Vrije Universiteit Brussel

FACULTY OF APPLIED SCIENCES
DEPARTMENT OF ELECTRONICS AND INFORMATICS

Wavelet-based Scalable Video Coding:
Algorithms and Complexity Models

Ioannis Andreopoulos

Brussels, 2005

Promoters:

Prof. Jan Cornelis

Dr. Adrian Munteanu

Thesis submitted to the Faculty of Applied Sciences of the Vrije Universiteit Brussel to obtain the degree of Doctor in the Applied Sciences

Examining Committee

Prof. Jan Cornelis – Vrije Universiteit Brussel – Promoter

Dr. Adrian Munteanu – Vrije Universiteit Brussel – Promoter

Prof. Hugo Thienpont – Vrije Universiteit Brussel –Chairman

Prof. Jean Vereecken – Vrije Universiteit Brussel – Vice-Chairman

Prof. Leo Van Biesen – Vrije Universiteit Brussel – Secretary

Dr. Peter Schelkens – Vrije Universiteit Brussel – Member

Prof. Thanos Stouraitis – University of Patras – Member

Prof. Mihaela van der Schaar – University of California Davis – Member

TABLE OF CONTENTS

ACKNOWLEDGMENTS

When I first decided to pursue doctorate studies in video compression, I was under the impression that:

(a) I would isolate myself in an underground (and preferably gothic) laboratory and emerge after four years;
(b) the result of my secluded studies would be a prototype design of a video compression chip that would compress video sequences better or faster than anything previously known to man, and an obscure book describing what I had done.

Now that the four years have passed, I have fortunately obtained the wisdom to say that all that was nonsense. Several people are to blame for this change in my views on research and the life of a researcher. First of all, Prof. Jan Cornelis who decided to give me the opportunity to study in the University of Brussels, and Prof. Thanos Stouraitis who facilitated this by first encouraging me to pursue challenging post-graduate studies in signal processing systems at the University of Patras, and then creating a link between the two universities.

Relating to point (b) above, the fact that this dissertation is less obscure and actually makes some sense to the scientific community is primarily the fault of two researchers who have made the personal choice to work with me on several topics. I am thinking of Dr. Adrian Munteanu and Prof. Mihaela van der Schaar. I thank them for many stimulating discussions and for their contributions during the preparation of several research papers. I can only hope that the result of my research will be a gratification for them after enduring me and correcting many of my errors during all this time. With respect to complexity-related topics, Dr. Peter Schelkens, Dr. Gauthier Lafruit and Dr. Konstantinos Masselos have done their best to share their views through stimulating discussions and successful collaboration. This dissertation is a mixture of algorithmic and complexity-related topics thanks to these five.
Last but not least, I would like to thank all the members of my jury committee for accepting to serve in this capacity.

Concerning point (a) from above, the fact that my vision of physical seclusion within the four walls of a laboratory did not materialize is due to the following people and their actions:

(I) Dr. Peter Schelkens and Prof. Jan Cornelis, who secured financial support for my work and also encouraged me to participate in MPEG meetings. The exploratory meetings on Interframe Wavelet Video Coding that I attended during 2002 and 2003 were essentially a kind of a fast-track graduate school for me, with many strange ideas and compression schemes floating around. In addition, the experience of attending meetings and conferences in foreign countries (especially in Asia) added a significant "social education" training element, which only now do I realize.
(II) Prof. Mihaela van der Schaar who invited me to spend the winter and spring of 2004 at the University of California Davis.
(III) Several friends who caused a most-welcome distraction from work while in Brussels or in Davis: I am especially thinking of Jenny and Antonis, as well as Alin, Alexandru, Fabio, Cristina, Paula, Joeri, Paulette, Liping, Aymeric, and others. My friends in Greece and my sister Efrosini have always been in my thoughts, whenever things got too boring at work.

Finally, I can safely say that my parents' efforts steered me towards an education that eventually led to this dissertation. I can only acknowledge this with great appreciation.

Yiannis Andreopoulos

March 2005

WAVELET-BASED SCALABLE VIDEO CODING: ALGORITHMS AND COMPLEXITY MODELS

Abstract

THE JPEG-2000 standard demonstrated that state-of-the-art coding performance can be obtained in still-image compression with a coding architecture that enables a rich set of features for the compressed bitstream, such as a precise rate-control mechanism and multiple qualities and resolutions of the same picture based on selective decoding of portions of the compressed bitstream. This is a natural consequence of the use of a scalable image compression algorithm based on the wavelet decomposition and embedded coding.

In this dissertation we investigate the extension of wavelet-based scalable coding to video signals. This problem appears to be of particular importance today, when ubiquitous video communication and video streaming takes place through unreliable (IP-based) wired and wireless media, and among terminals with different display and implementation capabilities. In the first part of our work, we examine algorithmic aspects of wavelet-based scalable video coding systems. In particular, novel algorithms are proposed for wavelet-domain (in-band) open-loop and closed-loop motion-compensated prediction. Our choice of deviating from the conventional temporal prediction in the spatial-domain representation (i.e. motion estimation and compensation using the original video signal) is motivated by the fact that, in this way, the multiresolution features of the discrete wavelet transform (DWT) can be exploited in order to provide an improved hierarchical representation of the video content across resolutions. One significant problem of our approach relates to the shift-variance of the DWT that hinders the performance of motion-compensated prediction in the wavelet domain. In fact, until recently, this has prevented the related research community from investigating in-band video coding. In this thesis, based on prior work, we attack the problem of shift-variance in a systematic way by proposing a new transform, termed the Complete-to-Overcomplete Discrete Wavelet Transform (CODWT), which effectively provides a shift-invariant (overcomplete) representation from the DWT, useful for in-band motion estimation. Several symmetry properties are proven for the CODWT and fast calculation algorithms are proposed that suit the application framework of in-band video coding.

Although motion estimation is utilizing the overcomplete DWT representation of the reference frame, motion compensation is always performed in the critically-sampled DWT. Hence the subsequently produced error-frames remain critically-sampled. This enables the possibility for existing state-of-the-art embedded wavelet coders to be employed for the coding of the error frames. Our extensive experimental evaluation reveals that the proposed systems provide comparable quality to the equivalent systems operating in the spatial domain, while at the same time a substantially-improved multiresolution decoding capability is achieved. In order to improve the prediction efficiency of the proposed video coding architectures, an advanced motion estimation algorithm is proposed that incorporates multihypothesis variable block-size prediction. Moreover, in the case of open-loop motion-compensated temporal filtering, we couple the proposed advanced prediction schemes in the wavelet-domain with in-band motion-compensated update schemes, thereby proposing a novel system that performs advanced in-band motion-compensated temporal filtering. This architecture is fully-scalable in quality, resolution and frame-rate and, at the same time, compares favourably with the state-of-the-art in video coding.

The second part of this dissertation is devoted to complexity-modeling aspects of wavelet-based scalable video coding. Two models for predicting the complexity of video coding algorithms under realistic conditions are proposed. The two proposals target different aspects of complexity; in particular, the first presents a model for the cache-behaviour of the two-dimensional DWT, which targets implementation platforms where the memory bottlenecks are expected to dominate the execution of data-dominated signal processing tools such as the DWT. The analysis in that section is based on analyzing the algorithmic flow and expressing the expected cache misses in analytical formulations. The second topic proposes a novel framework for complexity modeling of motion-compensated video decoders, which is based on a generic decomposition of the arithmetic (and potentially the memory) profile of these systems into a set of basis functions, without specific regard to the details of the algorithms involved. In each case, the models are experimentally validated by using real-life video coding tools and systems.

Chapter I

INTRODUCTION

1.1 The Need for Lossy Video Compression

THE digital revolution that we experience today has brought significant changes to the multimedia consumer market. Digital still-picture and video cameras permit the creation of image and video content directly in a digital format that allows for easy manipulation, transferring, copying and modification. This has given rise to significant research activity in the digital signal processing community, which evolves around many areas, such as content manipulation and protection, content-based search algorithms in multimedia data and, last but not least, image, video and audio compression. In fact, multimedia compression appears to be one of the most important stages in digital content handling and manipulation. This is due to the fact that, although the processing power of typical platforms has increased exponentially (as predicted by Moore's law), the capabilities of storage and transmission mediums (i.e. computer networks) appear to increase at a much slower pace. In this situation, the role of compression is to reduce the bandwidth of multimedia content so that efficient storage or transmission is permitted on a variety of devices and media. Thus, multimedia compression effectively bridges the gap between content production and content consumption.

With today's abundance of image and video content transferred over the Internet and other media, one may wonder whether practical compression algorithms can perform well in this role. In the particular area of video compression, the problem appears to be significantly harder than in image and audio coding; advances in image capturing technology and the processing capacity of today's digital platforms have allowed for video data generation and playback at high quality, resolution and frame-rate. For example, in the case of the Digital Cinema scenario [1], a typical setup could involve video frames consisting of ten million pixels, with 16-bit precision per pixel and replay rates of approximately fifty frames per second. For a two-hour movie, this would entail a volume of (roughly) $5.8 \cdot 10^{13}$ bits (or 58 Terra-bits) to be compressed. In terms of bandwidth, this corresponds to 8 Gbit/sec. Consequently, video compression algorithms have to reduce the storage requirement and transmission bandwidth by many orders of magnitude in order to match the capabilities of today's

typical storage and transmission mediums. Based on similar evidence, one can speculate that the increase of the volume of image and video data produced or consumed by the average user is increasing exponentially over time and will continue to do so in the near future [1].

Image and video compression seem to be able to cope with this effect due to the use of *lossy compression techniques*. The main idea is that, for each image or video frame, one can exploit known properties of human vision with the use of signal transforms, in order to produce a hierarchical or layered representation of the input content in space and time [1] [2] [3] [4]. In this representation, the significant visual information tends to be clustered in a small percentage of transform coefficients, while the remaining coefficients tend to constitute a *sparse* representation. Such transform representations can be efficiently quantized and coded with a variety of techniques; moreover, depending on the available bandwidth, a percentage of the transform-coefficient information is ignored. It is important to note that, in the case of video, the sparseness in the transform domain representation is significantly increased with the use of motion estimation and compensation techniques that exploit temporal similarities among neighbouring frames [2]. In many cases, depending on the performance of the utilized motion estimation model, as well as the transform and coding techniques, a visually near-lossless representation of the input video can be obtained after decoding. For example, modern state-of-the-art video coders can achieve compression ratios of more than 100:1 with little loss of visual quality. This comes as a result of more than thirty years of research.

Starting from the 1960's, digital image and video coding research utilized spatial DPCM coding. Transform coding techniques were investigated in the 1970's. A significant breakthrough was achieved in 1974, when Ahmed et al. [5] proposed the now-famous block-based Discrete Cosine Transform (DCT). In the same decade, researchers begun to experiment with motion-compensated prediction. This technology matured significantly over a number of years. Around 1985, the advances in processor speed permitted the first practical codecs utilizing block-based motion-compensated prediction with the DCT (MC-DCT) to operate in real time [2]. The success of these systems was tremendous, to the extent that all subsequent MPEG[1] and ITU-T (VCEG)[2] video coding standards were based on MC-DCT [6] [7]. Recent extensions to the basic MC-DCT technology achieved even more compression gains over pure frame-by-frame (intra) DCT coding for the same visual quality, at the expense of complexity. This is illustrated graphically in Figure I-1. As seen from the figure, there is an asymmetric evolution in codec complexity, where each new generation of encoders typically doubles the implementation complexity versus the previous state-of-the-art, while decoders appear to become more complex in a roughly linear fashion. As indicated in Figure I-1, more complex motion and scene modeling techniques tend to bring increased gains in compression, but this also increases the codec complexity, as well as the gap between encoder and decoder complexity. As a result, despite Moore's law holding up for a number of years, it appears that today's video compression systems are reaching a point, where further increase in video quality for a given bitrate will come at the cost of a significant increase in complexity that may prohibit real-time encoding. Consequently, instead of

[1] MPEG: Moving Picture Experts Group. It corresponds to working group 11 of subcommittee JTC1/SC29 of the International Organization for Standardization (ISO).

[2] ITU-T: International Telecommunication Union – Telecommunication Standarization Sector. VCEG: Video-Compression Experts Group.

focusing purely on compression, the research interest of the video coding community appears to be shifting towards efficient content distribution. This is discussed in detail in the following section.

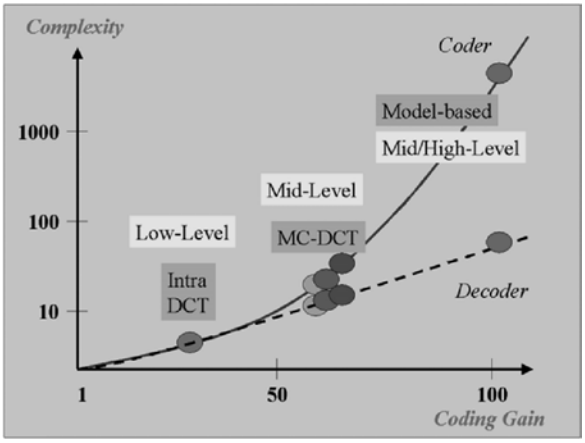


Figure I-1. Complexity of various video coding strategies versus coding gain [2].

1.2 Diversity in Digital Video Transmission – The Requirements for Scalable Video Coding

Apart from the continuous increases in the volume of video data to be handled, a significant challenge today involves the distribution of video content to a variety of consumers. Digital video communication during the 1980's and the first half of the 1990's was restricted to company executives using expensive, reliable channels with a high transmission bandwidth and Quality-of-Service (QoS) functionalities. However, the explosion of the World-Wide-Web (WWW) after 1995 changed the philosophy of the industrialized world into "connect anytime, anywhere". For example, today millions of people use video-streaming services at home, such as Apple's QuickTime™, Microsoft's MediaPlayer™ and Real-Network's RealPlayer™, in order to view movies, news, music videos, and other events via the WWW. Internet-based streaming has already manifested considerable commercial development of content distribution networks, such as the Akamai network [8]. These distribution networks use edge servers[3] on various locations across the globe that share multiple copies of the video content; users connected to the Internet access the video data via the closest edge server [8]. A newer phenomenon concerns video streaming via peer-to-peer networks [9] [10], which has the potential of diminishing broadcasting costs by creating distributed broadcasting networks, where a number of content consumers can also be used as broadcasters.

Finally, it is important to mention that significant breakthroughs in image-capturing technology increased the pervasiveness of image and video cameras into mobile computing devices, such as

[3] Multimedia-content servers at the network "edge", i.e. at the user-end of the Internet Backbone.

PDAs and cellphones. As a result, today even portable communication devices are beginning to act both as producers and consumers of digital video in computer networks.

In the vast majority of cases, video transmission occurs via unreliable networks, which use packet-based protocols that were originally designed for data communication (e.g. file-exchange). One typical example is the TCP/IP protocol used for the WWW, which does not support any QoS functionalities for robust and timely delivery of multimedia content. The integration of wireless services in the workplace and home environments adds further uncertainty concerning the distribution of video content to the end user, since wireless networks exhibit large bandwidth variations and packet losses, which depend on a number of uncontrollable factors. For the typical case of video streaming, today's situation is summarized in the illustration of Figure I-2., where various devices and networks are displayed along with their representative communication bandwidth.

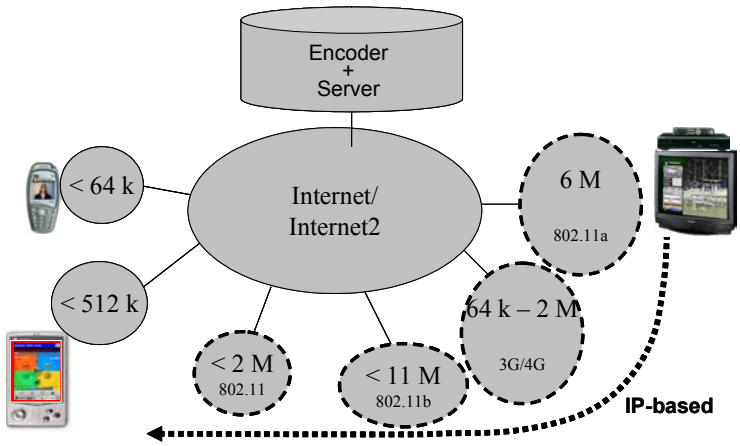


Figure I-2. A representative scenario for video streaming over the Internet, where several clients are connected via heterogeneous wireless network protocols.

In order to address the source-coding problems arising from the variability of transmission media for video communications, MPEG decided to perform an investigation of the application areas and the possible video-coding technologies that could potentially handle the communication requirements. This investigation started in 2002 [11] and is still on-going [12] [13]. Based on these efforts, there has been significant progress in identifying specific application areas, as well as source coding requirements [14].

Concerning the technological part, it was widely accepted that scalable coding appears to be the most viable technical solution to problems of video transmission over heterogeneous networks and user-terminals with diverse capabilities [4] [14]. Scalable video coding refers to a compression framework where content representations with different quality, resolution, and frame-rate can be extracted from parts of one compressed bitstream without the use of transcoding [4]. Previous research on the field [15] [16] [17] [18] [19] [20] [21] [22] demonstrated that a variety of coding techniques can be modified to produce a scalable bitstream with a varying degree of success.

A significant part of the success of a particular scalable coding algorithm relates to its compression efficiency. Consequently, one of the main requirements set for any scalable video coding algorithm is that it should perform as well as state-of-the-art non-scalable video coding in terms of visual quality [14], when both are compared for the vast majority of possible resolutions, quality levels and replay frame-rates.

A second major requirement concerns the end-to-end delay for compression and decompression. For applications involving video broadcasting and streaming, the end-to-end delay is not a very stringent requirement since, in the majority of today's cases, the content is compressed off-line. In general, it is considered that the overall streaming and decoding delay can be in the order of a few seconds [14]. Nevertheless, for other applications, such as conversational services and video monitoring systems, the delay requirements can be much more severe; in many cases, reliable operational systems should be bounded by an end-to-end delay in the order of 100 msecs [14].

In terms of the actual content scalability requirements, there is a series of issues identified by MPEG [14], which relate to the number of spatial, temporal and quality layers that a scalable bitstream should support, and last but not least, the associated complexity for encoding and decoding. These are elaborated further in the following subsections. It is important to note that the MPEG exploration activity has identified a number of secondary requests by industry concerning error-resiliency, object scalability, etc. [14]; since we do not address them in the proposals of this dissertation, they will not be discussed in more detail here.

1.2.1 Requirements for Spatial Scalability

Scalable video coding technology should support at least four spatial resolution levels [14], which could range (dyadically) from 1152×1408 to 144×176 pixels. This example includes the commonly used QCIF and CIF formats at the low-end of the scalability regime. The ability to provide multiple resolutions from one bitstream is important in a variety of applications. For example, multi-channel content production and distribution may involve the usage of different video resolutions [14], from studio-profile down to PDA resolution. In this way, different clients can be served by a single scalable bitstream. Another important application of resolution scalability is in the area of video surveillance and industrial monitoring systems, where the usage of this functionality is two-fold:

- In a typical scenario, multiple views from different locations are received at each monitoring station. Based on spatial scalability, each view can be enlarged on-the-fly, if necessary [14]. This limits the overall communication bandwidth between the camera network and the monitoring stations, since higher-resolution content is transmitted only following an alarm or a request from the user.

- Support for video storage with erosion functionality [14]. In surveillance and industrial monitoring systems, the importance of the recorded data, and thus the required resolution of the recorded video sequences, decreases over time. For example [14], the full resolution video needs only to be stored for three days, and sequences older than that are only required in a

medium resolution. If the sequences are older than one week, only a minimal resolution may be required for medium-to-long-term archiving.

Finally, a broad category of systems requiring resolution scalability consists of communication architectures utilizing the MPEG-21 Digital Item Adaptation (DIA) framework [23], where heterogeneous communication infrastructures are being utilizing for video streaming to a variety of end devices, as seen in Figure I-3. In this case, adaptation of the video resolution will be essential for satisfying the video decoder resolution and the channel bandwidth.

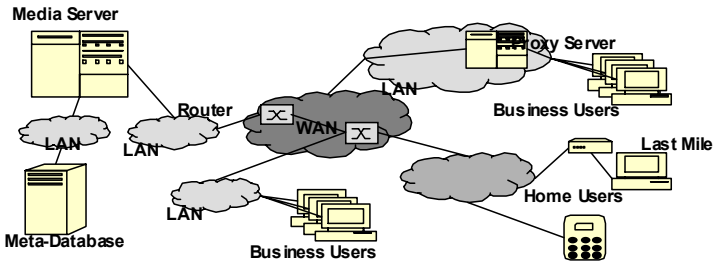


Figure I-3. A pictorial description involving adaptation to heterogeneous devices and network infrastructures with MPEG-21 DIA [14].

1.2.2 Requirements for Bitrate (Quality) Scalability

The representation of video into multiple qualities through scalable coding permits the video transmission to occur in a fine-granular set of bitrates. Hence, in practice, quality scalability is synonymous to bitrate scalability. According to MPEG requirements [14], practical quality-scalable coding schemes should provide a large variety of bitrates, that correspond to transmission rates provided by a variety of today's networks. A reasonable practical example of the bitrate ranges can be seen in the example of Figure I-2. It should be mentioned that for certain video archiving applications, such as the Digital Cinema industry, military surveillance and space exploration, or some medical applications, the possibility for lossless coding should be supported [14].

Obviously, the main application domains of quality scalability involve the broad area of video transmission over unreliable networks, where large bandwidth fluctuations can be efficiently handled by on-the-fly adapting the quality of the transmitted video. Another significant application is in content distribution, where in order to charge a different fee for higher quality content (requiring more bandwidth/storage), different quality levels should be provided by one bitstream [14].

1.2.3 Requirements for Frame-rate (Temporal) Scalability

Adapting the transmission and replay frame-rate consists of an efficient way for changing the video quality perceived by human observers. In addition, different frame-rates correspond to different complexity profiles for decoding the same video sequence. As a result, it is established that scalable video coding should support at least four levels of temporal scalability, which (in the majority of cases) can vary the decoding frame-rate dyadically [14]. Moreover, in order to satisfy a broad range of video applications, decoding of moving pictures with frame rates up to 60 Hz should be supported [14].

The main application domains demanding temporal scalability evolve around multi-channel content production and distribution, where the same bitstream will be viewed on a variety of devices supporting different temporal resolutions. For example, 7.5 Hz, 15 Hz, 30 Hz and 60 Hz should be supported in order to accommodate a broad range of clients, ranging from studio-level devices down to video-on-demand on a cellphone with limited processing capabilities. In addition, in this scenario, temporal scalability may be used to simulate fast forward/backward capabilities found in analog video playback devices, such as VCRs.

Finally, temporal scalability is very useful in the application areas that involve resolution scalability, such as surveillance and monitoring applications, where cameras are usually monitoring static areas and high frame-rate video is required only after an alarm is activated.

1.2.4 Requirements for Complexity Scalability

It is generally considered important that decoding complexity scales proportionally to the decoded temporal and spatial resolution [14]. Nevertheless, an equally important aspect of complexity scalability involves the establishment of a hierarchy of the video compression tools in terms of their average complexity profile. In this way, depending on the available resources at the decoding platform, on-the-fly adaptation of the compressed bitstream can occur so that real-time decoding is guaranteed, by selecting the sub-stream that leads to low-complexity decoding. To this end, since complexity is always relative to the algorithmic features as well as the implementation platform, in order to achieve reliable results, applicable complexity models should be used. In the context of multimedia algorithms, complexity modeling can be broadly defined as the procedure through which one obtains relative performance metrics for different algorithms (or different instantiations of one algorithm) with respect to: (a) the algorithm realization in terms of software or hardware; (b) the implementation platform.

Important applications of complexity scalability exist in the area of wireless video streaming to mobile computing devices such as cellphones and PDAs. In addition, depending on the target application, it may be required that encoding meets certain complexity bounds. This can be important in scenarios where the encoding devices are distributed over the network, as in the case of surveillance and monitoring applications for example.

1.3 Dissertation Outline

This dissertation is organized as follows. Chapter II presents a brief introduction to wavelet transforms, focusing on the multiresolution concept and the discrete wavelet transform. In addition, lifting-based discrete wavelet transforms are also presented. Since the proposed scalable video coding systems were based on wavelet decompositions along the spatial and temporal axes, this chapter serves as the basic background on transforms.

Chapter III focuses on fundamentals of scalable image and video coding. The basic tools of scalable image coding architectures are analyzed and an extension to video coding is presented. This section covers previous research and justifies our focus on one category of scalable video coding systems utilizing motion-compensated temporal filtering (MCTF) based on wavelet transforms.

Chapter IV presents our proposals for scalable video coding based on wavelet-domain motion estimation and compensation techniques. The proposed systems and architectures are built incrementally, based on predictive coding systems and MCTF-based coding architectures. In particular, this chapter introduces in-band motion compensated prediction within closed-loop and open-loop video coding systems. In addition, for open-loop systems, in-band motion-compensated temporal filtering is proposed by coupling in-band motion compensated prediction with in-band motion compensated update. The proposed systems are shown to have significant advantages from the functionality point of view. In addition, several experiments demonstrate that the proposed solutions present significant improvement in resolution scalability with little or no loss in coding efficiency.

Chapter V presents the theoretical foundation of complete-to-overcomplete discrete wavelet transforms used in the wavelet-domain video coding architectures of Chapter IV. Apart from the introduced mathematical formulation for the transform, significant symmetry properties are proven that lead to efficient architectures for the calculation of the single-rate and multi-rate parts of the transform.

Chapter VI discusses two complexity modeling topics for scalable video coding explored in the context of this dissertation. The first topic involves theoretical modeling of the relative performance of various proposals for the discrete wavelet transform realization in programmable processors. The focus is on the memory bottlenecks for the transform execution in widely-used cache-based processors. The second topic presents a complexity modeling framework for motion-compensated video decoders. Our approach is based on a generic decomposition of the arithmetic (and potentially the memory) profile of these systems into a set of basis functions, without specific regards to the details of the algorithms involved. For each case, experimental results with real coding systems realized in programmable processors substantiate the accuracy of the proposed models.

Finally, Chapter VII draws some general concluding remarks and outlines future research possibilities.

References

[1] A. Puri, J. Ribas-Corbera, W. Husak, and G. W. (eds), "Special issue on Technologies enabling Movies on Internet, HD DVD, and DCinema," *Signal Processing: Image Communication*, vol. 19, 2004.

[2] T. Sikora, "Trends and perspectives in image and video coding," *Proceedings of the IEEE*, vol. 93, pp. 6-18, 2005.

[3] G. Sullivan and T. Wiegand, "Video compression - from concepts to the H.264 standard," *Proceedings of the IEEE*, vol. 93, pp. 18-31, 2005.

[4] J.-R. Ohm, "Advances in scalable video coding," *Proceedings of the IEEE*, vol. 93, pp. 42-56, 2005.

[5] N. Ahmed, T. Natrajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, pp. 90-93, 1974.

[6] L. Chiariglione, "MPEG and Multimedia Communications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 5-18, 1997.

[7] L. Chiariglione, "Impact of MPEG Standards on Multimedia Industry," *Proceedings of the IEEE*, vol. 86, pp. 1222-1227, 1998.

[8] http://www.akamai.com.

[9] Y. Shan, I. V. Bajic, S. Kalyanaraman, and J. W. Woods, "Overlay multi-hop FEC scheme for video streaming over peer-to-peer networks," presented at IEEE International Conference on Image Processing, 2004.

[10] S. Khan, R. Schollmeier, and E. Steinbach, "A performance comparison of multiple description video streaming in peer-to-peer and content delivery networks," presented at IEEE International Conference on Multimedia and Expo, 2004.

[11] J.-R. Ohm and T. Ebrahimi, "Report of Ad-hoc Group on Exploration of Interframe Wavelet Technology in Video," ISO/IEC JTC1/SC29/WG11 m8295, March 2002.

[12] S. Pateux, "Exploration experiments on tools evaluation in wavelet video coding," Hong-Kong, CN, ISO/IEC JTC1/SC29/WG11 (MPEG), n6914, Jan. 2005.

[13] J.-R. Ohm and M. v. d. Schaar, "Call for proposals on scalable video coding technology," ISO/IEC JTC1/SC29/WG11 (MPEG), n5958, Oct. 2003.

[14] "Applications and requirements for scalable video coding," Palma de Mallorca, SP, ISO/IEC JTC1/SC29/WG11 (MPEG), n6830.

[15] H. M. Radha, M. v. d. Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding for multimedia streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, pp. 53-68, 2001.

[16] W. Li, "Streaming Video Profile in MPEG-4," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 301-317, 2001.

[17] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low Bit-Rate Scalable Video Coding with 3-D Set Partitioning in Hierarchical Trees (3-D SPIHT)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 1374-1387, 2000.

[18] V. Bottreau, B. Pesquet-Popescu, M. Bénetière, and B. Felts, "A Fully Scalable 3D Subband Video Codec," presented at IEEE International Conference on Image Processing, Thessaloniki, Greece, 2001.

[19] K. Shen and E. J. Delp, "Wavelet Based Rate Scalable Video Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 109-122, 1999.

[20] X. Yang and K. Ramchandran, "Scalable Wavelet Video Coding Using Aliasing-Reduced Hierarchical Motion Compensation," *IEEE Transactions on Image Processing*, vol. 9, 2000.

[21] A. Secker and D. Taubman, "Lifting-Based Invertible Motion Adaptive Transform (LIMAT) Framework for Highly Scalable Video Compression," *IEEE Transactions Image Processing*, vol. 12, pp. 1530-1542, 2003.

[22] X. Li, "Scalable Video Compression via Overcomplete Motion Compensated Wavelet Coding," *Signal Processing: Image Communication*, vol. 19, pp. 637-651, 2004.

[23] A. Vetro and C. Timmerer, "Digital item adaptation: Overview of standardization and research activities," *IEEE Transactions on Multimedia*, to appear.

Chapter II

WAVELET TRANSFORM

WHEN we deal with a given physical object, we encounter many of its different faces or representations. Whether for computer vision applications or for models in nuclear physics, every scientific field requires an appropriate representation of the measurements of the physical objects, which can be handled in a more natural and easy manner than the measurements themselves. In the area of signal processing, a way to obtain a specific representation is to decompose (transform) a signal x into elementary building blocks f_i of some importance, as: $x = \sum_i c_i f_i$, where f_i are simple waveforms (functions) and c_i are corresponding weighting factors (coefficients) of the decomposition. In addition, one may want that the waveforms have a specific physical meaning, i.e. they can correspond to the function of an (ideal) oscillator at a certain frequency of oscillation (Fourier basis) or the texture and edges of an image (families of multiresolution time-frequency bases). In the general case of continuous or discrete signals, what would be the desired properties of such a representation? After decades of research, it is generally acknowledged [1] [2] [3] [4] that the following properties are desired for practical decompositions of signals:

- A good rendition (reconstruction) of the original signal should be provided with as few as possible of the building blocks f_i (waveforms or functions).

- Time-frequency (or space-frequency) decompositions are preferred over pure frequency-domain based or pure time-domain analysis, since in most applications we are interested in information from both domains.

- A fast algorithm should exist in order to perform the decomposition, since otherwise a particular decomposition/representation might be only of theoretical importance.

- Multiresolution decompositions are preferred since various features of the signals of interest, such as edges and shapes in images, appear at various resolutions.

The order of importance of the above properties varies depending on the particular domain or application scenario.

2.1 Short-time Fourier Transform and Wavelet Transforms – Tilings in the Time-Frequency Plane

One of the classic tools to achieve different representations of a signal is the Fourier theory, for which a whole set of tools exists in literature: from purely continuous time, such as the Fourier integral, to the Discrete Fourier Transform (DFT) and the Fourier series expansion; moreover, the Fast Fourier Transform (FFT) provides a fast calculation for the transform decomposition. If we are given a pure frequency signal $e^{j\omega t}$, the Fourier-based methods will isolate a peak at the frequency w. However, for the simple case of a signal built by k pure oscillations of different frequencies occurring at adjacent intervals $\left[i_0 \quad i_1\right],\ldots,\left[i_{k-1} \quad i_k\right]$ in time, we obtain k peaks in the transform domain but the specific time localization is lost. This immediately points out the need for a time-frequency representation of a signal which could give us local information in time *and* in frequency.

In the Fourier case, the most obvious way to overcome this obstacle is to localize the sinusoids in the transform representation by windowing, that is, the basis functions now become $w(t-\tau)e^{j\omega t}$, where $w(t)$ is a "window" function with compact support allowing for time localization. This is traditionally called the windowed Fourier transform, or the short-time Fourier transform (STFT) and was originally introduced by Gabor [5]. By using the concept of the time-frequency plane, it is easy to show that the STFT separates this plane into adjacent square tiles. This is illustrated in the bottom part of Figure II-1(a) [6], where the shaded squares in the figure correspond to waveforms which are localized in the same time interval and in three adjacent frequency ranges, as shown in the top part of the figure.

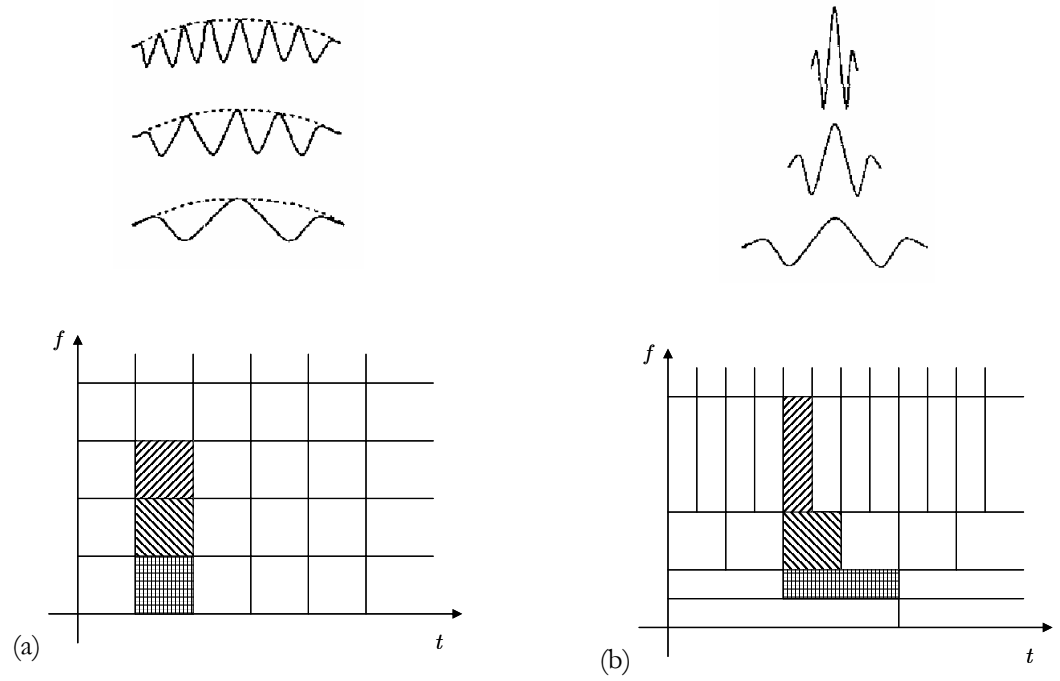


Figure II-1. Basis functions and corresponding tilings of the time-frequency plane [6]: (a) Short-time Fourier transform; (b) wavelet transform.

In the wavelet transform case, a different solution is offered: the precision of the frequency localization is logarithmic, i.e. proportional to the frequency range. Consequently, time localization becomes finer at the highest frequencies. This is illustrated in the bottom part of Figure II-1(b) and the corresponding wavelets are shown in the top part of the figure. It is important to notice though that one cannot obtain arbitrary localization in time and in frequency due to the uncertainly (Heisenberg) principle [4]. Nevertheless, wavelet theory based on multiresolution analysis and its generalizations offer a natural way to achieve an arbitrary tiling of the time-frequency plane that suits several applications in signal processing [6].

2.2 Continuous Wavelet Transform

A way to obtain a logarithmic time-frequency tiling such as the one illustrated in Figure II-1(b) is to define a family of functions [7] [8] [9]:

$$\psi_{\alpha,\tau}(t) = \frac{1}{\sqrt{\alpha}}\psi\left(\frac{t-\tau}{\alpha}\right), \qquad \alpha > 0, \tau \in \mathbb{R} \tag{2.1}$$

where ψ is a fixed function ("mother wavelet") that is well localized in time and frequency, τ defines the translation of this function in time and α is the scale parameter selected so that the functions $\psi_{\alpha,\tau}(t)$ have a constant (non infinite) norm, i.e.:

$$\left\|\psi_{\alpha,\tau}(t)\right\|^2 = \int_{-\infty}^{+\infty} \left|\psi_{\alpha,\tau}(t)\right|^2 dt = Q \tag{2.2}$$

By adapting the τ and α parameters under the constraint of (2.2), one can construct wavelets such as the ones shown in the top part of Figure II-1(b). Moreover, the localization of the mother wavelet can be ensured by bounding the mother wavelet function in time and frequency with [3]:

$$|\psi(t)| \le c(1+|t|)^{-1-\varepsilon}, \ |\Psi(\omega)| \le c(1+|\omega|)^{-1-\varepsilon} \tag{2.3}$$

for some $c, \varepsilon > 0$, where $\Psi(\omega)$ is the Fourier transform of ψ given by $\Psi(\omega) = \int_{-\infty}^{+\infty} \psi(t) e^{-j\omega t} dt$.

The Continuous Wavelet Transform (CWT) of an input signal $f(t)$ can now be defined as [7] [4]:

$$CWT_f(\alpha,\tau) = \langle f, \psi_{\alpha,\tau} \rangle = \frac{1}{\sqrt{\alpha}} \int_{-\infty}^{+\infty} f(t) \cdot \overline{\psi\left(\frac{t-\tau}{\alpha}\right)} dt \tag{2.4}$$

where $\overline{g}$ denotes the complex conjugate of function g and $\langle f, g \rangle$ denotes the inner product of signals or functions f, g. One notices that the CWT defined by (2.4) measures the similarity between the input signal $f(t)$ and a member of the family of functions of equation (2.1). The existence of an inverse transform for the CWT depends on the choice of ψ. In particular, if ψ satisfies the admissibility condition, expressed by [7]:

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty \tag{2.5}$$

then $f(t)$ can be reconstructed by the inverse continuous wavelet transform (ICWT) [7] [8] [9]:

$$f(t) = \frac{1}{C_\psi} \int_0^{+\infty} \frac{d\alpha}{\alpha^2} \int_{-\infty}^{+\infty} CWT_f(\alpha,\tau) \cdot \psi_{\alpha,\tau}(t)\, d\tau \tag{2.6}$$

A straightforward interpretation of this expression is that any signal $f(t)$ can be written as a superposition of shifted and dilated wavelets.

In today's applications, we are interested to work with discrete signals and consequently with discrete transforms, implementing orthonormal expansions. Daubechies [8] investigated the frames of the STFT and CWT, and proved that numerically stable reconstructions with milder constraints than in the STFT case are feasible for the CWT. The discretization of the scale and time-shift parameters is chosen as [8]:

$$\alpha = {\alpha_0}^m \text{ and } \tau = n\tau_0{\alpha_0}^m,\ \ m,n \in \mathbb{Z},\ \ \alpha_0 > 1, \tau_0 > 0 \tag{2.7}$$

The discretized family of wavelets is given by: $\psi_{m,n}(t) = {\alpha_0}^{-m/2}\psi({\alpha_0}^{-m}t - n\tau_0)$. It can be easily verified that the discretization presented in (2.7) satisfies the constraint of (2.2). Different values of m correspond to wavelets of different widths. Narrow, high-frequency wavelets are translated by smaller steps in order to "cover" the whole time axis, while wider, low-frequency wavelets are translated by larger steps. This contradicts with the sampling grid obtained for the STFT (which is uniform), corresponding to a constant time-frequency resolution analysis.

The simplicity and the elegance of the wavelet transform led to the discovery of wavelets that form orthonormal bases for square-integrable spaces by numerous researchers, e.g. see the work overviewed in [9] [8]. A formalization of such constructions by Mallat [1] created a framework for wavelet expansions called multiresolution analysis which is presented in more detail in the following.

2.3 Multiresolution Analysis

Based on the restrictions imposed on the sampling parameters α_0 and τ_0 by (2.7) as well as on the choice of the mother wavelet ψ, it is possible to control the redundancy in the reconstruction formula of equation (2.6), which is mathematically expressed by the double integration. In this way, based on a simplified version of (2.6) one can expand any function $f(t)$ in a wavelet basis. Indeed, for a given basis $\{g_n\}_{n\in\mathbb{N}}$, the existence of two constants $C \geq c > 0$, such that:

$$c\sum_n |a[n]|^2 \leq \int_{-\infty}^{+\infty} \left| \sum_n a[n] g_n(t) \right|^2 dt \leq C\sum_n |a[n]|^2 \tag{2.8}$$

independent of the choice of coefficients $a[n]$, has important consequences on the practical applications of an expansion of a function in $\{g_n\}_{n\in\mathbb{N}}$. In particular, (2.8) ensures the $L_2(\mathbb{R})$ stability of the basis $\{g_n\}_{n\in\mathbb{N}}$ [3]. It is important to notice that (2.8) is immediately satisfied in the case of an *orthonormal basis*, in which case we have $c = C = 1$. It was proven [8] that such bases can be obtained for the particular choice of $\alpha_0 = 2$ and $\tau_0 = 1$, i.e.:

$$\psi_{m,n}(t) = 2^{-m/2}\psi(2^{-m}t - n), \qquad m,n \in \mathbb{Z} \tag{2.9}$$

and an appropriate mother wavelet functions. The simplest example of such a function is the Haar wavelet, where ψ is a piecewise constant function given by:

$$\psi(t) = \begin{cases} 1, & t \in [0,0.5) \\ -1, & t \in [0.5,1) \\ 0, & \text{otherwise} \end{cases} \tag{2.10}$$

However, the family of wavelet basis functions obtained by the mother wavelet given in (2.10) suffers from a major disadvantage for many applications: due to the discontinuities of $\psi(t)$, the Haar family of wavelets cannot provide a good approximation for smooth functions. Hence, a major research objective during the early development of wavelet theory has been to provide systems that have the same multiscale structure of the Haar basis, but with the mother wavelet $\psi(t)$ being a more regular (smooth) function.

A strategy that achieves this goal and seems to be relevant for practical applications was proposed by Mallat [1]. In his seminal paper [1], he defined as *multiresolution analysis* a sequence of approximation subspaces (resolutions) $\{\mathbf{V}_m\}_{m\in\mathbb{Z}} \subset L_2(\mathbb{R})$ that satisfy the following requirements (note that we follow the convention of [9], where coarser subspaces (resolutions) are obtained by increasing m):

- The $\mathbf{V}_m$ are generated by a *scaling function* $\varphi \in L_2(\mathbb{R})$ in the sense that, for each m, the family of functions $\varphi_{m,n}(t) = 2^{-m/2}\varphi\left(\frac{t-n}{2^m}\right)$, $n \in \mathbb{Z}$, spans the space $\mathbf{V}_m$ and satisfies the stability condition given in (2.8) with $c = C = 1$. This means that the translations and dilations of φ form an orthonormal basis in $\mathbf{V}_m$.

- The subspaces are embedded, i.e. $\forall m \in \mathbb{Z}$, $\mathbf{V}_{m+1} \subset \mathbf{V}_m$.

- Orthogonal projectors $P_{\mathbf{V}_m}$ onto $\mathbf{V}_m$ satisfy $\lim_{m\to+\infty} P_{\mathbf{V}_m} f = 0$ and $\lim_{m\to-\infty} P_{\mathbf{V}_m} f = f$ for all $f \in L_2(\mathbb{R})$.

From these requirements it follows that if $f(t) \in \mathbf{V}_m$ then $f(\frac{t}{2}) \in \mathbf{V}_{m+1}$ and subspace $\mathbf{V}_m$ is invariant under translations of 2^m. The introduction of the scaling function φ is critical because, the continuous approximation of $f(t)$ at subspace (resolution) $\mathbf{V}_m$ can be computed as the orthogonal projection of the signal on this orthonormal basis:

$$\forall f(t) \in L^2(\mathbb{R}), \quad P_{\mathbf{V}_m} f(t) = \sum_{n=-\infty}^{+\infty} \langle f, \varphi_{m,n} \rangle \varphi_{m,n} \tag{2.11}$$

The inner products of the form:

$$a_m[n] = \langle f, \varphi_{m,n} \rangle \tag{2.12}$$

constitute the *discrete approximation* of $f(t)$ at the resolution that corresponds to subspace $\mathbf{V}_m$.

In addition, from the embeddedness requirement, φ can be established in an iterative manner using a fast algorithm, called the *pyramidal algorithm*. This was first proposed by Burt and Adelson [10] in the context of image coding. This algorithm has been modified in a form that performs the iterative calculation of the wavelet transform by Mallat [1]. In particular, if we define the discrete filter h whose impulse response is given by:

$$\forall n \in \mathbb{Z}, \quad h[n] = \left\langle \tfrac{1}{\sqrt{2}}\varphi(\tfrac{t}{2}), \varphi(t-n) \right\rangle \tag{2.13}$$

then the discrete approximation of a function $f(t)$ at scale m given by (2.12), can be computed iteratively by convolving $a_{m-1}[n]$ with $h[-n]$ followed by downsampling by two (i.e. keeping every other sample of the output):

$$a_m[n] = \sum_{k=-\infty}^{\infty} h[k-2n] \cdot a_{m-1}[k] = a_{m-1} * h[-2n] \tag{2.14}$$

Notice that (2.12) and (2.14) produce the same result, but the last equation does so iteratively by producing first the coefficients of the projection of the input function to all finer subspaces (resolutions[1]) $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_{m-2}, \mathbf{V}_{m-1}$.

Let $\mathbf{W}_m$ be the orthogonal complement of $\mathbf{V}_m$ in $\mathbf{V}_{m-1}$, satisfying $\mathbf{W}_m \oplus \mathbf{V}_m = \mathbf{V}_{m-1}$. The difference of information between the discrete approximation at subspace $\mathbf{V}_{m-1}$ and the approximation at the coarser resolution that corresponds to subspace $\mathbf{V}_m$ is determined by identifying an orthonormal basis in $\mathbf{W}_m$ and by projecting the signal f onto this basis. Mallat demonstrated that such orthonormal basis can be constructed by scaling and translating a function ψ, which is called orthogonal wavelet. The Fourier transform of the wavelet $\psi(t)$ is given by:

$$\Psi(\omega) = G(\omega/2)\Phi(\omega/2) \tag{2.15}$$

where $\Phi(\omega)$ is the Fourier transform of $\varphi(t)$, and:

$$G(\omega) = e^{-j\omega}\overline{H(\omega+\pi)}, \quad \text{with} \quad H(\omega) = \sum_{n=-\infty}^{\infty} h[n]e^{-jn\omega} \tag{2.16}$$

With these definitions, it can be shown that the family of functions $\psi_{m,n}(t) = 2^{-m/2}\psi\left(\frac{t-n}{2^m}\right)$ forms an orthonormal basis in $\mathbf{W}_m$. Projecting $f(t)$ onto this basis yields:

$$\forall f(t) \in L^2(\mathbb{R}), \quad P_{\mathbf{W}_m} f(t) = \sum_{n=-\infty}^{+\infty} \langle f, \psi_{m,n} \rangle \psi_{m,n} \tag{2.17}$$

The *discrete detail* of $f(t)$ at the resolution that corresponds to subspace $\mathbf{W}_m$ is then defined by the set of the inner products in relation (2.17):

$$d_m[n] = \langle f, \psi_{m,n} \rangle \tag{2.18}$$

[1] One can assume a discretization $a_0[n]$ of the input function $f(t)$ that equals the Nyquist sampling rate for $f(t)$. This can be the original input to the pyramid algorithm (discrete samples at subspace $\mathbf{V}_0$).

It can be shown that the discrete filter given by $g[n] = \left\langle \frac{1}{\sqrt{2}}\psi(\frac{t}{2}), \psi(t-n) \right\rangle$ has the transfer function $G(\omega)$ defined by (2.16). The discrete detail $d_m[n]$ can be computed via the pyramidal algorithm by convolving $a_{m-1}[n]$ with $g[-n]$ followed by a downsampling by 2:

$$d_m[n] = \sum_{k=-\infty}^{\infty} g[k-2n] \cdot a_{m-1}[k] = a_{m-1} * g[-2n] \tag{2.19}$$

The h and g filters are low-pass and band-pass filters respectively, with their impulse responses satisfying: $g[n] = (-1)^{1-n}\, h[1-n]$.

The pyramidal algorithm formalised by (2.14) and (2.19) decomposes the discrete approximation $a_{m-1}[n]$ of a function $f(t)$ into the approximation at a coarser resolution, $a_m[n]$, and the detail signal $d_m[n]$. Repeating in cascade this algorithm for $1 \le m \le M$ yields the Discrete (dyadic) Wavelet Transform (DWT) on M levels of the original discrete signal $a_0[n]$:

$$(a_M[n], d_M[n], d_{M-1}[n], \ldots, d_1[n]) \tag{2.20}$$

The block diagram of the wavelet analysis (or wavelet decomposition) process for one-dimensional signals is illustrated in left part of Figure II-2. In the inverse process (right part of Figure II-2), the discrete approximation $a_{m-1}[n]$ at the higher resolution is reconstructed from $a_m[n]$ and the detail signal $d_m[n]$ as:

$$a_{m-1}[n] = \sum_{k=-\infty}^{+\infty} h[n-2k] \cdot a_m[k] + \sum_{k=-\infty}^{+\infty} g[n-2k] \cdot d_m[k] \tag{2.21}$$

Thus, the Inverse Discrete Wavelet Transform (IDWT) determines $a_{m-1}[n]$ by upsampling the signals $a_m[n]$, $d_m[n]$ through insertion of zeros, followed by the application of the synthesis filters h, g. The original signal is iteratively reconstructed from its wavelet representation as shown in Figure II-2.b.

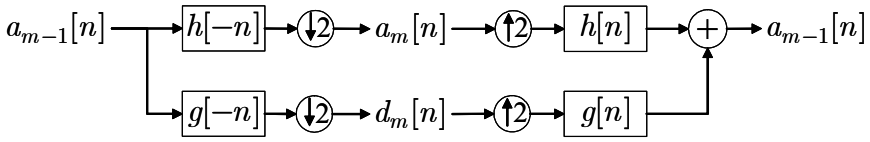


Figure II-2. Discrete wavelet transform: the discrete approximation $a_{m-1}[n]$ is decomposed into the approximation at the coarser resolution $a_m[n]$ and the detail signal $d_m[n]$. The inverse process reconstructs $a_{m-1}[n]$ from the coarse-resolution approximation and detail signals.

In the two-dimensional case, a multiresolution decomposition on M levels of a (discretized) image $f[i,j]$ results into a set of $3M+1$ subbands defining the DWT representation on M levels of f:

$$\left\{ aa_M[i,j],\ \ (ad_m[i,j], da_m[i,j], dd_m[i,j])_{1 \le m \le M} \right\} \tag{2.22}$$

The two-dimensional wavelet analysis consists of successively applying the one-dimensional DWT, using a row-column approach. We first convolve the rows of $aa_{m-1}[i,j]$ with a one-dimensional filter, retain every other sample, convolve the columns of the resulting signals with another one-dimensional filter and retain every other sample; for details, we refer to [9].

The formalism presented above shows that *orthonormal* wavelet expansions of discrete-time signals can be implemented using perfect reconstruction filter banks [1]. This is a very important theoretical finding that makes the link between the two different domains: signal expansions and filter-bank theory and design. As shown previously, in the case of orthonormal wavelet expansions, the analysis filters $h[-n], g[-n]$ are simply the time-reversed versions of the synthesis filters $h[n], g[n]$, and satisfy the perfect reconstruction condition, given by:

$$\begin{aligned} |H(\omega)|^2 + |H(\omega+\pi)|^2 &= 2 \\ |G(\omega)|^2 + |G(\omega+\pi)|^2 &= 2 \end{aligned} \tag{2.23}$$

The typical problem associated with orthonormal expansions is that there are no orthonormal linear-phase FIR filters satisfying the perfect reconstruction condition, apart of the trivial Haar basis. One can preserve linear phase (corresponding to symmetry for the wavelet) by relaxing the orthonormality constraint, and using biorthogonal bases [3] [6]. In case of *biorthogonal expansions*, we define the dual bases $\{\tilde{\varphi}_{m,n}\}$ and $\{\varphi_{m,n}\}$ (that correspond to subspaces $\widetilde{\mathbf{V}}_m$ and $\mathbf{V}_m$) as the two sets of functions that satisfy the biorthogonality constraint with respect to integer shifts k, l, given by:

$$\langle \tilde{\varphi}_{m,n-l}, \varphi_{m,n-k} \rangle = \delta[k-l]. \tag{2.24}$$

Then, for a signal $f(t) \in L^2(\mathbb{R})$, we can obtain the decomposition of $f(t)$ onto subspace $\widetilde{\mathbf{V}}_m$ using basis functions $\{\tilde{\varphi}_{m,n}\}$ and then synthesize $f(t)$ with the set of basis functions $\{\varphi_{m,n}\}$:

$$\begin{aligned} \sum_{n=-\infty}^{+\infty} \langle f, \varphi_{m,n} \rangle \tilde{\varphi}_{m,n} &= \sum_{n=-\infty}^{+\infty} \tilde{a}[n] \cdot \tilde{\varphi}_{m,n} \\ &= \sum_{n=-\infty}^{+\infty} \langle f, \tilde{\varphi}_{m,n} \rangle \varphi_{m,n} \\ &= \sum_{n=-\infty}^{+\infty} a[n] \cdot \varphi_{m,n} \end{aligned} \tag{2.25}$$

where:

$$\tilde{a}[n] = \langle f, \varphi_{m,n} \rangle, \;\; a[n] = \langle f, \tilde{\varphi}_{m,n} \rangle \tag{2.26}$$

are the transform coefficients obtained by projecting $f(t)$ onto $\{\varphi_{m,n}\}$ and $\{\tilde{\varphi}_{m,n}\}$, respectively.

In a similar way, one can define the set of biorthogonal wavelet functions $\{\tilde{\psi}_{m,n}\}$, $\{\psi_{m,n}\}$ and their corresponding subspaces $\widetilde{\mathbf{W}}_m$, $\mathbf{W}_m$. In this case, subspaces $\widetilde{\mathbf{V}}_m$ and $\widetilde{\mathbf{W}}_m$, as well as $\mathbf{V}_m$ and $\mathbf{W}_m$, are not orthogonal complements to $\widetilde{\mathbf{V}}_{m-1}$ and $\mathbf{V}_{m-1}$, respectively. Instead, this design preserves orthogonality in the analysis and synthesis duets of scaling functions and wavelets (thereby permitting

perfect reconstruction) and allows for increased freedom in the design of the basis functions. The perfect reconstruction condition satisfied by filter banks implementing biorthogonal expansions has been discovered by Vetterli [6], and is given by:

$$\begin{cases} \tilde{H}^*(\omega)H(\omega) + \tilde{G}^*(\omega)G(\omega) = 2 \\ \tilde{H}^*(\omega+\pi)H(\omega) + \tilde{G}^*(\omega+\pi)G(\omega) = 0 \end{cases} \tag{2.27}$$

where $\tilde{H}(\omega), \tilde{G}(\omega)$ and $H(\omega), G(\omega)$ are the Fourier transforms of the analysis and synthesis filter pairs respectively. The construction of orthonormal and biorthogonal wavelet bases has been heavily explored over the last decade, leading to a variety of wavelet classes useful in signal compression applications.

2.4 The Lifting Scheme

As seen the in previous section, through the multiresolution concept, the discrete wavelet transform was initially proposed as a (critically-sampled) time-frequency representation. The frequency-domain properties of the wavelet filters play a crucial role in the design of the DWT. For example, the number of vanishing moments of the analysis and synthesis wavelets (which expresses the degree of smoothness of the wavelet mother function) is equal to the number of zeros of $H(\omega)$ at $\omega = \pi$. Hence most of the early research on wavelet filter-bank designs was based on directly optimizing different tradeoffs in the Fourier domain. Sweldens [11] was the first to show an alternative discrete wavelet transform construction based on a series of signal prediction and update steps. Due to the fact that the coefficients of the transform are gradually build by a series of modifications (liftings), the algorithm was called lifting scheme.

2.4.1 Polyphase Representation of the Discrete Wavelet Transform

In order to study the idea behind the lifting scheme in more detail, we can define the perfect reconstruction conditions of the previous subsection in the Z domain. The two analysis filters are denoted as $H(z)$ (low-pass) and $G(z)$ (high-pass), while the corresponding synthesis low-pass and high-pass filters are denoted as $\widetilde{H}(z)$ and $\widetilde{G}(z)$. The perfect reconstruction conditions for this scheme are given in the Z -domain by:

$$\begin{aligned} &\widetilde{H}(z)H(z^{-1}) + \widetilde{G}(z)G(z^{-1}) = 2 \\ &\widetilde{H}(z)H(-z^{-1}) + \widetilde{G}(z)G(-z^{-1}) = 0 \end{aligned} \tag{2.28}$$

where for any FIR filter $F(z)$, $F(z) = \sum_k f_k z^{-k}$ is a Laurent polynomial that denotes the Z domain representation of this filter.

Early research on filter-bank design revealed that, instead of the conventional filtering-and-downsampling structure for the transform implementation (Figure II-2), an equivalent (but faster) way to realize the transform is with the use of the polyphase matrix of the analysis and synthesis filter-

bank [12]. This is illustrated in Figure II-3, where the Z -domain representation of the approximation (low-frequency) signal is denoted by $A_0^1(z)$, while the detail (high-frequency) signal is denoted by $D_0^1(z)$. As shown in the analysis stage of the figure, filtering with $H(z)$, $G(z)$ is replaced by filtering with the analysis polyphase matrix $E_0(z)$. It is important to notice that downsampling is performed prior to the actual filtering, thereby avoiding all the unnecessary computations that would be discarded by the downsampling in the original design of Figure II-2. The synthesis stage of Figure II-3 (with synthesis polyphase matrix $R_0(z)$) appears completely symmetrical to the analysis part and the signal reconstruction is performed by upsampling and addition of the corresponding results.

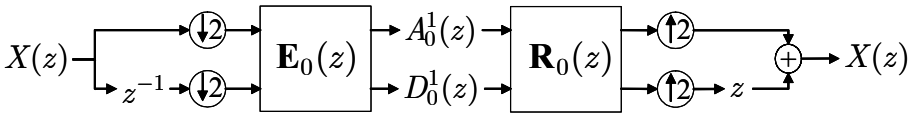


Figure II-3. Analysis and synthesis of a signal $X(z)$ with the use of the polyphase matrix.

The analysis polyphase matrix for the case of Figure II-3 is defined as:

$$\mathbf{E}_0(z) = \begin{bmatrix} H_0(z) & H_1(z) \\ G_0(z) & G_1(z) \end{bmatrix} \tag{2.29}$$

where $F_{\{0,1\}}(z)$ denote the Type-I polyphase components of filter $F(z)$, given by [12]:

$$\begin{aligned} F_0(z) &= \tfrac{1}{2}\left(F(z^{\frac{1}{2}}) + F(-z^{\frac{1}{2}})\right) \\ F_1(z) &= \tfrac{1}{2} z^{\frac{1}{2}} \left(F(z^{\frac{1}{2}}) - F(-z^{\frac{1}{2}})\right) \end{aligned}. \tag{2.30}$$

Notice that (2.30) separates the even and odd taps of $F(z)$ into filters $F_0(z)$ and $F_1(z)$ respectively. This corresponds to a separation of the even and odd sampling grid, followed by downsampling. This process can be written for a signal $X(z)$ based on the Type-I polyphase decomposition as [12]:

$$\begin{aligned} X_0(z) &= \tfrac{1}{2}\left(X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})\right) \\ X_1(z) &= \tfrac{1}{2} z^{-\frac{1}{2}} \left(X(z^{\frac{1}{2}}) - X(-z^{\frac{1}{2}})\right) \end{aligned}. \tag{2.31}$$

The difference between $F_1(z)$ and $X_1(z)$ is motivated by the convolution operation. Notice that (2.31) performs the signal separation and downsampling illustrated in the left part of Figure II-3.

The wavelet decomposition can be written as:

$$\begin{bmatrix} A_0^1(z) \\ D_0^1(z) \end{bmatrix} = \mathbf{E}_0(z) \begin{bmatrix} X_0(z) \\ X_1(z) \end{bmatrix}. \tag{2.32}$$

Moreover, based on Cramer's rule, the corresponding synthesis polyphase matrix is given by:

$$\mathbf{R}_0(z) = \frac{1}{\det \mathbf{E}_0(z)} \begin{bmatrix} G_1(z) & -H_1(z) \\ -G_0(z) & H_0(z) \end{bmatrix} \tag{2.33}$$

and the wavelet reconstruction is performed by:

$$\begin{bmatrix} X_0(z) \\ X_1(z) \end{bmatrix} = \mathbf{R}_0(z) \begin{bmatrix} A_0^1(z) \\ D_0^1(z) \end{bmatrix} \tag{2.34}$$

followed by the merging of the polyphase components:

$$X(z) = X_0(z^2) + zX_1(z^2) \tag{2.35}$$

Finally, under this formulation, an elegant expression the perfect reconstruction condition is [13]:

$$\det \mathbf{E}_0(z) = a \cdot z^{-k}, \quad a \in \mathbb{R}, k \in \mathbb{Z} \tag{2.36}$$

2.4.2 Factorizations of $\mathbf{E}_0(z)$ – Lifting Scheme in the Z domain

Daubechies and Sweldens [14] demonstrated that any biorthogonal analysis or synthesis filter-bank can be factorized in a succession of liftings (updates) and dual-liftings (predictions) applied to the polyphase decomposition (lazy wavelet [14]) of (2.31), up to shifting and multiplicative constants. This theorem can be demonstrated following a constructive proof based on the Euclidean algorithm for polynomial division [14].

In principle, since the factorization algorithm is based on polynomial division, there are many possible solutions that allow the realization of a given filter-bank with the lifting scheme. In general, one can mathematically express these factorizations as:

$$\mathbf{E}(z) = \begin{pmatrix} K & 0 \\ 0 & 1/K \end{pmatrix} \prod_{i=1}^{M} \begin{pmatrix} 1 & U^i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -P^i(z) & 1 \end{pmatrix} \tag{2.37}$$

where filters $P^i(z)$ and $U^i(z)$ indicate prediction (dual-lifting) and update (lifting) operators, respectively, and K, $1/K$ are scaling factors that ensure that the transform is (approximately) orthonormal. It is important to notice that, although for each $1 \leq i \leq M$ of (2.37) the prediction step appears to precede the update step, one can equivalently perform lifting factorizations with the alternative series of steps [14]. The schematic representation corresponding to equation (2.37) is illustrated in Figure II-4.

