform is applied to the spatial-domain intra prediction matrix $\boldsymbol{\phi}_1$ as follows:

$$\boldsymbol{\Phi}_1 = T(\boldsymbol{\phi}_1) = \boldsymbol{C}_F \, \boldsymbol{\phi}_1 \, \boldsymbol{C}_F^T \, , \tag{2.51}$$

where $\boldsymbol{C}_F$ represents the kernel forward integer transform matrix. This results in the frequency-domain compensation $\boldsymbol{\Phi}_1$ matrix for the horizontal prediction mode, requiring only 2 shifts and 12 additions (instead of 64 additions and 16 shifts for the forward transform):

$$\boldsymbol{\Phi}_1 = 4 \begin{bmatrix} \delta_{A,0,3} + \delta_{A,1,3} + \delta_{A,2,3} + \delta_{A,3,3} & 0 & 0 & 0 \\ 2\,(\delta_{A,0,3} - \delta_{A,3,3}) + \delta_{A,1,3} - \delta_{A,2,3} & 0 & 0 & 0 \\ \delta_{A,0,3} - \delta_{A,1,3} - \delta_{A,2,3} + \delta_{A,3,3} & 0 & 0 & 0 \\ \delta_{A,0,3} - 2\,(\delta_{A,1,3} - \delta_{A,2,3}) - \delta_{A,3,3} & 0 & 0 & 0 \end{bmatrix} . \tag{2.52}$$

The transform-domain prediction matrix $\boldsymbol{\Phi}_0$ is found analogously, and only contains non-zero coefficients in the top row. For DC prediction (intra 4×4 prediction mode 2), only the top-left coefficient will be non-zero (equal to 16 times the DC value). The transform-domain intra prediction matrices for all 4×4 modes are given in Appendix A, Section A.2.1. The transform-domain *compensation* matrices are obtained by applying these formulas to the accumulated requantization errors $\delta_{X,ij}$ ($X \in \{A, B, C, D\}$).

B. Intra 16×16 prediction

Similar simplified compensation matrices can be constructed for the intra 16×16 modes, where a great deal of symmetry can be found in the transformed intra prediction formulas for all modes.

After intra 16×16 prediction, the prediction differences are transformed as follows. Firstly, the 4×4 integer transform is applied to the 16 4×4 blocks in the macroblock. Secondly, the 16 DC coefficients are grouped in a 4×4 matrix. Subsequently, the forward Hadamard transform is applied to this DC coefficient matrix with Hadamard matrix $\boldsymbol{H}_4$:

$$\boldsymbol{H}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} . \tag{2.53}$$

As a result, one transformed DC matrix $\boldsymbol{\Phi}_{m,DC}$ (for mode $m = 0, 1, 2, 3$) and 16 transformed AC matrices $\boldsymbol{\Phi}_{m,AC,l}$ with $l = 0, 1, \ldots, 15$ are obtained.

For the DC mode (mode 2) it suffices to compensate the top-left position in the transform-domain DC matrix $\boldsymbol{\Phi}_{0,DC}$, while no compensation is required for the 16 AC 4×4 matrices (all zero matrices).

$$\boldsymbol{\Phi}_{0,DC} = 4 \begin{bmatrix} \sum_{i=0}^{15} \delta_{A,i,15} + \sum_{j=0}^{15} \delta_{B,15,j} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \tag{2.54}$$

For horizontal 16×16 prediction (mode 1), the transform-domain compensation matrix $\boldsymbol{\Phi}_{1,DC}$ for the Hadamard DC matrix becomes:

$$\boldsymbol{\Phi}_{1,DC} = 8 \begin{bmatrix} \alpha+\beta+\gamma+\delta & 0 & 0 & 0 \\ \alpha+\beta-\gamma-\delta & 0 & 0 & 0 \\ \alpha-\beta-\gamma+\delta & 0 & 0 & 0 \\ \alpha-\beta+\gamma-\delta & 0 & 0 & 0 \end{bmatrix} , \tag{2.55}$$

where

$$\begin{aligned} \alpha &= \sum_{i=0}^{3} \delta_{A,i,15} , \quad \beta = \sum_{i=4}^{7} \delta_{A,i,15} , \\ \gamma &= \sum_{i=8}^{11} \delta_{A,i,15} , \quad \delta = \sum_{i=12}^{15} \delta_{A,i,15} , \end{aligned} \tag{2.56}$$

with $\delta_{A,i,15}$, $i = 0, \ldots, 15$ being the 16 pixels in the rightmost column of the macroblock to the left of the current macroblock. The compensation matrix $\boldsymbol{\Phi}_{1,AC}$ for the four 4×4 blocks in the top row of the macroblock becomes:

$$\boldsymbol{\Phi}_{1,AC} = 4 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2\left(\delta_{A,0,15} - \delta_{A,3,15}\right) + \delta_{A,1,15} - \delta_{A,2,15} & 0 & 0 & 0 \\ \delta_{A,0,15} - \delta_{A,1,15} - \delta_{A,2,15} + \delta_{A,3,15} & 0 & 0 & 0 \\ \delta_{A,0,15} - 2\left(\delta_{A,1,15} - \delta_{A,2,15}\right) - \delta_{A,3,15} & 0 & 0 & 0 \end{bmatrix} , \tag{2.57}$$

with analog compensation matrices for the other 4×4 blocks. The compensation matrices for all modes are given in Section A.2.2.

The impact on the performance of requantization transcoders by using these techniques was discussed in [58, 59].

C. Motion-compensated prediction

A similar technique could be applied by forming a combination of MCP with motion vector $\boldsymbol{v}$ (resulting in the prediction matrix $\boldsymbol{\phi_v}$) and forward transform, hence obtaining the transform-domain compensation matrix $\boldsymbol{\Phi_v}$. A distinction can be made depending on whether the motion vector corresponds to a full-pixel, half-pixel, or quarter-pixel displacement. For the full-pixel case, the combination boils down to applying the H.264/AVC transform to the displaced block $\boldsymbol{\phi_v}$ of reference frame error values (no benefit is obtained by shifting the MCP to the transform domain). For half-pixel and quarter-pixel displacement, the matrix $\boldsymbol{\Phi_v}$ contains an increased number of calculations (in particular multiplications), when compared to the 16×2 multiplications (2 multiplications per fractional pixel) required in the pixel-domain interpolation formulas, given the 6-tap filter with symmetric weights $(1, -5, 20, 20, -5, 1)/32$.

Since interpolation is performed on (accumulated) requantization error values, another question is whether it is beneficial to use a relatively complex 6-tap Wiener filter during subpixel interpolation. In [60, 61], the problem of aliasing is mentioned as the main reason for using a Wiener filter with 6 or 8 taps instead of a bilinear filter. The 6-tap H.264/AVC interpolation filter is based on the separable 2-D Wiener filter proposed in [61], which was used to reduce drift caused by aliasing for multiresolution hybrid video coding.

Complexity can be reduced, however, by using a bilinear filter instead of a 6-tap filter. When compared to the 6-tap Wiener filter, a bilinear filter has higher passband attenuation and stopband permeability, resulting in a less accurate low-pass filter. Apart from arithmetic differences between the 6-tap and bilinear filters, the use of a bilinear filter will lead to more aliasing in the predicted signal for sub-pixel positions. If low complexity transcoding is required, however, the use of a bilinear interpolation filter could be considered in order to reduce complexity, as discussed in [62].

2.3.5 Overall architectures

Since the spatial and temporal compensation is not perfect due to non-linear operations, some errors can still accumulate and propagate. This is especially troublesome for intra-coded macroblocks, which contain a high number of dependencies (even within a single macroblock in the case of 4×4 intra pre-

diction). In intra-coded pictures (consisting entirely out of intra-coded macroblocks), the situation is the most severe. For optimum transcoding performance, 'mixed' architectures can be derived. By using a cascaded decoder-encoder architecture for intra-coded pictures, more reliable (drift-free) reference frames are formed. In [7], this possibility was examined. There, a mixed requantization architecture (MRA) was discussed but resulted in unreliable results, due to the absence of spatial compensation for intra-coded macroblocks in P and B pictures.

We introduced the use of mixed architectures with spatial compensation and/or temporal compensation in [63]. Depending on the picture and/or macroblock type, a different processing technique can be selected. Different overall architectures are hence obtained. The use of mixed architectures turns out to be a powerful technique for trading off complexity and rate-distortion performance, which will also be applied for H.264/AVC-to-SVC transcoding and spatial resolution reduction transcoding (as discussed in Chapter 3 and Chapter 4, respectively).

The overall architectures are determined by combining techniques for the individual picture/macroblock types.

Intra-coded pictures Intra-coded pictures have the benefit that reconstruction to the pixel domain can be executed with relatively low computational requirements. Single-loop compensation or open-loop transrating can also be applied, but result in reduced quality. Hence, given the relatively low extra computational cost, using the CPDT will be preferred for most applications.

Intra-coded macroblocks in P and B pictures Reconstruction of intra-coded macroblocks in P and B pictures has the advantage of being drift-free, but requires reconstruction of the surrounding macroblocks, in order to have the prediction pixels available for intra prediction. If one or more of the surrounding macroblocks uses MCP, a dependency on temporally predicted macroblocks is introduced, and reconstruction of MCP macroblocks becomes necessary[8]. Single-loop spatial compensation of these intra-coded macroblocks, however, can be applied without imposing reconstruction of surrounding MCP macroblocks, and results in highly improved quality over open-loop requantization.

MCP macroblocks For MCP macroblocks, a choice can be made between CPDT reconstruction (if the blocks upon which the current block depends are

[8]Note that this is not the case when the *constrained intra prediction flag* is enabled in the active picture parameter set of the H.264/AVC stream.

also reconstructed), temporal compensation, or open-loop transrating. Adding temporal compensation or CPDT reconstruction introduces extra computational complexity, as discussed further in Section 2.3.6.

Note that an even finer categorization can be applied, e.g., based on the used GOP structure. For the example case of hierarchical coding structures, pictures that are located lower in the temporal hierarchy are eligible for higher-complexity temporal compensation or pixel-domain reconstruction, while pictures in the highest temporal layers can be transrated open-loop with little effect on the quality of the output video. This is discussed in more detail in [64].

By taking into account the restrictions and remarks mentioned above, the following overall architectures are obtained. As a reference, CPDT can be applied to all picture and macroblock types. In the remainder of this chapter, this architecture will simply be referred to as 'CPDT' for brevity. The architecture which uses open-loop transrating for all blocks is denoted as 'OL'. By using the CPDT architecture for intra-coded pictures, spatial compensation for the remaining intra macroblocks (in P and B pictures), and open-loop requantization for MCP macroblocks, the MRA-SC architecture is obtained (mixed requantization with spatial compensation). This architecture will prove to provide significantly improved rate-distortion performance over open-loop transrating at a relatively low computational complexity cost. The architecture which incorporates both spatial and temporal compensation (hybrid compensation) in the MRA architecture will be referred to as MRA-Hybrid. The more traditional architecture with only temporal compensation is indicated as MRA-TC (MRA with temporal compensation) in the results section.
The techniques used for the 'CPDT', 'OL', 'MRA-TC', 'MRA-SC', and 'MRA-Hybrid' architectures are summarized below.

A. CPDT architecture

- CPDT for intra-coded pictures.
- CPDT for MCP macroblocks.
- CPDT for intra-coded macroblocks in P and B pictures.

B. OL architecture

- Open-loop for intra-coded pictures.
- Open-loop for MCP macroblocks.

- Open-loop for intra-coded macroblocks in P and B pictures.

C. MRA-SC architecture

- CPDT for intra-coded pictures.
- Open-loop for MCP macroblocks.
- Spatial compensation for intra-coded macroblocks in P and B pictures.

D. MRA-TC architecture

- CPDT for intra-coded pictures.
- Temporal compensation for MCP macroblocks.
- Open-loop for intra-coded macroblocks in P and B pictures.

E. MRA-Hybrid architecture

- CPDT for intra-coded pictures.
- Temporal compensation for MCP macroblocks.
- Spatial compensation for intra-coded macroblocks in P and B pictures.

2.3.6 Complexity discussion

The complexity of the overall architecture is determined by the techniques applied to the individual picture/macroblock types, i.e., intra-coded pictures, intra-coded macroblocks in P and B pictures, and MCP macroblocks.

Using CPDT for intra-predicted pictures results in a minor increase in complexity of the overall architecture. Given the low complexity of the intra prediction process (when compared to MCP), reconstruction of intra-coded pictures does not significantly impact the computational complexity of the overall architecture. As will be seen in the results section (Section 2.3.7), adding spatial compensation for intra-coded macroblocks in P and B pictures has only limited impact on the computational complexity.

Adding temporal compensation, however, requires a number of time-consuming operations in H.264/AVC.

- **Motion vector derivation** In H.264/AVC, median motion vector prediction is included, based on the motion vectors of surrounding (sub)macroblock partitions. This context-adaptivity strongly increases

complexity when decoding motion vectors. Motion vector derivation is even further complicated in the case of Skip and Direct blocks.

- **Interpolation** For quarter-pixel accuracy MCP, half-pixel values are obtained by using a 6-tap interpolation filter. This can be a complex operation, in particular for bipredictionally MCP blocks, with MVs pointing to half- or quarter-pixel positions.
- **Reference picture management** A number of algorithms have to be executed for reference picture management, such as decoded reference picture marking and reference picture list reordering. The use of multiple reference pictures for MCP severely increases memory requirements.

2.3.7 Results and discussion

A. Rate-distortion results

In order to evaluate the rate-distortion performance of the discussed architectures, we transcoded sequences with varying characteristics and spatial resolution: *Foreman* (QCIF resolution), *Stefan* (CIF), *Soccer* (4CIF), and *Stockholm* (720p). The sequences were encoded using the H.264/AVC Joint Model reference software (version 13.2) using default coding tools, Main profile, 5 reference pictures, CABAC entropy coding, and full rate-distortion optimization enabled. Two GOP structures were used, one with IBBP GOP (length 15 pictures), the other with hierarchically coded pictures (length 16 pictures). The sequences were encoded with starting QP_I values (for I pictures) 22, 27, 32, and 37, $QP_P = QP_I + 1$ (P pictures) and $QP_B = QP_I + 2$ (B pictures). These correspond to the values used in the VCEG common test conditions [65]. In the remainder of the section, the notation QP_1 corresponds to the starting QP for the I pictures.

The proposed transcoder architectures were implemented in software, and were used to generate transcoded streams. The output streams were obtained by requantizing with increased quantization parameters, i.e., $QP_2 = QP_1 + i$, with $i = 1, \ldots, 6$, for all picture types. By using fixed increases of the QP, the impact of rate control algorithms is eliminated in the performance of the transcoder algorithms. In all graphs, the R-D point corresponding to the original streams is added. Comparison with the rate points of the transcoded streams gives more insight into the overall rate-distortion impact of the presented architectures.

In Figure 2.12, rate-distortion results are shown for the *Stefan* sequence. It is clear that open-loop transcoding is not usable for H.264/AVC. The loss of information due to requantization results in unpredictable drift, which is

driven by the combination of temporal and, in particular, spatial prediction. The OL architecture results in PSNR values which are well below all levels of acceptability, with values of 28 dB down to 20 dB and lower. It is questionable whether this range of PSNR values can be regarded as being meaningful. Nonetheless, we present the OL results for completeness. Also, due to the randomness of open-loop drift, the R-D curves for OL can obtain a non-monotonic behavior. When increasing the QP, a different quantization step size is used, and drift characteristics will change in an unpredictable way.

By using the MRA-TC architecture (CPDT architecture for intra-coded pictures, and temporal compensation for MCP blocks), the output quality is already significantly improved relative to open-loop transcoding. Still, a gap of about 3 to 4 dB exists when compared to CPDT transcoding. These findings correspond to the results in [7]. Also, spatial drift might still result in non-monotonic behavior of the R-D curves, e.g., for the *Stockholm* sequence in Figure 2.16.

Remark that a large gap in bit rate occurs between the R-D points for $\Delta QP = 3$ and $\Delta QP = 4$, in particular for the open-loop transcoder. This is explained by the removal of small coefficients from the bitstream, when a given threshold of ΔQP is crossed during requantization. In particular, for typical values of $\epsilon = 1/3$ (intra coding) or $1/6$ (inter coding), requantization of coefficients with absolute value equal to one results in

$$Q'_2(Q_1^{-1}(1)) = 1 \tag{2.58}$$

for $\Delta QP \leq 3$, while

$$Q'_2(Q_1^{-1}(1)) = 0 \tag{2.59}$$

for $\Delta QP \geq 4$. It can be shown that this property holds irrespective of the starting QP_1, for the given ϵ value. Due to the accurate prediction in H.264/AVC, residual data consists to a large extent out of coefficients with small absolute value (in particular 1 and -1). When these coefficients disappear from the bitstream, a large decrease in bit rate occurs.

When looking at visual results, distinct visual artifacts still appear in MRA-TC-transcoded pictures containing intra-coded macroblocks or regions (see Figure 2.13(a) and Figure 2.13(c)). Drift can be noticed in several regions such as Foreman's cheek and helmet (Figure 2.13(a)), and around Stefan's head and the IBM logo (Figure 2.13(c)). Applying spatial compensation (MRA-SC) resolves this issue, and improves objective quality by 1 to 2 dB. Since only intra prediction is required for spatial compensation, and MCP is avoided (MCP is significantly more complex than intra prediction, among others due to the interpolation process), MRA-SC can be regarded as a low-complexity

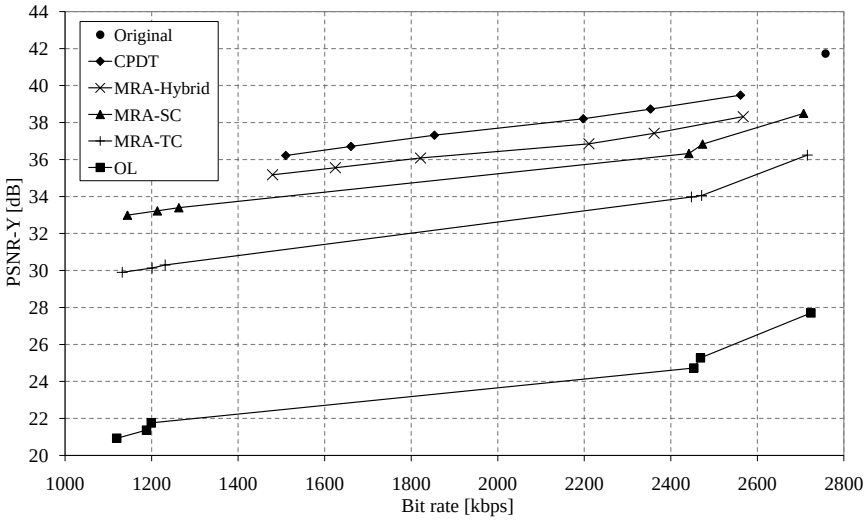


Figure 2.12: Rate-distortion results (*Stefan*, CIF, 30Hz, $QP_1 = 22$).

transcoder solution. The hybrid architecture with spatial and temporal compensation (MRA-Hybrid) results in additional gains of about 0.5 to 1 dB, depending on the motion characteristics of the sequence. When looking at visual results, artifacts are removed in the pictures, as can be seen from Figure 2.13(b) and Figure 2.13(d).

For low-motion sequences, the benefit (in a rate-distortion sense) of adding temporal compensation to the MRA-SC architecture vanishes for higher bit rate reductions ($\Delta QP > 3$), as can be seen from the detail (the OL architecture is removed from this graph) in Figure 2.14. Here it can be seen that, although temporal compensation in MRA-Hybrid improves the PSNR values for corresponding R-D points (with identical ΔQP), the amount of residual data increases rapidly (due to the reintroduction of small residual coefficients), resulting in superior rate-distortion performance of the MRA-SC architecture.

Results for the *Soccer* (4CIF) and *Stockholm* sequence (720p) are shown in Figure 2.15 and Figure 2.16, and are similar to the results for the lower resolutions. A gap of about 1 dB is found for the MRA-Hybrid architecture when compared to the cascaded decoder-encoder approach. Spatial compensation proves indispensable in order to obtain acceptable rate-distortion performance.

(a) Spatial drift when using MRA-TC architecture (detail of *Foreman*, CIF, frame 138, $QP_1 = 22$, $QP_2 = 26$)

(b) Spatial drift is removed after adding spatial compensation (MRA-Hybrid) (detail of *Foreman*, CIF, frame 138, $QP_1 = 22$, $QP_2 = 26$)

(c) Spatial drift when using MRA-TC (detail of *Stefan*, CIF, frame 177, $QP_1 = 22$, $QP_2 = 26$)

(d) Spatial drift is removed after adding spatial compensation (MRA-Hybrid) (detail of *Stefan*, CIF, frame 177, $QP_1 = 22$, $QP_2 = 26$)

Figure 2.13: Transcoded P pictures without and with spatial compensation.

B. Computational complexity analysis - timing results

In Table 2.5, timing results are shown for the different transcoder architectures. These results are obtained from our transcoder software. Although the code is non-optimized, the following results serve as an indication of the relative complexity of the techniques and architectures (all modules were implemented using a similar 'degree of optimization'). The processing speed is shown in frames per second. The results (in this section and the remainder of the text) were generated on a platform with an Intel Xeon X5355 processor operating

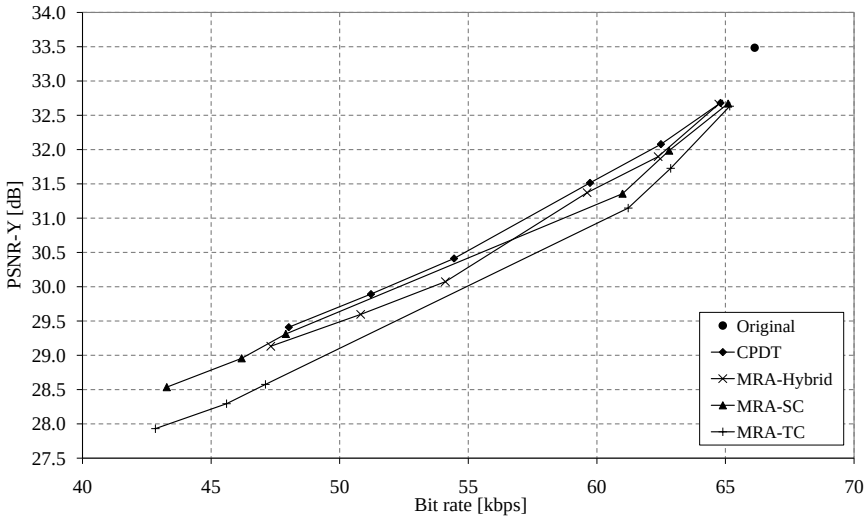


Figure 2.14: Rate-distortion results (*Foreman*, QCIF, 30Hz, $QP_1 = 32$).

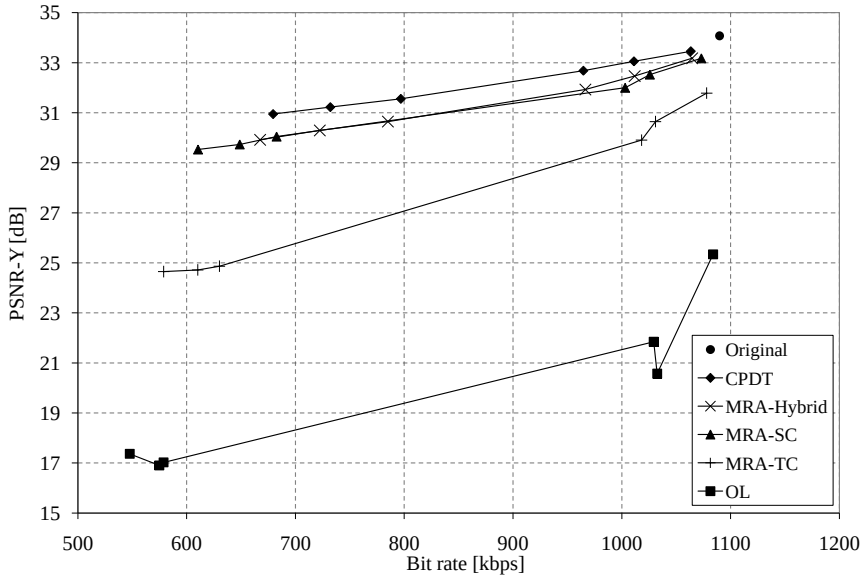


Figure 2.15: Rate-distortion results for transcoding architectures (*Soccer*, 4CIF, $QP_1 = 32$).

at 2.66 GHz, with 16 GB RAM. A Microsoft Windows XP environment was used.

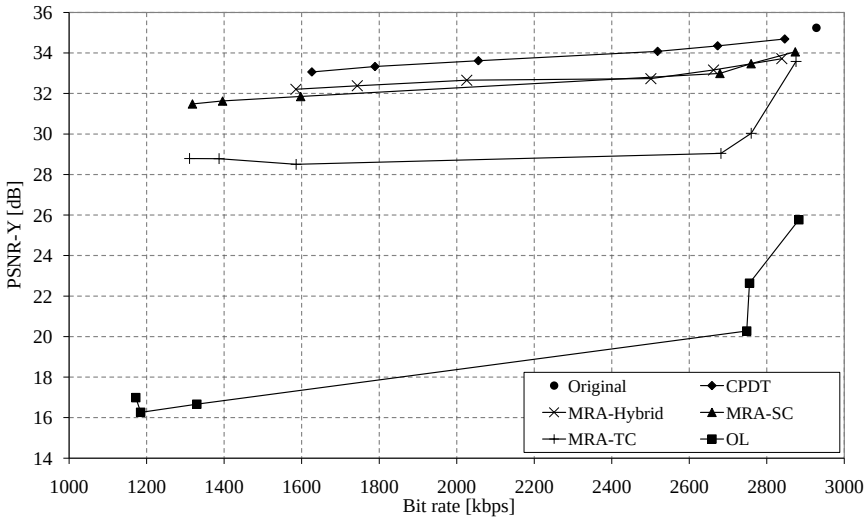


Figure 2.16: Rate-distortion results (*Stockholm*, 720p, $QP_1 = 27$).

Table 2.5: Processing speed for transcoding architectures [fps]

	Foreman (QCIF)	*Stefan* (CIF)	*Soccer* (4CIF)	*Stockholm* (720p)
CPDT	21.5	4.8	1.4	0.5
MRA-Hybrid	34.9	7.5	2.3	0.8
MRA-TC	35.6	7.7	2.5	0.8
MRA-SC	277.5	32.2	10.1	6.7
OL	410.8	40.1	16.4	8.9

It can be seen from the difference between MRA-TC and MRA-Hybrid that adding spatial compensation only has a minor impact on the processing speed. Traditional temporal compensation, however, has a large effect for H.264/AVC requantization transcoding, for the reasons mentioned in Section 2.3.6. The MRA-SC architecture is able to obtain transcoding speeds up to 80% of the open-loop transcoder, with significantly improved visual and rate-distortion results (as shown in the previous section).

2.4 High profile transrating

2.4.1 H.264/AVC High profile

So far, transrating was studied for the 4×4 integer transform [40], which was introduced in the first version of the H.264/AVC specification (July 2003). In this first version, three profiles were defined, i.e., the Baseline, Main, and Extended profiles. In the 2005 version, among others the High profile was added, hereby providing support for professional applications. Originally, the tools that were added to implement this functionality were labeled as 'Fidelity Range Extensions' (FRExt), indicating the support for higher-resolution, high-fidelity, or professional applications. Changes in the coding tools were targeted at providing significant improvements in coding efficiency for higher-resolution video material. The main difference between FRExt and non-FRExt H.264/AVC coding is the use of an 8×8 transform in addition to the 4×4 transform [66]. The impact of transrating on H.264/AVC High profile-coded sequences is discussed in the remainder of this section. The question is if the behavior of transrating architectures can be extended to higher-resolution content, which is typically encoded using High profile.

We discuss the case of High profile separately for a number of reasons. Firstly, due to the introduction of new coding tools, coding decisions will be altered. Secondly, the new intra prediction modes might change the (primarily spatial) drift behavior in High profile sequences. Also, given the importance of higher resolutions in future broadcasting and professional environments, a separate discussion for these applications and this type of content is justified. Additionally, high-resolution sequences pose increasing demands on (transcoding) hardware requirements. Given the high buffer demands and high number of macroblocks in these pictures, processing speed of reencoding techniques will drop rapidly. For real-time adaptation, fast transcoding techniques become even more important.

In future development of new standards, High profile will be regarded as the reference point, given its increased coding performance over Main profile. Even stronger, Main profile can be regarded as being 'outdated' by High profile (given that High profile tools are a superset of Main profile coding tools), hence the latter should be preferred in all practical video coding systems.

Note that, although the use and usefulness of High profile is not limited to high-resolution content, we primarily examine the case of higher resolutions. We investigate the tools that are common to all High profiles, and restrict the discussion to the case of 4:2:0 chroma subsampling. Requantization for 4:4:4 coding can be the focus of future research.

2.4.2 H.264/AVC High profile tools

A number of changes have been included in the High profile, when compared to the original Main profile. We briefly give an overview of these new tools.

A. 8×8 transform

Besides the integer 4×4 transform, which was originally provided in the H.264/AVC design [40], an 8×8 transform was added. The 4×4 integer transform has the advantage of reducing ringing artifacts in the video sequence. Longer base functions, however, become more important for high-fidelity content, for preservation of fine details and textures. The 8×8 transform is only used for prediction block sizes larger than or equal to 8×8 pixels, hereby avoiding a transformation across prediction block boundaries. High profile allows adaptive selection between the 4×4 and 8×8 transform (the used transform is indicated by the *transform size 8 × 8 flag* syntax element). The 2-D 8×8 transform is specified in a separable way as a 1-D horizontal row transform, followed by a 1-D vertical column transform, hereby allowing hardware-friendly butterfly operations. Normalization is required as is the case for the 4×4 transform. By including the normalization operation in the quantization the 8×8 integer transform is obtained:

$$\boldsymbol{C}_8 = 1/8 \cdot \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} .$$

The forward core transform can be written as

$$\boldsymbol{Y} = \boldsymbol{C}_8 \, \boldsymbol{X} \, \boldsymbol{C}_8^T \, . \tag{2.60}$$

The inverse 8×8 transform is specified as

$$\widehat{\boldsymbol{X}} = \boldsymbol{C}_8^T \, \widehat{\boldsymbol{Y}} \, \boldsymbol{C}_8 \, . \tag{2.61}$$

Post- and prescaling operations are postponed to the quantization, as is done for the 4×4 transform.

B. Quantization

Quantization in High profile was changed to include the possibility of using adaptive quantization matrices. In the most commonly used case, however, the *seq_scaling_matrix_present_flag* syntax element is set to false, and the default quantization matrices are used. In the latter case, forward quantization can be implemented in a similar way as for the 4×4 transform:

$$\begin{cases} |Z_{ij}| = (|W_{ij}|\ M_{8,ij} + \epsilon\, 2^{qbits}) \gg qbits \\ \text{sign}(Z_{ij}) = \text{sign}(W_{ij}) \end{cases} . \tag{2.62}$$

The $qbits$ parameter is slightly changed and equals $16 + \lfloor QP/6 \rfloor$, and

$$M_{8,ij} = \text{round}\Big((2^{16}\, E_{8,F,ij})/Q_{\text{step}}\Big) . \tag{2.63}$$

The values of $M_{8,ij}$ are displayed in Table 2.6. Inverse quantization is specified as:

$$\widehat{W}_{ij} = Z_{ij}\,(16\, V_{8,ij})\, 2^{\lfloor QP/6 \rfloor - 6} \tag{2.64}$$

where the value of 16 can be replaced by adaptive values as specified by scaling matrices in the bitstream. Inverse quantization multiplication factors are defined as

$$V_{8,ij} = \text{round}\Big(\frac{256\, Q_{\text{step}}\, E_{8,I,ij}}{2^{\lfloor QP/6 \rfloor}}\Big) . \tag{2.65}$$

The resulting values are shown in Table 2.7.

Table 2.6: Forward quantization multiplication factors $M_{8,ij}$.

	$M_{8,ij}$					
QP % 6	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$
0	13107	11428	20972	12222	16777	15481
1	11916	10826	19174	11058	14980	14290
2	10082	8943	15978	9675	12710	11985
3	9362	8228	14913	8931	11984	11259
4	8192	7346	13159	7740	10486	9777
5	7282	6428	11570	6830	9118	8640

C. Intra 8×8 prediction

As explained in the previous sections, in H.264/AVC intra macroblocks are not only allowed in intra-coded pictures, but also in P and B pictures. In Main profile, only intra 4×4 and 16×16 modes are available. The High profile,

Table 2.7: Inverse quantization multiplication factors $V_{8,ij}$.

	$V_{8,ij}$					
QP % 6	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$
0	20	18	32	19	25	24
1	22	19	35	21	28	26
2	26	23	42	24	33	31
3	28	25	45	26	35	33
4	32	28	51	30	40	38
5	36	32	58	34	46	43

however, also allows 8×8 intra prediction modes. By introducing these extra intra prediction modes, the gap between 4×4 and 16×16 intra prediction is closed. As a result, energy compaction by using larger transform blocks can be traded off against the size of the prediction block [67]. Since intra macroblocks can be included in any picture type (I, P, or B), precautions need to be taken to avoid drift.

As for 4×4 intra prediction, nine modes are allowed for 8×8 prediction. For the skew modes, which were only used for 4×4 prediction before FREext, precautions need to be taken since these modes might introduce visible artifacts into the prediction signal. To reduce the artifacts, the pixels used for prediction are low-pass filtered.

D. Entropy coding

Due to the enlarged size of the coding blocks, changes were made to the entropy coding. A large number of context models were added to allow context-based adaptive binary arithmetic coding (CABAC) for the new coding tools. The large increase is primarily needed for residual coefficient coding, given the larger block size introduced in FREext, and for 4:4:4 coding.

2.4.3 Requantization for High profile

A. Requantization for 8×8 prediction and 8×8 transform

Given the additional 8×8 transform and corresponding quantization, the requantization formulas and multiplier coefficients should be reconsidered. By combining the forward and inverse quantization formulas, it can be seen that

by using the default formulas an upscaling by the following factor will occur:

$$(\frac{E_{8,F,ij}}{Q_{\text{step}}})\,(\frac{16 \cdot 256\, Q_{\text{step}}\, E_{8,I,ij}}{64}) = 64\, E_{8,F,ij}\, E_{8,I,ij}\ . \tag{2.66}$$

The multiplication factors can be normalized as follows, resulting in the changed (position-independent) values of M'_8 and V'_8, as shown in Table 2.8:

$$M'_8 = 2^{13+\lfloor \frac{QP}{6} \rfloor}\, \frac{1}{Q_{\text{step}}} \tag{2.67}$$

and

$$V'_8 = \frac{32\, Q_{\text{step}}}{2^{\lfloor QP/6 \rfloor}}\ . \tag{2.68}$$

As a result, $M'_8\, V'_8 \gg 16 \gg 2 = 1$ (with the first shift operation being executed during forward (re)quantization, and the second during inverse quantization).

Table 2.8: Modified (position-independent) multiplier coefficients M'_8 and V'_8 for 8×8 transform block requantization.

$QP\%6$	M'_8	V'_8
0	13107	20
1	11916	22
2	10082	26
3	9362	28
4	8192	32
5	7282	36

B. Spatial drift

Both the temporal and spatial drift terms as found in Equation (2.45) and Equation (2.46) still apply. Spatial drift becomes more and more of an issue when the resolution increases, for the evident reason that spatial drift can propagate over a larger number of (macro)blocks. Intra-coded pictures can be reencoded at low cost using the CPDT architecture; for P and B pictures, however, large regions of intra-coded macroblocks are especially troublesome. This is the case, e.g., due to large changes in illumination between successive frames. Also, when the temporal distance between pictures and the available reference pictures increases, and little temporal correlation is found between the current and reference frame, intra-coded macroblocks are inserted more often. This can be noticed for example in the pictures coded in the temporal base layer of

hierarchical GOP structures. When scene cuts do not coincide with the intra-coded frames (for example, in the case of a fixed GOP structure), it is likely that a P picture is encountered consisting only out of intra-coded macroblocks (a B picture still has at least one reliable reference picture available). In the last case, even single-layer compensation techniques will result in visual degradation due to rounding errors in the non-linear intra prediction formulas.

2.4.4 Architectures for High profile transrating

For transrating of high-resolution sequences, spatial drift cannot be ignored. When large intra-coded areas are present in the sequences, requantization errors will accumulate and cause significant spatial drift. In the remainder of the section, we investigate transrating architectures that are useful for High profile sequences. It is clear that open-loop transrating can be excluded as a possible approach for intra-coded macroblocks. Reencoding using the CPDT architecture can be used as a reference.

A. Single-loop spatial compensation

8×8 compensation matrices can be derived similar to the 4×4 matrices derived above. A difference is the introduction of the low pass filter, leading to an extra source of rounding errors since integer arithmetic will, in certain cases, result in

$$\left\lfloor \frac{p_{n-1} + 2\,p_n + p_{n+1}}{4} \right\rfloor \neq \left\lfloor \frac{\hat{p}_{n-1} + 2\,\hat{p}_n + \hat{p}_{n+1}}{4} \right\rfloor + \left\lfloor \frac{e_{n-1} + 2\,e_n + e_{n+1}}{4} \right\rfloor , \tag{2.69}$$

where p_n and $\hat{p}_n$ represent the prediction pixels before and after requantization, respectively, and

$$p_n = \hat{p}_n + e_n \; . \tag{2.70}$$

We will show that nonetheless, this technique leads to acceptable results for most sequences and coding configurations.

The architecture is easily extended and shown in Figure 2.17, where N can be replaced by four or eight. T_N and Q_N represent 4×4 and 8×8 transform and quantization. 8×8 intra prediction includes the low pass filter.

B. Selective requantization

Since spatial propagation is pivotal in the accumulation of errors, and drift cannot be avoided due to requantization, it might be beneficial to avoid applying

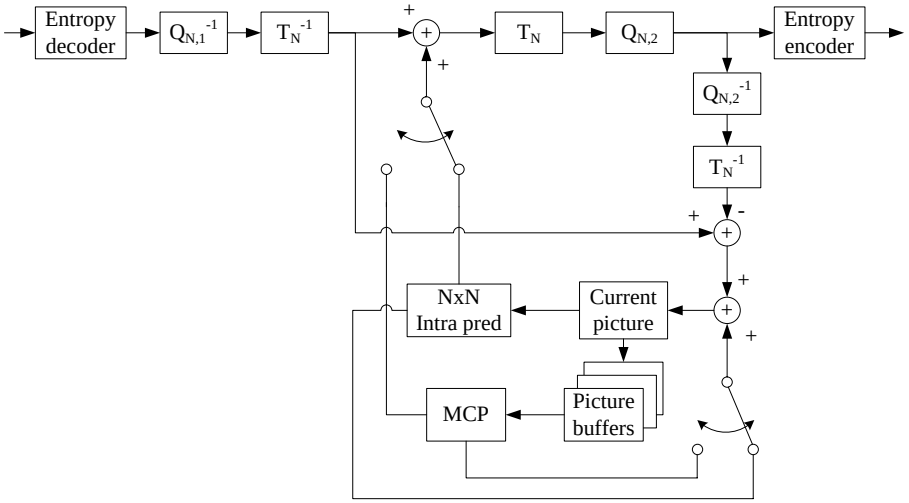


Figure 2.17: High profile requantization transcoder with single-loop spatial and temporal compensation.

requantization to intra-coded macroblocks. Since H.264/AVC allows quantization parameter selection on a macroblock basis (by varying the *mb_qp_delta* syntax element), one might choose not to requantize intra-coded macroblocks, and avoid spatial drift altogether. Although this leads to smaller transrating ratios (higher output bit rates), the absence of spatial drift proves to be a viable solution for high-quality transrating. This is a technique well-suited for small bit rate reductions. Larger reductions would lead to visibly different quality regions in the transrated sequences. In the results section, the following architecture is added in the comparison, which avoids requantization for intra-coded macroblocks:

MRA-Selective architecture

- CPDT for intra-coded pictures.
- Temporal compensation for MCP macroblocks.
- No requantization of intra-coded macroblocks in P and B pictures.

This architecture benefits from both temporal compensation and the absence of spatial drift. Its range of achievable bit rates, however, will be limited when compared to, e.g., the MRA-Hybrid architecture. Other combinations, however, might be introduced to overcome this issue. A combination of spatial

compensation and selective requantization for intra-coded macroblocks could for example be used to increase this range.

2.4.5 High profile transrating: results and discussion

For the tests, high-resolution sequences with various types of motion content were used. For 1080p content (1920x1080 pixels), we used the *CrowdRun* sequence, which is obtained from the commonly used HD test set provided by Sveriges Television (SVT). For 720p (1280x720 pixels), the *Night*, *ShuttleStart*, and *Crew* sequences were transrated. 160 frames were used from each sequence. For the *ShuttleStart* sequence, the first 300 (motionless) frames were skipped. Screenshots of the used sequences are shown in Figure 2.18. An IBBP GOP structure was used with a length of 16 pictures. IDR pictures were enabled to provide random access. Rate-distortion optimization was used, all prediction modes enabled with adaptive 4×4 or 8×8 transform selection, and CABAC entropy coding. Starting QP_1 values of 22, 27, 32, and 37 were used. In all graphs, the original rate point is included (highest rate point). As in Section 2.3.7, values of $QP_2 = QP_1 + i$ with $i = 1, \ldots, 6$ were used. The six lowest rate points correspond to these values of QP_2.

(a) *Night* sequence (1280x720)

(b) *Crew* sequence (1280x720)

(c) *ShuttleStart* sequence (1280x720)

(d) *CrowdRun* sequence (1920x1080)

Figure 2.18: Screenshots from used high definition sequences.

A. Rate-distortion results

Results for the *ShuttleStart* sequence with starting QP of 22 are shown in Figure 2.19. The curves for the hybrid single-loop compensation and selective requantization architectures mostly coincide. The bit rates for selective requantization are slightly higher, which can be expected due to the higher amount of intra residual data in the bitstream. The loss when compared to the cascaded architecture is about 0.5 dB.

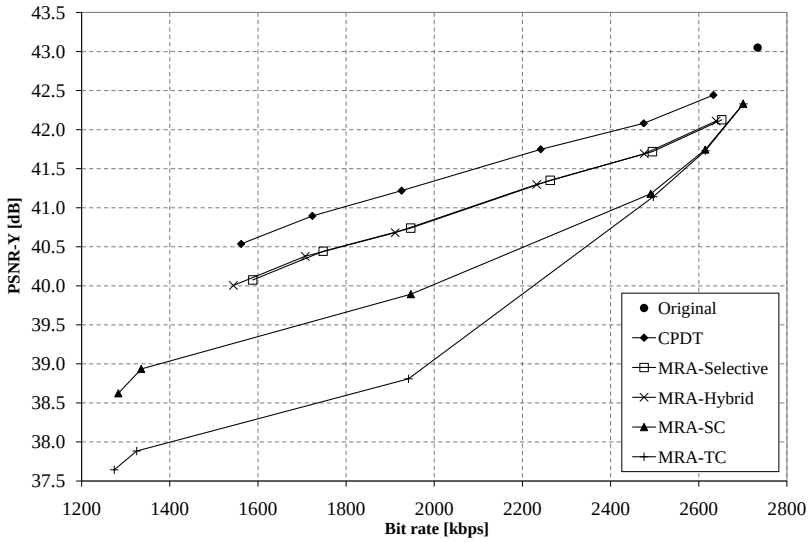


Figure 2.19: Rate-distortion results for *ShuttleStart* sequence ($QP_1 = 22$).

For a starting QP of 32 (Figure 2.20), losses are even smaller. Here, single-loop compensation outperforms selective requantization; the higher bit rate is not compensated by an adequate increase in PSNR. When looking at both Figure 2.19 and Figure 2.20, it appears that although temporal compensation is indispensable for lower starting QPs, the resulting gain is not necessarily present for higher values of QP_1. This was also noted for lower-resolution content, and applies to the High profile case as well.

For the *Night* sequence, the results are notably different. For single-loop compensation, losses are between 1 and 1.5 dB. Selective requantization is able to restrain the drift, with PSNR losses of less than 0.5 dB for the highest rate points ($1 \leq \Delta QP \leq 3$), and less than 1 dB for the lowest rate points ($4 \leq \Delta QP \leq 6$).

The larger gaps in rate-distortion performance for the *Night* sequence can be attributed to the large intra-coded regions in the P pictures, which were not

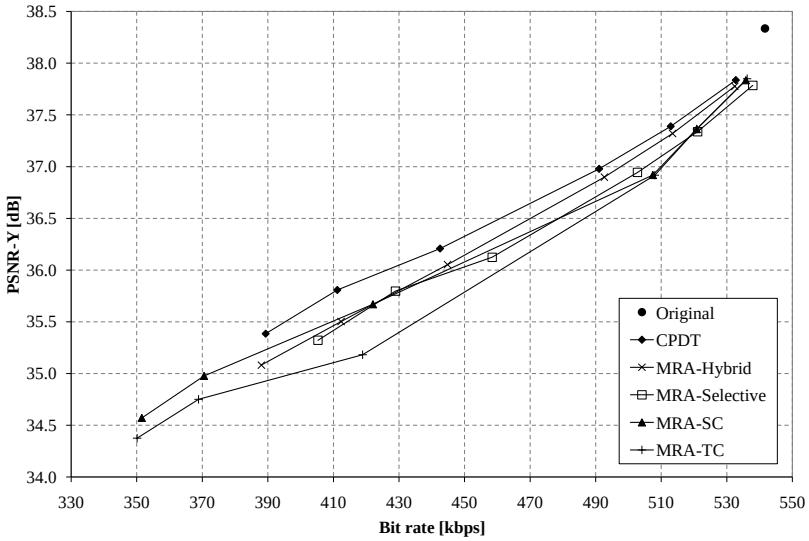


Figure 2.20: Rate-distortion results for *ShuttleStart* sequence ($QP_1 = 32$).

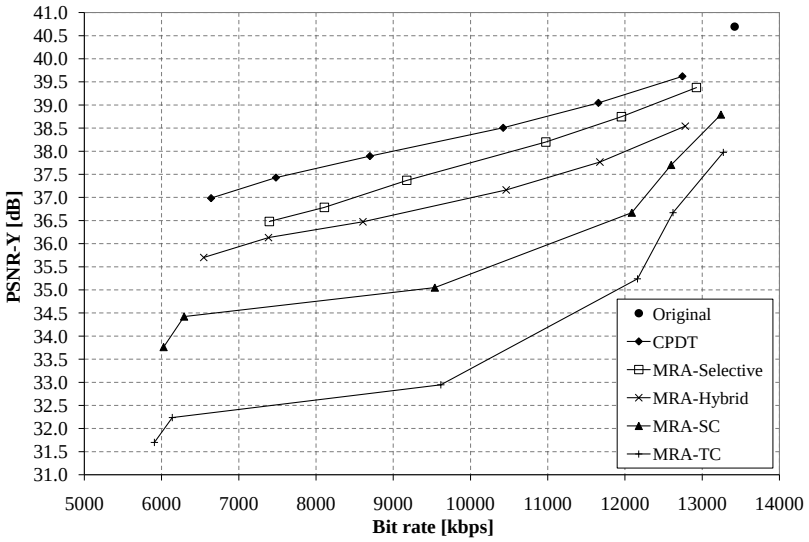


Figure 2.21: Rate-distortion results for *Night* sequence ($QP_1 = 22$).

present for the *ShuttleStart* sequence. This is illustrated in Figure 2.22, where the intra-coded macroblocks are highlighted.

For $QP_1 = 32$, the rate-distortion gaps are smaller. Similar to the *Shut-*

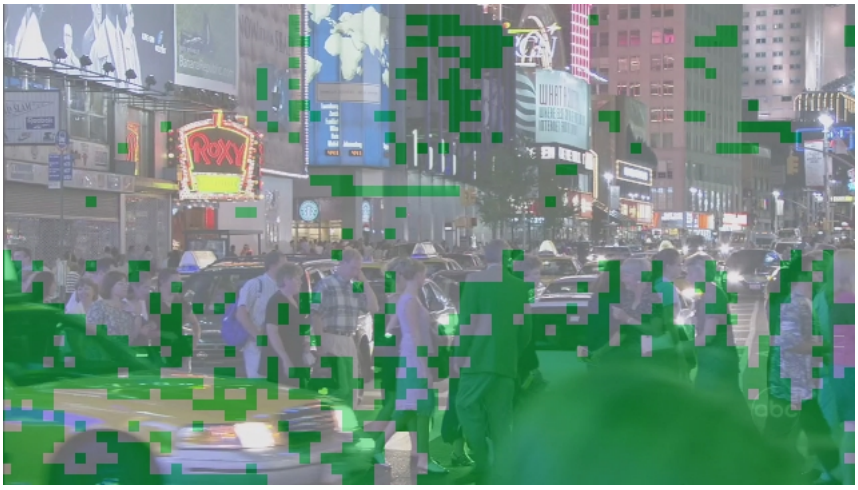


Figure 2.22: First P picture in *Night* sequence (picture nr. 3) with intra-coded macroblocks highlighted.

tleStart sequence, selective requantization leads to no gain.