As explained in [14], the lifting scheme combines many advantages, such as low implementation complexity by reducing the required number of multiplications and additions, integer-to-integer transforms as proposed independently by Calderbank et al [15] and Dewitte and Cornelis [16], and complete reversibility even if non-linear or adaptive filters are used in the prediction and update steps. In each case, the transform can be inverted by following the schematic of Figure II-4 from right to left and inverting the scaling factors, the direction of the data flow and the signs at the summation

points. Finally, the reconstructed signal is formed by upsampling followed by merging of the polyphase components, which is mathematically expressed in equation (2.35).

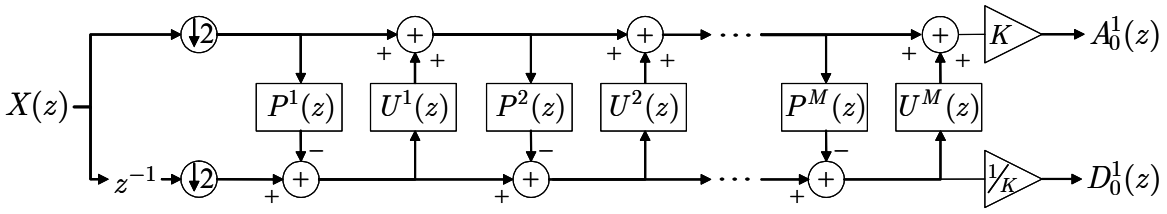


Figure II-4. Schematic representation of the wavelet decomposition based on the lifting-scheme.

The adaptivity permitted by the wavelet decomposition based on the lifting scheme has led to many interesting applications. For example, based on the lifting scheme, adaptive wavelet decompositions with or without bookkeeping [17] [18] and adaptive bandelet decompositions [19] have been proposed for image coding. In addition, video coding with motion compensated temporal filtering (MCTF) based on the lifting scheme has recently shown very promising results [20, 21] [22]. In this dissertation, the lifting scheme has been used both for the implementation of the spatial decomposition during video compression, as well as for the implementation of MCTF-based scalable video coding.

References

[1] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, 1989.

[2] D. L. Donoho, "Wedgelets: Nearly Minimax Estimation of Edges," Stanford University, USA, Technical Report 1997 1997.

[3] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure Applied Mathematics*, vol. 45, pp. 485-560, 1992.

[4] I. Daubechies, "The Wavelet Transform, Time-Frequency Localisation and Signal Analysis," *IEEE Transactions on Information Theory*, vol. 36, pp. 961-1005, 1990.

[5] D. Gabor, "Theory of Communication," *Journal of the Institute of Electrical Engineering*, vol. 93, pp. 429-457, 1946.

[6] M. Vetterli and C. Herley, "Wavelets and Filter Banks: Theory and Design," *IEEE Transactions on Signal Processing*, vol. 40, pp. 2207-2232, 1992.

[7] A. Grossmann and J. Morlet, "Decomposition of Hardy functions into square integrable wavelets of constant shape," *SIAM J. Math. Anal.*, vol. 15, pp. 723-736, 1984.

[8] I. Daubechies, *Ten lectures on wavelets*. Philadelphia, Pa.: Society for Industrial and Applied Mathematics, 1992.

[9] S. G. Mallat, *A wavelet tour of signal processing*. San Diego: Academic Press, 1998.

[10] P. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, vol. 31, pp. 482-540, 1983.

[11] W. Sweldens, "The Lifting Scheme: a Custom Design Construction of Biorthogonal Wavelets," *Journal of Appl. and Comput. Harmonic Analysis*, vol. 3, pp. 186-200, 1996.

[12] G. Strang and T. Nguyen, *Wavelets and filter banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[13] J. Kovacevic and M. Vetterli, "Nonseparable multidimensional perfect reconstruction filter banks and wavelet bases for R^n," *IEEE Transactions on Information Theory*, vol. 38, pp. 533-555, 1992.

[14] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, pp. 247-269, 1998.

[15] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet Transforms that Map Integers to Integers," *Journal of Applied Computational Harmonics Analysis*, vol. 5, pp. 332-369, 1998.

[16] S. Dewitte and J. Cornelis, "Lossless Integer Wavelet Transform," *IEEE Signal Processing Letters*, vol. 4, pp. 158-160, 1997.

[17] R. L. Claypoole, G. M. Davis, W. Sweldens, and R. G. Baraniuk, "Nonlinear wavelet transforms for image coding via lifting," *IEEE Transactions on Image Processing*, vol. 12, pp. 14449-1460, 2003.

[18] G. Piella and H. J. A. M. Heijmans, "Adaptive lifting schemes with perfect reconstruction," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1620-1631, 2002.

[19] E. LePennec and S. Mallat, "Sparse geometric image representations with bandelets," *IEEE Transactions on Image Processing*, to appear.

[20] B. Pesquet-Popescu and V. Bottreau, "Three Dimensional Lifting Schemes for Motion Compensated Video Compression," presented at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, Utah, USA, 2001.

[21] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with Lifting Implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 1183-1194, 2004.

[22] A. Secker and D. Taubman, "Lifting-Based Invertible Motion Adaptive Transform (LIMAT) Framework for Highly Scalable Video Compression," *IEEE Transactions Image Processing*, vol. 12, pp. 1530-1542, 2003.

Chapter III

SCALABLE IMAGE AND VIDEO CODING FUNDAMENTALS

THE JPEG-2000 standardization effort [1] demonstrated that state-of-the-art coding performance can be obtained in still-image compression with a coding architecture that enables a rich set of features for the compressed bitstream. In particular, unlike the previous JPEG standard, JPEG-2000 provides a precise rate-control mechanism based on embedded coding of wavelet coefficients. Moreover, multiple qualities and multiple resolutions of the same picture are possible within JPEG-2000, based on selected portions of the compressed bitstream [2]. This is a natural consequence from the use of a scalable image compression algorithm based on the wavelet decomposition and embedded coding.

In the area of video compression, similar functionalities have long been pursued, mainly via the use of extensions of the basic MPEG coding structure [3]. In terms of scalable video coding standards, this resulted in the fine-granularity scalable video coding extension of MPEG-4 video (MPEG-4 FGS) [4]. However, MPEG-4 FGS left much to be desired. In particular, the compression efficiency of FGS was not as good as the equivalent non-scalable (baseline) coder. In addition, the use of the conventional closed-loop video coding structure of MPEG-alike coders hindered the scalability functionalities.

As a result, recent research efforts on scalable video coding were targeted on extension of open-loop coding systems, such as JPEG-2000, to video coding. Although an extension of the basic technology of JPEG-2000 to three dimensions is feasible by extending its transform and coding modules to three dimensions [5] [6], this does not guarantee an efficient video coding system since motion-compensation tools are not included. On the other hand, it is important to notice that the design of motion-compensated three-dimensional wavelet decompositions has been a long withstanding problem [7] [8-10]. However, recent results indicate that there can be a simple solution [11] [12] based on lifting factorizations. Consequently, it was decided within MPEG that the exploration process of

this activity should be largely supported by the video standardization community. As a result, an ad-hoc group was setup within the MPEG video group to monitor and document interframe wavelet technology [13]. In turn, this resulted in a flurry of research activity on the topic, which lead to a call for proposals for the development of a new scalable video coding standard within MPEG [14].

In this chapter, we present an overview of the fundamental tools behind scalable image and video coding. The first section (3.1) is dedicated to the description of scalable image coding. Sections 3.2 – 3.4 present the basic tools that can be used for the extension of scalable image coding architectures to motion-compensated video coding via the use of an open-loop temporal decomposition structure. The overview of this structure is the topic of Section 3.5. In addition, the overview of the basic video coding systems is completed with the survey of the conventional (closed-loop) temporal prediction structure of the current MPEG systems. In each case, in order to facilitate the link of the presented methods and architectures with real-world compression schemes, indicative state-of-the-art systems are reviewed.

3.1 Wavelet-based Scalable Image Coding

The basic framework for a wavelet-based scalable image coding system is illustrated in Figure III-1. The encoder (top row) consists of the transform module W, which can be any critically-sampled discrete wavelet transform, the coder modules (Q, C, EC) that produce an entropy-coded stream of symbols (compressed bitstream), and the Opt module that instantiates a set of (optimized) truncation points within the compressed bitstream.

Prior to the decoding process (bottom row of Figure III-1), a bitstream extractor B performs the actual truncation of the compressed bitstream (based on the truncation points provided by the Opt module) in order to create a sub-bitstream that matches the channel bandwidth and the constraints set by the user in terms of quality or resolution of the decoded output image. This process only involves data handling operations and does not require any transcoding operations. In fact, B and Opt can interact similar to a client-server application: all potential quality and resolution levels of the decoded image are mapped to bitstream truncation points by the Opt module by utilizing *hint information* [15] [16] produced by the encoder. The user inputs the desired bitrate and resolution level at the B module, which in turn can request the appropriate bitstream truncation points and perform the actual truncation of the compressed bitstream. Within the conventional image compression framework, the hint information utilized in the Opt module is encapsulated in rate-distortion points [17]. However, extensions of this framework that involve complexity metrics can also be envisaged [15]. There are cases where Opt and B can be merged in one process at the encoder side, e.g. Tier-II in JPEG-2000 compression [17]. However, other practical systems have also performed the process in two distinct modules, e.g. scalable video codecs [18] [19].

Once the appropriate bitstream is available, the process of reconstruction of the decoded image occurs in a symmetric manner to the encoder: first the decoder modules decompress the extracted bitstream and perform inverse quantization (ED, D and Q^{-1} modules); then the inverse DWT (W^{-1} module) reconstructs the output in the spatial (pixel) domain. Obviously, the process can be pipelined for efficient implementation [20].

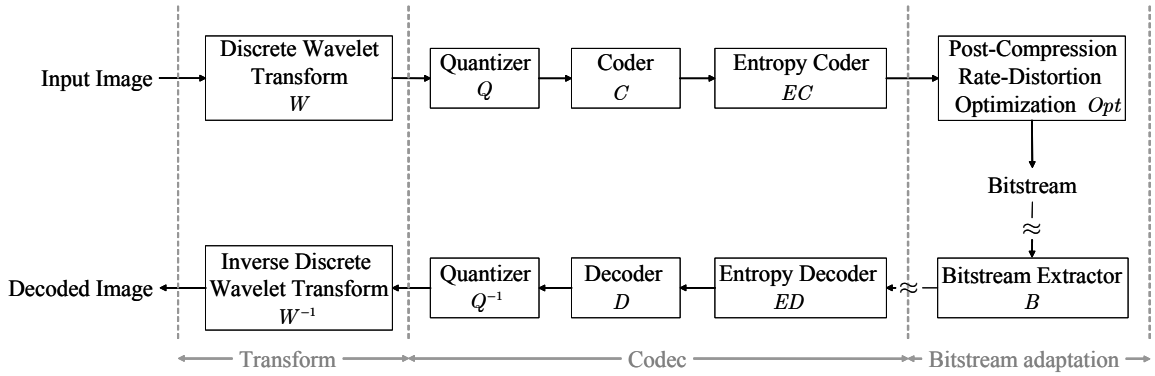


Figure III-1. Scalable image compression architecture based on the discrete wavelet transform. The symbol $\approx$ indicates the (potential) existence of a transmission channel.

In typical image coding applications, the transform W consists of the critically-sampled dyadic discrete wavelet transform. The dyadic decomposition (or sometimes called the Mallat decomposition [21]) separates low and high-frequency components in the row and column direction, and iterates the process only in the low-frequency subband. In this way, as explained in Chapter II, a multiresolution decomposition of the input image is obtained. It is important to point out that experimental results in wavelet-based coding [17] reveal that scalable coding utilizing this decomposition obtains, on average, the best compression performance for natural images.

3.1.1 Embedded Quantization in Wavelet-based Image Coding

The purpose of the quantizer Q in the system of Figure III-1 is to map (usually in a lossy manner) the wavelet coefficients into a set of indices (quantization cells). The coefficients are then represented by the parameters of this mapping operation [22]. The inverse quantization Q^{-1} uses these parameters and, based on the inverse mapping operation, reconstructs the wavelet coefficients at the decoder side. Note that, although the inverse mapping of Q^{-1} is the mathematical inverse of the (forward) mapping of Q, the reconstructed wavelet coefficients are (usually) not identical to the wavelet coefficients of the encoder because:

(a) the quantization mapping operation usually incurs loss;

(b) the reconstructed parameters of the mapping at the decoder side may not exactly coincide with the mapping parameters at the encoder side; this may happen due to transmission errors or incomplete receipt of the compressed bitstream.

Although many quantization schemes have been developed in the past, e.g. [22] [23] [17], for the quantization of the wavelet coefficients at a wide range of bitrates, the uniform quantizer was found to be nearly-optimal in the mean-square error sense. Moreover, it is common to use a double cell size around zero (deadzone), since this improves performance at low bitrates [17]. Nevertheless, the uniform quantizer cannot be used for embedded coding, since in embedded coding (by definition) the

partition cells of the higher-rate quantizers must be embedded in the partition cells of all the lower-rate quantizers [24].

An important category of embedded scalar quantizers is the family of embedded deadzone scalar quantizers. For this family, each input wavelet coefficient X is quantized to:

$$i_p = Q_p(X) = \begin{cases} sign(X) \cdot \left\lfloor \frac{|X|}{2^p \Delta} + \frac{\xi}{2^p} \right\rfloor & \text{if } \frac{|X|}{2^p \Delta} + \frac{\xi}{2^p} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where: $\lfloor a \rfloor$ denotes the integer part of a; $\xi < 1$ determines the width of the deadzone; $\Delta > 0$ is the basic quantization step size (basic partition cell size) of the quantizer family; $p \in \mathbb{Z}_+$ indicates the quantizer level (granularity), with higher values of p indicating coarser quantizers. In general, p is upper bounded by a value N, selected to cover the dynamic range of the input signal. The reconstructed value is given by the inverse operation:

$$y_i^p = Q_p^{-1}(i_p) = \begin{cases} 0 & i_p = 0 \\ sign(i_p) \cdot \left(|i_p| - \frac{\xi}{2^p} + \delta \right) 2^p \Delta & i_p \neq 0 \end{cases} \tag{3.2}$$

where $0 \leq \delta < 1$ specifies the placement of the reconstructed value y_i^p within the corresponding uncertainty interval (partition cell), defined as $C_{i_p}^p$, and i is the partition cell index, which is bounded by a predefined value for each quantizer level (i.e. $0 \leq i \leq M_p - 1$, for each p). Based on equation (3.1), it is rather straightforward to show that the quantizer Q_0 has embedded within it all the uniform deadzone quantizers with step sizes $2^p \Delta, p \in \mathbb{Z}_+$. Moreover, it can be shown that, under the appropriate settings, the quantizer index obtained by dropping the p least-significant bits (LSBs) of i_0 is the same as that which would be obtained if the quantization was performed using a step size of $2^p \Delta, p \in \mathbb{Z}_+$ rather than Δ. This means that if the p LSBs of i_0 are not available, one can still dequantize at a lower level of quality using the inverse quantization formula given in (3.2).

The most common option for embedded scalar quantization is the Successive Approximation Quantization (SAQ) [24]. SAQ is a particular instance of the generalized family of embedded deadzone scalar quantizers defined above. For SAQ, $M_N = M_{N-1} = \ldots = M_0 = 2$ and $\xi = 0$, which determines a deadzone width twice as wide as the other partition cells, and $\delta = 1/2$, which implies that the output levels y_i^p are in the middle of the corresponding uncertainty intervals $C_{i_p}^p$.

An example of the partition cells and quantizer indices obtained for SAQ for $N = 2$ is given in Figure III-2, in which binary notation is used with s denoting the sign bit. SAQ can be implemented via thresholding, by applying a monotonically decreasing set of thresholds of the form [24]:

$$\mathcal{T}_{p-1} = \mathcal{T}_p / 2 \tag{3.3}$$

with $N \geq p \geq 1$. The starting threshold $\mathcal{T}_N$ is of the form $\mathcal{T}_N = \alpha w_{\max}$, where $w_{\max}$ is the highest coefficient magnitude in the input wavelet decomposition, and α is a constant that is taken as $\alpha \geq 1/2$.

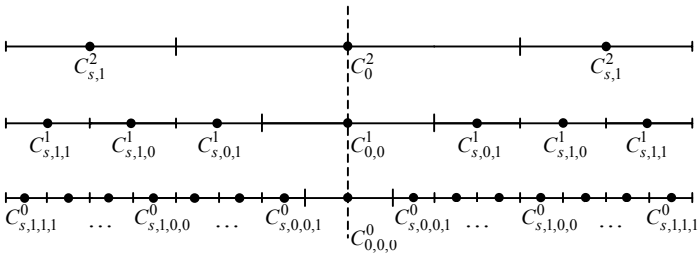


Figure III-2. Embedded deadzone scalar quantizer for $N = 2$, $\xi = 0$ and $\delta = 1/2$.

By using SAQ, the significance of the wavelet coefficients with respect to any given threshold $\mathcal{T}_p$ is indicated in a corresponding binary map, denoted by W^p, called the significance map. Denote by $w(\mathbf{k})$ the wavelet coefficient with coordinates $\mathbf{k} = (\kappa_1, \kappa_2, ..., \kappa_n)$ in the n-dimensional wavelet decomposition of a given input. The significance operator $\sigma^p(\cdot)$ maps any value $w(\mathbf{k})$ in the wavelet decomposition to a corresponding binary value $w^p(\mathbf{k})$ in W^p, according to the following rule:

$$w^p(\mathbf{k}) = \sigma^p(w(\mathbf{k})) = \begin{cases} 0, & if\ |w(\mathbf{k})| < \mathcal{T}_p \\ 1, & if\ |w(\mathbf{k})| \geq \mathcal{T}_p \end{cases} \tag{3.4}$$

In general, embedded coding of the input wavelet decomposition translates into coding the significance maps W^p, for every p with $N \geq p \geq 0$.

The SAQ has been adopted in a wide range of wavelet-based coding algorithms. The common choice for this particular type of scalar quantizer is motivated by the fact that, although sub-optimal in the rate-distortion sense, the link between SAQ and bitplane coding makes it attractive for embedded coding.

Concerning the actual coding process (C and EC modules of Figure III-1), all state-of-the-art wavelet-based coding techniques exploit the inter-band or intra-band statistical dependencies between the transform coefficients. Each category (inter or intra-band) is utilizing a different data structure in the coding process: usually wavelet trees (originally called zerotrees of wavelet coefficients [24]) are employed in the inter-band coding algorithms, whereas quadtrees or zero-blocks are used for intra-band coding. Since there is a very large amount of related research on these topics and many algorithms are derived based on empirical or statistical evidence, in the following we are limiting our description to a representative state-of-the-art scheme for each of the two categories found in the related literature.

3.1.2 Instantiation of an Inter-band Coder: Set Partitioning In Hierarchical Trees (SPIHT) Coding

The presentation of this subsection is a synopsis of [25]. Basically, SPIHT coding is an improved version of the original zerotree-based wavelet coding algorithm of [24]. The coding process can occur independently within areas of certain resolution levels, areas within wavelet subbands or even within

individual wavelet trees [26]. However, in full-resolution decoding, the best results are always obtained if the entire (multilevel) DWT is used, which is the way the original algorithm was proposed [25].

In SPIHT coding, the algorithm starts by scanning the low-frequency wavelet coefficients and proceeds gradually to higher-frequency coefficients. Essentially, the algorithm employs a dynamically-determined scanning pattern to encode the significance maps produced by SAQ. For further compression efficiency, the symbols indicating the scanning pattern for each significance map (bitplane) are context-based entropy coded [25] by using contexts constructed from previously-scanned positions in the significance map.

The novelty of the SPIHT algorithm in comparison to previous work (e.g. [24]) is in the scheme that produces the dynamic scanning pattern. This scheme emerges as an efficient instantiation of the family of algorithms based on the zerotree hypothesis [24]. The underlying model assumes that insignificant coefficients in the wavelet domain (with respect to a given SAQ threshold) tend to be clustered in wavelet trees, termed zerotrees [24]. An example is given in Figure III-3, where a wavelet tree is shown within the two-level DWT. The root of the tree consists of the three wavelet coefficients indicated on the upper left part (subband LL_2). Notice that the coefficient in the LL-band that is marked with a star is not the root of any wavelet tree in the SPIHT algorithm. Some parent-children dependencies are marked within the wavelet-tree example of Figure III-3. As seen there, the relation of parent (i, j) and children $O(i, j)$ is extended in another level to the indirect descendants of the parent coefficient, denoted by $L(i, j)$. In total, the direct and indirect descendants of each parent coefficient (i, j) are the descendants of this coefficient and are denoted by $D(i, j)$. The reference to a coefficient value is denoted by $c(i, j)$.

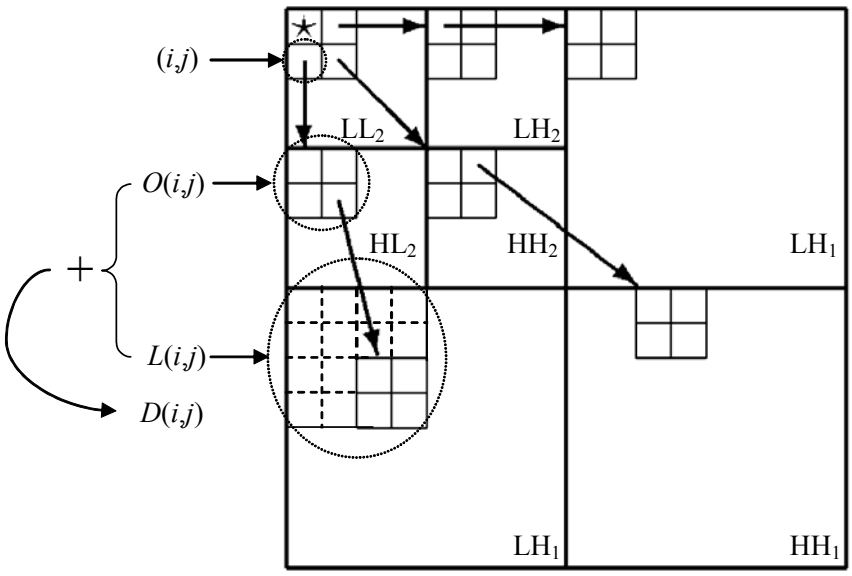


Figure III-3. Dyadic discrete wavelet decomposition and an example of a wavelet tree (zerotree) used in the SPIHT algorithm (based on [25]).

The pseudocode of the SPIHT algorithm is basing its efficiency on the gradual segmentation into isolated coefficients and coefficient descendants (direct or indirect), thereby creating subtrees in the wavelet domain that have roots in these descendants. The transform scan is performed breadth-first. Three dynamic linked lists are created; while the SAQ threshold is dyadically decreased, an increasing number of coefficients become significant with respect to the applied threshold. This is marked in the three lists and, based on their contents, the coefficient scanning takes place. In detail, these lists are:

- List of insignificant pixels (LIP): Contains pointers to the coefficients that are insignificant with respect to the current SAQ threshold.
- List of significant pixels (LSP): Contains pointers to the coefficients that have already been found significant with respect to the current or past SAQ thresholds.
- List of insignificant subtrees (LIS): Contains pointers to all the subtrees that have been created by the algorithm until the current point of execution. All coefficients indexed by this list are insignificant with respect to the current threshold and hence all their subtrees are also considered insignificant. In addition, these subtrees can be of two types; type A or type B. Type-A subtrees of a root (i, j) is the set $D(i, j)$ (direct and indirect descendants); type-B subtrees is the set $L(i, j)$ (indirect descendants).

For each SAQ threshold, the three lists are serially scanned in two passes. The pseudocode of SPIHT is given in Figure III-4. The coefficient significance with respect to threshold 2^n is checked with operator $\sigma^n(\cdot)$, which corresponds to (3.4) with $T_n = 2^n$.

Initialization: Output $n = \left\lfloor \log_2(\max_{\forall(i,j)}\{|c_{i,j}|\}) \right\rfloor$; set the LSP as an empty list; add the coordinates (i, j) of each wavelet coefficient to the LIP, and only those with descendants also to the LIS, as type-A entries

Sorting Pass:

For each entry (i, j) in the LIP do:

Output $\sigma^n(i, j)$; If $\sigma^n(i, j) = 1$ then move (i, j) to the LSP and output the sign of $c(i, j)$

For each entry (i, j) in the LIS do:

If the entry is of type A, then

Output $\sigma^n(\mathcal{D}(i, j))$; If $\sigma^n(\mathcal{D}(i, j)) = 1$ then

For each $(k,l) \in O(i, j)$ do:

Output $\sigma^n(k,l)$; If $\sigma^n(k,l) = 1$ then add (k,l) to the LSP and output the sign of $c(k,l)$

If $\sigma^n(k,l) = 0$ then add (k,l) to the end of the LIP

If $L(i, j) \neq 0$ then move (i, j) to the end of the LIS as an entry of type B, and go to the next step; else, remove entry (i, j) from the LIS and terminate the current Sorting Pass

If the entry is of type B then

Output $\sigma^n(L(i, j))$; If $\sigma^n(L(i, j)) = 1$ then

Add each $(k,l) \in O(i, j)$ to the end of the LIS as an entry of type A; remove (i, j) from the LIS

Refinement Pass: For each entry (i, j) in the LSP, except those included in the last sorting pass (i.e. with the same n), output the n-th most significant bit of $|c(i, j)|$

Quantization-step update: Decrement n by 1 and go to the Sorting Pass

Figure III-4. SPIHT pseudocode (from [25]).

3.1.3 Instantiation of an Intra-band Coder: QuadTree-Limited (QT-L) Coding

Various algorithms employing Quadtree (QT) decompositions have been reported in literature, targeting both the spatial and the transform domain. Briefly, the QT decomposition divides an image into two-dimensional homogeneous (in the image property of interest) blocks of variable size. Typically, the division operation is guided by a hypothesis test, in which a decision is made whether or not a block is homogeneous in the property of interest: if the test is positive then no division is necessary, else the block is divided into four adjacent blocks. The division operation is repeated recursively, until no further division is needed or the smallest possible block size is attained.

In contrast to quadtree decomposition applied in the spatial domain, some algorithms use the quadtree decomposition in the wavelet domain [27] [28] [29] [23]. The underlying models exploited in these codecs coupled with more advanced context-based entropy coding techniques have been used in the design of the QuadTree-Limited (QT-L) coding algorithm, initially introduced in [23] and published in [6]. The QT-L codec provides competitive compression performance against the state-of-the-art Embedded Block Coding with Optimized Truncation (EBCOT) algorithm [30] used in the JPEG-2000 standard. As such, it has been adopted for the coding of the quantized wavelet coefficients in the algorithms proposed in this dissertation. A detailed description of this algorithm is given in the following. Our presentation is based on [23].

In the QT-L algorithm, SAQ is applied so as to determine the significance of the wavelet coefficients with respect to a monotonically decreasing series of thresholds. As explained before, by using SAQ, the significance of the pixels in the wavelet decomposition $W = (w(\mathbf{k}))_{\mathbf{k}}$ with respect to any given threshold 2^{T_p} is indicated in a corresponding binary map W^p, called the significance map. Additionally, let us define a quadrant $Q(\mathbf{k}^0, \nu)$ as a $\nu \times \nu$ matrix containing some wavelet coefficients $w(\mathbf{k})$, where $\nu = 2^r, r \in \mathbb{N}$, and the vector $\mathbf{k}$ of coordinates is bounded by $\mathbf{k}^0 \le \mathbf{k} < \mathbf{k}^0 + \nu$. The significance with respect to the threshold 2^{T_p} of the coefficients from $Q(\mathbf{k}^0, \nu)$ is determined via a significance operator $\sigma^p(\cdot)$, which maps any value $w(\mathbf{k})$ to a corresponding binary value $w^p(\mathbf{k})$ in W^p:

$$\forall w(\mathbf{k}) \in Q(\mathbf{k}^0, \nu), \quad w^p(\mathbf{k}) = \sigma^p(w(\mathbf{k})) = \begin{cases} 0, & \text{if } |w(\mathbf{k})| < 2^{T_p} \\ 1, & \text{if } |w(\mathbf{k})| \ge 2^{T_p} \end{cases} . \tag{3.5}$$

The binary matrix that corresponds to $Q(\mathbf{k}^0, \nu)$, and which indicates the significance of the coefficients with respect to the applied threshold 2^{T_p}, is denoted as $Q_b^p(\mathbf{k}^0, \nu)$. Define a partition rule P that divides the matrix $Q_b^p(\mathbf{k}^0, \nu)$ into adjacent minors, where $\boldsymbol{\alpha} \in (0,1)^2$:

$$P: \quad Q_b^p(\mathbf{k}^0, \nu) = \left(Q_b^p\left(\mathbf{k}^0 + \boldsymbol{\alpha}\frac{\nu}{2}, \frac{\nu}{2}\right)\right) \text{ if } \exists w^p(\mathbf{k}) \in Q_b^p(\mathbf{k}^0, \nu) \text{ with } w^p(\mathbf{k}) = 1. \tag{3.6}$$

Basically, the QT-L algorithm builds quadtree decompositions corresponding to each significance map W^p; the partitioning rule P is applied recurrently on quadrants selecting sets of binary elements in the significance map and, correspondingly, sets of coefficients of the wavelet transform matrix.

Encoding the significance maps (i.e. the positions of the significant coefficients) is equivalent with the encoding of the corresponding quadtrees. A fixed, data-independent procedure is used to scan the quadtrees. An important difference with respect to its SQP predecessor [27] is that in the QT-L algorithm the partitioning process is *limited*, so that the quadtrees are not built up to the pixel level. Once the area ν^2 of the current node $Q_b^p(\mathbf{k},\nu)$ in the quadtree is lower than a predefined minimal quadrant area, the splitting process is stopped and the entropy coding of the coefficients within the quadrant $Q(\mathbf{k},\nu)$ is activated. Thus, in this case, the leaf nodes in the quadtree will consist of quadrants delimiting blocks of wavelet coefficients, and not single values, as was the case of the original SQP algorithm [27]. An example of a 4×4 binary matrix $Q_b^p\left(k_0^0,k_1^0,4\right)$ and the corresponding quadtree structure is illustrated in Figure III-5; notice that in this simple example, the quadtree has been built up to pixel level.

The QT-L algorithm performs a significance pass $S^{p_{max}}$ to encode the highest bit-plane $p = p_{max}$. Subsequently, the algorithm performs three coding passes to encode all the lower bit-planes p, $0 \le p < p_{max}$, namely, a Non-Significance pass (N^p), a Significance pass (S^p) and a Refinement Pass (R^p). During the significance pass S^p, the coordinates $\mathbf{k}$ of the coefficients $w(\mathbf{k})$ found as non-significant $\sigma^p(w(\mathbf{k}))=0$ and located in the immediate neighbourhood of significant coefficients are appended to a list called the list of non-significant coefficients (LNC). Also, the coefficients identified as significant $\sigma^p(w(\mathbf{k}))=1$ in the N^p, S^p coding passes are appended to a list called the refinement list (RL). During the next coding steps k, $0 \le k < p$, the significance of the coefficients recorded in the LNC is coded first.

The explanation as to why the construction of the quadtrees has been limited up to a certain quad-tree level is as follows: It is beneficial from the lossy coding point of view to include in the LNC *all* the non-significant coefficients that are among the 3^2-1 neighbours of a significant coefficient.

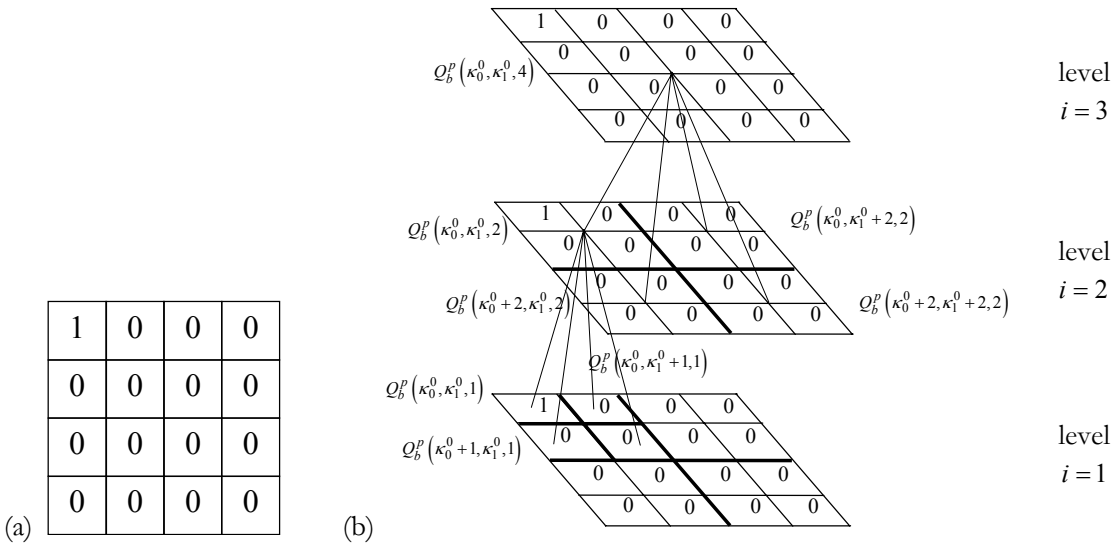


Figure III-5. Quadtree structure corresponding to $Q_b^p\left(k_0^0,k_1^0,4\right)$. (a) An example of a 4x4 binary matrix $Q_b^p\left(k_0^0,k_1^0,4\right)$; (b) The partition rule P is applied recurrently only on the binary matrices containing the non-zero element (thick lines indicate the successive partitioning in minors).

According to the clustering property of the wavelet transform, for every bit-plane p, $p \geq 1$, there is a high probability for *each* of these non-significant coefficients to become significant with respect to the lower SAQ thresholds 2^{T_k}, $0 \leq k < p$. If the quadtree was to be built up to pixel level, some of the non-significant coefficients that are potentially significant at lower thresholds would be neglected when building up the LNC. The limited quadtree-partitioning process eliminates this drawback. By using a Morton scanning path [31], the QT-L algorithm scans all the coefficients in the leaf nodes of the quadtree, and appends to the LNC the coordinates of those coefficients that are non-significant.

Each coding step k, $0 \leq k < p_{\max}$, starts with the significance of the coefficients in the LNC. Therafter, the encoder performs the next two coding passes, namely the non-significance and the refinement passes. The pseudo-code describing the non-significance pass N^p is given in Figure III-6. In a similar manner, the pseudo-code of the significance pass S^p is given in Figure III-7, while the pseudo-code of the refinement pass R^p is given in Figure III-8. Note that in these figures, the "SGN" and "NSG" are the acronyms for the significant and non-significant symbols respectively, and $Limited\_Area$ indicates the bound below which no quadtree partitioning is performed.

An important difference with respect to its SQP predecessor [27] is that the QT-L coder adopts a more elaborated context-conditioning phase and context-based entropy coding of the symbols generated in the three coding passes. Since simple memoryless models are usually not efficient enough, context-based arithmetic encoding should be used to improve the coding performance. This technique exploits the dependencies between the symbols to be encoded and the neighbouring symbols (the context). Four different sets of models $S_i, 1 \leq i \leq 4$ are used to encode the symbols generated by the coding passes, and the encoder automatically selects the appropriate set at each coding stage, as shown in Figure III-6, Figure III-7 and Figure III-8. These sets include: (1) the "Quadrant_Significance" model (S_1) used to encode the significance of the nodes in the quadtrees, (2) the "Pixel_Significance" (S_2) and "Pixel_Sign" (S_3) sets used to code the significance and the signs respectively of the coefficients in the non-significance and significance passes, and (4) the "Pixel_Refinement" set (S_4), used to entropy code the refinement information generated in the refinement pass.

Procedure QT-Limited Non-Significance Pass()
  For every $c_{\mathbf{q}}$ in LNC, parsed from head to tail do
    If $|c_{\mathbf{q}}| \geq 2^{T_p}$ do
      Entropy code SGN symbol with the S_2 model
      Entropy code the sign of $c_{\mathbf{q}}$ with the S_3 model
      Append the coordinates $\mathbf{q}$ to the RL; Remove $\mathbf{q}$ from the LNC
    Else
      Entropy code NSG symbol with the S_2 model
    End
  End
End

Figure III-6. Pseudo-code describing the Non-Significance pass (N^p) performed by the QT-L coder; note that S_2 denotes the Pixel_Significance model, respectively S_3 denotes the Pixel_Sign model.

The encoder assigns a number $N_{models}^{S_i}$ of context models (or states) C_n^i, $0 \le n < N_{models}^{S_i}$, for each set of models $S_i, 2 \le i \le 4$. A different probability model corresponds to each context state C_n^i, and thus the generated symbols are entropy coded with an adaptive arithmetic coder having the appropriate context model derived in the context-conditioning phase. The algorithm assigns each coefficient/quadrant to one of the several possible contexts depending on the values of the previously quantized coefficients. The basic idea for the context conditioning adopted in the coder is to quantize into a context number m (corresponding to a given context model C_m^i), the number of significant neighbourhood coefficients for a given coefficient position. The quantization performed for the sets $S_i, 2 < i \le 4$ is described by the following expression:

$$m = \left\lfloor \left(N_{models}^{S_i} - 1 \right) \cdot \frac{N_{sgn}}{N_{tot}} \right\rfloor, \quad 2 \le i \le 4 \tag{3.7}$$

where N_{sgn} is the number of the neighbouring coefficients declared as significant at the previous coding steps, N_{tot} is the total number of neighbours, and $\lfloor x \rfloor$ is the integer part of x. The total number of neighbours N_{tot} in (3.7) is set to 8 in two-dimensional coding and to 26 in three-dimensional coding. Finally, a single model S_1 is used for the adaptive arithmetic entropy coding of the significance of the quadrants.

Procedure QT-Limited Significance Pass($Q_b^p(\mathbf{k},\nu)$ **)**
  If $\forall c_{\mathbf{q}} \in Q(\mathbf{k},\nu),\ |c_{\mathbf{q}}| < 2^{T_{p+1}}$ do
    If $\exists\ c_{\mathbf{q}} \in Q(\mathbf{k},\nu),\ |c_{\mathbf{q}}| \ge 2^{T_p}$ do
      Entropy code SGN symbol with the S_1 model
      If $\nu^n \le \text{Limited\_Area}$ do
        Encode Block($Q_b^p(\mathbf{k},\nu)$)
      Else
        For every child $Q_b^p\left(\mathbf{k}_i, \nu/2\right),\ 0 \le i < 2^n$ of $Q_b^p(\mathbf{k},\nu)$ do
          QT-Limited Significance Pass($Q_b^p\left(\mathbf{k}_i, \nu/2\right)$)
        End
      End
    Else
      Entropy code NSG symbol with the S_1 model
    End
  Else
    If $\nu^n > \text{Limited\_Area}$ do
      For every child $Q_b^p\left(\mathbf{k}_i, \nu/2\right),\ 0 \le i < 2^n$ of $Q_b^p(\mathbf{k},\nu)$ do
        QT-Limited Significance Pass($Q_b^p\left(\mathbf{k}_i, \nu/2\right)$)
      End
    End
  End
End

Figure III-7. Pseudo-code describing the Significance pass (S^p) performed by the QT-L coder; note that S_1 denotes the Quadrant_Significance model.

Procedure Encode Block($Q_b^p(\mathbf{k},\nu)$ **)**
  For every coefficient $c_{\mathbf{q}}$ in $Q(\mathbf{k},\nu)$ do
    If $|c_{\mathbf{q}}| \geq 2^{T_p}$ do
      Entropy code SGN symbol with the S_2 model
      Entropy code the sign of $c_{\mathbf{q}}$ with the S_3 model
      Add the coordinates $\mathbf{q}$ to the end of the RL
    Else
      Entropy code NSG symbol with the S_2 model
      Add the coordinates $\mathbf{q}$ to the end of the LNC
    End
  End
End
Procedure QT-Limited Refinement Pass()
  For every $c_{\mathbf{q}}$ appended at previous coding stages to RL do
    Entropy code the bit-plane from $c_{\mathbf{q}}$ corresponding to T_p
    using the S_4 model
  End
End

Figure III-8. Pseudo-code describing the Refinement pass (R^p) performed by the QT-L coder, respectively the Encode_Block procedure called during the Significance pass; note that S_2 denotes the Pixel_Significance model, S_3 denotes the Pixel_Sign model, and S_4 denotes the Pixel_Refinement model .

3.1.4 Post-Compression Rate-Distortion Optimization and Bitstream Extraction

As explained in the previous subsections, the bitplane scanning produced by SAQ, combined with the embedded coding of the significance maps and (usually) context-based entropy coding of the produced stream of symbols, results in compression techniques that are scalable in bitrate. If the entire DWT of the input image is used, the bitstream produced by SAQ and zerotree or quadtree-based coding methods can be truncated at any arbitrary point and the reconstructed image at the decoder will have proportional quality to the reconstructed image produced by the equivalent set of uniform (double-deadzone) quantizers and coding at that bitrate. Essentially, this feature ensures that bitrate-scalability is provided "at little or no cost" to the compression efficiency of the methods described in subsections 3.1.2 and 3.1.3. A good conceptual parallelism of embedded coding is the binary finite precision of a real number. All real numbers can be represented by a string of binary digits (SAQ). For each digit that is added to the right, more precision is added. Nevertheless, this "encoding" can cease at any time and provide the "best" representation of the real number achievable within the framework of binary digit representation.

Additional research in the field revealed that the scalability features can be extended to include resolution scalability if the maximum size of a coding unit used by the coding module C is limited to the subbands of certain resolution levels. This is especially facilitated by intra-band coding algorithms,

which do not exploit dependencies across the scales of the wavelet transform. For example, for QT-L coding [23], one can apply the algorithm described in subsection 3.1.3 separately for the subbands of each resolution level of the critically-sampled DWT without any additional modifications. For a DWT decomposition in k levels, this will essentially produce k independently-decodable bitstreams. Nevertheless, for full-resolution decoding, one must establish the truncation point for each bitstream so that the best-possible quality (in the mean-square error sense) is obtained for the reconstructed image out of the entire set of k bitstreams. In general, this is a problem that manifests itself when the wavelet decomposition is split into a number of independent code-blocks, and each code-block is independently compressed, as is the case of the EBCOT algorithm [30] used in the JPEG-2000 standard, or the Embedded Zero-Block Coding algorithm [32]. Similar systems employing independent coding of wavelet trees can be found elsewhere [26] [33]. For all these cases, an optimal solution to this problem is based on rate-distortion optimized truncation. This is the operation performed within the PCRDO (Opt) module of Figure III-1. For details on this operation, the reader is referred to [17] [34, 35]. An example instantiation performing PCRDO in a wavelet-tree (inter-band) coding system can be found in [33]. In short, the PCRDO module is utilizing rate-distortion information gathered during encoding. The rate can be directly measured by the number of required bits after each pass of the encoding algorithm, e.g. Figure III-6 – Figure III-8. Moreover, it can be shown that the decrease in the distortion after a pass within each bitplane is linearly-related to the number of coefficients that have been found significant or have been refined in this pass. In this way, a distortion estimate can be formed in order to indicate the expected mean-square error after the termination of a coding pass. These points can be used within a Lagrangian optimization framework [34, 35], in order to provide optimal truncation points for a set of resolutions and bitrates. As a result, the bitstream extractor B (Figure III-1) can immediately provide the bitstream segments that correspond to the selected resolutions and bitrates at the decoder. Alternatively, in a server-based image or video distribution scenario, the rate-distortion points and the entire compressed bitstream can be stored at the server site, enabling clients to send requests for specific bitrates and resolutions. In this case, the bitstream truncation can occur at the B module using the PCRDO algorithm (and the rate-distortion points), and a specific bitstream matching the requested features can be send to the client [18]. We note that the distinction among the two cases is based on whether (1) the entire collection of rate-distortion points is kept with the compressed bitstream and the bitstream truncation is performed interactively (following client requests), or (2) the optimization is performed a-priori and a specific set of truncation points is satisfied, which is expected to cover all realistic scenarios.

3.2 Block-based Motion Compensated Prediction

In coding of video sequences, motion compensated prediction (MCP) is a powerful tool for reducing the temporal dependency between successive video frames. In its most generic form, MCP creates an affine mapping between areas of the current frame and future or past frames which are available to both encoder and decoder; they are typically denoted in literature as the reference frames [36]. As shown in the schema Figure III-9, given the reference frames and the mapping parameters, MCP creates the predicted frame, which approximates the current frame, and the displacement vectors, which are the parameters of the mapping operation. The MCP error is retained in the error frame. In

typical video coding systems [37], after MCP, the error frame and the displacement vectors are communicated to the decoder.

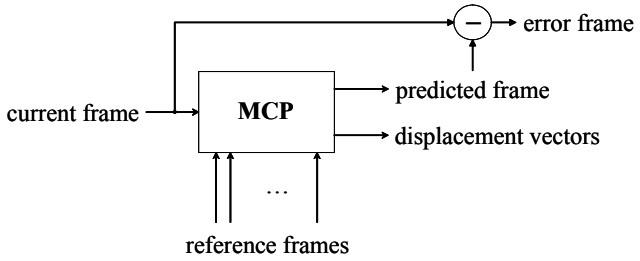


Figure III-9. Generic structure of motion compensated prediction.

Although accurate motion modelling for realistic video sequences has been the topic of interest of a vast number of publications (e.g. [38-41]), no existing scheme can capture the video sequence motion characteristics accurately and in a generic manner. Factors that hinder this effort are: occlusion and aperture effects; the appearance of new objects; scene illumination changes; changes in camera focus, etc. As a result, every solution for MCP solves the problem by alleviating certain, but not all of these effects. Moreover, the general rule is that, the more advanced the motion estimation (ME) algorithm used for MCP, the more complex it becomes to realize it in a practical video coding system.

As a result, the ME algorithms used in the vast majority of MCP-based video coding systems are block-based motion-estimation algorithms, which in turn lead to block-based motion-compensated prediction.

3.2.1 Block-based Motion Estimation

Figure III-10 shows an example of the basic block-based motion estimation algorithm. The current frame $A_t[m,n]$, consisting of $M \times N$ pixels, is split into non-overlapping blocks of size $B_m \times B_n$ pixels. Each block is predicted by performing a matching with the blocks indexed within the area of $S_m \times S_n$ pixels around the block position. The matching criterion used in practice is the sum of absolute differences (SAD) [37], since it provides accurate matches without the need for multiplication operations. For the particular example of Figure III-10, this optimization problem can be stated as:

$$(d_m^*, d_n^*) : \arg\min_{(d_m,d_n)} \sum_{i=0}^{B_m-1} \sum_{j=0}^{B_n-1} \mathcal{C}\left(A_t[m+i,n+j] - A_{t-1}[m+i-d_m, n+j-d_n]\right) \tag{3.8}$$

with $-\frac{S_m}{2} \le d_m < \frac{S_m}{2}$ and $-\frac{S_n}{2} \le d_n < \frac{S_n}{2}$, and $\mathcal{C}(a) = |a|$ for SAD matching, or $\mathcal{C}(a) = a^2$ for the sum of square differences (SSD). In all practical block-based algorithms, the desired solution (d_m^*, d_n^*) is constant over pixels $A_t[m+i,n+j]$, i.e. all the pixels of block at position (m,n) of the current frame are all associated with one solution (d_m^*, d_n^*).

The classical way of solving the example of (3.8) is the full-search (a.k.a. brute-force) algorithm [37]. In general, after the performance of ME, every pixel $A_t[m,n]$ of the current frame is associated with a pixel $A_{t-1}[m-d_m^*, n-d_n^*]$ of the reference frame via its *displacement vector* (d_m^*, d_n^*) (also referred as *motion vector*).

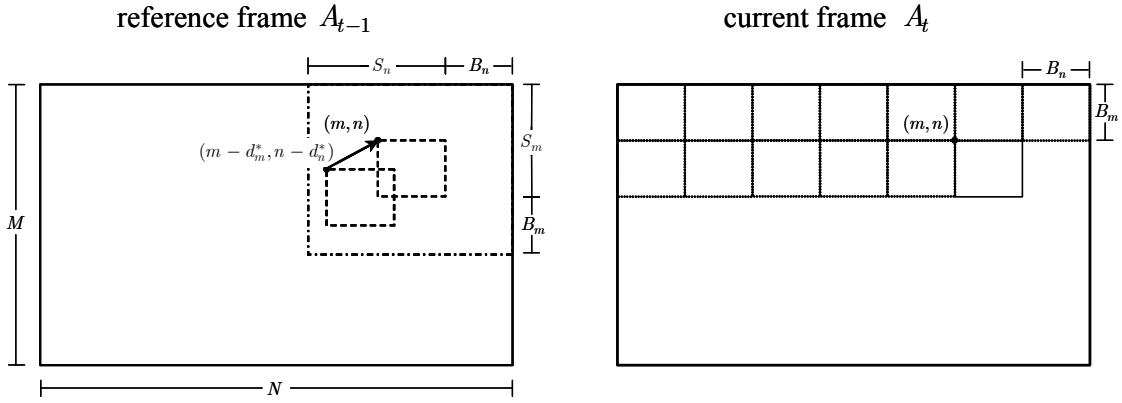


Figure III-10. An example of block-based motion estimation.

The block-matching algorithm presented before is designed for pixel-level translational motion between consecutive video frames. However, since the digitized video frames represent only a two-dimensional sampling of the real scene content captured by the camera, it has been long known that interpolation techniques can help increase the precision of block-based motion estimation. In reality, experimental and theoretical analysis [42] showed that, for MCP with block size 16×16 pixels, broadcast TV signals require interpolation with up to quarter-pixel accuracy; videophone signals typically require half-pixel accuracy. Based on Figure III-10, a hypothetical example of motion estimation with half-pixel interpolation is presented in Figure III-11. In this example, we assume that $B_m = B_n = 2$ and $S_m = S_n = 10$; only a part of the search area is shown in the figure. The integer pixel positions and the three interpolated (fractional) pixel positions form the grid of half-pixel positions; the different types of grid positions are indicated with different symbols. As shown in this example, the best matches in the full-pixel and half-pixel grid are found in neighbouring positions in the search area. Out of these two best matches, the one that corresponds to the smallest SAD can be chosen for MCP of the specified block. In general, for a certain sub-pixel accuracy, practical block-based motion estimation algorithms tend to perform hierarchical search located around the best match found over the coarser interpolation precision [37], following the assumption of homogeneous motion. For example, in the case of Figure III-11, a practical ME scheme starts by establishing the best match at the full-pixel position by performing full-search within the $S_m \times S_n$ neighbouring pixels. This process provides the displacement vector $(-1,4)$. Subsequently, an area of $S_m' \times S_n'$ half-pixel positions around the position of the best match is searched and the displacement vector $(-2,3.5)$ that corresponds to the smallest SAD is found. The process can continue iteratively by creating the quarter-pixel grid and searching in an area of quarter-pixel positions around the position of the best match. This process is a subclass of hierarchical or multiresolution ME algorithms [37]

which perform ME on a coarse grid (downsampled current and reference video frames) and iteratively refine the sampling grid to establish the position that minimizes the search criterion.

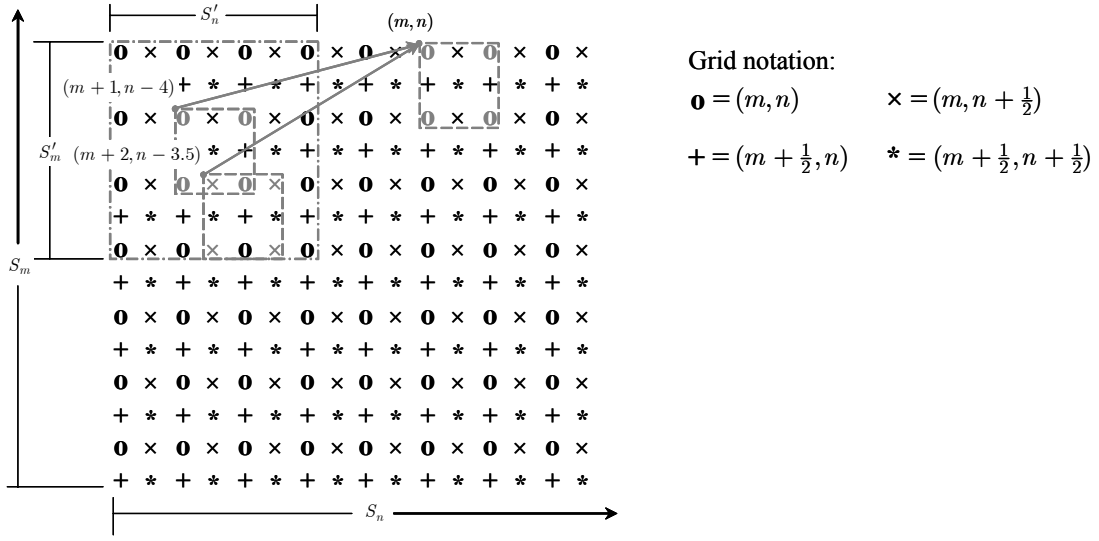


Figure III-11. Half-pixel interpolation for motion estimation. The current block size is 2×2 pixels and it is located at position (m,n). A part of the search area is depicted, with different symbols indicating the horizontally, vertically and diagonally interpolated positions. The position of the best match in the full-pixel and the half-pixel grid is marked with the grey squares. The displacement vectors for each case are also indicated.

In general, hierarchical ME schemes are one category of algorithms for fast ME. The literature on fast ME algorithms is vast (e.g. see [38] for an overview paper), since the complexity of encoders based on MCP is heavily dependent on the ME stage. We note however that for the case of block-based ME (equation (3.8)), only the classical full-search algorithm is guaranteed to obtain the position (within the predefined grid precision) that provides the global minimum of the matching criterion.

3.2.2 Motion Compensated Prediction and Fractional-pixel Interpolation

After the ME stage, each pixel in the current frame $A[m,n]$ is associated via its displacement vector $(d_m^{\mathcal{F}_1(t-1)}, d_n^{\mathcal{F}_1(t-1)})$ with a pixel in the reference frame, where $\mathcal{F}_{\Delta\tau}(\tau)$ denotes forward prediction via motion estimation that matches the reference frame at time instant τ with the frame at time instant $\tau+\Delta\tau$. As a result, after MCP, the error-frame is given by:

$$H_t[m,n] = A_t[m,n] - \mathcal{I}_{(i_m^{\mathcal{F}_1(t-1)}, i_n^{\mathcal{F}_1(t-1)})} A_{t-1}\left[m - \left\lceil d_m^{\mathcal{F}_1(t-1)}\right\rceil, n - \left\lceil d_n^{\mathcal{F}_1(t-1)}\right\rceil\right] \tag{3.9}$$

where: $i_m^{\mathcal{F}_1(t)} = \left\lceil d_m^{\mathcal{F}_1(t)}\right\rceil - d_m^{\mathcal{F}_1(t)}$, $i_n^{\mathcal{F}_1(t)} = \left\lceil d_n^{\mathcal{F}_1(t)}\right\rceil - d_n^{\mathcal{F}_1(t)}$ is the fractional-pixel (interpolated) position (with $\lceil a \rceil$ indicating the smallest integer that is larger than a); $\mathcal{I}_{(k,l)}A[m,n]$ indicates the interpolated pixel at fractional distance (k,l) from pixel $A[m,n]$, where $k,l=\{0,\frac{1}{R},\ldots,\frac{R-2}{R},\frac{R-1}{R}\}$ and R indicates

the interpolation precision (e.g. $R = 2$ for half-pixel accurate interpolation). One can equivalently modify (3.9) to perform backward MCP by associating each pixel in the current frame $A_t[m,n]$ via its displacement vector $(d_m^{\mathcal{B}_1(t+1)}, d_n^{\mathcal{B}_1(t+1)})$ with a pixel in reference frame A_{t+1}. In this case, backward prediction is performed (denoted by $\mathcal{B}_{\Delta\tau}(\tau)$), i.e. motion estimation that matches the reference frame at time instant τ with the frame at time instant $\tau - \Delta\tau$.

The interpolation operator $\mathcal{I}_{(k,l)}$ can be designed in many ways. In this dissertation, we follow the design of [18]. Under the assumption that the same motion vector is used within the region of support of the interpolation filter, separable FIR interpolation filters can be designed by windowing the impulse response of the sinc function in the position (k,l):

$$w_{m+k} = \sum_a w(a) \frac{\sin \pi(m + k - a)}{\pi(m + k - a)} \tag{3.10}$$

and equivalently for w_{n+l}. The window used is the Hamming window [18]. The derived filter coefficients for $R = 8$ are [18]:

$$w_{m+\frac{1}{8}} = \begin{bmatrix} -0.0072 & 0.0284 & -0.0902 & 0.9742 & 0.1249 & -0.0380 & 0.0105 & -0.0026 \end{bmatrix} \tag{3.11}$$

$$w_{m+\frac{1}{4}} = \begin{bmatrix} -0.0110 & 0.0452 & -0.1437 & 0.8950 & 0.2777 & -0.0812 & 0.0233 & -0.0053 \end{bmatrix} \tag{3.12}$$

$$w_{m+\frac{3}{8}} = \begin{bmatrix} -0.0117 & 0.0505 & -0.1624 & 0.7713 & 0.4465 & -0.1224 & 0.0363 & -0.0081 \end{bmatrix} \tag{3.13}$$

$$w_{m+\frac{1}{2}} = \begin{bmatrix} -0.0105 & 0.0465 & -0.1525 & 0.6165 & 0.6165 & -0.1525 & 0.0465 & -0.0105 \end{bmatrix} \tag{3.14}$$

and $w_{m+\frac{7}{8}}[i] = w_{m+\frac{1}{8}}[9-i]$, $w_{m+\frac{3}{4}}[i] = w_{m+\frac{1}{4}}[9-i]$, $w_{m+\frac{5}{8}}[i] = w_{m+\frac{3}{8}}[9-i]$ for $1 \le i \le 8$. Equivalent filters are also applied in the perpendicular direction (w_{n+l}).

To conclude our presentation of MCP, we note that, in terms of practical usage, at the core of all current MPEG video coding standards [43], MCP is utilized within the closed-loop video coding framework. Hence, as it will be explained in section 3.5, instead of the original reference frame, the decoded reference frame is used in these systems. Similarly, in systems based on motion-compensated temporal filtering [8] [44] [45] , this MCP is realizing the prediction step of the temporal decomposition.

3.3 Motion Compensated Update

In this section, we complement the framework of MCP presented before with the reverse operation, whose aim is to "update" the reference frame based on the error frame produced by MCP. To understand better the concepts involved in this process, we begin with an overview of the classical, lifting-based, wavelet decomposition and then discuss its extension that involves motion estimation and compensation.

As shown by the lifting framework [46] [47], any wavelet decomposition can be factored into a series of prediction and update operations. Prediction and update can be seen as dual operators used in the construction of perfect-reconstruction band-splitting systems [46]. Prediction uses the reference part of the signal to estimate the current part and produces an error signal that corresponds to the current part (and the prediction parameters). The update operation takes as input the error signal produced by the predictor and modifies the reference signal. Hence a new reference frame is constructed with improved prediction properties. Moreover, the update operation normalizes magnitude of the reference signal in order to produce an orthonormal decomposition [46] [47].

For the case of video signals, the DWT decomposition in the temporal domain can be performed by a (non-unique) series of prediction and update operations that utilize entire video frames [45]. This process can be seen in Figure III-12. An example with one GOP consisting of 6 input video frames is presented. Prediction is first applied to the input frames A_{2t+1}, $t = \{0,1,2\}$ using frames A_{2t} as references. The result of prediction is the production of error-frames H_t. Subsequently, the information of the H_t frames is inverted back to the reference frames, thereby creating the updated reference frames L_t. This process can iterate with the use of the L_t frames as input frames for the next temporal decomposition level [45]. The example of Figure III-12 corresponds to the Haar temporal decomposition and can be expressed analytically as:

$$H_t[m,n] = \tfrac{1}{\sqrt{2}}\left(A_{2t+1}[m,n] - A_{2t}[m,n]\right) \tag{3.15}$$

$$L_t[m,n] = \sqrt{2}A_{2t+1}[m,n] + H_t[m,n]. \tag{3.16}$$

The incorporation of motion information in temporal wavelet decompositions has been a research goal for quite some time [7] [8] [48]. As proposed recently in [45] [11], temporal wavelet decompositions with perfect reconstruction can be obtained by performing the lifting scheme in the direction of motion in the video sequence, provided that the conventional prediction and update steps shown in (3.15), (3.16) are replaced with motion compensated prediction and motion compensated update (MCU).

The example of Figure III-12 can be modified to include motion compensation if the prediction and update are performed via MCP and MCU. In particular, MCU can be performed via the system of Figure III-13 [19]. First the temporary frame Z_t is created by inverting the error frame samples of the H_t frames using the inverted motion-vector fields. The resulting frame is normalized and added to the reference frame to form the updated frame L_t. Since the motion vectors can originate from any block within the search area of the reference frame, this error-frame inversion using the inverse motion vectors will create pixels in frame L_t that are connected, multiconnected and unconnected to pixels in the corresponding error frame [45] [11] [8]. The state of connection of each pixel in frame L_t depends on whether there was one, many, or no vectors that originated from this pixel position in the original reference frame. To avoid strong motion-related artifacts in the output L_t frame and the irregular increase of the image-sample magnitudes in multi-connected areas, a normalization process divides the magnitude of the update samples for each pixel with the number of connections. Finally, before the update coefficients are added to the reference frame, they are scaled according to the lifting equation for the update step, taking into account the type of the specific connection (e.g. Haar or 5/3

filter-pair [47]). Additional scaling can be incorporated for the areas where the update samples have large magnitudes, or, alternatively, the update step can be adaptively disabled in areas where bad connections are encountered due to motion-prediction failure, as proposed in [18] [49, 50]. These approaches typically require additional signalling information to be transmitted to the decoder.

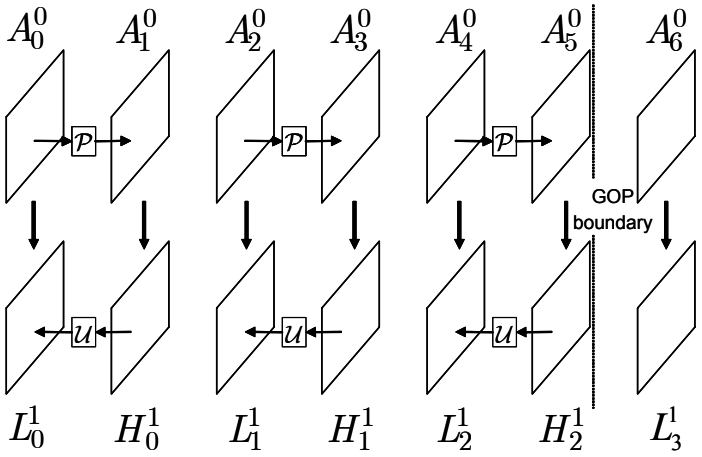


Figure III-12. Temporal wavelet decomposition via the application of predict and update steps.

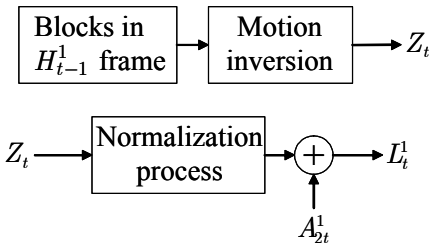


Figure III-13. The application of MCU in the motion-compensated temporal decomposition illustrated in Figure III-12.

In our realization of the update step, we choose to invert the information in the error frames to the immediately-lower integer-pixel position in the reference frame. In order to explain the motion-vector inversion process, we illustrate a simple (one-dimensional) example of a connected block in the reference frame in Figure III-14: during the prediction step, the block at position m in the current frame A_t is predicted by the block at position $m-2.5$ in the interpolated reference frame $\mathcal{I}A_{t-1}$. After the creation of the error frame H_t, the update step inverts the error-frame information to the block at position $m-3$ in the reference frame A_{t-1} (previous integer-pixel position). In this way, the updated block at position $m-3$ of frame L_t is created. Nevertheless, as mentioned before, in order to perform the appropriate weighting by the number of connections, in our actual implementation, we perform the error-frame inversion in the temporary frame Z_t and then normalize and add the result to the reference frame A_{t-1}, as shown in the schematic Figure III-13.

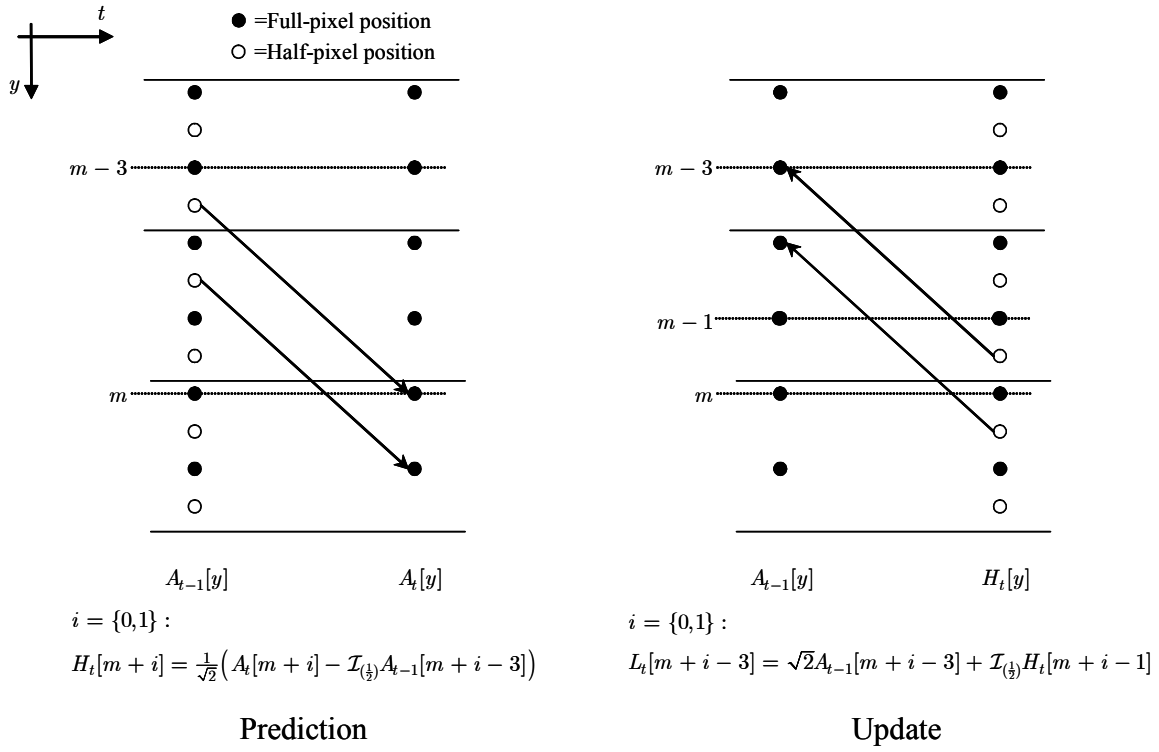


Figure III-14. The application of motion-compensated update step after the corresponding prediction step. A simple one-dimensional example with two consecutive frames is presented, where the block-size for MCP is $B_m = 2$, the displacement vector is $d_m = 2.5$, and $R = 2$.

In total, for each pixel (m,n), the motion compensated temporal wavelet decomposition described before, also called motion compensated temporal filtering, can be written in the case of Haar temporal filtering as:

$$c_u[m,n] = 0\,,\; Z_t[m,n] = 0 \tag{3.17}$$

$$H_t[m,n] = \tfrac{1}{\sqrt{2}}\left(A_{2t+1}[m,n] - \mathcal{I}_{(i_m^{\mathcal{F}_1(2t)}, i_m^{\mathcal{F}_1(2t)})}A_{2t}[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor]\right) \tag{3.18}$$

$$\begin{aligned} i_m^{\text{res}} &= \begin{cases} 0, & \text{if } i_m^{\mathcal{F}_1(2t)} = 0 \\ 1 - i_m^{\mathcal{F}_1(2t)}, & \text{otherwise} \end{cases}, i_n^{\text{res}} = \begin{cases} 0, & \text{if } i_n^{\mathcal{F}_1(2t)} = 0 \\ 1 - i_n^{\mathcal{F}_1(2t)}, & \text{otherwise} \end{cases}, \\ d_m^{\text{res}} &= \begin{cases} 0, \text{ if } i_m^{\mathcal{F}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases}, \qquad d_n^{\text{res}} = \begin{cases} 0, \text{ if } i_n^{\mathcal{F}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases} \end{aligned} \tag{3.19}$$

$$\begin{aligned} Z_t[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] &\leftarrow Z_t[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] \\ &\quad + \mathcal{I}_{(i_m^{\text{res}}, i_m^{\text{res}})}H_t\,[m - d_m^{\text{res}}, n - d_n^{\text{res}}\,] \\ c_u[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] &\leftarrow c_u[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] + 1 \end{aligned} \tag{3.20}$$

$$L_t[m,n] = \sqrt{2}A_{2t}[m,n] + \tfrac{1}{\max\{c_u[m,n],1\}}\,Z_t\,[m,n\,] \tag{3.21}$$

where $c_u[m,n]$ is the update connection map, $\max\{a,b\}$ returns the largest of a, b, and $a \leftarrow b$ indicates an assignment operation, i.e. the value of variable or expression b is assigned to variable or array element a. In particular, $c_u[m,n]$ is an $M \times N$ array that retains the number of connections to each pixel (m,n) of frame Z_t. Notice that, due to the fact that several areas in the current frame may be predicted by the same area in the reference frame, $c_u[m,n]$ can take any non-negative value. The equations (3.17)–(3.21) are applied separately to the entire frame, i.e. for all (m,n) with $1 \le m \le M$, $1 \le n \le N$, in the order expressed above. In particular, (3.17) performs the initialization of the connection map. Equation (3.18) performs MCP using forward prediction from frame A_{2t}. Motion inversion is performed by (3.20), which also updates the connection map. Finally, (3.21) performs MCU and weights each pixel in the update frame $L_t[m,n]$ by the number of connections. Notice that, if $A_{2t}[m,n]$ is unconnected, the proposed temporal filtering of (3.17)–(3.21) is simplified to the 1/2 filter, i.e. the motion-compensated Haar temporal filtering without the update step, since in this case $Z_t[m,n] = 0$.

This scheme can be extended to bidirectional MCP and MCU; this case corresponds to the motion compensated 5/3 temporal decomposition:

$$c_p[m,n] = \begin{cases} 2, \text{ if } \exists\left(d_m^{\mathcal{F}_1(2t)}, d_n^{\mathcal{F}_1(2t)}\right) \ \&\ \exists\left(d_m^{\mathcal{B}_1(2t+2)}, d_n^{\mathcal{B}_1(2t+2)}\right) \\ 1, \text{ otherwise} \end{cases} \tag{3.22}$$

$$c_u[m,n] = 0\,,\ Z_t[m,n] = 0 \tag{3.23}$$

$$\begin{aligned} H_t[m,n] = \tfrac{1}{\sqrt{2}}\Big[A_{2t+1}[m,n] - \tfrac{1}{c_p[m,n]}\Big(\mathcal{I}_{(i_m^{\mathcal{F}_1(2t)}, i_n^{\mathcal{F}_1(2t)})} A_{2t}[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] \\ + \mathcal{I}_{(i_m^{\mathcal{B}_1(2t+2)}, i_n^{\mathcal{B}_1(2t+2)})} A_{2t+2}[m - \lceil d_m^{\mathcal{B}_1(2t+2)} \rfloor, n - \lceil d_n^{\mathcal{B}_1(2t+2)} \rfloor] \Big) \Big] \end{aligned} \tag{3.24}$$

$$\begin{aligned} i_m^{\text{res}} &= \begin{cases} 0, \text{ if } i_m^{\mathcal{F}_1(2t)} = 0 \\ 1 - i_m^{\mathcal{F}_1(2t)}, \text{ otherwise} \end{cases}, i_n^{\text{res}} = \begin{cases} 0, \text{ if } i_n^{\mathcal{F}_1(2t)} = 0 \\ 1 - i_n^{\mathcal{F}_1(2t)}, \text{ otherwise} \end{cases}, \\ d_m^{\text{res}} &= \begin{cases} 0, \text{ if } i_m^{\mathcal{F}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases}, \qquad d_n^{\text{res}} = \begin{cases} 0, \text{ if } i_n^{\mathcal{F}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases} \end{aligned} \tag{3.25}$$

$$\begin{aligned} Z_t[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] &\leftarrow Z_t[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] \\ &\quad + \mathcal{I}_{(i_m^{\text{res}}, i_n^{\text{res}})} H_t[m - d_m^{\text{res}}, n - d_n^{\text{res}}] \\ c_u[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] &\leftarrow c_u[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] + 1 \end{aligned} \tag{3.26}$$

$$\begin{aligned} i_m^{\text{res}} &= \begin{cases} 0, & \text{if } i_m^{\mathcal{B}_1(2t)} = 0 \\ 1 - i_m^{\mathcal{B}_1(2t)}, \text{ otherwise} \end{cases}, \quad i_n^{\text{res}} = \begin{cases} 0, & \text{if } i_n^{\mathcal{B}_1(2t)} = 0 \\ 1 - i_n^{\mathcal{B}_1(2t)}, \text{ otherwise} \end{cases}, \\ d_m^{\text{res}} &= \begin{cases} 0, \text{ if } i_m^{\mathcal{B}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases}, \qquad d_n^{\text{res}} = \begin{cases} 0, \text{ if } i_n^{\mathcal{B}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases} \end{aligned} \tag{3.27}$$

$$\begin{aligned} Z_t[m - \lceil d_m^{\mathcal{B}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{B}_1(2t)} \rfloor] &\leftarrow Z_t[m - \lceil d_m^{\mathcal{B}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{B}_1(2t)} \rfloor] \\ &\quad + \mathcal{I}_{(i_m^{\text{res}}, i_n^{\text{res}})} H_{t-1}[m - d_m^{\text{res}}, n - d_n^{\text{res}}] \\ c_u[m - \lceil d_m^{\mathcal{B}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{B}_1(2t)} \rfloor] &\leftarrow c_u[m - \lceil d_m^{\mathcal{B}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{B}_1(2t)} \rfloor] + 1 \end{aligned} \tag{3.28}$$

$$L_t[m,n] = \sqrt{2}A_{2t}[m,n] + \frac{1}{\min\{\max\{c_u[m,n],1\},2\}\cdot\max\{c_u[m,n],1\}} Z_t[m,n] \tag{3.29}$$

where $c_p[m,n]$ is the prediction connection map, which counts the number of connections during the prediction step, and $\min\{a,b\}$ returns the smallest of a, b. The MCP and MCU steps described above follow the same principles as the motion compensated Haar temporal decomposition. For simplicity in the notation, we include both reference frames in the MCP of (3.24); in a practical implementation, similar to (3.22), equation (3.24) is adapted according to the existence of motion vectors $\left(d_m^{\mathcal{F}_1(2t)}, d_n^{\mathcal{F}_1(2t)}\right)$ and $\left(d_m^{\mathcal{B}_1(2t+2)}, d_n^{\mathcal{B}_1(2t+2)}\right)$. Similarly to the case of the motion compensated Haar temporal transform, equations (3.22)–(3.29) are applied separately to the entire frames in the order expressed above. Notice that, following the dependencies of the conventional 5/3 lifting decomposition, frames H_{t-1}, H_t and the forward and backward prediction parameters (motion vectors) derived from frame A_{2t} (which corresponds to L_t) are used during the update step. The temporal filtering of (3.22) – (3.29) is adapted according to the prediction and update connection maps $c_p[m,n]$ and $c_u[m,n]$. This is a useful feature since, in the general case of advanced prediction models, an adaptive motion compensated temporal decomposition is performed, where certain blocks are uni-directionally predicted and others use bidirectional prediction. This in turn, leads to the cases of MCTF with temporal filter-pairs 1/2, 2/2, 1/3 and 5/3. For each case, the correct scaling of $L_t[m,n]$ in (3.29) is ensured by the operation of the denominator of the multiplier of $Z_t[m,n]$, which takes the values $1,1,4,6,8,\ldots$ for $c_u[m,n] = \{0,1,2,3,4,\ldots\}$, respectively. In every case, the inverse transform can be performed by inverting the order of the equations and solving for frames $A_{2t}[m,n]$ and $A_{2t+1}[m,n]$. For example, for the motion compensated Haar temporal filtering of (3.17)–(3.21), the inverse is expressed as:

$$c_u[m,n] = 0\,,\; Z_t[m,n] = 0 \tag{3.30}$$

$$\begin{aligned} i_m^{\text{res}} &= \begin{cases} 0, \text{ if } i_m^{\mathcal{F}_1(2t)} = 0 \\ 1 - i_m^{\mathcal{F}_1(2t)}, \text{ otherwise} \end{cases}, i_n^{\text{res}} = \begin{cases} 0, \text{ if } i_n^{\mathcal{F}_1(2t)} = 0 \\ 1 - i_n^{\mathcal{F}_1(2t)}, \text{ otherwise} \end{cases}, \\ d_m^{\text{res}} &= \begin{cases} 0, \text{ if } i_m^{\mathcal{F}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases}, \qquad d_n^{\text{res}} = \begin{cases} 0, \text{ if } i_n^{\mathcal{F}_1(2t)} = 0 \\ 1, \text{ otherwise} \end{cases} \end{aligned} \tag{3.31}$$

$$\begin{aligned} Z_t[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] &\leftarrow Z_t[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] \\ &\quad + \mathcal{I}_{(i_m^{\text{res}}, i_n^{\text{res}})} H_t\,[m - d_m^{\text{res}}, n - d_n^{\text{res}}\,] \\ c_u[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] &\leftarrow c_u[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor] + 1 \end{aligned} \tag{3.32}$$

$$A_{2t}[m,n] = \tfrac{1}{\sqrt{2}}\Big(L_t[m,n] - \tfrac{1}{\max\{c_u[m,n],1\}} Z_t\,[m,n]\Big) \tag{3.33}$$

$$A_{2t+1}[m,n] = \sqrt{2}H_t[m,n] + \mathcal{I}_{(i_m^{\mathcal{F}_1(2t)}, i_n^{\mathcal{F}_1(2t)})} A_{2t}[m - \lceil d_m^{\mathcal{F}_1(2t)} \rfloor, n - \lceil d_n^{\mathcal{F}_1(2t)} \rfloor]\,. \tag{3.34}$$

One can express in a similar manner the inverse MCTF for the temporal filter described by (3.22)–(3.29).

3.4 Advanced Motion Compensated Prediction and Update

The simple MCP and MCU schemes presented before, although easily-realizable in the majority of today's platforms, have certain disadvantages. By limiting the algorithm to a fixed block size, no adaptation to the space-varying scene content is permitted. In addition, by limiting the search range to only one or two (past or future) reference frames, occlusion or aperture effects are not treated efficiently. Finally, the independent optimization of the prediction for each block may cause pixel discontinuities in the block borders, leading to blocking artifacts [51].

Extensions of the basic MCP scheme that attempt to alleviate the above problems include a family of temporal prediction schemes generally denoted as multihypothesis motion-compensated prediction (MH-MCP) [52] or motion-compensated prediction with superimposed signals [53]. This framework consists of a class of algorithms that generalize overlapped-block motion-compensated prediction (OB-MCP) [51] and multi-frame motion-compensated prediction (MF-MCP) [54]. An indicative example of MH-MCP is shown in Figure III-15. In this illustrative case, non-overlapping blocks of various sizes are predicted from one future and one past reference frame. Each prediction is a superposition of blocks from the reference frames. In the general case of MH-MCP using T reference frames that are situated at time instants $q = \{t - t^{\text{init}}, \ldots, t-1, t+1, \ldots, t + t^{\text{end}}\}$, the MH-MCP error-frame is given by:

$$\begin{aligned} H_t[m,n] = A_t[m,n] &- \sum_{q=t-t^{\text{init}}}^{t-1} \left(w_q[m,n] A_q[m - d_m^{\mathcal{F}_{t-q}(q)}, n - d_n^{\mathcal{F}_{t-q}(q)}] \right) \\ &- \sum_{q=t+1}^{t+t^{\text{end}}} \left(w_q[m,n] A_q[m - d_m^{\mathcal{B}_{q-t}(q)}, n - d_n^{\mathcal{B}_{q-t}(q)}] \right) \end{aligned} \tag{3.35}$$

where $w_q[m,n]$ are the space-varying multihypothesis weight factors of reference frame A_q [52]; $(d_m^{\mathcal{F}_{t-q}(q)}, d_n^{\mathcal{F}_{t-q}(q)})$ and $(d_m^{\mathcal{B}_{q-t}(q)}, d_n^{\mathcal{B}_{q-t}(q)})$ represent the forward and backward displacement vectors that corresponds to pixel $A_q[m,n]$, respectively. Notice that, for simplicity in the notation, the interpolation operation is not included in (3.35), i.e. we assume that for every (m,n): $(d_m^{\mathcal{F}_{t-q}(q)}, d_n^{\mathcal{F}_{t-q}(q)}) = (\lfloor d_m^{\mathcal{F}_{t-q}(q)} \rfloor, \lfloor d_n^{\mathcal{F}_{t-q}(q)} \rfloor)$. In addition, although in MH-MCP it is possible that a number of vectors for a given block are associated with the same reference frame (as shown in the example of Figure III-15), in this dissertation we always represent MH-MCP analytically with one vector per reference frame, as seen in (3.35). However, in our related experiments, all the possible combinations of MH-MCP are permitted.

For block-based MH-MCP, $w_q[m,n]$ is constant over a certain group of pixels (m,n) corresponding to a block of the q-th reference frame. A video encoder employing MH-MCP based on (3.35) has to estimate the optimal $w_q[m,n]$ and $(d_m^{\mathcal{F}_{t-q}(q)}, d_n^{\mathcal{F}_{t-q}(q)})$ or $(d_m^{\mathcal{B}_{q-t}(q)}, d_n^{\mathcal{B}_{q-t}(q)})$ (with the corresponding block-segmentation) of the T reference frames in order to minimize the variance of $H_t[m,n]$. This is performed by multihypothesis motion estimation (MH-ME) schemes. In practice, all the proposed algorithms limit complexity by presetting certain acceptable block sizes and small values for T. For example, in the H.264 standard [55], allowable block sizes are limited to $B_m, B_n = \{4, 8, 16\}$ (with all

the possible combinations permitted) while the maximum practical setting used in MPEG exploration experiments is $T = 5$ [56].

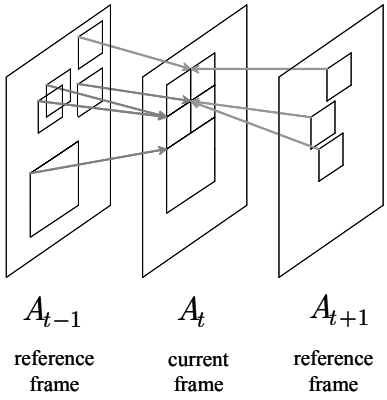


Figure III-15. An example of multihypothesis motion-compensated prediction.

A practical algorithm for MH-ME is proposed in Chapter 4. This algorithm also combines the MH-MCP with multihypothesis motion compensated update (MH-MCU), by adaptively performing the update step following the prediction and update connection mapping, as shown in subsection 3.3. In general, in order to retain an orthonormal temporal decomposition, one must perform MH-MCP and MH-MCU via a series of lifting steps that normalize the magnitude of each pixel of the low-frequency frame $L_t[m,n]$ according to the update connection map $c_u[m,n]$ [45]. As a result, for advanced MCTF we can assume that a total of Λ pairs of MH-MCP and MH-MCU steps take place. Similar to conventional lifting [47], the first lifting step is the trivial polyphase separation, albeit in the temporal direction, i.e.:

$$L_t^0[m,n] = A_{2t}[m,n] \tag{3.36}$$

$$H_t^0[m,n] = A_{2t+1}[m,n]\,. \tag{3.37}$$

For each subsequent pair of MH-MCP and MH-MCU steps λ, with $1 \le \lambda \le \Lambda$, the following procedure is performed.

Firstly, the MH-MCP that corresponds to step λ utilizes frames $H_q^{\lambda-1}$ with lifting coefficients α_q and frame $A_t^{\lambda-1}$; the set of permissible values for q depends on the specific lifting dependencies; without loss of generality, we assume that q is bounded by $t_p^{\text{init}}(\lambda)$ and $t_p^{\text{end}}(\lambda)$ around time instant t (and does not include t)[1]. For this case, MH-MCP can be expressed as:

[1] In general, if some frames at time instants $u(\lambda)$ (within the time instants $t_p^{\text{init}}(\lambda)$ and $t_p^{\text{end}}(\lambda)$) are not used, then $a_{u(\lambda)} = 0$.

$$H_t^{\lambda}[m,n] = L_t^{\lambda-1}[m,n] - \sum_{q=t-t_p^{\text{init}}(\lambda)}^{t-1} \left(w_q[m,n] \cdot \alpha_q \cdot H_q^{\lambda-1}[m - d_m^{\mathcal{F}_{t-q}(q)}, n - d_n^{\mathcal{F}_{t-q}(q)}] \right) - \sum_{q=t+1}^{t+t_p^{\text{end}}(\lambda)} \left(w_q[m,n] \cdot \alpha_q \cdot H_q^{\lambda-1}[m - d_m^{\mathcal{B}_{q-t}(q)}, n - d_n^{\mathcal{B}_{q-t}(q)}] \right) \quad . \tag{3.38}$$

Note that the parameters $t_p^{\text{init}}(\lambda)$, $t_p^{\text{end}}(\lambda)$, and $\alpha_{t-t_p^{\text{init}}(\lambda)},\ldots,\alpha_{-1},\alpha_1,\ldots,\alpha_{t+t_p^{\text{end}}(\lambda)}$, are directly taken from the lifting factorization of the chosen filter-bank [47], while $w_q[m,n]$ and $(d_m^{\mathcal{F}_{t-q}(q)}, d_n^{\mathcal{F}_{t-q}(q)})$ or $(d_m^{\mathcal{B}_{q-t}(q)}, d_n^{\mathcal{B}_{q-t}(q)})$ are produced by the utilized multihypothesis motion estimation algorithm. Moreover, similarly as before, interpolation is not considered in order to simplify the formulations.

For the corresponding MH-MCU, similarly as before, first the update connection map and the frame containing the motion inversion information (Z_t) are initialized by:

$$c_u[m,n] = 0\,,\ Z_t[m,n] = 0 \tag{3.39}$$

Subsequently, MH-MCU utilizes frames H_q^{λ} with lifting coefficients β_q and frame $L_t^{\lambda-1}$. Similarly as before, the set of permissible values for q depends on the specific lifting dependencies; without loss of generality, we assume that q is bounded by $t_u^{\text{init}}(\lambda)$ and $t_u^{\text{end}}(\lambda)$ around time instant t (and does not include t). For this case, MH-MCU can be performed by a series of steps that invert the motion information generated during MH-MCP of stage λ. Specifically, first the inversion of the forward prediction is performed, where for each $q = \{t - t_u^{\text{init}}(\lambda),\ldots,t-1\}$ we have:

$$\begin{aligned} Z_t[m - d_m^{\mathcal{F}_{t-q}(q)}, n - d_n^{\mathcal{F}_{t-q}(q)}] &\leftarrow Z_t[m - d_m^{\mathcal{F}_{t-q}(q)}, n - d_n^{\mathcal{F}_{t-q}(q)}] + w_q[m,n] \cdot \beta_q \cdot H_q^{\lambda}[m,n] \\ c_u[m - d_m^{\mathcal{F}_{t-q}(q)}, n - d_n^{\mathcal{F}_{t-q}(q)}] &\leftarrow c_u[m - d_m^{\mathcal{F}_{t-q}(q)}, n - d_n^{\mathcal{F}_{t-q}(q)}] + 1 \end{aligned} \tag{3.40}$$

$$L_t^{\lambda}[m,n] = \left[L_t^{\lambda-1}[m,n] + \frac{1}{\max\{c_u[m,n], t^{\mathcal{F}}(\lambda)\}} Z_t[m,n] \right]. \tag{3.41}$$

Then, after another initialization performed by equation (3.39), the inversion of the backward prediction is performed, where for each $q = \{t+1,\ldots,t+t_u^{\text{end}}(\lambda)\}$ we have:

$$\begin{aligned} Z_t[m - d_m^{\mathcal{B}_{q-t}(q)}, n - d_n^{\mathcal{B}_{q-t}(q)}] &\leftarrow Z_t[m - d_m^{\mathcal{B}_{q-t}(q)}, n - d_n^{\mathcal{B}_{q-t}(q)}] + w_q[m,n] \cdot \beta_q \cdot H_q^{\lambda}[m,n] \\ c_u[m - d_m^{\mathcal{B}_{q-t}(q)}, n - d_n^{\mathcal{B}_{q-t}(q)}] &\leftarrow c_u[m - d_m^{\mathcal{B}_{q-t}(q)}, n - d_n^{\mathcal{B}_{q-t}(q)}] + 1 \end{aligned} \tag{3.42}$$

$$L_t^{\lambda}[m,n] \leftarrow \left[L_t^{\lambda}[m,n] + \frac{1}{\max\{c_u[m,n], t^{\mathcal{B}}(\lambda)\}} Z_t[m,n] \right]. \tag{3.43}$$

Similarly as before, the parameters $t_u^{\text{init}}(\lambda)$, $t_u^{\text{end}}(\lambda)$ and $\beta_{t-t_u^{\text{init}}(\lambda)},\ldots,\beta_{-1},\beta_1,\ldots,\beta_{t+t_u^{\text{end}}(\lambda)}$, are directly taken from the lifting factorization of the chosen filter-bank [47]. Moreover, the scaling factors $t^{\mathcal{F}}(\lambda)$, $t^{\mathcal{B}}(\lambda)$ used in (3.41), (3.43) are derived from the scaling factors specified in the original lifting factorization [47]. Nevertheless, in this way the $t^{\mathcal{F}}(\lambda)$, $t^{\mathcal{B}}(\lambda)$ factors are not necessarily optimal for adaptive motion-compensated lifting decompositions such as the ones presented before. The reader is referred to [57] [58] for additional information on this topic.

3.5 Temporal Prediction Structures for Video Coding

In this section, we review the conventional closed-loop video coding structure as well as the new open-loop video coding schemes that perform a temporal decomposition using motion compensated temporal filtering. Both have been used in this dissertation to provide working video coding systems with scalability properties.

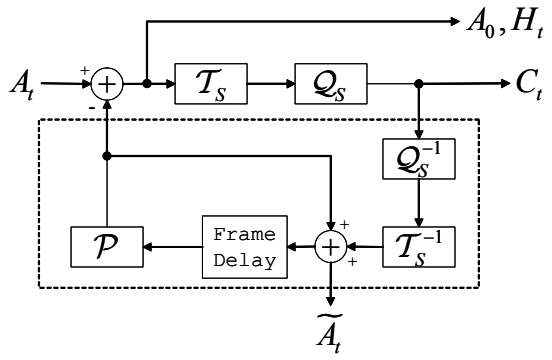
All the currently-standardized video coding schemes are based on a structure in which the two-dimensional spatial transform and quantization is applied to the error frame coming from closed-loop temporal prediction. A simple structure describing such architectures is shown in Figure III-16(a). The operation of temporal prediction $\mathcal{P}$ typically involves block-based motion-compensated prediction. The decoder receives the motion vector information and the compressed error-frame C_t and performs the identical loop using this information in order to replicate MCP within the $\mathcal{P}$ operator. Hence, in the decoding process (seen in the dashed area in Figure III-16(a)), the reconstructed frame at time instant t can be written as:

$$\widetilde{A}_t = \mathcal{P}\widetilde{A}_{t-1} + \mathcal{T}_S^{-1}\mathcal{Q}_S^{-1}C_t,\ \widetilde{A}_0 = \mathcal{T}_S^{-1}\mathcal{Q}_S^{-1}C_0. \tag{3.44}$$

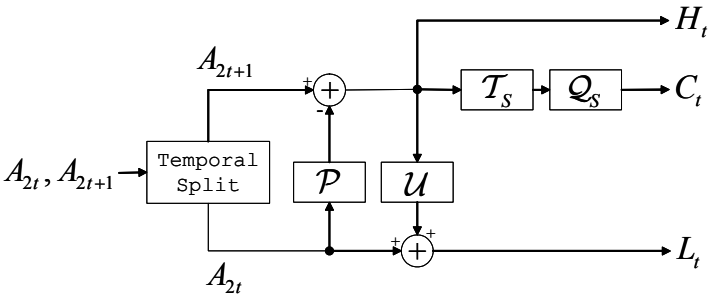
The recursive operation given by (3.44) creates the well-known drift effect between the encoder and decoder if the information used is different between the two sides, i.e. if $C_t \neq \mathcal{Q}_S\mathcal{T}_S H_t$ at any time instant t in the decoder. This is not uncommon in practical systems, since transmission errors or loss of compressed data due to limited channel capacity can be a dominant scenario in wireless or IP-based networks, where a number of clients compete for the available network resources. In general, the capability to seamlessly adapt the compression bitrate without transcoding, i.e. SNR scalability, is a very useful feature for such network environments. Solutions for SNR scalability based on the coding structure of Figure III-16(a) basically try to remove the prediction drift by artificially reducing at the encoder side the bitrate of the compressed information C_t to a base layer for which the network can guarantee the correct transmission [3]. An example of such a codec is the MPEG-4 FGS [4].

This however reduces the prediction efficiency [3], thereby leading to degraded coding efficiency for SNR scalability. To overcome this drawback, techniques that include a certain amount of enhancement layer information into the prediction loop have been proposed. For example, leaky prediction [59] gracefully decays the enhancement information introduced in the prediction loop in order to limit the error propagation and accumulation. Scalable coding schemes employing this technique achieve notable coding gains over the standard MPEG-4 FGS [4] and a good trade-off between low drift errors and high coding efficiency [59] [60]. Progressive Fine Granularity Scalable (PFGS) coding [61] yields also significant improvements over MPEG-4 FGS by introducing two prediction loops with different quality references. A generic PFGS coding framework employing multiple prediction loops with different quality references and careful drift control leads to considerable coding gains over MPEG-4 FGS, as reported in [62] [63].

To address this issue, several proposals suggested an open-loop system, depicted in Figure III-16(b), which incorporates recursive temporal filtering. This can be perceived as a temporal wavelet transform with motion compensation [8], i.e. motion-compensated temporal filtering.



(a)



(b)

Figure III-16. (a): The hybrid video compression scheme. (b): Motion-compensated temporal filtering. Notations: A_τ represents the input video frame at time instant $\tau = 0, t, 2t, 2t+1$; $\widetilde{A}_t$ is the reconstructed frame; H_t is the error frame, whereas L_t is the updated frame; C_t denotes the transformed and quantized error frame obtained by using the spatial operators $\mathcal{T}_S$ and $\mathcal{Q}_S$, respectively; $\mathcal{P}$ denotes temporal prediction, while $\mathcal{U}$ denotes the temporal update.

As described in the previous sections, this scheme begins with a separation of the input into even and odd temporal frames (temporal split). Then the temporal predictor performs MCP to match the information of frame A_{2t+1} with the information present in frame A_{2t}. Subsequently, the MCU operator $\mathcal{U}$ inverts the information of the prediction error back to frame A_{2t}, thereby producing, for each pair of input frames, an error frame H_t and an updated frame L_t. The MCU operator performs either motion compensation using the inverse vector set produced by the predictor [11] (as described previously in this chapter), or generates a new vector set by backward motion estimation [12]. The process iterates on the L_t frames, which are now at half temporal-sampling rate (following the multilevel operation of the conventional lifting), thereby forming a hierarchy of temporal levels for the input video. The decoder performs the mirror operation: the scheme of Figure III-16(b) operates from right to left, the signs of the $\mathcal{P}$, $\mathcal{U}$ operators are inverted and a temporal merging occurs at the end to join the reconstructed frames. As a result, having performed the reconstruction of the L_t, denoted by $\widetilde{L}_t$, at the decoder we have:

$$\widetilde{A}_{2t} = \widetilde{L}_t - \mathcal{U}\mathcal{T}_S^{-1}\mathcal{Q}_S^{-1}C_t,\ \widetilde{A}_{2t+1} = \mathcal{P}\widetilde{A}_{2t} + \mathcal{T}_S^{-1}\mathcal{Q}_S^{-1}C_t \qquad (3.45)$$

where $\widetilde{A}_{2t}$, $\widetilde{A}_{2t+1}$ denote the reconstructed frames at time instants $2t$, $2t+1$. As seen from (3.45), even if $C_t \neq \mathcal{Q}_S\mathcal{T}_S H_t$ in the decoder, the error affects locally the reconstructed frames $\widetilde{A}_{2t}$, $\widetilde{A}_{2t+1}$ and does not propagate linearly in time over the reconstructed video. Error-propagation may occur

only across the temporal levels through the reconstructed $\tilde{L}_t$ frames. However, after the generation of the temporal decomposition, embedded coding may be applied in each GOP by prioritizing the information of the higher temporal levels based on a dyadic-scaling framework, i.e. following the same principle of prioritization of information used in wavelet-based SNR-scalable image coding [24]. Hence, the effect of error propagation in the temporal pyramid is limited and seamless video-quality adaptation can be obtained in SNR scalability [44] [45]. In fact, experimental results obtained with the SNR-scalable MCTF video coders proposed in this dissertation, as well as the results obtained with other state-of-the-art algorithms [18] [10], suggest that this coding architecture can be comparable in rate-distortion sense to an equivalent non-scalable coder that uses the closed-loop structure.

This chapter is concluded with the presentation of two indicative coding systems that represent the current state-of-the-art in the closed-loop and open-loop temporal prediction structures, namely the Advanced Video Coder (AVC), also called as the H.264 coder, which was jointly standardized by MPEG and ITU-T [55], and the motion-compensated embedded zero-block coder (MC-EZBC) of [18]. While the AVC is a non-scalable coding scheme which is optimized for a certain set of quantization parameters, the MC-EZBC has the capability of scalability in bitrate, resolution and SNR.

3.5.1 Closed-loop temporal prediction – the Advanced Video Coder

The ISO MPEG-4 Advanced Video Coder (AVC) standard, or (equivalently) the ITU-T VCEG H.264 standard, is a recently completed video compression standard developed by the Joint Video Team (JVT) group, which brought together video-compression experts from the MPEG-4 video group and VCEG. Within a certain visual quality range for the compressed video, the new standard promises much higher compression than that achievable with earlier MPEG or ITU-T standards. Although it does not provide a scalable bitstream, the algorithm used in MPEG-4 AVC supports flexibilities in coding as well as organization of coded data that can increase resilience to errors or losses. The increase in coding efficiency comes at the expense of an increase in complexity with respect to earlier standards. Our presentation of the basic features of AVC [55] is based on the several comprehensive tutorials found in literature [64] [65] [66]. Several special issues in journals are also dedicated to this standard, e.g. see [67].

Similar to earlier MPEG and VCEG standards, the coding structure of AVC is based on closed-loop temporal prediction using motion estimation and compensation. The coding of video is performed frame-by-frame. Within each frame, the image data is partitioned in non-overlapping square areas (macroblocks – MBs) of 16×16 pixels in the luminance channel and 8×8 pixels in the chrominance channels. A pictorial representation of the compression/decompression architecture for a macroblock is given in Figure III-17 [65]. Currently, only 4:2:0 chroma format and 8-bit sample precision for luminance and chrominance pixel values is supported in the standard, but extensions to other source formats will probably be provided in the future. The macroblocks of each frame are organized in larger structures called slices, which can be areas of one frame, or an entire frame. The slices can be of type I, P and B, indicating, respectively, intra-coding, predictive coding and coding based on the B-frame concept that is found in earlier MPEG standards. However, although the naming conventions in AVC follow the traditional naming organization of hybrid video coders, the slice types include a

rich set of coding features; for example, P slices can also include intra-coded MBs and, although MCP performed in a P slice is unidirectional, frames from the past or the future can be used as references. Moreover, unlike in previous standards, B slices can use (simultaneously) prediction from future and past reference frames and, optionally, can also serve as reference slices.

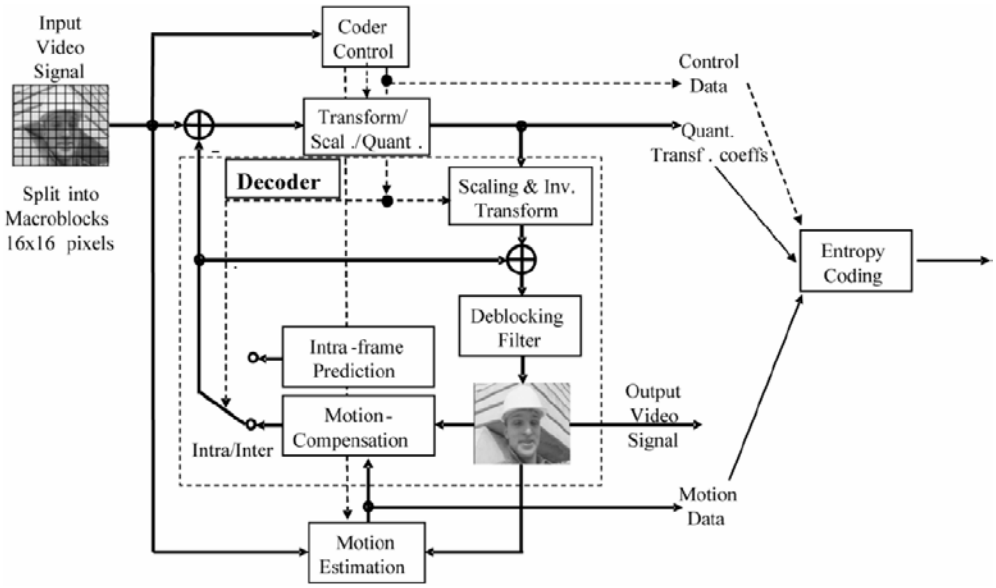


Figure III-17. Basic coding structure of AVC for a macroblock. From [65].

The coding tools of AVC are organized in profiles and levels. This is a structure inherited from previous standards and its aim is to help developers of the standard select an appropriate subset of tools (and parameters) that suit their application needs. Three profiles are defined: Baseline, Main and Extended. The main profile is generally considered to be the basic profile for the majority of applications since it provides the highest coding efficiency. Hence, its tools will be briefly outlined here. In terms of temporal prediction, this profile includes all slice types, i.e. I, P and B. Moreover, the temporal prediction is performed with variable block sizes ranging (dyadically) from 16×16 pixels down to 4×4 pixels; rectangular block sizes are permitted. Multiple reference frames can be used for MCP, selected from the past or the future. The prediction can be performed with varying weights in order to accommodate scene changes, fading, etc. Intra coding is also available within this profile; its features are a great improvement over previous standards, as it essentially performs directional spatial prediction by extrapolating the edges of previously-decoded parts of the current picture.

In terms of global prediction tools available to all profiles, AVC includes a deblocking mechanism, which is an adaptive technique for smoothing the predicted frames after MCP. This is essentially performed by applying an adaptive lowpass filter across the MCP block borders. The filter is placed within the MCP loop, as seen in Figure III-17. It is essentially an improved descendant of the deblocking mechanism found in the annex of H.263+ and its purpose is to improve both subjective and objective quality in areas where the motion model fails to provide an efficient prediction.

Finally, for the actual frame compression, a block-based DCT-alike transform is used, which is (primarily) a 4×4 matrix. The small transform block size is justified by the advanced prediction tools that decorrelate the input video well. Moreover, a hierarchical block transform is also specified where the transform block size can be extended to 16×16 pixels for the luminance channel and 8×8 for the chrominance channels. The transform coefficients are obtained with an approximation of the DCT, which was found to be as efficient as the fixed-point DCT used in prior standards. The AVC transform is using 16-bit precision, thereby enabling the use of 16-bit fixed-point arithmetic precision for the entire video coding system, and exact-match inverse transform. The last feature solves mismatches found in prior MPEG standards between different implementations of encoders and decoders. The actual coding of the transform coefficients is based on context-adaptive binary arithmetic coding (CABAC), which is a context-based arithmetic coding technique employing a large number of context-modes in order to enable the generation of small-alphabet symbol streams to the arithmetic coder. A simpler method for entropy coding based on context-adaptive variable-length coding (CAVLC) is also supported in order to enable implementations with a higher degree of parallelism. These coding engines are used for the coding of the motion-vector data as well.

3.5.2 Open-loop temporal prediction – Bidirectional MC-EZBC with Lifting Implementation

The recent research advances in open-loop temporal prediction led to a number of video coding schemes for scalable video coding with a temporal decomposition structure based on the wavelet transform. This is currently an on-going research direction with a large number of contributions from various groups. The interested reader can refer to special issues found in journals (e.g. [10]). Of relative interest are also the outcome of the recent call for proposals of MPEG [68], and overview papers [9].

In this subsection, we focus on a particular instantiation of a related scalable video coding system published recently [18]. It was termed MC-EZBC [18], which stands for "Motion-compensated embedded zero-block coding". Our choice of this system is motivated by the fact that its authors provided a related software implementation to the MPEG community during the early stages of the MPEG exploration activity on scalable video coding [13], thereby allowing for an open examination of their technology by video experts. Moreover, MC-EZBC incorporates all the basic tools found in state-of-the-art open-loop video coding systems. For example, the prediction is bi-directional (using one past and one future frame); variable block sizes are used for block-based MCP, which range (dyadically) from 64×64 pixels down to 4×4 pixels. In addition, recent instantiations of the algorithm also include selective intra-prediction following the AVC principles [69].

The basic coding structure of MC-EZBC is illustrated in Figure III-18. The decoder operates in the same way, following the procedure from right to left and performing inverse MCTF to reconstruct the output video frames. The open-loop coding mechanism is performed in a number of steps. First the input is read in groups of frames; typically 16 frames are processed together. The MCTF process of Figure III-18 is then applied to the input frames. Based on the motion field, if, during the temporal update step, the percentage of unconnected pixels of every two consecutive frames is smaller than a threshold, one-stage temporal decomposition using the motion-compensated Haar filtering is applied.

Otherwise, MCTF stops at the current temporal level. This procedure is recursively performed on the generated low temporal subband, until there is only one low temporal subband frame left. If the percentage of unconnected pixels never exceeds the threshold during this process, a motion-compensated temporal decomposition of the input group of frames in four levels is generated. Thus, the number of temporal decomposition levels is controlled by the percentage of unconnected pixels in the temporal pyramid. This process is followed by a four-level spatial transform applied to each frame in the temporal pyramid. Typically, the 9/7 filter-bank is used for this purpose. The compression of the produced frames is performed with the EZBC algorithm [32], which is a block-based embedded intra-subband coding algorithm utilizing context-based adaptive arithmetic coding. All spatio-temporal subbands are coded separately, in order to enable temporal and resolution scalability. The motion-vector information is encoded with a lossless DPCM and adaptive arithmetic coding scheme. Finally, the packetizer seen in Figure III-18, represents the post-compression optimization and bitstream extraction process; based on the accumulated rate-distortion information, a PCRDO scheme selects the appropriate number of bitplanes for each subband of each frame in order to satisfy a target bitrate for the compressed sequence. Then the appropriate sub-bitstreams are extracted and packed into a new stream to be used for decoding. In the actual implementation of the codec, the PCRDO process is a separate module at the encoder that produces admissible rate-distortion points, which are packed with the entire compressed bitstream. Subsequently, any target bitrate is satisfied by creating a new file using the bitstream extractor, whose size is controlled based on the process outlined in subsection 3.1.4. That is, PCRDO and bitstream extraction are performed by two distinct modules in this codec.

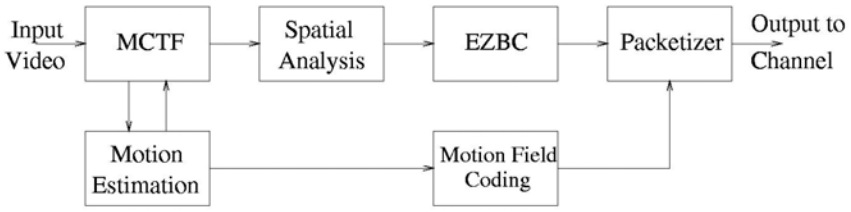


Figure III-18. Basic structure of MC-EZBC. From [70].

3.6 References

[1] M. Boliek, C. Christopoulos, and E. Majani, "JPEG2000 Part I Final Draft International Standard," ISO/IEC JTC1/SC29/WG1, Report September 25, 2000 2000.

[2] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 Still Image Coding System: An Overview," *IEEE Transactions on Consumer Electronics*, vol. 46, pp. 1103-1127, 2000.

[3] H. M. Radha, M. v. d. Schaar, and Y. Chen, "The MPEG-4 Fine-grained Scalable Video Coding for Multimedia Streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, pp. 53-68, 2001.

[4] W. Li, "Streaming Video Profile in MPEG-4," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 301-317, 2001.

[5] C. Brislawn and P. Schelkens, "JPEG 2000 Part 12: Extensions for Three-Dimensional and Floating Point Data Scope and Requirements document, draft version 1," ISO/IEC JTC1/SC29/WG1, Sydney, Australia, Report WG1N2378, November 12-16, 2001 2001.

[6] P. Schelkens, A. Munteanu, J. Barbarien, M. Galca, X. Giro i Nieto, and J. Cornelis, "Wavelet Coding of Volumetric Medical Datasets," *IEEE Transactions on Medical Imaging, Special issue on "Wavelets in Medical Imaging,"* vol. 22, pp. 441-458, 2003.

[7] T. Kronander, "Motion Compensated 3-dimensional Wave-form Image Coding," presented at IEEE International Conference on Accoustics, Speech and Signal Processing (ICASSP), 1989.

[8] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*, vol. 3, pp. 559-571, 1994.

[9] J.-R. Ohm, "Advances in scalable video coding," *Proceedings of the IEEE*, vol. 93, pp. 42-56, 2005.

[10] J. W. Woods and J.-R. Ohm, "Special issue on subband/wavelet interframe video coding," *Signal Processing: Image Communication*, vol. 19, 2004.

[11] B. Pesquet-Popescu and V. Bottreau, "Three Dimensional Lifting Schemes for Motion Compensated Video Compression," presented at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, Utah, USA, 2001.

[12] A. Secker and D. Taubman, "Motion-Compensated Highly Scalable Video Compression using Adaptive 3D Wavelet Transform Based on Lifting," presented at IEEE International Conference on Image Processing (ICIP), Thessaloniki, Greece, 2001.

[13] J.-R. Ohm and T. Ebrahimi, "Report of Ad-hoc Group on Exploration of Interframe Wavelet Technology in Video," ISO/IEC JTC1/SC29/WG11 m8295, March 2002.

[14] J.-R. Ohm and M. v. d. Schaar, "Call for proposals on scalable video coding technology," ISO/IEC JTC1/SC29/WG11 (MPEG), n5958, Oct. 2003.

[15] M. v. d. Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Transactions on Multimedia*, to appear.

[16] D. Mukerjee, E. Delfosse, J.-G. Kim, and Y. Wang, "Terminal and network quality of service," *IEEE Transactions on Multimedia*, vol. to appear.

[17] D. Taubman and M. W. Marcelin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*. Norwell, Massachusetts: Kluwer Academic Publishers, 2002.

[18] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with Lifting Implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 1183-1194, 2004.

[19] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. v. d. Schaar, J. Cornelis, and P. Schelkens, "In-band Motion Compensated Temporal Filtering," *Signal Processing: Image Communication*, vol. 19, pp. 653-673, 2004.

[20] C. Chrysafis and A. Ortega, "Line-Based, Reduced Memory, Wavelet Image Compression," *IEEE Transactions on Image Processing*, vol. 9, pp. 378-389, 2000.

[21] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, 1989.

[22] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, pp. 2325-2383, 1998.

[23] A. Munteanu, "Wavelet image coding and multiscale edge detection:Algorithms and applications," in *Dept. of Electronics and Information Processing (ETRO)*. Doctorate Thesis, Brussels: Vrije Universiteit Brussel, 2003, pp. 285.

[24] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445-3462, 1993.

[25] A. Said and W. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243-250, 1996.

[26] R. R. Shively, E. Ammicht, and P. D. Davis, "Generalizing SPIHT: A Family of Efficient Image Compression Algorithms," presented at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Istanbul, Turkey, 2000.

[27] A. Munteanu, J. Cornelis, G. Van der Auwera, and P. Cristea, "Wavelet-based lossless compression scheme with progressive transmission capability," *International Journal of Imaging Systems and Technology, Special Issue on Image and Video Coding, J. Robinson and R. D. Dony, Eds.*, vol. 10, pp. 76-85, 1999.

[28] A. Munteanu, J. Cornelis, and P. Cristea, "Wavelet-Based Lossless Compression of Coronary Angiographic Images," *IEEE Transactions on Medical Imaging*, vol. 18, pp. 272-281, 1999.

[29] A. Munteanu, J. Cornelis, G. Van der Auwera, and P. Cristea, "Wavelet Image Compression - The Quadtree Coding Approach," *IEEE Transactions on Information Technology in Biomedicine*, vol. 3, pp. 176-185, 1999.

[30] D. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, pp. 1158-1170, 2000.

[31] G. M. Morton, "A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing," IBM Ltd, Ottawa, Canada 1966 1966.

[32] S.-T. Hsiang and J. W. Woods, "Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling," presented at IEEE International Symposium on Circuits and Systems (ISCAS), Geneva, Switzerland, 2000.

[33] Y. Andreopoulos, P. Schelkens, N. D. Zervas, T. Stouraitis, C. E. Goutis, and J. Cornelis, "A Wavelet-Tree Image Coding System with Efficient Memory Utilization," presented at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, Utah, USA, 2001.

[34] D. Taubman, "Embedded, independent block-based coding of subband data," ISO/IEC JTC1/SC29/WG1, N871R, July 1998 1998.

[35] D. Taubman, "EBCOT: Embedded, block coding with optimized truncation," ISO/IEC JTC1/SC29/WG1, N1020R, October 1998 1998.

[36] M.-T. Sun and A. R. Reibman, *Compressed Video over Networks*, vol. 5. New York - Basel: Marcel Dekker, Inc., 2001.

[37] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards - Algorithms and Architectures*. Massachusetts (USA): Kluwer Academic Publishers, 1995.

[38] F. Dufeaux and F. Moscheni, "Motion Estimation Techniques for Digital TV: A Review and a New Contribution," *Proceedings of the IEEE*, vol. 83, pp. 858-876, 1995.

[39] J. Magarey and N. G. Kingsbury, "Motion estimation using complex wavelets," presented at SPIE Wavelet Applications in Signal and Image Processing IV, 1996.

[40] H.-W. Park and H.-S. Kim, "Motion Estimation Using Low-Band-Shift Method for Wavelet-Based Moving-Picture Coding," *IEEE Transactions on Image Processing*, vol. 9, pp. 577-587, 2000.

[41] A. Smolic, T. Sikora, and J.-R. Ohm, "Long-Term Global Motion Estimation and Its Application for Sprite Coding, Content Description, and Segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 1227-1242, 1999.

[42] B. Girod, "Motion-Compensating Prediction with Fractional-Pel Accuracy," *IEEE Transactions on Communications*, vol. 41, pp. 604-612, 1993.

[43] L. Chiariglione, "MPEG and Multimedia Communications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 5-18, 1997.

[44] S.-J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Transactions on Image Processing*, vol. 8, pp. 155-167, 1999.

[45] A. Secker and D. Taubman, "Lifting-Based Invertible Motion Adaptive Transform (LIMAT) Framework for Highly Scalable Video Compression," *IEEE Transactions Image Processing*, vol. 12, pp. 1530-1542, 2003.

[46] W. Sweldens, "The Lifting Scheme: a Custom Design Construction of Biorthogonal Wavelets," *Journal of Appl. and Comput. Harmonic Analysis*, vol. 3, pp. 186-200, 1996.

[47] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, pp. 247-269, 1998.

[48] D. Taubman and A. Zakhor, "Multirate 3-D Subband Coding of Video," *IEEE Transactions on Image Processing*, vol. 3, pp. 572-588, 1994.

[49] D. S. Turaga, M. v. d. Schaar, Y. Andreopoulos, A. Munteanu, and P. Schelkens, "Unconstrained Motion Compensated Temporal Filtering (UMCTF) for Efficient and Flexible Interframe Wavelet Video Coding," *Signal Processing: Image Communication*, to appear.

[50] D. S. Turaga and M. v. d. Schaar, "Unconstrained Motion Compensated Temporal Filtering," ISO/IEC JTC1/SC29/WG11 m8388, 2002.

[51] M. T. Orchard and G. J. Sullivan, "Overlapped Block Motion Compensation: An Estimation-Theoretic Approach," *IEEE Transactions Image Processing*, vol. 3, pp. 693-699, 1994.

[52] M. Flierl, T. Wiegand, and B. Girod, "Rate-Constrained Multihypothesis Prediction for Motion-Compensated Video Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 957-969, 2002.

[53] M. Flierl, "Doctorate Thesis: Video Coding with Superimposed Motion-Compensated Signals." Erlangen: University of Erlangen-Nuremberg, 2003.

[54] T. Wiegand, X. Zhang, and B. Girod, "Long-Term Memory Motion-Compensated Prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 70-84, 1999.

[55] T. Wiegand and G. Sullivan, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification," ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6 2003.

[56] M. v. d. Schaar and J.-R. Ohm, "Draft testing procedures for evidence on scalable video coding technology," ISO/IEC JTC1/SC29/WG11 (MPEG), N5167, Oct. 2002.

[57] B. Girod and S. Han, "Optimum motion-compensated lifting," *IEEE Signal Processing Letters*, to appear.

[58] C. Tillier, B. Pesquet-Popescu, and M. v. d. Schaar, "Improved update operators for lifting-based motion-compensated temporal filtering," *IEEE Signal Processing Letters*, to appear.

[59] S. Han and B. Girod, "SNR Scalable Coding with Leaky Prediction," ITU-T Q.6/SG16, VCEG-N53 2001.

[60] H. C. Huang, C.-N. Wang, and T. Chiang, "A Robust Fine Granularity Scalability Using Trellis Based Predictive Leak," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 372-385, 2002.

[61] F. Wu, S. Li, and Y.-Q. Zhang, "A Framework for Efficient Progressive Fine Granularity Scalable Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 332-344, 2001.

[62] Y. He, R. Yan, F. Wu, and S. Li, "H.26L-based fine granularity scalable video coding," ISO/IEC JTC1/SC29/WG1, M7788, December 2001 2001.

[63] F. Wu, S. Li, R. Yan, X. Sun, and Y.-Q. Zhang, "Efficient and Universal Scalable Video Coding," presented at IEEE International Conference on Image Processing (ICIP), Rochester, NY, USA, 2002.

[64] G. Sullivan and T. Wiegand, "Video compression - from Concepts to the H.264/AVC standard," *Proceedings of the IEEE*, vol. 93, pp. 18-31, 2005.

[65] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 560-576, 2003.

[66] A. Puri, X. Chen, and A. Luthra, "Video coding using the H.264/MPEG-4 AVC compression standard," *Signal Processing: Image Communication*, 2004.

[67] A. Luthra, G. Sullivan, and T. Wiegand, "Introduction to the special issue on the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, 2003.

[68] J.-R. Ohm, "Registered responses to the call for proposals on scalable video coding," ISO/IEC JTC1/SC29/WG11 (MPEG), m10569, March 2004.

[69] Y. Wu and J. W. Woods, "Recent improvements in the MC-EZBC video coder," ISO/IEC JTC1/SC29/WG11 (MPEG), m10396, Dec. 2003.

[70] J. W. Woods and P. Chen, "Improved MC-EZBC with quarter-pixel motion vectors," ISO/IEC JTC1/SC29/WG11 (MPEG), m8366, May 2002.

Chapter IV

IN-BAND ARCHITECTURES

ALTHOUGH there have been many proposals for wavelet-based motion-compensated video coding, it appears that the majority of research in the field has focused on applying the discrete wavelet transform within the classical closed-loop video-coding structure as a substitute to the discrete cosine transform [1] [2]. This has the potential of enabling bitrate scalability within closed-loop video compression with the use of embedded coding of wavelet coefficients [3]. However, many believed that it does not constitute a particularly-strong advantage for the video-compression industry to adopt wavelet-based systems, since bitrate scalability with similar performance can be achieved with embedded coding of DCT coefficients as well [4] [5].

Newer efforts have focused on applying wavelet decompositions in the temporal direction [6] [7] [8], as presented in the previous chapter. This stimulated a larger interest in the research and industrial community due to the fact that an open-loop coding architecture is provided, which can efficiently handle a broad range of bitrate scalability with seemingly no loss over the equivalent closed-loop system. Nonetheless, apart from the modifications in the temporal prediction structure, the spatial wavelet decomposition is still applied to the residue frames produced by MCP in the spatial domain. In this way, if resolution scalability is not a desired functionality, the use of wavelet transforms in the spatial decomposition can be completely avoided if an embedded DCT-based compression algorithm is used [4]. Consequently, from a functionality point-of-view, we reach the conclusion that the significant advantage offered by the DWT versus other spatial transforms comes from its multiresolution nature, which provides dyadically-reduced representations of the input video with critical sampling. This hints the potential that, similar to the image coding case [1], the DWT can be the unique solution for resolution scalability in video coding, without incurring a penalty to the full-resolution compression performance. We note here that, as shown in the seminal paper of Mallat [9], the DWT provides the optimal multiresolution basis among all representations with critical sampling.

On the other hand, the performance of MCP in the spatial domain (full resolution) can create problems for resolution scalability in video coding systems, as demonstrated by numerous contributions [10, 11] [12] [13] [14, 15] [16]. This can be mainly attributed to the fact that, in

conventional systems performing MCP in the spatial domain, the motion vectors are downsampled for low-resolution decoding, thereby causing a mismatch between the motion compensation performed at the encoder and decoder side [10]. This can be alleviated if motion compensation is performed in a multiresolution manner [17, 18], i.e. directly in the subbands of the critically-sampled DWT decomposition (in-band). However, due to downsampling operations involved in the critically-sampled DWT, this process tends to undermine coding efficiency [19]. As a result, it has not been extensively investigated in the relevant literature.

In this chapter, we propose several video coding systems and architectures for this alternative in-band approach. From the system perspective, these coding approaches perform the temporal prediction (and potentially temporal update) *after* the wavelet-based spatial decomposition. This leads to a class of video coding systems that utilize in-band motion-compensated prediction and in-band motion-compensated update steps. By emphasizing the order of operations, one calls the open-loop schemes that perform a wavelet-based temporal decomposition prior to the spatial DWT as "*t*+2D" systems. Consequently, the in-band schemes proposed in this chapter are "2D+*t*" systems [20]. It is shown in the various sections of this chapter that the "2D+*t*" systems can indeed provide better resolution scalability and, contrary to the popular belief, we demonstrate that the loss in coding efficiency can be minimized with the use of complete-to-overcomplete discrete wavelet transforms for the performance of in-band prediction and update. Several aspects of the proposed coding algorithms are analyzed in detail and extensive experimental evaluations are carried out in order to assess their compression performance.

4.1 In-band Block-based Motion-compensated Prediction

In this section we define block-based motion-compensated prediction in the wavelet domain (in-band). A major bottleneck for in-band MCP approaches, which has been reported multiple times in the literature (e.g. [21] [22]), is that the classical dyadic wavelet decomposition (also named the critically-sampled DWT representation) is only periodically shift-invariant [23], with a period that corresponds to the subsampling factor of the specific decomposition level. Hence, accurate motion estimation is not feasible by using only the critically-sampled representation.

Extensive research efforts have been spent in the recent years to overcome the shift-variance problem of the critically-sampled DWT. In the area of video compression, the basic idea is to construct from the critically-sampled decomposition of the reference frame its overcomplete DWT representation for every resolution level, which is shift invariant. This representation can be used to estimate the motion in the video frames in a subband-by-subband, level-by-level or multi-resolution fashion [22] [24, 25] [12, 26]. The key-benefit of this approach is that motion compensation is still performed in the critically-sampled DWT representation of the current frame. Hence, the produced error-frames remain critically-sampled [21] [22] [27] [28] [26] [24] and there is no overhead for the error-frame coding. Moreover, this ensures that conventional wavelet-based image coding techniques such as zerotree-based or quadtree-based schemes [29] [30] are directly applicable for error-frame coding.