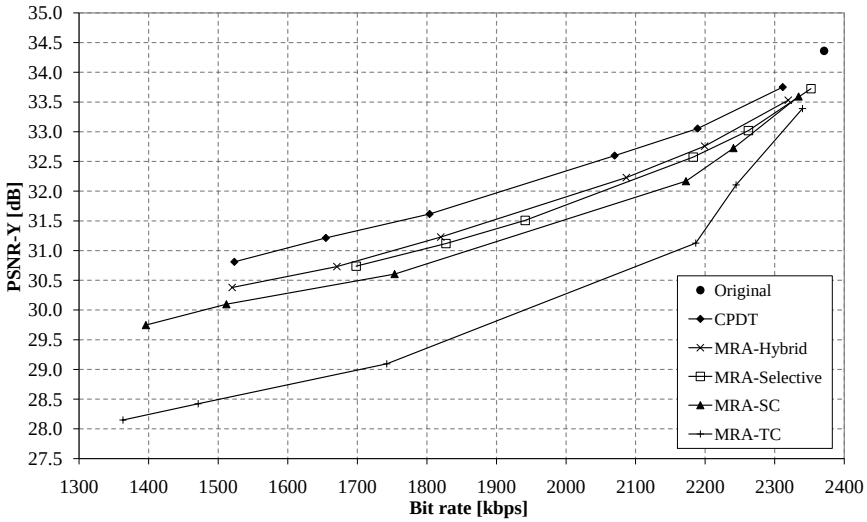


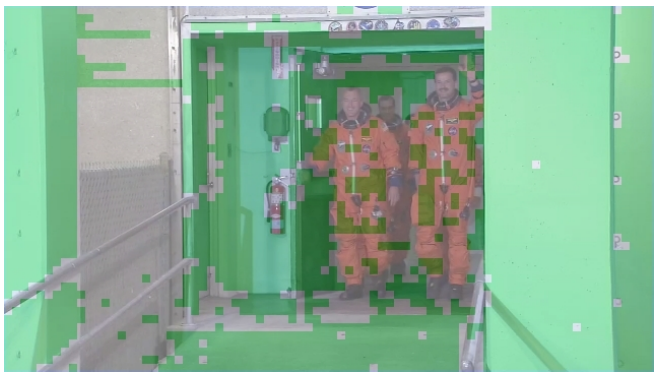
Figure 2.23: Rate-distortion results for *Night* sequence ($QP_1 = 32$).

The advantage of selective requantization especially becomes clear from the *Crew* sequence at high bit rates. Due to the illumination changes in successive pictures (cf. Figure 2.24(a) vs. Figure 2.24(b)), large intra-coded areas

are inserted, as can be seen in Figure 2.24(c).

(a) *Crew* sequence, frame 1 (B frame)

(b) *Crew* sequence, frame 2 (B frame)



(c) *Crew* sequence, frame 2 (B frame), with intra-coded macroblocks highlighted

Figure 2.24: Illumination changes in *Crew* sequence.

The effect becomes clear as demonstrated in Figure 2.25. The hybrid transrater results in a gap of 1 to 1.5 dB when compared to the CPDT transrater. Selective requantization is able to reduce the gap to 0.5 dB for the smallest bit rate reductions, with a widening gap towards 1 dB for the largest reductions.

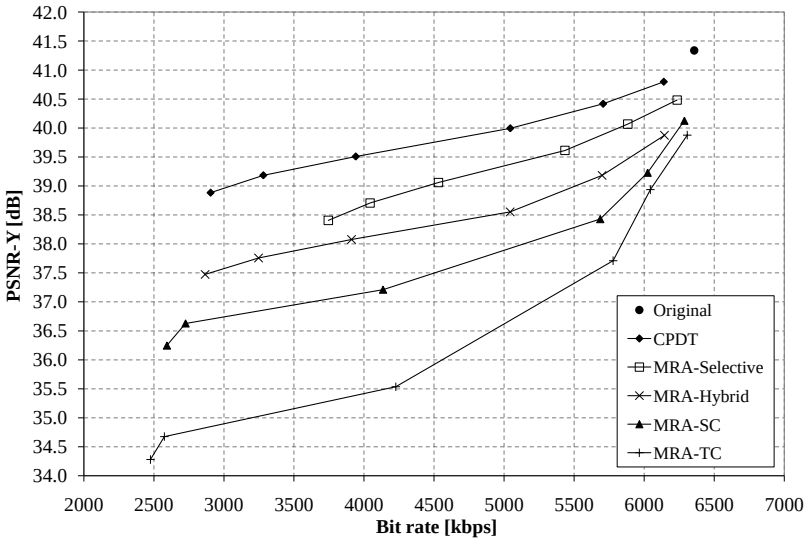


Figure 2.25: Rate-distortion results for *Crew* sequence ($QP_1 = 22$).

Similar results are obtained for the 1080p sequence *Crowdrun*, as shown in Fig. 2.26 and Fig. 2.27. For $QP_1 = 22$ and $\Delta QP = 6$, corresponding to a bit rate reduction of 45.4%, the MRA-Selective architecture results in a quality loss of approximately 0.75 dB. A loss of 0.3-0.5 dB can be noticed for the MRA-Selective and MRA-Hybrid architectures for $QP_1 = 32$.

B. Computational complexity analysis - timing results

For High profile, the timing results are shown in Table 2.9. As can be expected, the MRA-Selective architecture attains a slightly increased processing speed when compared to the MRA-Hybrid architecture, since only a high-level syntax change operation is required for the intra-coded macroblocks in P and B pictures. In the current implementation, both architectures are far from real-time, but nonetheless achieve a speed increase of about 60% when compared to the CPDT architecture, in addition of the benefit of highly reduced buffer requirements. The MRA-SC architecture, which is practicable in the lower bit rate range, or for sequences with low motion content, achieves a speed of more than 75% of open-loop transcoding, or a speed-up by more than 7 times over

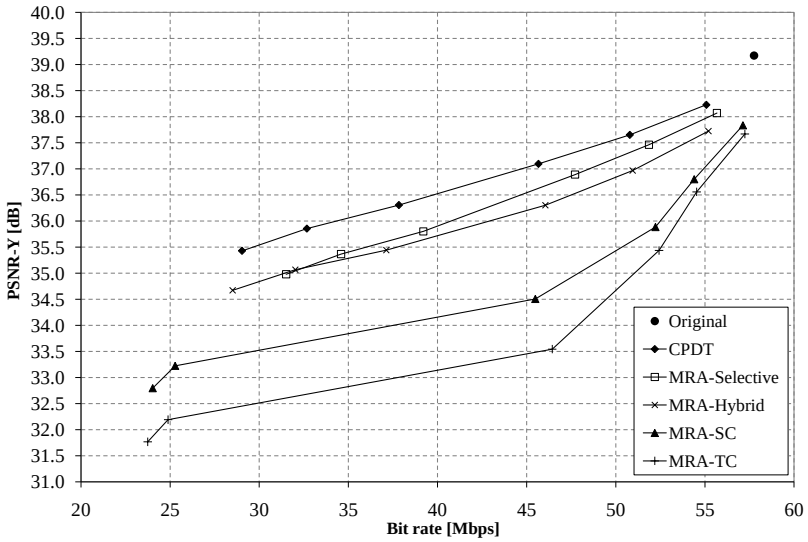


Figure 2.26: Rate-distortion results for *Crowdrun* sequence ($QP_1 = 22$).

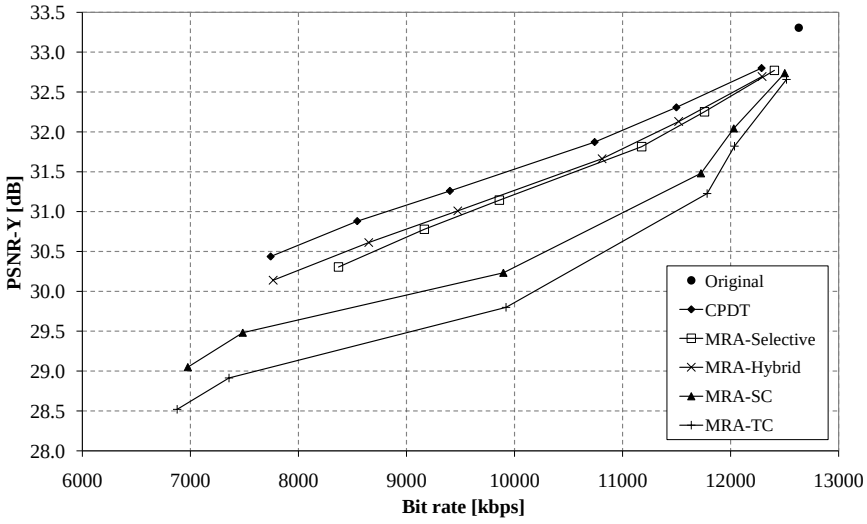


Figure 2.27: Rate-distortion results for *Crowdrun* sequence ($QP_1 = 32$).

the CPDT architecture. For the latter, buffer requirements are minimal, which is a huge benefit for transrating high-resolution sequences. Any architecture with temporal compensation would infer the presence of high-definition-size

buffers for every used reference picture.

Table 2.9: Processing speed for High profile transrating [fps]

	Crew (720p)	*Night* (720p)	*ShuttleStart* (720p)	*CrowdRun* (1080p)
CPDT	0.76	0.73	0.72	0.31
MRA-Selective	1.25	1.19	1.14	0.51
MRA-Hybrid	1.21	1.18	1.14	0.50
MRA-TC	1.21	1.18	1.14	0.50
MRA-SC	6.37	6.03	7.68	2.20
OL	8.38	7.69	9.90	2.69

2.5 Motion refinement

In the previous sections, transrating of residual data was examined, while all signalization data, including motion data (motion vectors, macroblock partitioning data, macroblock types, etc.) were passed unchanged to the output bitstream. Although the original motion information is optimized for the bit rate of the incoming bitstream (typically based on Lagrangian optimization techniques during motion estimation and mode decision at encoding time), this is not necessarily the case for the reduced bit rate of the output stream. When the quality gap between the input and output streams becomes larger, it is likely that rate-distortion efficiency in the output stream will benefit from an update in motion parameters.

Different algorithms have been examined for mode refinement in cascaded *pixel-domain* architectures. The possibility of refinement was discussed in [68]. A motion vector refinement search window of $[\pm 3, \pm 3]$ pixels was proposed, and a search window of $[\pm 1, \pm 1]$ pixels was mentioned to give satisfying results in most cases. In [69], Youn et al. proposed an adaptive motion refinement scheme. Firstly, a base motion vector was derived for a frame skipping transcoder (by temporally combining the input motion vectors), after which this motion vector can be refined with a delta motion vector. A fast algorithm was proposed to find the optimal or near-optimal delta motion vector. Because it is likely that the base motion vector will be located near the global optimum, the unimodal error surface assumption can reasonably be held in a small search window around this base motion vector, rendering fast search algorithms useful.

For H.264/AVC, CPDT-based techniques were proposed in [33, 34]. In the latter, motion reestimation complexity is reduced by evaluating only a limited set of modes during transcoding. In the proposed sets, larger block sizes were favored over smaller ones. This was based on the observation in [70] that sub-macroblock partitions contribute relatively less to rate-distortion performance; by only allowing blocks that are 8×8 and larger, more than 80% of the bit savings over 16×16 partitions can be captured. Also, their analysis stated that block sizes smaller than 8×8 tend to be useful only at relatively high bit rates. Hence, in [33], it was proposed to eliminate block sizes smaller than 8×8 from motion reevaluation.

In cascaded architectures with intermediate pixel-domain reconstruction, no constraints regarding motion refinement exist. No drift will occur, and full flexibility regarding the choice of output macroblock type or even frame type is possible. Most fast algorithms, however, base their output decisions on a refinement of the input parameters without completely reevaluating the search space. Here, we evaluate the motion refinement algorithm with bottom-up mode limitation for the cascaded pixel-domain architecture (CPDT).

Further, we examine if refining the motion vectors can be done in the *transform domain*. For the case of transform-domain solutions, the impact of the motion data change needs to be examined meticulously. A small change in motion vectors or partitioning information could lead to a severe mismatch and could very well result in drift. Proposals have been made in literature to approximate the distortion by using picture power spectrum. We show that this approximation has to be used carefully, since it does not take into account drift that is introduced in the stream. To accomplish this, the potential rate-distortion gain of tweaking motion information for the output stream is examined. Since a change in motion information induces a change in the motion-compensated prediction signal, a careful examination needs to be made of the change in both the rate and distortion.

We discuss the combination of a number of techniques and their practical applicability, and provide a comparison between pixel-domain and transform-domain approaches. Different aspects, such as Skip and Direct modes, have not been taken into account in existing approaches. These techniques are evaluated in a fully operational framework for bit rate reduction, with no simplifications or prior assumptions regarding the incoming bitstream, leading to a better understanding of the practical applicability of the techniques.

2.5.1 Motion-compensated prediction in H.264/AVC

A. Tree-structured motion-compensated prediction

H.264/AVC allows motion-compensated prediction with a granularity down to 4×4 pixels. Apart from 16×16 prediction, macroblocks can be divided in macroblock partitions of 16×8, 8×16, or 8×8 pixels. When the 8×8 macroblock type is chosen, a further subdivision in submacroblock partitions is allowed, where each submacroblock partition can have a size of 8×8, 8×4, 4×8, or 4×4 pixels. This is illustrated in Figure 2.28.

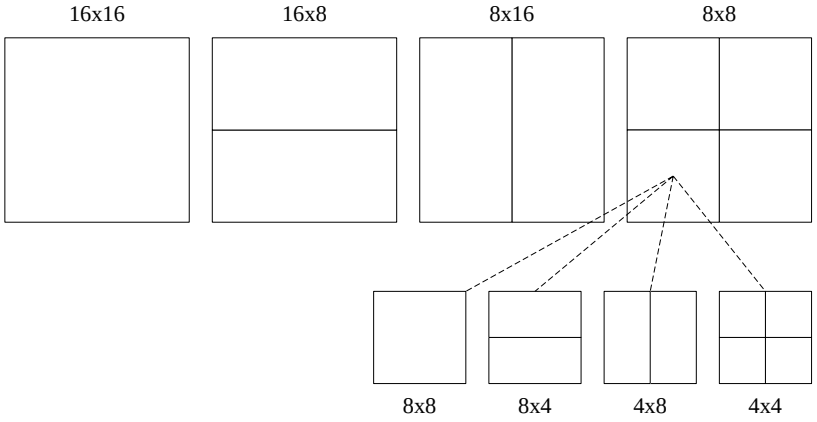


Figure 2.28: (Sub-)macroblock partitioning in H.264/AVC.

Up to 16 reference pictures can be stored and used for motion-compensated prediction. These reference pictures are specified using *reference indices*. Two reference picture lists are maintained, from hereon denoted as *list 0* and *list 1*. Typically, pictures from list 0 are used for forward prediction, while list 1 pictures are used for backward prediction. Additionally, bidirectional prediction can be applied, based on a (weighted) combination of pictures from both lists.

Reference indices have a granularity down to 8×8 pixels, i.e., each *macroblock partition* can have its own reference index, but all *submacroblock partitions* share the same reference index. The same holds for the prediction direction (forward, backward, or bidirectional prediction), i.e., H.264/AVC here provides a granularity down to 8×8 pixels.

Macroblock and submacroblock types in H.264/AVC are specified by the type of prediction used (predictive or bipredictive), the used prediction list, and the size of the partition, resulting in the (sub)macroblock types given in Table B.1 through Table B.4 in Appendix B.

B. Motion vector prediction

With the finer motion prediction granularity comes an increase in motion data. In order to avoid an inflation in motion vector data, motion vector prediction is used. Only the motion vector difference after prediction needs to be entropy coded in the bitstream. Motion vectors are typically estimated using a median predictor, where for each component (in x or y direction) of the vector the median value is taken from three surrounding motion vectors (if available).

This is illustrated in Figure 2.29. The partitions containing the pixel positions indicated by A, B, C, or D are used for motion vector prediction (when available). In general, the motion vector of partition X is predicted from surrounding partitions A, B, and C. When partition C is not available, the motion vector of partition D can be selected instead.

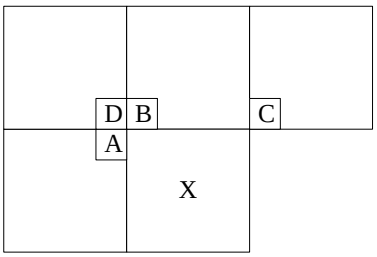


Figure 2.29: Motion vector prediction from neighboring partitions A, B, C, and/or D in H.264/AVC.

C. Skip and Direct macroblocks

Because of the considerably higher percentage of bits needed to code the motion information in H.264/AVC, efficient ways were sought to code motion vectors and reference indices. In particular for B pictures, with two motion vectors for bipredictive partitions, the benefit of efficient motion prediction becomes clear. Skip and Direct macroblocks increase rate-distortion performance by exploiting the spatiotemporal correlation between adjacent macroblocks.

In P pictures, for Skip macroblocks, no motion vector differences are sent in the bitstream, i.e., the used motion vector equals the (median) motion vector prediction. The use of a P Skip also indicates that no residual data is sent for the macroblock. Apart from the Skip mode, the temporal and spatial Direct modes are available in B pictures [71]. Originally, the B Direct mode only used temporal correlation by deriving its motion vector from the information

in the co-located block from the first *list 1* reference picture. For exploiting the spatial motion correlation, the spatial Direct mode was added. Firstly, the most appropriate reference picture is selected from the spatially adjacent blocks during the motion vector prediction process. Typically, motion vectors for the spatial Direct mode are derived by setting the motion vector to equal the motion vector prediction. However, performance was found to be considerably improved by partially taking into account the available temporal information. It was observed that for stationary regions, it is likely that co-located regions in adjacent pictures are also stationary [71]. In H.264/AVC, this is performed by firstly checking whether the co-located block in the first *list 1* reference picture is stationary, i.e., with motion vectors smaller than or equal to $(\pm 1, \pm 1)$ (in quarter-pixel units). The choice between temporal and spatial Direct mode can be made at a slice level, i.e., by appropriately setting the *direct spatial mv pred flag*. Both a *Direct_B_*16×16 macroblock type and a *Direct_B_*8×8 sub-macroblock type are available in H.264/AVC. The B Skip mode motion vectors and reference indices are calculated as for the B Direct mode, the difference being that no residual data is coded for the Skip mode, while for the Direct modes, residual data can still be present.

A sequence of skipped macroblocks can be efficiently represented in the bitstream, by coding a *macroblock skip run* syntax element for CAVLC entropy coding, or by inserting a *macroblock skip flag* for CABAC entropy coding.

2.5.2 Pixel-domain motion refinement

For CPDT transcoding, pixel-domain reconstructed pictures are available as a reference for motion-compensated prediction, and full flexibility is provided regarding possible refinement of motion parameters. A complete motion reestimation step could be performed, leading to a straightforward decoder-encoder cascade. Here, we investigate algorithms which result in a significant complexity reduction compared to the cascade, by reusing motion information from the input bitstream.

The CPDT architecture with motion refinement is shown in Figure 2.30. The intra prediction is omitted for simplicity. 'ME' indicates the motion estimation step, which receives the motion parameters of the incoming bitstream as input. Based on the derived optimal motion vector $\mathbf{v}'$ (which can differ from the input motion vector $\mathbf{v}$), motion-compensated prediction is performed at the encoder side.

During transcoding, the main goal is to minimize the complexity of the motion estimation and mode decision processes, in order to speed up the overall transcoding time. This can be done by benefiting from the information

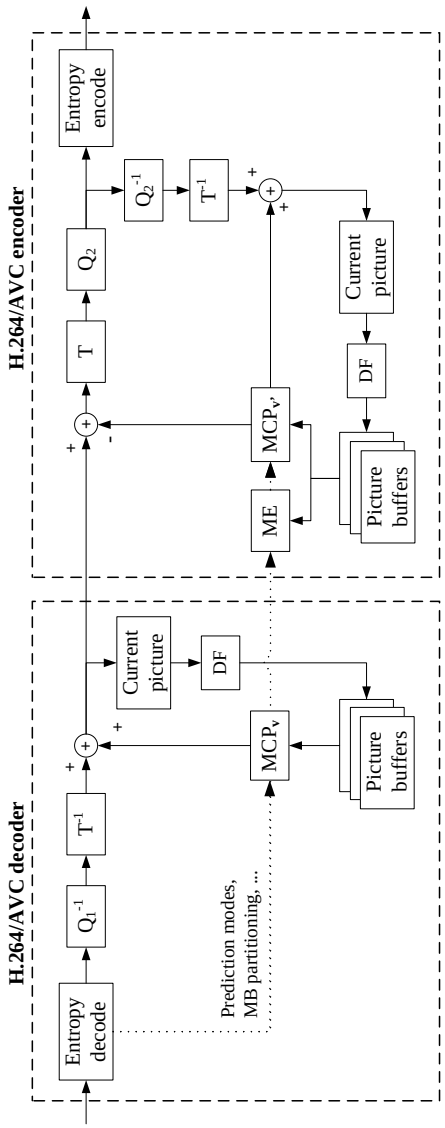


Figure 2.30: CPDT architecture with motion refinement.

from the incoming bitstream, which can be considered to be a good approximation of the optimal motion vector for the outgoing bitstream. Also, the observation that lower-rate streams benefit less from smaller partition sizes is advantageous. In lower-rate bitstreams, larger block sizes become more dominant, and the amount of submacroblock partitions tends to decrease. Hence, the most natural way of refining mode decisions for lower bit rates is by merging partitions, if the distortion introduced by the merging operation is small enough. During transcoding, the rate-distortion cost of merging macroblock partitions needs to be examined.

To derive the most appropriate output motion vector for the merged partition, the cost for each of the input motion vectors of the constituting partitions is evaluated (in rate-distortion sense, as explained further). This is illustrated in Figure 2.31 for the case of submacroblock partitions. If the initial macroblock partition consisted of four 4×4 submacroblock partitions and the case of 8×4 partitions is examined, the two constituting motion vectors are assessed for each partition, i.e., $\mathbf{v}_0$ vs. $\mathbf{v}_1$, and $\mathbf{v}_2$ vs. $\mathbf{v}_3$. In a similar way, the cost of 4×8 partitions is examined by evaluating $\mathbf{v}_0$ vs. $\mathbf{v}_2$, and $\mathbf{v}_1$ vs. $\mathbf{v}_3$. All four motion vectors are evaluated to obtain the cost of an 8×8 partition.

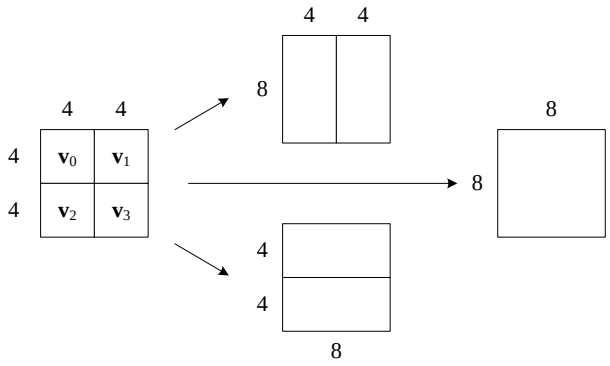


Figure 2.31: Submacroblock partition merging, starting from four 4×4 partitions.

Similarly, the merger of macroblock partitions is evaluated. In this case, the possibility exists that a different reference index is used for each partition, or that different prediction directions were used. After each possible merge operation, the rate-distortion cost is determined. The rate and distortion are obtained as described in the following subsections.

After merging, we evaluate the possibility of introducing Skip or Direct partitions. This process is described in closer detail in Section 2.5.4.

A. Rate calculation

Besides the residual data, a number of motion-related syntax elements contribute to the output data rate. The macroblock type and if necessary submacroblock types, reference picture indices, and motion vector differences need to be transmitted. If the macroblock is skipped, only a *macroblock skip run* (CAVLC entropy coding) or *macroblock skip flag* (CABAC) needs to be sent (one bit or less per skipped macroblock).

Conversion of submacroblock or macroblock types to a merged type leads to a further reduction of the length of the codeword, since the syntax elements for larger partition sizes are typically represented with fewer bits. Also, a removal of submacroblock partitions avoids the need to send submacroblock types.

The most time-consuming step during rate calculation is the determination of the motion vector predictor, in order to end up with the motion vector differences that need to be encoded in the bitstream. In most cases, the median motion vector predictor needs to be derived, which requires derivation of neighboring macroblock partitions. In case of B Direct or Skip macroblocks, complexity is even higher, and the co-located reference partition needs to be deduced.

B. Pixel-domain distortion calculation

Owing to the intermediate reconstruction in the CPDT transcoder, the distortion during motion refinement can easily be calculated for example using SAD or SSD. For a prediction block $\boldsymbol{S}_k$, the distortion of the displaced frame difference (DFD) associated with motion vector $\boldsymbol{v'} = (v'_x, v'_y)$ between the current picture s and the reconstructed reference picture s'_r (as indexed by the current reference index r) can be calculated as:

$$D_{DFD}(\boldsymbol{S}_k, \boldsymbol{v'}, r) = \sum_{(x,y)\in\boldsymbol{S}_k} \left| s[x,y] - s'_r[x - v'_x, y - v'_y] \right|^p \qquad (2.71)$$

with $p = 1$ for SAD calculation and $p = 2$ for SSD calculation. The size of the prediction block $\boldsymbol{S}_k$ is $M \times N$, where $M, N \in \{4, 8, 16\}$.

C. Rate-distortion optimized motion selection

Given the availability of pixel-domain reconstructed pictures in the transcoder, rate-constrained motion refinement can be performed as during motion estimation in the encoder by minimizing [72]:

$$\arg\min_{(\boldsymbol{v'},r)} \left\{ D_{\text{DFD}}(\boldsymbol{S}_k, \boldsymbol{v'}, r) + \lambda_{\text{MOTION}} \cdot R_{\text{MOTION}}(\boldsymbol{S}_k, \boldsymbol{v'}, r) \right\} . \qquad (2.72)$$

Here, R_{MOTION} represents the rate cost required for transmission of the motion vectors and reference indices. When using SSD during motion refinement, the Lagrangian parameter $\lambda_{\text{MOTION}} = \lambda_{\text{MODE}}$, where the latter is the Lagrangian parameter used during mode decision. When using SAD, $\lambda_{\text{MOTION}} = \sqrt{\lambda_{\text{MODE}}}$.

Mode decision is performed by minimizing

$$J_{\text{MODE}}(\boldsymbol{S}_k, \boldsymbol{I}_k) = \left\{ D_{\text{REC}}(\boldsymbol{S}_k, \boldsymbol{I}_k) + \lambda_{\text{MODE}} \cdot R_{\text{REC}}(\boldsymbol{S}_k, \boldsymbol{I}_k) \right\} \tag{2.73}$$

where $\boldsymbol{I}_k$ ranges over the available macroblock and submacroblock partition types. D_{REC} indicates the SSD between the original and reconstructed block pixels. The rate cost R_{REC} includes both the motion cost and the cost for coding the transform coefficients. The relation between λ_{MODE} and QP for use in H.264/AVC was empirically determined [72] as:

$$\lambda_{\text{MODE}} = 0.85 \cdot 2^{(QP-12)/3} \ . \tag{2.74}$$

2.5.3 Transform-domain motion refinement

We further investigated if motion refinement can be performed in the transform domain, so it can be applied for single-loop (or open-loop) transrating architectures. Transform-domain operations complicate motion refinement, due to the absence of a pixel-domain reference, which could result in additional drift in the bitstream. Transform-domain motion refinement has been studied for example in [73]. There, however, a number of simplifications and restrictions were used, such as the use of only one reference picture and constrained intra prediction. Only the case of hierarchical coding patterns was examined. Here, we enquire into the more general case of multiple reference pictures and unconstrained intra prediction.

The architecture for single-loop transcoding with motion refinement is shown in Figure 2.32. The functional blocks for intra prediction have been omitted for simplicity.

Working in the transform domain limits the possibilities for refinement due to the deficiency of pixel-domain reference pictures. Nonetheless, the motion parameter rate can be lowered by reducing the granularity of motion partitions. To accomplish this, we examine in successive steps if macroblock partitions can be merged together. If two merged (sub)macroblock partitions use the same motion vector and reference index, no loss is incurred during the merging operation. If the merged macroblock partitions contain different motion vectors (which is typically the case), however, a mismatch arises and the introduced distortion needs to be estimated.

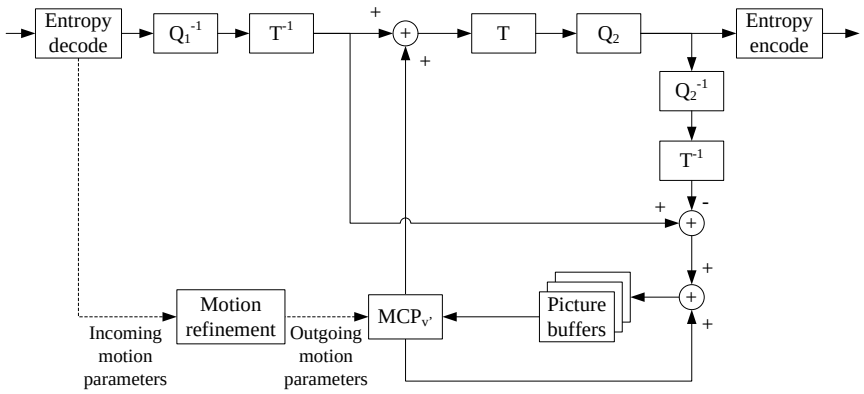


Figure 2.32: Single-loop architecture with motion refinement.

When merging *submacroblock* partitions, reference indices and prediction directions (forward, backward, or bidirectional prediction) do not have to be taken into consideration, since these are identical for all submacroblock partitions of a single 8×8 partition. When merging macroblock partitions (8×8 and larger), however, special care has to be taken to avoid merging partitions that contain motion vectors pointing to different reference pictures. Reference picture indices can have a granularity down to 8×8 pixels (i.e., all submacroblock partitions within a single 8×8 block will refer to the same reference picture). If macroblock partitions with different reference indices would be merged, serious artifacts would arise in the decoded video stream, in particular when the temporal distance between the two reference pictures increases.

This problem is aggravated in B pictures, where different prediction directions can be used for each macroblock partition, i.e., reference pictures can be selected from different lists (forward prediction list, backward prediction list, or both). When bidirectional prediction is used, the partition is predicted based on a weighted sum of prediction signals.

Since merging partitions with different reference indices or prediction directions would cause artifacts in the transcoded bitstream, this situation is avoided, and only merging partitions with identical reference indices and prediction direction is considered. This is illustrated in Figure 2.33 for an example where two bidirectionally predicted 8×8 partitions and two forward predicted 8×8 partitions are merged to a macroblock with 8×16 partitioning.

After each possible merge operation, the rate-distortion cost is evaluated. The rate and distortion are determined as described in the following paragraphs.

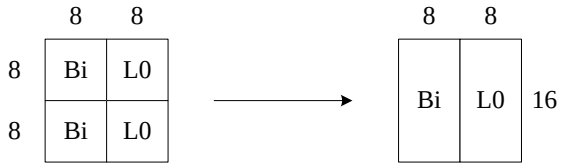


Figure 2.33: Macroblock partition merging.

Transform-domain distortion estimation

As shown in [73, 74], the distortion (D) introduced by motion vector variation can be estimated in the transform domain based on the picture power spectrum. The distortion, expressed as the SSD between the prediction signal p of the input motion vector and the prediction signal p' of the refined output motion vector, i.e., $D = \sum_{m=1}^{4} \sum_{n=1}^{4} (p[m,n] - p'[m,n])^2$ can hence be approximated as follows:

$$\begin{aligned} D &\approx \frac{1}{(2\pi)^2} \iint_{]-\pi,\pi]} S(\omega_1, \omega_2) \cdot (\boldsymbol{\omega} \Delta \mathbf{v})^2 \, d\boldsymbol{\omega} \\ &\approx \phi_x \Delta v_x^2 + \phi_y \Delta v_y^2 \end{aligned} \tag{2.75}$$

with $S(\omega_1, \omega_2)$ being the power spectral density of the prediction signal associated with the input motion vector $\mathbf{v}$. $\Delta \mathbf{v} = (\Delta v_x, \Delta v_y)$ expresses the difference between the input and candidate output motion vectors $\mathbf{v}$ and $\mathbf{v}'$. ϕ_x and ϕ_y are determined as follows:

$$\phi_x = \frac{1}{(2\pi)^2} \iint_{]-\pi,\pi]} S(\omega_1, \omega_2) \cdot \omega_1^2 \, d\omega_1 d\omega_2 \tag{2.76}$$

and

$$\phi_y = \frac{1}{(2\pi)^2} \iint_{]-\pi,\pi]} S(\omega_1, \omega_2) \cdot \omega_2^2 \, d\omega_1 d\omega_2 \, . \tag{2.77}$$

The power spectrum can be obtained by approximating the FFT using the 4×4 integer transform in H.264/AVC. Since the H.264/AVC transform does not include the normalization values, a quantization step is added (with $QP = 4$, corresponding to a $Q_{\text{step}} = 1$) to normalize the transform coefficients.

The above integrals for ϕ_x and ϕ_y are discretized by setting the frequencies ω_1 and ω_2 to $\pm \frac{k\pi}{N}$ with $k = 0, \ldots, 3$, and $N = 4$.

2.5.4 Skip and Direct mode detection

Particular care needs to be taken for Skip and Direct macroblocks, since they contribute to rate-distortion gains due to their efficiency, but also because they do not follow 'regular' motion vector prediction (in particular for B Direct and Skip macroblocks). This applies both to pixel-domain and transform-domain motion refinement.

After merging, Skip and Direct modes need to be reevaluated. For (macro)blocks which were previously encoded as Skip or Direct, surrounding motion vectors may have changed. This can result in a non-exact motion vector predictor (with motion vector differences which differ from zero) after motion refinement. In this case, a conversion to a non-Skip or non-Direct mode may be required.
In most cases, however, due to a coarser quantization of the motion field, a more accurate motion vector predictor will be obtained after refinement. In the output sequence, Direct and Skip modes will be selected more frequently, leading to a reduction of the bit rate. The coarser quantization of the residual coefficients will lead to a further reduction of the bit rate, due to the conversion of B_Direct macroblocks to B_Skip mode.

During motion refinement, we distinguish three steps leading to conversion and detection of Direct and Skip macroblocks.

- **Normalization** Firstly, Direct and Skip macroblocks from the incoming bitstream are converted to their 'normalized' equivalents. P_Skip macroblocks are converted to P_L0_16×16 macroblock types, B_Skip and B_Direct macroblocks are converted to B_8×8 partitions with forward, backward, or bidirectional prediction. In this step, relative motion vector difference values are converted to the absolute motion vector components.

- **Refinement & merging** By consecutively merging and refining partitions, larger partition sizes are obtained, as described in the previous sections.

- **Direct and Skip mode detection** After refinement, motion vector prediction is repeated, and the possibility of introducing Direct or Skip blocks is evaluated. In the case of pixel-domain refinement, no constraints exist, and these modes can be evaluated just as any other mode. For transform-domain refinement, Direct or Skip modes are only evaluated in case the reference index and prediction direction constraints are met. For P macroblocks, the reference index needs to equal zero, i.e., the first reference picture in prediction list 0 is used. For B partitions,

both the reference index and prediction direction requirements need to be fulfilled. This limitation will result in a significant advantage of pixel-domain refinement over the transform-domain approach.

2.5.5 Motion refinement: results and discussion

To test motion-refined transrating, we used sequences *Foreman*, *Stefan*, *Mobile & Calender*, and *Paris*. For initial tests, we used an IBBP GOP structure, with a GOP length of 16 pictures. Rate-distortion optimization was enabled, and CABAC was used for entropy coding. All modes were allowed during encoding (default setting), including Direct and Skip macroblocks. The different steps as described in 2.5.4 were followed, i.e., normalization of the original direct partitions, followed by (sub)macroblock partition merging and evaluation and detection of Direct and Skip partitions.

A. Pixel-domain transcoding

Pixel-domain motion refinement has the largest potential for rate-distortion improvement. Firstly, the absence of constraints on motion vector or reference index refinement can lead to a large reduction of the bit rate. Secondly, no drift will arise due to changes in motion parameters. The question remains if the gain in rate-distortion performance justifies the extra introduced computational complexity.

Figure 2.34 shows the results for the *Foreman* sequence with a starting $QP_1 = 22$, for values of $\Delta QP \in [1, 20]$. The curves show that motion refinement will result in growing gains when the ΔQP increases. This confirms the expectation that motion parameters will benefit from refinement when the gap in ΔQP increases. It also demonstrates that for small QP increases it is not advantageous to add a motion refinement step to the requantization process, since it can lead to R-D losses if not taken into account properly. This corresponds to the findings for H.263 in [69] that the performance obtained by using new motion vectors can be slightly worse than that obtained by using the incoming motion vectors when quantization levels are similar.

This effect is largely independent from QP_1 and the bit rate range. This is illustrated in Figure 2.35 and Figure 2.36, which show the results for the *Foreman* sequence for different starting QPs ($QP_1 \in \{22, 27, 32, 37\}$)[9] and a fixed value of $\Delta QP = 6$ and $\Delta QP = 12$, respectively. The obtained curves show that gains remain more or less constant over the bit rate range. For a $\Delta QP = 6$, minor gains are obtained (less than 0.2 dB). The benefit of motion

[9] As opposed to Figure 2.34, which starts from a fixed $QP = 22$.

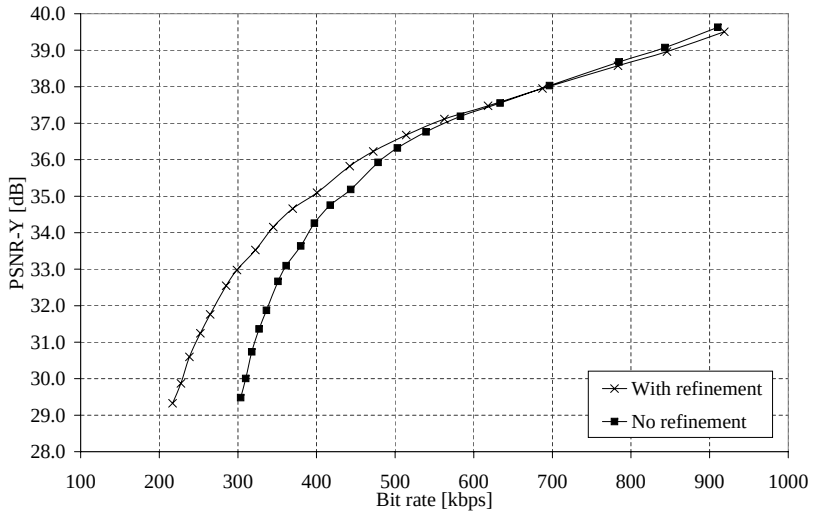


Figure 2.34: Rate-distortion results for pixel-domain motion refinement (*Foreman* sequence, $QP_1 = 22$, variable ΔQP).

refinement is more pronounced for $\Delta QP = 12$, with gains ranging from 0.5 to 0.7 dB. The highest rate points (with $QP_1 = 22$) for these two figures correspond to the $\Delta QP = 6$ and $\Delta QP = 12$ points in Figure 2.34.

For a large $\Delta QP = 18$, gains of more than 1.5 dB are obtained. It is clear that motion refinement becomes more important when the difference in quantization parameter increases.

The impact of motion refinement on the relative weight of motion and residual data in the bitstream is demonstrated in Figure 2.37 for the first two GOPs of the *Foreman* sequence, transcoded with a $\Delta QP = 6$ ($QP_1 = 22$). Although residual data is increased for most frames by a change of motion parameters, the overall rate drops due to the sharper reduction in motion data. Relative to the total bit rate, the share of motion data drops from 63% to 56% for B frames, and from 35% to 25% for P frames.

For the *Stefan* sequence, gains by using motion refinement are more limited when a starting $QP_1 = 22$ is used (Figure 2.38). For $QP_1 = 32$, the gains are more distinct, as can be seen in Figure 2.39. Note, however, that in this case the upper end of the QP spectrum is explored ($QP_2 \in [33, 51]$), resulting in very low quality output sequences.

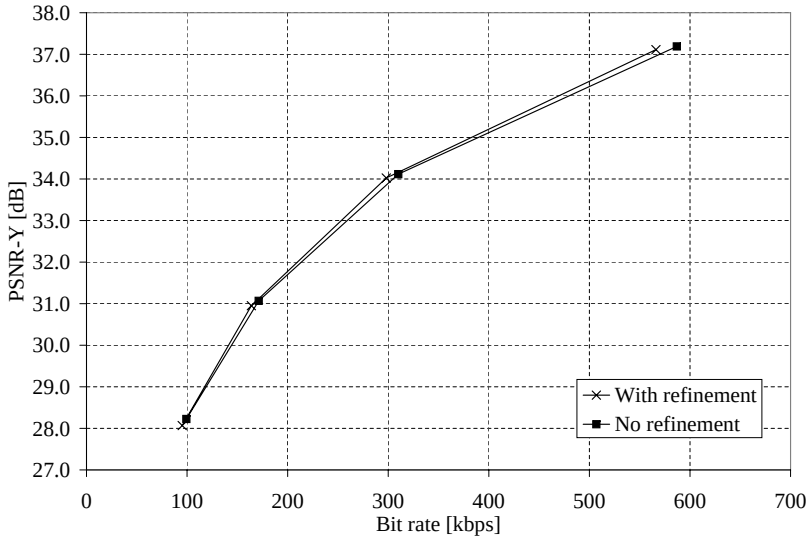


Figure 2.35: Rate-distortion results for pixel-domain motion refinement (*Foreman* sequence, $QP_1 \in \{22, 27, 32, 37\}$, $\Delta QP = 6$).

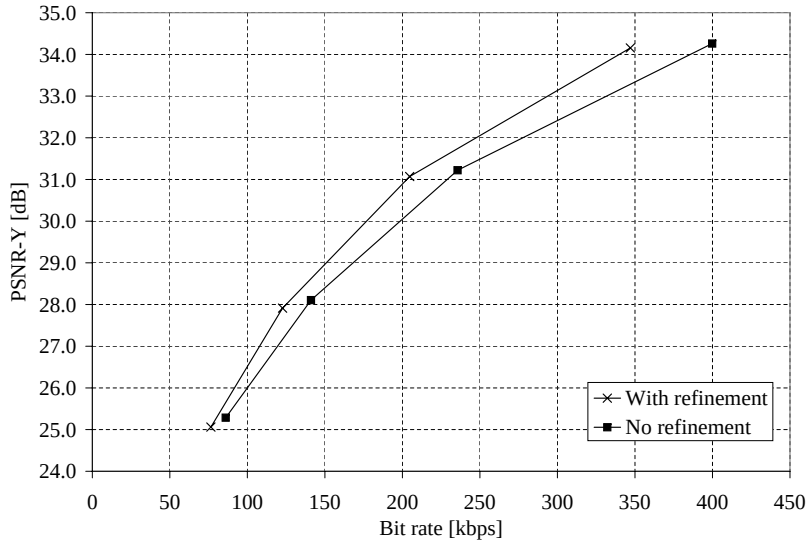
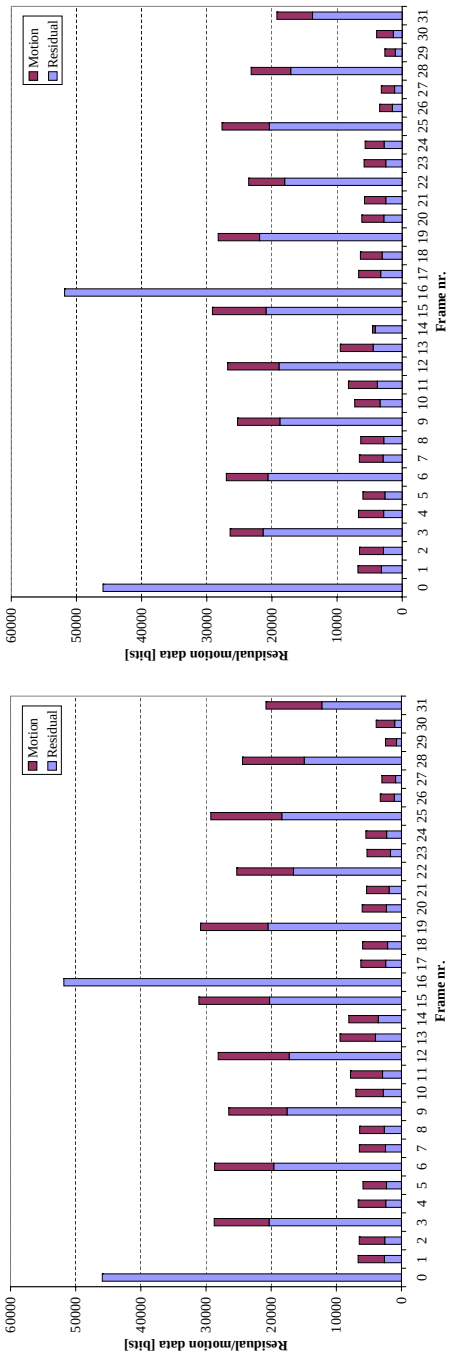


Figure 2.36: Rate-distortion results for pixel-domain motion refinement (*Foreman* sequence, $QP_1 \in \{22, 27, 32, 37\}$, $\Delta QP = 12$).



(a) No refinement

(b) With refinement

Figure 2.37: Impact of pixel-domain motion refinement on relative weight of residual and motion data (*Foreman*, $QP_1 = 22$, $\Delta QP = 6$).

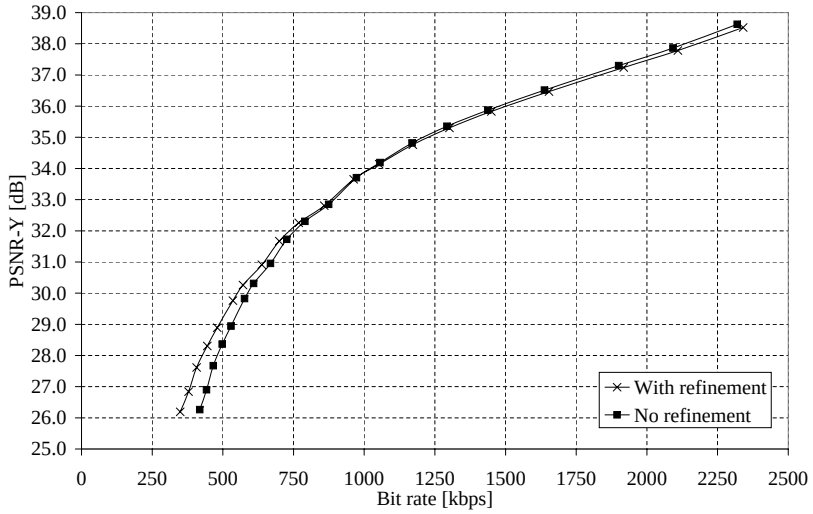


Figure 2.38: Rate-distortion results for pixel-domain motion refinement (*Stefan* sequence, $QP_1 = 22$, variable ΔQP).

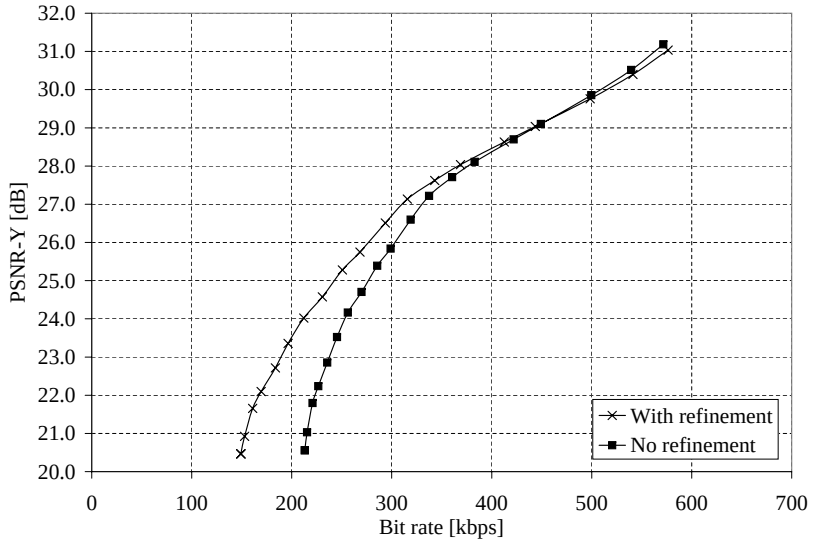


Figure 2.39: Rate-distortion results for pixel-domain motion refinement (*Stefan* sequence. $QP_1 = 32$, variable ΔQP).