In this section, we start with a simple example that justifies the use of the overcomplete discrete wavelet rransform (ODWT) for in-band MCP. Subsequently, the definition of 2-D block-based in-band motion-compensated prediction (IB-MCP) is given.

4.1.1 A Simple One-dimensional Example of Wavelet-domain Motion Compensated Prediction

Based on a simple example, we illustrate how in-band MCP can achieve equivalent performance as the spatial-domain MCP by making use of the ODWT. For simplicity in the notation we restrict ourselves to 1-D signals.

Assume the case of the 1-D signal $X[t]$ shown in Figure IV-1. All FIR signals and filters can be considered in the Z-domain as Laurent polynomials and the letter z is typically reserved for this purpose. The Z-domain representation of this signal is[1]:

$$X(z) = ... + a \cdot z^{-2} + b \cdot z^{-1} + c + d \cdot z + e \cdot z^{2} + ... \tag{4.1}$$

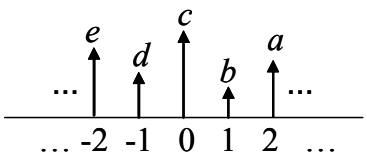


Figure IV-1. An 1-D signal in the time domain.

Based on the definition of the conventional (Type-I) polyphase transform [31], we can write the DWT of $X[t]$ as:

$$\begin{aligned} A_0(z) &= \tfrac{1}{2}\Big(H(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) + H(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}}) \Big) \\ D_0(z) &= \tfrac{1}{2}\Big(G(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) + G(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}}) \Big) \end{aligned} \tag{4.2}$$

where $U(z)$, $U = \{H, G\}$, are the analysis low- and high-pass DWT filters, respectively, and $S_0(z)$, $S = \{A, D\}$, are the low- and high-frequency DWT subbands of level one, respectively. Subscript 0 indicates that the even-numbered samples were kept after the downsampling operation. Although in wavelet-based image coding this typically suffices as a signal representation, we can define in the same manner the DWT that retains the odd-numbered samples after downsampling:

$$\begin{aligned} A_1(z) &= \tfrac{1}{2} z^{-\frac{1}{2}} \Big(H(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) - H(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}}) \Big) \\ D_1(z) &= \tfrac{1}{2} z^{-\frac{1}{2}} \Big(G(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) - G(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}}) \Big) \end{aligned} \tag{4.3}$$

[1] Note that we always consider finite-length signals, e.g. rows of images, as Laurent polynomials are always of finite degree in Z.

where subscript 1 indicates that the odd-numbered samples were kept after the downsampling operation. The difference in the DWT of equation (4.2) with the DWT of (4.3) stems from the definitions of the even and odd Type-I polyphase components of signal $X(z)$ [31]. We note here that the signals $S_i(z),\ i = \{0,1\}$ consist the single-level ODWT of the input signal $X(z)$ [23].

Now consider that $X[t]$ corresponds to a portion of a line in an input video frame. Moreover, consider a shift of $X[t]$, i.e. a translation of $X[t]$ (frame 1) to $X[t+k]$ (frame 2), $k \in \mathbb{Z}$. This corresponds to an ideal translational motion, where we consider that in the certain area covered by $X[t]$, motion between frame 1 and 2 was perfectly captured as a translation of the luminance values of $X[t]$ by k pixels. We note that, although this is an ideal scenario, which is usually only approximated in real video sequences, the considered block-based models are optimal only for the limited case of such types of motions [32].

We denote the shifted signal as $X_k[t] = X[t+k]$, or, equivalently, $X_k(z) = z^{-k}X(z)$. Motion compensation in the current (shifted) signal based on the reference signal is trivially written in this case as $\mathcal{MC}_{\text{spatial}}(X(z),k) = z^{-k}X(z)$ and the error signal is:

$$E_{\text{spatial}}(z) = X_k(z) - \mathcal{MC}_{\text{spatial}}(X(z),k) = 0 \tag{4.4}$$

In order to investigate how to perform in-band motion compensation in this case, we first need to express the DWT of the shifted signal. We note that the DWT of the reference signal $X[t]$ is given by equation (4.2). If the transform representation is shift invariant, in-band motion compensation should yield a zero error-signal for each subband.

By replacing $X(z)$ with $X_k(z)$ in (4.2), the DWT of $X_k[t]$ is given by:

$$S_{k,0}(z) = \tfrac{1}{2} z^{-\frac{k}{2}} \left(U(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) + (-1)^k U(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}}) \right) \tag{4.5}$$

Similar to the spatial-domain case, the trivial way to perform motion compensation in the low- and high-frequency subbands is $\mathcal{MC}_{\text{in-band}}(S_0(z),k) = z^{-\frac{k}{2}}S_0(z)$. If the translation was by an even number of pixels, i.e. $k = 2l$, $l \in \mathbb{Z}$, then the error signal for each subband is written as:

$$E\{S\}_{\text{in-band}}(z) = S_{2l,0}(z) - \mathcal{MC}_{\text{in-band}}(S_0(z),2l) = 0 \tag{4.6}$$

However, if the translation was by an odd number of pixels, i.e. $k = 2l+1$, $l \in \mathbb{Z}$, then:

$$E\{S\}_{\text{in-band}}(z) = S_{2l+1,0}(z) - \mathcal{MC}_{\text{in-band}}(S_0(z),2l+1) = 2U(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}}) \neq 0 \tag{4.7}$$

This means that, for translational motion, in-band motion compensation with the critically-sampled DWT of the current and reference signal is corresponding to the motion compensation in the spatial domain *only* if the translation is even, i.e. if it is a multiple of the downsampling factor. If the translation is by an odd number of pixels, no in-band vector in the critically-sampled DWT of the reference signal can provide zero motion compensation error. However, it is straightforward to verify that if we replace $X(z)$ with $X_k(z)$ in (4.3) and use the DWT samples produced by the "complementary" filtering-and-downsampling of (4.3), then we have:

$$\mathcal{MC}_{\text{in-band}}(S_i(z),k) = z^{-\frac{k-i}{2}} S_i(z) \tag{4.8}$$

with $k = 2l + i,\ l \in \mathbb{Z},\ i = \{0,1\}$ and:

$$E\{S\}_{\text{in-band}}(z) = S_{k,0}(z) - \mathcal{MC}_{\text{in-band}}\left(S_i(z),k\right) = 0 \tag{4.9}$$

This simple example shows that, for translational motion, motion-compensated prediction in the wavelet domain remains optimal, i.e. equivalent to the spatial-domain MCP, if the complementary wavelet coefficients are used for the odd-numbered signal translations [32]. This example can be extended to a higher number of decomposition levels where, again, the utilized polyphase component directly depends on the spatial shift [27] [28, 33]. Moreover, since in coding systems we have the DWT of $X(z)$, i.e. the subbands $S_0(z)$, in order to create $S_1(z)$, we need to perform an inverse transform to reconstruct $X(z)$, followed by the DWT shown in (4.3) [27]. Further developments have occurred recently in this topic [25] [33] [34] [35]. In addition, a generalized approach for this process [36] was recently termed as the complete-to-overcomplete discrete wavelet transform (CODWT). We refer to [36] and to the description of Chapter V for a comprehensive treatment of the topic and for a fast transform to generate the missing phase components of the ODWT of each decomposition level, starting from the critically-sampled decomposition.

Concerning the application of motion compensation with linear interpolation, we note that, if the DWT samples of both polyphase components are joined into one representation (yielding the so-called "à-trous" wavelet transform), then linear interpolation can occur in the wavelet domain directly, since the spatial DWT without downsampling and the spatial interpolation operation are both linear shift-invariant operations, hence their order of application can be interchanged [24].

In conclusion, for translational motion (where the commonly-used block-based spatial-domain MCP is optimal) [32], block-based in-band MCP remains optimal if the ODWT representation of the reference frame is used to perform motion estimation, and motion compensation occurs in the critically-sampled representation of the current frame. In this case, the equivalent model of the spatial-domain MCP system is produced. Any other techniques that may be devised to perform IB-MCP can approximate the performance of SD-MCP only in certain circumstances and hence they will not guarantee the equivalence between spatial and in-band motion compensated prediction for translational motion. In fact, with the use of the ODWT, in-band prediction is equivalent to a spatial-domain overlapped block-based motion compensation system, with the overlap area depending upon the support of the low- and high-pass reconstruction filters [11]. In the spatial domain, for non-uniform motions where neighbouring block-boundaries are misaligned, this produces a smoothed version of the non-overlapping block-based SD-MCP approach.

4.1.2 Extension in Two-dimensions

Similar to the conventional spatial-domain motion compensated prediction (SD-MCP) presented in Chapter III, in-band motion compensated prediction (IB-MCP) can significantly reduce the temporal dependency between successive video frames. Although less explored than SD-MCP, several IB-MCP schemes (e.g. mesh-based IB-MCP [37] or IB-MCP based on backward motion estimation [21]) can

be found in the recent literature. Nevertheless, for the capabilities of today's implementation platforms, block-based models seem to offer the best trade-offs in terms of system complexity versus prediction efficiency. Moreover, as it will be shown later in this chapter, block-based models provide many possibilities for practical rate-distortion optimization algorithms, which can dramatically increase the performance of a practical video coding system.

As stated in the previous subsection, one can establish the equivalence between IB-MCP and SD-MCP. In order to illustrate this in more detail, in this section we follow similar notation as in Chapter III. Figure IV-2 presents an example of MCP in the spatial domain, as introduced in the previous chapter. The current video frame A_t is separated in non-overlapping blocks of $B_m \times B_n$ pixels. For each block, a search is performed in the reference frame A_{t-1} to minimize a matching criterion between the blocks in the search area ($S_m \times S_n$ pixels) and the current block. The subsequently-produced motion vector (d_m^*, d_n^*), with $-\frac{S_m}{2} \le d_m^* < \frac{S_m}{2}$ and $-\frac{S_n}{2} \le d_n^* < \frac{S_n}{2}$, and the error frame H_t are compressed and transmitted to the decoder.

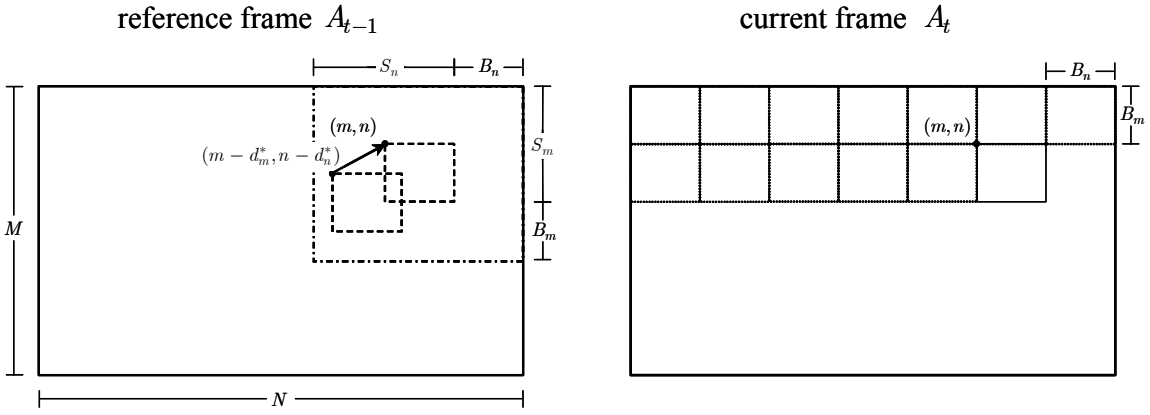


Figure IV-2. An example of spatial-domain block-based motion estimation.

Following the example of the previous subsection, Figure IV-3 presents an example of IB-MCP in the one-level DWT decomposition. The notations introduced in the figure follow the symbolism of Figure IV-2. Each subband U of the one-level 2-D DWT of the current frame is split into non-overlapping blocks of $\frac{B_m}{2} \times \frac{B_n}{2}$ wavelet coefficients. Each of these blocks is predicted by performing a matching with the blocks that lie in the area of $\frac{S_m}{2} \times \frac{S_n}{2}$ coefficients of the ODWT $\mathcal{S}^1_{(p_{U,m},p_{U,n})}\mathcal{T}^1_U A_{t-1}$ around the in-band block position (m,n) in the reference frame, with $p_{U,m}, p_{U,n} = \{0,1\}$. The matching criterion $\mathcal{C}(\cdot)$ used in practice corresponds to the sum of absolute differences, i.e. $\mathcal{C}(a) = |a|$. For the example of Figure IV-3, this problem can be stated as:

$$\forall U : \{(p_{U,m}^*, p_{U,n}^*), (d_{U,m}^*, d_{U,n}^*)\} = \arg \min_{\substack{(p_{U,m},p_{U,n}) \\ (d_{U,m},d_{U,n})}} \sum_{i=0}^{B_m/2-1} \sum_{j=0}^{B_n/2-1} \mathcal{C}\big(\mathcal{T}^1_U A_t[m+i, n+j] - \mathcal{S}^1_{(p_{U,m},p_{U,n})}\mathcal{T}^1_U A_{t-1}[m+i-d_{U,m}, n+j-d_{U,n}]\big) \tag{4.10}$$

where, for every subband U we have: $-\frac{S_m}{4} \le d_{U,m} < \frac{S_m}{4}$ and $-\frac{S_n}{4} \le d_{U,n} < \frac{S_n}{4}$. Notice that, for each subband U, the solution $\{(p^*_{U,m}, p^*_{U,n}),(d^*_{U,m}, d^*_{U,n})\}$ consists of the in-band displacement vector (or in-band motion vector) $(d^*_{U,m}, d^*_{U,n})$ and the ODWT phase component $(p^*_{U,m}, p^*_{U,n})$. Similar as in the spatial-domain case, the desired solution is constant over the wavelet coefficients $\mathcal{T}^1_U A_t[m+i, n+j]$ of the current frame A_t. This means that the coefficients of each non-overlapping block at position (m,n) of each subband U of the DWT of A_t are all associated with one solution: $\{(p^*_{U,m}, p^*_{U,n}),(d^*_{U,m}, d^*_{U,n})\}$.

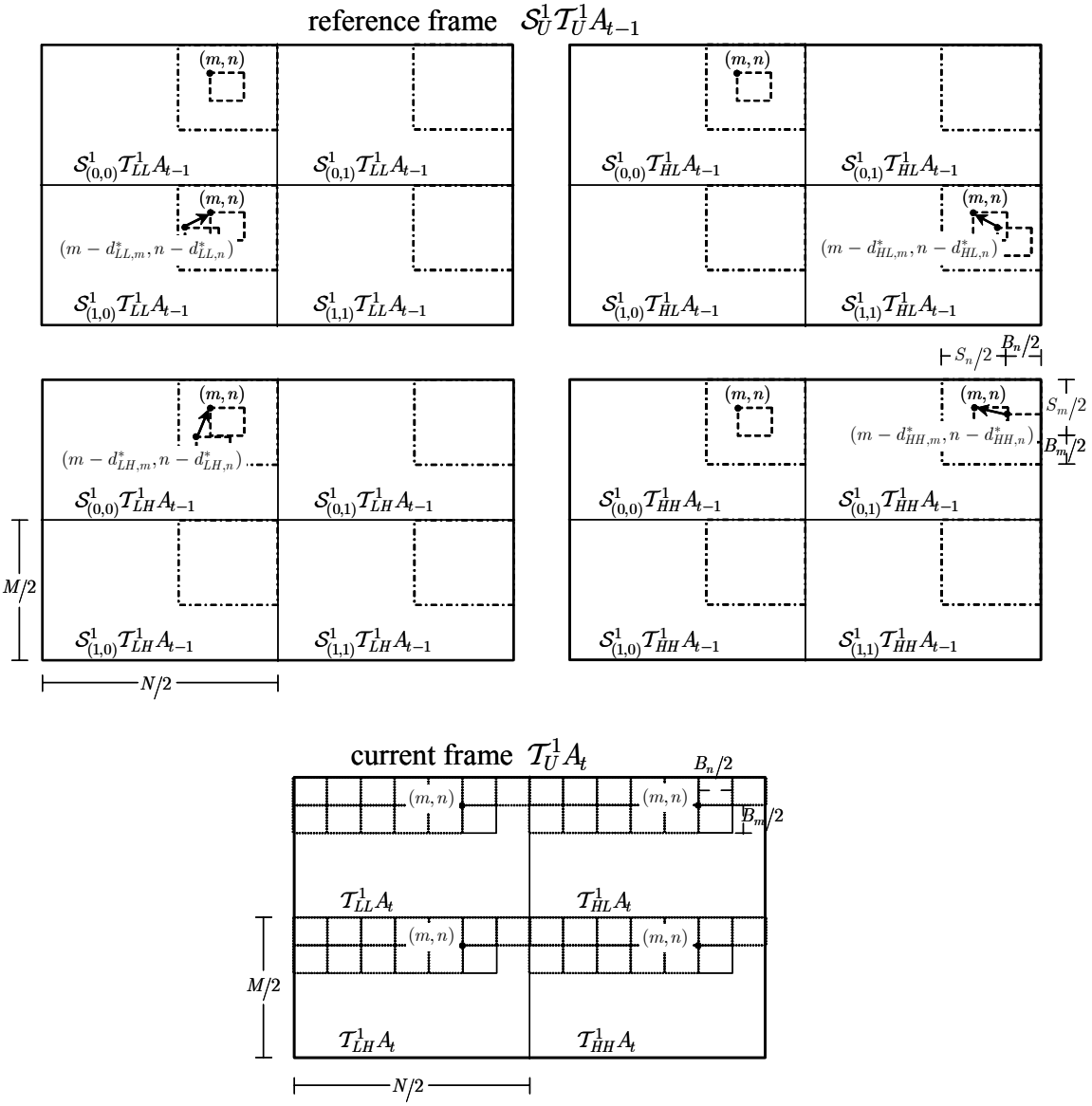


Figure IV-3. An example of block-based in-band motion-compensated prediction. Notations: $U = \{LL, LH, HL, HH\}$ denotes the 2-D DWT subbands (L, H stand for low- and high-pass filtering on rows and columns, respectively); $\mathcal{T}^l_U A$ is the subband U (of decomposition level l) of the DWT of frame A; $\mathcal{S}^l_{(p_i,p_j)}\mathcal{T}^l_U A$ is the ODWT-phase (p_i, p_j) of the wavelet subband $\mathcal{T}^l_U A$; $(d^*_{U,m}, d^*_{U,n})$ represent the in-band motion vector of subband U that corresponds to in-band position (m,n).

Similar to SD-MCP, the classical way of solving the example of (4.10) is the (brute force) full-search motion estimation algorithm. Other techniques based on hierarchical search can also be envisaged [22] in order to reduce the search complexity. Since the four blocks in the four subbands amount to a total of $B_m \times B_n$ wavelet coefficients and the total search area is also amounting to $S_m \times S_n$ wavelet coefficients, the total amount of SAD calculations performed for each block in order to solve (4.10) is equivalent to the number of SAD calculations of the spatial-domain motion estimation. However, one significant difference between the two examples of Figure IV-2 and Figure IV-3 is that, for the wavelet-domain area that corresponds to $B_m \times B_n$ pixels, four motion vectors are estimated instead of one; this provides the in-band case with the additional feature of frequency selectivity for MCP, i.e. different spatial frequencies can be predicted in a different way from the ODWT of the reference frame. This may increase the prediction efficiency, at the cost of providing more motion-vector data to be transmitted to the decoder [22].

An extension of the example of Figure IV-3 to more spatial decomposition levels for the DWT is straightforward. Since the downsampling of the transform reduces the resolution of each subband, one can opt for the use of dyadically-reduced block sizes in order to cover the same area in the spatial-domain, following the wavelet tree concept [27] presented in Chapter III. Alternatively, fixed-size blocks can be used for each resolution level [10]. Each case corresponds to a different motion-vector overhead, if motion estimation occurs separately in each subband and each resolution. In the general case of a wavelet decomposition in k-levels, for resolution level l, $1 \le l \le k$, we have:

$$\begin{aligned}\forall U,l : \left\{(p_{U,m}^{l,*}, p_{U,n}^{l,*}), \left(d_{U,m}^{l,*}, d_{U,n}^{l,*}\right)\right\} = \arg \min_{\substack{(p_{U,m}^{l}, p_{U,n}^{l}) \\ (d_{U,m}^{l}, d_{U,n}^{l})}} \sum_{i=0}^{B_m^l-1} \sum_{j=0}^{B_n^l-1} \mathcal{C}\Big(\mathcal{T}_U^l A_t[m+i, n+j] \\ -\mathcal{S}^l_{(p_{U,m}^{l}, p_{U,n}^{l})} \mathcal{T}_U^l A_{t-1}[m+i-d_{U,m}^l, n+j-d_{U,n}^l]\Big)\end{aligned} \tag{4.11}$$

where the notations are an extension of the notations used previously with the superscript l indicating the resolution level and the positions (m,n) at each level depending on the partitioning of each subband into non-overlapping blocks of $B_m^l \times B_n^l$ wavelet coefficients. For every subband U of level l, we have: $-\frac{S_m^l}{2} \le d_{U,m} < \frac{S_m^l}{2}$ and $-\frac{S_n^l}{2} \le d_{U,n} < \frac{S_n^l}{2}$ and $0 \le p_{U,m}^l < 2^l$, $0 \le p_{U,n}^l < 2^l$. If we want the search to be limited within an area in the wavelet domain that corresponds to the search area of the SD-MCP example of Figure IV-2, we set $S_m^l = \frac{S_m}{2^l}$ and $S_n^l = \frac{S_n}{2^l}$.

The generic motion estimation problem stated in (4.11) is also referred to as the multiresolution band-by-band wavelet-domain motion estimation [22] [26] [38]. Obviously, the motion estimation problem of (4.11) represents the broad class of motion estimation algorithms which are possible in the wavelet domain. Out of this broad class, several subclasses of wavelet-domain (in-band) motion estimation algorithms can be defined, which may be more interesting for practical video coding systems. For example, if we impose the constraint that one in-band displacement vector and one ODWT phase component is generated for all the subbands of a certain resolution of (4.11), which we denote by $\left\{(p_{\text{lbl},m}^{l,*}, p_{\text{lbl},n}^{l,*}), \left(d_{\text{lbl},m}^{l,*}, d_{\text{lbl},n}^{l,*}\right)\right\}$, i.e.:

$$\forall l : \begin{cases} p_{LL,m}^{l,*} = p_{LH,m}^{l,*} = p_{HL,m}^{l,*} = p_{HH,m}^{l,*} = p_{\mathrm{lbl},m}^{l,*} \\ p_{LL,n}^{l,*} = p_{LH,n}^{l,*} = p_{HL,n}^{l,*} = p_{HH,n}^{l,*} = p_{\mathrm{lbl},n}^{l,*} \\ d_{LL,m}^{l,*} = d_{LH,m}^{l,*} = d_{HL,m}^{l,*} = d_{HH,m}^{l,*} = d_{\mathrm{lbl},m}^{l,*} \\ d_{LL,n}^{l,*} = d_{LH,n}^{l,*} = d_{HL,n}^{l,*} = d_{HH,n}^{l,*} = d_{\mathrm{lbl},n}^{l,*} \end{cases} \tag{4.12}$$

then (4.11) is simplified to the so-called level-by-level wavelet-domain motion estimation [39] [10]. As we shall see in the following parts of this chapter, this ME algorithm is useful when scalability in resolution is of particular interest, since different motion vectors are generated for each spatial decomposition level.

Another subclass of IB-ME algorithms for the k-level DWT decomposition can be defined based on (4.11) if we set the following constraints:

- the non-overlapping blocks of each resolution l of the current frame have dyadically-decreased sizes, i.e.:

$$\forall l : \begin{cases} B_m^l = B_{\mathrm{wt},m} \cdot 2^{-l} \\ B_n^l = B_{\mathrm{wt},n} \cdot 2^{-l} \end{cases} . \tag{4.13}$$

  where $B_{\mathrm{wt},m}$ and $B_{\mathrm{wt},n}$ are the sizes of the corresponding area in the spatial domain.

- for each resolution level l, all blocks at the corresponding positions within each subband U of that level are considered together, as in the case of level-by-level wavelet-domain ME;

- the in-band displacement vector and the ODWT phase component for the first resolution level are given as for the case of the level-by-level wavelet-domain motion estimation:

$$\begin{cases} p_{\mathrm{wt},m}^{1,*} = p_{\mathrm{lbl},m}^{1,*} \\ p_{\mathrm{wt},n}^{1,*} = p_{\mathrm{lbl},n}^{1,*} \\ d_{\mathrm{wt},m}^{1,*} = d_{\mathrm{lbl},m}^{1,*} \\ d_{\mathrm{wt},n}^{1,*} = d_{\mathrm{lbl},n}^{1,*} \end{cases} . \tag{4.14}$$

- the in-band displacement vector and the ODWT phase component generated for each subsequent resolution level l satisfy the following relation:

$$\forall l > 1 : \begin{cases} p_{\mathrm{wt},m}^{l,*} = 2 \cdot p_{\mathrm{wt},m}^{l-1,*} + d_{\mathrm{wt},m}^{l-1,*} (\mathrm{mod}\, 2) \\ p_{\mathrm{wt},n}^{l,*} = 2 \cdot p_{\mathrm{wt},n}^{l-1,*} + d_{\mathrm{wt},n}^{l-1,*} (\mathrm{mod}\, 2) \\ d_{\mathrm{wt},m}^{l,*} = \left\lfloor \dfrac{d_{\mathrm{wt},m}^{l-1,*}}{2} \right\rfloor \\ d_{\mathrm{wt},n}^{l,*} = \left\lfloor \dfrac{d_{\mathrm{wt},n}^{l-1,*}}{2} \right\rfloor \end{cases} . \tag{4.15}$$

  where $\lfloor a \rfloor$ is the integer part of a and $a(\mathrm{mod}\, b) = a - b\left\lfloor \frac{a}{b} \right\rfloor$ is the modulo operation, $a, b \in \mathbb{Z}$.

The constraints of (4.13) – (4.15) lead to the so-called wavelet-tree (or wavelet-block) in-band motion estimation [22]. If we set $B_{\mathrm{wt},m} = B_m$ and $B_{\mathrm{wt},n} = B_n$, this algorithm results in the same overhead for the uncoded motion-vector information as in the spatial-domain case (Figure IV-2). This is due to the fact that, in this case, one wavelet tree corresponds to the area of an image block of $B_m \times B_n$ pixels, and one in-band motion vector is generated for each wavelet tree (equation (4.14)), which contains indices within an area of coefficients in the overcomplete wavelet domain that corresponds to $S_m \times S_n$ discrete positions.

Since interpolation of the reference frames in the spatial-domain motion-estimation algorithm is a practical tool to improve the efficiency of block-based algorithms in capturing the underlying scene motion, it is desirable to obtain an equivalent interpolation algorithm for the in-band motion-estimation algorithms. This can be achieved based on the observation that interpolation to sub-pixel accuracy can be performed directly in the ODWT of the reference frame(s), if their ODWT-domain phase components are interleaved to create the undecimated discrete wavelet transform (UDWT). This essentially stems from the fact that linear interpolation and the DWT without decimation are both linear shift-invariant operators and their order of application to the input signal can be interchanged [24] [40]. One such example is given for the low-frequency subband of the DWT in Figure IV-4. The single-level overcomplete representation of the LL subband depicted in the upper-left part of the figure is converted to its equivalent in the UDWT domain by interleaving the ODWT-phase components horizontally and vertically. The interleaving process for all the subbands of the ODWT is elaborated in the following subsection. Interpolation to the equivalent of half-pixel accuracy occurs in the UDWT domain by using the interpolation filters presented in Chapter III. Finally, the interpolated UDWT is converted to the interpolated ODWT. This process generates fractional ODWT phase components [12], which are depicted in the lower-right part of Figure IV-4. Note that the interpolation process is performed in the same manner for the high-frequency subbands of the ODWT. Higher decomposition levels and higher interpolation accuracy can be obtained following the same principles.

Similar to the fractional-pixel ME algorithms described in Chapter III, practical algorithms for fractional-phase in-band ME begin by searching for the best match of each in-band block of the current frame with a block in the integer-phase representation of the ODWT of the reference frame. Subsequently, in the neighbourhood of the best match found by the integer-phase ODWT-domain search algorithm, a new search is performed within the subbands with fractional ODWT phase [12]. The position of the best match is determined by the in-band displacement vector and the integer (or fractional) ODWT-phase component. As a result, in contradiction to spatial-domain ME, in the case of IB-ME the in-band displacement vector coordinates $\left(d^*_{U,m}, d^*_{U,n}\right)$ are always integers. Nevertheless, the ODWT phase component values $\left(p^*_{U,m}, p^*_{U,n}\right)$ can be decimal numbers, indicating fractional phases in the interpolated ODWT.

Finally, similar to the spatial-domain case, although the complexity of block-based IB-ME is reduced by the use of fast algorithms [41], only the (brute force) full-search algorithm is guaranteed to obtain the position (within the predefined grid precision) that provides the global minimum for (4.11).

Figure IV-4. Interpolation in the ODWT domain. An example of in-band interpolation to half ODWT-phase accuracy within the LL subband of the single-level ODWT of a video frame is given.

4.1.3 Overcomplete-to-Undecimated Discrete Wavelet Transform and the Correspondence of Phase Components in the ODWT Domain

The transformation from the ODWT to the UDWT, performed by the phase-interleaving process demonstrated in Figure IV-4, follows the following pattern.

The ODWT-phase components of a wavelet coefficient at position (m,n) in the LL subband of resolution level l, can be expanded in binary form as:

$$(p^l_{LL,m}, p^l_{LL,n}) = \left(\sum_{i=0}^{l-1} b_i 2^i, \sum_{j=0}^{l-1} b_j 2^j\right) \tag{4.16}$$

where $\forall i,j : b_i, b_j = \{0,1\}$ are the binary digits from the binary representation of the horizontal and vertical phase components. The corresponding position in the undecimated LL subband of level l is given by:

$$(m^l_{LL,\mathrm{UDWT}}, n^l_{LL,\mathrm{UDWT}}) = (m \cdot 2^l + \sum_{i=0}^{l-1} b_i 2^{l-1-i}, n \cdot 2^l + \sum_{j=0}^{l-1} b_j 2^{l-1-j}) . \qquad (4.17)$$

The last equation performs the interleaving of the ODWT phase components based on a bit-reversal scheme for the phase indices. This stems from the multilevel filtering-and-downsampling process of the (two-band) ODWT [23], with the assumption that the low-pass filter is linear phase [23] [42]. A pictorial explanation of the conversion process formulated in (4.17) is given in Figure IV-5 for the case of two decomposition levels.

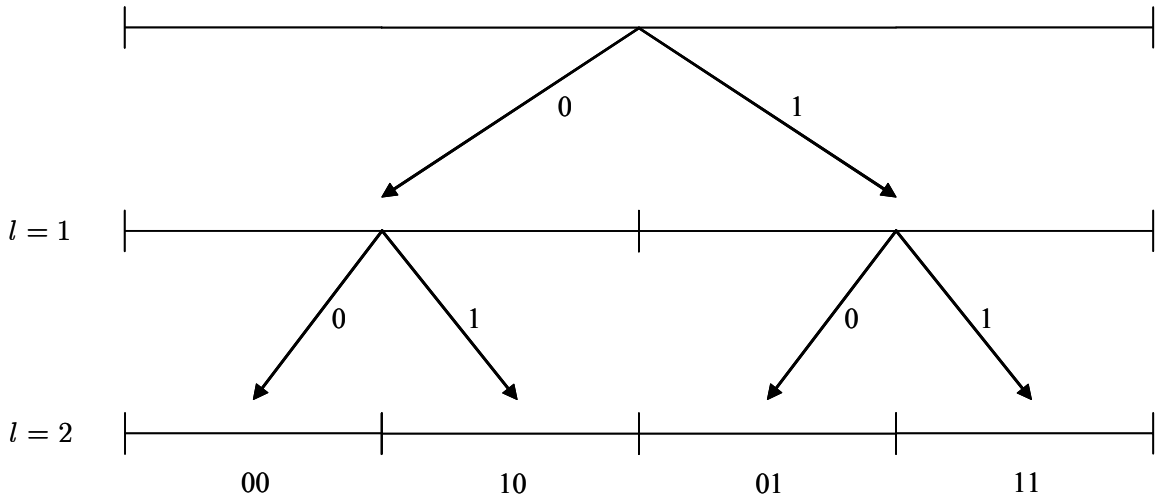


Figure IV-5. The interval corresponding to the input signal (top row) is decomposed in two sections corresponding to the even and odd ODWT-phase components at the first level ($l = 1$ – middle of the figure). In addition, each ODWT-phase component is decomposed in the same manner for $l = 2$. The binary representation at the bottom of the figure represents the corresponding position in the undecimated DWT of the input signal, which can be found by reversing the "path" of the ODWT phases taken at each level.

It is easy to show that the inverse transformation (UDWT to ODWT) for the LL subband is given by the same bit-reversal process. If interpolation is applied in the undecimated representation of the LL subband (as shown in the example of Figure IV-4), then we additionally define $l_R = l + \log_2 R$, where R indicates the interpolation precision (e.g. $R = 2$ for half ODWT-phase accurate interpolation in the wavelet domain). In addition, in this case the indices in the undecimated LL subband of level l given by (4.17) (which corresponded to the integer grid) are now multiplied by R. The part of the UDWT position that is relevant to the phase indices is then isolated and represented it in binary form as:

$$(m_{LL,\mathrm{UDWT}} (\mathrm{mod}\, 2^{l_R}), n_{LL,\mathrm{UDWT}} (\mathrm{mod}\, 2^{l_R})) = \left(\sum_{i=0}^{l_R-1} b_i^{\mathrm{r}} 2^i, \sum_{j=0}^{l_R-1} b_j^{\mathrm{r}} 2^j \right). \qquad (4.18)$$

where $\forall i \geq \log_2 R : b_i^{\mathrm{r}} = b_{i-\log_2 R}$ and $\forall j \geq \log_2 R : b_j^{\mathrm{r}} = b_{j-\log_2 R}$, while $\forall i, j \leq \log_2 R$ the $b_i^{\mathrm{r}}, b_j^{\mathrm{r}}$ represent the binary digits of the interpolated positions in the UDWT.

The coefficient position within the corresponding LL subband of the ODWT domain is given by:

$$(m,n) = \left(\left\lfloor \frac{m_{LL,\mathrm{UDWT}}}{2^{l_R}} \right\rfloor, \left\lfloor \frac{n_{LL,\mathrm{UDWT}}}{2^{l_R}} \right\rfloor \right) \tag{4.19}$$

while the interpolated (fractional) ODWT phase components are:

$$(p_{LL,m}^{l_R}, p_{LL,n}^{l_R}) = \left(\frac{1}{R} \sum\nolimits_{i=0}^{l_R-1} b_i^{\mathrm{r}} 2^{l_R-1-i}, \frac{1}{R} \sum\nolimits_{j=0}^{l_R-1} b_j^{\mathrm{r}} 2^{l_R-1-j} \right). \tag{4.20}$$

For the high-frequency subbands of each level l, a similar transformation is performed in order to obtain the UDWT from the ODWT. However, one significant difference is that, before the interleaving of the ODWT phases, the even-numbered (or zero) phases are exchanged (flipped) with the odd-numbered ones. For example, for the HH subband, which is produced by high-pass filtering-and-downsampling in the horizontal and vertical direction, we have:

$$p_{HH,m}^{l,\mathrm{f}} = \begin{cases} p_{HH,m}^{l} + 1, \text{ if } p_{HH,m}^{l} = 2a, a \in \mathbb{Z}_+ \\ p_{HH,m}^{l} - 1, \text{ otherwise} \end{cases}, \; p_{HH,n}^{l,\mathrm{f}} = \begin{cases} p_{HH,n}^{l} + 1, \text{ if } p_{HH,n}^{l} = 2a, a \in \mathbb{Z}_+ \\ p_{HH,n}^{l} - 1, \text{ otherwise} \end{cases} \tag{4.21}$$

Subsequently, the process is performed as shown in (4.16), (4.17) by replacing $(p_{LL,m}^{l}, p_{LL,n}^{l})$ with $(p_{HH,m}^{l,\mathrm{f}}, p_{HH,n}^{l,\mathrm{f}})$. Similarly, for the inverse, the identical process is followed (equations (4.18) – (4.20)) and the final ODWT phases of the HH subband are created by the inverse flipping process.

For the remaining DWT subbands (LH and HL), the conversion process is following a combination of the rules described before for high and low-frequency subbands, depending on whether the filter applied in the horizontal or vertical direction was a low-pass or a high-pass one.

The difference in the conversion of the low and high-frequency subbands is stemming from the definition of the conventional filter-banks used in image coding. [43] [44]. One peculiarity of the original filter-bank designs, and this is true for all the commonly-used wavelet filter-banks in compression applications [43], is that the low-frequency and high-frequency coefficients produced with the critically-sampled DWT correspond to different positions in the UDWT domain. For example, for the one-dimensional single-level (critically-sampled) decomposition with the original 9/7 filter-bank presented in [44] (and also for the filter-banks in [43] [45]), the coefficients that correspond to the even positions of the UDWT are produced for the low-frequency subband, and the coefficients corresponding to odd positions are produced for the high-frequency subband. This production of alternating positions for the different frequency bands is due to the filter delays of the original filter-bank: the original filter delays are designed such that the produced decomposition can be stored in-place [43] by alternating the low and high-frequency coefficients of the transform in even and odd-numbered memory positions. Equivalently, if one produces the coefficients that are missing from the critically-sampled decomposition, one receives the odd and even-numbered coefficients for the low and high-frequency part of the UDWT, respectively. This difference, if extended to multiple decomposition levels, creates the necessity for the flipping process of equation (4.21).

In general, the transformation from the ODWT to the UDWT domain is important when it comes to practical implementations. In particular, the UDWT domain positions provide an immediate correspondence to the spatial-domain positions, while the indices in the ODWT domain do not, since

the ODWT phase components are shuffled and each ODWT subband with a certain phase component is critically-sampled [42]. As a result, similar to the process of Figure IV-4 that was proposed for in-band interpolation, whenever particular embodiments of IB-ME or IB-MCP require a grouping of all the coefficients with the same ODWT phase component and in-band displacement vector, as seen for example in the level-by-level IB-ME constraints of (4.12), in reality we group the coefficients corresponding to the same location in the UDWT. For example, this means that, in order to satisfy the constraint of (4.12) we group the ODWT coefficients for which:

$$\begin{aligned}(m^l_{LL,\mathrm{UDWT}}, n^l_{LL,\mathrm{UDWT}}) &= (m^l_{LH,\mathrm{UDWT}}, n^l_{LH,\mathrm{UDWT}}) = (m^l_{HL,\mathrm{UDWT}}, n^l_{HL,\mathrm{UDWT}}) \\ &= (m^l_{HH,\mathrm{UDWT}}, n^l_{HH,\mathrm{UDWT}})\end{aligned} . \qquad (4.22)$$

Similarly, whenever arithmetic operations are performed to the ODWT phase components or the in-band displacement vectors, e.g. as seen in equation (4.15), we always implement the particular operations in the UDWT domain and then convert back to the ODWT-domain. However, although utilized in the actual implementations of the algorithms proposed in this chapter, the ODWT-to-UDWT conversion is not explicitly indicated in the analytical formulations in order to simplify notations; nevertheless, it is important to notice that we always imply its usage whenever cross-subband ODWT phase components or in-band displacement vectors are linked together, or arithmetic operations are performed on this data.

4.1.4 In-band Motion Compensated Prediction based on the ODWT of the Reference Frame

After the IB-ME stage described by (4.11), each coefficient of the DWT of the current frame, $\mathcal{T}^l_U A_t[m,n]$ with $1 \le l \le k$, is associated via its in-band displacement vector $(d^{\mathcal{F}^l_1(t-1)}_{U,m}, d^{\mathcal{F}^l_1(t-1)}_{U,n})$ and ODWT-phase $(p^{\mathcal{F}^l_1(t-1)}_{U,m}, p^{\mathcal{F}^l_1(t-1)}_{U,n})$ with a coefficient in the interpolated ODWT of the reference frame. Superscript $\mathcal{F}^l_{\Delta\tau}(\tau)$ denotes forward in-band prediction via in-band motion estimation that matches the in-band representation of level l of the reference frame at time instant τ with the current frame at time instant $\tau + \Delta\tau$. As a result, after IB-MCP, the wavelet-domain error frame is given by:

$$\begin{aligned}\mathcal{T}^l_U H_t[m,n] &= \mathcal{T}^l_U A_t[m,n] \\ &\quad - \mathcal{I}_{\left(i^{\mathcal{F}^l_1(t-1)}_{U,m}, i^{\mathcal{F}^l_1(t-1)}_{U,n}\right)} \mathcal{S}^l_{U,\left(\left\lfloor p^{\mathcal{F}^l_1(t-1)}_{U,m} \right\rfloor, \left\lfloor p^{\mathcal{F}^l_1(t-1)}_{U,n} \right\rfloor\right)} \mathcal{T}^l_U A_{t-1}[m - d^{\mathcal{F}^l_1(t-1)}_{U,m}, n - d^{\mathcal{F}^l_1(t-1)}_{U,n}] .\end{aligned} \qquad (4.23)$$

where: $i^{\mathcal{F}^l_1(t-1)}_{U,m} = p^{\mathcal{F}^l_1(t-1)}_{U,m} - \left\lfloor p^{\mathcal{F}^l_1(t-1)}_{U,m} \right\rfloor$, $i^{\mathcal{F}^l_1(t-1)}_{U,n} = p^{\mathcal{F}^l_1(t-1)}_{U,n} - \left\lfloor p^{\mathcal{F}^l_1(t-1)}_{U,n} \right\rfloor$ are the interpolated (fractional) ODWT-phase positions; $\mathcal{I}_{(x,y)}\mathcal{S}^l_{(i,j)}A[m,n]$ is the interpolated ODWT coefficient at fractional ODWT-phase position (x,y) from the ODWT coefficient $\mathcal{S}^l_{(i,j)}A[m,n]$ (derived as explained in Figure IV-4), where $x, y = \{0, \frac{1}{R}, \ldots, \frac{R-2}{R}, \frac{R-1}{R}\}$ and R indicates the interpolation precision (e.g. $R = 2$ for half ODWT-phase accurate interpolation in the wavelet domain). As mentioned in Chapter III, the interpolation operation $\mathcal{I}_{(x,y)}$ can be designed in many ways. For the experiments reported in this dissertation, the same interpolation filters are used for the spatial-domain and in-band video coding experiments. In this way, although not necessarily optimal in each case, interpolation is not affecting the outcome of comparative experiments between spatial-domain and in-band algorithms.

4.1.5 Advanced In-band Motion Compensated Prediction

Similarly to Chapter III, one can define advanced block-based MCP schemes in the wavelet domain, i.e. in-band multihypothesis motion compensated prediction (IBMH-MCP). These algorithms can provide enhanced adaptivity to the scene characteristics by changing the prediction parameters according to the content present in a series of reference frames; in addition, this process can have a different instantiation for each spatial resolution [39]. In the general case of IBMH-MCP that uses T reference frames situated in time at positions $q = \{t - t^{\text{init}}, \ldots, t-1, t+1, \ldots, t + t^{\text{end}}\}$, the wavelet-domain error-frame of resolution level l resulting from IBMH-MCP is given by:

$$\mathcal{T}_U^l H_t[m,n] = \mathcal{T}_U^l A_t[m,n] - \sum_{q=t-t^{\text{init}}}^{t-1} \left(w_{U,q}^l[m,n] \mathcal{S}^l_{U,\left(p^{\mathcal{F}^l_{t-q}(q)}_{U,m}, p^{\mathcal{F}^l_{t-q}(q)}_{U,n}\right)} \mathcal{T}_U^l A_q[m - d^{\mathcal{F}^l_{t-q}(q)}_{U,m}, n - d^{\mathcal{F}^l_{t-q}(q)}_{U,n}] \right) - \sum_{q=t+1}^{t+t^{\text{end}}} \left(w_{U,q}^l[m,n] \mathcal{S}^l_{U,\left(p^{\mathcal{B}^l_{q-t}(q)}_{U,m}, p^{\mathcal{B}^l_{q-t}(q)}_{U,n}\right)} \mathcal{T}_U^l A_q[m - d^{\mathcal{B}^l_{q-t}(q)}_{U,m}, n - d^{\mathcal{B}^l_{q-t}(q)}_{U,n}] \right) \quad (4.24)$$

where $w_{U,q}^l[m,n]$ are the space-varying multihypothesis weight factors of subband U of decomposition level l of the DWT of reference frame q, and $\left\{\left(p^{\mathcal{F}^l_{t-q}(q)}_{U,m}, p^{\mathcal{F}^l_{t-q}(q)}_{U,n}\right), \left(d^{\mathcal{F}^l_{t-q}(q)}_{U,m}, d^{\mathcal{F}^l_{t-q}(q)}_{U,n}\right)\right\}$, $\left\{\left(p^{\mathcal{B}^l_{q-t}(q)}_{U,m}, p^{\mathcal{B}^l_{q-t}(q)}_{U,n}\right), \left(d^{\mathcal{B}^l_{q-t}(q)}_{U,m}, d^{\mathcal{B}^l_{q-t}(q)}_{U,n}\right)\right\}$ represent the forward and backward displacement vectors (respectively) that correspond to wavelet coefficient $\mathcal{T}_U^l A_q[m,n]$. Note that the interpolation operation has been omitted in (4.24), for simplicity in the notation. Moreover, for simplicity in our description, the analytical formulation of IBMH-MCP seen in (4.24) does not include the case where more than one motion vectors originate from the same reference frame. Nevertheless, our experiments with IBMH-MCP included this case as well in order to incorporate the most generic instantiation of multihypothesis prediction.

For block-based IBMH-MCP, $w_{U,q}^l[m,n]$ is constant over a certain group of wavelet coefficients (m,n) corresponding to a wavelet-domain block of decomposition level l in subband U of the q-th reference frame. Overall, equation (4.24) presents a generalization of the spatial-domain MH-MCP presented in Chapter III, since different prediction modes can be created for different wavelet subbands and different resolution levels. Currently, except of the algorithms presented in this dissertation, there is no other work in the literature able to provide a practical in-band MH-ME algorithm for the generalized problem of block-based IBMH-MCP represented by (4.24). As it will be shown later in this chapter, our proposed solution provides a working system by restricting the number of allowable block sizes and the values of t^{init} and t^{end}. Moreover, although possible in theory, our solution is not performing joint optimization for different multihypothesis mode selection *across* the various subbands and resolution levels. Instead, it is restricted to one mode for all the subbands of a certain resolution level, and the optimization process is performed separately for each resolution. Nevertheless, as an outcome of our work, a working IBMH-MCP system is provided, which can demonstrate similar coding-efficiency improvement due to optimized in-band MH-ME as in the case of spatial-domain MH-ME.

4.2 Scalable Video Coding with In-band Prediction within the Closed-loop Structure

This section presents a modified closed-loop temporal prediction structure that operates in-band. This topic has already been targeted in a number of publications [21] [27] [24] [26] [46] [41]. Our work in this area focused on providing a structure that is based on the use of the ODWT for efficient IB-MCP and the use of a scalable coder for the intra- and error-frames, which enables resolution and bitrate scalability within the in-band closed-loop structure.

The basic design of an in-band closed-loop video codec is presented in Figure IV-6. In comparison to the conventional closed-loop video coding structure shown in Chapter III, the design of Figure IV-6 has the following differences:

- The spatial DWT decomposition $\mathcal{T}_U^l$ is moved out of the prediction loop since the entire process is performed in-band.
- A module that constructs the ODWT from the DWT of the reference frame ($\mathcal{S}_U^l$) is included in the prediction loop, i.e. a complete-to-overcomplete discrete wavelet transform [36].

In principle, the design of Figure IV-6 is repeated for each decomposition level l, $1 \le l \le k$. Moreover, for each resolution level, the design of the decoder for that resolution can be seen in the dotted rectangle. In order to display the reconstructed frames at resolution level l, the subbands $\mathcal{T}_U^l \widetilde{A}_t$ together with the subbands of the coarser resolution levels $(l+1,\ldots,k)$ are used in the IDWT that provides the output video.

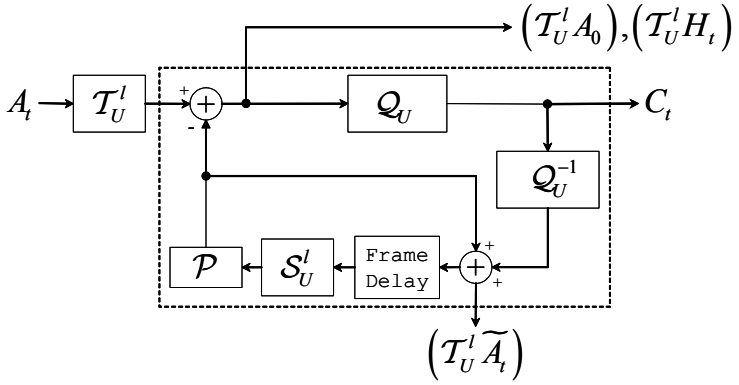


Figure IV-6. The in-band closed-loop video compression scheme for each resolution level l, $1 \le l \le k$. Notations: A_τ consists the input video frame at time instant $\tau = 0, t$; $\widetilde{A}_t$ is the reconstructed frame; H_t consists the error frame; C_t denotes the transformed and quantized error frame DWT subbands of level l using the DWT $\mathcal{T}_U^l$ and the subband quantization $\mathcal{Q}_U^l$; $\mathcal{P}$ denotes IB-MCP using the level-l ODWT subbands produced by the complete-to-overcomplete DWT module $\mathcal{S}_U^l$.

An important aspect of the CODWT module ($\mathcal{S}_U^l$) concerns the manner through which the ODWT is created for each resolution level l. In the scheme depicted in Figure IV-6, and for the remaining video coding architectures of this chapter, the $\mathcal{S}_U^l$ module appears to receive its input from a frame buffer (frame delay), which contains the critically-sampled subbands of resolution level l. This indicates that the ODWT is created from the DWT of level l. This construction is termed as "single-rate CODWT" in this dissertation, since, as shown in Chapter V and elsewhere [36] [47], a single-rate calculation scheme can be used for this CODWT computation. Nevertheless, it is important to note that the general formulation of the CODWT of each resolution l [36] demonstrates that, given the complete DWT decomposition of l-levels, the CODWT of each resolution uses all subbands of all resolutions $1,\ldots,l$ of the DWT. To differentiate from the previous case, this construction is termed as "multi-rate CODWT". Further details on the differences between the two cases and the architectures used for their computation are given in Chapter V.

An instantiation of a codec based on the design of Figure IV-6 can be seen in Figure IV-7; the upper part of the figure presents the encoder design. The coding process resembles the classical closed-loop coding structure [48]; thus, when coding in the intra-frame mode, the current frame is wavelet decomposed by the DWT module first; subsequently, each resolution is compressed with the SBC module (intra-frame) by a block-based intra-band coder, and context-based entropy coded by the EC module. Each resolution of this intra-frame is decompressed at a base-quality layer (SBD module) and the reconstructed DWT decomposition is used by the complete-to-overcomplete DWT (CODWT) module. This module operates in a subband-by-subband manner and constructs the subbands of the overcomplete transform of each level from the critically-sampled decomposition of that level. As explained before, the overcomplete transform contains all the information that is missing from the critically-sampled pyramid due to the subsampling operations performed at every level. For the complete-to-overcomplete DWT, the conventional low-band shift method can be used [27], or more advanced techniques that provide fast-calculation algorithms with identical algorithmic performance can be employed [36] [39]; these topics are elaborated in Chapter V.

In the inter-frame coding mode, motion compensation is performed in the critically-sampled DWT and the predicted frame is subsequently subtracted from the current frame (resolution-by-resolution), as shown in Figure IV-7. The result (error frame) of each resolution is encoded in the same manner as explained for the intra-frames. Subsequently, each decomposition level of the compressed error frame is decompressed by the SBD module to the base-quality layer and is added to the previously predicted frame of that resolution. The result is used as the new reference frame and is sent to the CODWT module. In this way, successive MCP within the closed-loop is realized. This scheme is readily generalized to include any combination of backward and forward prediction that is typically used in coding standards.

The decoder operates in a similar manner as noticed from the lower part of Figure IV-7. This shows that with the proposed video codec architecture, we decode transform-domain intra-frames and error-frames.

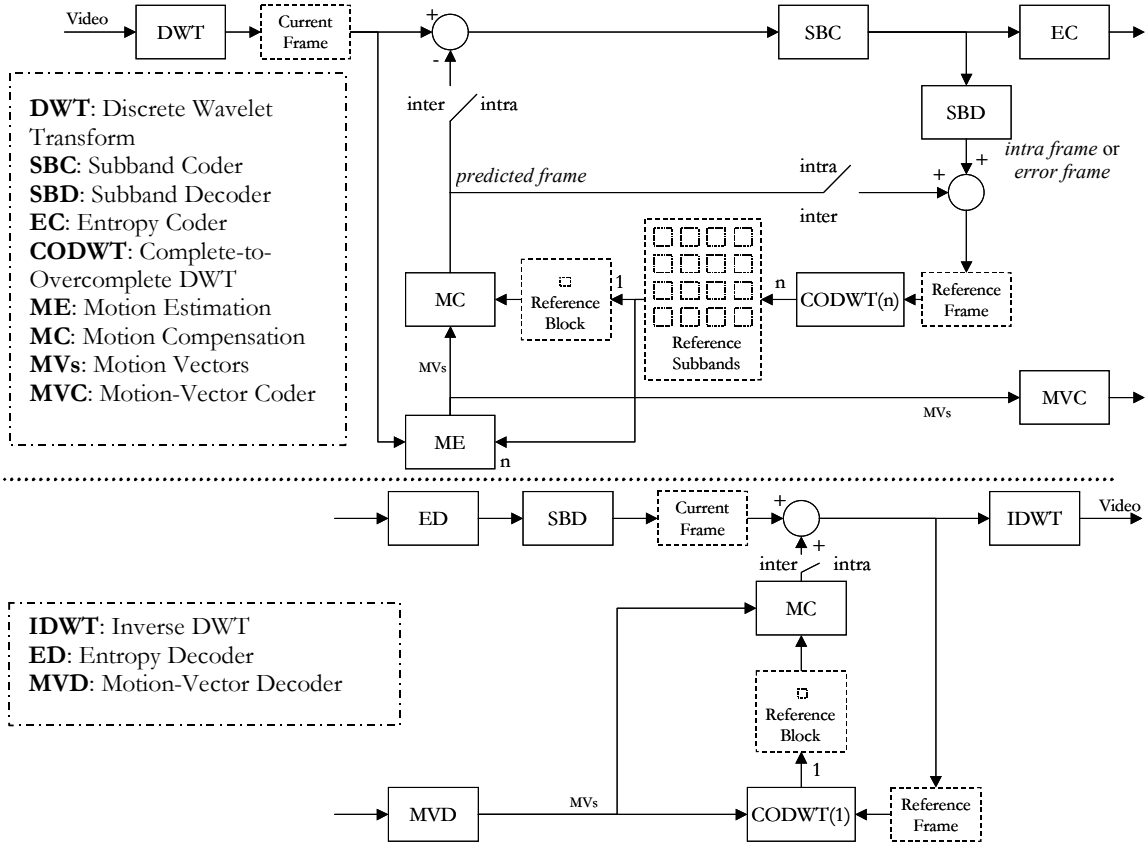


Figure IV-7. The proposed architecture of an in-band closed-loop wavelet video codec. A localized complete-to-overcomplete DWT is presented, where CODWT(n) refers to the construction of the n-th subband of the ODWT of each resolution level.

Due to the separate application of the ME/MC procedure at every resolution level, the decoding can stop after a specific resolution level has been received, thus accommodating a variety of decoded resolutions for the video clients. In addition, the decoding can stop at any desired quality level (corresponding to a rate larger than the base-layer bitrate), since the coding of the error-frames produced for every resolution level is embedded; as a result, a variety of transmission-bandwidths or picture-quality levels can be supported on the fly. Finally, by selecting specific frames to be used as references for the ME/MC procedures, the decoder can skip any of the remaining coded frames and achieve any desired frame-rate for the decoded video reproduction without drifting from the encoder.

4.3 Low-redundancy Overcomplete Discrete Wavelet Transforms for In-band Motion Compensated Prediction

As shown in the previous section, the main difference of the presented in-band predictive video coding architecture versus conventional predictive video coding lays in the use of the ODWT for IB-

ME and IB-MCP. In this section we are concerned with the optimality of the ODWT for this purpose.

Our initial motivation on using a system that is based on conventional critically-sampled biorthogonal discrete wavelet transforms such as the 9/7 filter-bank is the good coding efficiency obtained via the use of such transforms in still image coding [3]. Moreover, as shown before, the use of complete-to-overcomplete transforms enables the produced error-frames to be critically-sampled and an equivalent MCP system to the conventional SD-MCP is created in the wavelet domain. However, although the conventional ODWT is a shift-invariant transform that provides zero MCP error for the cases where the spatial-domain MCP error is zero, it may be possible to utilize low-redundancy, near shift-invariant, transforms with competitive prediction accuracy for IB-MCP and yet a reduced calculation overhead. Recently there has been an abundance of research on near shift-invariant transforms [49] [50] [51] [52] [53] with limited redundancy. In some cases, it is demonstrated that near shift-invariance can be attained with a redundancy of four in comparison to the two-dimensional multi-level critically-sampled DWT. This may have important implementation advantages since, independent of the number of decomposition levels (k), such low-redundancy transforms will produce a total of $(2M)\times(2N)$ wavelet coefficients for an $M \times N$ image during the ME stage, while the conventional ODWT will produce a total of $(2k \cdot M)\times(2k \cdot N)$ coefficients. Consequently, for $k > 1$, the filtering operations for the production of the reference-frame overcomplete representation as well as the SAD operations required for the multilevel IB-ME can be substantially-reduced with a low-redundancy transform. Moreover, the reduced number of ODWT-phases in the low-redundancy transforms indicates that one can also potentially reduce the required overhead for the wavelet-domain motion-vector information.

In order to evaluate a video coding system utilizing such a low-redundancy transform, we incorporated in the proposed closed-loop video coding system of Section 4.2 one of the most representative members found in the literature, i.e. the Q-shift dual-tree complex wavelet transform (DTCWT) [50]. The following subsection introduces the concepts behind the design of the DTCWT; Subsection 0 presents the proposed video coding system with IB-MCP using the DTCWT. The summary of the work presented in this section has been published in [54].

4.3.1 The Dual-Tree Complex Wavelet Transform

Research work on the design of critically-sampled discrete wavelet transforms revealed that one cannot provide FIR filters with perfect reconstruction properties and good frequency characteristics which are (near) shift-invariant [50]. After abandoning this goal, Kingsbury [50] was among the first to observe that approximate shift-invariance could be obtained by doubling the sampling rate at each level of the DWT decomposition. For one-dimensional signals, this corresponds to the use of two critically-sampled DWT pyramids. In two dimensions, four pyramids correspond to this observation. It was shown that the wavelet coefficients of each pyramid can be interpreted as the real and imaginary part of a wavelet decomposition with a complex-valued filter-bank [50] [52]. Hence, the transform was initially termed as the dual-tree complex wavelet transform [50]. However, as noted in [50], this separation into real and imaginary components is arbitrary, since all filter-banks used in the DTCWT consist of real-valued FIR filters.

A practical observation of Kingsbury's work [50], which is of crucial importance, is that a complex-valued filter-bank creates orthogonality in the phase of the wavelet coefficients of the two pyramids. This corresponds to the ODWT-phase components of the samples of both pyramids being equally-spaced among the phase-components of the (fully-redundant) ODWT. This is illustrated in the pictorial example of Figure IV-8. There, at each decomposition level, the conventional ODWT samples are equally-spaced (circles). Since the sampling-rate of each subband A_i^l, D_i^l (with $1 \le l \le 3$) is reduced by 2^l in comparison to the sampling rate of the input signal X, the number of ODWT-phases i is increased dyadically, i.e. $0 \le i \le 2^l - 1$ for each decomposition level l. On the other hand, in the two-pyramid case of the DTCWT (x-marks), although the sampling rate of each subband is also reduced by 2^l, there are only two phase components. As shown by Figure IV-8, these phases are also equally-spaced at each decomposition level by producing wavelet coefficients that correspond to half ODWT-phase intervals for $l > 1$. Notice that, for $l = 1$, both transforms produce ODWT-phase components at the same locations. Moreover, in this case, both transforms have the same redundancy.

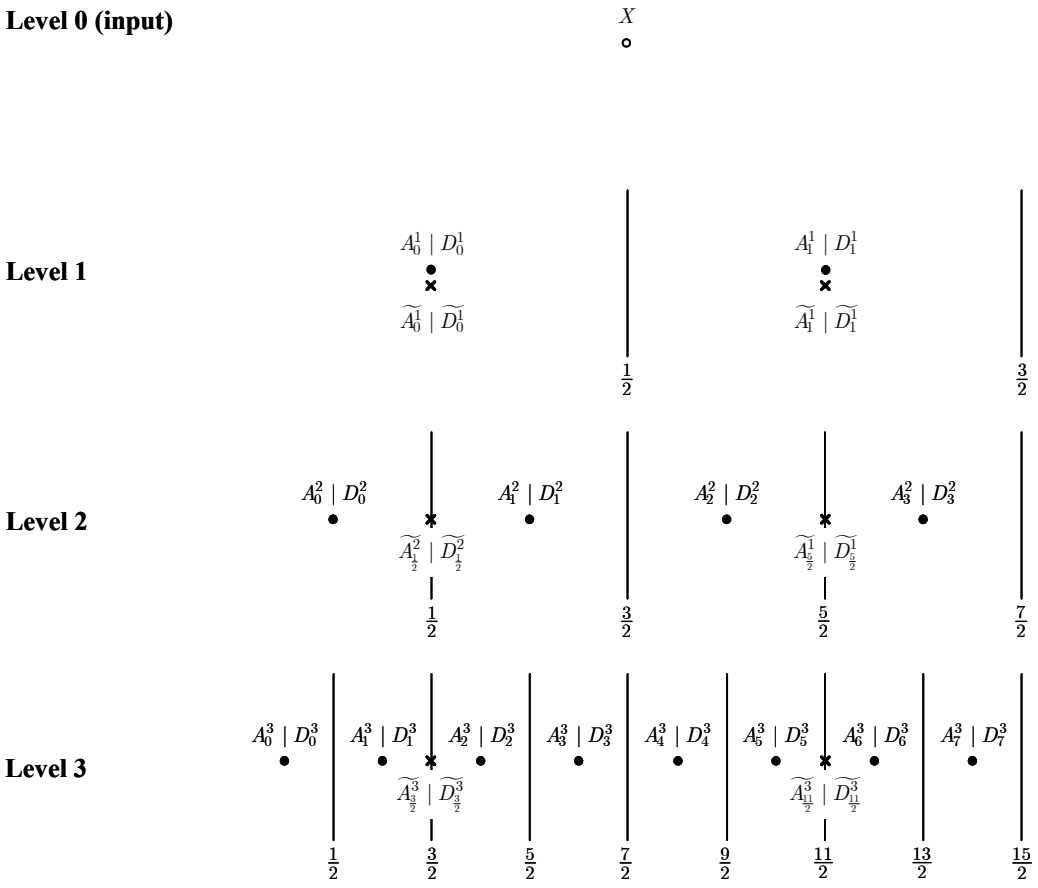


Figure IV-8. An example of equal sampling space between the in-band ODWT-phases of: (a) the conventional ODWT (circles), where integer ODWT-phase components are produced; (b) the dual-tree CWT (x-marks), where half ODWT-phase components are produced. The dual tree has limited redundancy (only by a factor of 2 for all levels versus the conventional DWT). For illustration purposes, the half-phase intervals of each level are indicated by the vertical lines.