B. Transform-domain transcoding

Gains for transform-domain motion refinement will be more limited than for pixel-domain refinement for a number of reasons. Firstly, the refinement pos-

sibilities are restrained by the choice of prediction direction and reference picture in the incoming bitstream. Secondly, the power spectrum estimation technique only holds in the close neighborhood of the original motion vector. Hence, motion vectors can only be adapted in a small search window.

Results are shown for the *Stefan* sequence in Figure 2.40. The curve shows that transform-domain refinement causes little or no gain for small or moderate reductions in bit rate. Only for a $\Delta QP \geq 10$ the gain becomes visible. Gains exceed 0.5 dB for values of $\Delta QP \geq 14$.

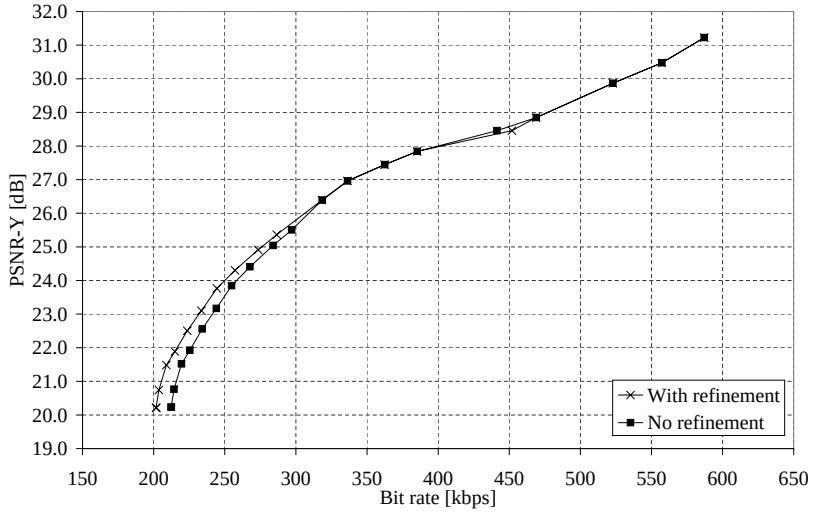


Figure 2.40: Rate-distortion results for transform-domain motion refinement (*Stefan* sequence, $QP_1 = 32$, variable ΔQP).

For a starting $QP_1 = 22$, gains are even more limited, as show in Figure 2.41. Only for $\Delta QP \geq 15$, gains becomes positive.

2.5.6 Computational complexity analysis - timing results

We evaluated the processing speed of transrating with pixel-domain and transform-domain refinement on different sequences. The results are shown in Table 2.10 for different CIF resolution sequences.

Pixel-domain refinement is shown to have a determining impact on the CPDT architecture. Although the search space is severely restrained by intelligently reusing incoming motion information, computational complexity suffers for a number of reasons. Firstly, repetitive evaluation of the displacement difference for all considered motion vectors and partition sizes during merging induces the burden of manyfold interpolation and memory access opera-

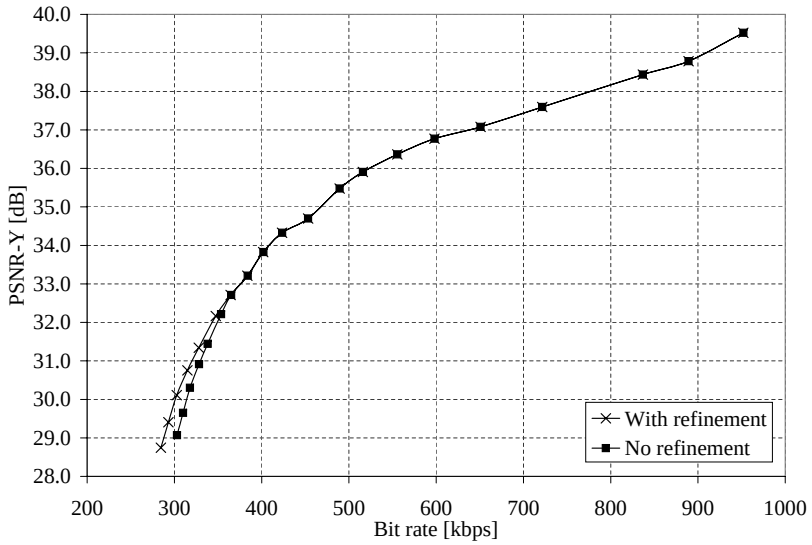


Figure 2.41: Rate-distortion results for transform-domain motion refinement (*Foreman* sequence, $QP_1 = 22$, variable ΔQP).

tions. Secondly, rate-distortion optimized evaluation requires the computation of motion vector differences, which implies the calculation of motion vector predictors and determination of neighboring macroblock partitions. Given the complexity of these algorithms in H.264/AVC, transrating speed is seriously compromised. Especially, Skip and Direct macroblock evaluation represent a computational challenge. For improved rate-distortion performance, however, this evaluation cannot be neglected.

Table 2.10: Processing speed for transrating without and with motion refinement [fps]

	Foreman	*Stefan*	*Mobile*	*Paris*
CPDT	6.84	6.40	6.05	6.15
MRA-Hybrid	11.44	10.65	10.04	10.22
CPDT-Ref	1.94	1.76	1.67	1.51
MRA-Hybrid-Ref	9.71	9.14	8.63	8.84

Using the distortion estimation based on picture power spectrum has a lesser influence on processing speed. This technique, however, can only be used in a limited search window around the incoming motion vector, and does not allow to evaluate changes in reference indices or prediction direction. The

technique is primarily useful for transform-domain operation, and can improve rate-distortion performance for large reductions of the bit rate. The impact on processing speed is limited to 15% of the MRA-Hybrid architecture without motion refinement.

2.6 Conclusions and original contributions

When compared to previous video coding standards, such as MPEG-1/2, a number of changes are required in order to make requantization transcoding practically usable. Spatial prediction introduces a new challenge in H.264/AVC and causes significant drift and visual artifacts when not taken care of properly. Although publications reported single-loop solutions not to be practically viable, we showed that a hybrid single-loop architecture with spatial and temporal compensation results in rate-distortion performance close to that of cascaded pixel-domain transrating, and even more importantly, the removal of disturbing visual artifacts in the transcoded pictures. Especially in the lower bit rate range, losses are constrained to less than 0.5 dB for typical sequences. When compared to previous solutions, such as in [7], this is made possible due to the addition of spatial compensation. The fast architecture using only spatial compensation (MRA-SC) results in highly improved rate-distortion curves over traditional single-loop architectures, at a processing speed of nearly 80% of the open-loop transcoder. Spatial compensation enables the possibility of a new class of low-complexity transcoders that are able to restrain the quality loss due to requantization.

We adjusted the introduced techniques for use with H.264/AVC High profile and investigated the applicability of single-loop transcoding techniques for high-resolution sequences. The results show that spatial drift becomes an obstacle when large intra-coded areas are present in the incoming bitstreams, such as for the *Night* and *Crew* sequences. The technique of selective requantization, which avoids requantization of intra-coded macroblocks altogether, results in slightly higher bit rates, but in gains of 0.5 to 1 dB for higher bit rates.

Furthermore, the impact of motion parameter refinement during transrating was investigated. As demonstrated, the adjustment of motion vectors becomes beneficial for moderate to large bit rate reductions. In particular, the use of pixel-domain refinement results in notable gains when the QP gap increases. Results show that transform-domain motion refinement can somewhat improve rate-distortion performance, but the increase is more limited than what has been previously reported in literature. Firstly, the theoretical gain that can be achieved is more limited than for pixel-domain refinement, given the restric-

tions during the merging phase and during Skip or Direct macroblock detection. If these restrictions are not obeyed, serious misprediction and artifacts could occur in the output bitstream. Secondly, the effect of drift should not be underestimated. Refinement of partitions will lead to a different MCP or intra prediction signal, with an error which can propagate spatially or temporally.

The author's work on requantization transcoding led to the following publications:

- Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Requantization transcoding for H.264/AVC video coding. Submitted to *Elsevier journal on Signal Processing: Image Communication*.

- Stijn Notebaert, Jan De Cock, Samie Beheydt, Jan De Lameillieure, and Rik Van de Walle. Mixed architectures for H.264/AVC digital video transrating. *Springer journal on Multimedia Tools and Applications*. August 2009.

- Jan De Cock, Stijn Notebaert, Peter Lambert, Davy De Schrijver, and Rik Van de Walle, Requantization transcoding in pixel and frequency domain for intra 16x16 in H.264/AVC. In *Lecture Notes in Computer Science*. August 2006.

- Jan De Cock, Stijn Notebaert, and Rik Van de Walle. Combined SNR and temporal scalability for H.264/AVC using requantization transcoding and hierarchical B pictures. In *Proceedings of IEEE International Conference on Multimedia & Expo (ICME)*. July 2007.

- Stijn Notebaert, Jan De Cock, and Rik Van de Walle. Improved H.264/AVC requantization transcoding using low-complexity interpolation filters for 1/4-Pixel motion compensation. In *Proceedings of the 2007 IEEE Symposium Series on Computational Intelligence*. April 2007.

- Jan De Cock, Stijn Notebaert, and Rik Van de Walle. A novel hybrid requantization transcoding scheme for H.264/AVC. In *Proceedings of the International Symposium on Signal Processing and its Applications (ISSPA)*. February 2007.

- Stijn Notebaert, Jan De Cock, Koen De Wolf, and Rik Van de Walle. Requantization transcoding of H.264/AVC bitstreams for intra 4x4 prediction modes. In *Lecture Notes in Computer Science*. November 2006.

- Jan De Cock, Stijn Notebaert, Koen De Wolf, Peter Lambert, and Rik Van de Walle. Low-complexity SNR transcoding for H.264/AVC. In *Proceedings of the Fourth IASTED International Conference on Communications, Internet and Information Technology*. November 2006.

Chapter 3

Heterogeneous transcoding from H.264/AVC to SVC

3.1 Rationale and related work

Recently, joint efforts of MPEG and VCEG have led to the standardization of a new state-of-the-art scalable video codec [1]. This scalable extension of H.264/AVC (published as Annex G of the H.264/AVC specification), denoted as SVC, makes it possible to encode scalable video bitstreams containing several quality, spatial, and temporal layers. By parsing and extracting, lower layers can easily be obtained, hence providing different types of scalability in a flexible manner.

The SVC design requires scalability to be provided at the encoder side by exploiting inter-layer dependencies during encoding. This implies that existing H.264/AVC content cannot benefit from the scalability tools in SVC due to the lack of intrinsic scalability provided in the bitstream at encoding time. Since a lot of technical and financial effort is currently being spent on the migration from MPEG-2 equipment to H.264/AVC, it is unlikely that a new migration to SVC will occur in the short term. Also, given the relatively high computational complexity of the SVC encoding process, it is likely that the dominance of single-layer encoders will continue to exist in the near future [75]. Since it is beneficial for broadcasters and content distributors to have scalable bitstreams at their disposal, efficient techniques for migration of single-layer content to a scalable format are desirable. In any case, the high cost of decoding and reencoding needs to be avoided.

Due to its computational efficiency, transcoding can be used for introducing scalability in compressed, single-layer bitstreams. In this way, reencoding can be avoided when migrating legacy content to a scalable format.

A number of techniques have been proposed in the past for introducing scalability in compressed bitstreams. The technique of generating multiple layers starting from a single-layer bitstream was studied in the context of data partitioning in [75, 76]. There, for MPEG-2, a data partitioning scheme was proposed based on the determination of the (causally) optimal breakpoint in DCT coefficient blocks. In [77], the problem of transcoding MPEG(-2/4) streams to (MPEG-4) Fine-Grained Scalability (FGS) was studied. A closed-loop solution, based on a simplified cascade of decoder and encoder, was found to be useful in the context of elastic storage of compressed video content. In [78], a technique was studied for transcoding of hierarchically coded H.264/AVC streams to streams with multiple FGS quality layers. Hierarchically coded B pictures were used to achieve combined temporal and quality scalability and to obtain improved rate-distortion performance.

Although an initial study on the topic was published in [78], based on an early version of SVC, no thorough investigation had been made of the subject. Several important elements in the SVC design, such as the different fidelity scalability techniques and the inter-layer prediction mechanisms, require a closer investigation.

In this chapter, novel fast architectures for H.264/AVC-to-SVC transcoding are designed which result in multiple-layer streams compliant with the final SVC specification [1]. Different possible architectures are examined, and the issues related to single-loop decoding and intra-coded macroblocks are explored in detail. These problems introduce additional challenges when compared to previously existing techniques, such as for MPEG-2 and MPEG-4 FGS. We propose techniques for each macroblock type and design overall architectures by combining the individual techniques.

The normative bitstream rewriting process [79] was added to the SVC specification and allows converting an SVC bitstream with multiple CGS layers into a single-layer H.264/AVC stream, i.e., the inverse of the proposed transcoding operation. We show that the changes that were introduced to support this functionality result in simplifications in the design of H.264/AVC-to-SVC transcoding architectures, and allow the introduction of different new architectures. Particularly, the changes to the reconstruction process for intra-coded macroblocks result in a larger degree of freedom for these macroblocks.

The first sections of this chapter deal with redistribution of residual data among the different layers. In Section 3.7, the possibility of adding motion data redistribution to the rewriting process is investigated. A multi-layer transcoder control model is introduced which allows a trade-off between rate-distortion performance in the different quality layers.

3.2 Scalable video coding

For a comprehensive overview of the scalable extension of H.264/AVC, we refer the reader to [49]. In this section, the provisions in SVC for quality scalability are briefly discussed. For improved rate-distortion performance over simulcast solutions, inter-layer prediction mechanisms were provided in the SVC design. These are discussed in Section 3.2.2. Inter-layer prediction has a major impact on compression performance, and plays an important role in the development of efficient architectures for H.264/AVC-to-SVC transcoding.

3.2.1 Quality scalability techniques

Different techniques for quality scalability (also denoted as fidelity or SNR scalability) have been investigated for SVC during its development. Although no designated technique for fine-grain quality scalability (FGS) has been included in the final design, two techniques remain that provide flexible adaptation, namely, coarse-grain scalability (CGS) and medium-grain scalability (MGS). Here, a brief overview of the available techniques is given. More information about the SNR scalability tools in the SVC specification can be found in [49].

A. Coarse-grain scalability

The CGS design uses techniques based on *dependency layers*. In every dependency layer, the same basic concepts for motion-compensated prediction and intra prediction are used as for single-layer coding. Apart from these single-layer coding techniques, additional inter-layer prediction mechanisms are provided. During *encoding*, a closed loop is formed for every dependency layer[1]. This is illustrated in Figure 3.1, where dependency layer 0 indicates the H.264/AVC-compliant base layer. By using inter-layer prediction mechanisms, enhancement dependency layers can contain quality refinements of transform coefficients in lower layers by using a decreasing quantization step size. Dependency layers are also used for spatial scalability [80]. The major difference is that for CGS no upsampling is required between successive dependency layers.

B. Medium-grain scalability

MGS was later added to the design [81, 82]. MGS uses techniques similar to CGS, but provides more flexibility. MGS allows the use of up to 16 *qual-*

[1]This contrasts to the single-loop *decoding* concept, which will be explained further on.

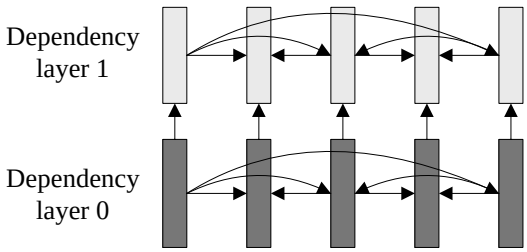


Figure 3.1: Dependencies for CGS quality scalability based on dependency layers.

ity levels per dependency layer, hereby significantly increasing the number of achievable rate extraction points. Also, the MGS quality levels can be removed at any point in the bitstream, while switching between CGS dependency layers is only possible at pre-defined points in the bitstream.

For MGS, no closed MCP loop is provided for every quality level during encoding. During MCP, except for so-called *key pictures*, only the full-quality reference pictures are used for prediction. This avoids that lower-quality versions need to be stored during encoding. As a result, drift will arise during decoding when MGS levels are dropped from the bitstreams. To prevent drift from spreading across the sequence, a key picture concept is used. While pictures inside of the GOP rely on the full-quality enhancement layer representation (which might not be available at the decoder side), the base layer of key pictures rely on the base quality reference pictures for MCP. In this way, the drift remains confined to the GOP, as illustrated in Figure 3.2.

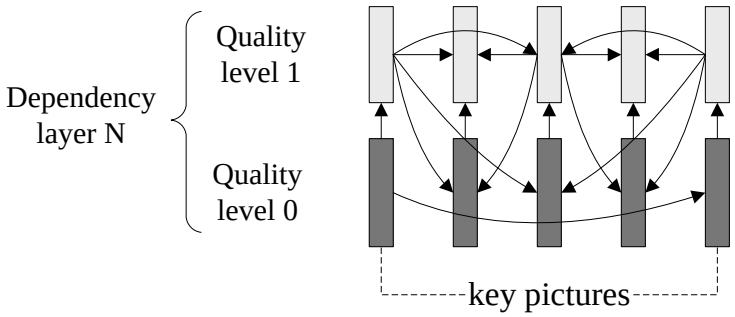


Figure 3.2: Dependencies and drift control for MGS quality scalability based on quality levels and key pictures.

C. Fine-Grain Scalability

During the standardization process, a complicated FGS design was long considered, based on progressive refinement (PR) slices. Due to the complexity of the design and the large syntax overhead, the PR design was eventually removed from the SVC specification. To achieve similar functionality, however, MGS can be used, allowing multiple quality extraction points. As an additional technique, two slice header syntax elements allow for a progressive transmission of residual coefficients in the bitstream (*scan idx start* and *scan idx end*). These syntax elements indicate for the blocks of residual coefficients the start and end position of the coefficients which are actually transmitted in the current quality level. In this way, the lowest-frequency coefficients can be grouped together in a slice, while higher-frequency coefficients can be sent in refinement quality levels.

3.2.2 Inter-layer prediction

In the SVC design, three inter-layer prediction mechanisms were introduced. These mechanisms extend the range of available prediction techniques, next to MCP and intra prediction. Inter-layer prediction results in improved coding efficiency of the SVC design over simulcast solutions. When compared to the scalability provisions in previous video coding standards, such as MPEG-2, H.263, and MPEG-4 Visual, significantly different inter-layer prediction mechanisms were designed. In previous standards, the inter-layer prediction methods are based on reconstructed samples of the lower layer signal. This is comparable to the inter-layer *intra* prediction technique in SVC. For the other inter-layer prediction mechanisms, lower-layer decisions and residual values propagate from the base layer to the reconstruction layer, and reconstruction is only performed in the reconstruction layer.

The design of the SVC encoder (for the case of CGS quality scalability) is displayed in Figure 3.3, including the three inter-layer prediction mechanisms.

A. Inter-layer residual prediction

In inter-layer residual prediction, the residual coefficients that are coded in lower layers can be used to predict the coefficients of higher layers, as indicated by the *residual prediction flag*. When using spatial scalability, the corresponding residual data in lower layers is upsampled using a bilinear filter [80], and used as prediction for the residual signal of the current macroblock.

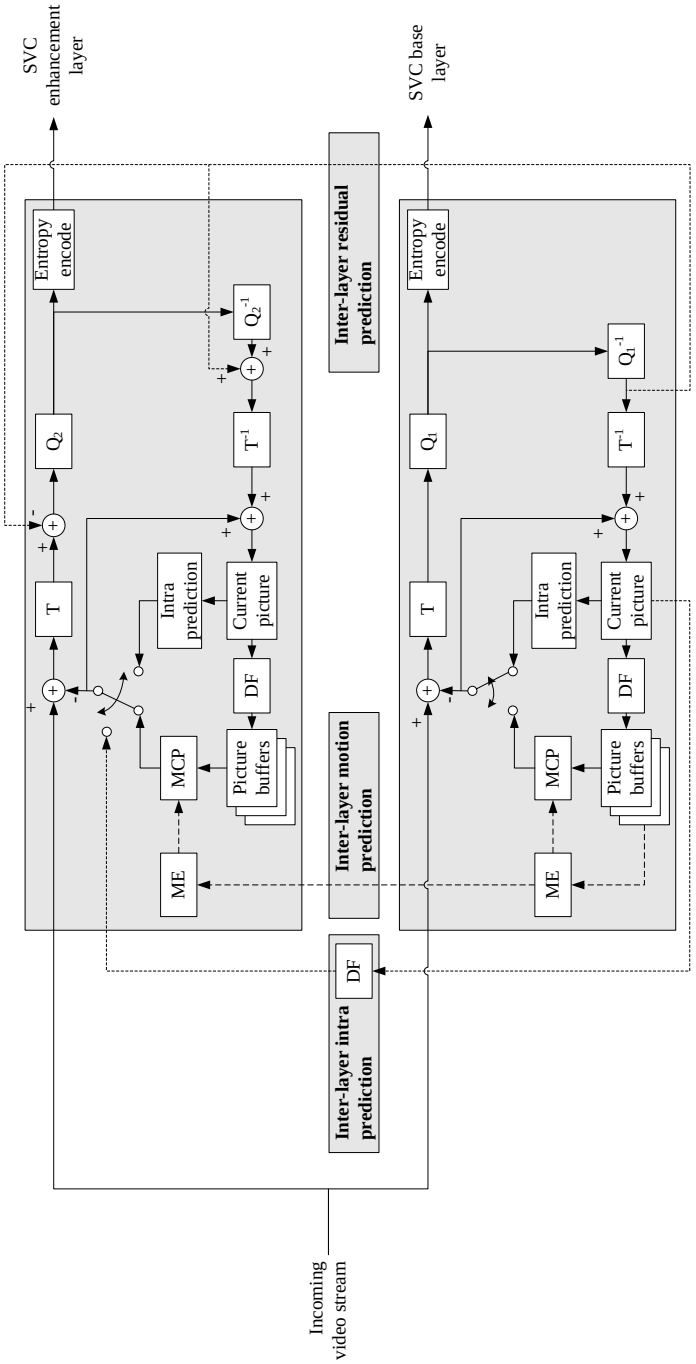


Figure 3.3: Design of CGS encoder displaying the inter-layer prediction mechanisms.

B. Inter-layer motion prediction

Each additional enhancement layer can reuse the motion information of underlying layers (reference picture indices and motion vectors). This can be represented in the enhancement layer syntax by setting the *base mode flag* to 1. The *base mode flag* specifies whether or not the mode and motion decisions of the reference layer can be reused[2]. When motion information from the reference layer is not exactly reused, it is also possible to form a motion vector predictor based on the MVs of the corresponding macroblock in the lower layers, as indicated by the *motion prediction flag*. In spatial scalability, the underlying MVs are scaled to form the predictor. In CGS, the motion vectors can be reused or refined without scaling.

C. Inter-layer intra prediction

For a given macroblock in the enhancement layer, if the co-located macroblock in the reference layer is intra-coded, the prediction can be formed by using inter-layer intra prediction (this is referred to as the I_BL macroblock type). When the co-located macroblock in the reference layer is intra-coded, and the I_BL macroblock type is selected, this can be signaled in the bitstream by setting the *base mode flag* to 1. In spatial scalability, a 4-tap filter is used to upscale the reconstructed pixels in the reference layer to form the prediction for the current layer. Since the case of quality scalability is examined, no upscaling is required, and the reconstructed lower layer intra-coded macroblock is used for prediction (after deblocking if required).

3.2.3 Constrained inter-layer prediction

A. Constrained intra prediction in H.264/AVC

In H.264/AVC, intra-coded macroblocks use the reconstructed pixels from neighboring (macro)blocks to form the prediction signal of the current (macro)block. Since different macroblock types can be used in P and B pictures, this means that prediction of intra-coded macroblocks can be based on surrounding (reconstructed) inter-coded macroblocks. An MCP loop is hence required to obtain the prediction pixels in this case. However, in the bitstream one may choose to disable intra prediction from inter-coded macroblocks by setting the *constrained intra prediction flag* syntax element in the picture pa-

[2] A special case exists for intra-coded macroblocks, resulting in macroblocks coded using inter-layer intra prediction.

rameter set. This option was provided to stop error propagation in environments with transmission errors that propagate due to MCP [32].

B. Single-loop decoding for SVC

In SVC, high decoder complexity is avoided by only requiring one MCP loop at the decoder[3]. This concept is known as single-loop decoding. During decoding, motion information and residual values are propagated from the lower layers to the target layer, and reconstruction is only performed for this target layer. The inter-layer intra prediction technique, however, conflicts with the single-loop decoding requirement. Since inter-layer intra prediction is based on reconstructed pixels from lower layers, dependencies might be introduced on surrounding inter-coded macroblocks in lower layers, which in turn imply an MCP decoding loop in these layers. As a result, the single-loop decoding rule would not be obeyed. In the SVC design, this is solved by restricting the use of inter-layer intra prediction. By imposing the constrained intra prediction requirement in lower layers, no dependencies are introduced in these layers, and the single-loop decoding concept is followed. More information regarding the single-loop decoding concept can be found in [83, 84].

During H.264/AVC-to-SVC transcoding, special attention has to be paid in order to maintain the single-loop decoding requirement. Tailored techniques are proposed that avoid a full reconstruction loop during transcoding.

3.3 H.264/AVC-to-SVC transcoding

3.3.1 Full decoder-encoder cascade (reencoding)

As in Chapter 2, the most straightforward, yet computationally most complex, transcoding solution is the cascade of decoder and encoder. Here, the output of an H.264/AVC decoder is fed to the SVC encoder (see Figure 3.3). Within the SVC encoder, MCP and intra prediction are performed for every dependency layer.

For intra-coded macroblocks, the encoder can choose to use inter-layer intra prediction as a better alternative to intra prediction in the enhancement layers. If inter-layer intra prediction results in an improved prediction signal, the I_BL macroblock type is selected.
For inter-coded macroblocks, the residual data after MCP can be further predicted by using inter-layer residual prediction. Deblocking filters (DF) are typically applied before both MCP and inter-layer intra prediction.

[3] An intra prediction loop is allowed in every layer, however.

The high computational complexity of this architecture is to a large extent determined by the presence of multiple MCP loops. This also implies that buffer requirements are high for the cascaded decoder-encoder solution, since multiple reference pictures can be used for MCP in every layer. Because of these reasons, it is beneficial for most applications to examine lower-complexity architectures.
In the remainder of this chapter, novel architectures with significantly lower complexity than the decoder-encoder cascade are presented. In particular, we aim at eliminating the MCP loops in the designed architectures. Transform-domain and partial transform-domain solutions avoid the need of pixel-domain reconstruction, and will be the main subject of the discussion. A second goal is to maximize rate-distortion performance, by exploiting the inter-layer prediction mechanisms.

3.3.2 Architectures for fast H.264/AVC-to-SVC transcoding

Given the difference in inter-layer prediction tools provided in the SVC design for each macroblock type, we make a distinction between inter-coded and (constrained or non-constrained, as will be explained further) intra-coded macroblocks. Techniques for each of these types are discussed. Typical H.264/AVC streams consist of a combination of intra-coded (available in I, P, and B pictures) and inter-coded macroblocks (available in P and B pictures). This implies the need for architectures that are able to process these different types. Based on the techniques presented in sections 3.3.3 through 3.3.6 for specific macroblock types, we present overall architectures for fast transcoding of H.264/AVC streams to SVC streams in Section 3.3.7. The presented overall architectures consist of a combination of techniques for each macroblock type:

- constrained intra-coded macroblocks
- non-constrained intra-coded macroblocks
- inter-coded macroblocks

The difference between constrained and non-constrained intra-coded macroblocks will be elaborated on in Section 3.3.4.

In the remainder of the chapter, the introduced architectures are depicted for three layers, to illustrate all inter-layer dependency mechanisms, including base layer, top layer, and intermediate layers. The architectures can readily be extended to more (up to 8)[4] or less layers.

[4]Note that level constraints for the Scalable Baseline, Scalable High, and Scalable High Intra profiles restrict the number of dependent dependency layers to three for practical use. Multiple MGS quality levels, however, are allowed for every dependency layer.

3.3.3 Inter-coded macroblocks

For inter-coded macroblocks, the traditional *open-loop* transcoder [5] can be extended for H.264/AVC-to-SVC transcoding as shown in Figure 3.4. In order to obtain multiple SVC dependency layers or quality levels after transcoding, the incoming residual coefficients are redistributed among the different layers. This can be achieved by requantization of the coefficients using the respective layer-dependent quantization parameter. For every layer, requantization using the desired QP is performed. For inter-coded macroblocks, inter-layer residual prediction allows a reduction of the amount of residual data by subtracting the accumulated inverse quantized coefficients from lower layers. This can be applied for every layer (except for the base layer). In this way, refinement of the (coarse) coefficients in the base layer will occur in every available enhancement layer at the decoder.

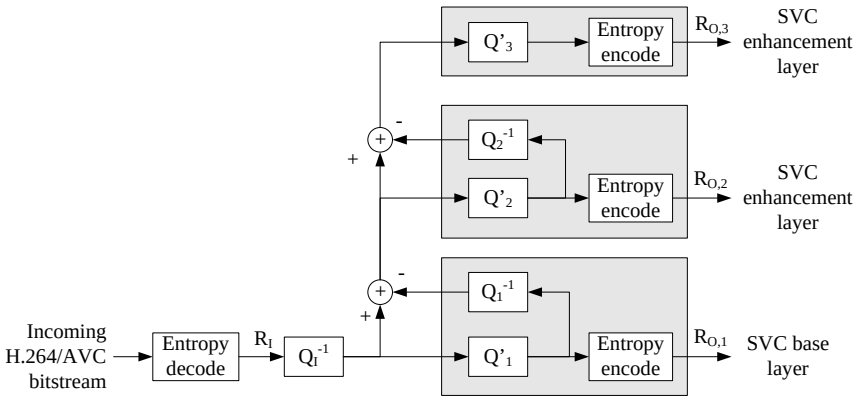


Figure 3.4: Open-loop transcoding architecture.

The output coefficients are obtained as follows:

$$R_{O,l} = Q'_l\Big(Q_I^{-1}(R_I) - \sum_{j=1}^{l-1} Q_j^{-1}(R_{O,j})\Big), \tag{3.1}$$

where R_I is the input residual coefficient and $R_{O,l}$ is the corresponding output coefficient for layer l, with l ranging from 1 to N. Q'_l denotes normalized requantization with the respective QP_l for layer l. Q_I^{-1} and Q_l^{-1} denote inverse quantization with the incoming QP_I and outgoing QP_l, respectively.

At the decoder side, the corresponding SVC macroblock decoding process for inter-coded macroblocks can be seen in Figure 3.5. Here, the coefficients

are inverse quantized and accumulated ($\sum_{l=1}^{N} Q_l^{-1}(R_l)$) before inverse transformation.

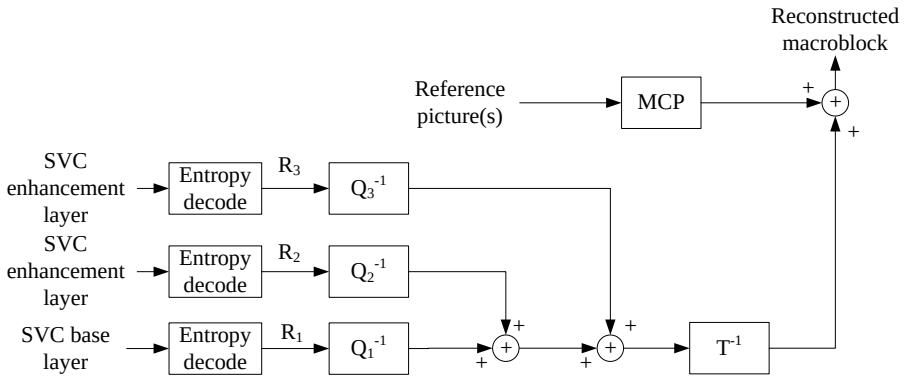


Figure 3.5: Reconstruction of inter-coded macroblocks in SVC.

In order to avoid loss of information and maximize quality, the incoming QP is reused as the QP of the top SVC layer, i.e., $QP_N = QP_I$. Although at first sight it seems that in this way, by decoding all layers (three layers in the shown example), perfect reconstruction is possible after transcoding (identical to reconstruction of the original single-layer bitstream), this is in general not the case. In fact, perfect reconstruction is only achieved when the accumulated sum of inverse quantized coefficients equals the inverse quantized original value R_I, i.e.,

$$Q_I^{-1}(R_I) = \sum_{l=1}^{N} Q_l^{-1}(R_{O,l}). \tag{3.2}$$

Since inverse quantization of a coefficient R in H.264/AVC can be written as $Q^{-1}(R) = R \cdot V \cdot 2^{\lfloor QP/6 \rfloor}$, Equation (3.2) becomes:

$$R_I \cdot V_I \cdot 2^{\lfloor QP_I/6 \rfloor} = \sum_{l=1}^{N} R_{O,l} \cdot V_l \cdot 2^{\lfloor QP_l/6 \rfloor}. \tag{3.3}$$

Here, V_l indicates the integer multiplier coefficient corresponding to QP_l. As introduced in Chapter 2, these values depend on both the QP and the position in the residual block. The subscript $_{ij}$ is omitted for notational simplicity. Since SVC adopts the quantization process of single-layer H.264/AVC, the multiplier values V are defined as was shown in Table 2.2.

It can be shown that the diophantine equation (3.3) does not hold for all combinations of coefficients and QPs, given the integer nature of the coefficients R and multiplier values V in Table 2.2.

An example for *two layers* illustrates this (see Figure 3.6). For small[5] values of $\Delta QP = QP_1 - QP_I$

$$R_{O,1} = Q'_1(Q_I^{-1}(R_I)) = R_I \ , \tag{3.4}$$

i.e., the output coefficient at the higher QP_1 for the base layer will still be identical to the input residual coefficient, due to the small ΔQP. This will lead to a minor difference between $Q_I^{-1}(R_I)$ and $Q_1^{-1}(R_{O,1})$. This difference, however, will be too small to result in an output coefficient in the enhancement layer, i.e.,

$$R_{O,2} = Q'_2\Big(Q_I^{-1}(R_I) - Q_1^{-1}(R_{O,1})\Big) = 0 \ . \tag{3.5}$$

By substituting $R_{O,1} = R_I$ and $R_{O,2} = 0$, Equation (3.3) becomes:

$$V_I = V_1 \ , \tag{3.6}$$

which is not the case when $(QP_1 - QP_I)\%6 \neq 0$, as seen in Table 2.2 (where $\%$ denotes the modulo operation).

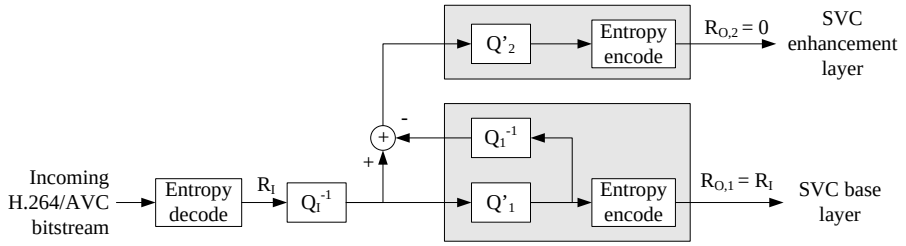


Figure 3.6: Open-loop transcoding example for two layers.

Since a different sum of inverse quantized values $\sum_{l=1}^{N} Q_l^{-1}(R_{O,l}) \neq Q_I^{-1}(R_I)$ can be obtained at the decoder side in this way, small differences will arise after inverse transformation, resulting in a non-identical reconstruction at decoder side (when compared to the original sequence), even when all layers are present at the decoder side.

Drift becomes even more of an issue when layers are dropped before decoding. Due to the open-loop character of this transcoder, requantization errors can propagate and cause temporal drift in the lower layers. To alleviate this, temporal compensation of requantization errors might be applied [6].

[5]Note that the threshold value depends on the incoming coefficient R_I, as well as on the incoming QP_I and the dead zone control parameter used for requantization $Q'_1(\cdot)$.

Although visual quality benefits somewhat from this approach, the resulting architecture would suffer from a large increase in computational complexity due to the added MCP loop for every layer, and the required reference picture buffers.

3.3.4 Intra-coded macroblocks

For intra-coded macroblocks, inter-layer dependencies cannot be exploited in a similar manner, since inter-layer residual prediction is not available for these macroblocks. Instead, inter-layer intra prediction is provided. In the design of transcoding architectures, the SVC concept of single-loop decoding should be taken into account [49]. Single-loop decoding requires that during inter-layer intra prediction, the co-located intra-coded macroblocks in lower layers cannot depend on prediction pixels from neighboring *inter-coded macroblocks*. If such a dependency was present, an additional MCP loop would be required in order to decode the lower layer(s). This is in contrast to the single-loop decoding design, which reduces decoder complexity by only requiring one MCP decoding loop (for the highest available layer). Therefore, to obtain a compliant SVC stream, for intra-coded macroblocks, dependencies on inter-coded macroblocks in lower layers should not be introduced during transcoding. When the I_BL macroblock type is selected in the SVC bitstream, lower layers are required to use constrained intra prediction. This adds an additional issue for incoming H.264/AVC streams that do not have the constrained intra prediction option enabled. Without reconstruction to the pixel domain (which implies the introduction of MCP loops in the architecture), the dependencies on inter-coded macroblocks cannot be removed.

From this, it becomes clear that other techniques are necessary for intra-coded macroblocks and pictures. Different cases are possible:

- **Intra-coded pictures** Reconstruction of pixel values and reencoding of the intra-coded pictures can be performed without requiring an MCP loop in every layer during transcoding.

- **Intra-coded macroblocks in MCP (P or B) pictures with constrained intra prediction** When constrained intra prediction is enabled in the original H.264/AVC stream, regions of intra-coded macroblocks can be reconstructed and reencoded as in the case of intra-coded pictures.

- **Intra-coded macroblocks in MCP pictures without constrained intra prediction** If no constrained intra prediction is used for the intra-coded macroblocks in P and B pictures in the incoming H.264/AVC

stream, it would be necessary to decode neighboring inter-coded macroblocks in every layer to obtain the prediction pixels at the edges of intra-coded regions. Since this requires adding MCP loops to the architecture, such a pixel-domain solution would greatly increase computational complexity. Hence, other solutions are necessary for fast, *single-loop transcoding* architectures. Since constrained intra prediction is typically disabled in H.264/AVC coding for improved coding efficiency, special attention is required when designing architectures for H.264/AVC-to-SVC transcoding.

In the remainder of this chapter, we make a distinction between the case of *constrained intra-coded macroblocks*, i.e., intra-coded macroblocks in intra-coded pictures[6] or MCP pictures with constrained intra prediction enabled on the one hand, and *non-constrained intra-coded macroblocks*, for intra-coded macroblocks that may rely on inter-coded macroblocks in MCP pictures on the other hand. The single-loop decoding requirement introduces additional difficulties for the latter category.

3.3.5 Constrained intra-coded macroblocks

A. Intra BL architecture

For constrained intra-coded macroblocks, inter-layer dependencies can be exploited by using the architecture shown in Figure 3.7. This architecture can also be derived from the cascaded decoder-encoder by removing MCP, and by removing intra prediction in the SVC enhancement layers. Two intra prediction loops are maintained, i.e., one in the decoder loop, and one in the SVC base layer loop. Inter-layer redundancies are exploited by subtracting the reconstructed lower layer intra macroblock from the decoded macroblock (i.e., inter-layer intra prediction). In the bitstream, this is reflected by enabling the *base mode flag*, resulting in macroblocks of the inferred I_BL type. In this way, quality refinement of the macroblocks will occur in every layer. Inter-layer deblocking can be applied, as indicated in the slice header of the SVC enhancement layers.

If desired, an intra prediction loop might be added for every enhancement layer, for refinement of the base layer intra prediction modes. In this way, a better mode in rate-distortion sense can be selected. Fast methods to achieve this can be used, as discussed for example in [34]. Although this adds complexity, no MCP loops are required when this is used for constrained intra-coded macroblocks.

[6] Intra-coded pictures can be regarded as an 'extreme case' of a constrained intra region.

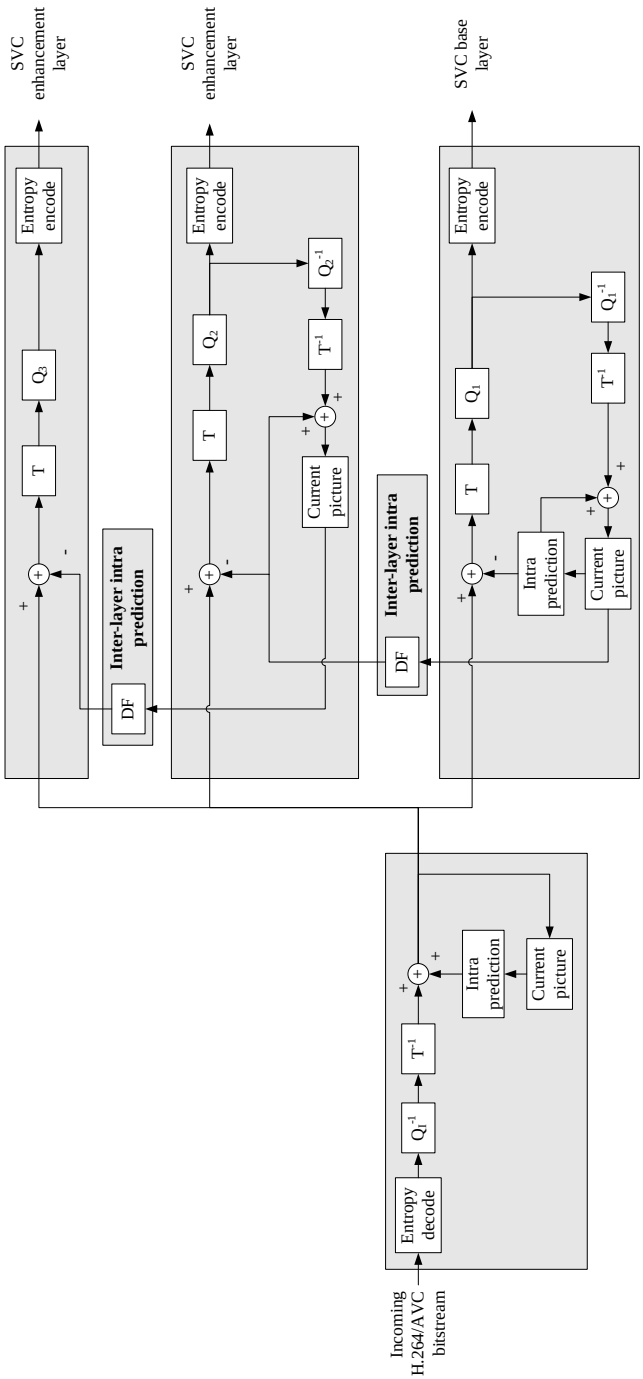


Figure 3.7: Intra BL architecture for constrained intra-coded macroblocks.

B. Intra copy

A low-complexity alternative is to concentrate all intra-coded data in the SVC base layer, while the enhancement layers contain no residual data (the I_BL macroblock type will allow the base layer data to propagate to higher layers). This can be regarded as a low-complexity technique, since only high-level syntax changes are required. In this way, the incoming residual data can simply be copied to the output bitstream. The rate-distortion performance of the output SVC sequence will benefit from this approach, since layered coding results in a rate-distortion penalty when compared to single-layer coding. Obviously, this approach offers no flexibility regarding rate distribution whatsoever for these macroblocks. Also, a higher bit rate will be obtained for the base layer, which is not desirable in many situations, and might pose a problem in rate-controlled systems.

3.3.6 Non-constrained intra-coded macroblocks

Both of the above architectures for constrained intra-coded macroblocks have the advantage of low computational complexity and memory requirements, and can readily be used for constrained intra-coded macroblocks. For intra-coded macroblocks in MCP pictures, the problem exists that reconstruction of neighboring MCP macroblocks is required if the *constrained intra prediction flag* was not enabled in the original H.264/AVC bitstream. Constrained intra prediction is a key element in the SVC single-loop decoding design. For this reason, other solutions need to be provided for non-constrained intra-coded macroblocks.

Intra-layer intra prediction (intra simulcast)

Since the use of the I_BL macroblock type implies the presence of constrained intra prediction in lower layers, it is not possible to benefit from this macroblock type without computationally complex macroblock reconstruction. The typical intra 4×4, 8×8, and 16×16 macroblock types, however, can still be used. In this way, the requirement of constrained intra prediction is dropped. Redundancy, however, will increase since inter-layer dependencies are not exploited for intra-coded macroblocks. This technique can be regarded as an *intra simulcast* solution. Clearly, rate-distortion performance will diminish when more intra-coded macroblocks are present in MCP pictures. Since a complete decoder loop is to be avoided (which would imply decoding MCP blocks), the approach here is to work (partially) in the transform domain. This can be seen from Figure 3.8. A decoder loop is avoided, and operations are per-

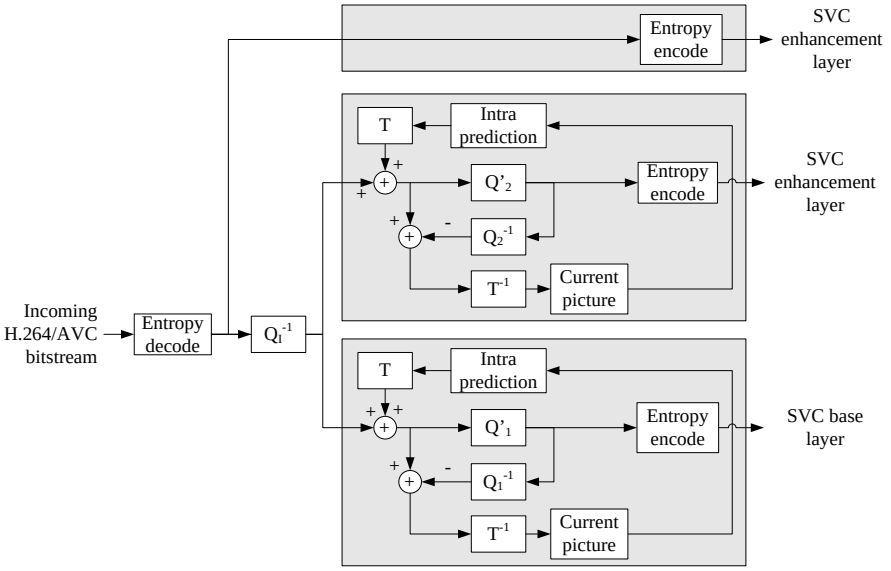


Figure 3.8: Intra-layer intra prediction (intra simulcast) for non-constrained intra-coded macroblocks.

formed on the residual data, instead of on the reconstructed pixel values. An inverse transform is still used, however, to obtain pixel-domain (difference) values for prediction. This avoids highly complex transform-domain intra prediction formulas, as in [57]. Note that the buffers do not contain decoded pixel values; instead, they contain requantization error values (after inverse transformation). These values are stored in the buffer for spatial compensation of neighboring macroblocks. This technique is also applied for single-layer H.264/AVC stream transcoding as discussed in Chapter 2.

3.3.7 Overall H.264/AVC-to-SVC transcoding architectures

The overall H.264/AVC-to-SVC transcoding architectures will be formed by a combination of the previously discussed techniques, depending on the picture and/or macroblock type. Based on the input sequence (whether or not constrained intra prediction is used), a switch is made between inter-layer or intra-layer intra prediction for intra-coded macroblocks. For intra-coded pictures, a selection can be made between the intra BL or intra copy architectures. For inter-coded macroblocks, the open-loop architecture with inter-layer residual prediction is used. Although open-loop requantization will lead to temporal drift in the MCP pictures, this technique offers a good compromise be-

tween output quality and computational complexity, since no MCP loops are required.

We distinguish between the following two (low-complexity) architectures (the difference lies in the technique used for the constrained intra-coded macroblocks):

A. Intra BL architecture

- Intra BL for constrained intra macroblocks.
- Intra-layer intra prediction for non-constrained intra macroblocks (I_N×N macroblock types).
- Open-loop transcoding for inter-coded macroblocks.

B. Intra copy architecture

- Intra copy for constrained intra macroblocks (no residual data in enhancement layers).
- Intra-layer intra prediction for non-constrained intra macroblocks (I_N×N macroblock types).
- Open-loop transcoding for inter-coded macroblocks.