In [50], a practical design is given for FIR filters that produce the ODWT-phase components that correspond to the sampling intervals of the DTCWT as shown in Figure IV-8. As demonstrated by this pictorial example, for $l = 1$, one can opt for the use of any ODWT based on conventional filter-banks. In fact, the $9/7$ filter-bank is recommended as a good practical example for the filters of $l = 1$ [50], due to its good performance in coding applications. For decomposition levels higher than one, a new family of filters-banks was derived in [50], which produces half ODWT-phase components as indicated by Figure IV-8. For levels higher than one, the design procedure produced even-length filters-pairs which are not strictly linear phase. Instead, since the target is to construct half ODWT-phase components, they correspond to an orthogonal basis and are designed to have a group delay of approximately $\frac{1}{4}$ sample (since the sampling rate is halved at the output of the filter-bank due to downsampling by two). As a result, this transform was additionally termed as Q-shift DTCWT [50], to separate it from other, similar, approaches. The resulting wavelet and scaling functions of the Q-shift DTCWT can be seen in Figure IV-9.

From the filter-bank family designed for the Q-shift DTCWT, we used the "Q-shift-06" filter-bank [50], which is an interesting case since it has good shift-invariance properties with only 6 non-zero taps, given in Table IV-I. The DTCWT decomposition structure is shown in Figure IV-10. We use the notations of [50] for consistency, were subscript 0 and 1 indicate low-pass and high-pass filters, respectively, and subscripts a and b distinguish the filters of each decomposition of the DTCWT. As mentioned before, for the filters of level one, the $9/7$ filter-bank is used. Moreover, for levels two and higher, the filters are: $H_{00a}(z) = z^{-1}H_L(z^{-1})$, $H_{01a}(z) = H_L(-z)$, $H_{00b}(z) = H_L(z)$, $H_{01b}(z) = z^{-1}H_L(-z^{-1})$. The reconstruction filters are just the time reverses (i.e. trees a and b are swapped.

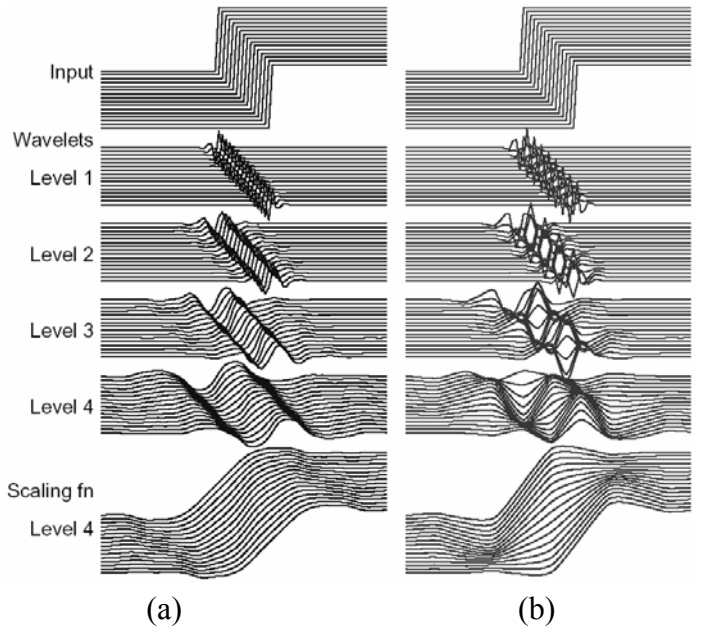


Figure IV-9 (Taken from [50]). Wavelet and scaling function components for 16 shifts of the step function (top), for: (a) Q-shift DTCWT; (b) the critically-sampled DWT. The good shift invariance of the DTCWT is demonstrated by the fact that all the corresponding wavelets (and scaling functions) are similar in shape for the various translations of the input step function.

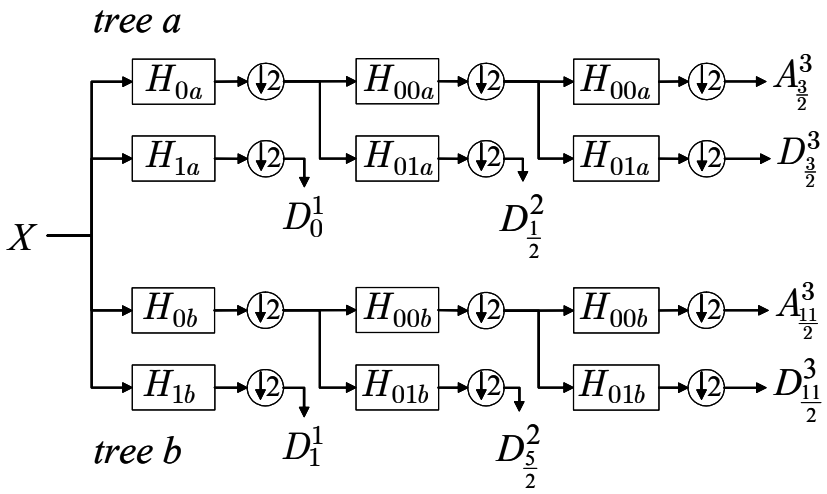


Figure IV-10. The dual tree complex wavelet transform.

Order in Z	$H_L(z)$
4	0.03516383657149
3	0
2	-0.08832942445107
1	0.23389032060724
0	0.76027236906613
-1	0.58751829772356
-2	0
-3	-0.11430183714425

Table IV-I. Filter taps used for the orthogonal filter-bank of the Q-shift DTCWT [50].

4.3.2 Closed-loop Video Coding with the Single-to-Dual Tree Discrete Wavelet Transform

There have been early attempts of utilizing the dual-tree complex wavelet transform for non-scalable video coding with some degree of success [55]. However, our work differentiates significantly from [55] because the DTCWT is utilized within the framework of Figure IV-7. As a result, in accordance to the design of Figure IV-7, the single-to-dual tree complex wavelet transform (SDTCWT) is performed in the CODWT module [54] and the produced error frames remain critically sampled.

The main difficulty in the SDTCWT process lays in the border-extension policy in order to have perfect reconstruction. In the original DTCWT implementation of [50], the dual tree is constructed starting from the input signal X. As a result, the border extension of every tree is performed using coefficients from the alternate one, as seen in Figure IV-11. This results in a smooth extension and prefect reconstruction is guaranteed. This extension however is not feasible if both trees are not available, and this is the case during the inverse transform performed in the SDTCWT process. For the first decomposition level, the problem can be easily solved by using classical point-symmetric

mirroring in the borders. However, the situation is more complicated for levels beyond one; there, typically, perfect reconstruction is guaranteed only for periodic extensions [56], since the filters of each single-tree wavelet transform correspond to an orthogonal filter-bank. Nevertheless, it is well known that this type of extension creates border distortions from the signal wrapping, which become intensified when the decomposition levels increase [57]. This effect is catastrophic if motion estimation is sought in the transform domain, as is the case in our application.

A potential solution for smooth border extension for orthogonal filter-banks is proposed in [57]. This approach requires the use of spline extrapolation techniques to perform the border extension. However, such a computationally-expensive process undermines the complexity-reduction benefit of the DTCWT. As a result, we proposed a new and simpler border extension technique in [54]. Our solution is based on the simple observation that, since the samples of tree b are not considered available during the decomposition and reconstruction of tree a, a border extension with symmetric mirroring can be used for the decomposition, and a computationally-inexpensive convergence technique can be used for the reconstruction. This is described below in more detail.

At any decomposition level l of tree a, with $1 < l \leq k$ the convergence technique starts by performing the inverse transform to the low and high-frequency subbands of level l, using half-point symmetric mirroring. After the application of the synthesis filters of pyramid a, i.e. filters $\widetilde{H}_{00a}$, $\widetilde{H}_{01a}$ (consisting of T_{DT} taps), only T_{DT} coefficients of the low-frequency subband borders of level $l-1$ are reconstructed incorrectly. After this step, the following steps are applied iteratively:

- Apply a decomposition with filters H_{00b}, H_{01b} to the T_{DT} low-frequency coefficients of level $l-1$ of tree a. Half-point symmetric mirroring is used and two sets of $\frac{T_{DT}}{2}$ coefficients are produced. Notice that we use the analysis filters of tree b (alternate tree) for this step.

- Perform an inverse transform for level l of tree a using the time-inversed sequences of the previously-produced coefficients as the half-point symmetric extension for the borders and produce a new set of T_{DT} low-frequency coefficients of level $l-1$.

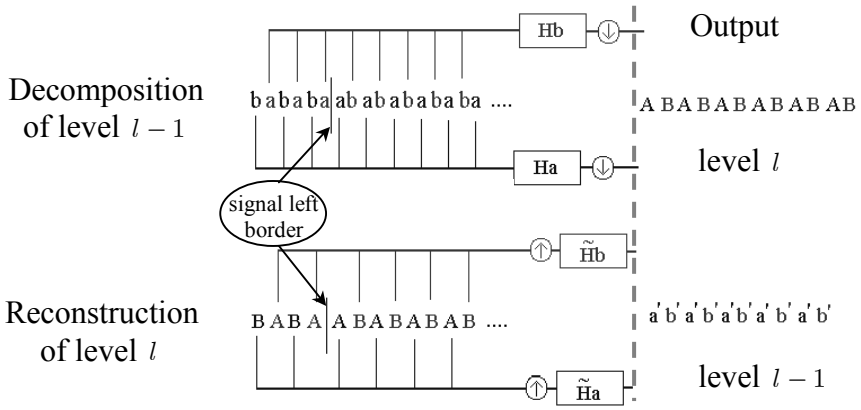


Figure IV-11. Border extension for the dual tree using mirroring with samples from the "opposite" tree [54]. This example corresponds to either the low or the high-frequency subband. The subband samples of the two trees are named 'a' and 'b' or 'A' and 'B' for the forward and inverse transforms respectively. The filters 'Ha', 'Hb' and '$\tilde{H}$a', '$\tilde{H}$b' correspond to the low- (or high-) pass filters of trees a and b for the forward and inverse transforms, respectively.

Intuitively, this algorithm is based on the following rationale: for the single-tree subbands of level l, first the inverse transform is performed and the low-frequency subband of level $l-1$ is reconstructed correctly, except of the border coefficients; subsequently, a decomposition with the analysis filters of the alternate tree is performed for these border coefficients in order to "correct" them; then the reconstruction is performed with the synthesis filters of the current tree by using the "corrected" coefficients as border extension; this process iterates so as for the border "correction" to gradually decrease the border reconstruction error of the low-frequency subband of level $l-1$ to zero. In practice, this simple technique was found to effectively reduce the total reconstruction error to values lower than 10^{-10} in less than 13 iterations using floating-point accuracy. Each iteration is only applying the analysis filters of tree b for T_{DT} times on average (since the inverse transform of tree a is in fact using the filters of the forward transform of tree b for levels $l > 2$). As a result, the computational overhead is limited to $12(T_{DT})^2$ multiplications, which is considered adequate. In addition, the symmetric mirroring policy used in the forward transform leads to a smooth extension that does not produce the border effects of the periodic extensions.

After solving the border-extension issues, the single-to-dual tree complex wavelet transform starts from a de-quantized, critically-sampled, decomposition (of k-levels) and reconstructs the input frame. To address resolution scalability, the reconstruction is parametrical to utilize high frequency coefficients from any number of decomposition levels. From the reconstructed frame, the DTCWT is constructed and the polyphase components of the produced subbands are kept in separate structures.

To quantify the shift-invariance properties of the DTCWT in our system, we simulated the SDTCWT in Matlab. The produced transform representation is (approximately) "shiftable" if the energy of all the phase components of a given subband (which consists of both trees) remains (approximately) constant when the input is shifted [49]. This property has been tested in the one-dimensional case using shifts of an impulse input, as demonstrated in Figure IV-12. In this case, different border-extension policies do not affect the result. To check the effect of the proposed border treatment, the experiments of Figure IV-13 show the energy fluctuations in the transform subbands of a random input signal, when the entire signal is translated by an integer number of pixels.

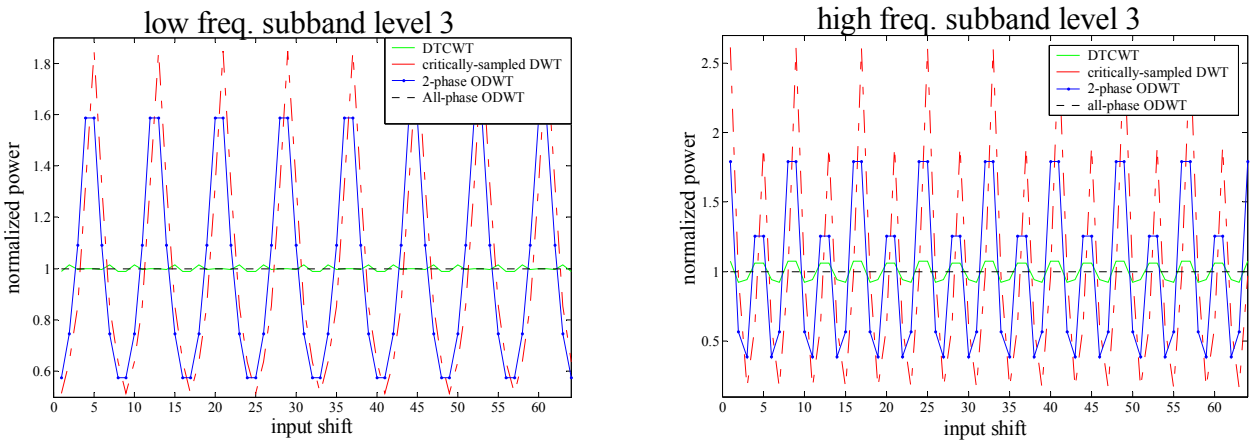


Figure IV-12. Normalized power for the subbands of level three for different shifts of an impulse. The following wavelet representations are considered: the 1-D DTCWT (9/7 filter-bank and the two orthogonal filter-banks based on the H_L filter of Table IV-I); the critically-sampled DWT (9/7 filter-bank), the ODWT using only two phases, and the all-phase ODWT (9/7 filter-bank).

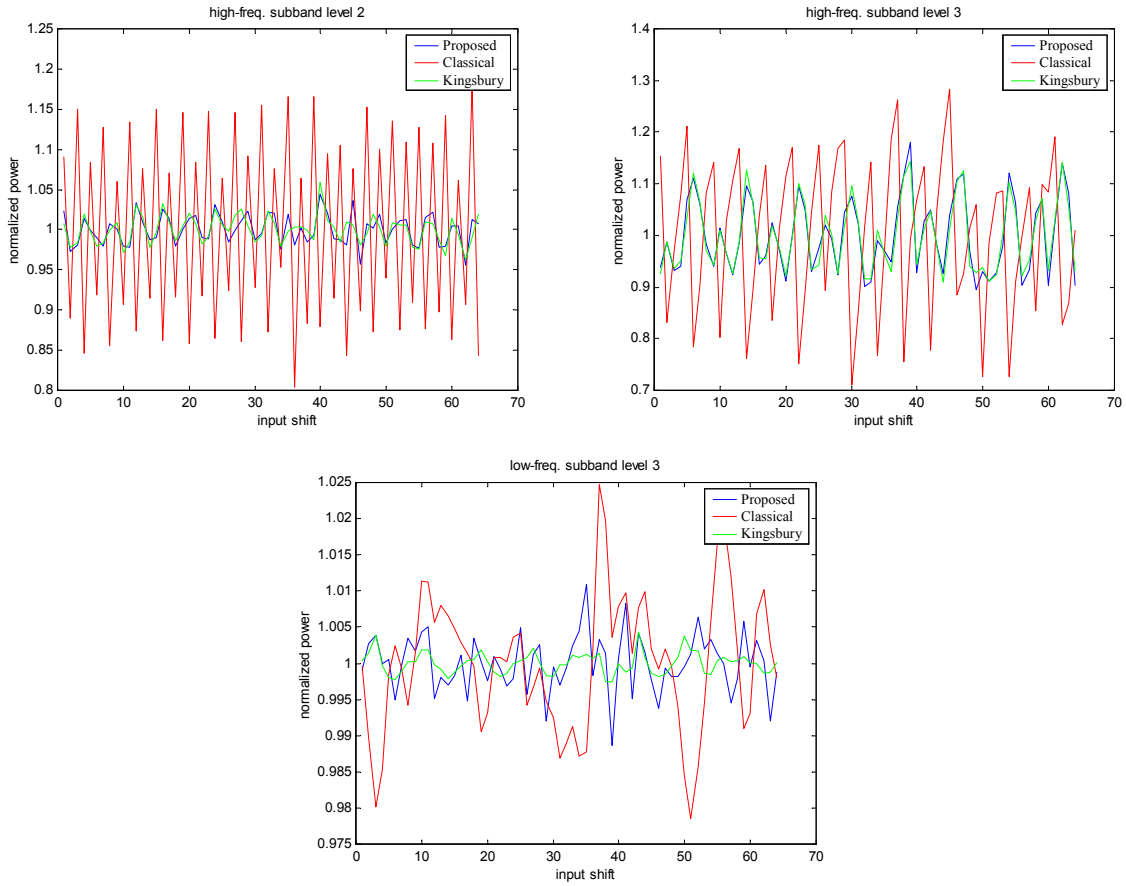


Figure IV-13. Normalized power in function of shifts of a random signal. The proposed border-extension technique yields similar power fluctuations with the border-extension proposed by Kingsbury [50], which uses additional samples from the alternate pyramid. The "classical" method uses the conventional periodic extension.

In the typical example of power fluctuations under different shifts of a one-dimensional impulse input, shown in Figure IV-12, the reduction of the power fluctuations seen in the case of DTCWT is an experimental verification of the near shift-invariance property. In addition, the two-phase ODWT (that possesses the same redundancy as the DTCWT) presents a higher fluctuation of the normalized subband power. Furthermore, for shifts of random input signals, the same fluctuation was noted on average for the dual-tree generated by the implementation of [50] and by our SDTCWT employing a different border extension. Figure IV-13 demonstrates that the proposed boundary-treatment technique, not only avoid edge artifacts caused by periodic extension, but also provides less power-fluctuations, which are found to be similar to the power fluctuations of the boundary-treatment approach used in [50].

4.4 Open-loop In-band Motion Compensated Prediction

Our investigation on the suitability of the conventional ODWT for in-band MCP led to the usage of low-redundancy overcomplete wavelet decompositions for in-band MCP (i.e. the two-phase ODWT and the DTCWT). These were proposed as alternative solutions for ODWT-based in-band video coding. In a similar manner, one significant aspect of scalable video coding in general, concerns the suitability of the closed-loop (DPCM) coding structure in the temporal domain for MCP-based video coding [58]. In fact, it has been recently proposed that, in terms of coding efficiency, an equivalent open-loop MCP structure can be as efficient as the closed-loop MCP for video coding [59]. This section extends these findings by proposing an in-band MCP video coding method within an open-loop temporal prediction structure. In particular, Subsection 4.4.1 presents the general structure of our system while Subsection 4.4.2 presents an example of the capabilities enabled by open-loop IB-MCP. Moreover, Subsection 4.4.3 presents on an algorithm to control the inherent distortion fluctuations of spatial-domain and in-band open-loop MCP.

4.4.1 Scalable Video Coding with Open-loop In-band Motion Compensated Prediction

The modification of the conventional closed-loop spatial-domain MCP structure to an open-loop architecture was originally proposed in several forms in [6, 60] [7] [8] [59] [61] [62] [63]. In essence, the unconstrained motion compensated temporal filtering (UMCTF) architecture [59] [63] represents a superset of all the efficient open-loop MCP architectures found in the literature because:

- the temporal decomposition is based on the lifting framework [43], hence it is computationally-efficient;
- perfect reconstruction is provided under arbitrary motion models [62]. For example, all the advanced prediction modes, such as arbitrary sub-pixel accuracy, multiple-reference MCP and multihypothesis MCP, proposed for closed-loop video coding can be accommodated. Moreover, in principle, UMCTF extends these modes with the update step of the temporal decomposition, thereby potentially leading to further increase in coding efficiency [64].

In practice, for a simple instantiation of UMCTF, the temporal update step can be omitted, thereby leading to an open-loop MCP-only video coding framework [63]. Such an instantiation can be seen in Figure IV-14, where an example of 8 input frames is shown. For each row $t = \{1,2,3\}$ (temporal decomposition level), each original frame L_{2k+1}^{t-1} (or A_{2k+1}^{0} for $t = 1$), with $0 \le k < 2^{-t} \cdot 8$ for the particular example, is predicted by its previous frame L_{2k}^{t-1} (or A_{2k}^{0} for $t = 1$) using block-based ME [63]. After the MCP process, it is replaced, as seen in Figure IV-14, by the produced error frame H_k^t. The low-frequency frames of each level (L_k^t) are produced by simply copying the reference frame, i.e. $L_k^t = L_{2k}^{t-1}$ (or $L_k^1 = A_{2k}^0$ for $t = 1$). This example presents an instantiation of open-loop MCP with unidirectional motion estimation and compensation using one reference frame [63].

The corresponding MCP architecture is seen in the top part of Figure IV-15. Similar to conventional MCTF, the temporal split separates the frames in even and odd sets, and the odd frames are predicted from the even frames using MCP. However, no update step is performed; hence the frames of the next temporal level correspond to the even frames of the previous level, as shown by the example of Figure IV-14.

In a similar fashion, open-loop in-band MCP, originally proposed in [10] [11], performs the temporal prediction in the wavelet domain, as seen in the bottom part of Figure IV-15. The spatial DWT is first applied to all input frames thereby generating subbands $\mathcal{T}_U^l A_\tau$, with $1 \le l \le k$, $U = \{LL, LH, HL, HH\}$ and $\tau = \{2t, 2t+1\}$. All the input frames of each resolution level l are subsequently split into even and odd sets and in-band MCP is performed with the use of the ODWT $\mathcal{S}_U^l \mathcal{T}_U^l A_{2t}$.

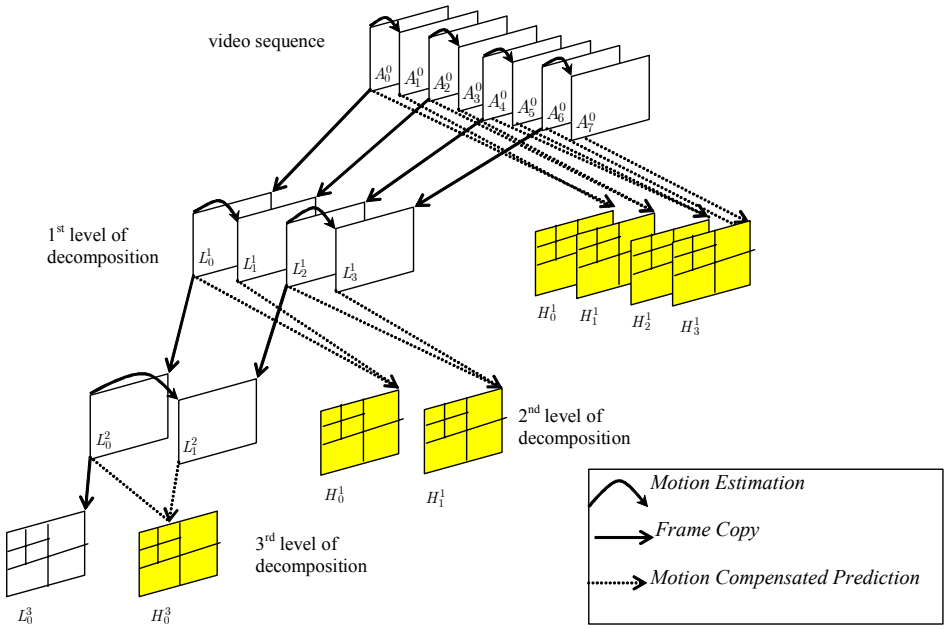


Figure IV-14. An example of open-loop IB-MCP with unidirectional motion estimation.

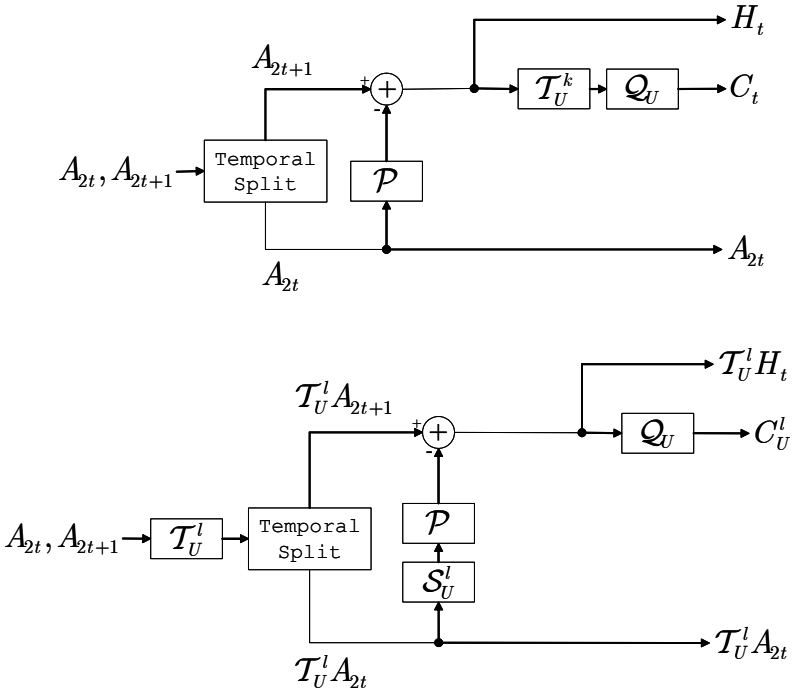


Figure IV-15. Open loop video coding structures. Top: Spatial-domain MCP. Bottom: In-band MCP for each resolution level l, $1 \le l \le k$.

In total, for each wavelet coefficient $\mathcal{T}_U^l A_{2t+1}[m,n]$, open-loop in-band MCP can be performed as:

$$\mathcal{T}_U^l H_t[m,n] = \mathcal{T}_U^l A_{2t+1}[m,n] - \mathcal{I}_{\left(i_{U,m}^{\mathcal{F}_1^l(2t)}, i_{U,n}^{\mathcal{F}_1^l(2t)}\right)} \mathcal{S}_{U,\left(\left|p_{U,m}^{\mathcal{F}_1^l(2t)}\right|, \left|p_{U,n}^{\mathcal{F}_1^l(2t)}\right|\right)}^{l} \mathcal{T}_U^l A_{2t}[m - d_{U,m}^{\mathcal{F}_1^l(2t)}, n - d_{U,n}^{\mathcal{F}_1^l(2t)}]. \quad (4.25)$$

where the same notations are used as in (4.23).

As illustrated by the practical instantiations of UMCTF given in [59] [63], open-loop MCP has two significant advantages over closed-loop MCP presented before.

The first advantage evolves around the enhanced bitrate scalability properties. It was shown experimentally (e.g. [59] [63]) that scalable video coding with open-loop MCP produces an embedded bitstream out of which subsets can be extracted to match a certain bitrate. The experimental rate-distortion performance of open-loop MCP appears to be comparable to optimized non-scalable closed-loop video coding [59]. In addition, although video coders based on closed-loop MCP can produce an embedded bitstream, due to the closed-loop temporal prediction, the bitrate adjustment is limited to rates that are higher than the bitrate that corresponds to the baselayer prediction. Moreover, while closed-loop prediction typically uses the baselayer information for MCP to avoid drifting from the encoder [65] [66], open-loop MCP-based schemes successively-improve the reference frames with increases in bitrate. As a result, for bitrate-scalable video coding, better prediction efficiency is expected from open-loop video coding architectures [58].

The second advantage of open-loop MCP stems from the different prediction structure of UMCTF versus the conventional closed-loop MCP. Due to the use of a temporal transform, a multiresolution decomposition is performed in the temporal direction. As a result, by skipping a certain number of temporal levels, one can obtain a decoded sequence with successively-reduced frame-rates. The original UMCTF proposal [63] demonstrates that, in principle, a large combination of decoded frame-rates can be obtained by selectively-decoding information from one compressed bitstream. Although temporal scalability is also present in conventional closed-loop prediction, e.g. with the use of B-frames in MPEG coding [67], such techniques offer limited choices for temporal scalability versus open-loop MCP [58].

As seen from Figure IV-15, open-loop in-band MCP follows the same temporal prediction structure albeit in the wavelet domain. As a result, the two advantages presented before are also valid for open-loop IB-MCP versus the conventional closed-loop IB-MCP. However, in comparison to spatial-domain open-loop MCP, in-band MCP inherits also the increased flexibility of in-band architectures across different spatial resolutions. To illustrate this, Figure IV-16 presents a generic video coding architecture that involves open-loop in-band MCP. As shown pictorially, each resolution can be temporally predicted with a different structure and the subsequently-produced motion vectors and texture information can be compressed using an embedded coder. This generic spatio-temporal decomposition allows for different temporal prediction structures for each spatial resolution level. Moreover, different motion estimation accuracies and different coding methods can be employed for low- and high-resolution representation of the input video content. These may be desired features for several applications. As an example, Subsection 4.4.2 presents a practical system that is based on the architecture of Figure IV-15 and can be backward compatible to MPEG video coders for the half-resolution video, due to the use of closed-loop MCP coding for the low-frequency subbands of the DWT.

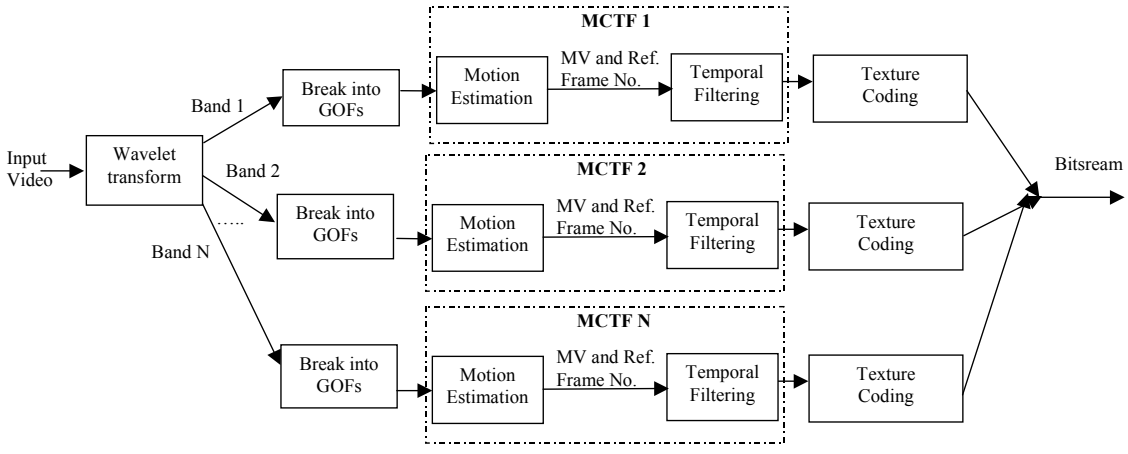


Figure IV-16. A generic architecture for multi-resolution in-band MCTF. After the spatial decomposition, the produced subbands are organized into groups of frames (GOFs) and are temporally decorrelated with a different temporal decomposition structure.

4.4.2 Test Case: Open-loop In-band Motion-compensated Prediction with Standard-compliant Base-layer

In this subsection we exploit the features of open-loop IB-MCP to investigate the possibility of backward compatibility of open-loop coding systems with existing closed-loop (MPEG) video coding systems.

The basic modification imposed in the open-loop IB-MCP consists of replacing the MCP-based wavelet video coding of the low-frequency bands with a standard-compliant motion-compensated DCT (MC-DCT) predictive scheme [68]. In this manner we obtain a base layer for low-resolution that is compliant to standardized MPEG-video while retaining the treatment of the residual low-frequency signal and the high-frequency bands within the MCTF-based wavelet coding. Our test-bed experimental evaluation [68] shows that, while spatial and temporal scalability are easily obtained in this manner without additional cost in comparison to the non-scalable equivalent, we observe a loss for bitrate scalability, similar to what is found in the relevant literature [69]. To overcome this disadvantage, we also present in this subsection a new multi-layer temporal prediction structure for the base-layer coding. This structure, although not fully standard-compliant, offers efficient bitrate scalability for the proposed scheme [68].

The architecture of a coder employing a standard-compliant base-layer for the low-frequency bands and MCTF coding for the high-frequency bands can be derived as a particular instantiation of the generic architecture of Figure IV-16. This is demonstrated in detail in Figure IV-17.

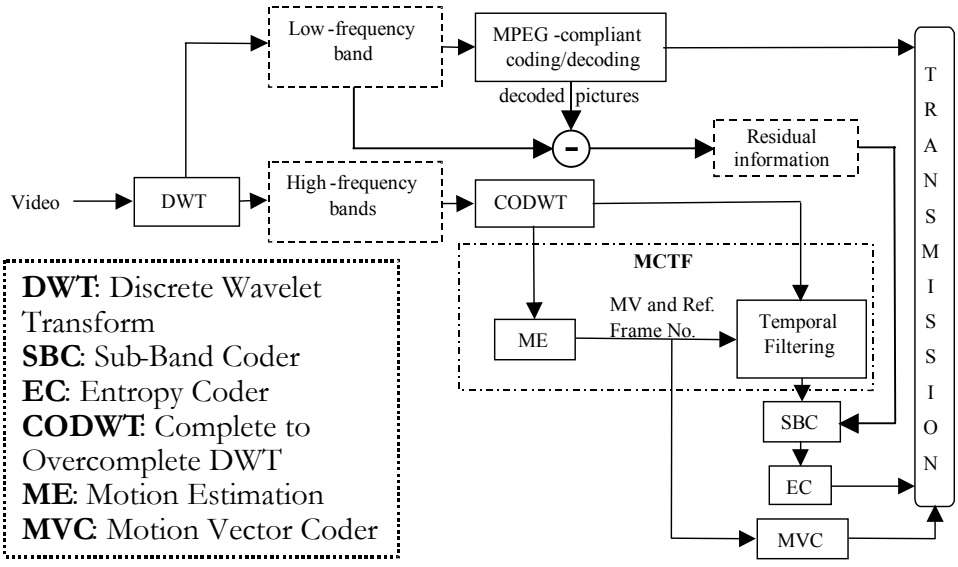


Figure IV-17. The modified open-loop IB-MCP encoding architecture with standard-compliant base-layer.

The low-frequency subband is scaled down and quantized to fit the dynamic range of the standard-compliant MC-DCT coder. After the compression process with the MC-DCT codec, the decoded pictures are scaled-up and subtracted from the original low-frequency subband content, producing the

residual low-frequency signal. This signal is coded by the subband coding technique used for the high-frequency subbands. In many cases, the decoded pictures can be generated without performing the actual decoding operation since the MC-DCT system buffers the coded references during the motion estimation routine. In total, the output bitstream consists of the standard-compliant (non-scalable) MC-DCT base-layer, and the scalable bitstream of the MCTF-based coding plus the scalable coding of the residual information of the low-frequency band. The latter can be used as an enhancement layer both for the low and high-resolution decoding. For the enhancement layer compression, any embedded coding scheme can be used. In our experiments, wavelet-based embedded coding has been used [70] [71]. However, other coding methods, such as matching pursuit coding, could also have been employed.

Despite the fact that the architecture of Figure IV-17 uses the conventional MC-DCT in the base-layer, improvements to the bitrate-scalability functionality can be envisaged. As shown in previous studies [72], for fine-grain scalability, the loss in coding performance comes from the fact that the high bitrate decoder is forced to use the low-quality references in the MCP loop of the base layer in order to avoid drift effects from successive prediction. However, this can be improved if an MCP prediction scheme based on a temporal decomposition is chosen for the base layer as well. In Figure IV-18 we illustrate such an architecture for the encoder. The figure demonstrates the encoding of a GOP with 8 frames. In fact, this figure demonstrates the design of an open-loop MCP-based encoder that uses DCT-based compression: the DCT of frame A_0^0 is quantized and entropy coded (I-frame coding –top of Figure IV-18), frame A_4^0 is predicted from frame A_0^0 and the residue is quantized and coded (middle of Figure IV-18), and so on. This coding architecture can be used in the coding scheme of Figure IV-17 as the replacement of the MPEG-compliant codec module. Decoding of the compressed error frames is also needed in this case, for the production of the residual information of the base-layer coding. As seen in Figure IV-17, this residual information is compressed using the subband coder.

For the production of the decoded base-layer video, the decoder architecture follows exactly the symmetric operation. As a result, the necessary modification on the standardized MPEG-decoder structure is the different processing order for the decoded frames, which is derived by following the processing order of Figure IV-18 in a bottom-up manner; this is shown in Figure IV-19, using MPEG notations (I-, P- and B-frames), where the superscript indicates the decoding temporal level (layer) and subscript r indicates residual error-frame information. As shown in the figure, the decoded residual information is added back to the decoded frames and the result is subsequently used as a reference. Figure IV-19 indicates that this leads to a multi-layer decoding architecture that essentially uses P and B frames within each GOP. Due to the different layers, the prediction follows a non-sequential pattern in time. In practice, a series of standard MC-DCT decoders can be used to decode the frames of each temporal layer separately (I-P3, P2-P2, and B1-B1-B1-B1) and then cross link the decoded output of the current temporal level with each lower temporal-layer reference.

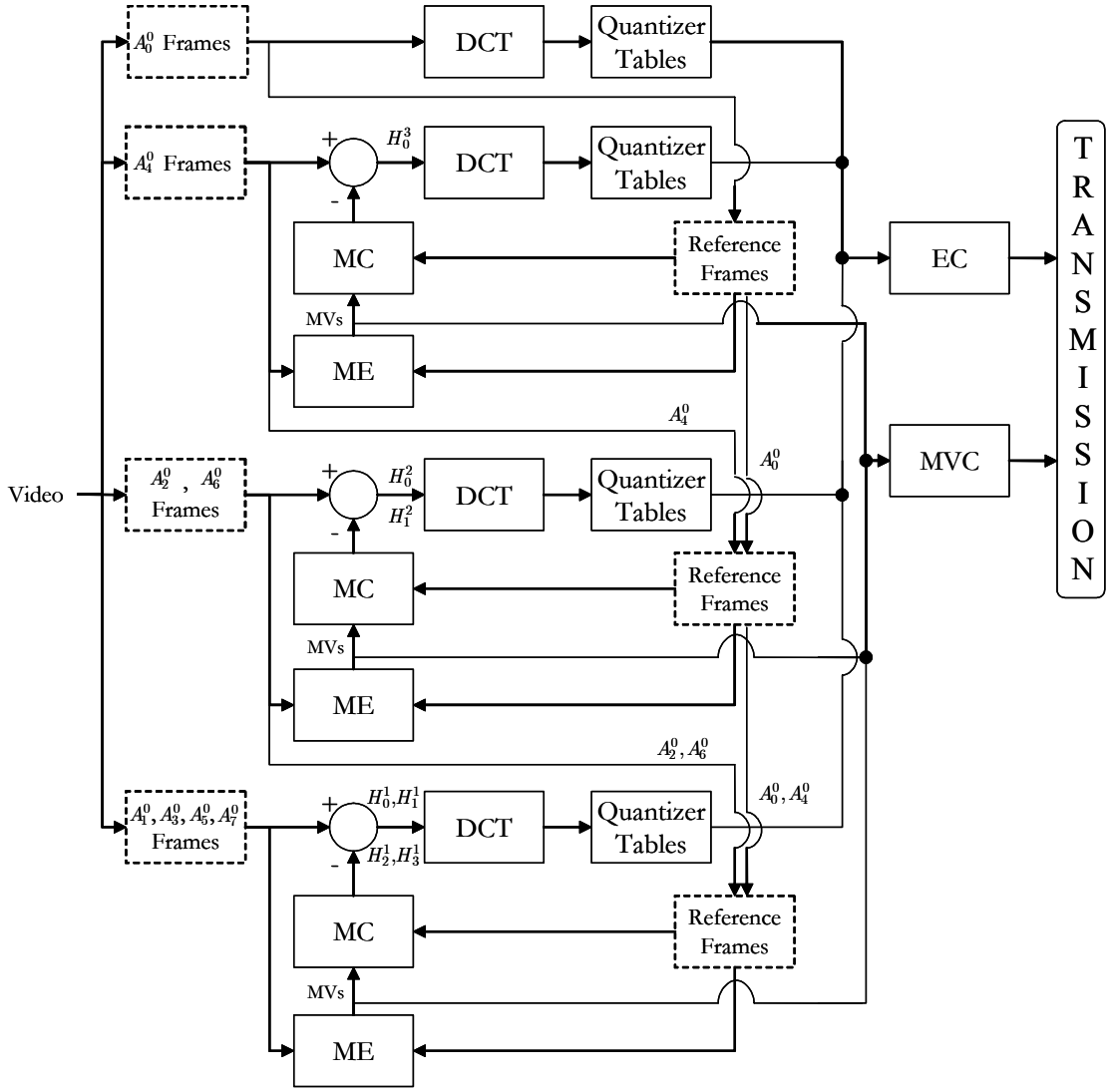


Figure IV-18. The modified multi-layer base-layer encoding.

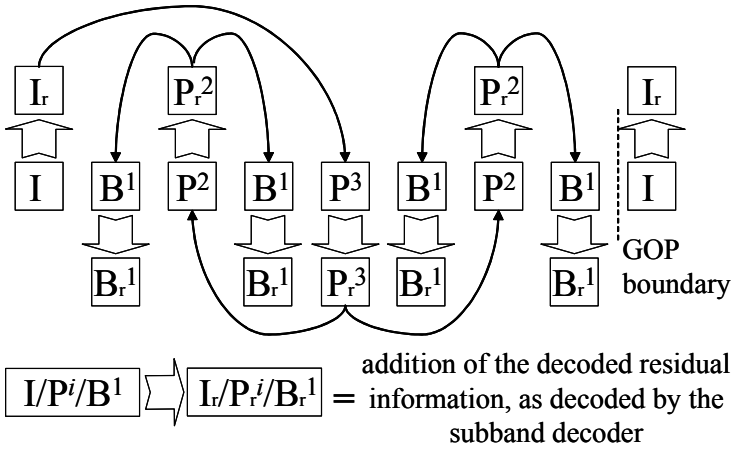


Figure IV-19. The modified decoder-side prediction structure, using MPEG video coding notations (I-, P- and B- frames).

4.4.3 Distortion Control for Open-loop Spatial-domain and In-band Motion Compensated Prediction

One problem that arises in open-loop MCP systems, such as the one of the previous subsection, is that significant variation in the quality level of the decoded video frames is observed. For illustration purposes, such an example is given in Figure IV-20, where it is shown that experimental PSNR values within each GOP (produced by a coder using open-loop MCP) can vary as much as 5 dB. These variations in the distortion can lead to bad visual quality and annoying flickering effects, especially at low bitrates. In this subsection, we investigate a control mechanism for limiting the PSNR fluctuations in decoding schemes based on open-loop spatial-domain and in-band MCP [73] [74]. We note that, although the compressed error-frames of each GOP are decompressed at (approximately) the same quality level in the experimental example of Figure IV-20, the lack of temporal motion-compensated update for the different temporal decomposition levels leads to periodically varying distortion [59]. Although the inclusion of temporal MCU may alleviate this effect, this may be undesirable due to backward compatibility issues (e.g. systems of Subsection 4.4.2) or due to the increase in the codec delay coming from the combination of MCP with MCU [59].

For the generic case of open-loop MCP with multiple references, the distortion in a decoded frame at any level in the temporal pyramid can be expressed as a function of the distortions in the reference frames at the same level. This allows for controlling the achieved PSNR fluctuations in the temporal pyramid in function of the percentage of pixels coming from each reference frame at the given point in the temporal structure. We formulate our approach for the case of bidirectional MCP, but the applicability of the proposed framework can be extended to the generalized form of UMCTF, if additional control parameters are utilized.

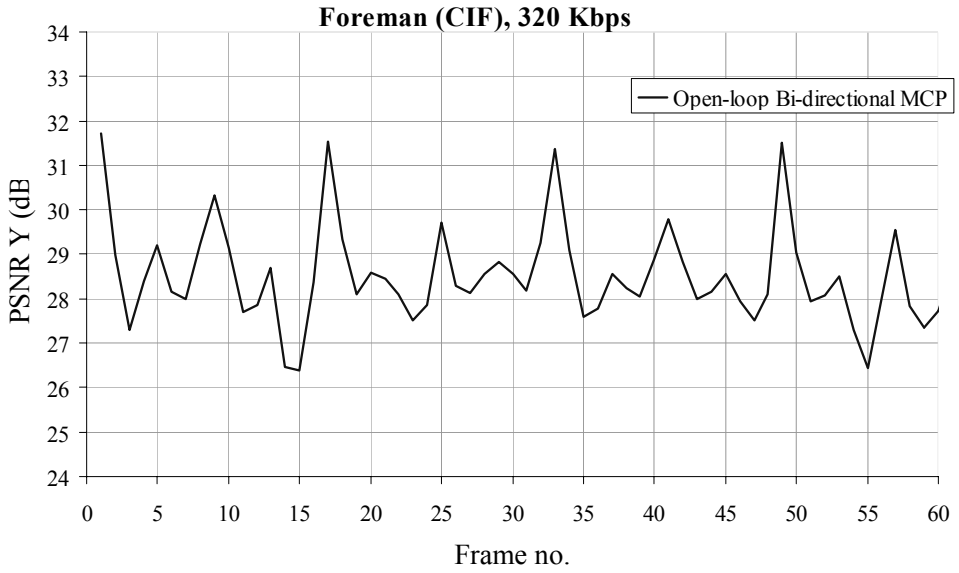


Figure IV-20. An example of frame-by-frame PSNR for the "Foreman" sequence using open-loop bidirectional MCP.

Within each GOP of the sequence, we denote the n-th reconstructed frame of temporal level t as $\overline{A}_n^t$. For simplicity we restrict the description to the performance of motion estimation and compensation (ME/MC) in the spatial domain. However, the derivations of this subsection also hold for systems that perform in-band ME/MC, as it will be shown later.

For the systems of Figure IV-15 that follow the temporal decomposition of Figure IV-14 (albeit including also bidirectional MCP), the process is reversed at the decoder, i.e. all the vertical arrows in Figure IV-14 change direction and the process occurs bottom-up, starting from the last temporal level. Thus, at the decoder we have $\overline{A}_{n\cdot 2^{t-u}}^u = \overline{A}_n^t,\ u = 0,..,t-1$, i.e. the reconstructed $\overline{A}_n^t$ frame at level t remains unchanged for all the temporal levels u smaller than t, and it eventually consists the output frame at position $n \cdot 2^t$ of the current GOP (if decoded at full frame-rate). If unidirectional or bidirectional MCP is used, the reconstruction of each $\overline{A}_{2n+1}^{t-1}$ frame at the decoder can be modelled by the system of Figure IV-21.

In this system, the inputs consist the reconstructed previous and next frame of level t of the temporal pyramid ($\overline{A}_n^t$ and $\overline{A}_{n+1}^t$ respectively) and the corresponding decoded error frame $\overline{H}_n^t$. Additionally, the motion vectors are inputs to the system of Figure IV-21 and they are separated to vectors that link the previous, the next, or a weighting of the previous and next frames ($v\left(\overline{A}_n^t\right)$, $v\left(\overline{A}_{n+1}^t\right)$ or $v\left(\overline{A}_n^t, \overline{A}_{n+1}^t\right)$) respectively. The output is the reconstruction of the current frame of the temporal pyramid at level $t-1$ (frame $\overline{A}_{2n+1}^{t-1}$).

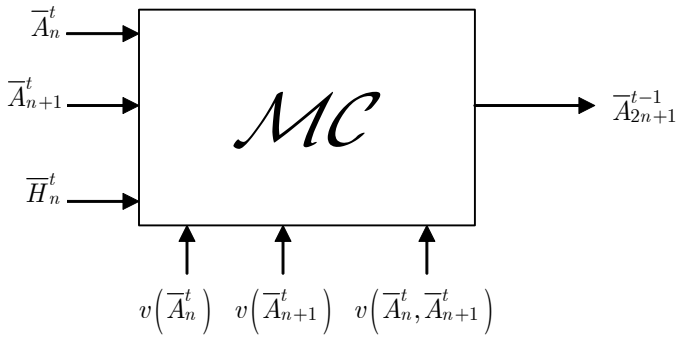


Figure IV-21. The system that performs the reconstruction of the frame $\overline{A}_{2n+1}^{t-1}$ in the temporal pyramid.

By analyzing the block-based operation of the system of Figure IV-21, for any block i of the reconstructed frame $\overline{A}_n^t$ it holds that:

$$(\overline{A}_{2n+1}^{t-1})_i = w_p(\overline{A}_n^t)_{v_p(i)} + w_m(\overline{A}_{n+1}^t)_{v_m(i)} + (\overline{H}_n^t)_i \qquad (4.26)$$

where i represents the index of the current block, $v_p(i)$, $v_m(i)$ are the motion vectors associated with this block (from the previous and next frame, respectively) and w_p, w_m are weights that change according to the MCP mode of the current block. Specifically:

- $w_p = 1$ and $w_m = 0$ for the MCP process that links the current frame with the previous one (frame $\overline{A}_n^t$),
- $w_p = 0$ and $w_m = 1$ for the MCP process that links the current frame with the next one (frame $\overline{A}_{n+1}^t$),
- $w_p = w_m = 0.5$ for the MCP process that links the current frame with both the previous and next one.

Naturally, for the case where $w_p = 1$, we have $v_m(i) = 0$ and, vice versa, for $w_m = 1$ we have $v_p(i) = 0$. An additional case exists with $w_p = w_m = 0$, in which the block is intra coded. For simplicity we omit this case from the presentation. However, it can be included in the overall description if the utilized motion estimation scheme is using selective intra coding.

For each block i, we estimate the mean square error of the output of the system of Figure IV-21 as:

$$\sum_x \sum_y \mathbf{E}\left\{ \left\| A_{2n+1}^{t-1}[x,y] - \overline{A}_{2n+1}^{t-1}[x,y] \right\|^2 \right\} = \mathbf{E}\{e^2_{A_{2n+1}^{t-1}}\}_i = \mathbf{E}\{(w_p e_{A_n^t} + w_m e_{A_{n+1}^t} + e_{H_n^t})^2\}_i \quad (4.27)$$

where (x,y) are the coordinates of each pixel belonging to block i, e_X is the random variable defining the reconstruction error of frame X, and $\mathbf{E}\{\bullet\}_i$ denotes the expectation operator calculated for each block i.

Under the assumption that no correlation exists between the decoding error of each block i of $A_n^t,\ A_{n+1}^t,\ H_n^t$, i.e. $\mathbf{E}\{e_{A_n^t} \cdot e_{A_{n+1}^t}\}_i = \mathbf{E}\{e_{A_n^t} \cdot e_{H_n^t}\}_i = \mathbf{E}\{e_{A_{n+1}^t} \cdot e_{H_n^t}\}_i = 0$, equation (4.27) becomes:

$$\mathbf{E}\{e^2_{A_{2n+1}^{t-1}}\}_i = w_p^2 \cdot \mathbf{E}\{e^2_{A_n^t}\}_i + w_m^2 \cdot \mathbf{E}\{e^2_{A_{n+1}^t}\}_i + \mathbf{E}\{e^2_{H_n^t}\}_i \quad (4.28)$$

We can introduce a control parameter a in the MCTF scheme by expressing the error of each block of the output shown in equation (4.27) as:

$$\mathbf{E}\{e^2_{A_{2n+1}^{t-1}}\}_i = (a+1) \cdot \max\left\{ \mathbf{E}\{e^2_{A_n^t}\}_i, \mathbf{E}\{e^2_{A_{n+1}^t}\}_i \right\} \quad (4.29)$$

Higher values for a indicate a larger increase in the expected distortion of the frame A_{2n+1}^{t-1} and hence a larger PSNR fluctuation in the decoded output is expected. For $a \to 0$, the PSNR behaviour is expected to be quasi-constant.

In open-loop wavelet-based compression schemes utilizing embedded coding, the utilized codec can estimate the mean-square error at the end of each integer or fractional bitplane pass through each frame [3] (e.g. when compressing the error frames $\overline{H}_n^t$ of any temporal level t). Moreover, this mean-square error can also be associated with the achieved rate in the embedded code. Hence, in order to link the derived result of (4.28) with the embedded coding process, our target is to establish a value for the mean square error of every frame $\overline{H}_n^t$ so that we expect the increase in the error of each reconstructed frame of temporal level $t-1$ to be bounded by (4.29). As a result, to make one value for the total decoding error of frame $\overline{H}_n^t$, we assume that p_p, p_m are the percentages of pixels in the

frame A_{2n+1}^{t-1} that are linked with the previous and next frames (A_n^t, A_{n+1}^t) respectively, during the ME process. It follows that $(1 - p_p - p_m)$ gives the percentage of pixels that were bidirectionally predicted from both the previous and next frames, under the assumption that no intra coding is used.

In total, by replacing equation (4.29) in (4.28) for each block i and solving for $\mathbf{E}\{e_{H_n^t}^2\}_i$, we reach the following formulations for the desired error of frame $\overline{H}_n^t$:

Case 1: If $\mathbf{E}\{e_{A_{2n+1}^{t-1}}^2\} = (a+1)\cdot\mathbf{E}\{e_{A_n^t}^2\}$:

$$\mathbf{E}\{e_{H_n^t}^2\} = (0.25p_n - 0.75p_p + a + 0.75)\cdot\mathbf{E}\{e_{A_n^t}^2\} + (0.25p_p - 0.75p_n - 0.25)\cdot\mathbf{E}\{e_{A_{n+1}^t}^2\}. \quad (4.30)$$

Case 2: If $\mathbf{E}\{e_{A_{2n+1}^{t-1}}^2\} = (a+1)\cdot\mathbf{E}\{e_{A_{n+1}^t}^2\}$:

$$\mathbf{E}\{e_{H_n^t}^2\} = (0.25p_n - 0.75p_p - 0.25)\cdot\mathbf{E}\{e_{A_n^t}^2\} + (0.25p_p - 0.75p_n + a - 0.75)\cdot\mathbf{E}\{e_{A_{n+1}^t}^2\}. \quad (4.31)$$

Using expressions (4.30), (4.31), one can establish a control-mechanism for the distortion variation, as outlined in the algorithm of Figure IV-22, in which point 2.5 in the description is not considered. If point 2.5 is included, the algorithm translates into a rate-control mechanism where the expected distortion at all the temporal levels satisfies (4.29). The proposed control scheme is applicable on the bitstream extractor of the embedded bit-stream, as long as an embedded wavelet-based compression scheme is used during the encoding, and a set of truncation points (R-D points) for each frame is generated during encoding. Additionally, for each frame, the percentage of blocks predicted from the previous and next frames has to be retained as well.

1. *During encoding*:
   1.1. Establish q rate-distortion points for each frame in the MCTF of the GOP. For each frame, keep also the percentage of the frame that was predicted from the previous and the next reference frame in the current temporal level (p_p, p_m) respectively.
2. *During the parsing stage*:
   2.1. For each GOP, establish the maximum number of temporal levels $t_{\max}$. Establish the value of a.
   2.2. For frame $A_0^{t_{\max}}$ (the remaining A-frame of the MCTF of the GOP), read the set of q distortion points $[D(e_{A_0^{t_{\max}}}^2)]_{u=1,..,q}$ and associated rates $[R(A_0^{t_{\max}})]_{u=1,..,q}$, which were produced during embedded coding of the frame.
   2.3. For each of the q distortion values $[D(e_{A_0^{t_{\max}}}^2)]_{u=1,..,q}$, and for all $t = t_{\max}, t_{\max}-1,..,1$, apply equations (4.30), (4.31) to establish the corresponding set of *expected* distortion points $[\mathbf{E}\{e_{H_n^t}^2\}]_u$ for the H_n^t frames in the GOP, with $0 \le n < 2^{t_{\max}-t}$.
   2.4. For each of the q points $[\mathbf{E}\{e_{H_n^t}^2\}]_u$, $u = 1,..,q$, identify for every frame H_n^t, $0 \le n < 2^{t_{\max}-t}$, the bitstream truncation point $\tilde{u}$ for which the distortion $[D(e_{H_n^t}^2)]_{\tilde{u}}$ is the closest to theoretical distortion $[\mathbf{E}\{e_{H_n^t}^2\}]_u$ calculated at the previous step. Keep this bitstream truncation point and also the associated rate $[R(H_n^t)]_{\tilde{u}}$.
   2.5. If rate-control is desired, scan the produced set of q distortion-rate points of frame $A_0^{t_{\max}}$ and iterate through steps 2.2–2.4, in order to match the average rate constraint.

Figure IV-22. Algorithm for (a) controlling the distortion variation (without considering 2.5), and (b) a rate-control scheme (considering 2.5).

Although the previous scheme was presented for the case of bidirectional MCP several expansions are possible. Below, we list a few ideas:

- Modify the proposed scheme for the case of in-band motion estimation and compensation, i.e. the bidirectional, k -level, IB-MCP scheme.
- Incorporate more reference frames in the proposed distortion-control framework.
- Incorporate the use of update lifting-steps during the temporal decomposition.

Concerning the first point, one can follow the same rationale as before in order to reach the conclusion of equations (4.30), (4.31): if a separate in-band MCP occurs per resolution level, then the procedure can be simply formalized separately for each spatial resolution level. In this case, one ends up with the set of equations (4.30), (4.31) with the expectation of the reconstruction-error calculated per resolution level, and also with the percentages of the blocks compensated from the previous and next frames (p_p, p_m) respectively, generated per resolution level.

Concerning the second point, longer filters can be incorporated in the proposed scheme by using more reference frames. In this case additional weights apart from w_p, w_m will be incorporated in the scheme (e.g. equations (4.30), (4.31)).

Finally, for the third point, the application of the update step modifies the reference frames during the MC process. Since several schemes are still under investigation for the efficient application of the update step (see for example [75]), this topic was not investigated. The application of the update step in the temporal decomposition approximates an orthonormal temporal decomposition; consequently the distortion fluctuations are limited. This is also practically demonstrated in the experimental section of this chapter. As a result, the tradeoffs of controlling the distortion via the proposed scheme in the case where the update lifting step is used in the temporal decomposition are less important.

4.5 In-band Motion Compensated Temporal Filtering

Open-loop motion compensated prediction can be extended to motion compensated temporal filtering with the inclusion of motion compensated update in the temporal lifting decomposition, as shown in Chapter III. In this section we present a framework for in-band motion compensated temporal filtering (IBMCTF) [12], which can be seen as an extension of the conventional spatial-domain motion compensated temporal filtering (SDMCTF). In the related literature, SDMCTF that uses wavelet-based coding techniques for the compression of the produced H and L frames, is also called as a "t+2D" transform, to emphasize the order of the application of the DWT [76] [13] [20]. Equivalently, IBMCTF corresponds to a "2D+t" decomposition structure.

As explained before, one particular aspect of the proposed in-band video coding systems is that, due to the spatial aliasing introduced by 2-D DWT, a complete-to-overcomplete discrete wavelet transform is performed. In this way aliasing-free reference frames are used during open-loop in-band motion compensated prediction and in-band motion compensated update (IB-MCU). This incurs an additional penalty in complexity versus the equivalent SDMCTF system. However, as it will be shown

in the experimental results of this chapter, IBMCTF enables improved spatial scalability versus SDMCTF. Moreover, based on a fast transform proposed in Chapter V, the computational and delay overhead introduced by the CODWT can be reduced.

4.5.1 In-band Motion Compensated Update

The application of IB-MCU is based on extending the concept of MCTF in the wavelet domain [12]. Figure IV-23 displays an instantiation of this framework. The IB-MCP step shown in Figure IV-23 is identical to that of the bottom part of Figure IV-15 and is expressed analytically by (4.25), with the right side of the equation multiplied by $\frac{1}{\sqrt{2}}$. For each decomposition level l, IB-MCU is performed by inverting the motion information for each subband and updating the reference subband of level l. If a non-zero ODWT phase $(p_{U,m}^{\mathcal{F}_1^l(2t)}, p_{U,n}^{\mathcal{F}_1^l(2t)})$ was used during IB-MCP, then a non-zero ODWT phase is taken from the error frame subbands $\mathcal{T}_U^l H_t$. This is performed by the second $\mathcal{S}_U^l$ operator of Figure IV-23, which precedes the IB-MCU operator $\mathcal{U}$.

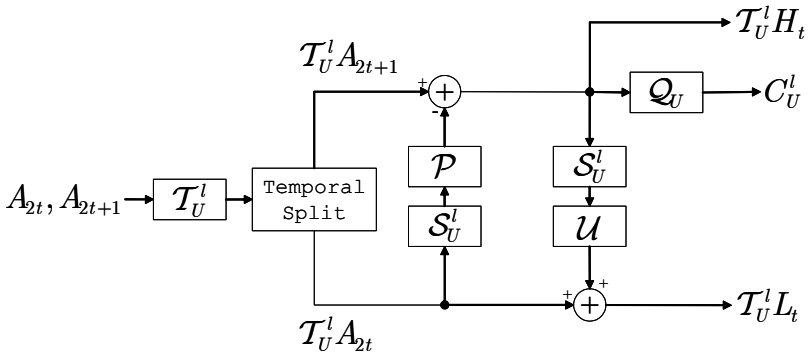


Figure IV-23. In-band motion compensated temporal filtering for each resolution level l, $1 \leq l \leq k$.