Other techniques for the non-constrained intra-coded and the inter-coded macroblocks would unavoidably lead to higher complexity. On the one hand, adding temporal compensation to inter-coded macroblocks introduces multiple MCP loops. On the other hand, introducing inter-layer prediction for the non-constrained intra-coded macroblocks founders on the single-loop decoding requirement. A conversion of non-constrained to constrained prediction would resolve this, but would imply full reconstruction of these macroblocks, and of the surrounding MCP macroblocks (again requiring MCP prediction loops).

3.4 Bitstream rewriting

Bitstream rewriting was added to the SVC design to allow low-complexity combination of CGS dependency layers or MGS quality layers to a single-layer H.264/AVC stream, hereby providing backward compatibility for existing H.264/AVC decoding equipment.

The technique for rewriting an SVC stream with multiple dependency layers to a single-layer H.264/AVC stream was introduced in [79]. Initially this was applied to CGS (the MGS functionality was added later on in the standardization process), but since the difference between CGS and MGS is primarily limited to a number of high-level syntax changes, the technique is also applicable to MGS quality levels.

A number of changes were required to the SVC syntax and coding tools in order to allow this kind of rewriting functionality. In particular, two major aspects of the decoding process were modified [49].

- For inter-layer intra prediction, the prediction signal is no longer formed by the reconstructed intra signal of the reference layer, but spatial intra prediction as in single-layer H.264/AVC is performed in the target layer. The residual signal is formed in the same way as for MCP macroblock types (see Figure 3.9). This also means that the constrained intra prediction requirement for the lower layers is dropped for I_BL macroblocks, in the case that the bitstream rewriting functionality is applied.

- Residual prediction is performed in the transform coefficient level domain. Not the inverse quantized coefficients, but the quantization levels are scaled and accumulated during decoding.

In Figure 3.9, both concepts are illustrated using the adapted reconstruction process of intra-coded macroblocks (compare with the original reconstruction process for MCP macroblocks in Figure 3.5). In the SVC specification, the inverse and forward quantization steps Q_1^{-1} and Q'_2, and Q_2^{-1} and Q'_3 are combined, resulting in efficient scaling formulas.

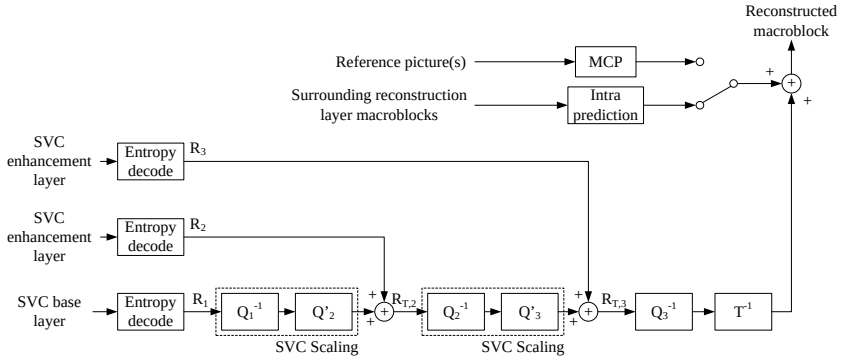


Figure 3.9: Reconstruction of intra-coded and MCP macroblocks using adapted coefficient accumulation process.

For the typical case where the QP decreases for every enhancement layer, i.e., $QP_l \leq QP_{l-1}$, for $l \geq 2$, scaling is performed as follows:

$$R_{T,l} = R_l + \left\lfloor \frac{cS_{(QP_{l-1}-QP_l)} \cdot R_{T,l-1} \cdot 2^{\lfloor \frac{QP_{l-1}-QP_l}{6} \rfloor}}{8} \right\rfloor \tag{3.7}$$

and $R_{T,1} = R_1$.

The cS values correspond to the ratio of H.264/AVC quantization step sizes, i.e.,

$$cS_{(QP_{l-1}-QP_l)} = \left\lfloor 8 \cdot \frac{Q_{\text{step}_{l-1}}}{Q_{\text{step}_l}} \right\rfloor. \tag{3.8}$$

The H.264/AVC quantization step sizes are reused, which were shown in Table 2.1.

From these values and Equation (3.8), the values for cS are obtained as shown in Table 3.1. Note that the cS values are standardized in a way in which

Table 3.1: Scaling values cS.

$\Delta QP\%6$	cS
0	8
1	9
2	10
3	11
4	13
5	14

they only depend on the difference in QP, instead of on the QPs themselves. Although these values are not exact for *all* combinations of $Q_{\text{step},1}$ and $Q_{\text{step},2}$, they provide a close approximation and allow rewriting to be based solely on integer arithmetic. More information on the bitstream rewriting process can be found in [49, 85].

3.5 Flexible H.264/AVC-to-SVC transcoding based on bitstream rewriting

Exploiting the bitstream rewriting functionality in H.264/AVC-to-SVC transcoding has a number of advantages. In this section, we show that architectures for H.264/AVC-to-SVC transcoding benefit from the changes that were introduced in the SVC syntax to allow bitstream rewriting.

If the typical case is examined where the QP of the top layer in the output SVC stream corresponds with the original QP of the incoming H.264/AVC stream, it becomes possible to improve the previously discussed architectures. When using the rewriting functionality, the fact that the accumulated sum of (scaled) coefficients equals the incoming coefficients can be exploited. This is a consequence of the requirement that residual prediction is performed in the transform coefficient level domain. In this case, this applies to both intra-coded and inter-coded macroblocks. The top layer coefficients (layer N) are calculated as

$$R_{O,N} = R_I - \left\lfloor \frac{R_{T,N-1} \cdot cS_{(QP_{N-1}-QP_N)} \cdot 2^{\lfloor \frac{QP_{N-1}-QP_N}{6} \rfloor}}{8} \right\rfloor . \tag{3.9}$$

Here, $R_{T,N-1}$ is the total accumulated sum of scaled output coefficients up to layer $N-1$. Note that during transcoding, $R_{T,N-1}$ is calculated as

$$R_{T,l} = Q'_l(Q_1^{-1}(R_I)) \tag{3.10}$$

for all layers $l < N$, while during decoding, Equation (3.7) is used.

In the SVC syntax, the *tcoeff level prediction flag* indicates the use of the changed transform coefficient level scaling and refinement process during decoding.

Based on this, the previous architectures can be redesigned. This indicates that using the rewriting functionality has the advantage that perfect reconstruction is achieved in the top layer of the SVC stream. While for the architectures in Section 3.3, errors were still possible when $(QP_{l-1} - QP_l)\%6 \neq 0$, the rewriting functionality allows for identical accumulation and reconstruction of the coefficients in the different layers.

Perfect reconstruction, however, does not apply to the other layers. The loss of information due to requantization will propagate in these lower layers, both spatially and temporally due to intra prediction and MCP, respectively. In particular, drift due to intra prediction has been shown to be a major issue in H.264/AVC video coding [58]. To overcome the intra drift issue, a number of strategies can be used, which will be discussed in the remainder of this section. In particular, the goal is to keep the computational complexity as low as possible, without compromising quality. As in Section 3.3, we discuss transcoding techniques for the different macroblock types, and develop overall architectures based on these techniques.

3.5.1 Inter-coded macroblocks

Slight changes are made to the open-loop transcoding architecture, resulting in the architecture shown in Figure 3.10. The standardized SVC scaling process

can be used for transform coefficient accumulation. It can be seen that the coefficient subtraction process in this architectures corresponds to the coefficient accumulation process in Figure 3.9 for MCP blocks.

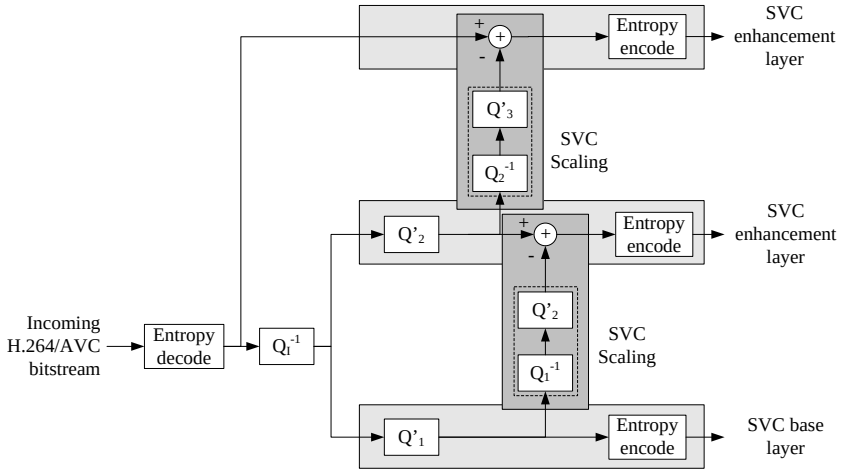


Figure 3.10: Open-loop transcoding for MCP macroblocks based on changed coefficient accumulation process.

Although spatial error propagation has a decisive impact on the total amount of drift in the transcoded sequences, it might be beneficial to add temporal compensation for inter-coded macroblocks to further improve quality of the lower SVC layers (note that no drift is present in the top layer due to perfect accumulation). Again, although temporal compensation improves quality, complexity will significantly increase due to the added MCP loops.

3.5.2 Intra-coded macroblocks

Apart from the architectures discussed in Section 3.3, the changes for bitstream rewriting allow new architectures for H.264/AVC-to-SVC transcoding for intra-coded macroblocks and pictures. These additional architectures have the benefit that they provide full flexibility regarding rate distribution and/or that they exploit inter-layer dependencies by using the inter-layer *residual* prediction mechanism, resulting in improved coding efficiency.

3.5.3 Constrained intra-coded macroblocks

Intra BL transcoding with inter-layer residual prediction (intra BL residual)

This architecture is based on the first change for the bitstream rewriting functionality, i.e., the updated reconstruction process for intra-coded macroblocks. This architecture is similar to the intra BL architecture, with the difference that transform coefficient values are accumulated as is done for inter-layer residual prediction, as opposed to the subtraction of reconstructed pixel values in the intra BL architecture. This architecture is depicted in Figure 3.11, and will for simplicity be referred to as the *intra BL residual* architecture. The SVC scaling formulas are used for accumulation of the residual coefficients in lower layers. For all layers except the top layer, an intra prediction loop is included, whereas for the intra BL architecture, this was only the case in the base layer.

Since the *tcoeff level pred flag* can be set at slice header level, a choice can be made between the intra BL and intra BL residual architecture for intra-coded pictures.

3.5.4 Non-constrained intra-coded macroblocks

A. Open-loop transcoding

As mentioned, the constrained intra prediction requirement is canceled when bitstream rewriting is enabled. In this way, the open-loop transcoding architecture (Figure 3.10) can also be applied to intra-coded pictures and intra-coded macroblocks in P and B pictures. Similar to temporal drift, however, spatial drift will arise due to spatial dependencies. Due to the large number of dependent blocks in H.264/AVC intra coding [40], errors will propagate rapidly, and result in serious artifacts in video streams extracted from lower layers.

B. Inter-layer residual prediction with spatial compensation

In order to alleviate the spatial drift problem of open-loop transcoding, another architecture is introduced for H.264/AVC-to-SVC transcoding which provides full flexibility, and is based on the changes introduced by bitstream rewriting. Similar to the intra-layer intra prediction (intra simulcast) architecture, it is possible to perform spatial compensation operations on intra-coded macroblocks in MCP pictures, hereby avoiding reconstruction of surrounding inter-coded macroblocks. Additionally, since constrained intra prediction is no longer a requirement in lower layers, this architecture benefits from inter-layer residual prediction, as is shown in Figure 3.12. When compared to the

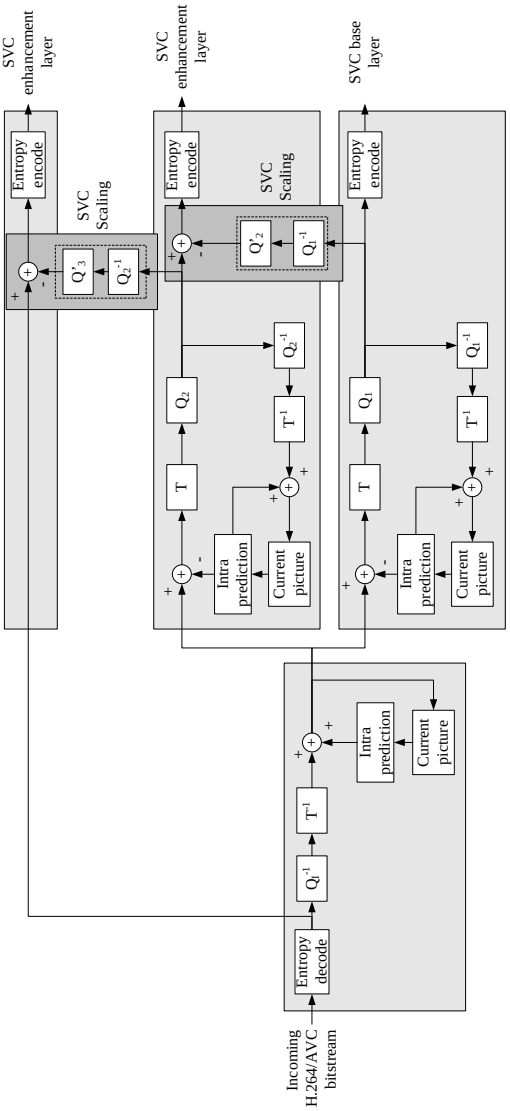


Figure 3.11: Intra BL transcoding with inter-layer residual prediction for constrained intra-coded macroblocks.

open-loop transcoding architecture, a spatial compensation loop is introduced which neutralizes the loss of information during requantization. In this way, the loss of information will not propagate due to intra prediction. To avoid this requantization error propagation, compensation techniques are used that were introduced in Chapter 2. This architecture has the advantage that no decoding

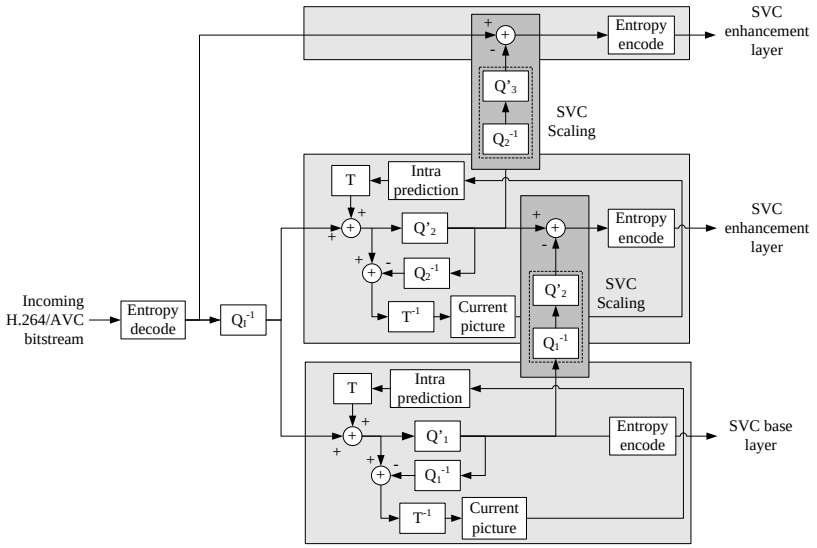


Figure 3.12: Inter-layer residual prediction with spatial compensation for non-constrained intra-coded macroblocks.

is necessary for MCP blocks, yet full flexibility is possible for requantization of intra-coded macroblocks in P and B pictures. In this way, an advantage is created over the intra copy or intra-layer intra prediction architectures, which lack either rate distribution flexibility or rate-distortion performance.

C. Intra copy

The intra copy technique was introduced in Section 3.3 for constrained intra-coded macroblocks. Due to the bitstream rewriting functionality, this technique also becomes available for non-constrained intra-coded macroblocks. When compared to open-loop transcoding, the spatial drift effect is avoided. However, the disadvantage of reduced rate distribution capabilities of the transcoder also applies here, i.e., restrictions are introduced for the achievable bit rates of the lower layers. In practice, concentrating the residual data in the base layer will be obtained by only increasing the macroblock QP for non-intra coded macroblocks in the base layer, while the QP will be unchanged for

intra macroblocks. This is reflected by a non-zero value of *mb qp delta* in the bitstream.

3.5.5 Overall architectures

The previously discussed transcoding techniques can be combined in different ways to obtain overall architectures with different computational complexity and rate-distortion performance. Low-complexity transcoding architectures can be constructed that are not based on multiple reconstruction MCP loops.

For intra-coded pictures, the intra BL, intra copy, or intra BL residual architectures can be used. For intra-coded macroblocks, open-loop transcoding can also be applied, but due to the drift which propagates rapidly in lower layers due to intra prediction, it is recommended to use either the intra copy technique or the single-loop compensation technique for full flexibility. For MCP blocks, the open-loop architecture can be used at minimal complexity. For all these architectures, perfect reconstruction is achieved when all layers are present at the decoder.

We propose four architectures, based on applicable combinations of the techniques discussed above (the intra BL architecture, which has been discussed in Section 3.3, is omitted here). In the results section, these architectures are identified using the prefix 'RW' for 'rewriting'.

A. Open-loop architecture (RW-OL)

- This architecture uses open-loop transcoding for all macroblock types. Inter-layer residual prediction is used for coefficient accumulation and refinement.

B. Intra BL residual architecture (RW-IR)

- Intra BL residual for constrained intra-coded macroblocks.
- Open-loop transcoding for non-constrained intra-coded macroblocks, and for inter-coded macroblocks (both use inter-layer residual prediction).

C. Intra BL residual architecture with spatial compensation (RW-IRSC)

- Intra BL residual for constrained intra-coded macroblocks.
- Inter-layer residual prediction with spatial compensation for non-constrained intra-coded macroblocks.

- Open-loop transcoding with inter-layer residual prediction for inter-coded macroblocks.

D. Intra copy architecture (RW-IC)

- Intra copy for constrained and non-constrained intra-coded macroblocks.

- Open-loop transcoding with inter-layer residual prediction for inter-coded macroblocks.

3.6 Results and discussion

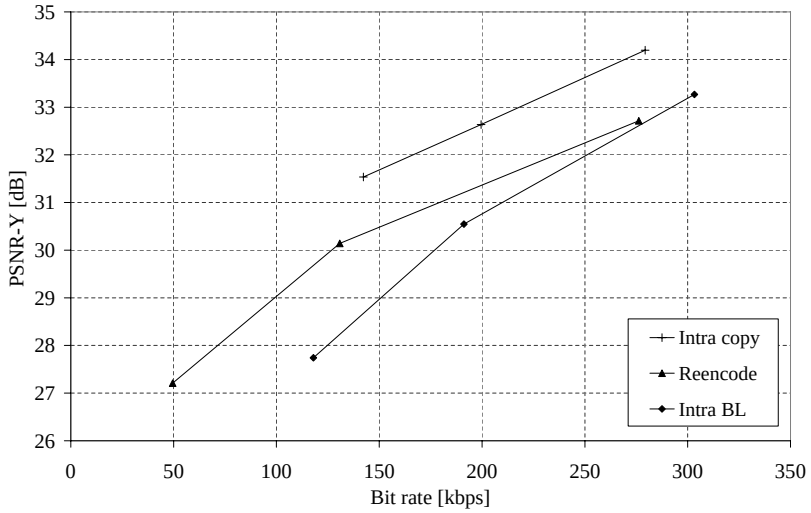
In this section, we show the results from the implementation of the architectures described in the previous sections. The test sequences *Mobile & Calender*, *Foreman*, and *Stefan* (CIF resolution) were used. The sequences were encoded using the H.264/AVC Joint Model (JM) reference software, version 13.2, using hierarchical GOP structures. Hierarchical patterns were used to allow comparison with the JSVM software, and to allow both SNR scalability and temporal scalability after transcoding. A GOP length of 8 and an intra period of 32 were used. We then transcoded the bitstreams to SVC streams with multiple dependency layers (CGS). Starting QPs of 22, 27, 32, and 37 were used. Constrained intra prediction was disabled in the encoded bitstreams for maximum compression performance, as is typically the case in H.264/AVC streams. CAVLC entropy coding was used.

3.6.1 Rate-distortion results

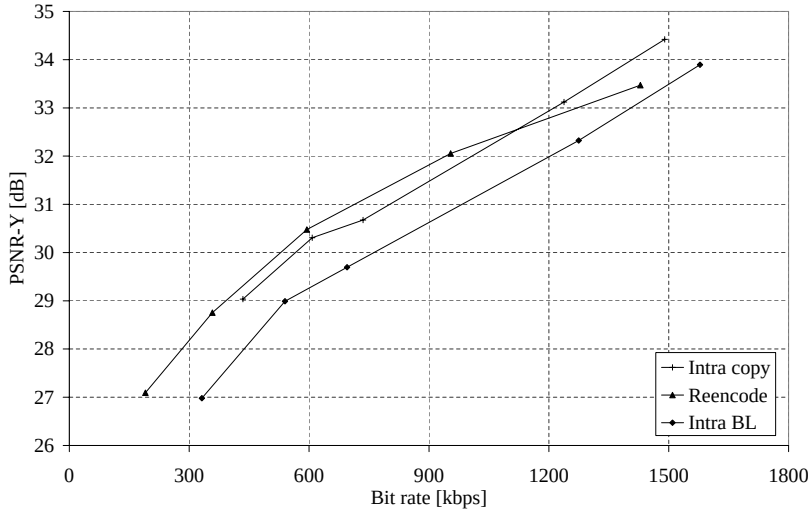
The extraction points of the layered output sequences are plotted in rate-distortion diagrams. The R-D point with the highest bit rate for each curve corresponds with the output SVC sequence where all layers are present. Every additional point on the curve corresponds with the R-D values of the output sequence where one or more dependency layers have been dropped (using the JSVM bitstream extractor software). The R-D point with the lowest bit rate corresponds to the base layer, after removal of all enhancement layers. We used fixed quantization parameters (as opposed to rate-controlled (trans)coding) to allow comparison with the JSVM reference software.

A. Architectures without bitstream rewriting functionality

First, we plot the results for the two architectures that do not make use of the bitstream rewriting functionality. For the *Foreman* sequence, the plot is shown in Figure 3.13(a) for 3 dependency layers. The intra copy technique



(a) Results for *Foreman* sequence (CIF, 30 Hz) with 3 layers ($\Delta QP = 6$, $QP_I = QP_N = 32$).



(b) Results for *Stefan* sequence with 5 layers ($\Delta QP = 3$, $QP_I = QP_N = 27$).

Figure 3.13: Results for architectures without bitstream rewriting functionality.

performs particularly well, and outperforms the intra BL technique by 1.5 to 3 dB. A number of effects contribute to this. Firstly, all intra-coded data is partitioned in the base layer, which corresponds to single-layer coding, while enhancement layers contain no residual data. This can be efficiently indicated in the enhancement layers by using the slice skip flag. Since single-layer coding is more efficient than layered coding, this results in a substantial benefit for the intra copy technique, given the large share of intra-coded data in the bitstream. Secondly, by concentrating all intra data in the base layer, reference frames with maximum available quality are created for MCP of other frames. This results in an improvement of subjective and objective quality when compared to the case where lower-quality reference frames are used. This can be regarded as an extreme case of the QP distribution for hierarchical coding, where typically a lower quantization parameter is used for lower temporal layers. The technique of coding frequently used reference pictures with higher fidelity than other reference pictures was proposed in [86], and resulted in bit rate reductions of up to 10% compared to the JSVM quantization method. In the proposed approach, these findings are also found, and indicate that assigning maximum fidelity to intra-coded pictures has a beneficial impact on the entire sequence.

For reference, the rate-distortion curves for reencoding are also plotted. The curves show that the intra copy architecture is able to outperform reencoding. This, however, comes at the cost of limited rate flexibility. For identical values of ΔQP between the successive layers, reencoding can obtain lower rate points than when using the intra copy architecture.

Results for the *Stefan* sequence with 5 layers are shown in Figure 3.13(b). Here, reencoding outperforms both transcoding architectures for the lower extraction points, while intra copy scores best when all (or all but one) layers are present at the decoder. Using the decoder-encoder cascade results in improved rate-distortion performance for lower extraction points, but in degraded performance for the highest extraction points. This can be explained by the fact that in the JSVM software, every layer is optimized separately, i.e., no global optimization across all layers is performed [87]. This leads to a successive refinement of mode decisions and motion vector information, which are optimal for every layer in itself. Although refinement of motion information in SVC can be performed in a relatively efficient way by using inter-layer motion prediction, the inter-layer prediction technique is still outperformed by single-layer coding. This leads to the effect that R-D points for reencoding are above the transcoding architectures, but below when all layers are present. The latter results from the fact that mode decisions and motion vector information are optimal for single-layer coding at the highest extraction point.

B. Architectures based on bitstream rewriting

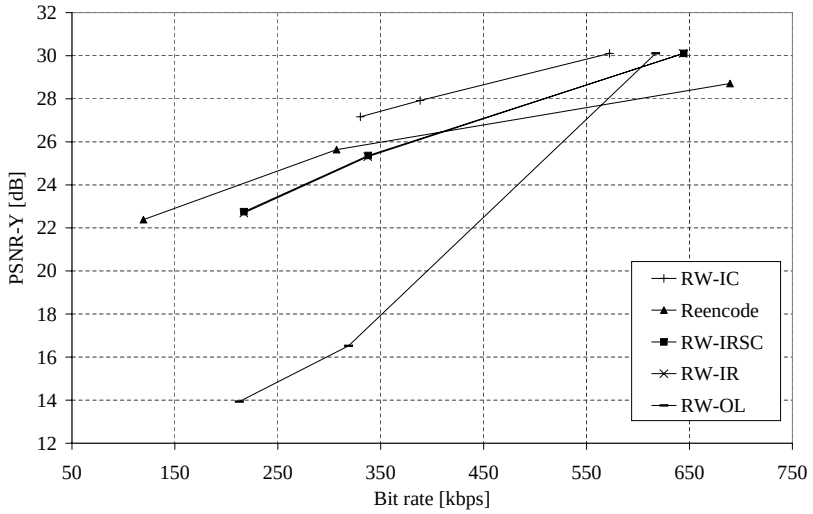
Here, the curves of the proposed architectures are given that are based on the bitstream rewriting syntax changes, and compared to the decoder-encoder cascade. The resulting plots for the *Mobile & Calender* sequence with 3 dependency layers are shown in Figure 3.14(a). It can be seen that all four transcoder architectures lead to perfect reconstruction when all layers are present at the decoder (the top layer R-D points have identical PSNR values), even when $\Delta QP \neq 6$. In the plot, the open-loop architecture is included. It is clear that, although perfect reconstruction is achieved when all layers are present, temporal and spatial drift in the lower layers results in unacceptable results for this architecture, with a PSNR loss of nearly 10 dB for the lowest extraction points.

When taking a closer look at the three most promising architectures (i.e., after removal of the open-loop architecture), similar results can be seen for the *Foreman* sequence (Figure 3.15; in these plots, we used values of $\Delta QP = 3$ or $\Delta QP = 6$, and 3 or 5 output layers).

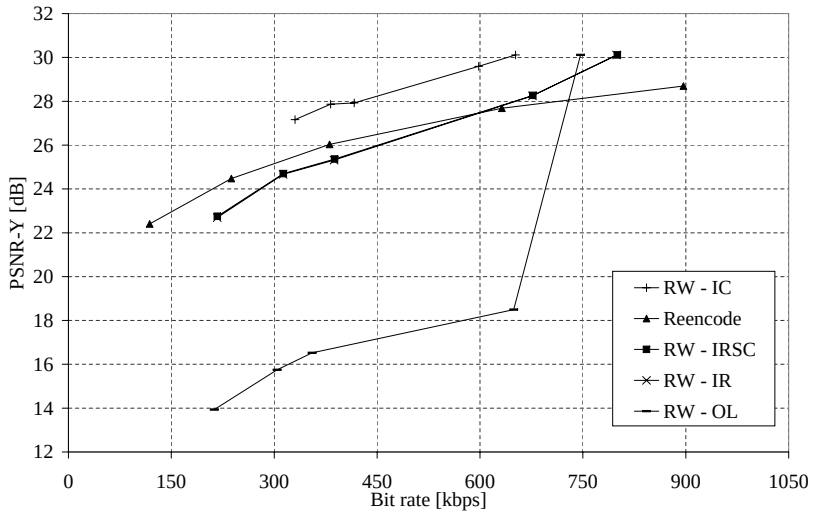
It can be seen that spatial drift results in significant losses, which can either be coped with using spatial compensation techniques, or with the intra copy technique. The curve for the intra copy architecture is well above the other architectures, but has the significant disadvantage that rate distribution flexibility is seriously compromised by partitioning all intra residual data in the base layer. In this way, achieving predefined rate points becomes difficult.

When looking at a large range of bit rates, such as for an SVC sequence containing 5 layers with a $\Delta QP = 6$ between each successive layer, results are obtained as shown in Figure 3.15(d). Here, the intra copy technique performs well for a limited range of bit rates. When the QP gap increases, the rate-distortion curve will drop rapidly. This is also the case for the *Stefan* sequence, where the intra copy curve declines rapidly, as shown in Figure 3.16(d). Here, the R-D curve for the intra copy architecture drops below the quality of the architecture with intra BL residual and spatial compensation (RW-IRSC). The rapid decline of the intra copy architectures can be explained by the fact that saturation will occur near the bit rate of the sum of motion data plus intra-coded residual data. The nearer you get to this saturation point, the more inter-coded residual data has to be removed, and quality will rapidly decline for the inter-coded pictures.

The limited achievable bit rate range for the RW-IC architecture might be circumvented by using a combined architecture, which applies a mixture of intra copy and spatial compensation techniques. The resulting architecture will span a larger range of bit rates (i.e., down to the lowest extraction point for RW-IRSC). For such a combined architecture, rapid saturation can also be avoided, and the resulting R-D curve will approach the lowest point for RW-



(a) Results for *Mobile & Calender* sequence with 3 layers ($\Delta QP = 6$, $QP_I = QP_N = 32$).

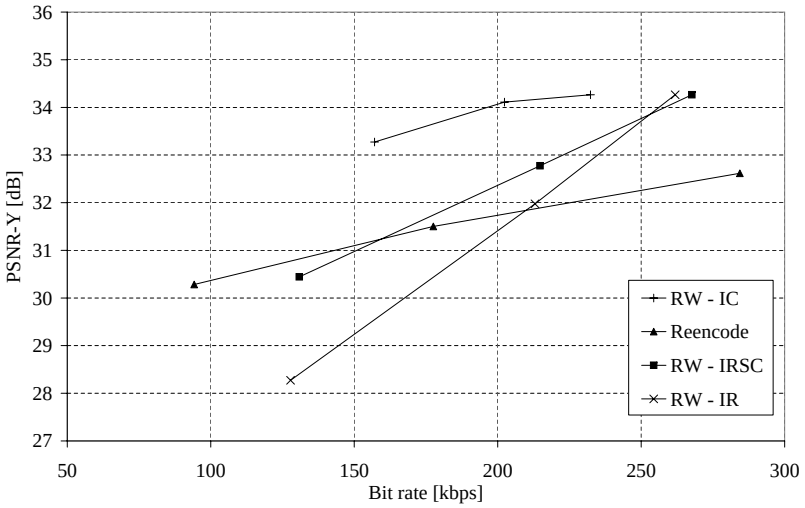


(b) Results for *Mobile & Calender* sequence with 5 layers ($\Delta QP = 3$, $QP_I = QP_N = 32$).

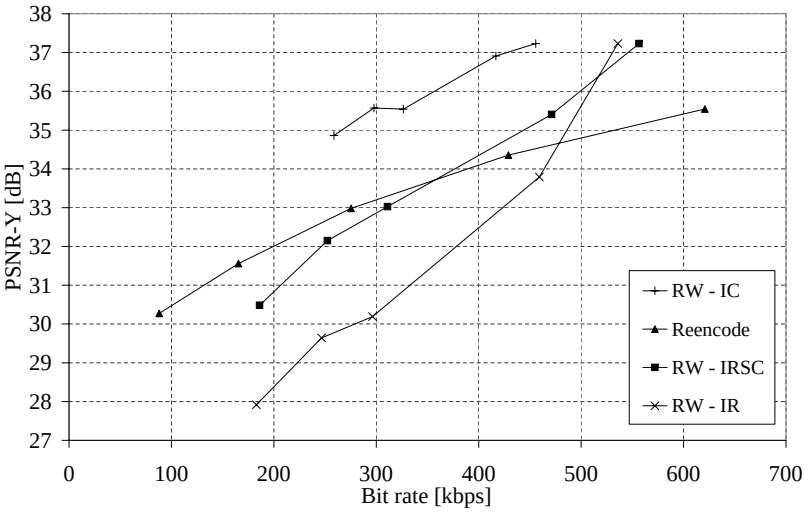
Figure 3.14: Results for *Mobile & Calender* sequence (CIF, 30 Hz).

IRSC more smoothly.

For the *Stefan* sequence (Figure 3.16), the same architectures perform best, i.e., the intra copy architecture outperforms reencoding, as long as saturation is not reached. When compared to the *Foreman* results, the gap between the



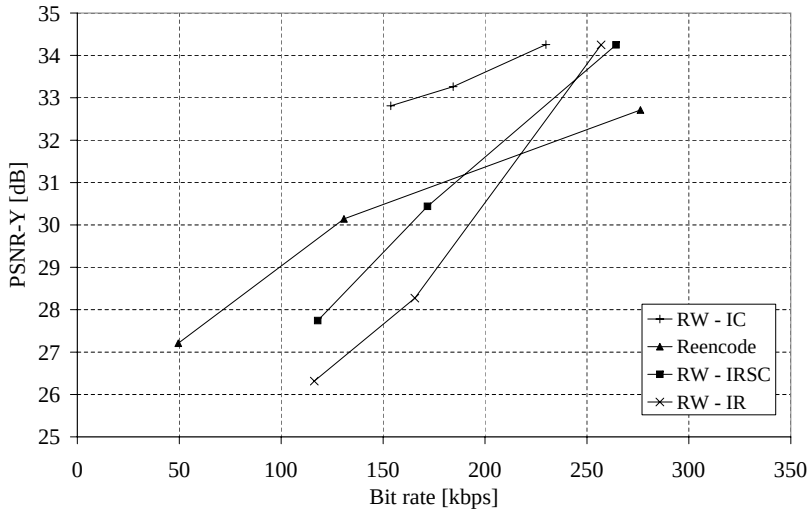
(a) Results for *Foreman* sequence with 3 layers ($\Delta QP = 3$, $QP_I = QP_N = 32$).



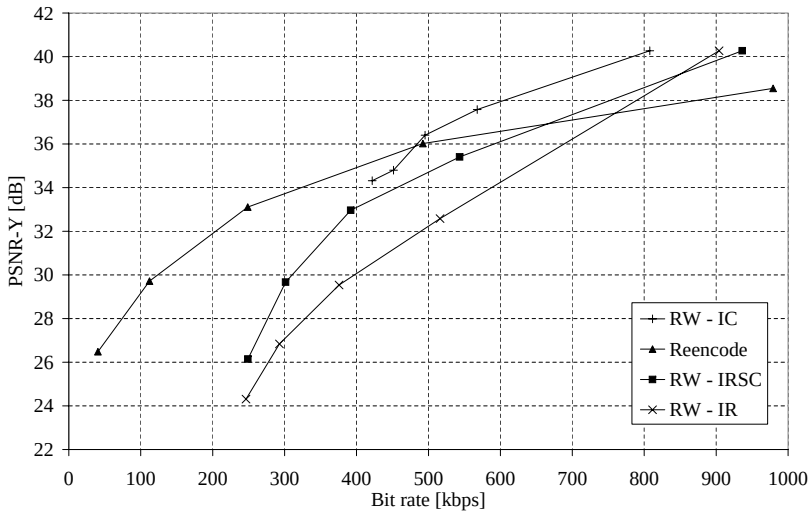
(b) Results for *Foreman* sequence with 5 layers ($\Delta QP = 3$, $QP_I = QP_N = 27$).

Figure 3.15: Results for *Foreman* sequence (CIF, 30 Hz).

RW-IRSC and RW-IR architectures increases, up to 4 dB. This is due to the higher motion activity of the *Stefan* sequence, and the presence of regions of intra-coded macroblocks in the pictures. For the latter, spatial compensation in these macroblocks has an important advantageous impact and is able to restrict spatial drift.



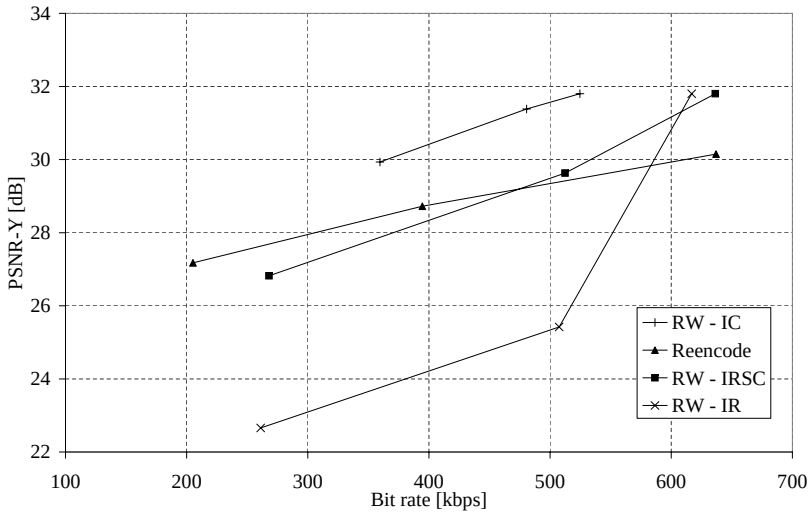
(c) Results for *Foreman* sequence with 3 layers ($\Delta QP = 6$, $QP_I = QP_N = 32$).



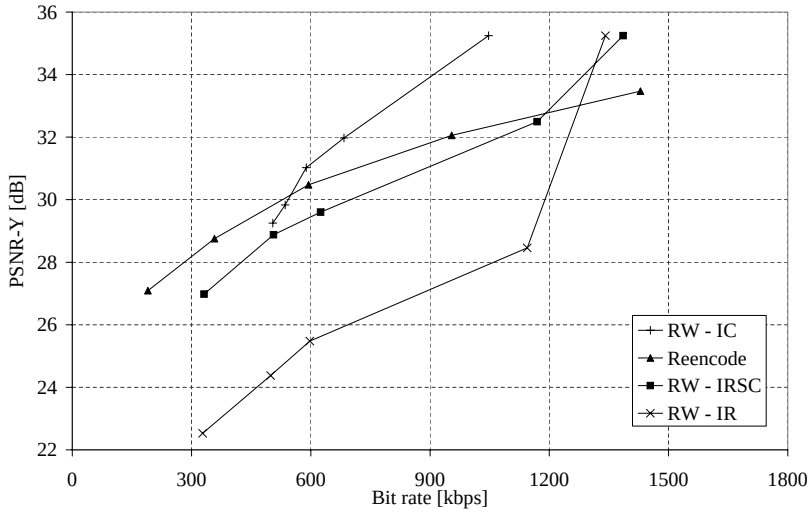
(d) Results for *Foreman* sequence with 5 layers ($\Delta QP = 6$, $QP_I = QP_N = 22$).

Figure 3.15: Results for *Foreman* sequence (CIF, 30 Hz) - continued.

In Figure 3.17 and Figure 3.18, the 96 first frames (three intra periods) of the *Foreman* and *Stefan* sequences are shown with their corresponding PSNR values after transcoding using the architectures with bitstream rewriting functionality (except for the open-loop architecture). The output SVC streams each contain three dependency layers. For every architecture, identical symbols are



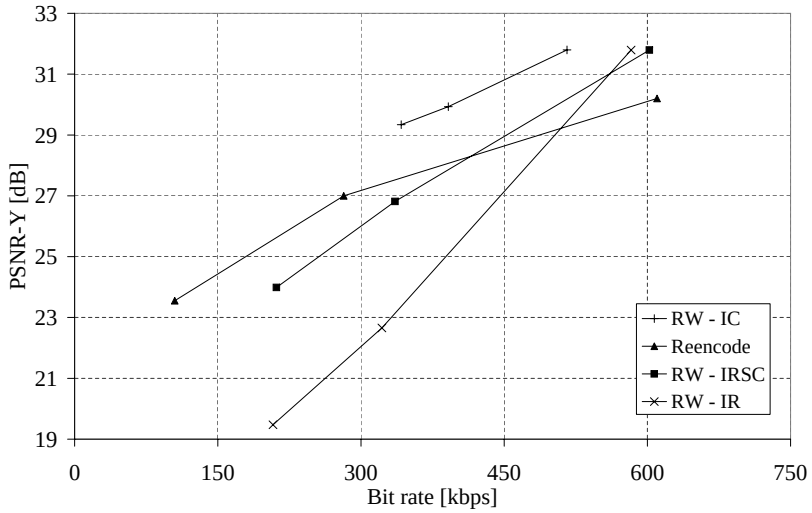
(a) Results for *Stefan* sequence with 3 layers ($\Delta QP = 3$, $QP_I = QP_N = 32$).



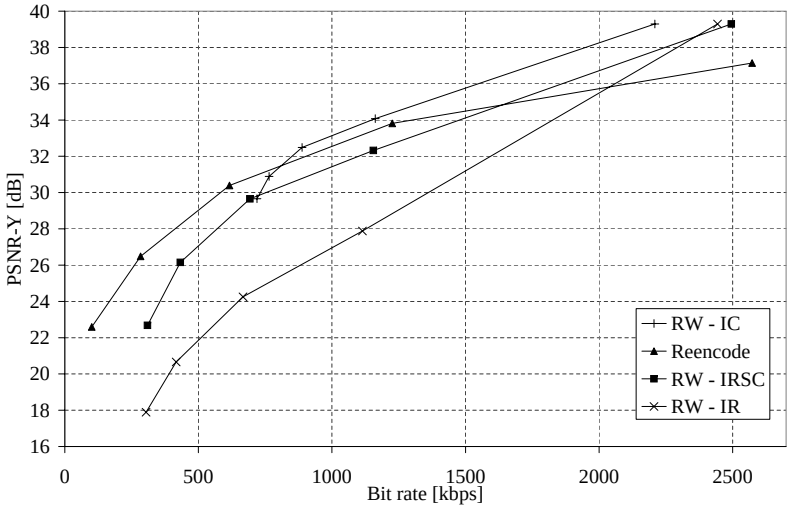
(b) Results for *Stefan* sequence with 5 layers ($\Delta QP = 3$, $QP_I = QP_N = 27$).

Figure 3.16: Results for *Stefan* sequence (CIF, 30 Hz).

used in the plots. The lowest curve for every architecture indicates the situation in which only the lowest layer is decoded. All three architectures obtain identical values when all layers are present. When looking at the four lower curves, the beneficial impact of adding spatial compensation (RW-IRSC) can be seen, while for the RW-IR curve, sharp peaks and falls are obtained, depending on



(c) Results for *Stefan* sequence with 3 layers ($\Delta QP = 6$, $QP_I = QP_N = 32$).



(d) Results for *Stefan* sequence with 5 layers ($\Delta QP = 6$, $QP_I = QP_N = 22$).

Figure 3.16: Results for *Stefan* sequence (CIF, 30 Hz) - continued.

the position in the intra period. This indicates severe drift in the pictures. When spatial compensation is not performed (RW-IR), significant drift will arise throughout the intra period, resulting in rapidly declining PSNR values and degraded rate-distortion performance. The large variability in PSNR for the RW-IR architecture is also reflected in a disturbing flickering effect in the

decoded video stream, which limits the usability of this architecture.

Partitioning all intra data in the lower layer has a beneficial impact on the overall quality, as can be seen from the curves for the intra copy architecture. The plot shows that the quality of the frames for the intra copy architecture remains well above the other plots. As mentioned above, this is partially explained by the fact that the intra-coded pictures are coded with maximum quality, resulting in reliable reference frames for the remainder of the intra period.

From Figure 3.19 the benefit of exploiting the bitstream rewriting functionality becomes clear. A comparison is made between the best performing architecture for both bitstream rewriting disabled (Intra copy architecture) and for rewriting enabled (RW-IC). The results show that transcoding from H.264/AVC to SVC benefits from the bitstream rewriting functionality, not only for the enhancement layer (where perfect reconstruction is achieved), but also for the lower layers. Curves for the RW-IC architecture outperform the Intra copy architecture for all extraction points.

Table 3.2 (3 layers) and Table 3.3 (5 layers) display the overhead induced by the H.264/AVC-to-SVC conversion (as demonstrated for the *Stefan* sequence; similar results are obtained for the other sequences). These values represent the cost of introducing scalability in H.264/AVC bitstreams. The results are shown for the architectures with bitstream rewriting enabled. All these architectures obtain identical PSNR values for the top layer. The overhead is measured as the ratio of the total bit rate of the output SVC stream (with all layers present), compared to the bit rate of the original H.264/AVC stream.

The values show that RW-IC clearly has the lowest overhead, as could also be expected from its superior rate-distortion performance as shown in Figure 3.14 through Figure 3.16. Using RW-IC, scalability can be introduced at a relatively low cost. The overhead increases when more layers are inserted.

Also, it becomes clear that more overhead is created when a value of $\Delta QP = 3$ is used, when compared to $\Delta QP = 6$. This effect can be attributed to the quantization step alignment when a $\Delta QP = 6$ is used (the quantization step size is doubled for an increase of the QP by 6), which leads to efficient coefficient accumulation. This alignment is not present for $\Delta QP = 3$.

3.6.2 Computational complexity analysis - timing results

In Table 3.4, timing results are shown for the different transcoder architectures. These results are obtained from our non-optimized transcoder software, and serve as an indication of the complexity of the techniques and architectures. The processing speed is shown in frames per second. Reencoding is performed

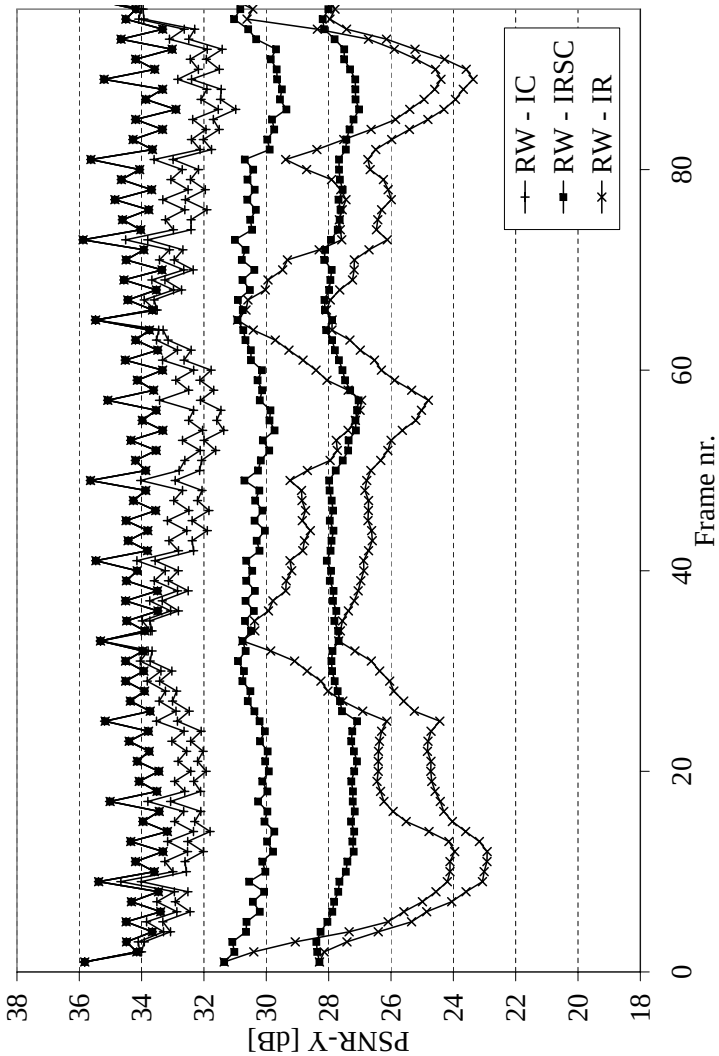


Figure 3.17: *Foreman* sequence, 3 layers, $\Delta QP = 6$, $QP_I = QP_N = 32$ (96 frames).

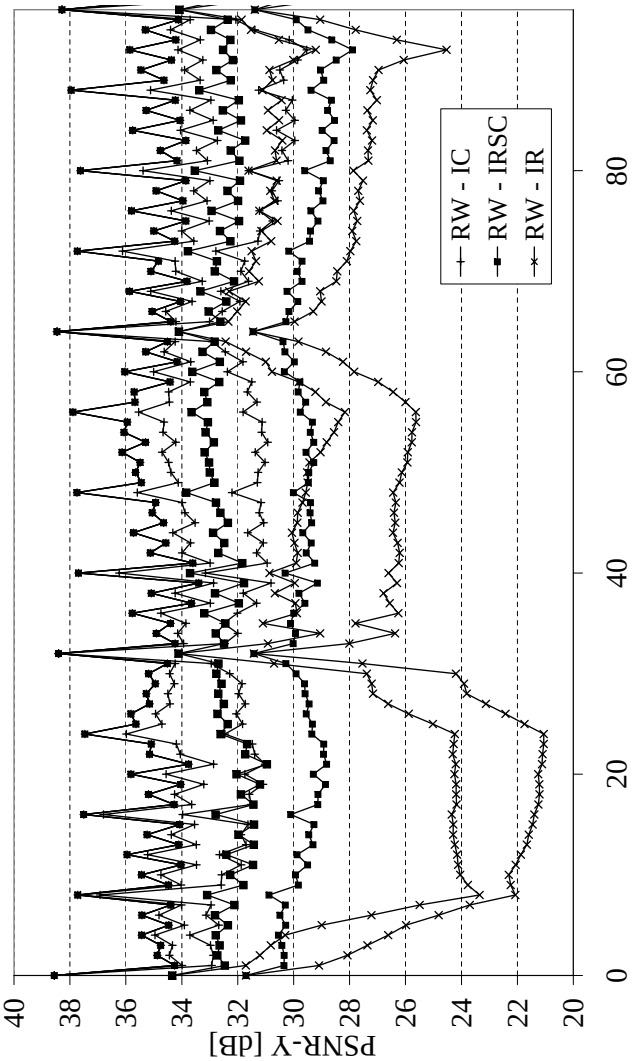
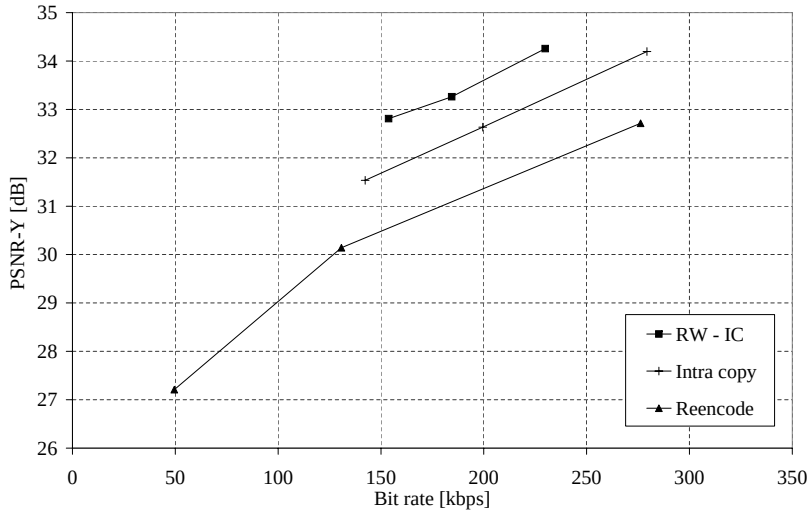
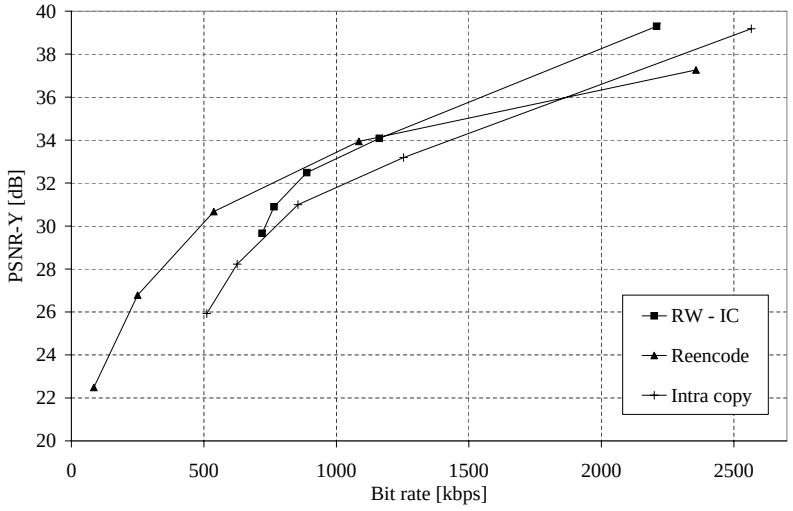


Figure 3.18: *Stefan* sequence, 3 layers, $\Delta QP = 3$, $QP_I = QP_N = 27$ (96 frames).



(a) Results for *Foreman* sequence with 3 layers ($\Delta QP = 6$, $QP_I = QP_N = 32$).



(b) Results for *Stefan* sequence with 5 layers ($\Delta QP = 6$, $QP_I = QP_N = 22$).

Figure 3.19: Comparison of architectures with and without bitstream rewriting functionality.

using the JSVM 9.12 reference software. Note that many optimizations are possible for the reference software as well.

The six transcoding architectures behave similarly regarding processing speed. As can be expected, the RW-IC architecture has lowest overhead, since

Table 3.2: Overhead caused by introducing scalability (*Stefan* sequence, 3 layers, [%]).

	$\Delta QP = 3$				$\Delta QP = 6$			
	QP_I				QP_I			
	22	27	32	37	22	27	32	37
RW-OL	29.3	29.7	33.3	39.0	18.7	19.8	26.1	38.2
RW-IR	31.2	33.1	39.2	46.3	20.4	23.2	31.5	45.5
RW-IRSC	32.9	35.8	43.7	52.0	22.0	25.9	36.0	50.4
RW-IC	19.7	16.7	18.5	26.0	12.0	11.5	16.2	26.0

Table 3.3: Overhead caused by introducing scalability (*Stefan* sequence, 5 layers, [%]).

	$\Delta QP = 3$				$\Delta QP = 6$	
	QP_I				QP_I	
	22	27	32	37	22	27
RW-OL	41.9	45.0	53.2	66.7	24.8	29.1
RW-IRSC	48.5	56.1	71.2	89.4	30.5	38.7
RW-IR	45.4	51.1	63.1	80.5	27.8	34.5
RW-IC	26.6	25.0	32.0	48.8	15.6	18.0

only high-level syntax operations are required for intra-coded macroblocks, while inter-coded macroblocks are transcoded open-loop. RW-OL adds complexity by also requantizing the intra-coded macroblocks, resulting in a slightly lower processing speed. The RW-IR architecture further adds a decoder loop and an encoder loop for every layer for the intra-coded macroblocks. Since this only applies to intra-coded pictures, the additional overhead is restricted when compared to RW-OL. RW-IRSC further incorporates fast single-layer compensation loops for non-constrained intra-coded macroblocks, resulting in only a limited increase of the computational complexity.

Similar conclusions can be drawn for the architectures without bitstream rewriting functionality. It can be seen that for all architectures, computational complexity is only affected to a minor extent, since only spatial prediction loops or high-level syntax operations are used. No temporal compensation is required, thereby avoiding the high complexity of multiple sub-pixel motion-compensated prediction loops. In the presented results, computational complexity is mainly determined by common modules such as entropy decoding,

entropy encoding, parameter derivation, and I/O operations, leading to comparable results within a single column. Adding more layers to the SVC sequences, however, results in a noticeable increase in complexity, due to the added inter-layer prediction steps and the added entropy coding operations.

Table 3.4: Processing speed for transcoding and reencoding [fps]

	Foreman		*Stefan*		*Mobile*	
	3 layers	5 layers	3 layers	5 layers	3 layers	5 layers
Intra BL	20.2	12.9	19.4	12.4	20.1	13.0
Intra copy	21.1	13.5	20.2	12.9	21.1	13.6
RW-OL	22.3	13.7	21.5	13.2	21.2	13.1
RW-IR	21.4	12.9	20.6	12.5	20.2	12.4
RW-IRSC	20.7	12.4	19.8	11.9	20.2	12.3
RW-IC	22.8	14.0	22.0	13.6	21.5	13.3
Reencode	0.06	0.05	0.05	0.05	0.06	0.05

3.7 Motion-refined rewriting

3.7.1 Rationale

As demonstrated in the previous sections, transcoding can be used for introducing scalability in compressed, single-layer bitstreams.

Most existing techniques only focus on residual data transcoding, i.e., without taking into account the motion data in the bitstream. In these schemes, residual data is distributed (or refined) among the different layers, but all motion data is concentrated in the base layer. For larger reductions of the base layer bit rate, however, it becomes beneficial to also adjust the motion parameters, i.e., motion vectors, macroblock partitioning, reference picture indices, etc. This was demonstrated by the experiments for single-layer transcoding in Section 2.5. For the case of H.264/AVC-to-SVC transcoding, coarser motion information will be included in the base layer while enhancement layers contain further refinements of the motion data.

For the complexity reasons mentioned above, decoding (which requires pixel-domain reconstruction) and reencoding (with its included motion estimation process) should be avoided. We start from the fast techniques discussed in the previous sections for residual data rewriting and extend these to include motion vector refinement. As a result, the presented architecture operates completely in the transform domain and avoids the time-consuming

steps involved in decoding and reencoding. Again, adjustment of the motion parameters should be performed in a prudent way, since changed values could lead to misprediction during motion compensation. This could lead to significant distortion and artifacts which could propagate and cause drift in the video stream.

We provide a multi-layer control model that allows to trade off base layer vs. enhancement layer R-D performance. Hence, liberty is provided for the implementation to distribute motion data bits among the most appropriate layer. Multi-layer *encoder* control is currently not included in the JSVM reference encoder software (instead, a bottom-up process is followed). An initial assessment of its beneficial impact on rate-distortion efficiency, however, has been studied in [87]. Here, the concepts are applied to the rewriting case. Although motion-refined encoding was not studied in [87], in this section the concepts are extended to include the more general case of motion refinement.

3.7.2 Motion vector prediction

In the previously discussed rewriting architectures, we only examined residual data rewriting. For these approaches, motion data was copied from the input H.264/AVC stream to the output SVC stream. In the SVC stream, the motion data is concentrated in the base layer, and is identical for all layers. In the following sections, we investigate if refining (or optimizing) the motion data can be used to further improve rate-distortion efficiency in the desired layer(s).

The presented techniques benefit from the motion prediction mechanisms in the SVC design to allow small changes in motion information between the different layers to be coded efficiently in the bitstream. These mechanisms are explained briefly in the remainder of this section.

A. Motion vector prediction in SVC

Since the coding characteristics (such as quality) vary among the SVC layers, it is beneficial to reflect these differences by tuning the motion parameters in every layer. The possibility to update and refine motion information in successive layers is provided in the SVC design. This allows different motion vectors or reference indices to be used for the same macroblock in different layers.

Two separate techniques can be used in SVC to exploit motion information redundancy in the bitstream.

Intra-layer motion prediction Similar to single-layer H.264/AVC, median motion vector prediction can be used in every layer (cf. Section 2.5.1). The

motion vectors of surrounding (sub)macroblock partitions in the same layer are used to form the motion vector prediction.

An important difference, however, occurs for Direct macroblocks in B pictures (i.e., B_Direct_16x16 macroblocks, or B_8x8 macroblocks containing B_Direct_8x8 submacroblocks). In enhancement layers, the derivation for motion vectors in Direct macroblocks is changed when compared to the H.264/AVC Direct mode. This has to be taken into account when a Direct mode is used in the enhancement layer. Under certain conditions, this leads to a recalculation of Direct modes.

Inter-layer motion prediction Although the motion information will not be identical, a lot of similarity will still be found between the different layers. In many cases, this similarity will lead to a more efficient prediction of the motion information. In SVC, inter-layer redundancy between motion information in base and enhancement layers can be exploited by taking the reference layer motion vector as prediction of the current motion vector. As a result, only the difference between both motion vectors needs to be sent in the enhancement layer.

The choice of intra-layer or inter-layer motion vector prediction is indicated by using the *base mode flag* and *motion prediction flags*. When the *base mode flag* is set, all partitioning information, reference indices, and motion vectors are copied from the reference layer to the reconstruction layer, and are reused as such for motion-compensated prediction of the macroblock. If the *base mode flag* is not set, for every macroblock partition, a *motion prediction flag* can be set, which indicates whether a motion vector prediction needs to be formed from the reference layer motion information (i.e., inter-layer motion prediction), or via 'traditional' median prediction (i.e., intra-layer motion prediction).

3.7.3 Motion data rewriting

While the original motion information is optimized for the bit rate of the incoming bitstream (or of the top layer of the outgoing SVC stream), this is not necessarily the case for the lower layers of the output SVC stream. When the quality gap between successive layers becomes larger, rate-distortion efficiency in the lower layers will benefit from a change in motion parameters, similar to the gain shown for single-layer coding in Section 2.5. Akin to the approach used for redistribution of residual data, loss of motion information is avoided. This means that for the top layer, motion information will be identical to that of the incoming bitstream. This also implies that a reduction of motion

data in lower layers will have to be compensated by refinement of the data in the higher layers. The cost of these refinement bits is dictated by the accuracy of the SVC motion prediction mechanisms.

Firstly, we lay out the approach for motion refinement in lower layers. Next, the rate calculation and distortion estimation techniques are discussed that will be used for rate-distortion optimized motion decision in Section 3.7.4. For clarity, the techniques are discussed for two quality layers.

A. Motion refinement

As mentioned in Section 2.5.1, H.264/AVC allows a large degree of flexibility in macroblock partitioning, with (sub)macroblock partitions down to 4×4 pixels. In lower-rate bitstreams, larger block sizes become more dominant, and the amount of submacroblock partitions tends to decrease. This property was exploited in Section 2.5 for motion refinement in single-layer bitstreams. For the lower layers in the SVC streams, macroblock partition merging can be performed in a similar way.

After each possible merge operation, the rate-distortion cost is determined, as will be explained in Section 3.7.4. The rate and distortion are determined as described in the following subsection.

B. Rate calculation and distortion estimation

Different motion-related syntax elements in SVC base and enhancement layer syntax determine the output motion data rate. For the base layer, this corresponds to the elements discussed in Section 2.5.2: the macroblock type and if necessary submacroblock types need to be transmitted, along with reference picture indices and motion vector differences. Here also, if the macroblock is skipped, only a *macroblock skip run* (CAVLC entropy coding) or *macroblock skip flag* (CABAC) needs to be sent (one bit or less per skipped macroblock).

For the enhancement layer, a number of scenarios are possible, depending on the choice for inter-layer or intra-layer motion prediction. In case all motion information of a macroblock can be reused from the base layer, only the *base mode flag* is set and coded in the bitstream. If this is not the case, but a reliable approximation can be formed based on the base layer motion information, *motion prediction flags* can still be used to indicate that the reference indices can be copied from the base layer, and that a predictor can be formed based on the base layer. As an alternative, intra-layer motion vector prediction can be used to achieve the same result, and might result in improved coding efficiency in certain cases.

A trade-off needs to be sought between base layer rate and enhancement layer rate. If loss of information is avoided in the H.264/AVC-to-SVC conversion, a reduction of information in the base layer will have to be counterbalanced by inserting refinement information in the enhancement layer. More information on this trade-off is given in Section 3.7.4.

The distortion can be estimated by using picture power spectrum techniques discussed in Section 2.5.3.

3.7.4 Multi-layer control for H.264/AVC-to-SVC rewriting

The rate and distortion caused by every motion refinement step are obtained using the techniques discussed in Section 3.7.3. As mentioned, a reduction of the rate in a lower layer will lead to an increase of the bit rate in higher layers, leading to a trade-off between the different layers. The decision whether or not the evaluated refinement will be executed depends on the impact of the rate and distortion in every layer. We use a multi-layer control mechanism which attaches a weight factor to every layer. The value of this weight factor depends on the scenario in which the rewriter is used. Based on the weight factors and the rate and distortion costs in every layer, a joint optimization approach is obtained. This multi-layer control mechanism is discussed in the remainder of this section.

For simplicity, the optimization is discussed for two layers, i.e., the base layer (indicated as *layer 1*) and one enhancement layer (*layer 2*). In this case, *base* layer coding decisions are made by minimizing

$$D_1(\boldsymbol{p}_1) + \lambda_1 R_1(\boldsymbol{p}_1) \;, \tag{3.11}$$

where $\boldsymbol{p}_i$ (with $i \in \{1, 2\}$) encompasses the mode decisions $\boldsymbol{m}_i$ and motion vectors $\boldsymbol{v}_i$ for each layer i, respectively. This leads to the well-known functional used for rate-distortion optimized motion evaluation, as used for example in the JSVM encoder software. The Lagrangian multipliers λ_i are derived as in [72].

Additionally, the cost of the *enhancement* layer is taken into account by also minimizing the enhancement layer distortion $D_2(\boldsymbol{p}_2|\boldsymbol{p}_1)$ given the total bit rate $R_1(\boldsymbol{p}_1) + R_2(\boldsymbol{p}_2|\boldsymbol{p}_1)$ [88]. Weighting factor w is used to determine the trade-off between base layer and enhancement layer coding efficiency, leading to the cost functional

$$\min_{\boldsymbol{p}_1, \boldsymbol{p}_2} \big\{ (1-w) \cdot (D_1(\boldsymbol{p}_1) + \lambda_1 R_1(\boldsymbol{p}_1)) + w \cdot (D_2(\boldsymbol{p}_2|\boldsymbol{p}_1) + \lambda_2 (R_1(\boldsymbol{p}_1) + R_2(\boldsymbol{p}_2|\boldsymbol{p}_1))) \big\} \;. \tag{3.12}$$

The case is examined where the motion information becomes identical to the information from the incoming bitstream when all layers are present in the SVC stream, i.e., no quality loss occurs after transcoding when no layers are dropped from the bitstream. This corresponds to a data partitioning scenario.

In this way, the distortion for the enhancement layer is eliminated, i.e., $D_2(\boldsymbol{p}_2|\boldsymbol{p}_1) = 0$, and the minimization problem becomes:

$$\min_{\boldsymbol{p}_1,\boldsymbol{p}_2} \left\{(1-w)\cdot(D_1(\boldsymbol{p}_1)+\lambda_1 R_1(\boldsymbol{p}_1)) + w\cdot\lambda_2(R_1(\boldsymbol{p}_1)+R_2(\boldsymbol{p}_2|\boldsymbol{p}_1))\right\} . \tag{3.13}$$

For $w = 0$, the functional reduces to the case where no joint optimization is performed, i.e.,

$$\min_{\boldsymbol{p}_1} \left\{D_1(\boldsymbol{p}_1) + \lambda_1 R_1(\boldsymbol{p}_1)\right\} \tag{3.14}$$

and only the base layer cost is minimized. In this case, base layer motion refinement will occur more frequently, since the cost of refinement bits is not taken into account. For $w = 1$, the expression

$$\min_{\boldsymbol{p}_1,\boldsymbol{p}_2} \left\{R_1(\boldsymbol{p}_1) + R_2(\boldsymbol{p}_2|\boldsymbol{p}_1)\right\} \tag{3.15}$$

remains, under the side condition that reconstruction is identical when both layers are present in the bitstream. Typically, in this case, the optimum is achieved when all motion data is concentrated in the base layer, de facto corresponding to single-layer coding. Exceptions occur, however, when inter-layer motion prediction outperforms intra-layer median motion vector prediction. In these cases, identical motion data is obtained in the top layer at a reduced total bit rate, i.e., where the scalable stream (locally) outperforms the single-layer bitstream.

3.7.5 Motion-refined rewriting: results and discussion

A. Implementation and setup

The tests for motion-refined rewriting were performed for two layers, i.e., one base layer and one enhancement layer. A similar setup was used as in Section 3.6. The *Foreman*, *Stefan*, *Mobile & Calender*, and *Paris* sequences (CIF resolution) were encoded using the Joint Model (single-layer) reference software, using hierarchical coding. These single-layer bitstreams were transcoded with and without motion refinement. Starting quantization parameters (QP_I) of 22, 27, 32, and 37 were used. In order to avoid loss in residual data, these values were set equal to the quantization parameter QP_2 of the output SVC

enhancement layer, i.e., $QP_2 = QP_I$. For the output base layer, a higher quantization parameter QP_1 is set, i.e., $QP_1 = QP_I + \Delta QP = QP_2 + \Delta QP$. In order to cover typical use cases of SVC streams, ΔQP values of 6 and 12 were used. A GOP length of 8 pictures was used with an intra period of 16. The streams were transcoded to SVC streams with CGS quality scalability.

In the tests, we focus on reduction of motion data in P and B pictures. Nonetheless, rate-distortion results are presented for the overall streams, i.e., including intra-coded pictures without refinement.

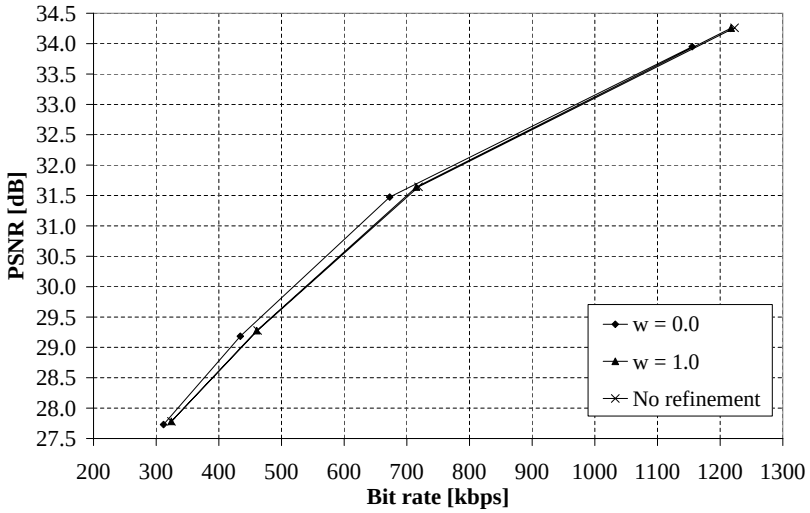
B. Rate-distortion results

In Figure 3.20(a), the rate-distortion results are shown for the base layer of the Stefan sequence, for $\Delta QP = 6$. By setting the enhancement layer weight to one (i.e., $w = 1.0$), the rate-distortion curve practically coincides with the curve without motion refinement. By setting the enhancement layer weight to zero ($w = 0.0$), rate-distortion performance is improved by approximately 5%, in particular in the lower bit rate range. For the highest rate point, a reduction of the bit rate is found (by 5.5%, from 1223 kbps to 1155 kbps) at a marginal gain in rate-distortion performance (the curve is located marginally higher for the higher bit rate range). These results correspond with the theoretical model and illustrate that, although distortion increases somewhat by merging partitions, the motion refinement model only allows a merge if the rate reduction is large enough to improve overall rate-distortion efficiency.

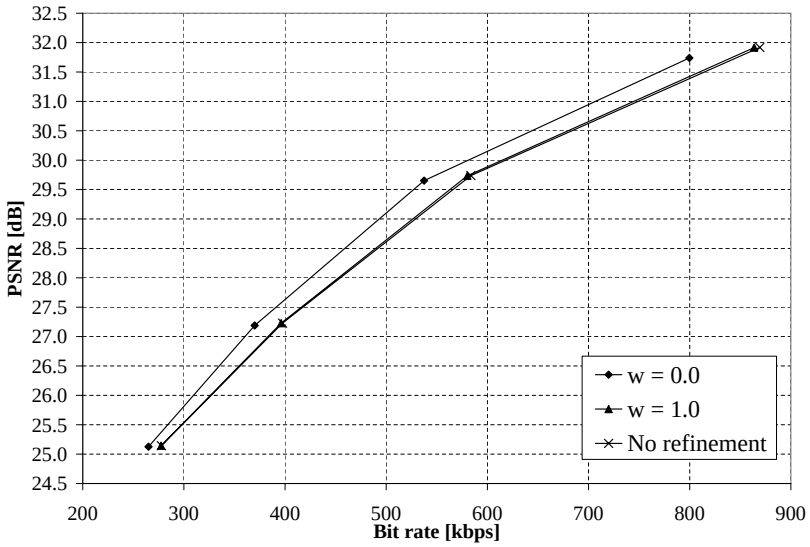
In Figure 3.20(b), the results are shown for the same sequence, but with a $\Delta QP = 12$ between the base and enhancement layer. As could be expected, a larger gap in quantization parameters (resulting in lower base layer bit rates) will lead to a higher degree of refined macroblocks in the stream. This leads to more potential for the motion-refined rewriting architecture, and gains of up to 0.5 dB. Overall bit rates are reduced by 5% for the lower bit rate range to 8% for higher bit rates.

Similar results were obtained for the other sequences. The rate-distortion curves for the *Mobile & Calender* sequence are given in Figure 3.21. The gains were found to be comparable, with bit rate reductions of 6 to 8% for $\Delta QP = 12$.

Results for the top layer are shown in Figure 3.22. Here, the overhead of motion partitioning becomes clear. Note that, since reconstruction is perfect in all cases (when compared to the original single-layer stream), identical PSNR values are obtained for all R-D points at a given QP. Hence, only the corresponding rate values are of interest in these charts. For $w = 1.0$, no overhead is incurred when compared to the case where no refinement is used and both



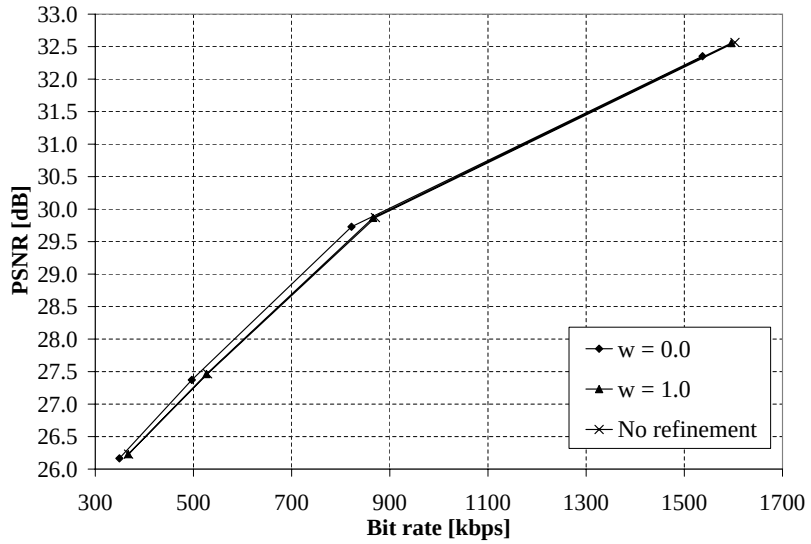
(a) Results for *Stefan* sequence ($\Delta QP = 6$)).



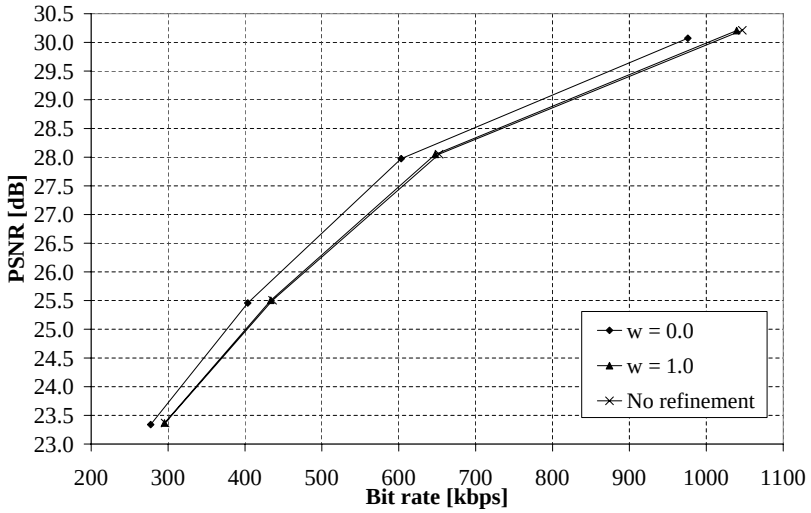
(b) Results for *Stefan* sequence ($\Delta QP = 12$)).

Figure 3.20: Base layer R-D results for *Stefan* sequence ($\Delta QP = 6$ and $\Delta QP = 12$).

curves practically coincide. On the contrary, the total bit rate is even somewhat reduced (but for all sequences $<1\%$). This is caused by cases where inter-layer motion vector prediction is more efficient than regular H.264/AVC intra-layer motion vector prediction. When the weight of the enhancement layer dimin-



(a) Results for *Mobile & Calender* sequence ($\Delta QP = 6$)).



(b) Results for *Mobile & Calender* sequence ($\Delta QP = 12$)).

Figure 3.21: Base layer R-D results for *Mobile & Calender* sequence ($\Delta QP = 6$ and $\Delta QP = 12$).

ishes, the total bit rate will slowly increase, leading to the curves of $w = 0.5$ and $w = 0.0$. This increase in bit rate corresponds with the rate-distortion model, which states that for low values of w, the base layer rate-distortion

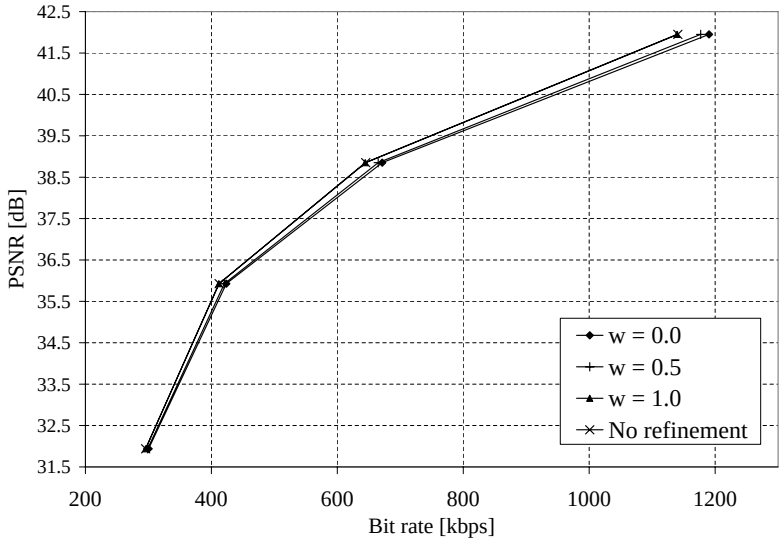
performance behavior is optimized without taking into account the overall bit rate. The more merging operations are performed in the base layer, the more information needs to be injected into the enhancement layer to reconstruct the original motion information. Since this introduces some redundancy in the bitstream (e.g., a macroblock type syntax element needs to be sent in both layers in case of refinement), the overall bit rate will start to increase. The overhead of motion refinement on the overall bit rate, compared to the bit rate without motion refinement, is shown in Table 3.5 for the *Foreman* sequence, for both $\Delta QP = 6$ and $\Delta QP = 12$. The small negative overhead for $w = 1.0$ can be noted in the bottom row of the table.

Table 3.5: Overhead of motion refinement in total bit rate (*Foreman* sequence, [%]).

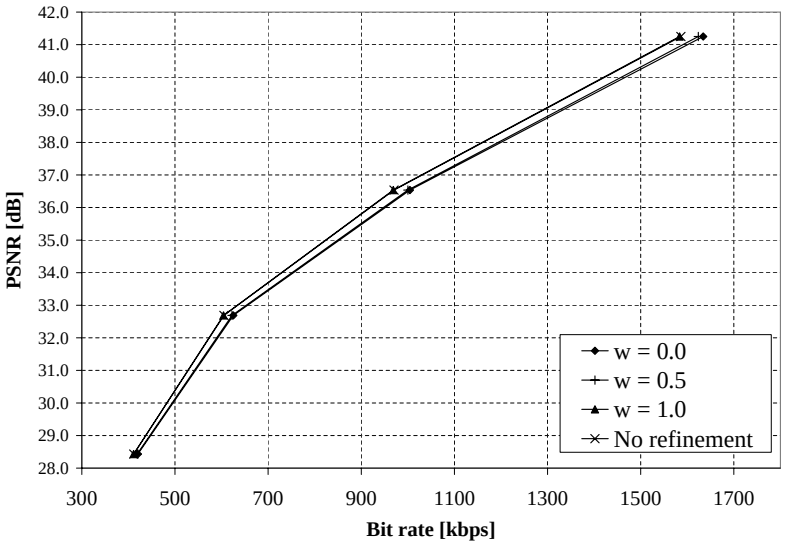
	$\Delta QP = 6$				$\Delta QP = 12$			
	QP_I				QP_I			
w	22	27	32	37	22	27	32	37
0.0	4.34	4.07	2.76	1.59	4.61	4.15	2.80	1.62
0.2	4.12	4.02	2.76	1.59	4.61	4.15	2.80	1.62
0.4	3.67	3.28	2.15	1.32	4.61	4.15	2.80	1.62
0.6	1.92	1.76	1.05	0.68	4.51	4.14	2.80	1.62
0.8	0.13	0.13	0.06	0.08	3.90	3.35	2.04	1.11
1.0	-0.11	-0.12	-0.11	0.01	-0.12	-0.12	-0.12	0.01

3.8 Conclusions and original contributions

In this chapter, we discussed architectures that are able to efficiently transcode single-layer H.264/AVC bitstreams to SVC streams containing multiple quality layers. Different techniques can be applied for each macroblock type for optimum performance. Furthermore, the concept of single-loop decoding has to be taken into account, resulting in the need for appropriate precautions concerning constrained and non-constrained intra-coded macroblocks. We have shown that the changes in the syntax that allow bitstream rewriting can be applied for H.264/AVC-to-SVC transcoding, resulting in flexible architectures. For these architectures, perfect reconstruction is achieved when all layers are available in the decoded bitstream, in contrast to the architectures that do not use the bitstream rewriting functionality. The intra copy architectures were shown to outperform reencoding for limited reductions in bit rate. When lower bit rate extraction points are required, quality declines. The RW-IRSC archi-



(a) Results for *Foreman* sequence ($\Delta QP = 6$)).



(b) Results for *Paris* sequence ($\Delta QP = 12$)).

Figure 3.22: Top-layer R-D results for *Foreman* and *Paris* sequences.

tecture provides a better compromise between rate distribution flexibility and rate-distortion performance. In particular, the addition of spatial drift compensation (when compared to the RW-IR architecture) results in a large improvement in rate-distortion performance. Owing to the absence of MCP loops in

the design of the proposed architectures, complexity is kept low, and sequences can be transcoded at a fraction of the time needed for reencoding using the JSVM reference software.

Additionally, we presented techniques for motion-refined rewriting of H.264/AVC streams to SVC. A multi-layer transcoder control algorithm was introduced that provides a trade-off in rate and distortion between the considered layers. By setting the weight factors appropriately, the model allows rate-distortion performance to be improved for the desired layer(s). Even though operations are performed entirely in the transform domain, it was shown that distortion caused by motion refinement is accurately taken into account in the model. Although additional distortion is introduced due to changes in the motion data, the presented approach intelligently decides whether or not refinement in the motion data should occur, leading to an improvement in rate-distortion performance. In the implementation results, gains of up to 0.5 dB were obtained for the base layer.

The author's work on H.264/AVC-to-SVC transcoding led to the following publications:

- Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Architectures for fast transcoding of H.264/AVC to quality-scalable SVC streams. *IEEE Transactions on Multimedia*. 2009. November 2009.
- Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Motion-refined rewriting of H.264/AVC-coded video to SVC streams. Submitted to *Elsevier Journal of Visual Communication and Image Representation*.
- Jan De Cock, Stijn Notebaert, Kenneth Vermeirsch, Peter Lambert, and Rik Van de Walle. Transcoding of H.264/AVC to SVC with Motion Data Refinement. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. November 2009.
- Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Advanced bitstream rewriting from H.264/AVC to SVC. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. October 2008.
- Jan De Cock, Stijn Notebaert, and Rik Van de Walle. Transcoding from H.264/AVC to SVC with CGS layers. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. September 2007.

Chapter 4

Spatial resolution reduction for H.264/AVC

4.1 Rationale and related work

In heterogeneous multimedia environments, it is beneficial to adjust the resolution of video streams to the display capabilities of the receiving devices. While content is created often in a single, e.g., high-resolution format, the video streams are displayed on a large number of devices with varying characteristics such as display resolution, processing power, or battery life. Reducing the spatial resolution of the video stream can tailor the video to the needs and capabilities of these devices. Additionally, a reduction of spatial resolution induces a reduction of the bit rate of the video stream, hereby limiting network bandwidth and storage requirements.

While pixel-based resolution reduction can be used, i.e., decoding, pixel-domain downscaling, and subsequently reencoding, other solutions are desired which limit the computational complexity or memory size requirements. As in the previous chapters, transcoding is used as a means to speed up the conversion process, by reprocessing information contained in the original bitstream such as motion vectors, prediction modes, and residual data. Techniques for previous video coding standards such as MPEG-1 or MPEG-2 have been examined in e.g., [6,89,90]. In this chapter, we discuss efficient spatial resolution reduction transcoding for H.264/AVC. When compared to previous video coding standards, a number of issues arise that require closer attention in order to obtain high-quality transcoded video sequences.

For MPEG-2, low-complexity frequency-domain techniques for spatial resolution reduction are typically used. These techniques are based on the synthesis of reduced-resolution blocks using the low-order frequency components

of the original DCT blocks. Arbitrary downsizing in the transform domain can be accomplished in this way, as described for example in [91–93].

In [94], a study was made of the drift during reduced-resolution decoding of MPEG-2 sequences. A minimal-drift decoder was described that eliminates the motion vector cause of the drift, and achieves the theoretically minimum drift caused by DCT downsampling. Although the proposed decoder substantially improves the drift, the quality of the decoded sequences rapidly declines over a number of pictures.

Different reduced spatial resolution transcoding architectures were assessed in [95], where the authors examined the problem of transcoding from MPEG-2 to MPEG-4 Visual with a focus on temporal drift compensation. Drift due to motion vector misalignment during downscaling has been discussed for example in [96] for MPEG-2 video transcoding. These architectures were shown to result in a minor quality loss when compared to reencoding.

More recently, spatial resolution transcoding was examined for H.264/AVC. In [35, 97, 98], the problem of mode decision was elaborated on, to provide a speed-up in cascaded decoder-encoder architectures. In previous publications, these cascaded architectures have been the primary focus of spatial resolution transcoding for H.264/AVC. Due to their double-loop nature, however, complexity and buffer requirements remain high when compared to fast open-loop or single-loop transcoding architectures, such as those developed for previous video coding standards [5].

So far, little has been written on reduced-complexity architectures for downscaling in H.264/AVC. A number of improvements of H.264/AVC video coding, such as submacroblock partitioning (with partition sizes down to 4×4 pixels) and the use of multiple reference pictures, introduce new challenges when developing fast techniques for H.264/AVC video stream downsizing and prohibit straightforward application of previously existing techniques. In this chapter, we highlight and tackle these new problems, with a focus on the case of dyadic downscaling. Open-loop techniques for arbitrary downscaling of H.264/AVC have been discussed in [99]. As will be shown in Section 4.3.1, however, the applicability of these open-loop techniques is restrained by a number of features of H.264/AVC. Only when several constraints are imposed on the input bitstream can these techniques lead to acceptable results. Without proper measures, significant artifacts and drift arise in the video stream. After a discussion of the advanced coding tools that cause limitations for H.264/AVC, we introduce a fast architecture that tackles the identified issues in Section 4.3.2. The problems related to these coding tools seem to be neglected (or avoided) in several publications. For practical transcoding systems, however, these issues cannot be ignored and need a deeper investigation. Un-

avoidably, mismatches will result in quality degradation; the question is how the building blocks of the transcoding architecture, such as for integer transform downsampling and the interpolation filters, should be designed in order to result in minimal drift and acceptable quality.

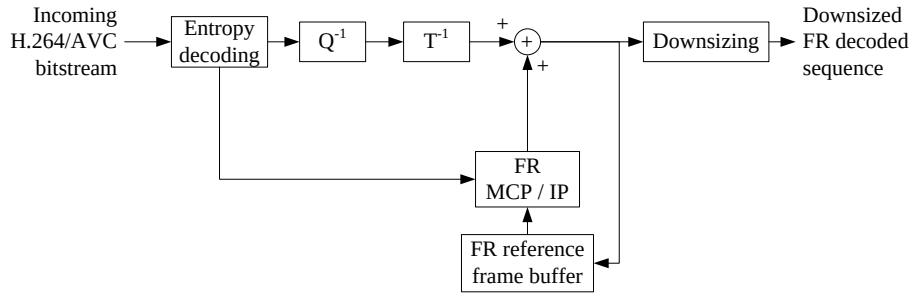
In order to lay a basis for the proposed spatial transcoding architecture in this chapter, we start by discussing reduced-resolution *decoding*. Reduced-resolution decoding encounters a subset of the problems related to spatial transcoding, and will lead to a better understanding of the issues at hand, such as the handling of motion vectors, residual data, interpolation filters, and intra prediction.

The remainder of this chapter is organized as follows. In Section 4.2, we proceed by examining techniques for reduced-resolution decoding. In Section 4.3.1, mapping-related problems are described that are encountered when downscaling H.264/AVC streams. In Section 4.3.2, we lay out a novel architecture for spatial resolution reduction transcoding. Section 4.3.3 and Section 4.3.4 give more details on the mode and motion data mapping processes, and the residual data conversion process, respectively. In Section 4.3.5, implementation results are provided for the presented architecture. Finally, this chapter is concluded in Section 4.4.

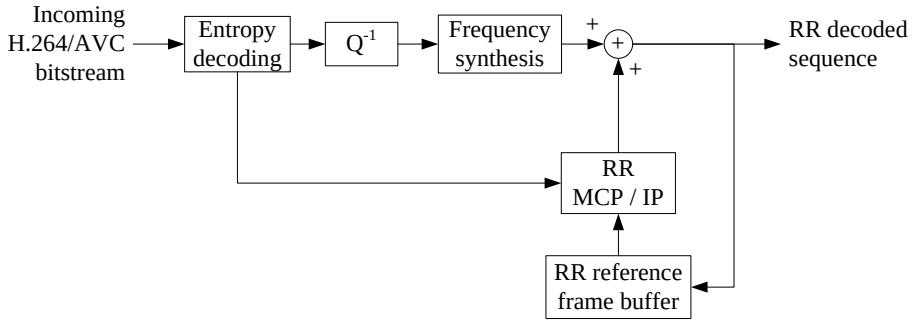
4.2 Reduced-resolution decoding for H.264/AVC

With the increasing adoption of high-resolution content (e.g., 720p, 1080i, and 1080p which are becoming mainstream), and the advent of even higher-resolution video, such as ultra high definition video (UHDV, 7680x4320 pixels) [100], the need for solutions that provide backward compatibility with existing display devices increases. The chain of decoding, pixel-domain downsizing, and subsequently displaying could be used to accomplish the downscaling, but would require excessive buffer use. With the use of H.264/AVC and multiple reference picture MCP, this would put a heavy burden on the memory requirements of the display or converter device. The question is whether efficient algorithms can be constructed that allow reduced-resolution (RR) decoding of such content. In this section, the quality, complexity, and practicality of reduced-resolution decoders are evaluated. We investigate the algorithmic modules needed to construct such a reduced-resolution decoding system in H.264/AVC.

The full-resolution H.264/AVC decoder with subsequent subsampling is shown in Figure 4.1(a). A full-resolution multiple-reference frame buffer is maintained, from which the values are fetched to execute motion-compensated prediction or intra prediction in the full-resolution pixel domain.



(a) Full-resolution H.264/AVC decoder with subsequent subsampling



(b) Reduced-resolution H.264/AVC decoder

Figure 4.1: Full-resolution vs. reduced-resolution H.264/AVC decoding.

The FR decoder is compared with the RR decoder, of which an example is shown in Figure 4.1(b). The frame buffer contains downsized reference frames, and motion-compensated prediction is performed in the reduced-resolution domain. In H.264/AVC, intra prediction introduces additional difficulties due to the absence of full-resolution prediction pixels of neighboring macroblocks. In Figure 4.1(b), frequency synthesis is used to handle the conversion of residual data to the reduced resolution. This technique was already used for MPEG-2 downconversion, and can be adapted for use in H.264/AVC, as explained in the next section.

Overall, three major issues are present for reduced-resolution decoding. Firstly, techniques (such as frequency synthesis) are needed for residual data downscaling with minimal loss of quality. Secondly, motion-compensated prediction in the reduced-resolution domain requires attention because of the mismatch in motion vector accuracy and the need for adjusted interpolation in the reduced resolution. Thirdly, intra prediction poses an additional challenge which is new in H.264/AVC. These issues are addressed in the following sec-

tions and new solutions are provided.

4.2.1 Residual data downsizing

A. 2×2 **cut**

Several approaches have been proposed in literature for downconversion of texture information. A first approach is to cut out the low-frequency components of each transform block, after which a reduced-size inverse transform is applied. For MPEG-2, this gave rise to the 4×4 *cut* technique. From each 8×8 DCT transform block, the low-frequency 4×4 block of coefficients is extracted. After inverse 4×4 DCT transform, the residual values are obtained.

In H.264/AVC, we propose to use an analogous technique that cuts out 2×2 blocks from the 4×4 blocks of integer transform coefficients. As will be seen further on, the loss of the 2×2 cut when compared to the more complex frequency synthesis technique is negligible. For High profile 8×8 integer blocks, a similar 4×4 cut can be devised; in the remainder of this chapter, however, the focus is on 4×4 integer transform blocks.

A downside of the MPEG-2 4×4 cut or H.264/AVC 2×2 cut technique is that the high-frequency information in the individual blocks is lost. This is illustrated in Figure 4.2, where only the information indicated in gray is retained in the output block.

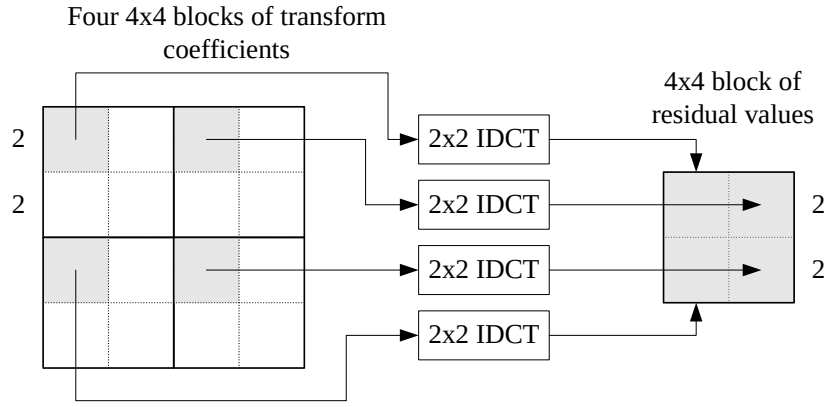


Figure 4.2: 2x2 cut in H.264/AVC.

As a first step, position-independent inverse quantization is performed. The inverse quantization formula is obtained by removing the 4×4 normalization values $E_{I,ij}$ from the multiplier coefficients V_{ij}. Here also, the values

V' of Table 2.3 are obtained by calculating:

$$V' = 16 \cdot Q_{\text{step}} \ . \tag{4.1}$$

Subsequently, the 2×2 DCT matrix can be used for inverse transformation of each cut-out 2×2 block:

$$\boldsymbol{D}_2 = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix} \ , \tag{4.2}$$

which can be implemented by using the 2×2 Hadamard matrix

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.3}$$

and by postponing multiplication by the constant factor $(\sqrt{2}/2)^2$. An additional renormalization factor of $1/2$ is finally applied to compensate for the difference in normalization between the 2×2 and 4×4 DCT transforms. This leads to the 2×2 blocks of residual coefficients, which can be grouped by four, and used during reduced-resolution motion-compensated prediction.

B. Frequency synthesis

As mentioned, a downside of using the 2×2 cut technique is that the high-frequency information is simply discarded during the downconversion. In [20, 21], frequency synthesis was used for downconversion. In frequency synthesis, four 8×8 blocks of a macroblock (in the case of MPEG-2) are subject to a global transformation. In this way, a single frequency domain block is realized using information in the entire macroblock. Using frequency synthesis, more of the block energy is captured and the frequency content of the block is represented more accurately. From the synthesized block, the low-order frequency components are cut out. Similarly, frequency synthesis in H.264/AVC can be realized by capturing the energy from four 4×4 blocks of coefficients, from which an 8×8 block of DCT coefficients $\boldsymbol{A}'_{k,l}$ is derived. The resulting 4×4 block of coefficients $\tilde{\boldsymbol{a}}'_{k,l}$ is derived by capturing the low-frequency coefficients from this 8×8 block and performing a 4×4 inverse DCT. This is illustrated in Figure 4.3.