In particular, two aspects have to be treated when inverting the wavelet domain motion information generated by (4.25): inversion of an in-band motion vector $(d_{U,m}^{\mathcal{F}_1^l(2t)}, d_{U,n}^{\mathcal{F}_1^l(2t)})$ with non-zero integer ODWT phase component $(\lfloor p_{U,m}^{\mathcal{F}_1^l(2t)} \rfloor, \lfloor p_{U,n}^{\mathcal{F}_1^l(2t)} \rfloor)$, and, additionally, inversion of motion vectors pointing to the interpolated wavelet coefficients at a non-zero fractional ODWT phase $(i_{U,m}^{\mathcal{F}_1^l(2t)}, i_{U,n}^{\mathcal{F}_1^l(2t)})$ of the reference frame. For both cases, a strategy following the technique used to obtain arbitrary sub-pixel accuracy in spatial-domain MCTF is proposed [12]. A pictorial example showing the one-dimensional case of a single-level in-band prediction and update is given in Figure IV-24. As shown there, we opt to invert the in-band motion vector to the immediately-lower UDWT position in the reference frame. Notice that, as described in subsection 4.1.3, we always assume that arithmetic operations applied in ODWT phases and in-band displacement vectors are always performed in the UDWT domain, i.e. after the ODWT-to-UDWT transformation, and the result is converted back to the ODWT domain. Hence, we always assume that the motion inversion occurs in the UDWT domain. For the example of Figure IV-24 where a single-level ODWT is performed, the UDWT representation of the *LL* subband has an immediate correspondence to the ODWT domain since even ODWT-phases correspond to even UDWT positions and odd ODWT-phases correspond to odd UDWT positions.

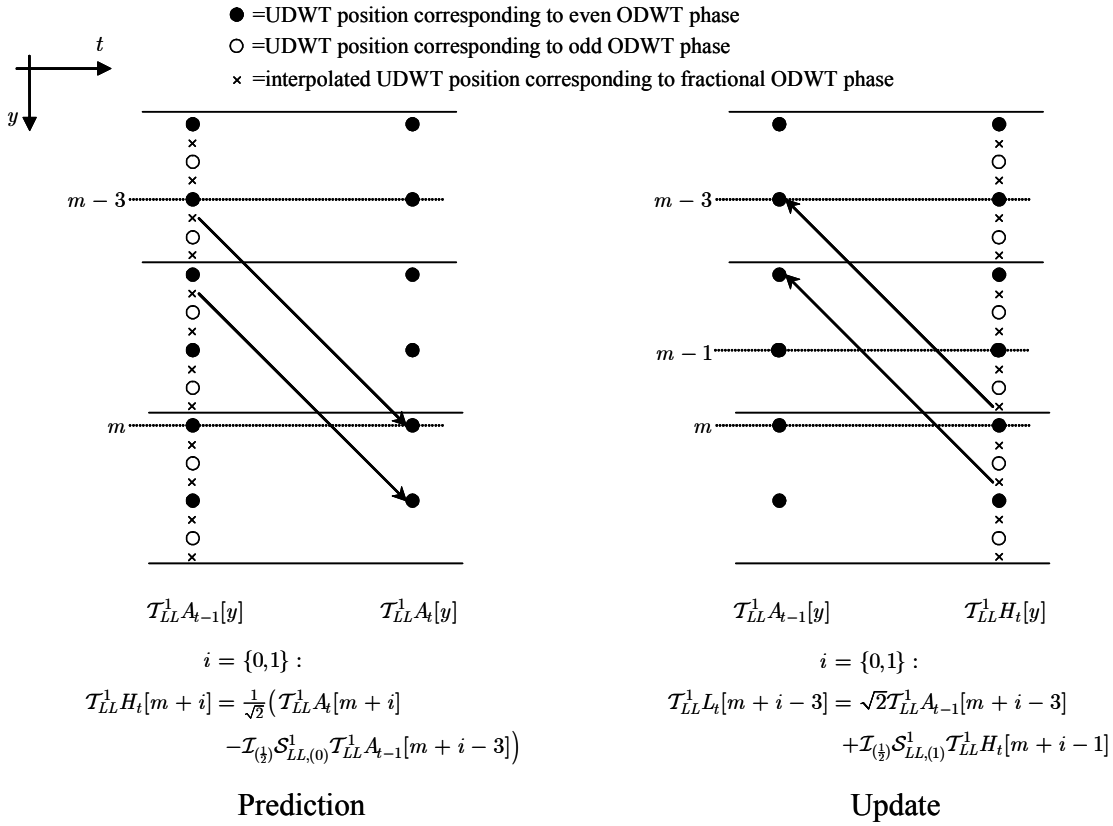


Figure IV-24. The application of in-band motion-compensated update step after the corresponding in-band prediction step. A simple one-dimensional example with two consecutive frames (LL subbands) is presented, where the in-band block-size for MCP is $B^1_m = 2$, the in-band displacement vector is $d^1_{LL,m} = 3$, the ODWT-phase component is $p^1_{LL,m} = 0.5$, and $R = 2$.

In total, the in-band MCU can be formulated as follows. It is first defined that:

$$i^{\text{res}}_{U,m} = \begin{cases} 0, & \text{if } i^{\mathcal{F}^l_1(2t)}_{U,m} = 0 \\ 1 - i^{\mathcal{F}^l_1(2t)}_{U,m}, & \text{otherwise} \end{cases}, \tag{4.32}$$

$$p'_{U,m} = \begin{cases} \left| p^{\mathcal{F}^l_1(2t)}_{U,m} \right|, & \text{if } i^{\mathcal{F}^l_1(2t)}_{U,m} = 0 \\ \left| p^{\mathcal{F}^l_1(2t)}_{U,m} \right| - 1, & \text{otherwise} \end{cases} \tag{4.33}$$

$$p^{\text{res}}_{U,m} = \begin{cases} 0, & \text{if } p'_{U,m} = 0 \\ 2^l - p'_{U,m}, & \text{otherwise} \end{cases} \tag{4.34}$$

$$d^{\text{res}}_{U,m} = \begin{cases} 0, & \text{if } p'_{U,m} = 0 \\ 1, & \text{otherwise} \end{cases}. \tag{4.35}$$

In addition, we define $i^{\text{res}}_{U,n}$, $p^{\text{res}}_{U,n}$, $d^{\text{res}}_{U,n}$ in the same successive manner. Then, for each wavelet coefficient (m,n) of subband U of temporal level l, the IB-MCU process, can be written as [12]:

$$c^l_U[m,n] = 0\,,\ \mathcal{T}^l_U Z_t[m,n] = 0 \tag{4.36}$$

$$\begin{aligned}\mathcal{T}^l_U Z_t[m - d^{\mathcal{F}^l_1(2t)}_{U,m}, n - d^{\mathcal{F}^l_1(2t)}_{U,n}] \leftarrow\ & \mathcal{T}^l_U Z_t[m - d^{\mathcal{F}^l_1(2t)}_{U,m}, n - d^{\mathcal{F}^l_1(2t)}_{U,n}] \\ & + \mathcal{I}_{(i^{\text{res}}_{U,m}, i^{\text{res}}_{U,n})} \mathcal{S}^l_{U,(p^{\text{res}}_{U,m}, p^{\text{res}}_{U,n})} \mathcal{T}^l_U H_t\left[m - d^{\text{res}}_{U,m}, n - d^{\text{res}}_{U,n}\right]\end{aligned} \tag{4.37}$$

$$c^l_U[m - d^{\mathcal{F}^l_1(2t)}_{U,m}, n - d^{\mathcal{F}^l_1(2t)}_{U,n}] \leftarrow c^l_U[m - d^{\mathcal{F}^l_1(2t)}_{U,m}, n - d^{\mathcal{F}^l_1(2t)}_{U,n}] + 1$$

$$\mathcal{T}^l_U L_t[m,n] = \sqrt{2}\mathcal{T}^l_U A_{2t}[m,n] + \tfrac{1}{\max\{c^l_U[m,n],1\}} \mathcal{T}^l_U Z_t\,[m,n] \tag{4.38}$$

where $c^l_U[m,n]$ is the connection map of subband U of level l, and $a \leftarrow b$ indicates an assignment operation, i.e. the value of variable or expression b is assigned to variable or array element a. Each of (4.36) – (4.38) is performed separately for all the subbands of each decomposition level. Similar to spatial-domain MCTF, the successive definitions of (4.32)–(4.35) perform phase inversion of the in-band motion vector of the interpolated ODWT of the error frame: first the fractional (interpolated) ODWT-phase component $(i^{\mathcal{F}^l_1(2t)}_{U,m}, i^{\mathcal{F}^l_1(2t)}_{U,n})$ is inverted to $(i^{\text{res}}_{U,m}, i^{\text{res}}_{U,n})$ and then the integer part of the ODWT-phase component $(\lfloor p^{\mathcal{F}^l_1(2t)}_{U,m} \rfloor, \lfloor p^{\mathcal{F}^l_1(2t)}_{U,n} \rfloor)$ is inverted to $(p^{\text{res}}_{U,m}, p^{\text{res}}_{U,n})$. Finally, the in-band motion vector is modified by $(d^{\text{res}}_{U,m}, d^{\text{res}}_{U,n})$.

Notice that, for the case where $\mathcal{T}^l_U A_{2t}[m,n]$ is unconnected, the proposed in-band temporal filtering of (4.36) – (4.38) is simplified to the $1/2$ filter, i.e. the temporal Haar without the update step, since $\mathcal{T}^l_U Z_t\,[m,n] = 0$.

4.5.2 Advanced In-band Motion Compensated Prediction and Update

As for the case of spatial-domain MCP and MCU, one can define in-band MCP and MCU incorporating variable block-size, multi-frame, multihypothesis ME techniques. Such an example can be realized with long temporal filters.

In general, in order to retain an orthonormal temporal decomposition, one must perform in-band MH-MCP and in-band MH-MCU via a series of lifting steps that normalize the magnitude of each wavelet coefficient of subband U and level l of the temporal-average (or temporal low-frequency) frame $\mathcal{T}^l_U L_t[m,n]$, according to the update connection map $c^l_U[m,n]$. This follows the normalization process performed in long temporal filters during the conventional SDMCTF [62]. As a result, for advanced MCTF within each subband U of each resolution level l, we can assume that a total of Λ^l_U pairs of MH-MCP and MH-MCU steps take place. Notice that, in principle, the separate application of temporal filtering across the subbands and resolutions of the spatial decomposition of the input frames allows for different temporal filters to be applied in different spatial frequencies and spatial-resolutions. The proposed formulation of this subsection enables this by specifically expressing the temporal lifting steps performed in the subbands of each decomposition level of the spatial transform.

Similar to conventional lifting [43], the first temporal lifting step for all subbands of all resolutions is the trivial polyphase separation, albeit in the temporal direction, i.e.:

$$\mathcal{T}_U^l L_t^0[m,n] = \mathcal{T}_U^l A_{2t}[m,n] \tag{4.39}$$

$$\mathcal{T}_U^l H_t^0[m,n] = \mathcal{T}_U^l A_{2t+1}[m,n] \tag{4.40}$$

For each subband of each resolution level, each subsequent pair of MH-MCP and MH-MCU steps λ_U^l, with $1 \le \lambda_U^l \le \Lambda_U^l$, is performed according to the following procedure.

Firstly, the MH-MCP that corresponds to step λ_U^l utilizes subbands $\mathcal{T}_U^l H_q^{\lambda-1}$ with temporal lifting coefficients $\alpha_{U,q}^l$ and subbands $\mathcal{T}_U^l A_t^{\lambda-1}$; the set of permissible values for q depends on the specific lifting dependencies of the specific subband U and spatial decomposition level l; without loss of generality, we assume that q is bounded by $t_p^{\text{init}}(\lambda_U^l)$ and $t_p^{\text{end}}(\lambda_U^l)$ around the time instant t (and does not include t). For this case, MH-MCP for each coefficient (m,n) of each subband U of each resolution level k can be expressed as:

$$\begin{aligned}\mathcal{T}_U^l H_t^{\lambda_U^l}[m,n] = \mathcal{T}_U^l L_t^{\lambda_U^l - 1}[m,n] - \sum_{q=t-t_p^{\text{init}}(\lambda_U^l)}^{t-1} \left(w_{U,q}^l[m,n] \cdot \alpha_{U,q}^l \cdot \mathcal{S}^l_{\left(p_{U,m}^{\mathcal{F}_{t-q}^l(q)}, p_{U,n}^{\mathcal{F}_{t-q}^l(q)}\right)} \mathcal{T}_U^l H_q^{\lambda_U^l - 1}[m - d_{U,m}^{\mathcal{F}_{t-q}^l(q)}, n - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}] \right) \\ - \sum_{q=t+1}^{t+t_p^{\text{end}}(\lambda_U^l)} \left(w_{U,q}^l[m,n] \cdot \alpha_{U,q}^l \cdot \mathcal{S}^l_{\left(p_{U,m}^{\mathcal{B}_{q-t}^l(q)}, p_{U,n}^{\mathcal{B}_{q-t}^l(q)}\right)} \mathcal{T}_U^l H_q^{\lambda_U^l - 1}[m - d_{U,m}^{\mathcal{B}_{q-t}^l(q)}, n - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}] \right)\end{aligned} \tag{4.41}$$

Note that, for each subband U of each spatial decomposition level l, the parameters $t_p^{\text{init}}(\lambda_U^l)$, $t_p^{\text{end}}(\lambda_U^l)$, and $\alpha_{U,t-t_p^{\text{init}}(\lambda_U^l)}^l, \ldots, \alpha_{U,-1}^l, \alpha_{U,1}^l, \ldots, \alpha_{U,t+t_p^{\text{end}}(\lambda_U^l)}^l$, are directly taken from the lifting factorization of the chosen filter-bank [43], while $w_{U,q}^l[m,n]$ and $(d_{U,m}^{\mathcal{F}_{t-q}^l(q)}, d_{U,n}^{\mathcal{F}_{t-q}^l(q)})$ or $(d_{U,m}^{\mathcal{B}_{q-t}^l(q)}, d_n^{\mathcal{B}_{q-t}^l(q)})$ are produced by the utilized in-band multihypothesis motion estimation algorithm. For simplicity in notations, we did not include the interpolation operation in (4.41).

For the corresponding MH-MCU, for each subband U of each decomposition level l, the update connection map, c_U^l, and the frame containing the motion inversion information, $\mathcal{T}_U^l Z_t$, are first initialized as:

$$c_U^l[m,n] = 0\,,\ \mathcal{T}_U^l Z_t[m,n] = 0 \tag{4.42}$$

Subsequently, MH-MCU utilizes frames $\mathcal{T}_U^l H_q^{\lambda_U^l}$ with lifting coefficients $\beta_{U,q}^l$ and frame $\mathcal{T}_U^l L_t^{\lambda_U^l - 1}$. Similarly as before, the set of permissible values for q depends on the specific lifting dependencies for the temporal filtering of the specific subband and spatial decomposition level; without loss of generality, we assume that q is bounded by $t_u^{\text{init}}(\lambda_U^l)$ and $t_u^{\text{end}}(\lambda_U^l)$ around time instant t (and does not include t). For this case, MH-MCU can be performed by a series of steps that invert the motion information generated during MH-MCP of stage λ_U^l. Specifically, first the inversion of the forward prediction is performed, where for each $q = \{t - t_u^{\text{init}}(\lambda_U^l), \ldots, t-1\}$ we have:

$$p_{U,m}^{\text{res}} = \begin{cases} 0, & \text{if } p_{U,m}^{\mathcal{F}_{t-q}^l(q)} = 0 \\ 2^l - p_{U,m}^{\mathcal{F}_{t-q}^l(q)}, & \text{otherwise} \end{cases}, \; p_{U,n}^{\text{res}} = \begin{cases} 0, & \text{if } p_{U,n}^{\mathcal{F}_{t-q}^l(q)} = 0 \\ 2^l - p_{U,n}^{\mathcal{F}_{t-q}^l(q)}, & \text{otherwise} \end{cases} \tag{4.43}$$

$$d_{U,m}^{\text{res}} = \begin{cases} 0, \text{ if } p_{U,m}^{\mathcal{F}_{t-q}^l(q)} = 0 \\ 1, \quad \text{otherwise} \end{cases}, \; d_{U,n}^{\text{res}} = \begin{cases} 0, \text{ if } p_{U,n}^{\mathcal{F}_{t-q}^l(q)} = 0 \\ 1, \quad \text{otherwise} \end{cases} \tag{4.44}$$

$$\begin{aligned} \mathcal{T}_U^l Z_t[m - d_{U,m}^{\mathcal{F}_{t-q}^l(q)}, n - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}] &\leftarrow \mathcal{T}_U^l Z_t[m - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}, n - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}] \\ &\quad + w_{U,q}^l[m,n] \cdot \beta_{U,q}^l \cdot \mathcal{S}_{U,(p_{U,m}^{\text{res}}, p_{U,n}^{\text{res}})}^l \mathcal{T}_U^l H_q^{\lambda_U^l}[m - d_m^{\text{res}}, n - d_n^{\text{res}}] \end{aligned} \tag{4.45}$$

$$c_U^l[m - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}, n - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}] \leftarrow c_U^l[m - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}, n - d_{U,n}^{\mathcal{F}_{t-q}^l(q)}] + 1$$

$$\mathcal{T}_U^l L_t^{\lambda_U^l}[m,n] = \left[\mathcal{T}_U^l L_t^{\lambda_U^l - 1}[m,n] + \frac{1}{\max\{c_U^l[m,n], t^{\mathcal{F}}(\lambda_U^l)\}} \mathcal{T}_U^l Z_t[m,n] \right] \tag{4.46}$$

Subsequently, after another initialization performed by (4.42), the inversion of the backward prediction is performed, where for each $q = \{t+1, \ldots, t + t_u^{\text{end}}(\lambda_U^l)\}$ we have:

$$p_{U,m}^{\text{res}} = \begin{cases} 0, & \text{if } p_{U,m}^{\mathcal{B}_{q-t}^l(q)} = 0 \\ 2^l - p_{U,m}^{\mathcal{B}_{q-t}^l(q)}, & \text{otherwise} \end{cases}, \; p_{U,n}^{\text{res}} = \begin{cases} 0, & \text{if } p_{U,n}^{\mathcal{B}_{q-t}^l(q)} = 0 \\ 2^l - p_{U,n}^{\mathcal{B}_{q-t}^l(q)}, & \text{otherwise} \end{cases} \tag{4.47}$$

$$d_{U,m}^{\text{res}} = \begin{cases} 0, \text{ if } p_{U,m}^{\mathcal{B}_{q-t}^l(q)} = 0 \\ 1, \quad \text{otherwise} \end{cases}, \; d_{U,n}^{\text{res}} = \begin{cases} 0, \text{ if } p_{U,n}^{\mathcal{B}_{q-t}^l(q)} = 0 \\ 1, \quad \text{otherwise} \end{cases} \tag{4.48}$$

$$\begin{aligned} \mathcal{T}_U^l Z_t[m - d_{U,m}^{\mathcal{B}_{q-t}^l(q)}, n - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}] &\leftarrow \mathcal{T}_U^l Z_t[m - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}, n - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}] \\ &\quad + w_{U,q}^l[m,n] \cdot \beta_{U,q}^l \cdot \mathcal{S}_{U,(p_{U,m}^{\text{res}}, p_{U,n}^{\text{res}})}^l \mathcal{T}_U^l H_q^{\lambda_U^l}[m - d_m^{\text{res}}, n - d_n^{\text{res}}] \end{aligned} \tag{4.49}$$

$$c_U^l[m - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}, n - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}] \leftarrow c_U^l[m - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}, n - d_{U,n}^{\mathcal{B}_{q-t}^l(q)}] + 1$$

$$\mathcal{T}_U^l L_t^{\lambda_U^l}[m,n] \leftarrow \left[\mathcal{T}_U^l L_t^{\lambda_U^l}[m,n] + \frac{1}{\max\{c_U^l[m,n], t^{\mathcal{B}}(\lambda_U^l)\}} \mathcal{T}_U^l Z_t[m,n] \right] \tag{4.50}$$

The parameters $t_u^{\text{init}}(\lambda_U^l)$, $t_u^{\text{end}}(\lambda_U^l)$, and $\beta_{U,t-t_u^{\text{init}}(\lambda_U^l)}^l, \ldots, \beta_{U,-1}^l, \beta_{U,1}^l, \ldots, \beta_{U,t+t_u^{\text{end}}(\lambda_U^l)}^l$, are directly taken from the lifting factorization of the chosen filter-bank [43]. In addition, $t^{\mathcal{F}}(\lambda_U^l)$ and $t^{\mathcal{B}}(\lambda_U^l)$ are derived from the scaling factor of the lifting formulation of the wavelet decomposition.

4.6 An Advanced Motion Estimation Algorithm for MCTF

In this section we present an advanced motion estimation algorithm and its instantiation for MCTF. In particular, our proposal consists of a new algorithm for optimized multihypothesis motion estimation for MCP and MCU using block-based ME/MC [12]. The motion estimation algorithm operates following a macroblock concept, i.e. the current frame (or wavelet subbands of each

resolution level for IBMCTF) is partitioned into non-overlapping blocks of $B \times B$ pixels (or $\frac{B}{2^l} \times \frac{B}{2^l}$ wavelet coefficients for each subband of resolution l). Then the algorithm establishes predictions for the macroblocks based on a set of reference frames and produces a set of motion-vectors and the predicted error frame (or error subbands of each resolution). After the performance of the prediction step for a sufficient number of frames, the corresponding update step inverts the error-frame information to the reference frames using the inverse motion vector set and creates the temporally lowpass-filtered frames to be used for the subsequent temporal levels.

A pruning algorithm for variable block-size ME has already been proposed in the context of MCTF [61]. Our approach differs from [61] in the use of multihypothesis, the use of multiple reference frames, which correspond to longer temporal filters, and the more exhaustive approach for the macroblock prediction-mode selection.

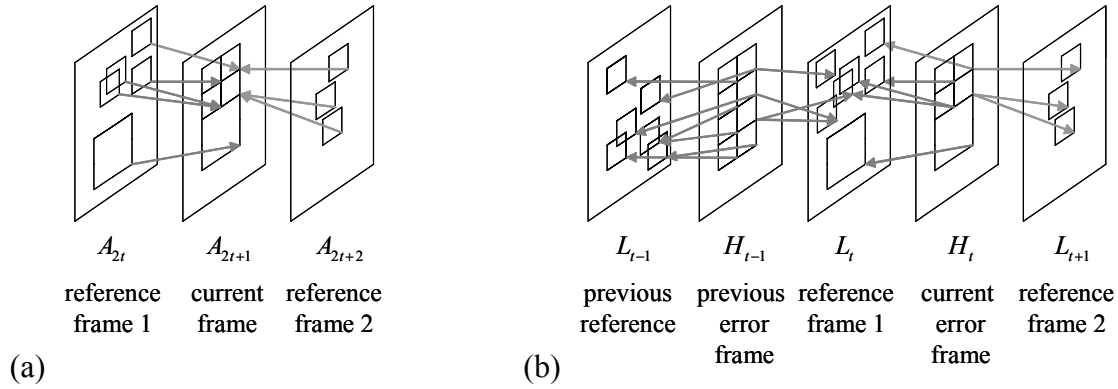


Figure IV-25. (a) Examples of variable block size multihypothesis prediction of frame A_{2t+1} using frames A_{2t}, A_{2t+2} as references. The arrows indicate the prediction direction. (b) The corresponding example of the update step: The information of the produced error frame from the multihypothesis prediction is used to update the two reference frames. To complete the creation of the L_t frame, the previous update step with error-frame H_{t-1} is necessary, leading to a temporal dependency of five frames.

Multihypothesis prediction has been originally proposed in the context of closed-loop video coding in order to improve prediction efficiency [64, 77, 78]. Figure IV-25(a) illustrates that the basic concept can be seen as a generalization of bidirectional prediction: each block may be predicted using a unidirectional or bidirectional prediction with one or two reference blocks. To utilize such a technique in MCTF-based coding, we couple the prediction step with the corresponding update step as shown in Figure IV-25(b): the current error frame is used to update the reference frames and create a set of L-frames.

The example of Figure IV-25 corresponds in general to temporal filtering using the motion-compensated 5/3 filter-bank. However, as demonstrated in Chapter III, due to the adaptive properties of the optimized multihypothesis MCP and MCU, the temporal filtering is locally adapted to: motion-compensated 1/3 filter-bank (5/3 with no update step), motion-compensated bidirectional 2/2 (Haar) filter-bank and the motion-compensated, bidirectional, 1/2 filter-bank (Haar with no update step). In

general, the optimized multihypothesis MCTF proposed in this section can be seen as a rate/distortion optimized adaptive motion-compensated temporal lifting decomposition with bookkeeping: the algorithm performs a best-basis selection process in the direction of motion and indicates the decisions to the decoder using the motion-vector information. Although our experiments are restricted to the biorthogonal motion compensated filter-banks with maximally one predict-and-update step, generalizations to smoother wavelet families that capture better the long-term temporal correlations can be envisaged, as demonstrated by the formulation of subsection 4.5.2.

4.6.1 Prediction Step

Although the proposed ME is based on the algorithm of [77], its novelty lays in the joint optimization process for the multihypothesis prediction with variable block sizes and in its ability to generate a rate-constrained motion-vector data set for a *multiple* set of rates, *without* multiple application of the complex multihypothesis estimation step. The latter is possible by performing the operation of optimized prediction for each macroblock in three different phases, as shown in Figure IV-26.

- *Macroblock split*. Starting from a macroblock of $B \times B$ pixels (or $\frac{B}{2^l} \times \frac{B}{2^l}$ coefficients in the case of IBMCTF at resolution level l), a splitting process generates a number of non-overlapping subblocks. In the presented experiments we follow a quadtree splitting approach to P partition levels, where each level p_s contains $2^{p_s} \times 2^{p_s}$ subblocks, $0 \le p_s < P$. For each level p_s, a number of subblocks have been pruned-out due to the selection of their parent block during pruning. As a result, for each level, the following steps are only performed to the subblocks that have been selected during the pruning step of the previous level.

- *Multihypothesis ME*. For the current subblock, a local splitting to its four quadrants is performed. A number of hypotheses M is established for the subblock and its four quadrants, with $M = 2$ in our experiments. The case of $M = 0$ can correspond to the use of intra-prediction modes based on the causal neighbourhood around the current subblock (or subblock quadrant). For the current subblock (or subblock quadrant) and each hypothesis $m = 1,..,M$, we apply the multihypothesis estimation algorithm of [77] without a rate constraint. This means that, for $m > 1$, the algorithm initiates all motion vectors and then iterates by estimating one vector at a time so that the prediction error of the current subblock is minimized. When no vector was modified during the last iteration, the algorithm terminates. As a result, for each hypothesis m, the combination of motion vectors that minimizes the produced error-frame distortion in the certain subblock (or subblock quadrant) of the macroblock is chosen. In our experiments, we use the sum of absolute differences as the distortion measure. The motion vector data for each hypothesis and the corresponding distortion are kept in a structure for use in the following steps.

- *Pruning process*: The pruning process performed for the current subblock is given in Figure IV-27. Three distinct passes occur: The first pass, **RD_Estimation(i)**, identifies the rate and distortion of each possible combination n_i (partitioning point) of hypotheses of the current subblock or its quadrants. The second pass, **RD_Prune**, scans the list of acceptable points $\mathcal{N}$ to establish a monotonically decreasing slope value S_j for each point n_j, $n_j \in \mathcal{N}$,

similar to the post-compression rate-distortion optimization procedure of JPEG2000 [3]. Finally, the third pass minimizes the Lagrangian cost function $R(n_j) + \lambda \cdot D(n_j)$ by establishing the partitioning point $n_{j_{\min}} = \arg\min_{n_j \in \mathbf{N}} (|S_j - \lambda^{-1}|)$, i.e. the partitioning point with a slope value closest to λ^{-1}. The splitting and hypothesis number for each subblock (or for the subblock's quadrants) is used for motion compensation, if no additional partition levels are to be performed. Otherwise, an additional level of macroblock split occurs and the multihypothesis ME and pruning occur for the subblocks that have been selected in the previous levels, and their quadrants.

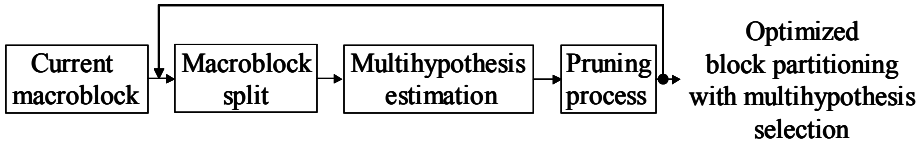


Figure IV-26. Proposed motion estimation for MH-MCP.

For the current subblock at (r_s, c_s) of partition level p_s, $0 \le p_s < P$ in the macroblock:

If $p_s = 0$ then Set $\mathcal{N} = \varnothing, \mathcal{N}_{deleted} = \varnothing$ `// Initialize to zero: no MB partition exists yet.`

Else $\mathcal{N} = \mathcal{N}_{deleted}$ `// Restore partition points excluded before.`

For every $p : p \leftarrow \{0,1\}$ `// For all the different combinations of hypotheses in`

For every $i : i \leftarrow \{0,1,\ldots,(M+1)^{2^{2p}} \text{-}1\}$ `// the subblock (` $p = 0$ `) and its quadrants (` $p = 1$ `).`

Begin_Estimation(i):

Set $R \leftarrow 0$, Set $D \leftarrow 0$

For every quadrant of the subblock at (b_r, b_c): $b_r, b_c \in [0, 2^p - 1], b_r \in \mathbb{N}$

Set $h(b_r, b_c) \leftarrow \left(\left\lfloor i / (M+1)^{b_r 2^p + b_c} \right\rfloor \bmod (M+1) \right)$ `// mod: modulo operator`

Set $R \leftarrow R + \text{Estimate}_R(b_r, b_c, h(b_r, b_c))$

Set $D \leftarrow D + \text{Estimate}_D(b_r, b_c, h(b_r, b_c))$ `// precalculated from the ME step`

Set $R \leftarrow R + R_{rem}$, $D \leftarrow D + D_{rem}$ `// rate,distortion of remaining parts of MB`

Create $n_i = \left\{ R, D, \bigcup_{b_r, b_c} h(b_r, b_c), [(r_s, c_s), p_s] \right\}$

For every $n_j, n_j \in \mathcal{N}$

If $n_j(R) < n_i(R)$ and $n_j(D) \le n_i(D)$ then Goto **End_Estimation(i)**

Add n_i to $\mathcal{N}$

End_Estimation(i)

`// prune the list of valid R-D points to make a monotonically-decreasing slope`

Begin_RD_Prune:

Set $l = 0$, Set $n_l(R) = 0$, Set $n_l(D) = \infty$ `// Set the initial point`

For every $n_j, n_j \in \mathcal{N}$

Set $\Delta R_j \leftarrow n_j(R) - n_l(R)$, Set $\Delta D_j \leftarrow n_l(D) - n_j(D)$, Set $S_j \leftarrow \dfrac{\Delta D_j}{\Delta R_j}$

If $l \neq 0$ and $S_j > S_l$ then Move n_l from $\mathcal{N}$ to $\mathcal{N}_{deleted}$, Goto **Begin_RD_Prune**

else Set $l = j$

End_RD_Prune

Begin_Estimation_Truncation:

Input λ `// input of the control parameter`

For every $n_j, n_j \in \mathcal{N}$

If $S_j - \lambda^{-1} < S_{j+1} - \lambda^{-1}$ then Set n_j as the stop point

Apply_partition(n_j) `// apply the vectors and partition data of` n_j

End_Estimation_Truncation

Figure IV-27. Pseudocode of the pruning algorithm for multihypothesis variable block-size prediction for each macroblock. We use the following notations: $\mathcal{N}$ is a list of partitioning points in the macroblock, it contains items in the form $n_i = \left\{ R, D, \bigcup_{b_r, b_c} h(b_r, b_c), [(r_s, c_s), p_s] \right\}$ where R, D is the rate, distortion respectively, $\bigcup_{b_r, b_c} h(b_r, b_c)$ is the union of the quadrants of the subblock (b_r, b_c) each having a hypothesis $h(b_r, b_c)$ and $(r_s, c_s), p_s$ indicate the subblock coordinates in the macroblock; $\mathcal{N}_{deleted}$ contains points that have been removed from $\mathcal{N}$; $\text{Estimate}_R(\bullet)$ estimates the rate for coding the motion vector data of the subblock (b_r, b_c) using the first order entropy, $\text{Estimate}_D(\bullet)$ estimates the prediction error of the subblock (b_r, b_c); R_{rem}, D_{rem} contain the rate, distortion of the remaining area of the macroblock (besides area covered by the current subblock); λ is the Lagrangian control parameter.

4.6.2 Update Step

The application of the update step for the creation of the L_t frame (Figure IV-25) occurs in two consecutive phases, and it follows the design presented analytically in Chapter III: first the update information is created by inverting the error frame samples of the H_{t-1}, H_t frames using the inverted motion-vector fields. To avoid strong motion-related artifacts in the output L_t frame and the irregular increase of the image-sample magnitudes in multi-connected areas, a normalization process divides the magnitude of the update samples for each pixel with the number of connections. Finally, before the update coefficients are added to the reference frame, they are scaled according to the lifting equation for the update step, taking into account the type of the specific connection (Haar or 5/3 filter-bank]). For the case of IB-MCU, analytical formulations for this process are given in subsection 4.5.1 and 4.5.2. Although there have been proposals in the literature with motion estimation algorithms for MCU [8], practical designs have shown that these approaches do not significantly improve the compression performance of the coding system, while the overhead required for the additional motion-vector information is significantly increased [75]. Hence, these approaches are not investigated in this dissertation.

4.7 Experimental Evaluations

This section presents experimental evaluations of the in-band video coding algorithms of this chapter versus their corresponding video coding algorithms in the spatial domain, as well as versus DCT-based closed-loop video coders. Various experimental evaluations of the different algorithms and tools were performed in a variety of publications, see for example [10-12, 26, 46, 47, 54, 68, 73, 74, 76, 79]. This chapter summarizes these results and presents them in a more organized and uniform manner. All the algorithms tested in this section were implemented in "C++" and tested in an Intel Pentium-IV platform, with the exception of the closed-loop video coders, where the invocation of the "C++" codec modules (ME/MC, etc.) during encoding and decoding was done via a Matlab script.

For all the basic video coding algorithms, the experimental evaluation was performed on set of five CIF test video sequences that represent a large variety of video content. The range of bitrates chosen for our tests corresponds to a broad range of practical bitrates for coding of CIF content, namely the region of 200 – 2000 kbps. Finally, wherever possible, all the algorithms were used with the same set of settings, which allows for a more thorough evaluation of the results. The following subsection analyzes the overall settings used for our comparisons, while the subsequent subsection (4.7.2) presents the experimental evaluation of the proposed coders using these settings. The two particular tools proposed within the open-loop prediction structure, namely the MPEG-compliant base-layer architecture for open-loop IB-MCP (Subsection 4.4.2) and the distortion-control algorithm of Subsection 4.4.3, are experimentally evaluated in Subsections 4.7.4.1 and 4.7.4.2, respectively. Subsection 4.7.5 evaluates the advanced motion estimation algorithm presented in Section 4.6, under a variety of settings. Finally, for the best settings of the coders that utilize the proposed motion estimation, a comparison against the state-of-the-art in non-scalable coding is carried out in Subsection 4.7.7.

4.7.1 Common Experimental Settings

4.7.1.1 Temporal Prediction Structure

For the coding experiments of the following subsections, all the video-coding schemes under comparison utilized a GOP structure with 16 frames. This is a standard scenario in video coding tests [80]; for sequences with replay rate equal to 30 Hz (30 frames-per-second), this scenario provides random access of the compressed bitstream at (approximately) half-second intervals. This is useful for providing fast browsing and "rewind" functionalities.

Concerning the prediction structure, all the coding schemes that utilize the closed-loop architecture performed successive prediction using one I-frame and 15 P-frames; all frames were predicted from their (dequantized) previous frame. The open-loop architectures utilized 4 temporal decomposition levels; at each temporal level, unidirectional prediction from the previous frame was utilized. As a result, the utilized temporal decomposition corresponds to one I-frame and 15 unidirectionally predicted frames. This setting ensures that any potential differences in performance do not stem from the utilized MCP algorithm, but rather from the different frame-scanning pattern of each prediction structure (i.e. open loop vs. closed-loop) and, potentially, from the use of different settings for the spatial transform and the frame encoding (e.g. spatial-domain vs. in-band schemes – DWT-based vs. DCT-based coding).

4.7.1.2 Utilized Test Material

In our experiments we used five representative CIF sequences (with replay rate equal to 30 Hz). They have been chosen to represent a variety of content. Specifically, two of the chosen sequences ("Coastguard" and "Mobile") contain camera panning with a variety of textures and object motions in the scene. Two other sequences ("Silent" and "Foreman") present talking persons with and without random camera motion and complicated scene textures. Finally, the sequence "Stefan" is a sports video from a tennis game. All sequences have been downloaded from ftp://ftp.tnt.uni-hannover.de, which is currently the standard MPEG FTP site with video test material. These sequences have been used extensively during the experimental evaluation of the recent MPEG exploration on scalable video coding technology [80]. Each sequence originally consists of 300 frames. However, for the purposes of our tests, all sequences have been extended by 4 frames, by repeating the last frame. This stems from the fact that we opted to process an integer number of GOPs.

4.7.1.3 Motion Compensated Prediction Parameters

For all our tests, the codecs under comparison used block-based motion estimation with the full-search (exhaustive) algorithm in order to determine the optimal motion parameters for each block. For the search process, the accuracy of the implemented codecs was set to quarter-pixel; for this purpose, the interpolators presented in Chapter 3 were used. The utilized block size was set to 16×16 pixels for SD-MCP, as this is the standard macroblock size used in MPEG codecs. The

search range was set to ± 16 pixels around the current block position. We note that, although full-search motion estimation was performed for the full-pixel positions within the search range, the fractional-pixel position of the best match was identified by refining the search around the position of the best-match found in the full-pixel grid.

Concerning the in-band codecs, for IB-MCP performed in the single-level spatial DWT decomposition, an in-band block is considered to consist of all four blocks of 8×8 wavelet coefficients at the corresponding locations of all four wavelet subbands. For IB-MCP performed in a two-level spatial DWT decomposition, an in-band block at each resolution level was considered to consist of all blocks of $(32 \cdot 2^{-l}) \times (32 \cdot 2^{-l})$ wavelet coefficients of all DWT subbands of level l, $1 \leq l \leq k$. We note that this scenario corresponds to the level-by-level in-band motion estimation algorithm, presented by equations (4.11) and (4.12), with dyadically-reduced block sizes for each resolution level. For all cases, the search range was set to $\pm(16 \cdot 2^{-l})$ in-band positions around the current in-band block position. For each level of IB-MCP, motion estimation (with the full search algorithm) was used to determine the optimal in-band motion parameters for each block. During motion estimation, all the phases of the ODWT were used and fractional phases corresponding to the equivalent of quarter-pixel accuracy were created in the UDWT domain, by using the interpolators presented in Chapter 3. Similarly as in the spatial-domain motion estimation, the fractional-phase position of the best match was identified by refining the search around the position of the best match found in the integer-phase ODWT grid.

The previously-described settings for the in-band codecs present only one out of many possible configurations for in-band block sizes and search ranges. In general, we opted for the specific setting due to the fact that, on average, it appeared to produce similar overhead for the motion-vector information with the spatial-domain case. This is due to the fact that the chosen block sizes and search ranges for the in-band case resemble our selections for the spatial-domain schemes.

4.7.1.4 Spatial Transform – Error-frame Compression

For all the proposed codecs in this dissertation, the $9/7$ spatial transform was used [44] (with floating-point precision), as it consists one of the best-performing wavelet filter-banks in the literature and is also adopted in the JPEG-2000 standard [45]. Moreover, independent experiments [81] demonstrated that 16-bit integer-arithmetic implementations of this filter-bank achieve similar coding performance to the floating-point implementation, hence, from the complexity point of view, it represents a competitive solution to efficient DCT-based schemes. For all the CIF sequences, 4 spatial decomposition levels were performed. For the in-band coding schemes, this means that $4 - k$ additional spatial decomposition levels were performed to the residue of the LL subbands of the k-level IB-MCP.

Concerning the compression aspects of the proposed schemes, the produced wavelet coefficients were quantized with successive approximation quantization [29] and then coded with the QuadTree Limited coder [70] [71], which uses quadtree coding of the significance maps and context-based arithmetic coding. This algorithm was presented in Chapter III.

4.7.1.5 Rate Control – Scalability Features

For all the proposed coding schemes involving the closed-loop architecture, no rate-allocation was designed. Instead, similar to the test-model implementations of H.263+ [82] and H.264 [83], the distortion of the compressed video is controlled via the quantization factor for the I- and P- frames. In particular, for the QT-L coder, the coding always stopped at a certain pass of a certain bitplane; the choice of both the pass-number and bitplane-number is controlled by the user through a parameter file. By setting these parameters to the same value for all the frames prior to compression, small PSNR variation is achieved throughout large portions of each sequence.

In general, for all the closed-loop coders compared in this dissertation, the experimental rate-distortion points were always produced by varying the quantization parameters of each codec and measuring the produced compressed file-size and the achieved MSE in all the colour channels of the decoded video. As a result, all the closed-loop coders of this chapter are not scalable in bitrate. Moreover, the use of a GOP structure with only I- and P-frames does not provide the possibility for frame-rate scalability. As a result, although we have proposed fully-scalable extensions of these coding frameworks (e.g. see [26, 46]), we do not elaborate on their results in this dissertation. This is due to the fact that bitrate and frame-rate scalability is performed in a much more efficient manner within the open-loop architectures. For example, independent experiments [26] [15] demonstrated that bitrate scalability within the in-band closed-loop coding framework always incurs a coding penalty, similar to what is observed in conventional bitrate-scalable closed-loop video coding (e.g. the MPEG-4 FGS [65]). This motivated us to compare against the closed-loop coding algorithms in this chapter using a simple, non-scalable framework. Our choice provides the possibility of an elaborate experimental evaluation of the different schemes under the same conditions, i.e. without the influence of rate-control algorithms or coding-efficiency loss incurred by bitrate and frame-rate scalability.

In contradiction to the closed-loop predictive coding, open loop architectures exhibit seamless bitrate scalability due to the fact that rate-allocation is performed post compression. Simple and efficient algorithms exist in the literature that can match each bitrate accurately [75]. Usually, such algorithms consider the compressed bitstream of the entire sequence during the rate allocation. However, this can be localized to independent rate allocation for each GOP, if so desired. Our choice of using the entire sequence corresponds to the (commonly-used) noise-free case where the entire sequence is compressed once and stored in a video server. From this compressed bitstream, different bitrates need to be extracted and losslessly-transmitted at a later time to a variety of terminals (video clients), each having a certain connection bandwidth, display resolution and replay frame-rate.

The bitstream extraction is performed by an extractor that follows the principles of the bitrate-scaling method used in [75]. In practice, this method is equivalent to the post-compression rate-distortion optimization (PCRDO) of JPEG-2000 [3], applied for 3-D data sets, i.e. the entire video sequence. As demonstrated in [75] [3], the extraction method used in the open-loop codecs of this dissertation can match each bitrate with a mismatch that is smaller than 0.5% for each desired bitrate. Similar to the common setting used in the literature [75], the rate needed to represent the rate-distortion points used for the PCRDO was not taken into account in the total bitrate measurement of each experimental R-D point. We note though that, if transmitted to the terminal, these points will consume additional bandwidth. Nevertheless, their exclusion from the total bitrate of each experimental point makes the

comparison between open- and closed-loop architectures more accurate. Moreover, these points may affect the bitrate in a significant manner only in the low-rate regime. Finally, their use is mandatory only in the case when one needs to perform bitrate-scaling operations in order to transmit sequences of lower quality, resolution or frame-rate to other terminals.

4.7.1.6 Motion Vector Coding

Concerning the motion parameters of each proposed coding scheme, the motion vectors of each frame (and resolution in the case of in-band MCP) are compressed using adaptive arithmetic coding; no spatial or temporal motion-vector correlations were exploited. The motion-vector bitrate was taken into account during the rate-allocation. Although scalable coding of motion vectors enhances the performance for the low-bitrate regime [84] [85], we did not incorporate such coding schemes in our comparisons or any other advanced context-based motion-vector coding techniques [86]. Their impact on the overall coding performance is, typically, noticeable for the low-rate regime.

4.7.2 Spatial-domain versus In-band Motion Compensated Prediction

In the first set of comparisons, Figure IV-28 – Figure IV-32 present the obtained results with the codecs having the following features:

- Closed-loop spatial-domain motion compensated prediction ("Closed-loop SD-MCP"), following the settings of the previous subsection.
- Closed-loop in-band motion compensated prediction with one level of in-band ME/MC ("Closed-loop IB-MCP, 1 level"); the settings of this codec follow the ones defined in Subsection 4.7.1, with $k = 1$.
- An H.263+ implementation [82]; the settings described in the previous subsection have been used. As a result, all the optional features of the codec such as deblocking, use of B-frames, etc., have not been used.

All the bitrates reported in Figure IV-28 – Figure IV-32 were measured from the compressed file-sizes. For each original frame $A[m,n]$ and each reconstructed frame $\widetilde{A}[m,n]$ consisting of $M \cdot N$ pixels represented in 8-bit precision, we use the classical peak-signal-to-noise ratio (PSNR) metric:

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{M\cdot N}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(A[m,n]-\widetilde{A}[m,n])^2} \tag{4.51}$$

In addition, for each sequence, the mean PSNR used in our comparisons is defined by [87] [12]:

$$Mean\_PSNR = \frac{4\cdot PSNR\_Y + PSNR\_U + PSNR\_V}{6} \tag{4.52}$$

where $PSNR\_Y$, $PSNR\_U$, $PSNR\_V$ represent the average PSNR (in dB) for each of the Y, U, V color channels of all the frames in the sequence, respectively. This is a commonly-used metric for

objective video quality measurements that incorporate the contributions of all color channels in the overall distortion. Our extensive experimentation revealed that, in most cases, the bitrate for the chrominance channels occupies roughly 30% of the overall bitrate for the texture information; hence we find the weighting used in (4.52) to be appropriate. As such, the mean PSNR metric is used in the majority of the results of this chapter to report the performance of the different schemes under comparison.

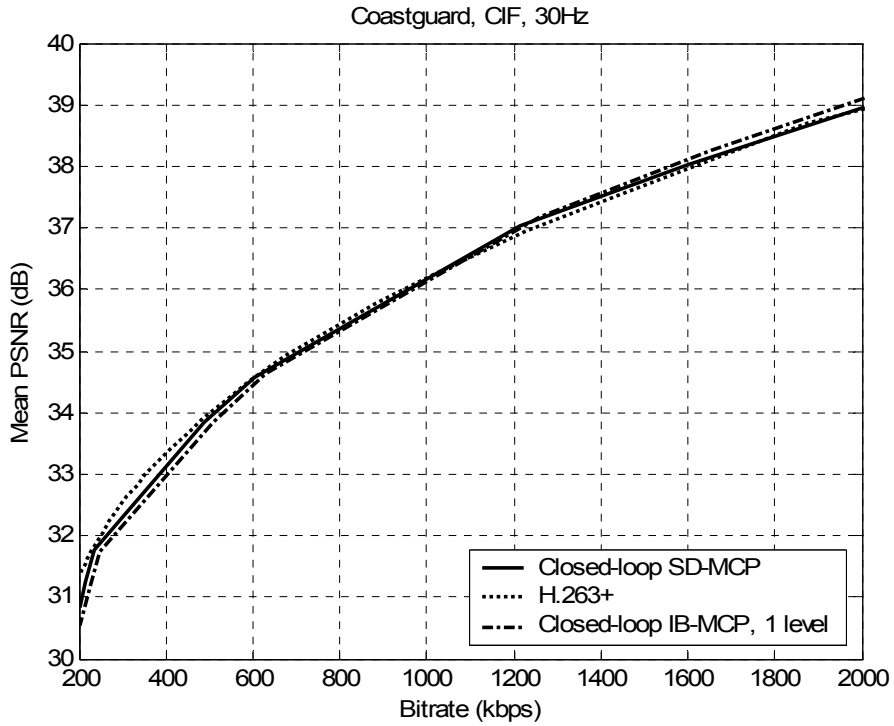


Figure IV-28. Comparison of closed-loop SD-MCP and IB-MCP architectures for the "Coastguard" sequence.

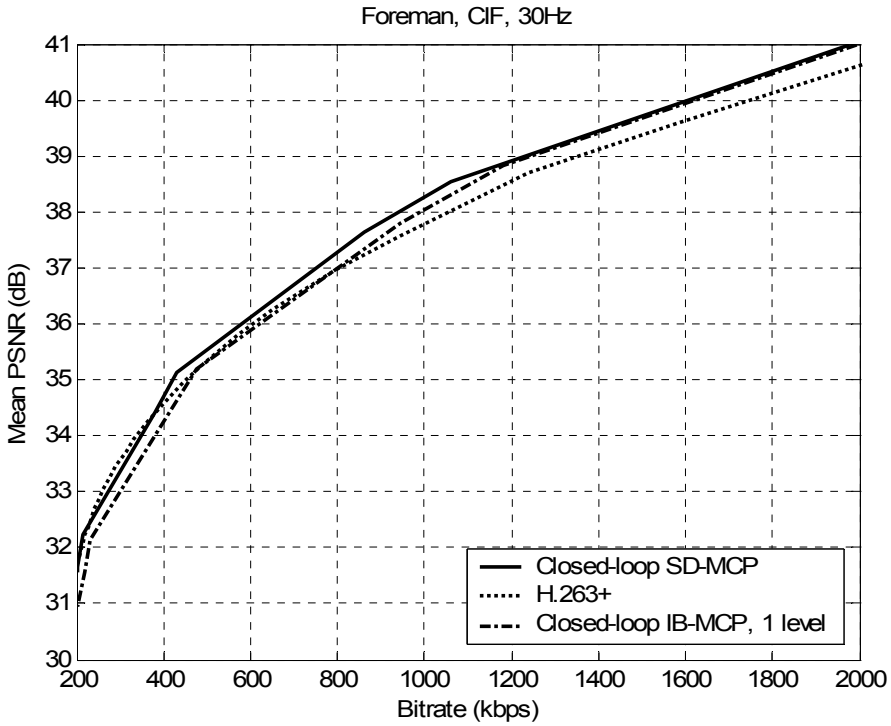


Figure IV-29. Comparison of closed-loop SD-MCP and IB-MCP architectures for the "Foreman" sequence.

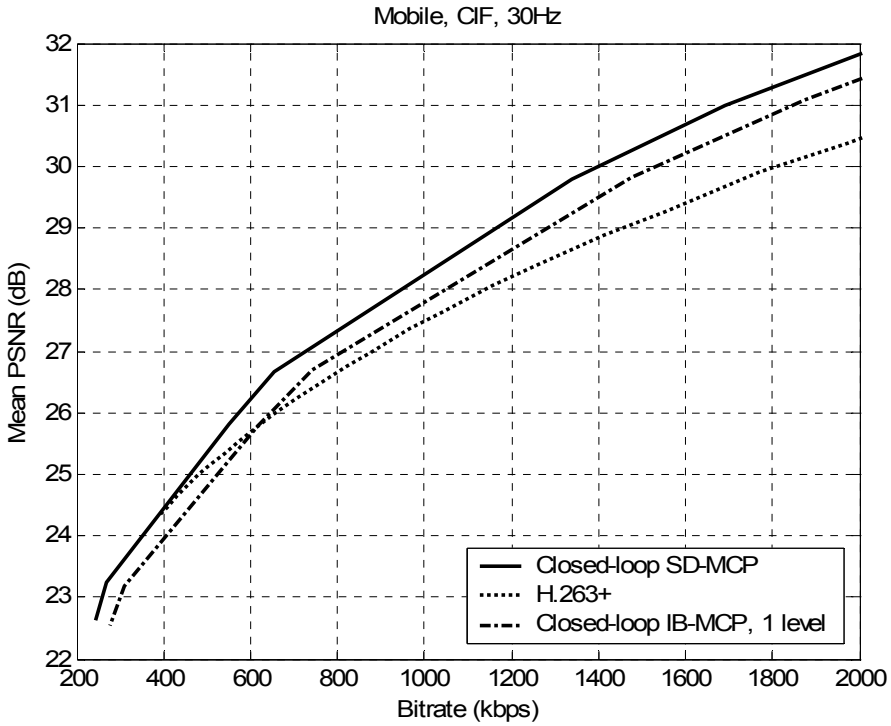


Figure IV-30. Comparison of closed-loop SD-MCP and IB-MCP architectures for the "Mobile" sequence.

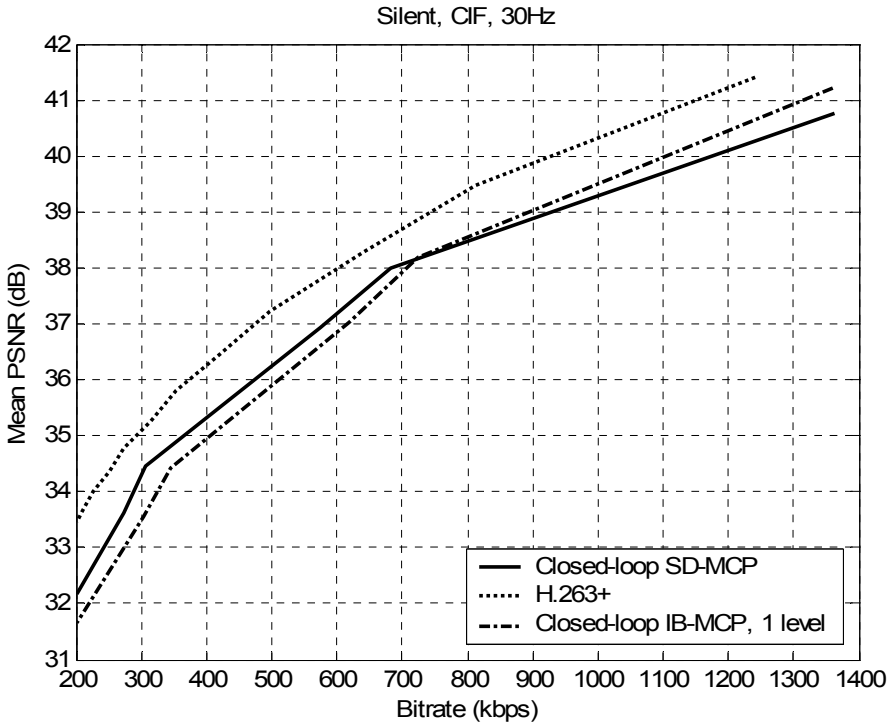


Figure IV-31. Comparison of closed-loop SD-MCP and IB-MCP architectures for the "Silent" sequence.

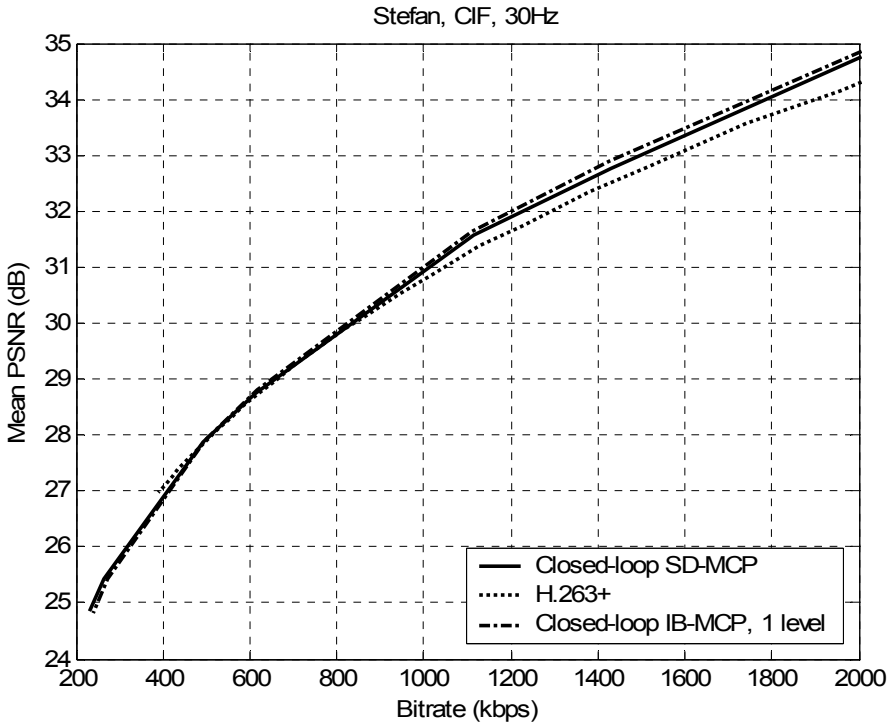


Figure IV-32. Comparison of closed-loop SD-MCP and IB-MCP architectures for the "Stefan" sequence.

The results of Figure IV-28 – Figure IV-32 show that the SD-MCP and IB-MCP closed-loop wavelet-based video compression schemes achieve comparable objective performance to the DCT-based H.263+, under the same encoding settings. In particular, it was found that they outperform H.263+ for all the test sequences, with the exception of the "Silent" sequence. The average margin of improvement offered by the wavelet solutions varies from 0.3 dB to more than 2.0 dB ("Mobile" sequence). On the other hand, in the "Silent" sequence, a loss of 1.0 dB (on average) is observed.

Among the two wavelet-based coding schemes, the results show that in all cases the two codecs appear to produce comparable results; one exception to this rule is observed in the "Mobile" sequence, where the SD-MCP based codec appears to outperform the IB-MCP based codec by 0.3 – 0.5 dB over the entire range of bitrates.

An example of a frame-by-frame comparison of the three schemes can be seen in Figure IV-33. All codecs appear to produce similar PSNR fluctuations across different frames for the quantization settings that correspond to approximately the same bitrate. This is a natural consequence of the quantizer-based rate-control that is present in all schemes under comparison. Concerning subjective evaluation of the results, it was observed that H.263+ and SD-MCP wavelet-based video coding generated similar types of artifacts, which mainly consist of strong blocking effects and blurriness at low bitrates. On the contrary, the IB-MCP codec displayed no blocking artifacts; instead ringing across the frame edges was observed, and also ringing effects were seen at the areas where the other coders displayed blocking artifacts. An example of the different types of visual artifacts can be seen in Figure IV-34.

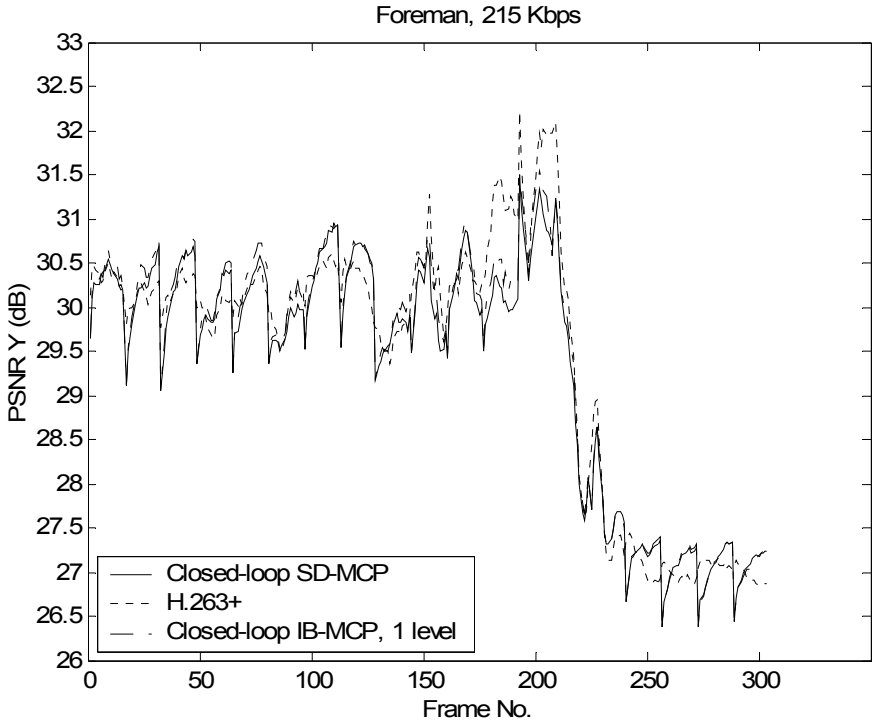


Figure IV-33. An example of a frame-by-frame comparison between SD-MCP, H.263+ and IB-MCP, at (approximately) the same bitrate.

Figure IV-34. An example of the artifacts generated at low bitrates (12-th frame of the "Foreman" sequence); from left to right, top to bottom: original frame, H.263+ (205 kbps, PSNR Y: 30.0 dB), SD-MCP (215 kbps, PSNR Y: 30.4 dB), IB-MCP, 1 level (216 kbps, PSNR Y: 29.8 dB).

In the second set of comparisons, Figure IV-35 – Figure IV-39 present the obtained results with the codecs having the following features:

- Open-loop spatial-domain motion compensated prediction ("Open-loop SD-MCP"), following the settings of Subsection 4.7.1.
- Open-loop in-band motion compensated prediction with one level of in-band ME/MC ("Open-loop IB-MCP, 1 level"); the settings of this codec follow the ones defined in Subsection 4.7.1, with $k = 1$.
- The H.263+ implementation [82], with the settings used previously (Figure IV-28 – Figure IV-32).

In this case, the results with the H.263+ scheme are repeated in order to serve as an anchor for the comparison between closed-loop and open-loop MCP architectures.