In H.264/AVC, the residual values are obtained by converting the H.264/AVC integer transform blocks to DCT blocks, subsequently discarding high frequency DCT coefficients, and applying an inverse DCT to the resulting blocks. This is expressed as follows. At first, the inverse integer transform is

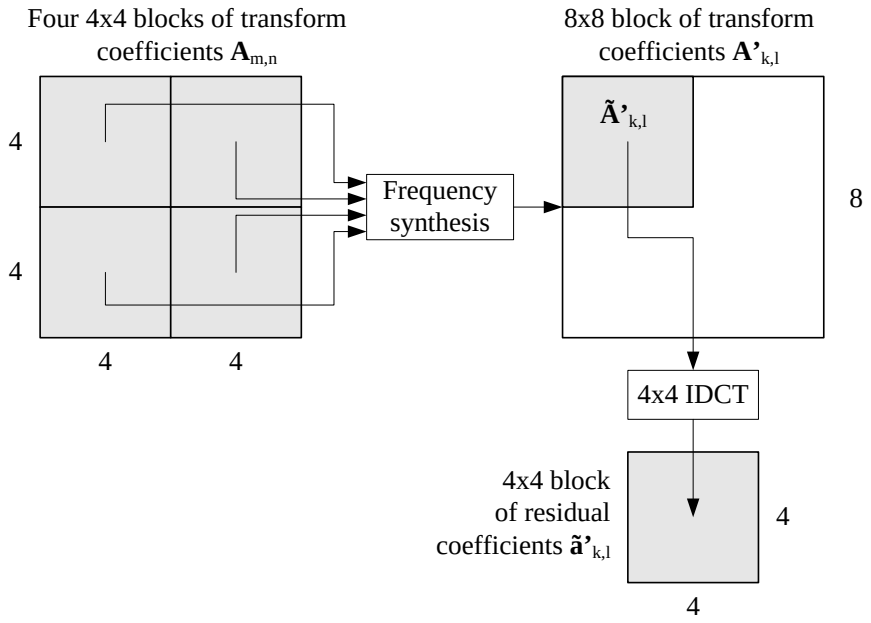


Figure 4.3: Frequency synthesis in H.264/AVC.

applied to the 4×4 blocks $\boldsymbol{A}_{m,n}$, with m and n ranging over the dimensions in 4×4 blocks of the frame:

$$\boldsymbol{a}_{m,n} = \boldsymbol{C}_I^T \; \boldsymbol{A}_{m,n} \; \boldsymbol{C}_I \; . \tag{4.4}$$

Here, the core inverse transform matrix $\boldsymbol{C}_I$ is defined as in Section 2.2.1. A traditional forward 8×8 DCT transform is applied to a group of four 4×4 blocks

$$\boldsymbol{A}'_{k,l} = \boldsymbol{D}_8 \left[\begin{array}{cc} \boldsymbol{a}_{2k,2l} & \boldsymbol{a}_{2k,2l+1} \\ \boldsymbol{a}_{2k+1,2l} & \boldsymbol{a}_{2k+1,2l+1} \end{array} \right] \boldsymbol{D}_8^T \; . \tag{4.5}$$

From this 8×8 block, the high-frequency components are discarded, resulting in the 4×4 DCT blocks

$$\tilde{\boldsymbol{A}}'_{k,l} = \boldsymbol{L}_{4\times8} \; \boldsymbol{A}'_{k,l} \; \boldsymbol{L}_{4\times8}^T \tag{4.6}$$

with $\boldsymbol{L}_{4\times8} = \left[\begin{array}{cc} \boldsymbol{I}_4 & \boldsymbol{0} \end{array} \right]$. The result is inverse DCT transformed:

$$\tilde{\boldsymbol{a}}'_{k,l} = \boldsymbol{D}_4^T \; \tilde{\boldsymbol{A}}'_{k,l} \; \boldsymbol{D}_4 \; . \tag{4.7}$$

The downsized residual signal $\tilde{\boldsymbol{a}}'_{k,l}$ is added to the reduced-resolution MCP signal, as derived in the following section.

A similar approach for arbitrary resizing was discussed in [99], which allows the target DCT frame to be constructed as a whole from the 4×4 integer transform blocks in the original frame. The resizing operation was represented as a multiplication using fixed matrices. This approach, however, only works when a number of simplifications are used, such as the absence of intra-coded macroblocks in P and B pictures.

4.2.2 Motion vector derivation for reduced-resolution MCP

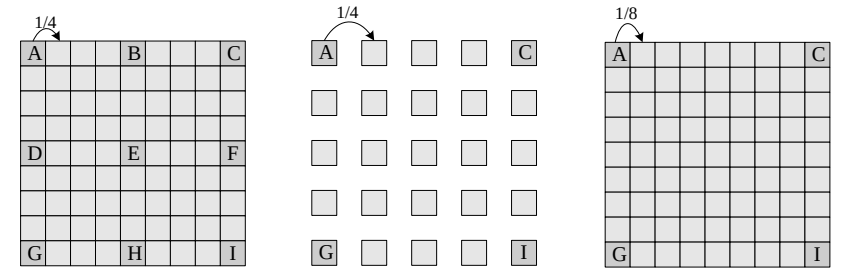
When decoding at a reduced resolution, an important issue is how to adjust the available motion information from the input bitstream to the desired output resolution. Motion vectors have 1/4-pel accuracy in H.264/AVC, which corresponds to 1/8-pel accuracy for the reduced resolution in the case of dyadic downscaling. Different strategies can be used for MCP in the reduced resolution, with or without loss of motion vector accuracy. The impact of the different strategies on the reconstruction is discussed in the remainder of this section.

In the case of motion vector truncation, both components of the motion vector are simply divided by two and used as such for the reduced resolution. A full-pixel displacement of ± 1 is converted to a half-pixel displacement of $\pm 1/2$, a half-pixel displacement of $\pm 1/2$ is converted to a quarter-pixel displacement of $\pm 1/4$. An original quarter-pixel displacement of $\pm 1/4$ will be lost. Loss of information is unavoidably introduced by the truncation process and the MCP prediction signal will show additional distortion when compared to full-resolution interpolation.

When using motion vector truncation with quarter-pixel accuracy, MCP can for example be executed using the 'default' H.264/AVC interpolation process, applied to the reduced-resolution reference picture: interpolation based on a 6-tap Wiener filter is used to obtain the half-pixel values, followed by a bilinear interpolation filter to generate the quarter-pixel values.

The effect of reduced-resolution interpolation after motion vector truncation is illustrated in Figure 4.4(b). It is clear that in this manner accuracy is lost when compared to the full-resolution sub-pixel interpolation (Figure 4.4(a)).

Since motion vector truncation leads to additional distortion, we examine the benefit of using eighth-pixel motion interpolation for the reduced resolution, so the same granularity can be achieved for both resolutions. In the context of coding efficiency improvement, the use of 1/8-pel displacement vectors was examined in [60] and compared with the standardized quarter-pixel interpolation process of H.264/AVC. Interpolation is performed in three steps (instead of the two-step process for quarter-pel interpolation in H.264/AVC).



(a) Full-resolution subpixel interpolation

(b) Reduced-resolution subpixel interpolation (truncation)

(c) Reduced-resolution subpixel interpolation (1/8-pel)

Figure 4.4: Full-resolution vs. reduced-resolution subpixel interpolation.

The result of the interpolation process is shown in Figure 4.4(c). As a result, the same positions are reached as for the full-resolution interpolation process.

The three-step interpolation process is illustrated in Figure 4.5. The question remains which filters h_1, h_2, and h_3 are best suited to reach the sub-pixel values for half-, quarter-, and even eighth-pixel positions.

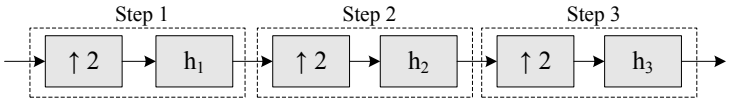


Figure 4.5: Three-step 1/8-pel interpolation.

4.2.3 Interpolation filters for reduced-resolution MCP

A. Evaluated filter combinations

Different filter combinations of (h_1, h_2, h_3) can be used to obtain the half-pel, 1/4-pel, and 1/8-pel values, respectively (cf. Figure 4.5). As argumented in [60], the quality of the overall interpolation process is mainly determined by the selection of the h_1 filter. For this reason, a more complex filter is typically selected to obtain the half-pel values. We evaluated a number of possibilities for the h_1, h_2, and h_3 filters; the filter tap length of h_2 and h_3 is less than or equal to the tap length of h_1 in all cases.

As 8-tap filter for half-pel interpolation, the 8-tap Wiener filter was used. This filter was chosen in [60] for half-pixel interpolation, and was selected for its lower stopband permeability over the H.264/AVC 6-tap Wiener filter.

As 6-tap filters, the 6-tap Wiener filter (used for h_1, similar to H.264/AVC) and Hamming-windowed Sinc filter (used for h_2, as in [60]) were evaluated. Furthermore, the 'Telenor 4-tap filter' with filter tap coefficients $(-4, 20, 20, -4)/32$ was used [101], along with a bilinear filter. To summarize, the following filters and filter coefficients were used:

Table 4.1: Coefficients for interpolation filters h_1, h_2, and h_3.

Filter	Coefficients
8-tap Wiener	(-1,3,-6,20,20,-6,3,-1)/32
6-tap Wiener (h_1)	(1,-5,20,20,-5,1)/32
6-tap Hamming (h_2)	(2,-21,147,147,-21,2)/256
4-tap 'Telenor'	(-4,20,20,-4)/32
2-tap	(16,16)/32

The tested combinations of filters are summarized by listing the number of taps for each filter. The resulting tuples (for 1/4-pel interpolation) and triples (for 1/8-pel interpolation) are shown in Table 4.2, and correspond to the filter combinations tested in [101].

Table 4.2: Evaluated interpolation filter combinations of (h_1, h_2, h_3).

1/8-pel	1/4-pel
(882)	(82)
(862)	(62)
(842)	(42)
(662)	
(642)	
(442)	

B. Optimized filters

Besides using interpolation filters investigated for use in H.264/AVC, we examined the use of optimized filters for motion-compensated prediction in the reduced-resolution domain. These reduced-resolution MCP interpolation filters are optimized for the used downsampling filter in a least squares sense. For MPEG-2, the selection of optimized filters is described in [21, 102]. Limited results, however, were reported for this technique. MSE values were provided that show drift, even over the course of a limited number of predictive pictures. A considerable improvement, however, is obtained over bilinear interpolation.

The question is whether optimized filters can be used in an H.264/AVC context and whether they will result in improved visual quality over, e.g., the 8-tap Wiener filter.

The optimized filters are calculated by comparing full-resolution MCP followed by downconversion on the one hand with downconversion followed by reduced-resolution MCP on the other hand. Here, filters are needed for the latter case. This is illustrated in Figure 4.6. In particular, optimized reduced-resolution downsizing filter matrices $\boldsymbol{N}_i$ ($i = 1, \ldots, 4$) are sought that minimize the difference between $\tilde{\boldsymbol{g}}$ and $\hat{\tilde{\boldsymbol{g}}}$, i.e., the output blocks obtained by the two processes:

$$\arg\min_{\boldsymbol{N}_i} \left\| \tilde{\boldsymbol{g}} - \hat{\tilde{\boldsymbol{g}}} \right\|^2 . \tag{4.8}$$

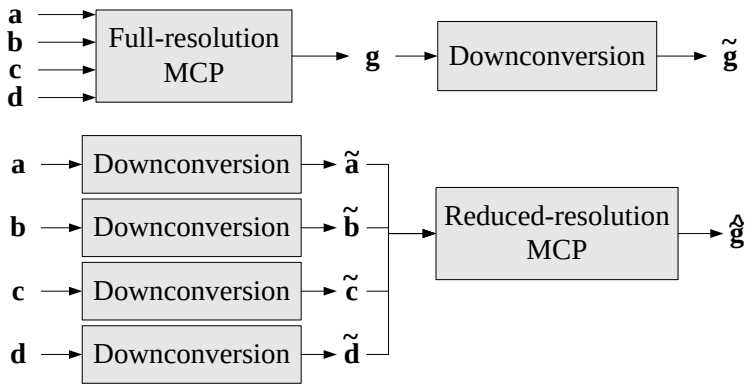
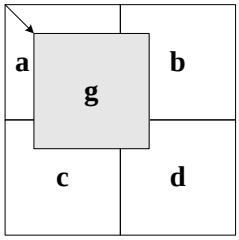


Figure 4.6: Construction of output blocks $\tilde{\boldsymbol{g}}$ and $\hat{\tilde{\boldsymbol{g}}}$.

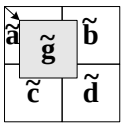
The full-resolution and reduced-resolution MCP processes, along with the corresponding notation, are demonstrated in Figure 4.7(a) and Figure 4.7(b), respectively.

Full-resolution MCP For full-resolution MCP, the following operations are performed. To simplify the operations, the blocks $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ are converted to one-dimensional vectors by appropriate scanning, resulting in 64×1 vectors. Firstly, MCP is applied to these vectors:

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \boldsymbol{g}_3 \\ \boldsymbol{g}_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_a & \boldsymbol{S}_b & \boldsymbol{S}_c & \boldsymbol{S}_d \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} . \tag{4.9}$$



(a) Full-resolution MCP of block $\boldsymbol{g}$ based on full-resolution blocks $\boldsymbol{a}$ through $\boldsymbol{d}$.



(b) Reduced-resolution MCP of block $\tilde{\boldsymbol{g}}$ based on reduced-resolution blocks $\tilde{\boldsymbol{a}}$ through $\tilde{\boldsymbol{d}}$.

Figure 4.7: Comparison between FR MCP (resulting in block $\boldsymbol{g}$) and RR MCP (resulting in block $\tilde{\boldsymbol{g}}$).

The 64×64 prediction matrices $\boldsymbol{S}_a$ through $\boldsymbol{S}_d$ are derived based on the overlap of blocks $\boldsymbol{a}$ to $\boldsymbol{d}$, and are dependent on the used motion vector. The coefficients of these sparse vectors are dependent on whether full-pixel, half-pixel, or quarter-pixel displacement is used. The resulting prediction block $\boldsymbol{g}$ (represented by a 64×1 vector) is downsized, e.g., using the frequency synthesis technique discussed above, leading to 16×1 vector $\tilde{\boldsymbol{g}}$ (the downconversion is represented by multiplication with matrix $\boldsymbol{X}$):

$$\tilde{\boldsymbol{g}} = \boldsymbol{X}\ \boldsymbol{g}\ . \tag{4.10}$$

Reduced-resolution MCP For reduced-resolution MCP, prediction is performed on downsized versions of blocks $\boldsymbol{a}$ to $\boldsymbol{d}$, denoted as $\tilde{\boldsymbol{a}}$ to $\tilde{\boldsymbol{d}}$ (16×1 vectors). Subsequently, MCP with optimized 16×16 filter matrices $\boldsymbol{N}_i$ (with i ranging from 1 to 4) is executed on these blocks:

$$\hat{\tilde{\boldsymbol{g}}} = \left[\ \boldsymbol{N}_1 \quad \boldsymbol{N}_2 \quad \boldsymbol{N}_3 \quad \boldsymbol{N}_4\ \right] \begin{bmatrix} \tilde{\boldsymbol{a}} \\ \tilde{\boldsymbol{b}} \\ \tilde{\boldsymbol{c}} \\ \tilde{\boldsymbol{d}} \end{bmatrix}\ . \tag{4.11}$$

The filters $\boldsymbol{N}_i$ can be derived by minimizing Equation (4.8), resulting in

$$\boldsymbol{N}_1 = \boldsymbol{X}\boldsymbol{S}_a\boldsymbol{X}^+ \tag{4.12}$$
$$\boldsymbol{N}_2 = \boldsymbol{X}\boldsymbol{S}_b\boldsymbol{X}^+ \tag{4.13}$$
$$\boldsymbol{N}_3 = \boldsymbol{X}\boldsymbol{S}_c\boldsymbol{X}^+ \tag{4.14}$$
$$\boldsymbol{N}_4 = \boldsymbol{X}\boldsymbol{S}_d\boldsymbol{X}^+ \tag{4.15}$$

where $\boldsymbol{X}^+$ is the (Moore-Penrose) pseudo inverse of $\boldsymbol{X}$:

$$\boldsymbol{X}^+ = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1} \; . \tag{4.16}$$

4.2.4 Reduced-resolution intra prediction

Although techniques for MCP blocks can be inspired on algorithms developed for previous video coding standards, intra-coded macroblocks (again) introduce new difficulties in H.264/AVC. Since intra-coded macroblocks are allowed in P and B pictures, provisions need to be made for spatial resolution reduction of typical H.264/AVC video streams.

Given the availability of the reduced-size reconstructed pictures, it is possible to reconstruct intra-coded macroblocks. For optimum quality results, the intra prediction is performed in the *original* (FR) resolution. In a first step, the surrounding prediction pixels that are used to form the 4×4 or 16×16 prediction are obtained. Two options can occur:

- The prediction pixels belong to an intra-coded macroblock (the current or a neighboring macroblock). In this case, the full-resolution prediction pixels can be used. Although this requires a full-resolution buffer, its size can remain limited, given that only the edge pixels need to be stored for further prediction of neighboring macroblocks (31 pixels at most).

- The prediction values belong to an MCP macroblock. In this case, no full-resolution values are available. Hence, the prediction pixels are upscaled (for low complexity, this was done using bilinear upsampling). This case is illustrated in Figure 4.8.

After forming the H.264/AVC intra prediction signal in the full-resolution domain, the incoming residual data is added. By following this approach, the internal detail is preserved during decoding. After intra decoding, the result is downscaled as is done for intra-coded pictures, resulting in the reduced-resolution intra-coded macroblocks.

Given the relatively low complexity of intra prediction (in particular when compared to motion-compensated prediction), the impact on the overall complexity is kept low using this approach.

The results of this approach are demonstrated in Figure 4.9. In Figure 4.9(a), the reduced-resolution (RR) picture is shown when only the MCP macroblocks have been decoded in the reduced resolution. The intra-coded macroblocks are shown in black. In the second step, the pixels from the first step are used to form the prediction signal for intra-coded macroblocks. The result after intra decoding is shown in Figure 4.9(b) (*Foreman*, CIF to QCIF, frame 3 (coded as P picture)).

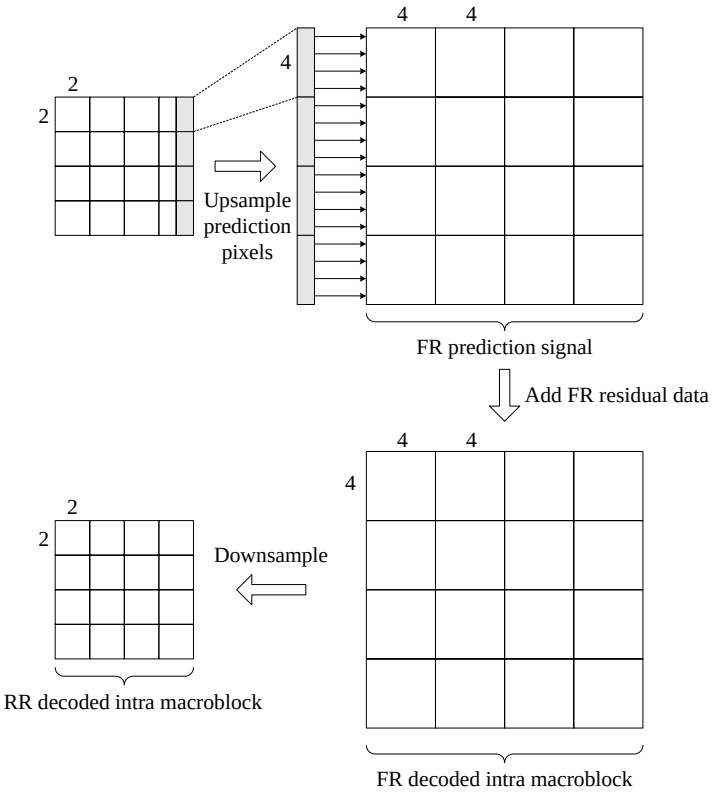


Figure 4.8: Reduced-resolution intra prediction.

(a) Step 1: reduced-resolution MCP decoding

(b) Step 2: add reduced-resolution intra-coded macroblocks

Figure 4.9: Reduced-resolution intra prediction (*Foreman*, CIF to QCIF).

4.2.5 Results and discussion

Reduced-resolution decoding was evaluated for the *Akiyo*, *Paris*, and *Foreman* sequences (original: CIF resolution), and *Harbour* and *Soccer* sequences (original: 4CIF resolution). A closed IBBP GOP structure was used, with a GOP length of 13 pictures. The sequences were originally encoded using the Joint Model reference software, with quantization parameter values of 22, 27, 32, and 37.

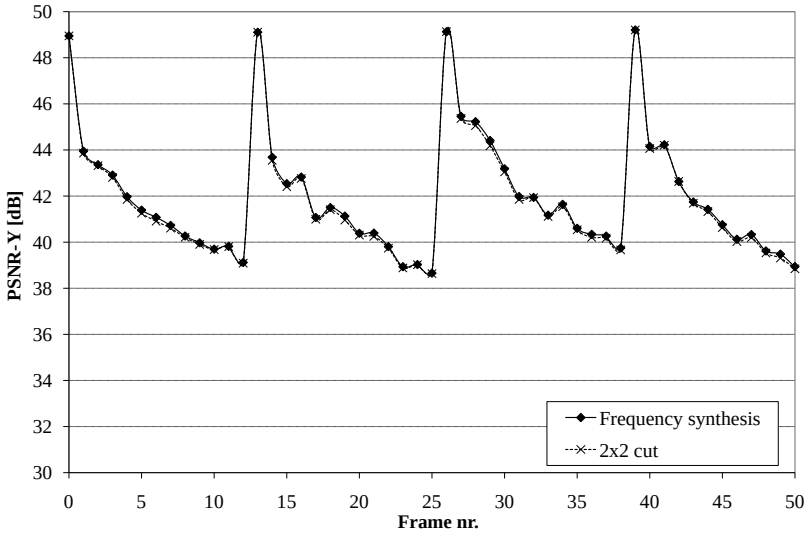
A. Residual data downsizing

Figure 4.10 shows results for the *Foreman* sequence, which was decoded at QCIF resolution from its original CIF resolution. Both the 2×2 cut and the frequency synthesis technique are evaluated. Full-pixel MCP was used during encoding, to distinguish the loss in motion vector accuracy from the loss of residual data. Both curves present the same behavior. Since intra-coded pictures are fully decoded and subsequently downscaled, a high PSNR is obtained for these pictures. After the intra-coded picture in every GOP, quality will fall back to PSNR values of 45 dB and below. The quality will decrease steadily until the end of the GOP. Note that the PSNR gap decreases for higher QP values (Figure 4.10(b)), which renders the reduced-resolution approach somewhat more useful for low-bit rate applications.

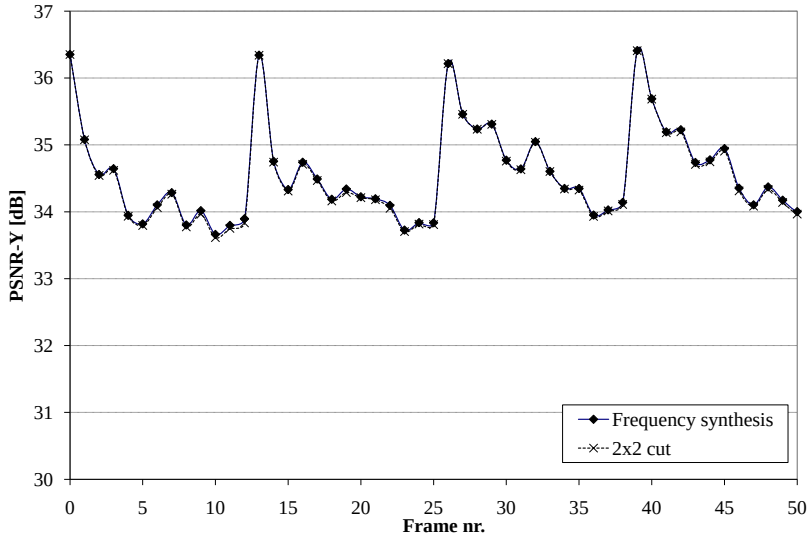
Results show that a small increase in PSNR is obtained for frequency synthesis when compared to the 2×2 cut method (limited to 0.2 dB). However, none of both filters is able to put a halt to drift.

B. Motion vector accuracy and interpolation filters

The impact of the motion vector accuracy and the discussed interpolation techniques becomes clear from Figure 4.11 for the *Foreman* sequence with subpixel MCP (down to quarter-pixel level) enabled. For the higher-complexity interpolation techniques (8-tap and 6-tap filters), the filter combination with best performance is shown in the graphs, i.e., combination (862). The lower-complexity combination (442) performs approximately 0.5 dB worse for QP 22 (Figure 4.11(a)). For QP 37 (Figure 4.11(b)), the gap is reduced to less than 0.2 dB. Especially for lower bit rate video, using reduced-complexity interpolation filters can be considered without compromising quality. The other filter triples result in PSNR values in between the (862) and (442) combinations, and are omitted for clarity. 1/8-pel interpolation is clearly able to reduce the loss due to motion vector inaccuracy when compared to the 1/4-pel approaches,



(a) 2×2 cut vs. frequency synthesis (*Foreman*, CIF to QCIF, QP 22).



(b) 2×2 cut vs. frequency synthesis (*Foreman*, CIF to QCIF, QP 37).

Figure 4.10: Comparison between 2×2 cut and frequency synthesis techniques for residual data downsizing.

i.e., the (82) and (62) combinations[1]. A gain of more than 1.5 dB is noticed to-

[1]The (42) approach practically coincides with the (62) curve, and can be used to reduce complexity.

wards the end of the GOP for the 1/8-pel approaches. For both motion vector truncation (1/4-pel interpolation) and 1/8-pel interpolation, however, quality declines towards the end of the GOP.

A minor increase in PSNR (but less than 0.1 dB) was noticed for the optimized filters described in 4.2.3. The difference is minimal since the obtained optimized filters are well approximated by the 8-tap Wiener filter.

Similar results were found for the *Soccer* sequence, decoded at CIF resolution from its original 4CIF format (Figure 4.11(c)).

4.3 Spatial resolution reduction transcoding

Reduced-resolution decoding can be regarded as a first step towards transcoding to lower resolutions. Extra difficulties are introduced when the input data needs to be converted to a compliant output bitstream, which is decodable at the receiver-end decoder. For this, a mapping needs to be carried out of input syntax elements to compliant output syntax elements in the reduced resolution.

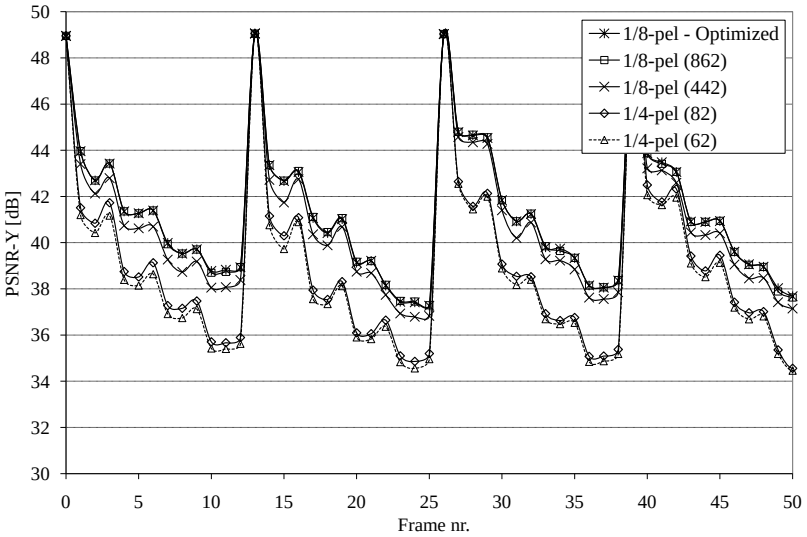
4.3.1 Issues in H.264/AVC spatial resolution mapping

In this section, we focus on mapping problems related to spatial resolution reduction transcoding of motion-compensated pictures in H.264/AVC. Four major issues arise when downscaling H.264/AVC video streams without fully decoding and reencoding. If not taken care of properly, spatial resolution reduction transcoding would result in artifacts and drift in the output video stream.

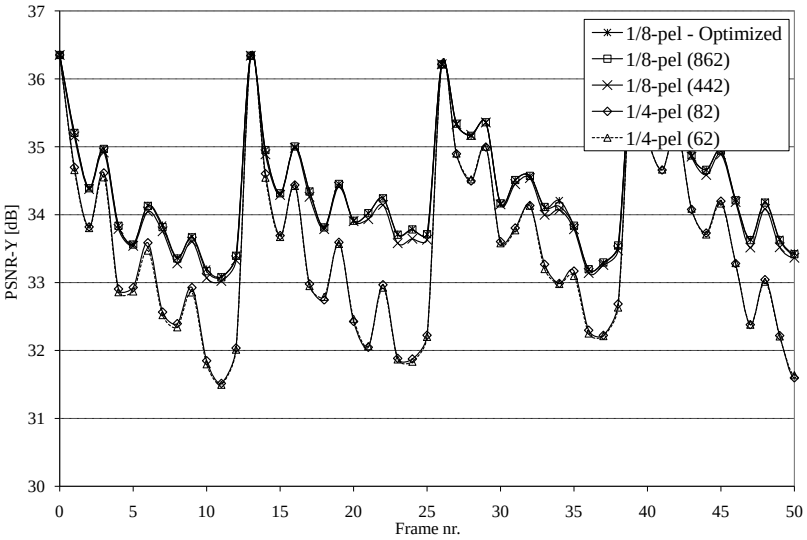
These four problems are illustrated in Figure 4.12 for a possible mapping; shaded areas correspond to blocks subject to misprediction.

A. Submacroblock partitions

Firstly, the H.264/AVC tree-structured motion compensation design allows block sizes smaller than 8×8 pixels, i.e., submacroblock partitions down to 8×4, 4×8, or 4×4 pixels. For these submacroblock partitions, there are no respective counterparts in the reduced-resolution domain. When downsizing H.264/AVC-coded sequences, this means that a straightforward mapping of macroblock partitions and motion vectors from four macroblocks to one is not possible if submacroblock partitions are used in the original stream. This is illustrated in Figure 4.12(a), where the bottom-right macroblock in the original resolution contains 4×8 and 4×4 submacroblock partitions.



(a) 1/8-pel vs. motion vector truncation (*Foreman*, CIF to QCIF, QP 22).

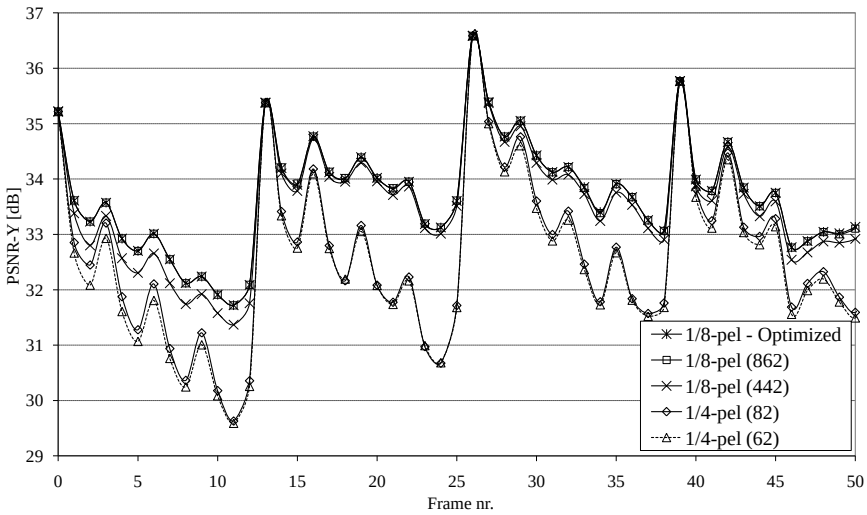


(b) 1/8-pel vs. motion vector truncation (*Foreman*, CIF to QCIF, QP 37).

Figure 4.11: Comparison between 1/8-pel and motion vector truncation.

B. Multiple reference pictures

A second problem is related to the multiple reference picture motion-compensated prediction design. In H.264/AVC, up to 16 reference pictures



(c) 1/8-pel vs. motion vector truncation (*Soccer*, 4CIF to CIF, QP 37).

Figure 4.11: Comparison between 1/8-pel and motion vector truncation - continued.

can be used. The reference picture lists can contain both short-term and long-term reference pictures. Reference picture indices can vary for every macroblock partition; the same reference picture, however, will be shared by all *sub*macroblock partitions in the same macroblock partition. This corresponds to a reference picture granularity down to 8×8 pixels. A misprediction occurs after downscaling H.264/AVC streams if different reference pictures were used in the original stream to predict a single macroblock. Up to four different reference picture indices need to be mapped to a single reference picture index. In Figure 4.12(b), an example is shown where macroblocks two and four (in raster scan order) are predicted based on multiple reference pictures.

C. Variable prediction direction in B pictures

A further complication occurs in B pictures, where in a single macroblock the choice can be made between motion-compensated prediction based on forward prediction (reference pictures in reference picture list 0), backward prediction (reference picture list 1), or bidirectional prediction (prediction from both lists). This selection can vary with the same granularity as for the reference indices, i.e., a different choice can be made for every 8×8 block of pixels. This is illustrated in Figure 4.12(c) for a possible mapping of four macroblocks containing macroblock partitions using forward prediction (list 0, abbreviated as 'L0'), backward prediction (list 1, 'L1'), and bidirectional prediction (both

lists, 'Bi').

D. Intra-coded macroblocks

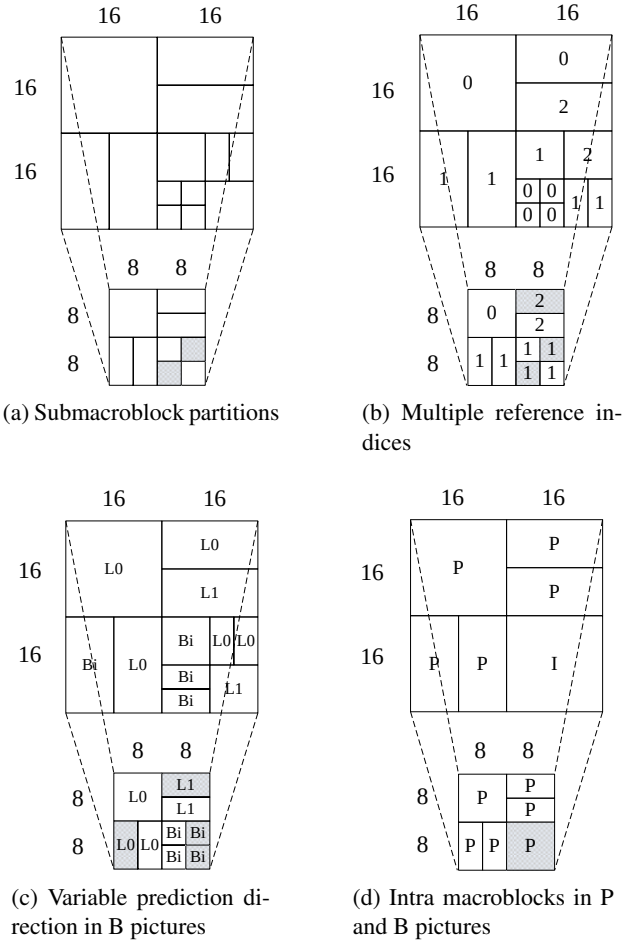
Fourthly, the availability of intra-coded macroblocks in P and B pictures requires an update of techniques available in previous video coding standards. Intra-coded macroblocks form their prediction based on the pixels of surrounding, reconstructed blocks, and can be inserted at any location in P and B pictures. Reconstruction in the reduced resolution can be tackled as explained in Section 4.2.4. In the case of transcoding, the input intra-coded macroblock additionally needs to be converted to a compliant output macroblock type. The use of an MCP macroblock type can be considered during conversion.

From here on, macroblocks that do not suffer from any of these complications will be referred to as 'directly-mappable macroblocks' (DM macroblocks), whereas the 'non-directly-mappable macroblocks' (NDM macroblocks) are restricted by one of the above reasons. The NDM macroblocks require closer attention for spatial resolution reduction transcoding.

4.3.2 Transcoding Architectures

A. Cascaded decoder-encoder architecture

In the case of spatial resolution reduction, the cascaded decoder-encoder approach will completely decode the original sequence, downscale the decoded frames in the pixel domain, and subsequently reencode the downsized frames [5]. As in the previous chapters, this approach can be regarded as a reference for rate-distortion performance, but has low computational efficiency due to highly complex operations at the encoder side. In order to speed up the reencoding process, the time-consuming motion estimation process can be avoided by using motion mapping, i.e., finding suitable macroblock partitions, reference indices, and motion vectors based on the information available in the original bitstream. A number of mode mapping algorithms have been presented in literature that limit the loss when compared to rate-distortion optimal mode and motion estimation, e.g., in [35, 97]. The resulting architecture is shown in Figure 4.13. Both motion-compensated prediction and intra prediction can be used to form the most appropriate prediction signal. In the following sections, we investigate further simplifications of this transcoder architecture, leading to a dynamic-complexity spatial resolution reduction transcoder.



(a) Submacroblock partitions

(b) Multiple reference indices

(c) Variable prediction direction in B pictures

(d) Intra macroblocks in P and B pictures

Figure 4.12: Coding tools in H.264/AVC leading to mapping-related problems; areas subject to mapping-related errors are indicated in gray.

B. Open-loop architecture for DM macroblocks

If the input video stream contains only DM macroblocks, a straightforward mapping can be used from the incoming macroblock partitions, reference indices, and motion vectors to the output bitstream. Downsizing techniques in the compressed domain as in [99] can be used in this case, based on an open-loop transcoder architecture, as shown in Figure 4.14. The downsampling can be performed in the pixel domain as well as in the compressed domain. In the latter case, frequency synthesis techniques can be used, as discussed in

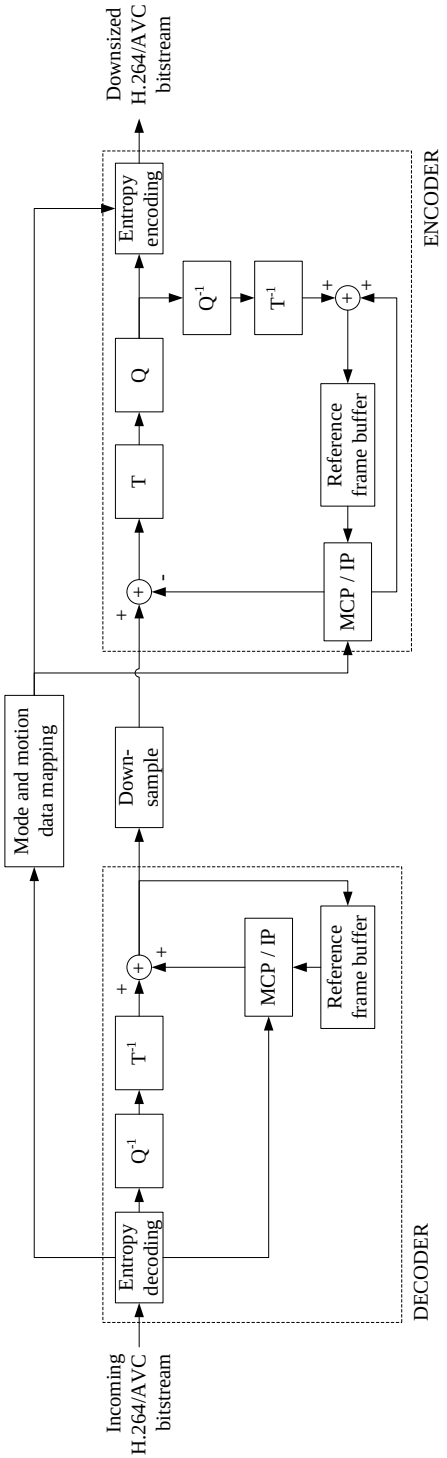


Figure 4.13: Cascaded spatial resolution transcoder with mode and motion data mapping.

Section 4.2.1.

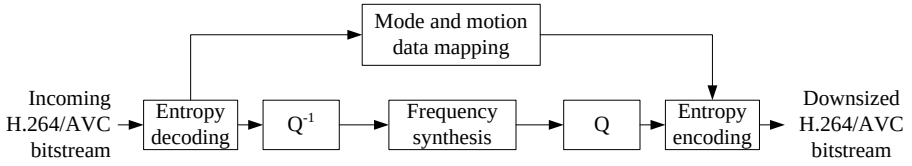


Figure 4.14: Open-loop spatial resolution transcoder (with frequency synthesis used for residual data downsizing).

The open-loop architecture can be considered as the most efficient transcoding approach, but has no provisions to update residual data when a change is required to the motion parameters when NDM macroblocks are encountered in the incoming bitstream.

C. Proposed architecture for DM and NDM macroblocks

When NDM macroblocks are present, open-loop mapping of transform coefficients results in serious errors due to wrongfully used reference pictures or motion vectors for the downsized stream. For these macroblocks, a correction of the MCP reference block is required. For this reason, we introduce the architecture as shown in Figure 4.15. In this architecture, correction of the errors (represented by the second motion compensation step) is only required for NDM macroblocks. To allow this correction operation, a *reduced-resolution* reference picture is created, by using reduced-resolution decoding approaches as introduced in Section 4.2. This is represented in Figure 4.15 by the first motion compensation loop.

The reduced-resolution pixel-domain reconstruction allows to solve the shortcomings of transform-domain transcoding solutions. In particular, it allows to correct the residual data blocks in case an update in mode or motion information is required. The signals in bold in Figure 4.15 refer to the residual data blocks, before and after the successive operations. The upper-case signals indicate transform-domain blocks, while the lower-case signals correspond to pixel-domain residual data blocks. More detail on the residual data mapping process is given in Section 4.3.4.

A number of differences with the cascaded decoder-encoder architecture can be identified that reduce computational complexity. Firstly, for DM macroblocks, open-loop downsampling can be applied. Note that for these blocks, the first motion compensation step is also performed, because each block can be used itself as reference for future motion-compensated prediction. Sec-

ondly, complexity is reduced by only applying motion compensation at reduced resolution. In the case of dyadic downscaling, complexity is reduced by a factor four for this operation. Also, this results in reference pictures that need only be stored at the reduced resolution. The use of the high-resolution buffers from the first loop in the cascaded architecture is avoided. Finally, the second motion compensation step is only required for correction of NDM macroblocks. This additional motion compensation operation uses the updated motion vector, or adjusted reference picture, and compensates for the mismatches mentioned in Section 4.3.1. As will be explained later, this second motion compensation loop can also be used for refinement of DM macroblocks.

Intra-coded macroblocks Due to the relatively low computational complexity of intra prediction, reencoding of intra-coded pictures is applied. The complexity of the overall architecture is influenced only to a minor extent. Downsampling of the intra frames is executed according to the sine-windowed Sinc-function, which is also recommended as downsampling filter for spatial scalability in SVC [103]. This filter can be defined as follows:

$$f(x) = \begin{cases} \frac{\sin(\pi \frac{x}{D})}{\pi \frac{x}{D}} \sin\left(\frac{\pi}{2}(1 + \frac{x}{ND})\right) & |x| < ND \\ 0 & \text{otherwise} \end{cases}$$

with a decimation parameter $D = 2.5$ and $N = 3$ lobes for the Sinc-function on each side. For practical applications and complexity reasons, the filter size is limited to 12 taps.

Intra-coded macroblocks in P and B pictures will prevent transform-domain downscaling solutions to be practically viable in H.264/AVC. Although transform-domain intra prediction has been mentioned as an alternative approach to removing intra macroblocks from P and B pictures, non-linear operations result in highly degraded quality and highly complex computations, requiring an extensive amount of floating-point multiplications [104]. In the proposed approach, it becomes possible to convert the intra macroblocks to intra or inter macroblocks in the downsized stream, since the low-resolution reconstructed pictures are available in the reduced-resolution decoder. Reconstruction of intra-coded macroblocks is obtained by using the technique in 4.2.4.

Dynamic complexity refinement So far, the introduced reduced-complexity architecture only used the second motion compensation step for correction of NDM macroblocks, while DM macroblocks could be transcoded open-loop. It is possible, however, to use the second motion compensation step to further refine the motion vectors or in order to select a more appropriate choice

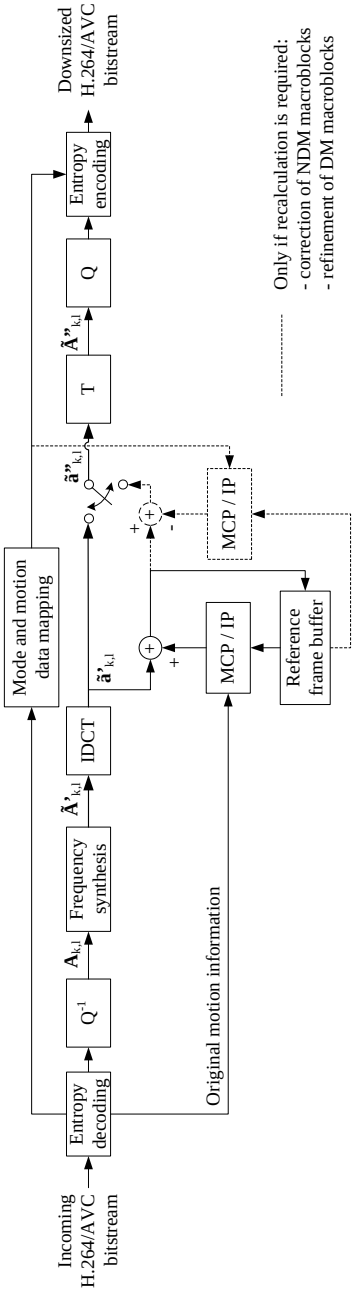


Figure 4.15: Proposed spatial resolution transcoder.

of reference picture. The amount in which this functionality is used, can be dynamically varied, resulting in a trade-off between rate-distortion efficiency and computational complexity. From the results in Section 4.3.5, the impact of this refinement step becomes clear. Rate-distortion performance is improved in a significant way by refining motion vectors and macroblock partitioning. In order to achieve this, techniques can be used as in Section 4.3.3.

4.3.3 Mode and motion data mapping

A. Mapping for DM macroblocks

For DM macroblocks, the original partitions can be mapped to their respective downscaled counterparts in a straightforward manner. As an example, a single macroblock is mapped to an 8×8 partition, and incoming macroblock partitions are mapped to 8×8, 8×4, 4×8, or 4×4 submacroblock partitions. This results in the use of 8×8 partitions for every downscaled macroblock. The motion vectors are mapped accordingly, while the reference indices and prediction directions remain identical. If desired, the motion parameters that are obtained in this way can be further refined, as will be discussed later in this section.

B. Mapping for NDM macroblocks

Particular care needs to be taken for the case of NDM macroblocks. For NDM macroblocks, the appropriate output motion parameters need to be determined, i.e., motion vectors, reference indices, and prediction directions. In a first step, an initial mapping is formed, leading to an H.264/AVC-compliant bitstream. As is the case for DM macroblocks, further refinement can be applied if desired. The initial mapping consists of the three following steps.

Motion vector mapping When submacroblock partitions were used in the original bitstream, no downscaled counterpart exists for these partitions. A new motion vector needs to be derived in a rate-distortion optimized sense, based on the motion information from the corresponding (up to four) incoming motion vectors. The motion vectors from the incoming bitstream are treated as candidate output motion vectors, and are evaluated in a rate-distortion optimized way. After derivation of the new output motion vector, correction of the corresponding residual values is performed using the second motion compensation loop in the proposed architecture. The pixels for which correction is required depends on the choice of motion vector for the submacroblock partition.

Reference index and prediction direction mapping An additional problem in H.264/AVC downscaling occurs when multiple reference indices were used to code a single macroblock. In this case, a selection needs to be made of the best suited reference index for the composed macroblock partition. In B pictures, this is further complicated when multiple prediction directions were used within macroblocks, i.e., a combination of forward, backward, and/or bidirectional prediction.

Selection of the appropriate motion parameters is performed similar to the motion vector selection, by rate-distortion optimized evaluation of the candidate reference indices and prediction directions.

Mapping of intra macroblocks in P and B pictures When intra macroblocks are encountered, a proper macroblock conversion needs to be performed. When the four input macroblocks are intra-coded, the corresponding output macroblock is likely to be intra-coded also. Otherwise, a *mixed* block is found. In this case, the possibility of using an intra-coded macroblock in the output stream is investigated. As an alternative, MCP is evaluated using motion vectors derived in the same way as described above for the case of updated reference indices.

C. Motion refinement

Once an H.264/AVC-compliant bitstream is obtained by mapping DM and NDM macroblocks, refinement of the motion parameters can be applied, depending on the requirements of the scenario in which the transcoder is used. Further refinement somewhat increases complexity, yet improves rate-distortion performance. In the presented architecture, motion refinement is evaluated by using the bottom-up limitation mode decision process. Besides the initial mapping (as described above), the potential benefit of merging (sub)macroblock partitions is evaluated. The merging process is executed as previously described in Section 2.5

Overall complexity is determined by the amount of macroblocks to which this process is applied, and by the amount of refinement steps that are applied to each macroblock. Note that an increase in number of refined macroblocks also induces a refinement of residual data for these macroblocks, i.e., the activation of the second motion compensation loop.

D. Rate-distortion optimized mode selection

An important advantage of the proposed architecture is that pixel-domain (reduced-size) reference pictures are created which can be used as a reference

for future prediction. These pictures allow the architecture to correct artifacts that would otherwise arise in NDM macroblocks. The availability of these reference pictures also enables us to calculate the displacement difference for every set of motion parameters, and perform rate-distortion optimized motion selection. Given the evaluated partition type, motion vector, and reference index, the output rate and distortion can be determined. Using this calculation, a choice is made based on Lagrangian minimization, as elaborated on in Section 2.5.2.

4.3.4 Residual data mapping

Here, we revisit the mapping process for residual data for the case of spatial resolution reduction transcoding. As a start, residual data downsizing can be reused as explained for reduced-resolution decoding in Section 4.2.1. Again, both the 2×2 cut and frequency synthesis techniques can be used (with comparable quality results). The resulting values are used in the reduced-resolution reconstruction loop, as shown in Figure 4.15. The output 4×4 matrices $\tilde{\boldsymbol{a}}'_{k,l}$ (corresponding with the notation in Section 4.2.1) are used to construct the reduced-size reference pictures in the first motion compensation loop. This technique can be reused as such for residual data of DM macroblocks.

The changes introduced by motion parameters downsizing, however, are not reflected in this technique. If a valid H.264/AVC bitstream needs to be output, changes in the motion information are required, which in turn necessitates an update of the residual data. The use of submacroblock partitions, multiple reference indices, or intra macroblocks in the original stream will lead to significant artifacts in the transcoded video stream if the output residual coefficients are not updated accordingly. Adjustments are required in case: NDM macroblocks are present, as is the case for typical H.264/AVC bitstreams; or, if motion data refinement of DM macroblocks is applied. In these cases, the approach as discussed in Section 4.3.3 implies a refinement of the residual data using the second motion compensation loop in our proposed architecture. In order to obtain the output residual data, the motion compensation loop uses the newly derived prediction direction, reference index (or indices) and motion vector(s), leading to residual data blocks $\tilde{\boldsymbol{a}}''_{k,l} \neq \tilde{\boldsymbol{a}}'_{k,l}$.

4.3.5 Results and discussion

The architecture as described in the previous section was implemented in software and tested using sequences with varying statistics, namely *Akiyo*, *Paris*, and *Foreman* (original: CIF resolution), and *Harbour* and *Soccer* (original:

4CIF resolution). The sequences were encoded using the Joint Model reference software, version 13.0. All sequences were encoded with an IBBP GOP structure and using CABAC entropy coding, with rate-distortion optimization enabled. Quantization parameter values of 22, 27, 32, and 37 were used. The reencode curves were obtained by successively decoding, pixel-domain downscaling, and reencoding. For downscaling, the SVC reference software down converter tool was used, which makes use of the 12-tap sine-windowed Sinc-function. The downscaled sequences were reencoded using the same encoder options in the reduced resolution (JM 13.0). In the tests, a GOP length of 12 frames was used. For PSNR comparison, the downsampled version of the original sequences was used. Here also, the sine-windowed Sinc filter was used for this operation. This filter is also used for the intra-coded pictures during transcoding. For the other pictures, however, the reconstruction at lower spatial resolution might not approximate the downsampled version of the original sequences.

A. Results for sequences containing DM macroblocks only

Firstly, it is worthwhile to look at results for sequences containing only DM macroblocks, i.e., when no submacroblock partitions, multiple reference pictures, or intra macroblocks are used in the original bitstreams. Figure 4.16 (*Foreman*) shows that in this case, the open-loop solution with frequency synthesis performs reasonably well, leading to rate-distortion losses limited to less than 2 dB when compared to full decoding and reencoding, with significant computational complexity savings. Similar results are obtained for *Paris* (Figure 4.17). Here, however, the gap increases towards higher bit rates.

B. Results for sequences containing NDM macroblocks

In Figure 4.18, results are shown for the *Paris* sequence containing NDM macroblocks in the input sequence. From these curves, it is clear that open-loop processing of residual data results in highly distorted results, and unacceptable quality loss. The presented transcoding architecture with correction of NDM macroblocks (indicated as 'no refinement') is able to significantly improve the R-D results. By additionally refining DM macroblocks, the output quality can further be improved. By refining all macroblocks, the curve indicated as 'with refinement' is achieved. Hence, this indicates the best achievable rate-distortion performance. For the generated results, the refinement step resulted in R-D points with slightly lower PSNR values; bit rate, however, decreased significantly after refinement. After the operation, results are able to approach the decoder-encoder cascade within 1-2 dB.

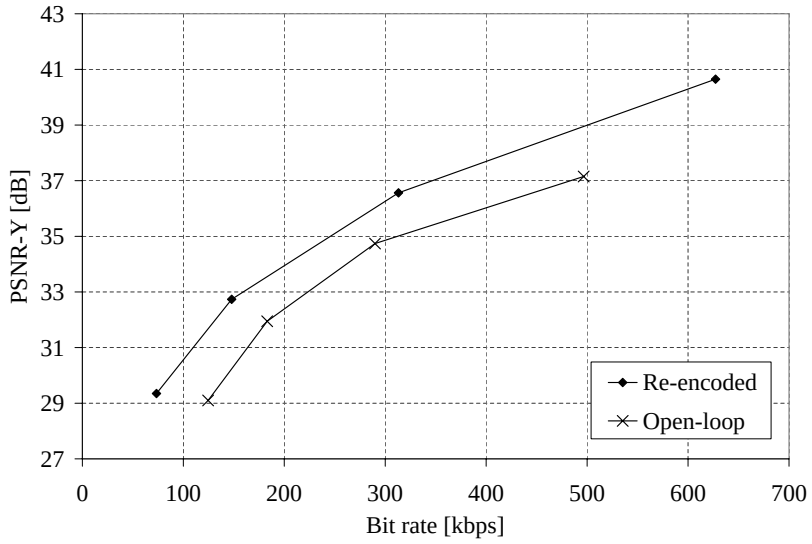


Figure 4.16: Rate-distortion performance for sequences containing only DM macroblocks (*Foreman*, CIF to QCIF).

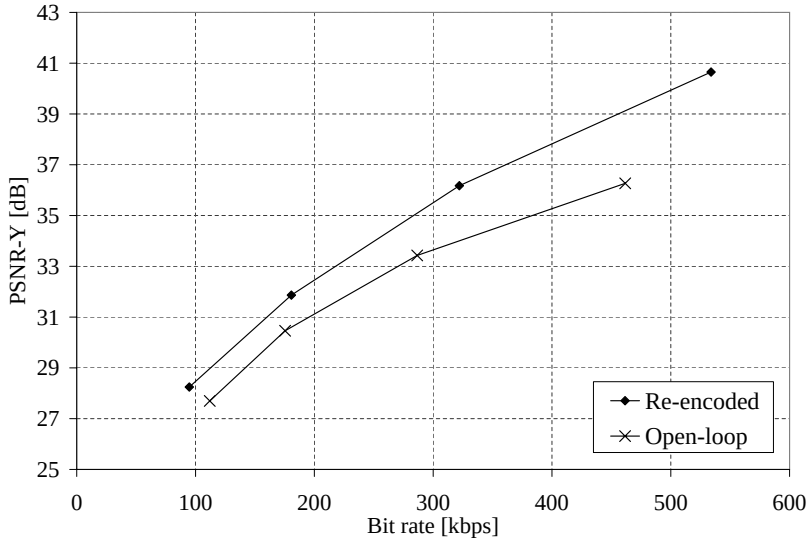


Figure 4.17: Rate-distortion performance for sequences containing only DM macroblocks (*Paris*, CIF to QCIF).

Results for the *Akiyo* sequence under the same conditions (i.e., containing NDM macroblocks) are shown in Figure 4.19. Here, the loss of open-loop

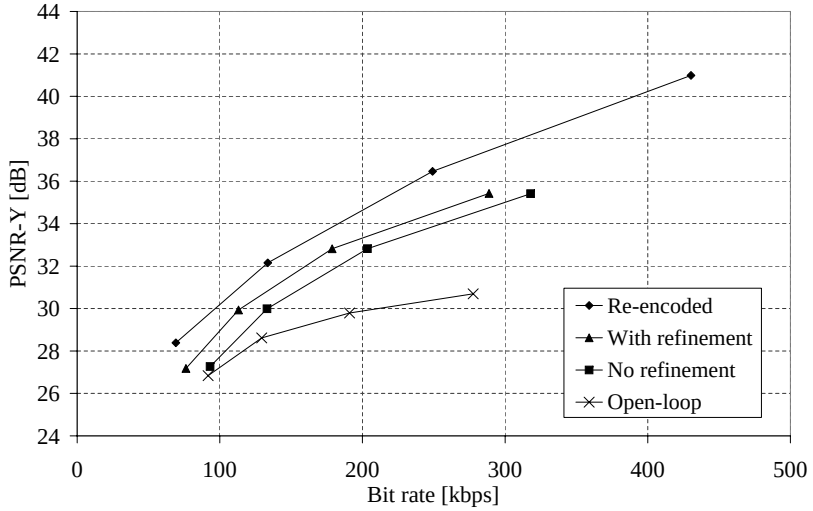


Figure 4.18: Rate-distortion performance for sequences containing NDM macroblocks (*Paris*, CIF to QCIF).

transcoding is less significant, due to the reduced amount of motion in the sequence. Quality is improved only to a minor extent due to correction of NDM macroblocks. Here, due to the low motion content, only a small number of macroblocks uses submacroblock partitioning, multiple reference indices, or intra macroblocks. In fact, more than 90% of the macroblocks are DM macroblocks for this sequence, even for low bit rates. This limits the achievable R-D gain of NDM macroblock correction. Refining DM macroblocks, however, can further improve the quality of the output bitstream by more than 1 dB.

Results for the *Soccer* and *Harbour* sequences, transcoded from 4CIF to CIF, are shown in Figure 4.20 and Figure 4.21. From these results, it can be seen that the presented algorithm results in larger reductions of the bit rate than reencoding for the same QP values. When compared to reencoding, more high-frequency information is removed from the bitstream. Here also, the loss in rate-distortion performance remains limited to 1-2 dB, and significant gains are obtained when compared to open-loop transcoding. This is particularly so for high-motion sequences, such as *Soccer* (Figure 4.20). For the *Harbour* sequence, the loss when compared to reencoding is limited to 0.5-1 dB in the lower bit rate range, but diverges towards higher bit rates.

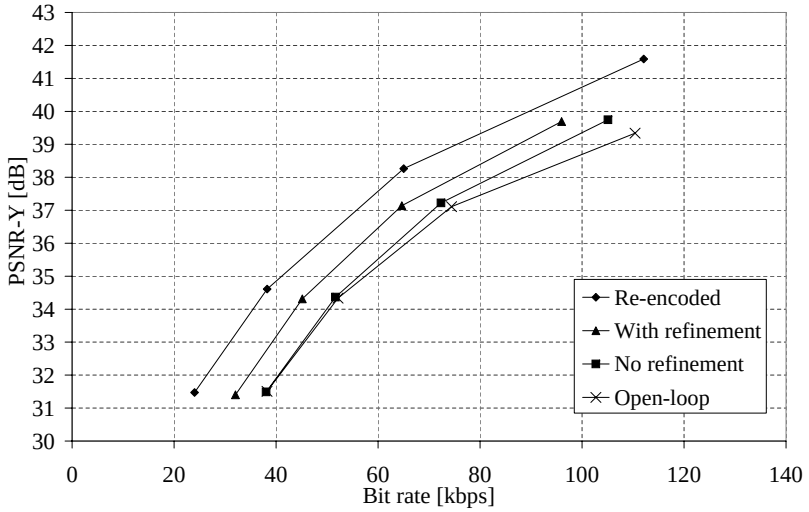


Figure 4.19: Rate-distortion performance for sequences containing NDM macroblocks (*Akiyo*, CIF to QCIF).

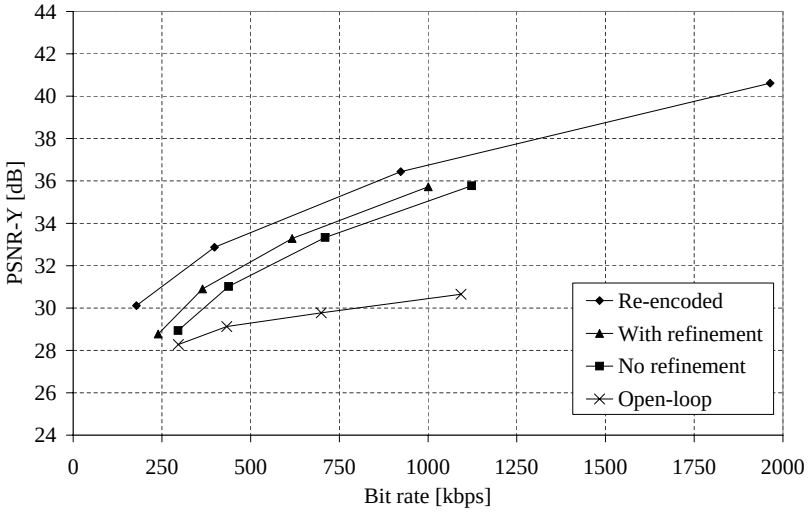


Figure 4.20: Rate-distortion performance for sequences containing NDM macroblocks (*Soccer*, 4CIF to CIF).

C. Complexity results

Computational complexity increases when mode and motion vector refinement is used, given that more macroblocks will be using the second motion compen-

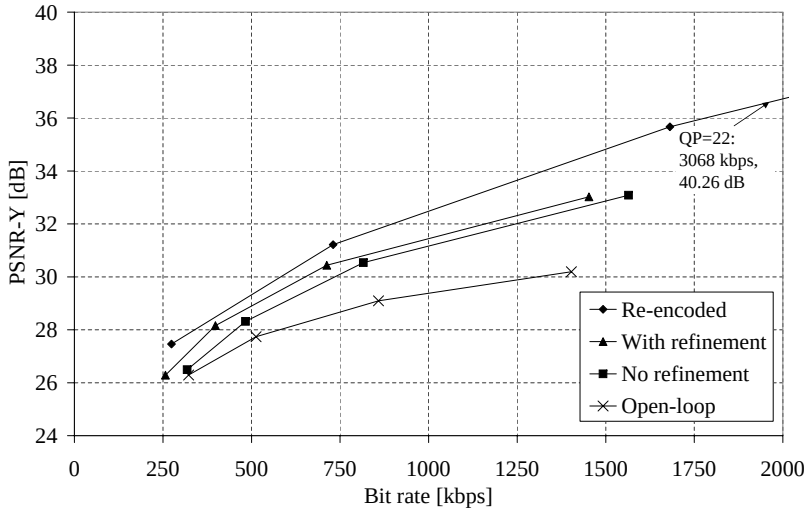


Figure 4.21: Rate-distortion performance for sequences containing NDM macroblocks (*Harbour*, 4CIF to CIF).

sation step. In this way, a trade-off is made between computational complexity and bit rate reduction. As an indication of complexity, timing results are given in Table 4.3, for reencoding, open-loop transcoding, and transcoding without and with refinement. The results are shown relative to the time needed for reencoding. From Table 4.3, it can be seen that significant computational complexity gains are made when compared to reencoding. When open-loop transcoding is applicable, a reduction of more than 97% is obtained. Depending on the amount of refinement applied to the downscaled bitstream, transcoding achieves timing gains of 89% to 96% relative to reencoding. This amount can be varied dynamically, depending on the available computational resources in the transcoding system.

4.4 Conclusions and original contributions

In this chapter, we discussed and evaluated computationally efficient techniques for both reduced-resolution decoding and spatial reduction transcoding for H.264/AVC. Although quality typically declines over successive frames (both for decoding and transcoding), reduced-resolution decoding might be considered a useful technique for limited GOP lengths. The problem of the loss of motion vector accuracy can be tackled by applying 1/8-pel interpolation techniques. Fixed interpolation filters were shown to result in a performance

Table 4.3: Timing results relative to reencoding [%]

	Akiyo (CIF to QCIF) QP				*Paris* (CIF to QCIF) QP			
	22	27	32	37	22	27	32	37
Open-loop	2.6	2.5	2.3	2.3	2.6	2.5	2.4	2.4
No refinement	4.4	4.2	3.9	3.8	4.9	4.7	4.6	4.1
With refinement	10.5	10.0	9.4	8.8	11.2	10.9	10.6	9.3

	Soccer (4CIF to CIF) QP				*Harbour* (4CIF to CIF) QP			
	22	27	32	37	22	27	32	37
Open-loop	2.7	2.4	2.2	2.2	2.8	2.6	2.5	2.5
No refinement	4.9	4.6	4.2	4.1	5.3	5.1	5.0	4.8
With refinement	11.5	10.7	9.7	9.3	11.9	11.6	11.1	11.0

close to that of optimized filters. The loss of residual data, however, causes rapid quality degradation; this applies to both the frequency synthesis and 2×2 cut techniques. Future research might focus on how to preserve more of the energy during residual data downsampling.

Moreover, we highlighted a number of problems as found in H.264/AVC spatial resolution reduction transcoding. These restrictions inhibit the use of straightforward, open-loop, residual data synthesis techniques. A low-complexity transcoder architecture is presented, which is able to handle both directly-mappable as well as non-directly-mappable macroblocks. Complexity is reduced by performing motion compensation only in the reduced-resolution domain. The refinement step of the proposed architecture can be used to further improve rate-distortion performance, at the cost of additional complexity. In this manner, a dynamic-complexity transcoder is made possible. Complexity, however, remains many times lower than that of reencoding, with reductions of 89 to 96% depending on the amount of refinement applied.

The author's work on this subject led to the following publications:

- Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Spatial Resolution Reduction Transcoding with Dynamic Complexity for H.264/AVC. Submitted to *Multimedia Systems Journal*.

- Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Ef-

ficient spatial resolution reduction transcoding for H.264/AVC. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. October 2008.

Chapter 5

Conclusions

The relevance and usefulness of scalable video coding cannot be ignored. But neither can the reality that single-layer coding will continue to dominate at least the next couple of years. The downsides of layered coding - reduced rate-distortion performance, computational complexity (mainly during encoding), and the resulting lack of available real-time encoders and decoders, will put up barriers for a potential breakthrough of SVC (at least in the short term). Given the recent (and still ongoing) migration from MPEG-2 (which is still prevalent) to H.264/AVC, and the accompanying financial repercussions, it is at present unsure if an additional investment in scalability provisions will come to pass.

Hence, transcoding remains an important technique in the adaptation of video streams. Different gradations of transcoding solutions can be considered, of which the most complex is the decoder-encoder cascade. The trick to reduce computational complexity is to work as much as possible in the compressed domain.

In Chapter 2, bit rate reduction transcoding based on requantization of residual information has been examined. A double-loop CPDT architecture with reuse of mode and motion information was considered as the (drift-free) reference when looking for fast transrating architectures. In the first part of this chapter, we focused on identifying lower-complexity architectures than CPDT. The main problem for these architectures is the introduction of error propagation. Without appropriate action, requantization of residual coefficients results in severe drift. For MPEG-2, the sources of drift and tailored solutions were investigated in a number of papers published over the last decade. The solutions as found in those publications are no longer representative for H.264/AVC video coding, however. New, extended, and improved coding tools complicate transcoding, as a result of intricate interdependencies between different syntax elements. In particular, the open-loop transrating architecture

has proven not to be useful, although this was a viable solution for MPEG-2. As shown in Chapter 2, two drift terms can be identified for transrating. The temporal drift term was also present for MPEG-2 transcoding and propagates along with motion-compensated prediction. The spatial drift term is new in H.264/AVC due to intra prediction and causes significant errors. Within a single frame, intra-coded macroblocks cause quality degradation without proper countermeasures. In Chapter 2, a spatial compensation technique was presented which diminishes the effect of spatial drift. An important contribution is the use of adaptive (mixed) transrating architectures which provide a trade-off between computational complexity and drift reduction by switching techniques based on the picture and macroblock type. From the results, it is shown that the architectures with spatial compensation clearly outperform traditional architectures with temporal compensation only.

The solutions were extended to support High profile, by introducing 8×8 luma intra prediction and the 8×8 integer transform in the architectures, and by updating the appropriate quantization formulas. For high-resolution streams, similar conclusions are drawn. When large intra-coded regions are present in the stream, however, spatial compensation loses its efficacy. This is caused by integer arithmetic rounding errors, which result in non-perfect compensation. Although this is an effect which is also present for low-resolution sequences, its effect becomes especially troublesome for high-resolution video. When such a scenario is encountered (which can be detected during pre-processing of the streams), selective requantization can be applied, i.e., by avoiding requantization altogether for intra-coded macroblocks or for a subset of these macroblocks. As a result, spatial drift is further reduced or avoided.

Towards the high end of the complexity spectrum, solutions can be constructed with higher complexity than the CPDT transrater with mode and motion reuse. Additional complexity can be spent on refinement of mode and motion information. The pixel-domain transrater with motion refinement has been shown to result in significant gains. We also investigated whether refinement can be applied for the reduced-complexity architectures, i.e., in the transform domain. Although motion refinement leads to a better-matching motion field, the changes caused by merging motion partitions in the transform domain introduce mismatches leading to additional distortion. Careful examination in a rate-distortion sense is required to avoid performance losses. In particular for larger QP differences, transform-domain motion refinement will prove to result in improved rate-distortion results. Complexity, however, will increase due to the motion merging and refinement steps.

In Chapter 3, we provided solutions for creating scalable bitstreams starting from single-layer H.264/AVC streams. Architectures have been con-

structed that redistribute the residual data among different layers. At the decoder, coefficients are accumulated based on SVC inter-layer residual prediction. When using the regular coefficient accumulation process, however, reconstruction is not identical to the original input coefficients of the single-layer H.264/AVC stream. An additional problem is that of intra-coded macroblocks, which use a different mechanism for inter-layer prediction, i.e., inter-layer intra prediction. In particular, the single-loop decoding requirement has to be fulfilled, hence requiring tailored techniques for these macroblocks.

In the presented architectures, the problem of non-exact coefficient accumulation is solved by making use of the bitstream rewriting functionality available in SVC. Using this functionality, perfect reconstruction is achieved, i.e., when all layers are present, identical quality is obtained as for the original single-layer H.264/AVC stream. As a second benefit, intra-coded macroblocks are no longer reconstructed based on inter-layer intra prediction, but in the same way as MCP macroblocks by inter-layer residual prediction. Hence, the constraints imposed by single-layer decoding are lifted. As it turns out, rate-distortion performance of the presented architectures benefits from the bitstream rewriting functionality, not only for the top layer extraction point (i.e., the perfect reconstruction point), but also for the lower-rate extraction points.

Furthermore, refinement of motion data during H.264/AVC-to-SVC conversion has been investigated. When no motion refinement is applied, motion data is concentrated in the base layer. We examined whether refinement of motion data in lower layers can lead to an increase in rate-distortion performance for the desired layer(s). This is achieved by introducing a multi-layer optimization approach, and was demonstrated for two SVC quality layers. As a downside of motion refinement, motion data will be spread across the two layers, resulting in a slightly increased bit rate for the stream when all layers are present.

Spatial resolution reduction without decoding in full resolution is clearly a challenging operation. Both reduced-resolution *decoding* and *transcoding* have been investigated in Chapter 4. Reduced-resolution decoding can be used for short GOP lengths when a number of optimizations are used, such as 1/8-pel interpolation and 8-tap interpolation filters. Nonetheless, quality rapidly declines, leading to limited applicability for longer GOP structures. Additional conversion issues are present for reduced-resolution transcoding, such as submacroblock partitions and multiple reference indices. These issues render open-loop techniques useless, as they would result in disturbing artifacts in the decoded stream without appropriate action. A low-complexity architecture is proposed which solves these issues by including a reduced-resolution reconstruction loop. Given the available pixel-domain reference, residual data

can be updated reflecting the changes in motion data.

Further research can focus on how to capture more of the residual data energy, and hence reduce the drift in reduced-resolution decoding and transcoding.

And the future?

At this very moment, requirements are being drafted of what the next-generation video codec of VCEG and MPEG will look like. As of now, it is unclear whether such a project will be embarked upon by a joint team of both parties. But after the successes of MPEG-2 and H.264/AVC, no one doubts that future coding specifications will benefit from a joint approach, both from a technical and commercial perspective. In the current status of the project, ambitiously coded-named H.NGC (for 'next-generation coding') or HVC (for 'high-performance video coding'), the main requirements focus on coding efficiency and computational complexity, leaving scalability as a 'desirable' feature. And in the case that a scalable extension of such a standard would be developed, it is unclear what the impact of the novel coding tools will be on layered coding.

In any way, transcoding will not lose its spot in video adaptation. On the contrary, high-definition video, which is the target of the current efforts, and whose importance will only continue to surge, will pose increasing demands on video coding, decoding, and adaptation processes. And due to the introduction of (yet) another video coding standard, efficient heterogeneous transcoding between the new and existing standards will remain a technically challenging task.

Appendix A

Pixel-domain and transform-domain intra prediction

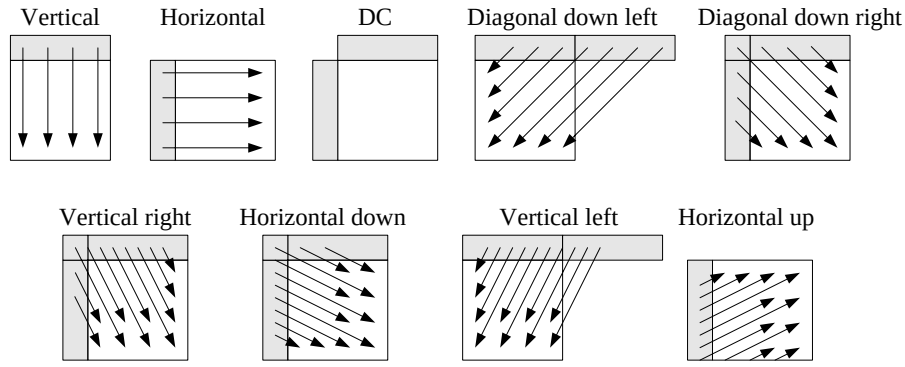
Intra prediction is used in H.264/AVC to exploit the spatial redundancy between neighbouring pixels. A block is predicted using previously encoded and reconstructed pixels of surrounding blocks. In H.264/AVC, a macroblock can be predicted using nine 4×4 or four 16×16 intra prediction modes (or nine 8×8 modes for the High profiles). The denotation of the surrounding blocks (block A, B, C, and D) is shown in Fig. A.1. The block width and height can be equal to 4 or 16, depending on the use of 4×4 or 16×16 intra prediction. This appendix gives a brief overview of the 4×4 and 16×16 luma intra prediction, which are common to all profiles. The matrices which we derived for transform-domain intra prediction are given in Section A.2.

Block D	Block B	Block C
Block A	Current block	

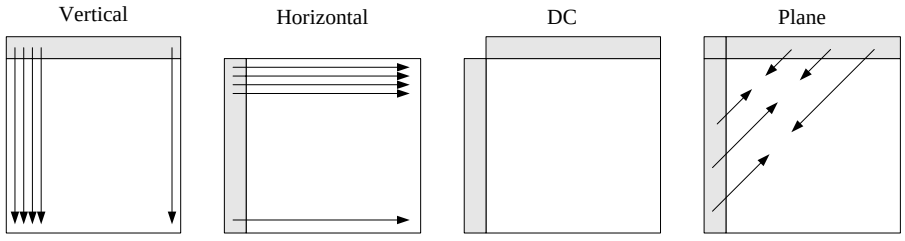
Figure A.1: Determination of neighboring blocks in H.264/AVC.

A.1 Pixel-domain intra prediction formulas

The nine intra 4×4 modes are shown in Fig. A.2(a) and the four intra 16×16 modes (vertical, horizontal, DC, and plane prediction) are shown in Fig. A.2(b).



(a) Intra 4×4 prediction modes



(b) Intra 16×16 prediction modes

Figure A.2: Intra 4×4 and 16×16 prediction modes.

The formulas for all intra prediction modes are given below. The values $p_{X,ij}$ represent the reconstructed pixels (or requantization error values in the case of single-loop compensation) of neighboring blocks, with $X \in \{A, B, C, D\}$ and $i, j = 0, \ldots, 3$ in the case of 4×4 prediction and $i, j = 0, \ldots, 15$ in the case of 16×16 prediction.

$$\boldsymbol{p}_0 = \begin{bmatrix} p_{B,3,0} & p_{B,3,1} & p_{B,3,2} & p_{B,3,3} \\ p_{B,3,0} & p_{B,3,1} & p_{B,3,2} & p_{B,3,3} \\ p_{B,3,0} & p_{B,3,1} & p_{B,3,2} & p_{B,3,3} \\ p_{B,3,0} & p_{B,3,1} & p_{B,3,2} & p_{B,3,3} \end{bmatrix} . \tag{A.1}$$

$$\boldsymbol{p}_1 = \begin{bmatrix} p_{A,0,3} & p_{A,0,3} & p_{A,0,3} & p_{A,0,3} \\ p_{A,1,3} & p_{A,1,3} & p_{A,1,3} & p_{A,1,3} \\ p_{A,2,3} & p_{A,2,3} & p_{A,2,3} & p_{A,2,3} \\ p_{A,3,3} & p_{A,3,3} & p_{A,3,3} & p_{A,3,3} \end{bmatrix} . \tag{A.2}$$

$$\boldsymbol{p}_2 = \begin{bmatrix} \alpha & \alpha & \alpha & \alpha \\ \alpha & \alpha & \alpha & \alpha \\ \alpha & \alpha & \alpha & \alpha \\ \alpha & \alpha & \alpha & \alpha \end{bmatrix} , \tag{A.3}$$

where

$$\alpha = \frac{\sum_{i=0}^{3} p_{A,i,3} + \sum_{j=0}^{3} p_{B,3,j}}{8} . \tag{A.4}$$

In case block A or block B is not available, only the pixels from the available block are taken into account. If none of both are available, $\alpha = 128$.

The remaining pixel-domain prediction matrices are given in Equation (A.5) to Equation (A.10).

$$\boldsymbol{p}_3 = \frac{1}{4}\begin{bmatrix} p_{B,3,0} + 2\,p_{B,3,1} + p_{B,3,2} & p_{B,3,1} + 2\,p_{B,3,2} + p_{B,3,3} & p_{B,3,2} + 2\,p_{B,3,3} + p_{C,3,0} & p_{B,3,3} + 2\,p_{C,3,0} + p_{C,3,1} \\ p_{B,3,1} + 2\,p_{B,3,2} + p_{B,3,3} & p_{B,3,2} + 2\,p_{B,3,3} + p_{C,3,0} & p_{C,3,0} + 2\,p_{C,3,1} + p_{C,3,2} & p_{C,3,1} + 2\,p_{C,3,2} + p_{C,3,3} \\ p_{B,3,2} + 2\,p_{B,3,3} + p_{C,3,0} & p_{B,3,3} + 2\,p_{C,3,0} + p_{C,3,1} & p_{C,3,0} + 2\,p_{C,3,1} + p_{C,3,2} & p_{C,3,1} + 2\,p_{C,3,2} + p_{C,3,3} \\ p_{B,3,3} + 2\,p_{C,3,0} + p_{C,3,1} & p_{C,3,0} + 2\,p_{C,3,1} + p_{C,3,2} & p_{C,3,1} + 2\,p_{C,3,2} + p_{C,3,3} & p_{C,3,2} + 3\,p_{C,3,3} \end{bmatrix}. \tag{A.5}$$

$$\boldsymbol{p}_4 = \frac{1}{4}\begin{bmatrix} p_{A,0,3} + 2\,p_{D,3,3} + p_{B,3,0} & p_{D,3,3} + 2\,p_{B,3,0} + p_{B,3,1} & p_{B,3,0} + 2\,p_{B,3,1} + p_{B,3,2} & p_{B,3,1} + 2\,p_{B,3,2} + p_{B,3,3} \\ p_{D,3,3} + 2\,p_{A,0,3} + p_{A,1,3} & p_{A,0,3} + 2\,p_{D,3,3} + p_{B,3,0} & p_{D,3,3} + 2\,p_{B,3,0} + p_{B,3,1} & p_{B,3,0} + 2\,p_{B,3,1} + p_{B,3,2} \\ p_{A,0,3} + 2\,p_{A,1,3} + p_{A,2,3} & p_{D,3,3} + 2\,p_{A,0,3} + p_{A,1,3} & p_{A,0,3} + 2\,p_{D,3,3} + p_{B,3,0} & p_{D,3,3} + 2\,p_{B,3,0} + p_{B,3,1} \\ p_{A,1,3} + 2\,p_{A,2,3} + p_{A,3,3} & p_{A,0,3} + 2\,p_{A,1,3} + p_{A,2,3} & p_{D,3,3} + 2\,p_{A,0,3} + p_{A,1,3} & p_{A,0,3} + 2\,p_{D,3,3} + p_{B,3,0} \end{bmatrix}. \tag{A.6}$$

$$\boldsymbol{p}_5 = \frac{1}{4}\begin{bmatrix} 2\,p_{D,3,3} + 2\,p_{B,3,0} & 2\,p_{B,3,0} + 2\,p_{B,3,1} & 2\,p_{B,3,1} + 2\,p_{B,3,2} & 2\,p_{B,3,2} + 2\,p_{B,3,3} \\ p_{A,0,3} + 2\,p_{D,3,3} + p_{B,3,0} & p_{D,3,3} + 2\,p_{B,3,0} + p_{B,3,1} & p_{B,3,0} + 2\,p_{B,3,1} + p_{B,3,2} & p_{B,3,1} + 2\,p_{B,3,2} + p_{B,3,3} \\ p_{D,3,3} + 2\,p_{A,0,3} + p_{A,1,3} & 2\,p_{D,3,3} + 2\,p_{B,3,0} & 2\,p_{B,3,0} + 2\,p_{B,3,1} & 2\,p_{B,3,1} + 2\,p_{B,3,2} \\ p_{A,0,3} + 2\,p_{A,1,3} + p_{A,2,3} & p_{A,0,3} + 2\,p_{D,3,3} + p_{B,3,0} & p_{D,3,3} + 2\,p_{B,3,0} + p_{B,3,1} & p_{B,3,0} + 2\,p_{B,3,1} + p_{B,3,2} \end{bmatrix}. \tag{A.7}$$

$$\boldsymbol{p}_6 = \frac{1}{4}\begin{bmatrix} 2\,p_{D,3,3} + 2\,p_{A,0,3} & p_{A,0,3} + 2\,p_{D,3,3} + 2\,p_{B,3,0} & p_{D,3,3} + 2\,p_{B,3,0} + 2\,p_{B,3,1} & p_{B,3,1} + 2\,p_{B,3,2} + p_{B,3,3} \\ 2\,p_{A,0,3} + 2\,p_{A,1,3} & p_{D,3,3} + 2\,p_{A,0,3} + p_{A,1,3} & 2\,p_{D,3,3} + 2\,p_{A,0,3} & p_{A,0,3} + 2\,p_{D,3,3} + 2\,p_{B,3,0} \\ 2\,p_{A,1,3} + 2\,p_{A,2,3} & p_{A,0,3} + 2\,p_{A,1,3} + p_{A,2,3} & 2\,p_{A,0,3} + 2\,p_{A,1,3} & p_{D,3,3} + 2\,p_{A,0,3} + p_{A,1,3} \\ 2\,p_{A,2,3} + 2\,p_{A,3,3} & p_{A,1,3} + 2\,p_{A,2,3} + p_{A,3,3} & 2\,p_{A,1,3} + 2\,p_{A,2,3} & p_{A,0,3} + 2\,p_{A,1,3} + p_{A,2,3} \end{bmatrix}. \quad \text{(A.8)}$$

$$\boldsymbol{p}_7 = \frac{1}{4}\begin{bmatrix} 2\,p_{B,3,0} + 2\,p_{B,3,1} & 2\,p_{B,3,1} + 2\,p_{B,3,2} & 2\,p_{B,3,2} + 2\,p_{B,3,3} & 2\,p_{B,3,2} + 2\,p_{B,3,3} \\ p_{B,3,0} + 2\,p_{B,3,1} + p_{B,3,2} & p_{B,3,1} + 2\,p_{B,3,2} + p_{B,3,3} & p_{B,3,2} + 2\,p_{B,3,3} + p_{C,3,0} & p_{B,3,3} + 2\,p_{C,3,0} + p_{C,3,1} \\ 2\,p_{B,3,1} + 2\,p_{B,3,2} & 2\,p_{B,3,2} + 2\,p_{B,3,3} & 2\,p_{B,3,3} + 2\,p_{C,3,0} & 2\,p_{C,3,0} + 2\,p_{C,3,1} \\ p_{B,3,1} + 2\,p_{B,3,2} + p_{B,3,3} & p_{B,3,2} + 2\,p_{B,3,3} + p_{C,3,0} & p_{B,3,3} + 2\,p_{C,3,0} + p_{C,3,1} & p_{C,3,0} + 2\,p_{C,3,1} + p_{C,3,2} \end{bmatrix}. \quad \text{(A.9)}$$

$$\boldsymbol{p}_8 = \frac{1}{4}\begin{bmatrix} 2\,p_{A,0,3} + 2\,p_{A,1,3} & p_{A,0,3} + 2\,p_{A,1,3} + p_{A,2,3} & 2\,p_{A,1,3} + 2\,p_{A,2,3} & p_{A,1,3} + 2\,p_{A,2,3} + p_{A,3,3} \\ 2\,p_{A,1,3} + 2\,p_{A,2,3} & p_{A,1,3} + 2\,p_{A,2,3} + p_{A,3,3} & 2\,p_{A,2,3} + 2\,p_{A,3,3} & p_{A,2,3} + 3\,p_{A,3,3} \\ 2\,p_{A,2,3} + 2\,p_{A,3,3} & p_{A,2,3} + 3\,p_{A,3,3} & p_{A,3,3} & p_{A,3,3} \\ p_{A,3,3} & p_{A,3,3} & p_{A,3,3} & p_{A,3,3} \end{bmatrix}. \quad \text{(A.10)}$$

The four 16×16 prediction matrices are constructed as follows (the construction formulas are provided for notational brevity). For 16×16 vertical prediction (mode 0),

$$p_{0,ij} = p_{B,15,j} \quad \text{for } i,j = 0,\ldots,15 \; . \tag{A.11}$$

For 16×16 horizontal prediction (mode 1),

$$p_{1,ij} = p_{A,i,15} \quad \text{for } i,j = 0,\ldots,15 \; . \tag{A.12}$$

DC prediction (mode 2) assigns the average value of 32 surrounding pixels to all positions in the prediction matrix:

$$p_{2,ij} = \frac{\sum_{k=0}^{15} p_{A,k,15} + \sum_{k=0}^{15} p_{B,15,k}}{32} \quad \text{for } i,j = 0,\ldots,15 \; . \tag{A.13}$$

Plane prediction (mode 3) is constructed as follows.

$$p_{3,ij} = a + b\,(j-7) + c\,(i-7) \quad \text{for } i,j = 0,\ldots,15 \; , \tag{A.14}$$

and

$$a = 16\,(p_{A,15,15} + p_{B,15,15}) \tag{A.15}$$

$$b = \frac{5\,H}{64} \tag{A.16}$$

$$c = \frac{5\,V}{64} \; . \tag{A.17}$$

H and V are defined as

$$H = \sum_{k=0}^{6} (k+1)\,(p_{B,15,8+k} - p_{B,15,6-k}) + d \tag{A.18}$$

$$V = \sum_{k=0}^{6} (k+1)\,(p_{A,8+k,15} - p_{A,6-k,15}) + d \; , \tag{A.19}$$

with

$$d = 8\,(p_{B,15,15} - p_{D,15,15}) \; . \tag{A.20}$$

A.2 Transform-domain intra prediction formulas

A.2.1 Intra 4×4 prediction

The transform-domain formulas are obtained by applying the forward core transform to the standardized intra prediction formulas as given above, i.e.,

$$\boldsymbol{P}_m = T(\boldsymbol{p}_m) = \boldsymbol{C}_F \, \boldsymbol{p}_m \, \boldsymbol{C}_F^T \, , \tag{A.21}$$

This results in the following matrices.

$$\boldsymbol{P}_0 = 4 \begin{bmatrix} p_{B,3,0} + p_{B,3,1} + p_{B,3,2} + p_{B,3,3} & 0 & 0 & 0 \\ 2\left(p_{B,3,0} - p_{B,3,3}\right) + p_{B,3,1} - p_{B,3,2} & 0 & 0 & 0 \\ p_{B,3,0} - p_{B,3,1} - p_{B,3,2} + p_{B,3,3} & 0 & 0 & 0 \\ p_{B,3,0} - 2\left(p_{B,3,1} - p_{B,3,2}\right) - p_{B,3,3} & 0 & 0 & 0 \end{bmatrix}^T . \tag{A.22}$$

$$\boldsymbol{P}_1 = 4 \begin{bmatrix} p_{A,0,3} + p_{A,1,3} + p_{A,2,3} + p_{A,3,3} & 0 & 0 & 0 \\ 2\left(p_{A,0,3} - p_{A,3,3}\right) + p_{A,1,3} - p_{A,2,3} & 0 & 0 & 0 \\ p_{A,0,3} - p_{A,1,3} - p_{A,2,3} + p_{A,3,3} & 0 & 0 & 0 \\ p_{A,0,3} - 2\left(p_{A,1,3} - p_{A,2,3}\right) - p_{A,3,3} & 0 & 0 & 0 \end{bmatrix} . \tag{A.23}$$

$$\boldsymbol{P}_2 = 2 \begin{bmatrix} \sum_{i=0}^{3} p_{A,i,3} + \sum_{j=0}^{3} p_{B,3,j} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \tag{A.24}$$

Symmetry can still be found for the transform-domain diagonal down left prediction matrix:

$$\boldsymbol{P}_3 = \begin{bmatrix} \alpha & \beta & \gamma & \delta \\ \beta & \epsilon & \zeta & \eta \\ \gamma & \zeta & \theta & \iota \\ \delta & \eta & \iota & \kappa \end{bmatrix} , \tag{A.25}$$

where α through κ are defined as in Equation (A.27).
Similarly, for the transform-domain diagonal down right prediction matrix:

$$\boldsymbol{P}_4 = \begin{bmatrix} \alpha & -\beta & \gamma & -\delta \\ \beta & \epsilon & -\zeta & \eta \\ \gamma & \zeta & \theta & -\iota \\ \delta & \eta & \iota & \kappa \end{bmatrix} , \tag{A.26}$$

with α through κ being defined as in Equation (A.28).