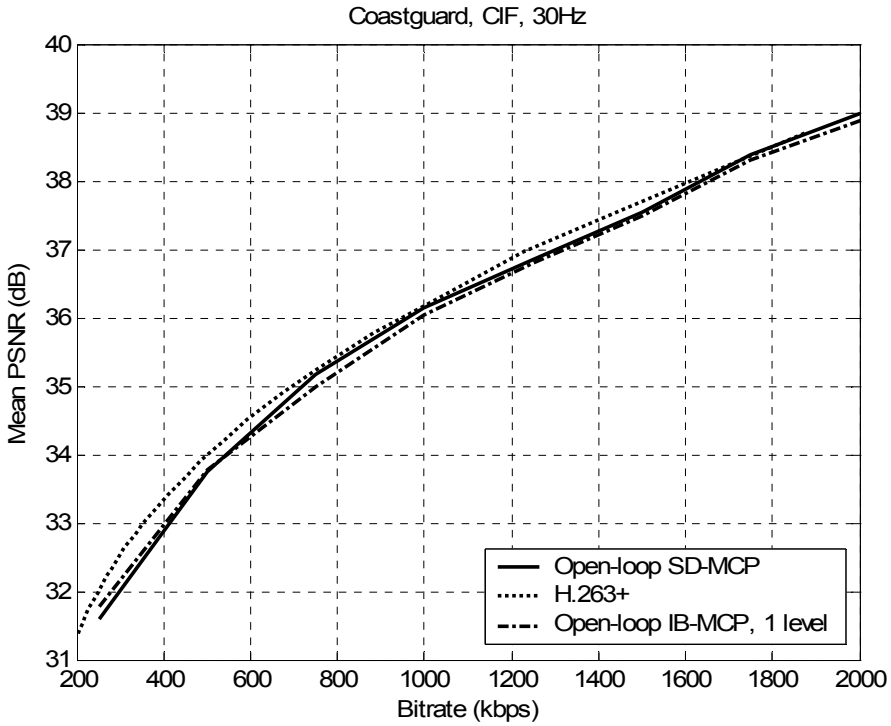


Figure IV-35. Comparison of open-loop SD-MCP and IB-MCP architectures for the "Coastguard" sequence.

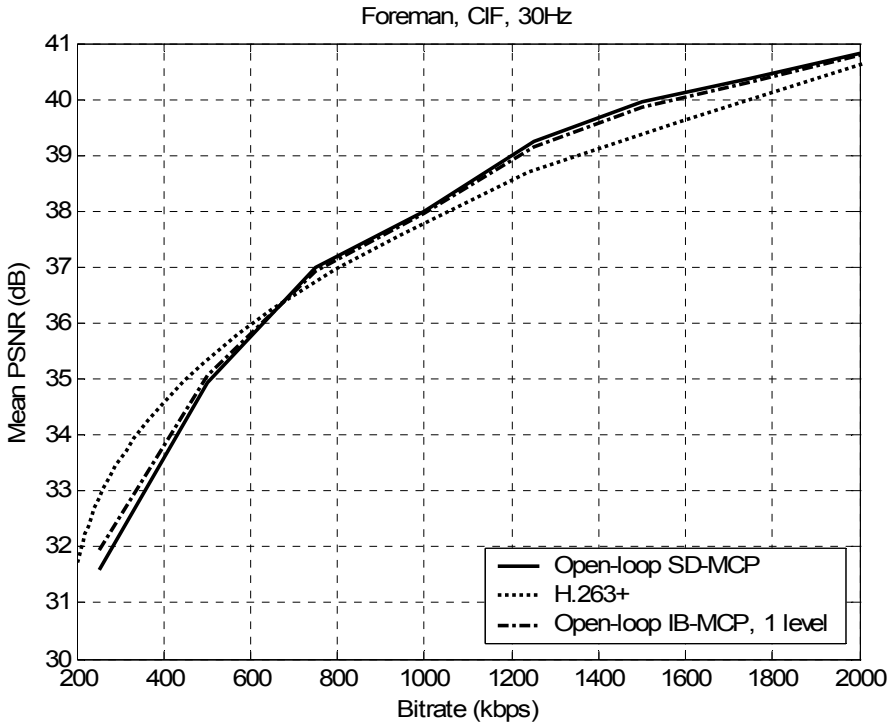


Figure IV-36. Comparison of open-loop SD-MCP and IB-MCP architectures for the "Foreman" sequence.

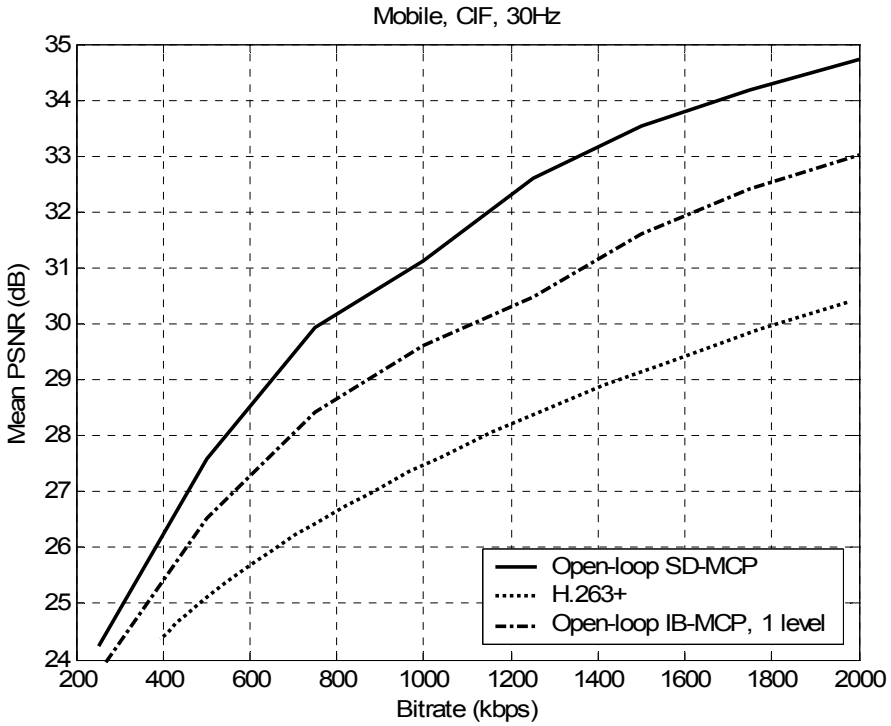


Figure IV-37. Comparison of open-loop SD-MCP and IB-MCP architectures for the "Mobile" sequence.

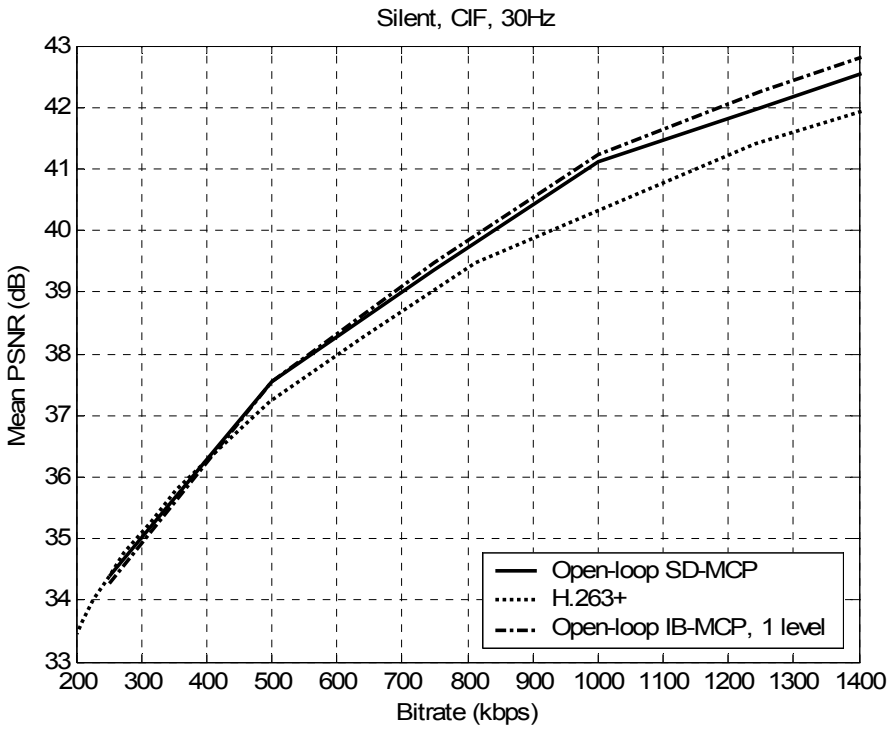


Figure IV-38. Comparison of open-loop SD-MCP and IB-MCP architectures for the "Silent" sequence.

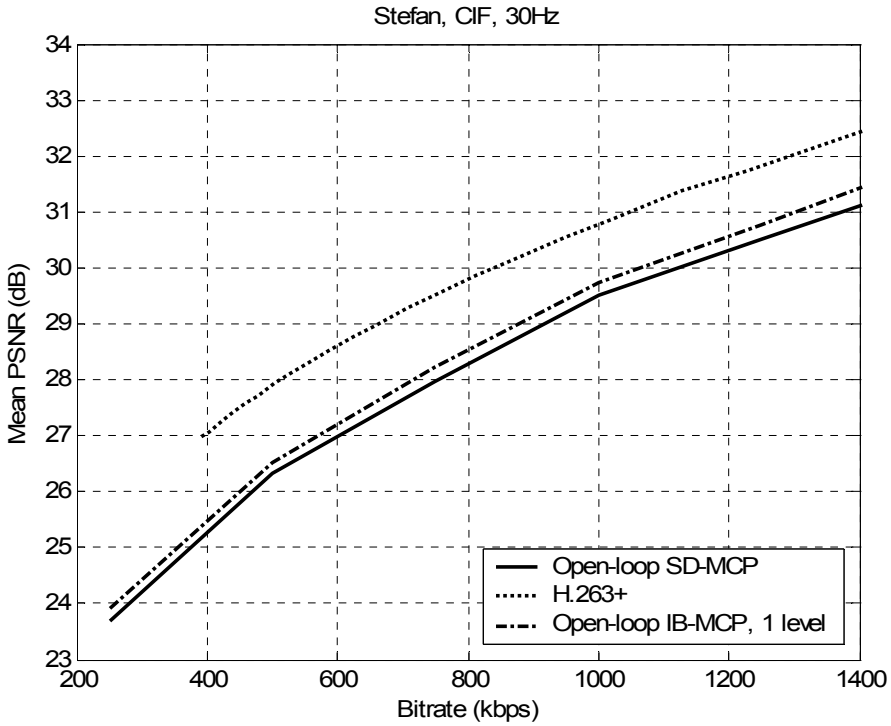


Figure IV-39. Comparison of open-loop SD-MCP and IB-MCP architectures for the "Stefan" sequence.

The results of Figure IV-35 – Figure IV-39 present significant differences with the previously-reported results for the closed-loop architectures. First, it is important to emphasize that all the results presented for each sequence with the open-loop wavelet-based coders were produced by extracting certain portions of one single compressed bitstream. On the contrary, the closed-loop schemes require separate compression and decompression for each experimental point of Figure IV-28 – Figure IV-32. Nevertheless, our experiments demonstrate that bitrate scalability does not appear to cause any PSNR saturation at high bitrates, an effect that is common for bitrate-scalable closed-loop video coders [65].

From the obtained results of Figure IV-35 – Figure IV-39, it appears that, for sequences with fast motion characteristics, the open-loop coding schemes are inferior to closed-loop MCP schemes. This becomes evident by comparing the results for the "Stefan" sequence, i.e. Figure IV-32 with Figure IV-39. Although the closed-loop wavelet-based schemes are comparable or superior to H.263+ for this sequence, the open-loop wavelet-based coders are outperformed by the DCT-based coder by approximately 1.5 dB in mean PSNR. We attribute this significant loss in coding efficiency to the fact that: (a) a fixed search range was used for all levels of the temporal decomposition structure of the spatial-domain and in-band open-loop MCP schemes, regardless of the distance between consecutive frames; (b) it appears that the simple block-based motion estimation used for the present experiments fails notably when the frame distance between consecutive frame increases, which is the case of the highest temporal levels of the open-loop schemes.

On the other hand, another notable difference in the results of the open-loop schemes appears in the "Mobile" sequence (Figure IV-30 and Figure IV-37). The experiments demonstrate that the open-loop architectures provide an enormous coding gain versus H.263+, which, for the SD-MCP scheme can be more than 4 dB in the medium and high-bitrate regime. This seems to agree with independent observations about the efficiency of open-loop architectures for this particular sequence [75]. This test sequence is known to contain a high degree of spatial aliasing due to a large number of (uniformly-moving) heavily-textured areas in every frame. It appears that wavelet-based open-loop MCP schemes with fractional-pixel accuracy treat this type of content more efficiently. This is also observed in the "Silent" sequence, where the background behind the speaker in the sequence consists of a static, heavily-textured, area; the results Figure IV-38 indicate that the open-loop wavelet-based architectures outperform the H.263+ coder; in practice, the comparison with their closed-loop equivalents (Figure IV-31) reveals that an increase of almost 2 dB is achieved in the mean PSNR at high bitrates.

Finally, in sequences with medium texture complexity and medium motion activity, namely "Coastguard" and "Foreman", the results appear to be rather comparable between open-loop and closed-loop architectures. One exception to this appears in the low-bitrate regime for the "Foreman" sequence, where the open-loop wavelet-based schemes perform worse than both the H.263+ and their closed-loop equivalents. We would like to reiterate that, apart from the different temporal prediction structure, open-loop SD-MCP used exactly the same prediction, spatial transform and compression settings as closed-loop SD-MCP; moreover the MCP settings also match the ones used for H.263+. Hence, as mentioned earlier, we attribute our observations to the fundamental differences between open-loop and closed-loop MCP.

Concerning the comparison between spatial-domain and in-band MCP within the open-loop architecture, the experiments presented in Figure IV-35 – Figure IV-39 demonstrate that comparable coding efficiency is provided by both schemes. One significant exception is again observed in the "Mobile" sequence, where it appears that the SD-MCP based coder outperforms its in-band equivalent by a significant margin, which can reach almost 2.0 dB in mean PSNR for high-bitrates. It appears that the natural "de-blocking" mechanism present in IB-MCP (due to the performance of the inverse DWT after motion compensation at the decoder) decreases the coding efficiency significantly in cases where motion is predicted very accurately and no boundary discontinuities exist among neighbouring blocks. Due to the fact that, for this particular sequence, the chosen block-based motion estimation works exceptionally-well within the open-loop architecture, it appears that the loss in coding efficiency for the IB-MCP versus SD-MCP (which was also observed in the results of Figure IV-30) is magnified.

An example of a frame-by-frame comparison of the three schemes can be seen in Figure IV-40. These results demonstrate that the open-loop MCP systems exhibit a significant PSNR fluctuation among consecutive frames. Both IB-MCP and SD-MCP produce approximately the same variance in PSNR across time; in fact the two curves are almost indistinguishable from the figure. In terms of visual quality, the significant fluctuations of the open-loop MCP schemes tend to be observable at low bitrates as flickering in the sequence. This appears to be the main drawback in going from a closed-loop, non-scalable, MCP-based coding scheme to its open-loop, bitrate-scalable, equivalent.

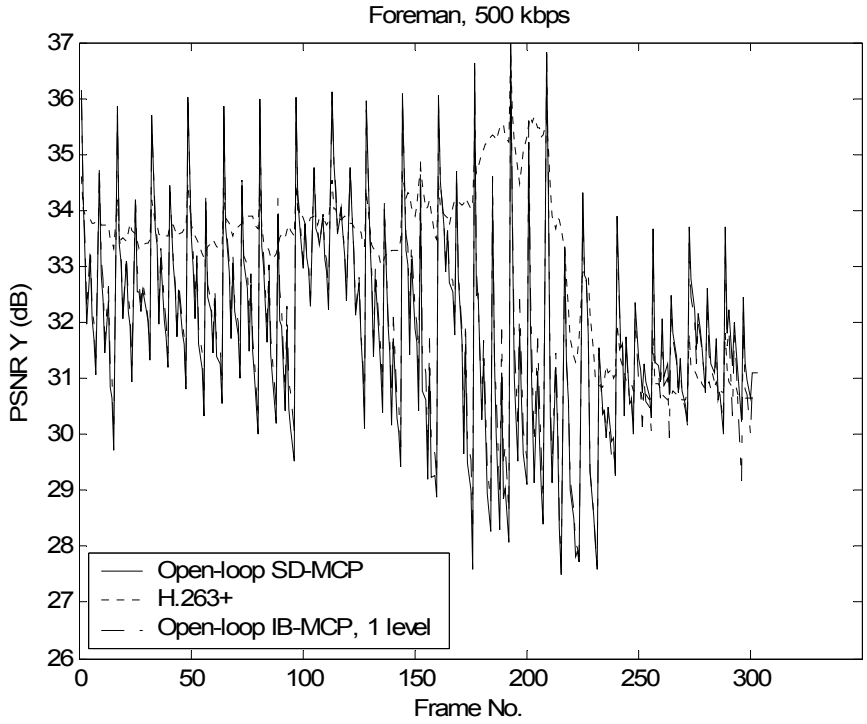


Figure IV-40. An example of a frame-by-frame comparison between open-loop SD-MCP, H.263+ and open-loop IB-MCP, at (approximately) the same bitrate.

4.7.3 Comparisons Between Different In-band Motion Compensated Prediction Schemes

In order to elaborate on the performance of different in-band architectures, we present a comparison of several alternatives for IB-MCP. Due to the fact that the same prediction structure is used in all cases, namely closed-loop MCP following the settings on 4.7.1, any differences among the alternative schemes can safely be attributed to the spatial decomposition structure. For this reason, we do not elaborate on the same comparisons under the open-loop MCP schemes.

Figure IV-41 – Figure IV-45 present the obtained results with the closed-loop in-band MCP scheme having the following features:

- One level of in-band ME/MC ("IB-MCP, 1 level"); the settings of this codec follow the ones defined in Subsection 4.7.1, with $k = 1$.
- Two levels of in-band ME/MC ("IB-MCP, 2 level, multi-rate CODWT"); the codec settings follow the ones defined in Subsection 4.7.1, with $k = 2$; moreover, a multi-rate construction of the CODWT is performed for decomposition level two, which utilizes the DWT subbands of levels one and two; note that the total number of spatial decomposition levels remains equal to four, i.e. two additional spatial levels are performed after the two-level IB-MCP.

- Two levels of in-band ME/MC ("IB-MCP, 2 level, single-rate CODWT"); all the settings of this codec are as defined above with the exception that a single-rate construction of the CODWT is performed for decomposition level two, which utilizes only the DWT subbands of level two.

- Two levels of in-band ME/MC ("IB-MCP, 2 level, DTCWT"); all the settings are as before, with the exception that the DTCWT framework is used for in-band prediction, as described in Section 4.3; note that, the DTCWT for decomposition level two is constructed by utilizing only the subbands of level two (equivalent to the single-rate CODWT).

- Two levels of in-band ME/MC ("IB-MCP, 2 level, dual-phase ODWT"); all the settings are as for the "IB-MCP, 2 level, single-rate CODWT", with the exception that only two integer phases (equidistant) are used for decomposition level two; this ensures that this coder has similar complexity to the DTCWT-based coder defined above.

- Two levels of in-band ME/MC ("IB-MCP, 2 level, single-phase, critically-sampled DWT"); all the settings are as before, with the exception that only the zero phase is used for each decomposition level l, $1 \leq l \leq 2$; this represents the case where the DWT is used for in-band MCP, instead of the proposed overcomplete-transform based frameworks.

In this case the results with the "IB-MCP, 1 level" scheme are used as an anchor for the comparison with the results presented before in Figure IV-28 – Figure IV-32.

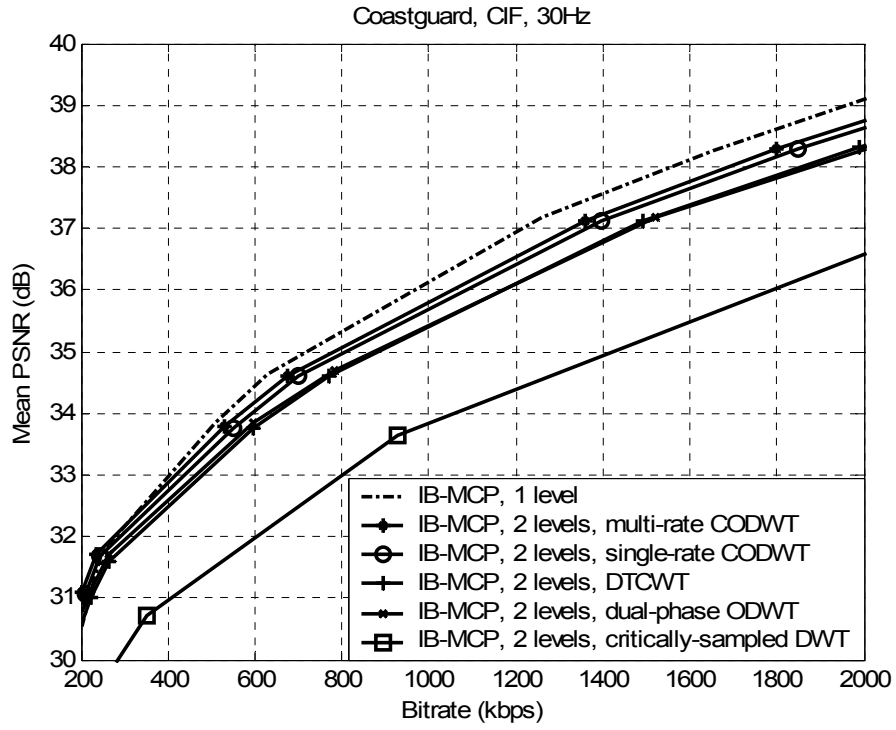


Figure IV-41. Comparison of closed-loop in-band prediction schemes for the "Coastguard" sequence.

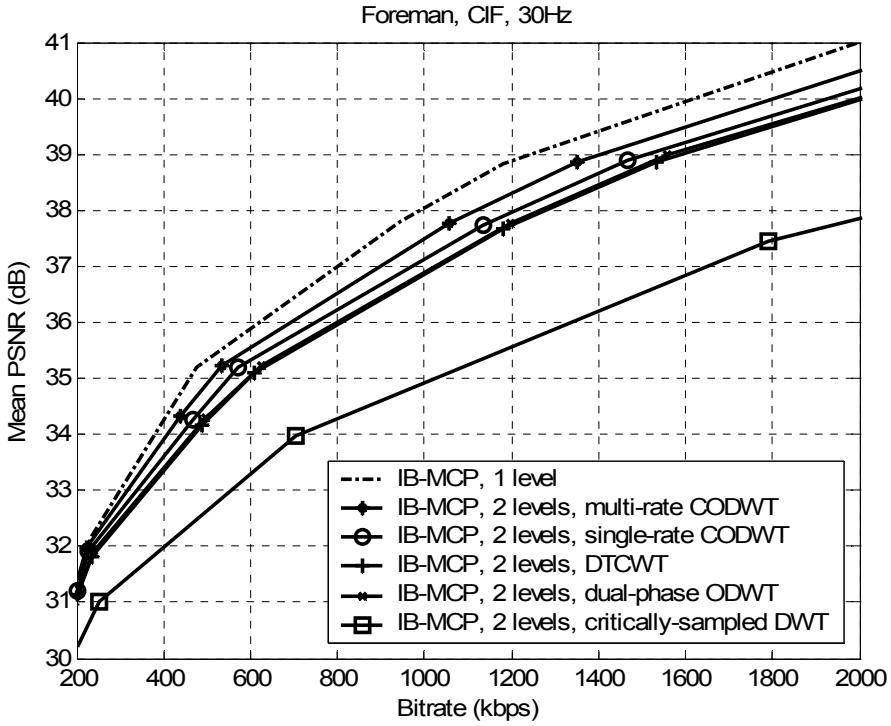


Figure IV-42. Comparison of closed-loop in-band prediction schemes for the "Foreman" sequence.

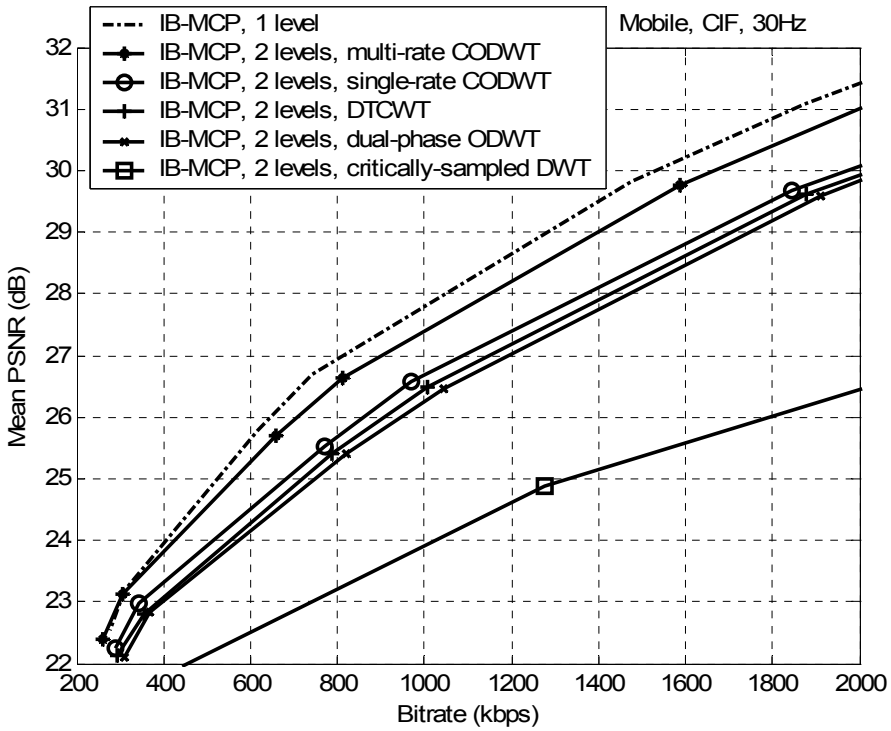


Figure IV-43. Comparison of closed-loop in-band prediction schemes for the "Mobile" sequence.

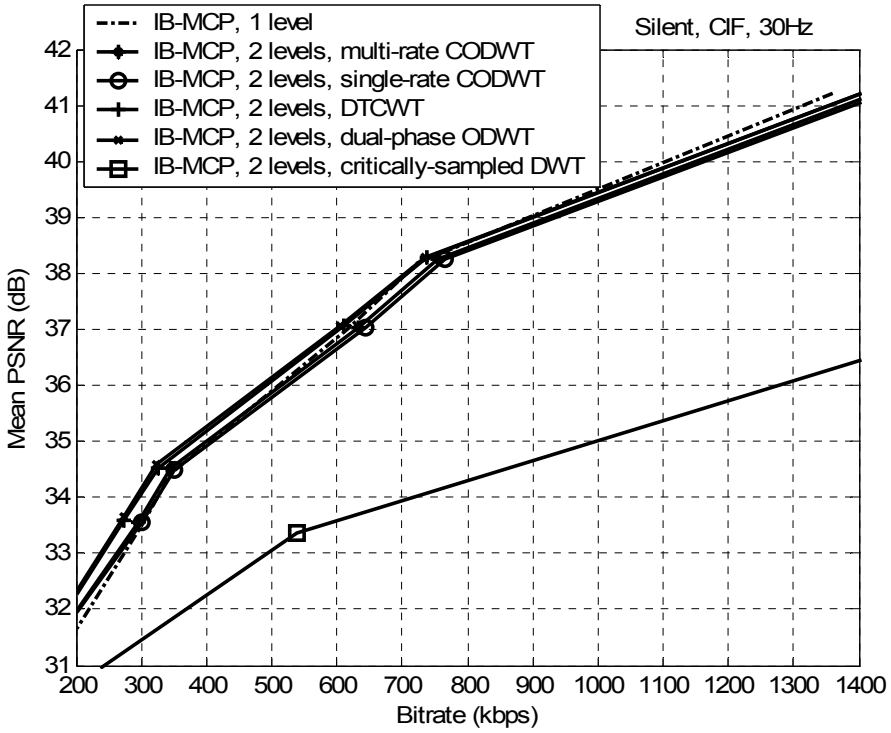


Figure IV-44. Comparison of closed-loop in-band prediction schemes for the "Silent" sequence.

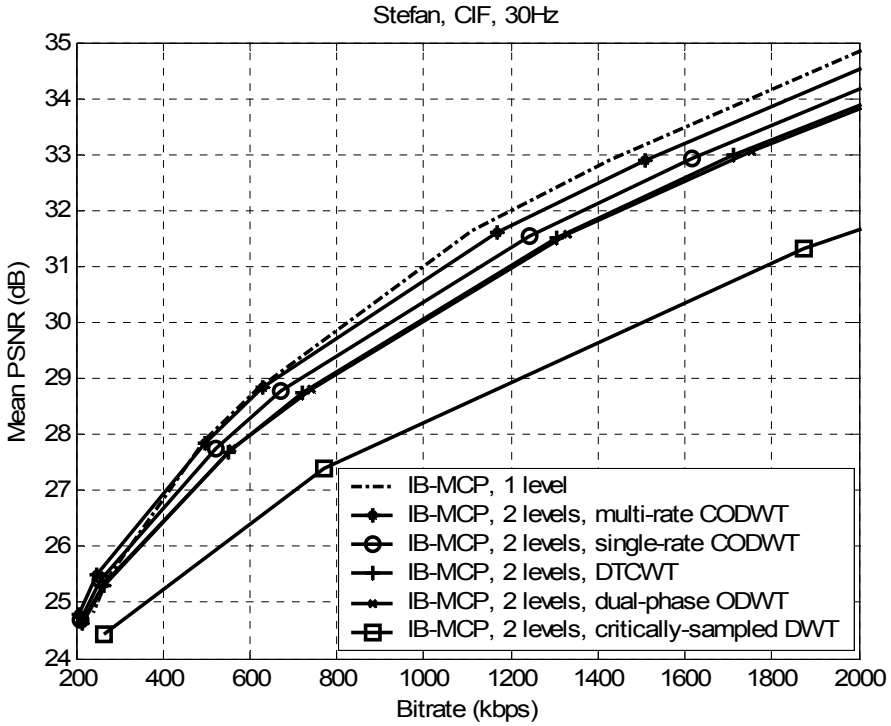


Figure IV-45. Comparison of closed-loop in-band prediction schemes for the "Stefan" sequence.

The results of Figure IV-41 – Figure IV-45 indicate that, under the experimental settings chosen for this section, the best-performing solution in terms of mean PSNR is the ODWT-based scheme with one level of in-band MCP. One exception appears to be the "Silent" sequence, where the two-level in-band MCP system appears to be superior for the majority of bitrates. Additional experiments with more in-band MCP levels confirmed that the one-level in-band MCP scheme provides, on average, superior objective performance. Nevertheless, for all the practical cases presented here, it appears that the loss incurred by the "IB-MCP, 2 level, multi-rate CODWT" scheme versus the reference "IB-MCP, 1 level" scheme is always bounded by 0.5 dB, over all bitrates. Moreover, in medium and low bitrates the loss in mean PSNR becomes negligible for this case, i.e. lower than 0.2 or 0.1 dB. Finally, we note that, in most sequences, for the low-bitrate regime, the two-level IB-MCP scheme even outperforms the equivalent one-level scheme. We attribute the observed differences in performance to the following factors:

- Our choice of block size for in-band ME/MC always requires less bitrate for the motion information in the case of a two-level in-band MCP system, versus the 1-level scheme. However, a coarser motion-vector field is produced due to the use of larger block sizes in our experimental settings for $l = 2$, which may deteriorate the system performance in the case of complex-motion scenes.

- At the decoder, the two-level IB-MCP scheme is performing two levels of inverse transform after in-band MC. Consequently a stronger "de-blocking" is performed at the decoder, as explained in the beginning of this chapter. Although this may be beneficial for the cases of irregular motion where blocking artifacts are observed, it can be disadvantageous for regular motion sequences, like "Coastguard" and "Mobile", where the block-based motion model does not create significant artifacts in the error frames.

Concerning the first point, we note that there are many choices for setting up a multi-level in-band ME/MC system. We refer to our work in [11] [79] [26] [46] for some examples and also to independent work based on similar schemes presented in [24] [41] [15] [13]. For example, as shown in the analysis of [11], if one selects the block size for all levels of the multi-level in-band ME/MC system equal to the block-size of the "IB-MCP, 1 level" scheme, then, in the worst case, the bitrate of the multi-level system is expected to be twice that of the one-level system. Although this can bring some improvement in the experimental results for certain sequences [11], this restricts the multi-level in-band MCP schemes to high bitrates (typically above 1 Mbit [11]) due to the increased motion-vector bitrate. Moreover, visual inspection of the results of [11] in comparison to the present results, demonstrated that (on average) the fixed block-size setup used in [11] produced more ringing artifacts. This is attributed to the fact that the in-band block boundaries do not coincide in the same location at the decoded frames and, overall, significant ringing artifacts can be produced after the multi-level inverse DWT at the decoder.

Concerning the second point, one can circumvent the problem of increased smoothness from the IDWT by setting an adaptive system that, for some frames, or even certain areas within frames, the coding scheme can switch to using $k = 1$ (single-level in-band MCP). An interesting variation of the proposed systems that proposes solutions in this direction can be found in [13] [81]. Nevertheless, as

confirmed by our experiments, the "IB-MCP, 2 level, multi-rate CODWT" scheme already exhibits a bounded loss in performance and can even outperform the single-level IB-MCP scheme in some cases, e.g. the "Silent" sequence. Hence, for the proposed systems, if one can adaptively switch to this scheme, this suffices for good coding performance in the vast majority of cases. Consequently, adaptive decomposition schemes do not appear to be necessary in the proposed systems, unless more in-band MCP levels are to be considered.

The experiments of Figure IV-41 – Figure IV-45 demonstrate that the remaining schemes under comparison produce inferior results against the two previously-analyzed systems. In particular, the "IB-MCP, 2 level, single-rate CODWT" coding scheme appears to underperform the equivalent multi-rate CODWT scheme by 0.1 – 0.5 dB (on average, over all bitrates). One exception to this is in the "Mobile" sequence, where a significant loss in mean PSNR is observed, close to 1.0 dB. Concerning the "IB-MCP, 2 level, DTCWT" and "IB-MCP, 2 level, dual-phase ODWT" systems, the results confirm that they perform worse than the "IB-MCP, 2 level, single-rate CODWT", but not considerably so. In detail, the loss in performance appears to be negligible for the "Mobile" and "Silent" sequences; for all the remaining sequences, the mean-PSNR loss is bounded by (approximately) 0.4 dB. Among the DTCWT-based scheme and the two-phase ODWT-based scheme, we conclude that there is no favourite with respect to mean-PSNR performance. In fact, for all sequences, the two schemes appear to be comparable. This indicates that, instead of using a different transform system, such as the dual-tree complex wavelet transform, one can obtain the same objective performance in the proposed video coding schemes by simply using two equidistant phases of the ODWT. We note that the two schemes are identical for the first level of IB-MCP since the DTCWT consists of the 9/7 filter-bank in this case. Furthermore, the two additional spatial decomposition levels performed after the two-level IB-MCP, also utilize the same filters (9/7 filter-bank).

The last system under comparison in Figure IV-41 – Figure IV-45 ("IB-MCP, 2 level, single-phase, critically-sampled DWT" scheme) appears to perform considerably worse from all the previous scenarios. This justifies the proposed frameworks that involve the use of overcomplete wavelet transforms for in-band motion estimation and compensation. Nevertheless, this system has the minimum complexity out of all schemes under comparison. Consequently, it may be favoured in certain cases where very limited computation resources are available.

All the systems involved in the comparisons of Figure IV-41 – Figure IV-45 present the same behaviour in terms of fluctuations in PSNR across time, due to the fact that only the spatial-transform system is changed and not the temporal prediction structure. An example of this is given in Figure IV-46, where, at (approximately) the same bitrate, almost all PSNR curves coincide[2]. Finally, in terms of visual quality, our overall observations agree with the objective comparisons of all systems. In particular, noticeable differences can be observed mainly among the single-phase ODWT system and the remaining coders, and, in some cases, among the "IB-MCP, 2 level, multi-rate CODWT" and "IB-MCP, 2 level, single-rate CODWT" systems.

[2] In the results of Figure IV-46, the "IB-MCP, 2 level, single-phase, critically-sampled DWT" scheme is not present since, as presented in Figure IV-41 – Figure IV-45, its performance is significantly inferior from all other cases.

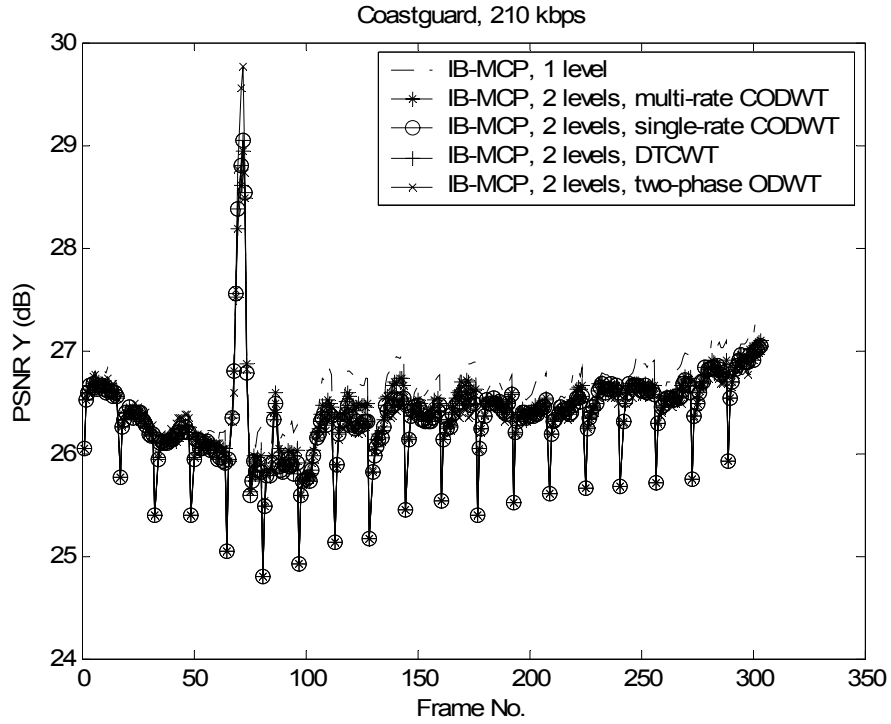


Figure IV-46. An example of a frame-by-frame comparison between various in-band motion-compensated prediction systems, at (approximately) the same bitrate.

4.7.4 Experimental Evaluation of Tools for Open-loop Architectures

This section evaluates the performance and usability of the tools proposed in Subsections 4.4.2 and 4.4.3, i.e. the proposed standard-compatible base-layer architectures for in-band motion compensated prediction and the distortion-control algorithm for open-loop spatial-domain and in-band MCP. We note that, for the comparisons of this subsection, we focused on representative video sequences and summarize the behaviour of each tool; moreover, since all the basic architectures have been experimentally compared previously, the present evaluation will not be as expansive as before.

4.7.4.1 Performance of Open-loop In-band Motion Compensated Prediction with Standard-compliant Base-layer

In our tests, we used an MPEG2/H263-alike closed-loop motion-compensated DCT scheme [88] as the base-layer coding scheme for the *LL* subbands (half-resolution). For the motion estimation of the MC-DCT coder, a block-size of 8x8 was used, and interpolation to half-pixel was performed directly given the coefficients of the critically-sampled *LL* subbands. A standard rate-control mechanism ensured the target bitrate for the non-scalable MC-DCT coder and it was set to 200 kbps. For the spatial transform, the 9/7 filter-bank was used with one decomposition level. The open-loop MCP used bidirectional ME/MC for the temporal decomposition [73, 74] with half-pixel accurate motion

estimation and a block-size of 8×8 coefficients. Due to the separate performance of ME/MC in the high-frequency subbands, for the CIF resolution the base-layer bitrate is increased to 300 kbps. The QuadTree-Limited embedded image coder [71] was used for the compression of the intra and error frames of the enhancement layer. In this coder, the quality of the decoded output of the enhancement layer of each of the two resolutions was controlled by decoding all frames to the same integer or fractional bitplane.

The average PSNR for various bitrates obtained with the architecture of Figure IV-17 (with the standard-compliant MC-DCT scheme), is shown by the "Closed-loop MC-DCT base-layer" curves of Figure IV-47 for the first 64 frames of the sequences "Coastguard" and "Foreman". Although this scheme is scalable in frame-rate/resolution/SNR and the base-layer coding is a (closed-loop) standard-compliant MC-DCT, for bitrates above the base-layer (300 Kbps), it can be seen that this architecture is not efficient at CIF resolution in comparison to the non-scalable equivalent MC-DCT coder that encodes the CIF input sequence. However, for the QCIF resolution, identical results are obtained with the two schemes at the base-layer bitrate of QCIF (200 Kbps), as shown in Table IV-II; additionally in a limited range around the low bitrate region of the CIF resolution (300-450 Kbps) the scalable scheme performs adequately.

This loss in SNR scalability is considerably reduced by the architecture of Figure IV-18 (open-loop MC-DCT, base-layer at 200 Kbps for the DCT-based compression), as indicated by the "Open-loop MC-DCT base-layer" curve of Figure IV-47. In this implementation, since we are not constrained with full compatibility to a standard, we use the prediction filters of [36] [33] for the performance of the half-pixel spatial interpolation in the base-layer ME/MC. Essentially, in this context, these filters produce the missing samples resulting from the downsampling process of the spatial wavelet transform. Thus, they can be perceived as the optimum interpolation technique given the fact that in the scalable coding system, for the low-frequency subband that originates from the CIF resolution, these filters alleviate completely the aliasing effect caused by the filtering and downsampling of the spatial transform. We find that part of the good coding performance of the scalable approach can be attributed to this effect.

To conclude, we discuss also the results of the open-loop MC-DCT architecture when decoding at QCIF resolution, versus the non-scalable MC-DCT standard architecture that encodes separately the uncoded low-frequency subband as an original sequence. These results are seen in Table IV-II. It appears that the proposed MC-DCT architecture obtains good PSNR performance in comparison to the non-scalable, standard, MC-DCT coding. The gain in performance comes both from the different prediction structure and from the improved interpolation process for the reference frames.

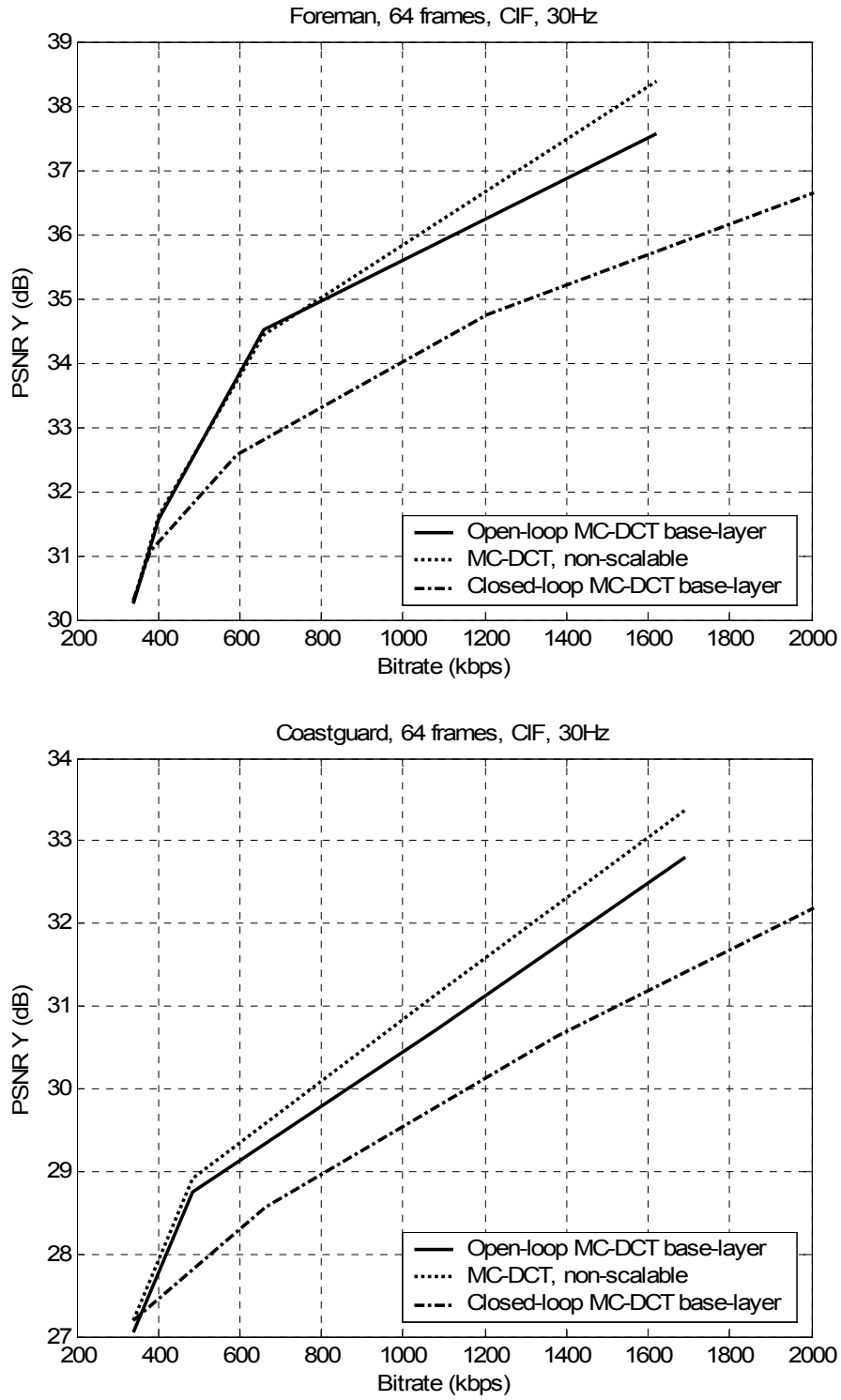


Figure IV-47. Results with standard-compliant MC-DCT schemes within open-loop IB-MCP for CIF resolution.

Foreman, 64 frames, QCIF, 30 Hz	Open-loop MC-DCT base-layer	MC-DCT non-scalable	Closed-loop MC-DCT base-layer
Bitrate (kbps)	PSNR Y (dB)	PSNR Y (dB)	PSNR Y (dB)
200	32.27	31.67	31.66
280	33.96	33.08	32.43
540	37.53	36.10	34.22
Coastguard, 64 frames, QCIF, 30 Hz	Open-loop MC-DCT base-layer	MC-DCT non-scalable	Closed-loop MC-DCT base-layer
Bitrate (kbps)	PSNR Y (dB)	PSNR Y (dB)	PSNR Y (dB)
200	29.93	29.29	29.29
340	31.42	31.07	30.41
580	33.83	33.24	32.37

Table IV-II. Results with standard-compliant MC-DCT schemes within closed-loop IB-MCP for QCIF resolution.

4.7.4.2 Performance of the Distortion-control Algorithm for Spatial-domain and In-band Motion Compensated Prediction

In order to evaluate the proposed control scheme, we incorporated the basic steps of the algorithm of Figure IV-22 in the spatial-domain and in-band MCP schemes. In our experiments, the Foreman sequence was used as a typical example of a video sequence that involves complex motion with fast and slow moving regions.

Concerning the codec settings, the experimental conditions of Subsection 4.7.1 were used, with the following differences: (a) only three levels of the spatial transform were performed for each frame; (b) in the case of IB-MCP, in-band ME/MC was performed for each spatial level, with fixed block size equal to 8×8 wavelet coefficients (for all subbands of each level); (c) the ME/MC was bidirectional, using the previous and next frame of each temporal decomposition level. For the bitstream-parsing stage, instead of the rate-control described in Subsection 4.7.1.5, the distortion values produced during encoding for the A_0^4 were used and the corresponding distortions for the H_n^t frames, $1 \leq t \leq 4$ were constructed, based on equations (4.30), (4.31). After this process, the total bitrate was measured by truncating the bitstream of each compressed frames based on the derived (distortion-controlled) bitstream-truncation points. Hence, the last step of Figure IV-22 that shifts the distortion control into a rate control was omitted from this evaluation.

Although the parameter a can be specified separately per GOP, we chose two cases, $a = 0.2$ and $a = 1.0$ for the entire sequence. Moreover, as explained in Subsection 4.4.3, the proposed distortion-control algorithm was applied in the bitstream-parser stage of the SD-MCP and IB-MCP systems; as a result, no modification is required for the encoding part and the functionality of bitrate-scalability of both schemes is not affected. The resulting mean PSNR for the SD-MCP decoding is shown in Figure IV-48. Additionally, Figure IV-49 presents the PSNR variation of the Y channel for the SD-MCP

after decoding at a low bitrate. The corresponding results for the IB-MCP system are presented in Figure IV-50 and Figure IV-51, respectively. Our experiments demonstrate that the variation in the achieved PSNR over the sequence can be controlled, and it appears to be significantly reduced if small values for the control parameter a are used, as theoretically expected. This however comes at the expense of loss in the average PSNR, which becomes significant for decoding at high bitrates. It is observed however that, at low rates, the loss in the mean PSNR is limited. By combining this result with the observation that the PSNR variations are mainly visible in low bitrate decoding, we reach the conclusion that the proposed method can be beneficial for controlling the PSNR variation in the decoded video at low bitrates. The selective application of the proposed control mechanism for a certain bitrate range is possible, since the entire process takes place at the bitstream-extraction stage and does not affect encoding or decoding.

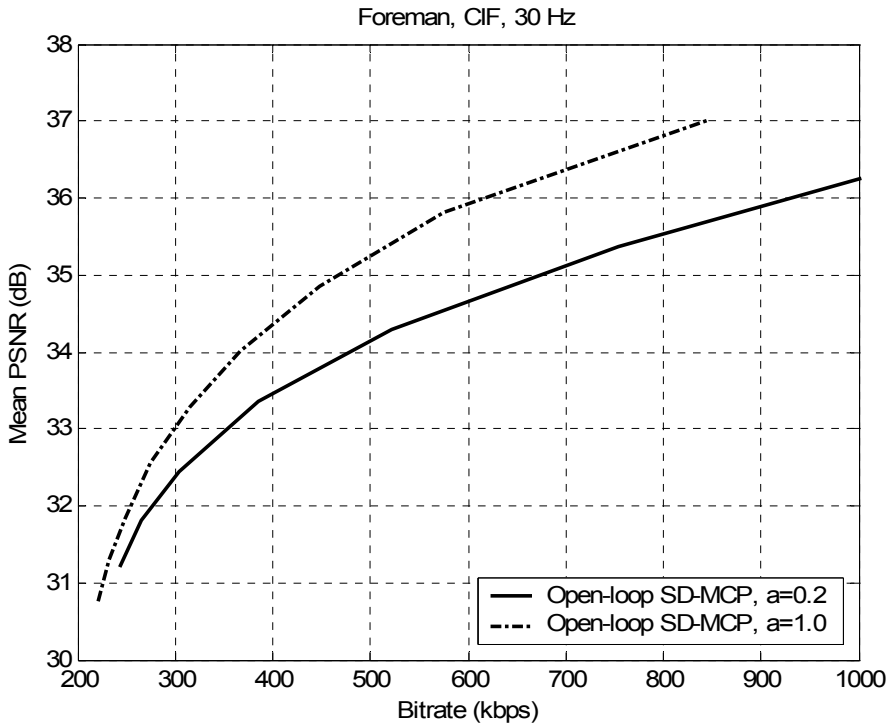


Figure IV-48. The mean PSNR for two parameter settings in the control-mechanism implemented in the spatial-domain MCP system.

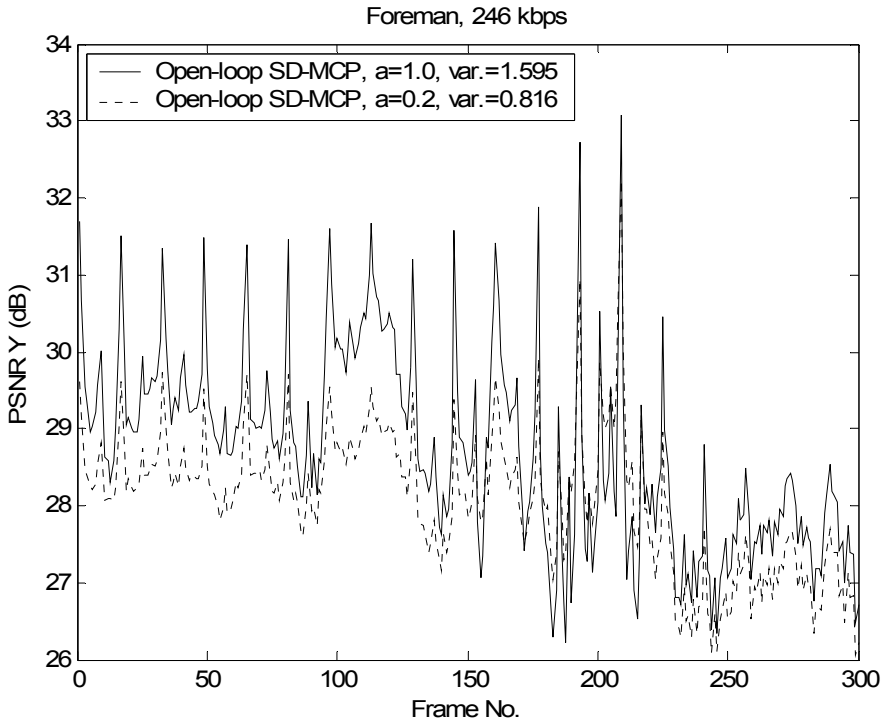


Figure IV-49. The PSNR variation (Y channel) for the SD-MCP system at a low-bitrate setting (approximately 246 kbps).

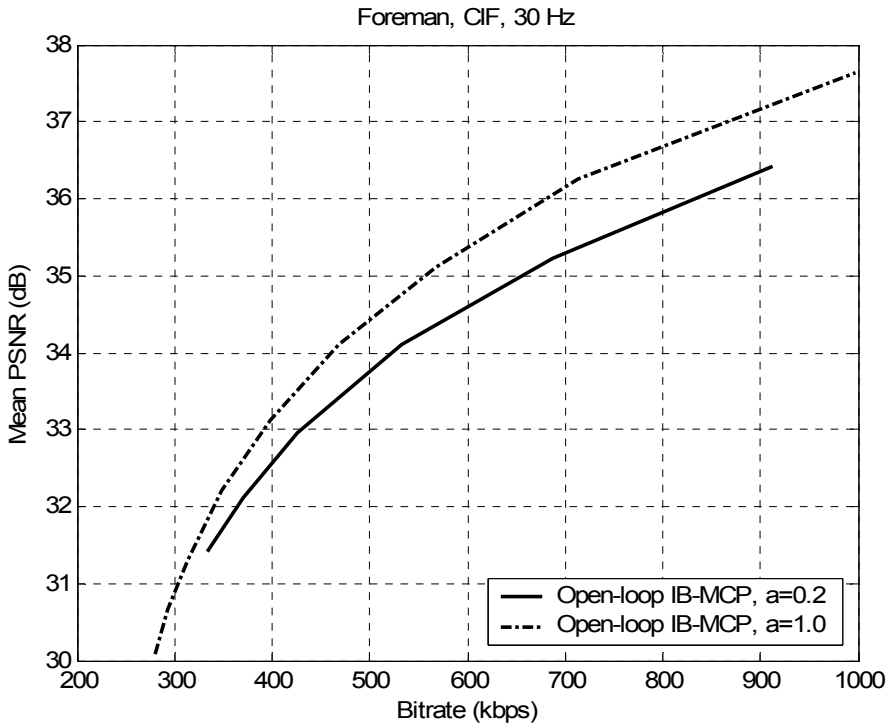


Figure IV-50. The mean PSNR for two parameter settings in the control-mechanism implemented in the in-band MCP system.

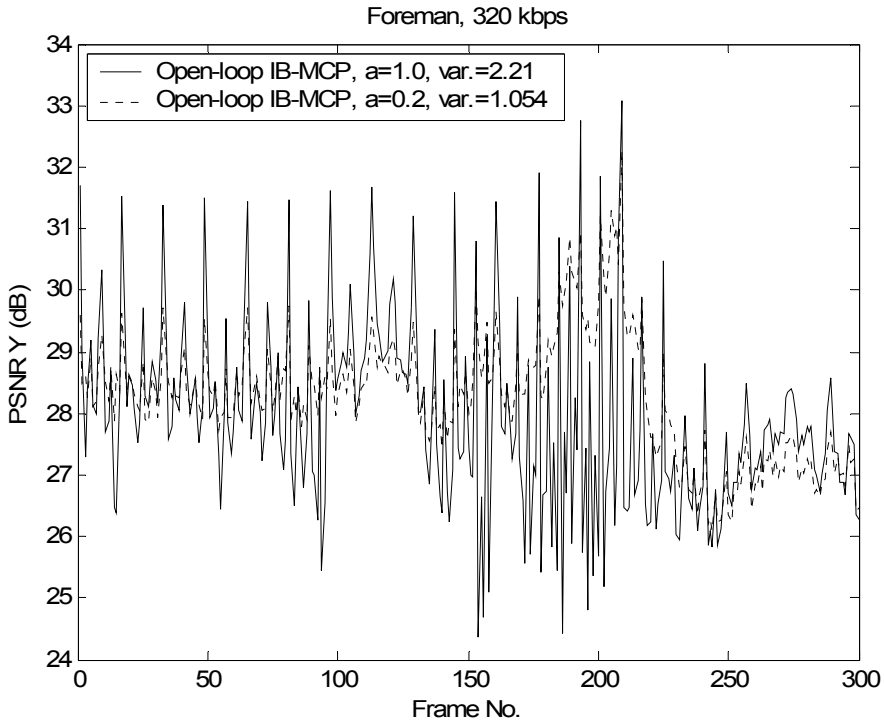


Figure IV-51. The PSNR variation (Y channel) for the IB-MCP system at a low-bitrate setting (approximately 320 kbps).

4.7.5 Performance of Advanced Motion Estimation for Spatial-domain and In-band Motion Compensated Temporal Filtering

For all our experiments in this subsection, we use four temporal decomposition levels. The lifting-based implementation of the temporal transforms (Haar or 5/3) is described in Section 4.5. For the case of the 1/2 and 1/3 filters (i.e. prediction-only without update), the frames of each temporal level t remain the original input frames, scaled by $(\sqrt{2})^{t-1}$.

The variable block-size multihypothesis motion estimation algorithm of Section 4.6 is used. For the case of IBMCTF, block-based motion estimation with a block size of $B_m \times B_n$ pixels in the spatial domain corresponds to k separate motion estimations (for a total of k spatial-resolution levels of in-band motion estimation) with triplets of blocks of $\frac{B_m}{2^l} \times \frac{B_n}{2^l}$ wavelet coefficients in the high-frequency subbands of each resolution l, $1 \le l < k$, as described in Section 4.6. The number of hypotheses used was set to 1 or 2, and this always corresponds to temporal multihypothesis prediction. If multiple in-band motion estimations are performed ($k > 1$), the number of vectors corresponding to each spatial location in the IBMCTF codec varies according to the temporal filtering of each resolution.

We applied the proposed MCTF technique with a different number of features enabled for each experiment in order to assess the impact on the coding efficiency of MCTF. The results are given in Figure IV-52 – Figure IV-56 for the IBMCTF and SDMCTF. Since no attempt was made to jointly

optimize the prediction performance across resolutions, in the experiments of this subsection we set $k = 1$ (one level of in-band MCP and MCU); three additional levels of spatial transform are performed in the MCTF residuals of the LL subbands in order to match the SDMCTF codec in the number of spatial decomposition levels (four). The objective comparison shows that, in both the SDMCTF and IBMCTF architectures, a large PSNR gain comes from the use of multihypothesis and longer temporal filters (5/3). Additionally, the use of the update step improves the objective performance, especially at high-bitrates. Furthermore, both architectures appear to produce comparable results for full-resolution decoding across the entire range of bitrates.

Concerning implementation complexity of the IBMCTF and SDMCTF encoders, we found that the use of all the advanced tools for the motion estimation (macroblocks pruned using 5 partition levels and 2 hypotheses) incurs a penalty of a factor of 4-6 times increase in execution time as compared to conventional block-based full search motion estimation. This result corresponds to execution-time comparisons of our platform-independent "C" implementation running on an Intel Pentium IV processor. However, preliminary testing indicates that several optimizations for motion-vector search using spiral or diamond search patterns can decrease this complexity factor significantly. In addition, the effect of the advanced prediction tools is much less severe for decoding. Our experiments indicate that only an increase by a factor of 2 is observed. Finally, the IBMCTF encoder using the proposed prediction-filters approach for the CODWT runs (on average) for approximately 150% of the SDMCTF encoding time with the same settings; IBMCTF decoding runs, on average, about 3 times slower than SDMCTF decoding.

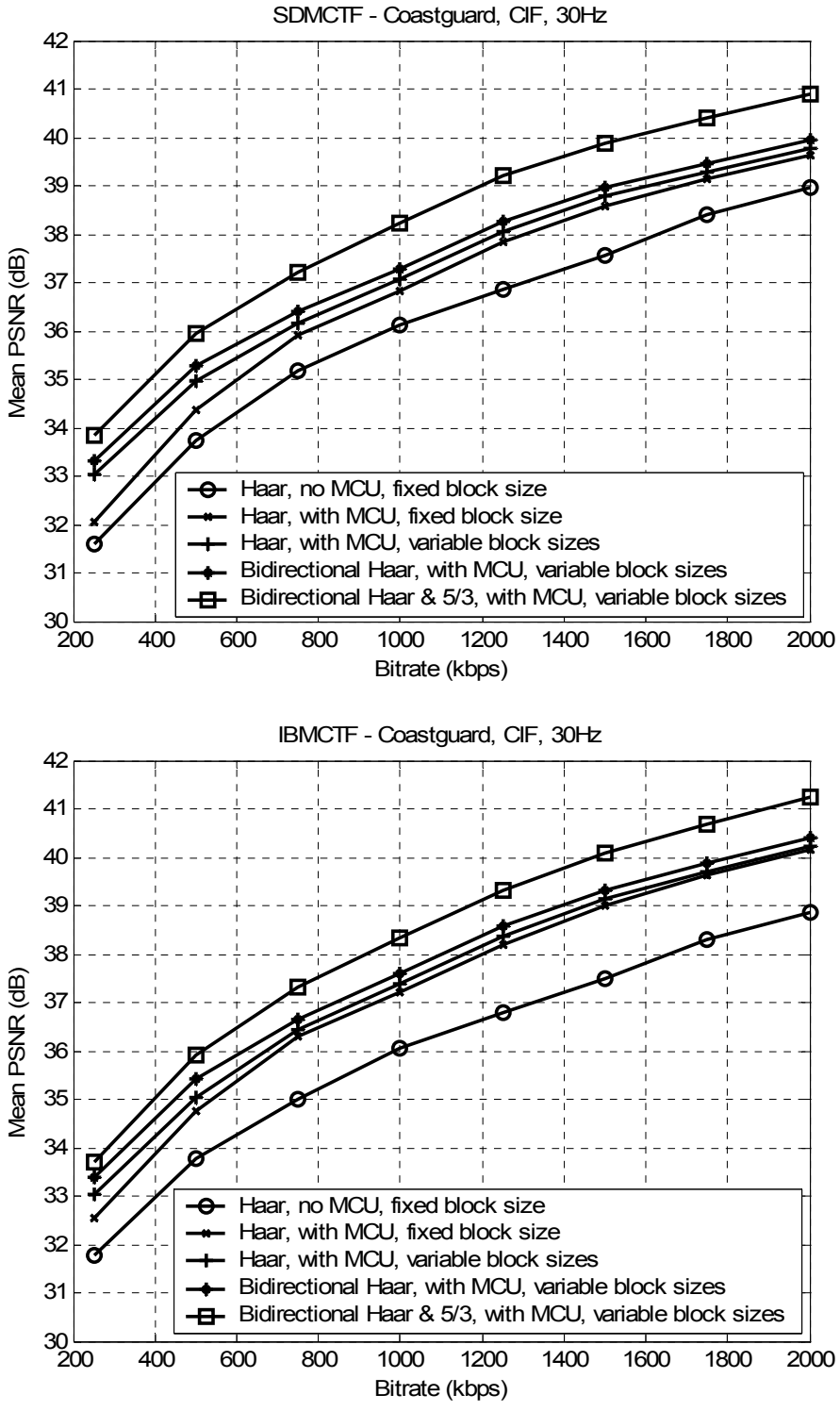


Figure IV-52. Comparison of different MCTF schemes for the "Coastguard" sequence in both spatial-domain and in-band frameworks.