For the other modes ($\boldsymbol{P}_5$ to $\boldsymbol{P}_8$), little or no symmetry is found in the matrices (Equation (A.29) to Equation (A.32)). For these modes, no gain in efficiency is obtained when compared to pixel-domain intra prediction followed by the forward integer transform.

$$
\begin{aligned}
\alpha &= 8\,(p_{B,3,2} + p_{C,3,2}) + 4\,p_{B,3,1} + p_{B,3,0} + 12\,(p_{B,3,3} + p_{C,3,1}) + 5\,p_{C,3,3} + 14\,p_{C,3,0} \\
\beta &= 10\,p_{B,3,2} + 2\,p_{B,3,0} + 7\,(p_{B,3,1} + p_{B,3,3} - p_{C,3,1}) - 9\,p_{C,3,3} - 10\,p_{C,3,2} \\
\gamma &= 2\,p_{B,3,1} + p_{B,3,0} - 2\,p_{B,3,3} - 2\,p_{C,3,1} + 3\,p_{C,3,3} - 2\,p_{C,3,0} \\
\delta &= p_{B,3,1} + p_{B,3,0} + p_{B,3,3} - p_{C,3,1} - 2\,p_{C,3,3} \\
\epsilon &= 9\,(p_{B,3,2} + p_{C,3,2}) + 4\,p_{B,3,0} + 12\,(p_{B,3,1} - p_{B,3,3} - p_{C,3,1}) + 16\,p_{C,3,3} - 26\,p_{C,3,0} \\
\zeta &= -4\,p_{B,3,2} + 3\,p_{B,3,1} + 2\,p_{B,3,0} - 9\,p_{B,3,3} + 9\,p_{C,3,1} - 5\,p_{C,3,3} + 4\,p_{C,3,2} \\
\eta &= -3\,p_{B,3,2} + p_{B,3,1} + 2\,p_{B,3,0} - p_{B,3,3} - p_{C,3,1} + 3\,p_{C,3,3} + 2\,p_{C,3,0} - 3\,p_{C,3,2} \\
\theta &= -4\,p_{B,3,2} + p_{B,3,0} + p_{C,3,3} + 6\,p_{C,3,0} - 4\,p_{C,3,2} \\
\iota &= -2\,p_{B,3,2} - p_{B,3,1} + p_{B,3,0} + 3\,p_{B,3,3} - 3\,p_{C,3,1} + 2\,p_{C,3,2} \\
\kappa &= p_{B,3,2} - 2\,p_{B,3,1} + p_{B,3,0} + 2\,p_{B,3,3} + 2\,p_{C,3,1} - p_{C,3,3} - 4\,p_{C,3,0} + p_{C,3,2}
\end{aligned}
\tag{A.27}
$$

$$
\begin{aligned}
\alpha &= 4\,p_{B,3,2} + 8\,p_{B,3,1} + 12\,p_{B,3,0} + 14\,p_{D,3,3} + p_{B,3,3} + 12\,p_{A,0,3} + 8\,p_{A,1,3} + 4\,p_{A,2,3} + p_{A,3,3} \\
\beta &= 7\,p_{B,3,2} + 10\,p_{B,3,1} + 7\,p_{B,3,0} + 2\,p_{B,3,3} - 7\,p_{A,0,3} - 10\,p_{A,1,3} - 7\,p_{A,2,3} - 2\,p_{A,3,3} \\
\gamma &= 2\,p_{B,3,2} - 2\,p_{B,3,0} - 2\,p_{D,3,3} + p_{B,3,3} - 2\,p_{A,0,3} + 2\,p_{A,2,3} + p_{A,3,3} \\
\delta &= p_{B,3,2} + p_{B,3,0} + p_{B,3,3} - p_{A,0,3} - p_{A,2,3} - p_{A,3,3} \\
\epsilon &= -12\,p_{B,3,2} - 9\,p_{B,3,1} + 12\,p_{B,3,0} + 26\,p_{D,3,3} - 4\,p_{B,3,3} + 12\,p_{A,0,3} - 9\,p_{A,1,3} - 12\,p_{A,2,3} - 4\,p_{A,3,3} \\
\zeta &= -3\,p_{B,3,2} + 4\,p_{B,3,1} + 9\,p_{B,3,0} - 2\,p_{B,3,3} - 9\,p_{A,0,3} - 4\,p_{A,1,3} + 3\,p_{A,2,3} + 2\,p_{A,3,3} \\
\eta &= -p_{B,3,2} + 3\,p_{B,3,1} + p_{B,3,0} - 2\,p_{D,3,3} - 2\,p_{B,3,3} + p_{A,0,3} + 3\,p_{A,1,3} - p_{A,2,3} - 2\,p_{A,3,3} \\
\theta &= -4\,p_{B,3,1} + 6\,p_{D,3,3} + p_{B,3,3} - 4\,p_{A,1,3} + p_{A,3,3} \\
\iota &= -p_{B,3,2} - 2\,p_{B,3,1} + 3\,p_{B,3,0} + p_{B,3,3} - 3\,p_{A,0,3} + 2\,p_{A,1,3} + p_{A,2,3} - p_{A,3,3} \\
\kappa &= 2\,p_{B,3,2} - p_{B,3,1} - 2\,p_{B,3,0} + 4\,p_{D,3,3} - p_{B,3,3} - 2\,p_{A,0,3} - p_{A,1,3} + 2\,p_{A,2,3} - p_{A,3,3}
\end{aligned}
\tag{A.28}
$$

$$
\begin{aligned}
P_{5,0,0} &= 10\,p_{B,3,2} + 15\,p_{B,3,1} + 16\,p_{B,3,0} + 11\,p_{D,3,3} + 3\,p_{B,3,3} + 5\,p_{A,0,3} + 3\,p_{A,1,3} + p_{A,2,3} \\
P_{5,0,1} &= -17\,p_{B,3,2} - 14\,p_{B,3,1} + 6\,p_{B,3,0} + 14\,p_{D,3,3} - 6\,p_{B,3,3} + 9\,p_{A,0,3} + 6\,p_{A,1,3} + 2\,p_{A,2,3} \\
P_{5,0,2} &= 4\,p_{B,3,2} - 5\,p_{B,3,1} - 8\,p_{B,3,0} - p_{D,3,3} + 3\,p_{B,3,3} + 3\,p_{A,0,3} + 3\,p_{A,1,3} + p_{A,2,3} \\
P_{5,0,3} &= -p_{B,3,2} + 3\,p_{B,3,1} - 2\,p_{B,3,0} - 3\,p_{D,3,3} - 3\,p_{B,3,3} + 2\,p_{A,0,3} + 3\,p_{A,1,3} + p_{A,2,3} \\
P_{5,1,0} &= 7\,p_{B,3,2} + 2\,p_{B,3,1} - 2\,p_{D,3,3} + 5\,p_{B,3,3} - 5\,p_{A,0,3} - 5\,p_{A,1,3} - 2\,p_{A,2,3} \\
P_{5,1,1} &= -9\,p_{B,3,2} + 13\,p_{B,3,1} + 21\,p_{B,3,0} + 7\,p_{D,3,3} - 10\,p_{B,3,3} - 8\,p_{A,0,3} - 10\,p_{A,1,3} - 4\,p_{A,2,3} \\
P_{5,1,2} &= -3\,p_{B,3,2} - 12\,p_{B,3,1} + 6\,p_{B,3,0} + 12\,p_{D,3,3} + 5\,p_{B,3,3} - p_{A,0,3} - 5\,p_{A,1,3} - 2\,p_{A,2,3} \\
P_{5,1,3} &= 8\,p_{B,3,2} - p_{B,3,1} - 7\,p_{B,3,0} + 11\,p_{D,3,3} - 5\,p_{B,3,3} + p_{A,0,3} - 5\,p_{A,1,3} - 2\,p_{A,2,3} \\
P_{5,2,0} &= -p_{B,3,1} - p_{D,3,3} + p_{B,3,3} - p_{A,0,3} + p_{A,1,3} + p_{A,2,3} \\
P_{5,2,1} &= p_{B,3,2} + 4\,p_{B,3,1} - 4\,p_{D,3,3} - 2\,p_{B,3,3} - 3\,p_{A,0,3} + 2\,p_{A,1,3} + 2\,p_{A,2,3} \\
P_{5,2,2} &= -2\,p_{B,3,2} - p_{B,3,1} + 4\,p_{B,3,0} - p_{D,3,3} + p_{B,3,3} - 3\,p_{A,0,3} + p_{A,1,3} + p_{A,2,3} \\
P_{5,2,3} &= 3\,p_{B,3,2} - 3\,p_{B,3,1} + 3\,p_{D,3,3} - p_{B,3,3} - 4\,p_{A,0,3} + p_{A,1,3} + p_{A,2,3} \\
P_{5,3,0} &= p_{B,3,2} + p_{B,3,1} - p_{D,3,3} - p_{A,2,3} \\
P_{5,3,1} &= -2\,p_{B,3,2} - p_{B,3,1} + 3\,p_{B,3,0} + p_{D,3,3} + p_{A,0,3} - 2\,p_{A,2,3} \\
P_{5,3,2} &= p_{B,3,2} - p_{B,3,1} - 2\,p_{B,3,0} + p_{D,3,3} + 2\,p_{A,0,3} - p_{A,2,3} \\
P_{5,3,3} &= -p_{B,3,2} + 2\,p_{B,3,1} - p_{B,3,0} - 2\,p_{D,3,3} + 3\,p_{A,0,3} - p_{A,2,3}
\end{aligned}
\tag{A.29}
$$

$$
\begin{aligned}
P_{6,0,0} &\equiv p_{B,3,2} + 3\,p_{B,3,1} + 5\,p_{B,3,0} + 11\,p_{D,3,3} + 16\,p_{A,0,3} + 15\,p_{A,1,3} + 10\,p_{A,2,3} + 3\,p_{A,3,3} \\
P_{6,0,1} &\equiv -2\,p_{B,3,2} - 5\,p_{B,3,1} - 5\,p_{B,3,0} - 2\,p_{D,3,3} + 2\,p_{A,1,3} + 7\,p_{A,2,3} + 5\,p_{A,3,3} \\
P_{6,0,2} &\equiv p_{B,3,2} + p_{B,3,1} - p_{B,3,0} - p_{D,3,3} - p_{A,1,3} + p_{A,3,3} \\
P_{6,0,3} &\equiv -p_{B,3,2} - p_{D,3,3} + p_{A,1,3} + p_{A,2,3} \\
P_{6,1,0} &\equiv 2\,p_{B,3,2} + 6\,p_{B,3,1} + 9\,p_{B,3,0} + 14\,p_{D,3,3} + 6\,p_{A,0,3} - 14\,p_{A,1,3} - 17\,p_{A,2,3} - 6\,p_{A,3,3} \\
P_{6,1,1} &\equiv -4\,p_{B,3,2} - 10\,p_{B,3,1} - 8\,p_{B,3,0} + 7\,p_{D,3,3} + 21\,p_{A,0,3} + 13\,p_{A,1,3} - 9\,p_{A,2,3} - 10\,p_{A,3,3} \\
P_{6,1,2} &\equiv 2\,p_{B,3,2} + 2\,p_{B,3,1} - 3\,p_{B,3,0} - 4\,p_{D,3,3} + 4\,p_{A,1,3} + p_{A,2,3} - 2\,p_{A,3,3} \\
P_{6,1,3} &\equiv -2\,p_{B,3,2} + p_{B,3,0} + p_{D,3,3} + 3\,p_{A,0,3} - p_{A,1,3} - 2\,p_{A,2,3} \\
P_{6,2,0} &\equiv p_{B,3,2} + 3\,p_{B,3,1} + 3\,p_{B,3,0} - p_{D,3,3} - 8\,p_{A,0,3} - 5\,p_{A,1,3} + 4\,p_{A,2,3} + 3\,p_{A,3,3} \\
P_{6,2,1} &\equiv -2\,p_{B,3,2} - 5\,p_{B,3,1} - p_{B,3,0} + 12\,p_{D,3,3} + 6\,p_{A,0,3} - 12\,p_{A,1,3} - 3\,p_{A,2,3} + 5\,p_{A,3,3} \\
P_{6,2,2} &\equiv p_{B,3,2} + p_{B,3,1} - 3\,p_{B,3,0} - p_{D,3,3} + 4\,p_{A,0,3} - p_{A,1,3} - 2\,p_{A,2,3} + p_{A,3,3} \\
P_{6,2,3} &\equiv -p_{B,3,2} + 2\,p_{B,3,0} + p_{D,3,3} - 2\,p_{A,0,3} - p_{A,1,3} + p_{A,2,3} \\
P_{6,3,0} &\equiv p_{B,3,2} + 3\,p_{B,3,1} + 2\,p_{B,3,0} - 3\,p_{D,3,3} - 2\,p_{A,0,3} + 3\,p_{A,1,3} - p_{A,2,3} - 3\,p_{A,3,3} \\
P_{6,3,1} &\equiv -2\,p_{B,3,2} - 5\,p_{B,3,1} + p_{B,3,0} + 11\,p_{D,3,3} - 7\,p_{A,0,3} - p_{A,1,3} + 8\,p_{A,2,3} - 5\,p_{A,3,3} \\
P_{6,3,2} &\equiv p_{B,3,2} + p_{B,3,1} - 4\,p_{B,3,0} + 3\,p_{D,3,3} - 3\,p_{A,1,3} + 3\,p_{A,2,3} - p_{A,3,3} \\
P_{6,3,3} &\equiv -p_{B,3,2} + 3\,p_{B,3,0} - 2\,p_{D,3,3} - p_{A,0,3} + 2\,p_{A,1,3} - p_{A,2,3}
\end{aligned}
\tag{A.30}
$$

$$
\begin{aligned}
P_{7,0,0} &= 15\,p_{B,3,2} + 10\,p_{B,3,1} + 3\,p_{B,3,0} + 16\,p_{B,3,3} + 6\,p_{C,3,1} + 13\,p_{C,3,0} + p_{C,3,2} \\
P_{7,0,1} &= 14\,p_{B,3,2} + 17\,p_{B,3,1} + 6\,p_{B,3,0} - 6\,p_{B,3,3} - 11\,p_{C,3,1} - 18\,p_{C,3,0} - 2\,p_{C,3,2} \\
P_{7,0,2} &= -5\,p_{B,3,2} + 4\,p_{B,3,1} + 3\,p_{B,3,0} - 8\,p_{B,3,3} + 4\,p_{C,3,1} + p_{C,3,0} + p_{C,3,2} \\
P_{7,0,3} &= -3\,p_{B,3,2} + p_{B,3,1} + 3\,p_{B,3,0} + 2\,p_{B,3,3} - 3\,p_{C,3,1} + p_{C,3,0} - p_{C,3,2} \\
P_{7,1,0} &= 2\,p_{B,3,2} + 7\,p_{B,3,1} + 5\,p_{B,3,0} - 7\,p_{C,3,1} - 5\,p_{C,3,0} - 2\,p_{C,3,2} \\
P_{7,1,1} &= -13\,p_{B,3,2} + 9\,p_{B,3,1} + 10\,p_{B,3,0} - 21\,p_{B,3,3} + 12\,p_{C,3,1} - p_{C,3,0} + 4\,p_{C,3,2} \\
P_{7,1,2} &= -12\,p_{B,3,2} - 3\,p_{B,3,1} + 5\,p_{B,3,0} + 6\,p_{B,3,3} - 3\,p_{C,3,1} + 9\,p_{C,3,0} - 2\,p_{C,3,2} \\
P_{7,1,3} &= p_{B,3,2} - 8\,p_{B,3,1} + 5\,p_{B,3,0} + 7\,p_{B,3,3} + p_{C,3,1} - 8\,p_{C,3,0} + 2\,p_{C,3,2} \\
P_{7,2,0} &= -p_{B,3,2} + p_{B,3,0} - p_{C,3,0} + p_{C,3,2} \\
P_{7,2,1} &= -4\,p_{B,3,2} - p_{B,3,1} + 2\,p_{B,3,0} + p_{C,3,1} + 4\,p_{C,3,0} - 2\,p_{C,3,2} \\
P_{7,2,2} &= -p_{B,3,2} - 2\,p_{B,3,1} + p_{B,3,0} + 4\,p_{B,3,3} - 2\,p_{C,3,1} - p_{C,3,0} + p_{C,3,2} \\
P_{7,2,3} &= 3\,p_{B,3,2} - 3\,p_{B,3,1} + p_{B,3,0} + 3\,p_{C,3,1} - 3\,p_{C,3,0} - p_{C,3,2} \\
P_{7,3,0} &= p_{B,3,2} + p_{B,3,1} - p_{C,3,1} - p_{C,3,2} \\
P_{7,3,1} &= p_{B,3,2} + 2\,p_{B,3,1} - 3\,p_{B,3,3} + p_{C,3,1} - 3\,p_{C,3,0} + 2\,p_{C,3,2} \\
P_{7,3,2} &= -p_{B,3,2} + p_{B,3,1} - 2\,p_{B,3,3} + p_{C,3,1} + 2\,p_{C,3,0} - p_{C,3,2} \\
P_{7,3,3} &= -2\,p_{B,3,2} + p_{B,3,1} + p_{B,3,3} - 2\,p_{C,3,1} + p_{C,3,0} + p_{C,3,2}
\end{aligned}
\tag{A.31}
$$

$$
\begin{aligned}
P_{8,0,0} &= 3\,p_{A,0,3} + 10\,p_{A,1,3} + 15\,p_{A,2,3} + 36\,p_{A,3,3} \\
P_{8,0,1} &= 5\,p_{A,0,3} + 7\,p_{A,1,3} + 2\,p_{A,2,3} - 14\,p_{A,3,3} \\
P_{8,0,2} &= p_{A,0,3} - p_{A,2,3} \\
P_{8,0,3} &= p_{A,1,3} + p_{A,2,3} - 2\,p_{A,3,3} \\
P_{8,1,0} &= 6\,p_{A,0,3} + 17\,p_{A,1,3} + 14\,p_{A,2,3} - 37\,p_{A,3,3} \\
P_{8,1,1} &= 10\,p_{A,0,3} + 9\,p_{A,1,3} - 13\,p_{A,2,3} - 6\,p_{A,3,3} \\
P_{8,1,2} &= 2\,p_{A,0,3} - p_{A,1,3} - 4\,p_{A,2,3} + 3\,p_{A,3,3} \\
P_{8,1,3} &= 2\,p_{A,1,3} + p_{A,2,3} - 3\,p_{A,3,3} \\
P_{8,2,0} &= 3\,p_{A,0,3} + 4\,p_{A,1,3} - 5\,p_{A,2,3} - 2\,p_{A,3,3} \\
P_{8,2,1} &= 5\,p_{A,0,3} - 3\,p_{A,1,3} - 12\,p_{A,2,3} + 10\,p_{A,3,3} \\
P_{8,2,2} &= p_{A,0,3} - 2\,p_{A,1,3} - p_{A,2,3} + 2\,p_{A,3,3} \\
P_{8,2,3} &= p_{A,1,3} - p_{A,2,3} \\
P_{8,3,0} &= 3\,p_{A,0,3} + p_{A,1,3} - 3\,p_{A,2,3} - p_{A,3,3} \\
P_{8,3,1} &= 5\,p_{A,0,3} - 8\,p_{A,1,3} + p_{A,2,3} + 2\,p_{A,3,3} \\
P_{8,3,2} &= p_{A,0,3} - 3\,p_{A,1,3} + 3\,p_{A,2,3} - p_{A,3,3} \\
P_{8,3,3} &= p_{A,1,3} - 2\,p_{A,2,3} + p_{A,3,3}
\end{aligned}
\tag{A.32}
$$

A.2.2 Intra 16×16 prediction

A. Mode 0: vertical prediction

For the case of vertical 16×16 prediction, the transform-domain compensation matrix $\boldsymbol{P}_{0,DC}$ is obtained:

$$\boldsymbol{P}_{0,DC} = 8 \begin{bmatrix} \alpha+\beta+\gamma+\delta & 0 & 0 & 0 \\ \alpha+\beta-\gamma-\delta & 0 & 0 & 0 \\ \alpha-\beta-\gamma+\delta & 0 & 0 & 0 \\ \alpha-\beta+\gamma-\delta & 0 & 0 & 0 \end{bmatrix}^T , \tag{A.33}$$

where

$$\alpha = \sum_{j=0}^{3} p_{B,15,j} \ , \quad \beta = \sum_{j=4}^{7} p_{B,15,j} \ ,$$

$$\gamma = \sum_{j=8}^{11} p_{B,15,j} \ , \quad \delta = \sum_{j=12}^{15} p_{B,15,j} \ .$$

The compensation matrices $\boldsymbol{P}_{0,AC,l}$ become:

$$\boldsymbol{P}_{0,AC,l} = 4 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2\left(p_{B,15,k} - p_{B,15,k+3}\right) + p_{B,15,k+1} - p_{B,15,k+2} & 0 & 0 & 0 \\ p_{B,15,k} - p_{B,15,k+1} - p_{B,15,k+2} + p_{B,15,k+3} & 0 & 0 & 0 \\ p_{B,15,k} - 2\left(p_{B,15,k+1} - p_{B,15,k+2}\right) - p_{B,15,k+3} & 0 & 0 & 0 \end{bmatrix}^T , \tag{A.34}$$

where

$$k = 4\,(l \bmod 4) \quad \text{for } l = 0, 1, \ldots, 15 \ . \tag{A.35}$$

As a result, the matrices $\boldsymbol{P}_{0,AC,l}$ are identical for the four 4×4 blocks in every column of the macroblock.

B. Mode 1: horizontal prediction

For horizontal 16×16 prediction (mode 1), the transform-domain compensation matrix $\boldsymbol{P}_{1,DC}$ for the Hadamard DC matrix becomes:

$$\boldsymbol{P}_{1,DC} = 8 \begin{bmatrix} \alpha+\beta+\gamma+\delta & 0 & 0 & 0 \\ \alpha+\beta-\gamma-\delta & 0 & 0 & 0 \\ \alpha-\beta-\gamma+\delta & 0 & 0 & 0 \\ \alpha-\beta+\gamma-\delta & 0 & 0 & 0 \end{bmatrix} , \tag{A.36}$$

where

$$\alpha = \sum_{i=0}^{3} p_{A,i,15} \ , \quad \beta = \sum_{i=4}^{7} p_{A,i,15} \ ,$$

$$\gamma = \sum_{i=8}^{11} p_{A,i,15} \ , \quad \delta = \sum_{i=12}^{15} p_{A,i,15} \ ,$$

with $p_{A,i,15}$, $i = 0, \ldots, 15$ being the 16 pixels in the rightmost column of the macroblock to the left of the current macroblock. The compensation matrices $\boldsymbol{P}_{1,AC,l}$ become:

$$\boldsymbol{P}_{1,AC,l} = 4 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2\left(p_{A,k,15} - p_{A,k+3,15}\right) + p_{A,k+1,15} - p_{A,k+2,15} & 0 & 0 & 0 \\ p_{A,k,15} - p_{A,k+1,15} - p_{A,k+2,15} + p_{A,k+3,15} & 0 & 0 & 0 \\ p_{A,k,15} - 2\left(p_{A,k+1,15} - p_{A,k+2,15}\right) - p_{A,k+3,15} & 0 & 0 & 0 \end{bmatrix} , \tag{A.37}$$

with

$$k = 4\,(l \gg 2) \quad \text{for } l = 0, 1, \ldots, 15 \ . \tag{A.38}$$

As a result, the matrices $\boldsymbol{P}_{1,AC,l}$ are identical for the four 4×4 blocks in every row of the macroblock.

C. Mode 2: DC prediction

For the DC mode (mode 2) it suffices to compensate the top-left position in the transform-domain DC matrix $\boldsymbol{P}_{0,DC}$, while no compensation is required for the 16 AC 4×4 matrices (all zero matrices).

$$\boldsymbol{P}_{0,DC} = 4 \begin{bmatrix} \sum_{i=0}^{15} p_{A,i,15} + \sum_{j=0}^{15} p_{B,15,j} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \tag{A.39}$$

D. Mode 3: plane prediction

Computationally, the most complex mode is the plane prediction mode. By applying the forward transform to the H.264/AVC prediction formulas, the following is obtained. Similarly to the pixel-domain formulas, let a, b, c and d be defined as:

$$a = 64(p_{B,15,15} + p_{C,15,15}) \tag{A.40}$$

$$b = 5\left[\sum_{i=0}^{6}(i+1)(p_{B,15,6-i} - p_{B,15,8+i}) + d\right] \tag{A.41}$$

$$c = 5\left[\sum_{j=0}^{6}(j+1)(p_{C,6-j,15} - p_{C,8+j,15}) + d\right] . \tag{A.42}$$

with

$$d = 8(p_{C,15,15} - p_{D,15,15}) \tag{A.43}$$

The following transform-domain DC compensation matrix is obtained:

$$\boldsymbol{P}_{3,DC} = \begin{bmatrix} a - 3/32b - 3/32c & b/4 & 0 & b/8 \\ c/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ c/8 & 0 & 0 & 0 \end{bmatrix} . \tag{A.44}$$

Further, let

$$b_2 = b/512 \tag{A.45}$$

$$c_2 = c/512 . \tag{A.46}$$

Then the transform-domain compensation matrices for the 16 4×4 blocks are all identical and are obtained as follows:

$$\boldsymbol{P}_{3,AC,l} = \begin{bmatrix} 0 & 7b_2 & 0 & b_2 \\ 7c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ c_2 & 0 & 0 & 0 \end{bmatrix} \quad \text{for } l = 0, 1, \ldots, 15 . \tag{A.47}$$

As it turns out, the frequency-domain matrices for the plane prediction can be obtained at a minimal computational cost. In the pixel domain, apart from the derivation of $a, b,$ and c, every position requires two multiplications. Hence, operation in the frequency domain reduces operations due to the elimination of the forward and inverse transforms (both DCT and Hadamard), and due to the sparseness of the frequency-domain compensation matrices.

Appendix B

H.264/AVC macroblock and submacroblock types

In the tables below, an overview is given of the H.264/AVC macroblock and submacroblock types. The numbers given in the first column of the tables correspond to the *mb type* or *sub mb type* syntax elements, which specify the selected type in the bitstream.

Table B.1: Macroblock types for P pictures.

mb type	Name	# of partitions
0	P_L0_16×16	1
1	P_L0_L0_16×8	2
2	P_L0_L0_8×16	2
3	P_8×8	4
-	P_Skip	1

Table B.2: Submacroblock types for P pictures.

sub mb type	Name	# of partitions
0	P_L0_8×8	1
1	P_L0_8×4	2
2	P_L0_4×8	2
3	P_L0_4×4	4

Table B.3: Macroblock types for B pictures.

mb type	Name	# of partitions
0	B_Direct_16×16	n/a
1	B_L0_16×16	1
2	B_L1_16×16	1
3	B_Bi_16×16	1
4	B_L0_L0_16×8	2
5	B_L0_L0_8×16	2
6	B_L1_L1_16×8	2
7	B_L1_L1_8×16	2
8	B_L0_L1_16×8	2
9	B_L0_L1_8×16	2
10	B_L1_L0_16×8	2
11	B_L1_L0_8×16	2
12	B_L0_Bi_16×8	2
13	B_L0_Bi_8×16	2
14	B_L1_Bi_16×8	2
15	B_L1_Bi_8×16	2
16	B_Bi_L0_16×8	2
17	B_Bi_L0_8×16	2
18	B_Bi_L1_16×8	2
19	B_Bi_L1_8×16	2
20	B_Bi_Bi_16×8	2
21	B_Bi_Bi_8×16	2
22	B_8×8	4
-	B_Skip	n/a

Table B.4: Submacroblock types for B pictures.

sub mb type	Name	# of partitions
0	B_Direct_8×8	4
1	B_L0_8×8	1
2	B_L1_8×8	1
3	B_Bi_8×8	1
4	B_L0_8×4	2
5	B_L0_4×8	2
6	B_L1_8×4	2
7	B_L1_4×8	2
8	B_Bi_8×4	2
9	B_Bi_4×8	2
10	B_L0_4×4	4
11	B_Bi_4×4	4
12	B_Bi_4×4	4

Publications

Journal Papers

1. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Architectures for fast transcoding of H.264/AVC to quality-scalable SVC streams. *IEEE Transactions on Multimedia*. 2009. November 2009.

2. Peter Lambert, Pedro Debevere, Jan De Cock, Jean-Franois Macq, Natalie Degrande, Danny De Vleeschauwer, and Rik Van de Walle. Real-time error concealing bitstream adaptation methods for SVC in IPTV systems. Journal of Real-time Image Processing. 2009. 79-90.

3. Stijn Notebaert, Jan De Cock, Samie Beheydt, Jan De Lameillieure, and Rik Van de Walle. Mixed architectures for H.264/AVC digital video transrating. *Springer journal on Multimedia Tools and Applications*. August 2009.

4. Sarah De Bruyne, Davy Van Deursen, Jan De Cock, Wesley De Neve, Peter Lambert, and Rik Van de Walle. A compressed-domain approach for shot boundary detection on H.264/AVC bit streams. Signal Processing: Image Communication. 2008. 473 - 498.

5. Jan De Cock, Stijn Notebaert, Peter Lambert, Davy De Schrijver, and Rik Van de Walle, Requantization transcoding in pixel and frequency domain for intra 16x16 in H.264/AVC. In *Lecture Notes in Computer Science*. August 2006.

Submitted Journal Papers

1. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Requantization transcoding for H.264/AVC video coding. Submitted to *Elsevier journal on Signal Processing: Image Communication*.

2. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Motion-refined rewriting of H.264/AVC-coded video to SVC streams. Submitted to *Elsevier Journal of Visual Communication and Image Representation*.

3. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Spatial Resolution Reduction Transcoding with Dynamic Complexity for H.264/AVC. Submitted to *Multimedia Systems Journal*.

Papers in conference proceedings

1. Jan De Cock, Stijn Notebaert, Kenneth Vermeirsch, Peter Lambert, and Rik Van de Walle. Transcoding of H.264/AVC to SVC with Motion Data Refinement. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. November 2009.

2. Kenneth Vermeirsch, Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Evaluation of transform performance when using shape-adaptive partitioning in video coding. In *Proceedings of the 27th Picture Coding Symposium (PCS)*. May 2009.

3. Stijn Notebaert, Jan De Cock, Kenneth Vermeirsch, Peter Lambert, and Rik Van de Walle. Leveraging the quantization offset for improved requantization transcoding of H.264/AVC video. In *Proceedings of the 27th Picture Coding Symposium (PCS)*. May 2009.

4. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Advanced bitstream rewriting from H.264/AVC to SVC. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. October 2008.

5. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Efficient spatial resolution reduction transcoding for H.264/AVC. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. October 2008.

6. Stijn Notebaert, Jan De Cock, Kenneth Vermeirsch, Peter Lambert and Rik Van de Walle. Improved dynamic rate shaping for H.264/AVC video streams. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. October 2008.

7. Kenneth Vermeirsch, Jan De Cock, Stijn Notebaert, Peter Lambert and Rik Van de Walle. Extended Macroblock Bipartitioning Modes for

H.264/AVC Inter Coding. In *Proceedings of the 10th IEEE International Symposium on Multimedia (ISM)*. December 2008. 142-147.

8. Kenneth Vermeirsch, Jan De Cock, Stijn Notebaert, Peter Lambert and Rik Van de Walle. Increased Flexebility in Inter Picture Partitioning. In *Proceedings of the 2008 IEEE 10th workshop on Multimedia Signal Processing (MMSP)*. October 2008. 284-288.

9. Stijn Notebaert, Jan De Cock, Peter Lambert, and Rik Van de Walle. Rate-controlled Requantization Transcoding for H.264/AVC video streams. In *Proceedings of SPIE Optics and Photonics*. Vol. 7073. August 2008.

10. Maarten Wijnants, Wim Lamotte, Bart De Vleeschauwer, Filip De Turck, Bart Dhoedt, Piet Demeester, Peter Lambert, Dieter Van de Walle, Jan De Cock, Stijn Notebaert and Rik Van de Walle. Optimizing User QoE through Overlay Routing, Bandwith Management and Dynamic Transcoding. In *Proceedings of the 9th International Symposium on a World of Wireless, Mobile and Multimedia Networks*. June 2008.

11. Jan De Cock, Stijn Notebaert, and Rik Van de Walle. Transcoding from H.264/AVC to SVC with CGS layers. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. September 2007.

12. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Bridging the Gap: Transcoding from Single-Layer H.264/AVC to Scalable SVC Video Streams. In *Proceedings of the 9th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)*. Vol. 4678. August 2007.

13. Jan Lievens, Dieter Van de Walle, Jan De Cock, Joeri Barbarien, Rik Van de Walle, and Peter Schelkens. Low-complexity MPEG-2 to H.264 transcoding. In *Proceedings of SPIE Optics and Photonics*. August 2007.

14. Stijn Notebaert, Jan De Cock, Peter Lambert, and Rik Van de Walle. Requantization Transcoding for Reduced-Complexity H.264/AVC Video Coding Applications. In *Proceedings of the ninth IASTED international conference on Signal and Image Processing (SIP)*. Vol. 576. August 2007. 299-304.

15. Jan De Cock, Stijn Notebaert, and Rik Van de Walle. Combined SNR and temporal scalability for H.264/AVC using requantization transcod-

ing and hierarchical B pictures. In *Proceedings of IEEE International Conference on Multimedia & Expo (ICME)*. July 2007.

16. Stijn Notebaert, Jan De Cock and Rik Van de Walle. Improved H.264/AVC requantization transcoding using low-complexity interpolation filters for 1/4-Pixel motion compensation. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*. April 2007.

17. Jan De Cock, Stijn Notebaert, and Rik Van de Walle. A novel hybrid requantization transcoding scheme for H.264/AVC. In *Proceedings of the International Symposium on Signal Processing and its Applications (ISSPA)*. February 2007.

18. Stijn Notebaert, Jan De Cock, Kenneth Vermeirsch, and Rik Van de Walle. Complexity and Quality Assessment of MPEG-2 to H.264/AVC Intra Transcoding Architectures. In *Proceedings of the 9th International Symposium on Signal Processing and its Applications (ISSPA)*. February 2007.

19. Koen De Wolf, Davy De Schrijver, Jan De Cock, Wesley De Neve, and Rik Van de Walle. Performance Evaluation of Adaptive Residual Interpolation, a Tool for Inter-layer Prediction in H.264/AVC Scalable Video Coding. In *Proceedings of the 15th Scandinavian Conference (SCIA)*. Vol. 4522. June 2007. 740-749.

20. Jan De Cock, Stijn Notebaert, Koen De Wolf, Peter Lambert, and Rik Van de Walle. Low-complexity SNR transcoding for H.264/AVC. In *Proceedings of the Fourth IASTED International Conference on Communications, Internet and Information Technology (CIIT)*. November 2006.

21. Stijn Notebaert, Jan De Cock, Koen De Wolf, and Rik Van de Walle. Requantization transcoding of H.264/AVC bitstreams for intra 4x4 prediction modes. In *Lecture Notes in Computer Science*. November 2006.

22. Stijn Notebaert, Jan De Cock, Davy De Schrijver, Koen De Wolf, and Rik Van de Walle. Quality Analysis of Requantization Transcoding Architectures for H.264/AVC. In *Proceedings of SPIE Optics and Photonics*. August 2006.

23. Davy De Schrijver, Wesley De Neve, Davy Van Deursen, Jan De Cock, and Rik Van de Walle. On an evaluation of transformation languages

in a fully XML-driven framework for video content adaptation. In *Proceedings of the first international conference on innovative computing, information and control (ICICIC*). August 2006. 213-216.

24. Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. Hardware/software co-design for H.264/AVC intra frame encoding. In *Proceedings of Euromedia 2006*. May 2006. 56-60.

25. Wesley De Neve, Dieter Van Rijsselbergen, Charles Hollemeersch, Jan De Cock, Stijn Notebaert and Rik Van de Walle. GPU-Assisted Decoding of Video Samples Represented in the YCoCg-R Color Space. In *Proceedings of the 13th ACM International Conference on Multimedia*. November 2005. 447-450.

26. Koen De Wolf, Yves Dhondt, Jan De Cock, and Rik Van de Walle. Complexity Analysis of Interpolation Filters for Scalable Video Coding. In *Proceedings of Euromedia*. April 2005. 93-96.

References

[1] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), ITU-T and ISO/IEC JTC 1. *Advanced Video Coding for Generic Audiovisual Services*, Version 1: May 2003, Version 2: May 2004, Version 3: Mar. 2005, Version 4: Sept. 2005, Version 5 and Version 6: June 2006, Version 7: Apr. 2007, Version 8 (including SVC extension): November 2007.

[2] ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2 Video), ITU-T and ISO/IEC JTC 1. *Information technology - Generic coding of moving pictures and associated audio information: Video*, February 2002.

[3] Enzo Castelli. Standards conversion and chrominance transcoding problems in the exchange of television programs. *IEEE Transactions on Broadcast and Television Receivers*, 12(2):43–54, May 1966.

[4] Gertjan Keesman, Robert Hellinghuizen, Fokke Hoeksema, and Geert Heideman. Transcoding of MPEG bitstreams. *Signal Processing: Image Communication*, 8(6):481–500, September 1996.

[5] Anthony Vetro, Charilaos Christopoulos, and Huifang Sun. Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, pages 18–29, March 2003.

[6] Pedro A. A. Assunção and Mohammed Ghanbari. A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(8):953–967, December 1998.

[7] Damien Lefol, Dave Bull, and Nishan Canagarajah. Performance evaluation of transcoding algorithms for H.264. *IEEE Transactions on Consumer Electronics*, 52(1):215–222, February 2006.

[8] Xiaoming Sun and Pin Tao. Rapid algorithms for MPEG-2 to H.264 transcoding. In *Proceedings of the Pacific Rim Conference on Multimedia (PCM)*, November 2005.

[9] Hari Kalva, Branko Petljanski, and Borko Furht. Complexity reduction tools for MPEG-2 to H.264 video transcoding. In *WSEAS Transactions on Information Science & Applications*, March 2005.

[10] Chen Chen, Ping-Hao Wu, and Homer Chen. MPEG-2 to H.264 transcoding. In *Proceedings of the Picture Coding Symposium (PCS)*, December 2004.

[11] Yeping Su, Jun Xin, Anthony Vetro, and Huifang Sun. Efficient MPEG-2 to H.264/AVC intra transcoding in transform-domain. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2005.

[12] Tuanjie Qian, Jun Sun, Dian Li, Xiaokang Yang, and Jia Wang. Transform domain transcoding from MPEG-2 to H.264 with interpolation drift-error compensation. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(4):523–534, April 2006.

[13] Haruhisa Kato, Akio Yoneyama, Yasuhiro Takishima, and Yosuke Kaji. Coding mode decision for high quality MPEG-2 to H.264 transcoding. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2007.

[14] Sandro Moiron, Sérgio M. M. de Faria, Pedro A. A. Assunção, Vítor M. M. da Silva, and Antonio Navarro. Fast interframe transcoding from H.264 to MPEG-2. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2007.

[15] Kai-Tat Fung and Wan-Chi Siu. Low complexity H.263 to H.264 video transcoding using motion vector decomposition. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2005.

[16] Kai-Tat Fung and Wan-Chi Siu. Conversion between DCT coefficients and IT coefficients in the compressed domain for H.263 to H.264 video transcoding. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2005.

[17] Gerardo Fernandez-Escribano, Jens Bialkowski, Hari Kalva, Pedro Cuenca, Luis Orozco-Barbosa, and André Kaup. H.263 to H.264 transcoding using data mining. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2007.

[18] Gerardo Fernandez-Escribano, Jens Bialkowski, José A. Gamez, Hari Kalva, Pedro Cuenca, Luis Orozco-Barbosa, and André Kaup. Low-complexity heterogeneous video transcoding using data mining. *IEEE Transactions on Multimedia*, 10(2):286–299, February 2008.

[19] Jens Bialkowski, Marcus Barkowsky, Florian Leschka, and André Kaup. Low-complexity transcoding of inter coded video frames from H.264 to H.263. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2006.

[20] Anthony Vetro, Huifang Sun, Paul DaGraca, and Tommy Poon. Minimum drift architectures for 3-layer scalable DTV decoding. *IEEE Transactions on Consumer Electronics*, 44(3):527–536, August 1998.

[21] Anthony Vetro and Huifang Sun. Frequency domain down-conversion of HDTV using an optimal motion compensation scheme. *Journal of Imaging Science and Technology*, 1998.

[22] Jenq-Neng Hwang and Tzong-Der Wu. Motion vector re-estimation and dynamic frame-skipping for video transcoding. In *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers*, November 1998.

[23] Jenq-Neng Hwang, Tzong-Der Wu, and Chia-Wen Lin. Dynamic frame-skipping in video transcoding. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, December 1998.

[24] Kai-Tat Fung, Yui-Lam Chan, and Wan-Chi Siu. Dynamic frame skipping for high-performance transcoding. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2001.

[25] Kai-Tat Fung, Yui-Lam Chan, and Wan-Chi Siu. New architecture for dynamic frame-skipping transcoder. *IEEE Transactions on Image Processing*, 11(8):886–900, August 2002.

[26] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Analysis of hierarchical B pictures and MCTF. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, July 2006.

[27] Alexandros Eleftheriadis and Dimitris Anastassiou. Constrained and general dynamic rate shaping of compressed digital video. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 1995.

[28] Alexandros Eleftheriadis and Pankaj Batra. Dynamic rate shaping of compressed digital video. *IEEE Transactions on Multimedia*, 8(2):297–314, February 2006.

[29] Huifang Sun, Wilson Kwok, and Joel Zdepski. Architectures for MPEG compressed bitstream scaling. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(2):191–199, April 1996.

[30] Yasuyuki Nakajima, Hironao Hori, and Tamotsu Kanoh. Rate conversion of MPEG coded video by re-quantization process. In *IEEE International Conference on Image Processing (ICIP)*, October 1995.

[31] Pedro A. A. Assunção and Mohammed Ghanbari. Transcoding of MPEG-2 video in the frequency domain. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 1997.

[32] Thomas Wiegand, Gary Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[33] Damien Lefol and Dave Bull. Mode refinement algorithm for H.264 inter frame requantization. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2006.

[34] Damien Lefol, Dave Bull, and Nishan Canagarajah. Mode refinement algorithm for H.264 intra frame requantization. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006.

[35] Peng Zhang, Yan Lu, Qingming Huang, and Wen Gao. Mode mapping method for H.264/AVC spatial downscaling transcoding. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2004.

[36] Huifeng Shen, Xiaoyan Sun, Feng Wu, Houqiang Li, and Shipeng Li. A fast downsizing video transcoder for H.264/AVC with rate-distortion optimal mode decision. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, July 2006.

[37] Chih-Hung Li, Chung-Neng Wang, and Tihao Chiang. A multiple-window video embedding transcoder based on H.264/AVC standard. *EURASIP Journal on Advances in Signal Processing*, 2007.

[38] Bo Shen. Perfect requantization for video transcoding. *Multimedia Tools and Applications*, 35(2):163–173, November 2007.

[39] Bo Shen. Optimal requantization-based rate adaptation for H.264. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, July 2006.

[40] Henrique S. Malvar, Antti Hallapuro, Marta Karczewicz, and Louis Kerofsky. Low-complexity transform and quantization in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):598–603, July 2003.

[41] Antti Hallapuro, Marta Karczewicz, and Henrique Malvar. *Low Complexity Transform and Quantization - Part I: Basic Implementation*. Joint Video Team, Doc. JVT-B038, Geneva, Switzerland, January 2002.

[42] Joint Video Team. *Joint Model reference software*. `http://iphome.hhi.de/suehring/tml/index.htm`.

[43] Oliver Werner. Requantization for transcoding of MPEG-2 intraframes. *IEEE Transactions on Image Processing*, 8(2):179–191, February 1999.

[44] Bo Shen. Modeled analysis on requantization error. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, October 2005.

[45] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.

[46] Ishfaq Ahmad, Xiaohui Wei, Yu Sun, and Ya-Qin Zhang. Video Transcoding: an Overview of Various Techniques and Research Issues. *IEEE Transactions on Multimedia*, 7(5):793–804, October 2005.

[47] Thomas Wiegand, Xiaozheng Zhang, and Bernd Girod. Long-term memory motion-compensated prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):70–84, February 1999.

[48] Sarah De Bruyne, Davy Van Deursen, Jan De Cock, Wesley De Neve, Peter Lambert, and Rik Van de Walle. A compressed-domain approach for shot boundary detection on H.264/AVC bit streams. *Signal Processing: Image Communication*, 23(7):473–498, August 2008.

[49] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[50] Peter List, Anthony Joch, Jani Lainema, Gisle Bjøntegaard, and Marta Karczewicz. Adaptive deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):614–619, July 2003.

[51] Shigenobu Minami and Avideh Zakhor. An optimization approach for removing blocking effects in transform coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(2):74–82, April 1995.

[52] George A. Triantafyllidis, Dimitrios Tzovaras, and Michael G. Strintzis. Blocking artifact reduction in frequency domain. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2001.

[53] George A. Triantafyllidis, Dimitrios Tzovaras, and Michael G. Strintzis. Blocking artifact detection and reduction in compressed data. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(10):877–890, October 2002.

[54] Bo Shen. Efficient deblocking and optimal quantizer selection for video transcoding. In *IEEE International Conference on Image Processing (ICIP)*, September 2003.

[55] Huifang Sun, Xuemin Chen, and Tihao Chiang. *Digital video transcoding for transmission and storage*. CRC Press, 2005.

[56] Jan De Cock, Stijn Notebaert, and Rik Van de Walle. A Novel Hybrid Requantization Transcoding Scheme for H.264/AVC. In *Proceedings of the International Symposium on Signal Processing and its Applications (ISSPA)*, February 2007.

[57] Chen Chen, Ping-Hao Wu, and Homer Chen. Transform-domain intra prediction for H.264. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2005.

[58] Stijn Notebaert, Jan De Cock, Koen De Wolf, and Rik Van de Walle. Requantization transcoding of H.264/AVC bitstreams for Intra 4x4 prediction modes. In *Proceedings of the Pacific Rim Conference on Multimedia (PCM)*, November 2006.

[59] Jan De Cock, Stijn Notebaert, Peter Lambert, Davy De Schrijver, and Rik Van de Walle. Requantization Transcoding in Pixel and Frequency Domain for Intra 16x16 in H.264/AVC. In *Proceedings of ACIVS (Advanced Concepts for Intelligent Vision Systems)*, 2006.

[60] Thomas Wedi and Hans Georg Musmann. Motion- and aliasing-compensated prediction for hybrid video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):577–586, July 2003.

[61] Oliver Werner. Drift analysis and drift reduction for multiresolution hybrid video coding. *Signal Processing: Image Communication*, 8(5):387–409, July 1996.

[62] Stijn Notebaert, Jan De Cock, and Rik Van de Walle. Improved H.264/AVC requantization transcoding using low-complexity interpolation filters for 1/4-pixel motion compensation. In *Proceedings of the IEEE Symposium on Computational Intelligence in Image and Signal Processing (CIISP)*, 2007.

[63] Jan De Cock, Stijn Notebaert, and Rik Van de Walle. Combined snr and temporal scalability for H.264/AVC using requantization transcoding and hierarchical B pictures. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, July 2007.

[64] Stijn Notebaert, Jan De Cock, Samie Beheydt, Jan De Lameillieure, and Rik Van de Walle. Mixed architectures for H.264/AVC digital video transrating. *Multimedia Tools and Applications*, 2009.

[65] T.K. Tan, Gary Sullivan, and Thomas Wedi. *Recommended Simulation Common Conditions for Coding Efficiency Experiments Revision 1*. Video Coding Experts Group, Doc. VCEG-AE10, Marrakech, Morocco, January 2007.

[66] Detlev Marpe, Thomas Wiegand, and Stephen Gordon. H.264/MPEG4-AVC Fidelity Range Extensions: Tools, profiles, performance, and application areas. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2005.

[67] Mathias Wien. Variable block-size transforms for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):604–613, July 2003.

[68] Niklas Björk and Charilaos Christopoulos. Transcoder architectures for video coding. *IEEE Transactions on Consumer Electronics*, 44(1):88–98, February 1998.

[69] Jeongnam Youn, Ming-Ting Sun, and Chia-Wen Lin. Motion vector refinement for high-performance transcoding. *IEEE Transactions on Multimedia*, 1(1):30–40, March 1999.

[70] Anthony Joch, Faouzi Kossentini, and Panos Nasiopoulos. A performance analysis of the ITU-T draft H.26L video coding standard. In *Proceedings of the 12th International Packet Video Workshop*, April 2002.

[71] Alexis M. Tourapis, Feng Wu, and Shipeng Li. Direct mode coding for bipredictive slices in the H.264 standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(1):119–126, January 2005.

[72] Thomas Wiegand, Heiko Schwarz, Anthony Joch, Faouzi Kossentini, and Gary J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, July 2003.

[73] Huifeng Shen, Xiaoyan Sun, and Feng Wu. Fast H.264/MPEG-4 AVC transcoding using power-spectrum-based rate-distortion optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(6):746–755, June 2008.

[74] Andrew Secker and David Taubman. Highly scalable video compression with scalable motion coding. *IEEE Transactions on Image Processing*, 13(8):1029–1041, August 2004.

[75] Alexandros Eleftheriadis and Pankaj Batra. Optimal data partitioning of MPEG-2 coded video. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(10):1195–1209, October 2004.

[76] Alexandros Eleftheriadis and Dimitris Anastassiou. Optimal data partitioning of MPEG-2 coded video. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, November 1994.

[77] Eric Barrau. MPEG video transcoding to a fine-granular scalable format. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2002.

[78] Huifeng Shen, Xiaoyan Sun, Feng Wu, Houqiang Li, and Shipeng Li. Transcoding to FGS streams from H.264/AVC hierarchical B-pictures. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2006.

[79] C. Andrew Segall. *SVC-to-AVC bit-stream rewriting for Coarse Grain Scalability*. Joint Video Team, Doc. JVT-T061, Klagenfurt, Austria, July 2006.

[80] C. Andrew Segall and Gary J. Sullivan. Spatial scalability within the H.264/AVC scalable video coding extension. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1121–1135, September 2007.

[81] Isabelle Amonou, Nathalie Cammas, Sylvain Kervadec, and Stephane Pateux. *Enhanced SNR scalability for layered CGS coding using quality layers*. Joint Video Team, Doc. JVT-S044, Geneva, Switzerland, April 2006.

[82] Isabelle Amonou, Nathalie Cammas, Sylvain Kervadec, and Stephane Pateux. *Signaling FGS NAL units*. Joint Video Team, Doc. JVT-S044, Klagenfurt, Austria, July 2006.

[83] Heiko Schwarz, Tobias Hinz, Detlev Marpe, and Thomas Wiegand. Constrained inter-layer prediction for single-loop decoding in spatial scalability. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2005.

[84] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. *Further results on constrained inter-layer prediction*. Joint Video Team, Doc. JVT-0074, Busan, Korea, April 2005.

[85] C. Andrew Segall and Jie Zhao. Bit-stream rewriting for SVC-to-AVC conversion. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2008.

[86] Martin Winken, Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Joint optimization of transform coefficients for hierarchical B picture coding in H.264/AVC. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2007.

[87] Heiko Schwarz and Thomas Wiegand. R-D optimized multi-layer encoder control for SVC. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2007.

[88] Kannan Ramchandran, Antonio Ortega, and Martin Vetterli. Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders. *IEEE Transactions on Image Processing*, 3(5):533–545, September 1994.

[89] Masaru Sugano, Yasuyuki Nakajima, Hiromasa Yanagihara, and Akio Yoneyama. An efficient transcoding from MPEG-2 to MPEG-1. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2001.

[90] Bo Shen, Ishwar Sethi, and Vasudev Bhaskaran. Adaptive motion vector resampling for compressed video down-scaling. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(6):929–936, September 1999.

[91] Haiyan Shu and Lap-Pui Chau. An Efficient Arbitrary Downsizing Algorithm for Video Transcoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):887–891, June 2004.

[92] Haiyan Shu and Lap-Pui Chau. The Realization of Arbitrary Downsizing Video Transcoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(4):540–546, April 2006.

[93] Vasant Patil and Rajeev Kumar. A fast arbitrary factor video re-sizing algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(9):1164–1171, September 2006.

[94] Robert Mokry and Dimitris Anastassiou. Minimal error drift in frequency scalability for motion-compensated DCT coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4):392–406, August 1994.

[95] Peng Yin, Anthony Vetro, Bede Liu, and Huifang Sun. Drift compensation for reduced spatial resolution transcoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(1):1009–1020, November 2002.

[96] Bo Shen. Submacroblock motion compensation for fast down-scale transcoding of compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10):1291–1302, October 2005.

[97] Huifeng Shen, Xiaoyan Sun, Feng Wu, Houqiang Li, and Shipeng Li. A fast downsizing video transcoder for H.264/AVC with rate-distortion optimal mode decision. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, pages 2017–2020, July 2006.

[98] Yap-Peng Tan and Haiwei Sun. Fast motion re-estimation for arbitrary downsizing video transcoding using H.264/AVC standard. *IEEE Transactions on Consumer Electronics*, 50(3):887–894, August 2004.

[99] Vasant Patil and Rajeev Kumar. A fast arbitrary factor H.264/AVC video resizing algorithm. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2007.

[100] Shinichi Sakaida, Kazuhisa Iguchi, Nao Nakajima, Yukihiro Nishida, Atsuro Ichigaya, Eisuke Nakasu, Masaaki Kurozumi, and Seiichi Gohshi. The Super Hi-Vision Codec. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, September 2007.

[101] Thomas Wedi. *Interpolation filters for motion compensated prediction with 1/4 and 1/8-pel accuracy*. Video Coding Experts Group (ITU-T Q.15/SG16), Doc. Q15-J-14, Osaka, Japan, May 2000.

[102] Anthony Vetro and Huifang Sun. On the motion compensation within a down-conversion decoder. *Journal of Electronic Imaging*, 7, 1998.

[103] Shijun Sun and Julien Reichel. *AHG Report on Spatial Scalability Resampling*. Joint Video Team, Doc. JVT-R006, Bangkok, Thailand, January 2006.

[104] Chen Chen, Ping-Hao Wu, and Homer Chen. Transform-domain intra prediction for H.264. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, May 2005.