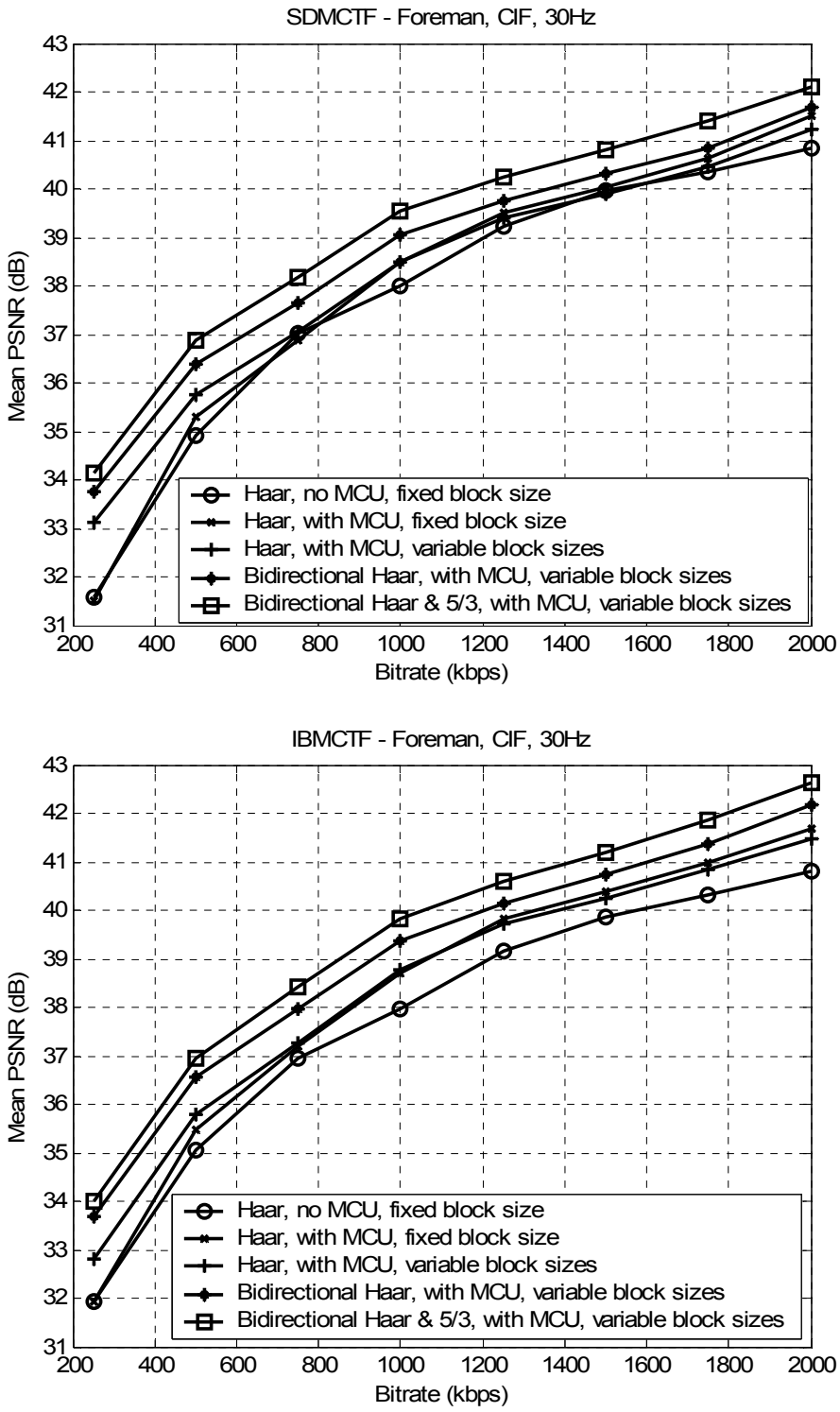


Figure IV-53. Comparison of different MCTF schemes for the "Foreman" sequence in both spatial-domain and in-band frameworks.

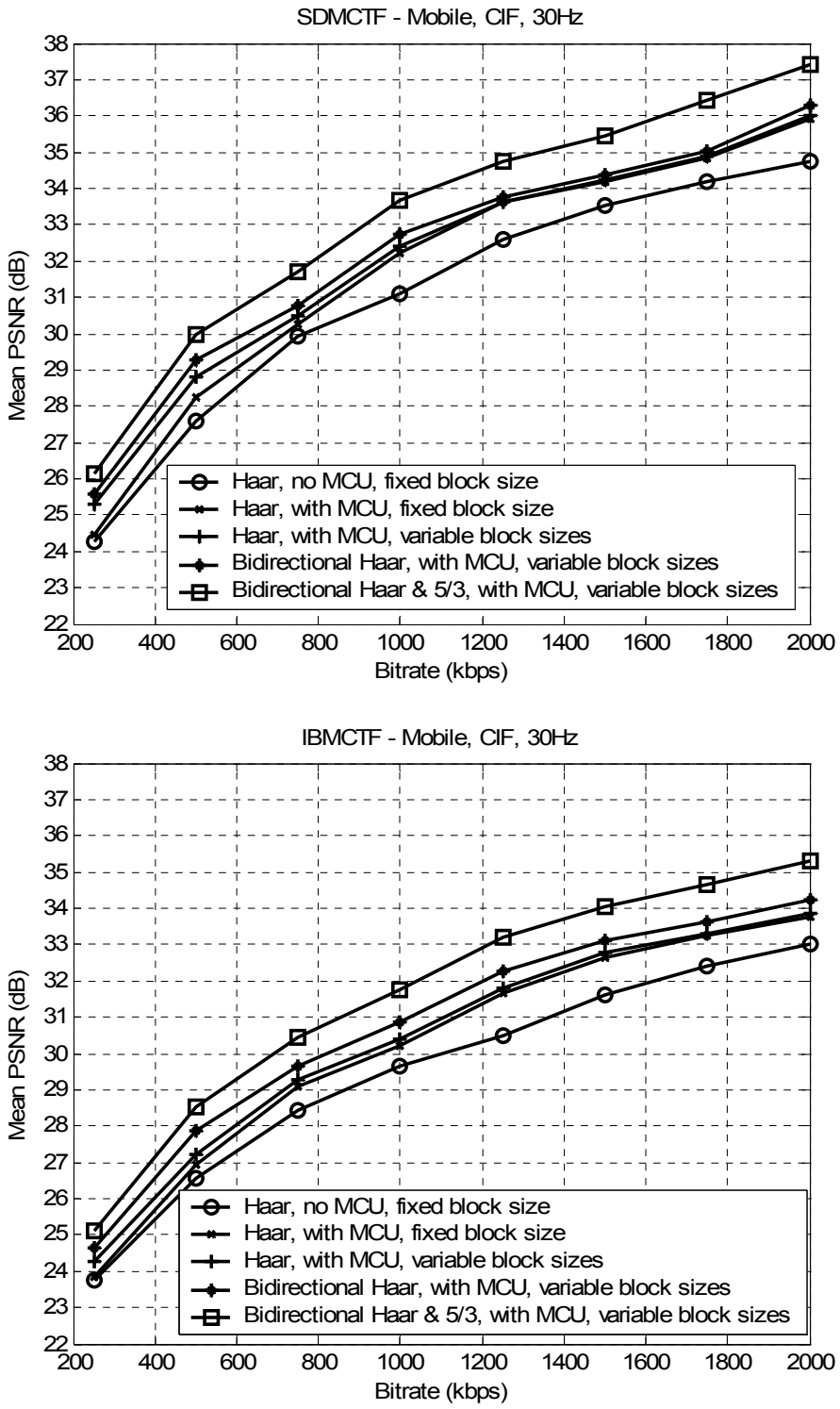


Figure IV-54. Comparison of different MCTF schemes for the "Mobile" sequence in both spatial-domain and in-band frameworks.

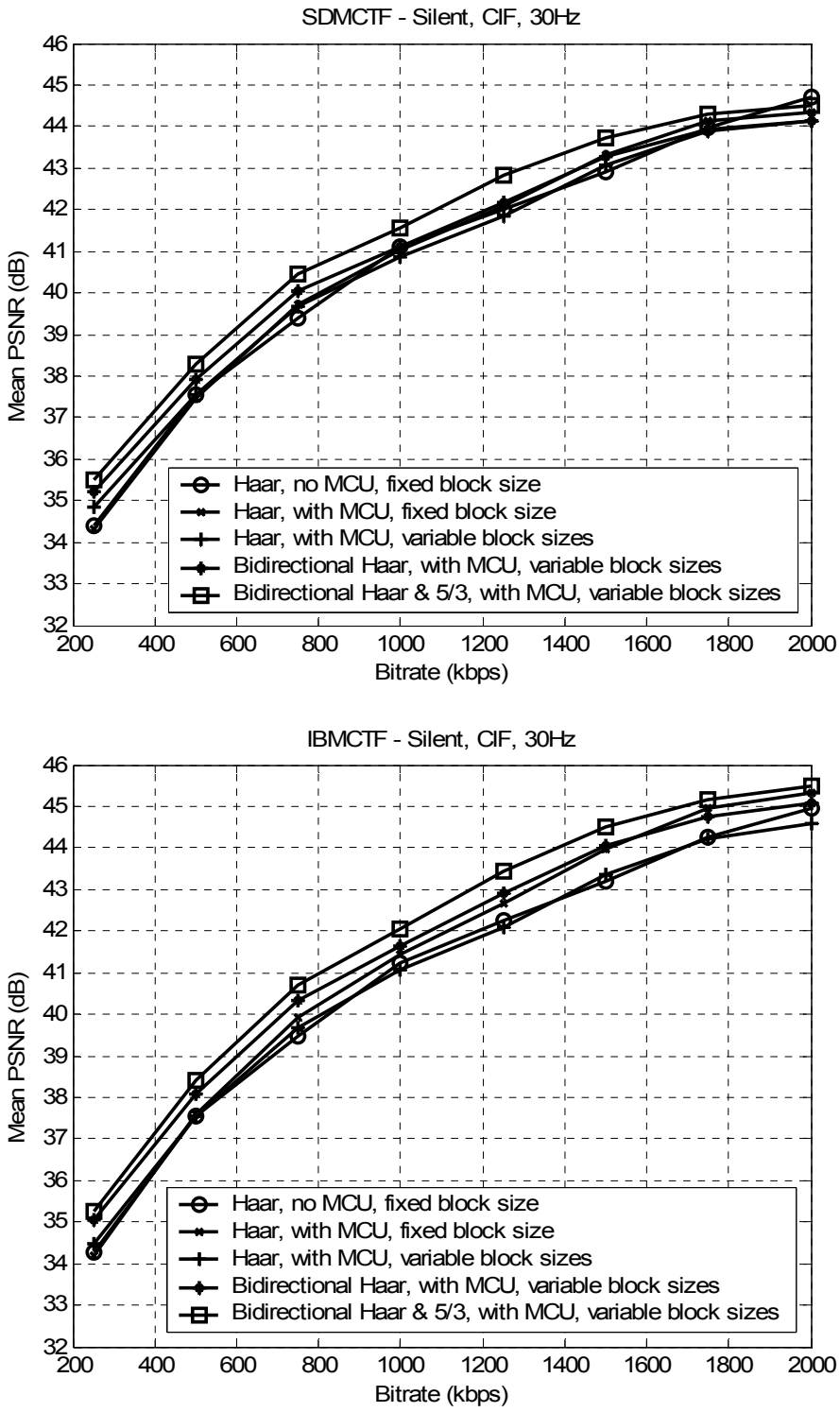


Figure IV-55. Comparison of different MCTF schemes for the "Silent" sequence in both spatial-domain and in-band frameworks.

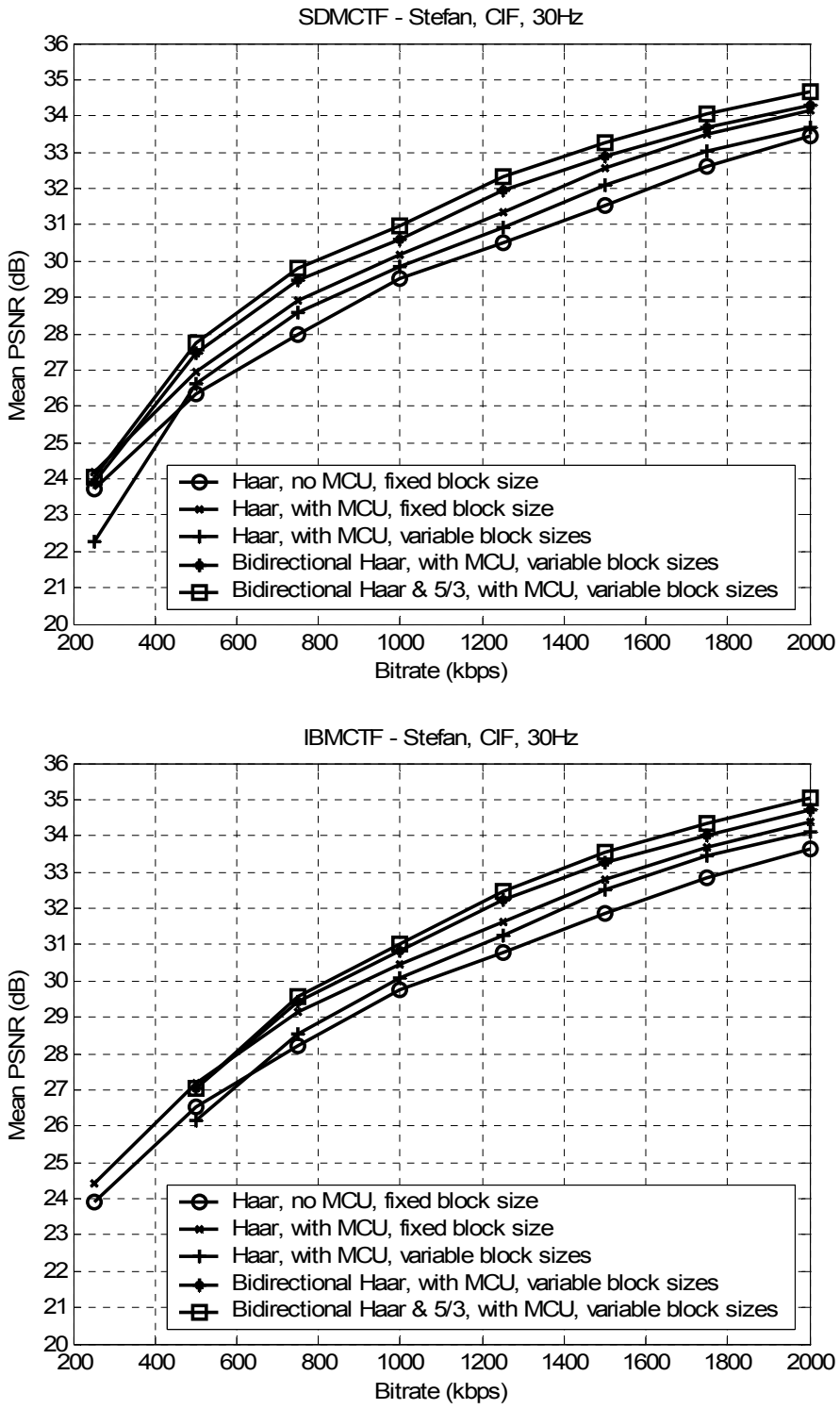


Figure IV-56. Comparison of different MCTF schemes for the "Stefan" sequence in both spatial-domain and in-band frameworks.

4.7.6 Comparison for Low-resolution Decoding

We present an evaluation for the quality offered for low-resolution decoding in both spatial-domain and in-band architectures. Our focus is on the open-loop architectures, since previous results demonstrated that these architectures can be fully-scalable, while producing comparable results with the equivalent closed-loop coders when operating in non-scalable mode.

It is generally difficult to establish an objective metric for evaluation under temporal and resolution scalability since, under the MCTF framework, every coder creates its own unique reference sequence for low-resolution/low frame-rate video. Nevertheless, the ability of that reference sequence to serve as a unique original is questionable. In general, both the spatial decomposition and the motion model used during the temporal transform seem to affect the results of low-resolution decoding. In total, we opted for a visual evaluation of the decoded results at half resolution (from CIF to QCIF resolution). For the temporal transform, the variable-block size algorithm of section 4.6 was used but we set $M = 1$ and only one reference frame was used. In this way, we evaluate the results for low-resolution decoding with the typical motion-compensated Haar temporal transform. Instead of performing the evaluation at certain bitrates, the lack of a common reference for both codecs led us to decode at a very high bitrate, which corresponds to the "original" sequence generated by each codec at low resolution, and compare the results visually. All potential artifacts observed in this setting are not related to coarse quantization, but to the motion mismatch between full and half resolution.

Typical output frames for the SDMCTF and IBMCTF decoding at QCIF resolution are shown in Figure IV-57 – Figure IV-59. The results demonstrate that several artifacts are generated in the SDMCTF-based schemes. The presence of artifacts in these cases depends on the existence of MCU: stronger artifacts appear in SDMCTF schemes with MCU is used. Notice that this is true even though the decoding occurred at 30 Hz, i.e. the original frame-rate. The practical cause of the observed artifacts in SDMCTF is the application of the motion compensation at reduced resolution at the decoder (by performing vector subsampling), which causes a mismatch between encoder and decoder. It appears that this mismatch manifests stronger artifacts when the utilized motion model involves MCU. Moreover, it is observed that picture quality tends to be particularly bad in frames with fast and irregular motion.

In the case of IBMCTF, Figure IV-57 – Figure IV-59 present visual results with two example instantiations with MCP and MCU for the temporal decomposition and one or two in-band MCTF levels in the spatial decomposition. We note that the two-level IBMCTF system used the single-rate CODWT, which does not utilize any information from higher-resolution subbands. It is observed that the "decoder reference" picture quality at low resolution is always superior from the equivalent SDMCTF systems. Moreover, we observed that the presence of MCU in the temporal decomposition does not appear to affect the results in a noticeable way. Finally, the decomposition utilizing two in-band MCTF levels appears to be the best for low-resolution decoding among the two IBMCTF alternatives. However, as shown in the results of Figure IV-41 – Figure IV-45, this scheme is expected to lead, on average, to a higher loss in coding efficiency for the full-resolution decoding.

The above conclusions were also verified for various bitrates, ranging from 100 – 1000 kbps for QCIF resolution [12], by visual inspection of the decoding results of the schemes under comparison. As a

general conclusion, concerning spatial scalability with two-resolutions, it appears that, although small artifacts may sporadically appear, the single-level IBMCTF scheme provides the best trade-off for good resolution-scalability and equivalent objective performance at full-resolution decoding with SDMCTF. Nevertheless, for a scalable coder supporting more that two video resolutions, two in-band MCTF levels, or higher, may be justified.

Figure IV-57. "Stefan" sequence decoded at a very high bitrate in QCIF (30 Hz). From top to bottom, left to right: SDMCTF (with MCU); SDMCTF (without MCU); IBMCTF (with MCU), 1 in-band ME/MC level, IBMCTF (with MCU), 2 in-band ME/MC levels.

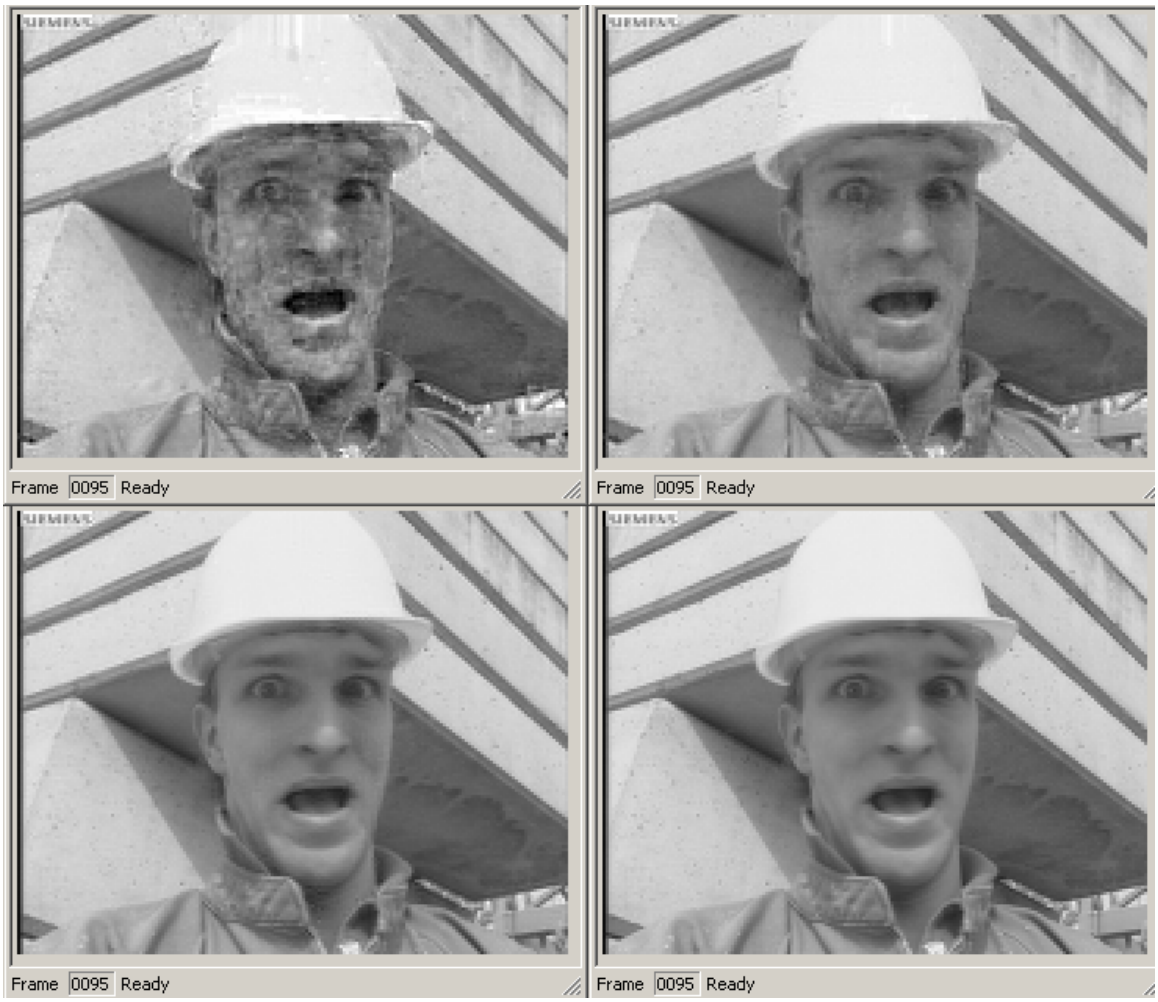


Figure IV-58. "Foreman" sequence decoded at a very high bitrate in QCIF (30 Hz). From top to bottom, left to right: SDMCTF (with MCU); SDMCTF (without MCU); IBMCTF (with MCU), 1 in-band ME/MC level, IBMCTF (with MCU), 2 in-band ME/MC levels.

Figure IV-59. "Silent" sequence decoded at a very high bitrate in QCIF (30 Hz) resolution. From top to bottom, left to right: SDMCTF (with MCU); SDMCTF (without MCU); IBMCTF (with MCU), 1 in-band MCTF level, IBMCTF (with MCU), 2 in-band MCTF levels.

4.7.7 Comparisons with the State-of-the-art in Non-scalable Video Coding

In order to compare the best instantiation of the open-loop SDMCTF and IBMCTF video coding schemes against the state-of-the-art, following recent MPEG tests [89], we chose seven 4:2:0 test sequences with 704×576 pixels in the luminance channel and a replay frame-rate of 60 Hz. The sequences were downloaded from the MPEG test repository ftp://ftp.tnt.uni-hannover.de. Most sequences come from originally larger video clips that have been cropped to the specific size for test purposes. For all cases, 576 frames were encoded. We opted for this scenario for our tests since it involves medium-resolution video content, which is currently possible to display even in TFT screens that appear in PDAs and cell-phones. Hence, we believe that this content encapsulates better the potential future markets of such technologies than CIF-resolution video. Moreover, due to the

scalability properties of the open-loop schemes, one can always generate smaller resolutions of the input content post-encoding (e.g. CIF-resolution video at 30 Hz).

Table IV-III illustrates the coding results obtained with the spatial-domain and in-band MCTF using the proposed multihypothesis prediction and update step within the 5/3 temporal decomposition, which, as demonstrated before, produce the best results from all the possible experimental settings for our codecs. We compare with the current state-of-the-art in non-scalable video coding, i.e. the MPEG-4/ITU-T VCEG AVC/H.264 codec. The AVC results were produced with the settings described in [80], using the jm4.2 software package of the codec. These settings correspond to the codec profile that includes all the advanced features like rate-distortion optimization, the CABAC scheme and full-search variable block-size motion estimation that utilizes five reference frames. To enable random-access capabilities, an intra frame was imposed every 32 frames; moreover, three B-frames were used between successively-predicted P-frames. Finally, we note that the search range was set to ± 32 pixels instead of ± 64 mentioned in [80]; this enabled the simulations with this codec to run in a reasonable amount of time. All the bitrates presented in Table IV-III were obtained with AVC by separate encoding and decoding with quantization settings 24, 27, 30 and 33 and measuring the size of the compressed bitstream. Subsequently, the scalable coders under comparison extracted and decoded bitstreams that correspond to these bitrates from one compressed bitstream; the accuracy that each target bitrate was matched was always within 1%.

Concerning the proposed scalable codecs, for the SDMCTF and IBMCTF we used 1/8 and 1/4-pixel accurate motion estimation with full-search, respectively. This difference in the interpolation precision enabled both codecs to run in (roughly) the same amount of execution time, which was comparable to the AVC encoding time. We note that, although 1/8-pixel accurate motion estimation would benefit the IBMCTF codec, we found that the potential increase in the coding efficiency did not justify the additional encoding and decoding complexity. For both SDMCTF and IBMCTF schemes, the search range in our experiments corresponded to ± 16, ± 24, ± 32, ± 40, ± 48 pixels for temporal levels 1 to 5, respectively. This results in an average search range of approximately ± 23 pixels per frame, which is about 72% of the average search range of the motion estimation in AVC. Moreover, in all cases, only two reference frames were used within the motion estimation of the SDMCTF and IBMCTF schemes, instead of the five reference frames used in AVC. In order to provide approximately the same motion-vector bitrate for both SDMCTF and IBMCTF schemes, we set $\lambda = 62.5$ (Lagrangian multiplier) for the variable block-size ME pruning in SDMCTF (see Figure IV-27) and $\lambda = 31.25$ for the IBMCTF-based codec; it was found that these settings produced motion vector bitrates that were comparable (within 25% margin) with each other. For both MCTF-based frameworks, a maximum of $M = 2$ hypotheses were used and each macroblock was pruned with dyadically-reduced block sizes ranging from 128×128 down to 8×8 pixels (64×64 to 4×4 wavelet coefficients at each subband (with $k = 1$) for the IBMCTF scheme). The 9/7 filter-bank was used for the spatial decomposition (5 resolution levels). In the temporal direction, according to the pruning of the motion information during the predict step, the bidirectional Haar and the 5/3 filter-banks were used. An implementation with a sliding window was chosen to avoid large PSNR fluctuations at the GOP borders.

ShuttleStart				
Codec / Bitrates (kbps):	**475**	**753**	**1335**	**2387**
SDMCTF/Mean PSNR (dB):	41.32	42.64	43.83	44.46
IBMCTF/Mean PSNR (dB):	41.78	43.28	44.68	45.52
MPEG-4 AVC/Mean PSNR (dB):	40.38	41.81	43.25	44.72
Raven				
Codec / Bitrates (kbps):	**1010**	**1651**	**3041**	**5438**
SDMCTF/Mean PSNR (dB):	38.37	39.90	41.52	42.59
IBMCTF/Mean PSNR (dB):	38.47	40.20	41.99	43.23
MPEG-4 AVC/Mean PSNR (dB):	38.33	39.86	41.47	43.11
Soccer				
Codec / Bitrates (kbps):	**1909**	**3001**	**5250**	**9246**
SDMCTF/Mean PSNR (dB):	35.76	37.33	38.96	40.79
IBMCTF/Mean PSNR (dB):	35.71	37.47	39.25	41.22
MPEG-4 AVC/Mean PSNR (dB):	37.01	38.60	40.30	42.13
City				
Codec / Bitrates (kbps):	**1202**	**2148**	**4869**	**10865**
SDMCTF/Mean PSNR (dB):	36.35	38.10	39.80	41.11
IBMCTF/Mean PSNR (dB):	36.61	38.41	40.24	41.74
MPEG-4 AVC/Mean PSNR (dB):	36.17	37.70	39.41	41.37
Crew				
Codec / Bitrates (kbps):	**2121**	**3451**	**6245**	**11215**
SDMCTF/Mean PSNR (dB):	35.83	37.37	38.81	40.10
IBMCTF/Mean PSNR (dB):	35.72	37.42	38.96	40.44
MPEG-4 AVC/Mean PSNR (dB):	37.09	38.49	39.99	41.60
Harbour				
Codec / Bitrates (kbps):	**2969**	**5707**	**11092**	**17909**
SDMCTF/Mean PSNR (dB):	37.19	38.84	40.47	41.92
IBMCTF/Mean PSNR (dB):	37.53	39.30	41.11	42.84
MPEG-4 AVC/Mean PSNR (dB):	35.75	37.49	39.5	41.33
Sailormen				
Codec / Bitrates (kbps):	**1368**	**2727**	**5708**	**12440**
SDMCTF/Mean PSNR (dB):	35.62	37.75	39.40	40.68
IBMCTF/Mean PSNR (dB):	35.01	37.08	38.67	40.36
MPEG-4 AVC/Mean PSNR (dB):	35.31	36.85	38.50	40.53

Table IV-III. Comparison of SDMCTF and IBMCTF-based schemes in terms of mean PSNR against the state-of-the-art in non-scalable video coding. Seven standard-definition sequences are used (704×576 pixels, 4:2:0, 576 frames, replay rate 60 Hz).

The results of Table IV-III show that, for the majority of the sequences of the test, multihypothesis SDMCTF-based and IBMCTF-based video coding is comparable or superior in terms of mean PSNR to the highly-optimized AVC over a large range of bitrates. A significant loss in performance was only observed for the "Crew" and "Soccer" sequences. This is mainly attributed to the complex motion present in these sequences, which requires the use of advanced intra-prediction modes, only present in AVC. We note however that the proposed schemes retain the advantages offered by fully-embedded coding, since each sequence was compressed once and all bitrates for the results of Table IV-III were extracted from a single compressed bitstream without transcoding.

Among the two proposed alternatives, it appears that both systems perform equivalently in mean PSNR. Moreover, similar to subsection 4.7.6, the results for low-resolution decoding reveal that the IBMCTF-based system provides significantly-improved quality for low resolution. This is demonstrated in Figure IV-60 and Figure IV-61, where we display example frames of four of the sequences used in Table IV-III, when decoded at a very high bitrate with reduced resolution and frame-rate. The results demonstrate that, even with an advanced motion estimation algorithm, such as the ones used for the experiments of Table IV-III, resolution scalability is more efficient in IBMCTF-based systems.

In terms of visual quality for full-resolution decoding versus AVC, unofficial visual testing revealed that the proposed codecs provide comparable or superior visual quality in cases of scenes with complex textures (e.g. the "Harbour" and "Raven" sequences). However, AVC [90] appeared to provide improved visual quality in the majority of the test sequences by reducing blocking artifacts and flickering based on its in-loop deblocking filter and the advanced intra-prediction. In general, it can be said that, on average, the visual quality provided with the proposed SDMCTF and IBMCTF schemes is comparable to the one provided by AVC for the middle and higher bitrates of the tests of Table IV-III, where blocking artifacts and ringing are less present.

Figure IV-60. "Soccer" and "Crew" sequences decoded at a very high bitrate in QCIF (15 Hz) and CIF (30 Hz) resolutions. Left: SDMCTF; right: IBMCTF.

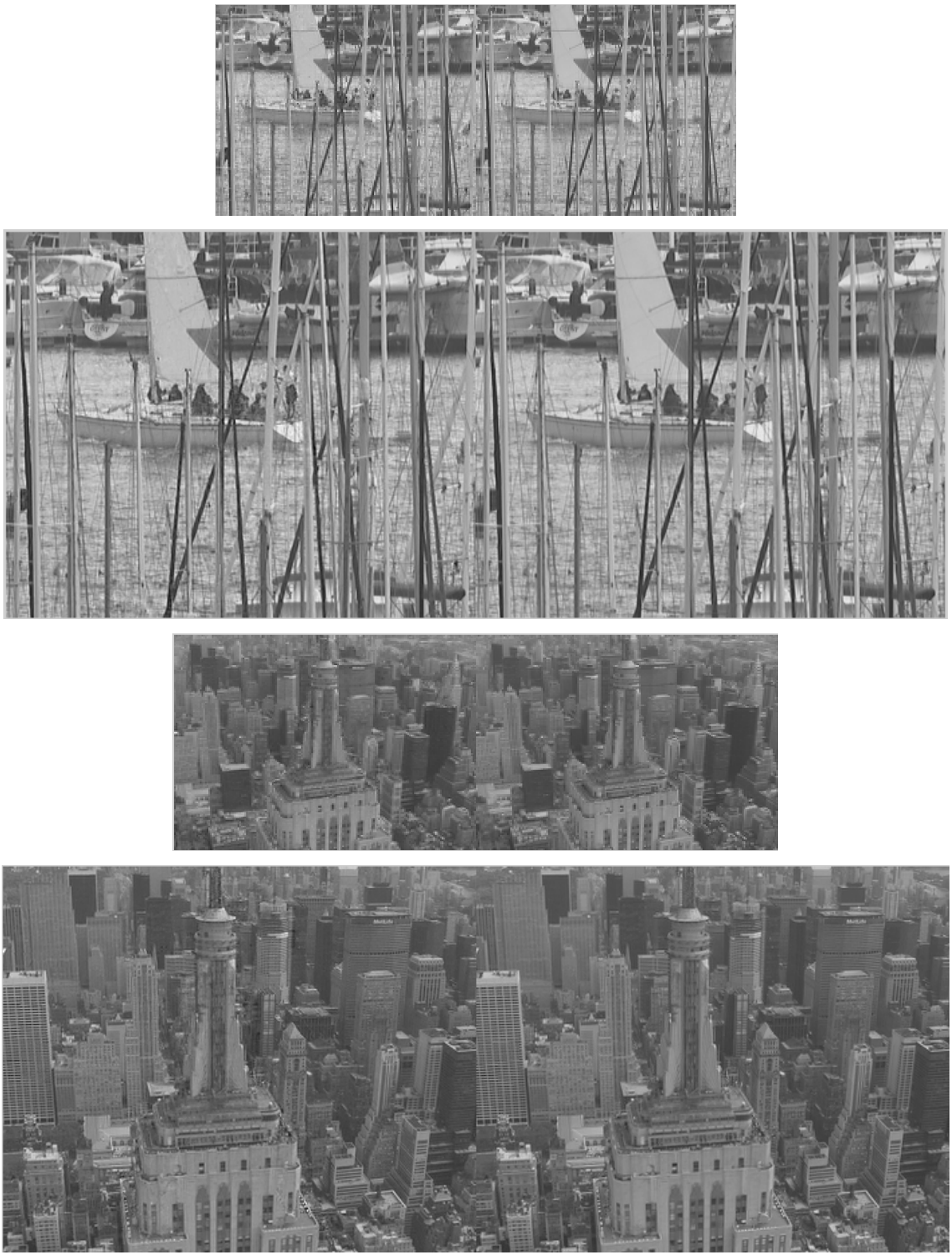


Figure IV-61. "Harbour" and "City" sequences decoded at a very high bitrate in QCIF (15 Hz) and CIF (30 Hz) resolutions. Left: SDMCTF; right: IBMCTF.

4.8 Discussion and Conclusions

This chapter introduced in-band motion compensated prediction within closed-loop and open-loop video coding systems. In addition, for open-loop systems, in-band motion-compensated temporal filtering was proposed by coupling in-band motion compensated prediction with in-band motion compensated update. In all cases it was found that, with the use of a complete-to-overcomplete transform for the performance of motion estimation and compensation in the wavelet domain, an equivalent system with the conventional closed-loop and open-loop frameworks can be provided in-band. This was shown to have several advantages from the functionality point of view, e.g. for base-layer compatibility (Subsection 4.4.2). In general, different prediction structures for each resolution level may satisfy different application requirements (Figure IV-16). Moreover, experimental evidence (Subsection 4.7.6, Figure IV-57 – Figure IV-61) confirmed that the in-band performance of MCP (and MCU) provides substantially-improved resolution scalability versus that provided by conventional systems that perform MCP in the spatial domain. Similar to the way an open-loop system provides bitrate scalability at seemingly no cost in compression efficiency versus the equivalent closed-loop system, in-band video coding based on the ODWT appears to provide improved resolution scalability at little or no cost in compression efficiency for the full resolution.

Finally, it was shown that the use of advanced motion estimation benefits both spatial-domain and in-band systems in the same manner. Moreover, objective evaluation for coding a large variety of standard-definition video sequences in a wide range of bitrates revealed that fully-scalable wavelet-based systems using advanced motion-compensated temporal filtering are (on-average) comparable or superior to the current state-of-the-art in non-scalable coding, i.e. the MPEG-4 Advanced Video Coder.

In a concluding summary, this chapter makes the following contributions:

- In-band video coders with closed-loop prediction are proposed in Section 4.2. The proposed framework was among the early proposals in the relevant literature that utilize overcomplete wavelet transforms for in-band MCP.

- In Section 4.3, a low-redundancy transform is adapted and used within the in-band video coding framework in order to investigate the performance of transforms with limited redundancy. The experimental evaluation reveals that the chosen low-redundancy transform appears to be comparable with the proposed ODWT-based system when limited to the same redundancy (two phases). As a result, it appears that, for the systems of interest, the use of low-redundancy transforms does not provide a significant benefit versus the equivalent operation within the ODWT-based framework.

- Section 4.4 proposed an in-band MCP framework within open-loop video coding architectures, in order to combine the efficient bitrate scalability of open-loop coding with the inherent resolution scalability of the in-band frameworks.

- Within this category of systems, two important aspects were investigated: base-layer compatibility with a motion-compensated DCT-based system for the low-resolution, and control of the distortion fluctuations inherent in open-loop MCP-based video coding.

- Within the open-loop video coding architecture, in-band motion-compensated temporal filtering was proposed in Section 4.5, and a generalized formulation for arbitrary motion-compensated lifting-based temporal decompositions for in-band MCTF was given.

- In order to improve the prediction efficiency of the proposed systems, Section 4.6 proposes an advanced motion estimation algorithm based on multi-frame multihypothesis prediction using variable block sizes. For coding of standard-definition sequences, the best-performing instantiation of the algorithm gives very promising results.

References

[1] S. A. Martucci, I. Sodagar, T. Chiang, and Y. Zhang, "A Zerotree Wavelet Video Coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 109-118, 1997.

[2] D. Marpe and H. L. Cycon, "Very low bit-rate video coding using wavelet-based techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 85-94, 1999.

[3] D. Taubman and M. W. Marcelin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*. Norwell, Massachusetts: Kluwer Academic Publishers, 2002.

[4] J. Li, J. Li, and C.-C. J. Kuo, "An embedded DCT approach to progressive image compression," presented at IEEE International Conference on Image Processing (ICIP), 1996.

[5] H. S. Malvar, "Fast progressive image coding without wavelets," presented at IEEE Data Compression Conference (DCC), Snowbird (UT), USA, 2000.

[6] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*, vol. 3, pp. 559-571, 1994.

[7] B. Pesquet-Popescu and V. Bottreau, "Three Dimensional Lifting Schemes for Motion Compensated Video Compression," presented at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, Utah, USA, 2001.

[8] A. Secker and D. Taubman, "Motion-Compensated Highly Scalable Video Compression using Adaptive 3D Wavelet Transform Based on Lifting," presented at IEEE International Conference on Image Processing (ICIP), Thessaloniki, Greece, 2001.

[9] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, 1989.

[10] Y. Andreopoulos, M. Van der Schaar, A. Munteanu, J. Barbarien, P. Schelkens, and J. Cornelis, "Fully-scalable wavelet video coding using in-band motion compensated temporal filtering," presented at International Conference on Acoustics, Speech and Signal Processing, ICASSP2003, 2003.

[11] Y. Andreopoulos, M. Van der Schaar, A. Munteanu, J. Barbarien, P. Schelkens, and J. Cornelis, "Open-loop, in-band motion-compensated temporal filtering for objective full-scalability in wavelet video coding," ISO/IEC JTC1/SC29/WG11, Shanghai, China, M9026, October 2002 2002.

[12] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. v. d. Schaar, J. Cornelis, and P. Schelkens, "In-band Motion Compensated Temporal Filtering," *Signal Processing: Image Communication*, vol. 19, pp. 653-673, 2004.

[13] N. Mehrseresht and D. Taubman, "Spatial Scalability and Compression Efficiency within a Flexible Motion Compensated 3D-DWT," presented at IEEE International Conference on Image Processing (ICIP), Singapore, SG, 2004.

[14] C. Mayer and S. Albert, "Scalable video coding with in-band prediction," presented at Proceedings of SPIE Electronic Imaging - Image and Video Communications and Signal Processing, Santa Clara, US, 2003.

[15] C. Mayer, "Motion compensated in-band prediction for wavelet-based spatially scalable video coding," presented at Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Hong-Kong, CN, 2003.

[16] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 1009-1020, 2002.

[17] S. Zafar, Y.-Q. Zhang, and B. Jabbari, "Multiscale video representation using multiresolution motion compensation and wavelet decomposition," *IEEE Journal of Selected Areas in Communications*, vol. 11, pp. 24-35, 1993.

[18] Y.-Q. Zhang and S. Zafar, "Motion-compensated wavelet transform coding for color video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, pp. 285-296, 1992.

[19] H. Ito and N. Farvardin, "On Motion Compesation of Wavelet Coefficients," presented at International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1995.

[20] J.-R. Ohm, "Advances in scalable video coding," *Proceedings of the IEEE*, vol. 93, pp. 42-56, 2005.

[21] J. Skowronski, "Pel-recursive Motion Estimation and Compensation in Subbands," *Signal Processing: Image Communication*, vol. 14, pp. 389-396, 1999.

[22] H. S. K. Kim and H. W. Park, "Wavelet-based moving-picture coding using shift-invariant motion estimation in wavelet domain," *Signal Processing: Image Communication*, vol. 16, pp. 669-679, 2001.

[23] H. Sari-Sarraf and D. Brzakovic, "A shift-invariant discrete wavelet transform," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2621-2626, 1997.

[24] X. Li, L. Kerofsky, and S. Lei, "All-phase motion compensated prediction in the wavelet domain for high performance video coding," presented at IEEE International Conference on Image Processing (ICIP), Thessaloniki,Greece, 2001.

[25] X. Li, "New results on phase shifting in the wavelet space," *IEEE Signal Processing Letters*, vol. 10, pp. 193-195, 2003.

[26] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens, and J. Cornelis, "Wavelet-based Fully-Scalable Video Coding with In-band Prediction," presented at IEEE Signal Processing Symposium (SPS), Leuven, Belgium, 2002.

[27] H.-W. Park and H.-S. Kim, "Motion Estimation Using Low-Band-Shift Method for Wavelet-Based Moving-Picture Coding," *IEEE Transactions on Image Processing*, vol. 9, pp. 577-587, 2000.

[28] G. Van der Auwera, A. Munteanu, P. Schelkens, and J. Cornelis, "Scalable Wavelet Video-Coding with In-Band Prediction - the Bottom-up Overcomplete Discrete Wavelet Transform," presented at IEEE International Conference on Image Processing (ICIP), Rochester, New York, USA, 2002.

[29] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445-3462, 1993.

[30] A. Munteanu, J. Cornelis, G. Van der Auwera, and P. Cristea, "Wavelet Image Compression - The Quadtree Coding Approach," *IEEE Transactions on Information Technology in Biomedicine*, vol. 3, pp. 176-185, 1999.

[31] G. Strang and T. Nguyen, *Wavelets and filter banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[32] A. Jain and K. Jain, "A displacement metric for frame difference," *IEEE Transactions on Communications*.

[33] G. Van der Auwera, A. Munteanu, P. Schelkens, and J. Cornelis, "Bottom-up Motion Compensated Prediction in the Wavelet Domain for Spatially Scalable Video Coding," *IEE Electronics Letters*, vol. 38, pp. 1251-1253, 2002.

[34] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens, and J. Cornelis, "A new method for complete-to-overcomplete discrete wavelet transforms," presented at 14th International Conference on Digital Signal Processing, DSP2002, Santorini, Greece, 2002.

[35] M. Vrankic, K. Egiazarian, and A. Gotchev, "Phase shifting in the Quincunx filter bank," presented at Proc. DSPA'04, 2004.

[36] Y. Andreopoulos, A. Munteanu, G. V. d. Auwera, P. Schelkens, and J. Cornelis, "Complete-to-Overcomplete Discrete Wavelet Transforms: Theory and Application," *IEEE Transactions on Signal Processing*, to appear.

[37] S. Cui, Y. Wang, and J. E. Fowler, "Mesh-based Motion Estimation and Compensation in the Wavelet Domain using a Redundant Transform," presented at IEEE International Conference on Image Processing (ICIP), Rochester, NY, USA, 2002.

[38] J. Barbarien, Y. Andreopoulos, A. Munteanu, P. Schelkens, and J. Cornelis, "Motion vector coding for in-band motion compensated temporal filtering," presented at IEEE International Conference on Image Processing, ICIP 2003, Barcelona, Spain, 2003.

[39] Y. Andreopoulos, M. Van der Schaar, A. Munteanu, J. Barbarien, P. Schelkens, and J. Cornelis, "Complete-to-overcomplete discrete wavelet transforms for fully-scalable video coding with MCTF," presented at SPIE Visual Communications and Image Processing, VCIP2003, Lugano, Switzerland, 2003.

[40] S. G. Mallat, *A wavelet tour of signal processing*. San Diego: Academic Press, 1998.

[41] X. Li, "Scalable Video Compression via Overcomplete Motion Compensated Wavelet Coding," *Signal Processing: Image Communication*, vol. 19, pp. 637-651, 2004.

[42] R. R. Coifman and D. L. Donoho, "Translation invariant de-noising," in *Wavelets and Statistics: Lecture Notes in Statistics 103*, A. Antoniadis and G. Oppenheim, Eds. New York: Springer-Verlag, pp. 125-150.

[43] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, pp. 247-269, 1998.

[44] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure Applied Mathematics*, vol. 45, pp. 485-560, 1992.

[45] ISO/IEC, "JPEG 2000 image coding system," ISO/IEC JTC1/SC29/WG1, IS 15444-1, December 2000 2000.

[46] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, J. Barbarien, P. Schelkens, and J. Cornelis, "Wavelet-based fine granularity scalable video coding with in-band prediction," ISO/IEC JTC1/SC29/WG11, Jeju, South-Korea, Report MPEG2002/m7906, March 2002 2002.

[47] Y. Andreopoulos, A. Munteanu, G. V. d. Auwera, J. Cornelis, and P. Schelkens, "Single-rate calculation of overcomplete discrete wavelet transforms for scalable coding applications," *Signal Processing*, to appear.

[48] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards - Algorithms and Architectures*. Massachusetts (USA): Kluwer Academic Publishers, 1995.

[49] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multiscale transforms," *IEEE Transactions on Information Theory*, vol. 38, pp. 587-607, 1992.

[50] N. G. Kingsbury, "Complex Wavelets for Shift Invariant Analysis and Filtering of Signals," *Journal of Appl. and Comput. Harmonic Analysis*, vol. 10, pp. 234-253, 2001.

[51] J. Magarey and N. G. Kingsbury, "Motion estimation using complex wavelets," presented at SPIE Wavelet Applications in Signal and Image Processing IV, 1996.

[52] I. W. Selesnick, "Hilbert Transform Pairs of Wavelet Bases," *IEEE Signal Processing Letters*, vol. 8, pp. 170-173, 2001.

[53] I. W. Selesnick, "The Design of Approximate Hilbert Transform Pairs of Wavelet Bases," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1144-1152, 2001.

[54] F. Verdicchio, Y. Andreopoulos, A. Munteanu, J. Barbarien, P. Schelkens, J. Cornelis, and A. Pepino, "Scalable Video Coding with In-band Prediction in the Complex Wavelet Transform," presented at IEEE Advanced Concepts for Intelligent Vision Systems (ACIVS), Ghent, BE, 2002.

[55] K. Sivaramakrishnan and T. Nguyen, "A Uniform Transform-domain Video Codec based on Dual Tree Complex Wavelet Transform," presented at IEEE International Conference on Accoustics, Speech and Signal Processing (ICASSP), Salt Lake City, UT, USA, 2001.

[56] C. Brislawn, "Preservation of Subband Symmetry in Multirate Signal Coding," *IEEE Transactions on Signal Processing*, vol. 43, pp. 3046-3050, 1995.

[57] J. R. Williams and K. Amaratunga, "A Discrete Wavelet Transform without Edge Effects using Wavelet Extrapolation," Massachusetts Institute of Technology, Intelligent Systems Laboratory, IESL no. 95-02 1995.

[58] J.-R. Ohm and T. Ebrahimi, "Report of Ad-hoc Group on Exploration of Interframe Wavelet Technology in Video," ISO/IEC JTC1/SC29/WG11 m8295, March 2002.

[59] D. S. Turaga, M. v. d. Schaar, Y. Andreopoulos, A. Munteanu, and P. Schelkens, "Unconstrained Motion Compensated Temporal Filtering (UMCTF) for Efficient and Flexible Interframe Wavelet Video Coding," *Signal Processing: Image Communication*, to appear.

[60] T. Kronander, "Motion Compensated 3-dimensional Wave-form Image Coding," presented at IEEE International Conference on Accoustics, Speech and Signal Processing (ICASSP), 1989.

[61] S.-J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Transactions on Image Processing*, vol. 8, pp. 155-167, 1999.

[62] A. Secker and D. Taubman, "Lifting-Based Invertible Motion Adaptive Transform (LIMAT) Framework for Highly Scalable Video Compression," *IEEE Transactions Image Processing*, vol. 12, pp. 1530-1542, 2003.

[63] D. S. Turaga and M. v. d. Schaar, "Unconstrained Motion Compensated Temporal Filtering," ISO/IEC JTC1/SC29/WG11 m8388, 2002.

[64] M. Flierl and B. Girod, "Video Coding with Motion-compensated Lifted Wavelet Transforms," *Signal Processing: Image Communication*, vol. 19, pp. 561-575, 2004.

[65] H. M. Radha, M. v. d. Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding for multimedia streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, pp. 53-68, 2001.

[66] W. Li, "Streaming Video Profile in MPEG-4," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 301-317, 2001.

[67] S. Battista, F. Casalino, and C. Lande, "MPEG-4: A Multimedia Standard for the Third Millennium, Part 1," *IEEE Multimedia Magazine*, vol. 6, pp. 74-83, 1999.

[68] Y. Andreopoulos, M. v. d. Shaar, A. Munteanu, P. Schelkens, and J. Cornelis, "Spatio-temporal-SNR Scalable Wavelet Video Coding with Motion-compensated DCT Base-layer Architectures," presented at IEEE International Conference on Image Processing, Barcelona, SP, 2003.

[69] U. Benzler, "Spatial Scalable Video Coding using a Combined Subband-DCT Approach," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 1080-1087, 2000.

[70] A. Munteanu, "Wavelet image coding and multiscale edge detection:Algorithms and applications," in *Dept. of Electronics and Information Processing (ETRO)*. Doctorate Thesis, Brussels: Vrije Universiteit Brussel, 2003, pp. 285.

[71] P. Schelkens, A. Munteanu, J. Barbarien, M. Galca, X. Giro i Nieto, and J. Cornelis, "Wavelet Coding of Volumetric Medical Datasets," *IEEE Transactions on Medical Imaging, Special issue on "Wavelets in Medical Imaging,"* vol. 22, pp. 441-458, 2003.

[72] Y. He, R. Yan, F. Wu, and S. Li, "H.26L-based fine granularity scalable video coding," ISO/IEC JTC1/SC29/WG1, M7788, December 2001 2001.

[73] A. Munteanu, Y. Andreopoulos, M. Van der Schaar, P. Schelkens, and J. Cornelis, "Control of the distortion variation in video coding systems based on motion compensated temporal filtering," presented at IEEE International Conference on Image Processing, ICIP 2003, Barcelona, Spain, 2003.

[74] Y. Andreopoulos, A. Munteanu, P. Schelkens, and J. Cornelis, "Control of the Distortion Variation in Motion Compensated Temporal Filtering," ISO/IEC JTC1/SC29/WG11 m9253, 2002.

[75] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with Lifting Implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 1183-1194, 2004.

[76] Y. Andreopoulos, A. Munteanu, M. v. d. Schaar, J. Cornelis, and P. Schelkens, "Comparison between "t+2D" and "2D+t" Architectures with Advanced Motion Compensated Temporal Filtering," ISO/IEC JTC1/SC29/WG11 m11045, July 2004.

[77] M. Flierl, T. Wiegand, and B. Girod, "Rate-Constrained Multihypothesis Prediction for Motion-Compensated Video Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 957-969, 2002.

[78] M. Flierl, "Doctorate Thesis: Video Coding with Superimposed Motion-Compensated Signals." Erlangen: University of Erlangen-Nuremberg, 2003.

[79] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens, and J. Cornelis, "Scalable Wavelet Video-Coding with In-Band Prediction - Implementation and Experimental Results," presented at IEEE International Conference on Image Processing (ICIP), Rochester, New York, USA, 2002.

[80] M. v. d. Schaar and J.-R. Ohm, "Draft testing procedures for evidence on scalable video coding technology," ISO/IEC JTC1/SC29/WG11 (MPEG), N5167, Oct. 2002.

[81] D. Taubman, D. Maestroni, R. Mathew, and S. Tubero, "SVC Core Experiment 1, description of UNSW contribution," ISO/IEC JTC1/SC29/WG11 (MPEG), m11441, Oct. 2004.

[82] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H.263+: video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 849-866, 1998.

[83] T. Wiegand, "H26L Test Model Long-Term Number 9 (TML-9) draft 0," ITU/TSS/SG16 (VCEG), Document VCEG-N83 D1, December 2001 2001.

[84] J. Barbarien, A. Munteanu, F. Verdicchio, Y. Andreopoulos, J. Cornelis, and P. Schelkens, "Scalable motion vector coding," *Electronics Letters*, vol. 40, pp. 932-934, 2004.

[85] A. Secker and D. Taubman, "Highly scalable video compression with scalable motion coding," *IEEE Transactions on Image Processing*, vol. 13, pp. 1029-1041.

[86] J. Barbarien, Y. Andreopoulos, A. Munteanu, P. Schelkens, and J. Cornelis, "Coding of motion vectors produced by wavelet-domain motion estimation," presented at Proceedings of Picture Coding Symposium, Saint Malo, FR, 2003.

[87] A. Golwelkar, I. Bajic, and J. W. Woods, "Response to call for evidence on scalable video coding," ISO/IEC JTC1/SC29/WG11 (MPEG), m9723, July 2003.

[88] MSSG, "MPEG Software Simulation Group," http://www.mpeg.org/MPEG/MSSG/.

[89] J.-R. Ohm and M. v. d. Schaar, "Call for proposals on scalable video coding technology," ISO/IEC JTC1/SC29/WG11 (MPEG), n5958, Oct. 2003.

[90] T. Wiegand and G. Sullivan, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification," ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6 2003.

Chapter V

COMPLETE-TO-OVERCOMPLETE DISCRETE WAVELET TRANSFORMS

RECENTLY, wavelet-based image and video coding systems that utilize the overcomplete discrete wavelet transform have been proposed in the literature [1] [2] [3] [4] [5] [6]. As analyzed in the previous chapter, the good coding performance of these techniques stems from wavelet-domain operations that require shift-invariance, such as in-band motion estimation and compensation, and in-band motion compensated temporal filtering. The ODWT is used for this purpose since it is a redundant version of the discrete wavelet transform that attains shift-invariance [7] [8].

Given the input signal, the classical construction of the ODWT is trivial by using for example the "à-trous" algorithm [7] [9]. However, as explained in the previous chapter, in wavelet-based coding systems the codec always processes the critically-sampled (complete) DWT subbands. Hence, a complete-to-overcomplete DWT has to take place: first the inverse DWT is performed in order to reconstruct the input signal, followed by the ODWT. Furthermore, Chapters III and IV demonstrated that, in fully-scalable image and video-coding systems, the critically-sampled DWT subbands are processed in a *resolution-scalable* manner (from coarse to fine resolution). In such environments, it is imperative that both ends of the system *independently* construct the identical ODWT information at each resolution level in order to avoid drift effects. Given the resolution levels that are available at both the encoder and decoder, a separate loop is used at each level for the CODWT in order to construct identical ODWT representations at both sides. This approach has several disadvantages:

- A direct method for the derivation of the ODWT subbands from the DWT is not provided. Instead, the ODWT construction is a cascade of the inverse DWT and the "à-trous" calculation. As a result, the reconstruction of the input signal is required. This causes significant calculation overhead and delay since the input signal has the highest sampling rate.

- A multi-rate calculation scheme is used in every case with a cascade of upsampling and downsampling operations. As a result, even for high-speed high-parallel implementations, the achievable percentage of hardware utilization is low since the filtering of every level has to be pipelined with the production of the results of the previous and the next level.

In this chapter, we present a new theory for the CODWT, formulated for any arbitrary level k of the transform, in which the ODWT of level k is produced *directly* from the DWT of k-levels [10] [11] [12]. The reconstruction of the input signal is not required and, furthermore, no upsampling is performed. Our initial findings reported in [10] [11], as well as relevant work reported recently by Li [13] and Van der Auwera et al [14], present similar approaches for performing phase shifting in the wavelet domain. This chapter generalizes these ideas to a transform that performs the direct construction of an *arbitrary* phase in the k-level wavelet domain (Proposition 2). In addition, we propose a direct solution for the (all-phase) CODWT (Proposition 3) and present an efficient calculation scheme for the transform. Apart from the new theoretical aspects arising from the proposed transform, we demonstrate the significant practical benefits of the new approach under the application scenarios that require resolution scalability. In such cases, the proposed CODWT is computed using a *single-rate* calculation scheme while providing exactly the same results as the conventional multi-rate approach.

5.1 Theoretical Derivations

5.1.1 Notations and Symbolism

In this chapter, bold-faced capital and lower letters indicate matrices and vectors respectively, while $\mathbf{I}$ denotes the identity matrix. Calligraphic notation is reserved for operators (e.g., $\mathcal{D}$ denotes the polyphase separation). All signals and filters are considered in the Z-domain as Laurent polynomials and the letter z is reserved for this purpose. All the used indices are integers. For all matrices, vectors, signals and filters, the superscripts denote the decomposition level, except for superscript T that denotes transposition. Subscripts are used to enumerate signals or filters in matrices and vectors. Additionally, they are used to indicate polyphase components; each case is identified from the context.

The polyphase separation of a signal or filter $X(z)$ is given by:

$$\mathcal{D}X(z) = [X_0(z) \quad X_1(z)]^T,$$

with the commutative operation given by:

$$\mathcal{D}^{-1}\mathcal{D}X(z) = X_0(z^2) + zX_1(z^2) = X(z).$$

For the DWT, the Type-I analysis polyphase matrices that produce the even and odd polyphase components of the transform are denoted as $\mathbf{E}_0(z)$, $\mathbf{E}_1(z)$ respectively; their definitions are:

$$\mathbf{E}_0(z) = \begin{bmatrix} H_0(z) & H_1(z) \\ G_0(z) & G_1(z) \end{bmatrix}, \; \mathbf{E}_1(z) = \mathbf{E}_0(z) \begin{bmatrix} 0 & z \\ 1 & 0 \end{bmatrix},$$

where $H(z)$, $G(z)$ are the low-pass and high-pass analysis DWT filters, respectively. Notice that in the last equation the even and odd polyphase components are produced by shifting the analysis filter-bank with the delay-permutation matrix [15]:

$$\begin{bmatrix} 0 & z \\ 1 & 0 \end{bmatrix}.$$

One can opt to shift either the input signal, or the analysis filters; both approaches are equivalent in the theoretical derivations, but shifting the analysis filters makes the implementation easier.

The corresponding Type-II synthesis polyphase matrices are:

$$\mathbf{R}_i(z) = [\mathbf{E}_i(z)]^{-1}, \; i = \{0,1\},$$

e.g.:

$$\mathbf{R}_0(z) = \frac{1}{\det \mathbf{E}_0(z)} \begin{bmatrix} G_1(z) & -H_1(z) \\ -G_0(z) & H_0(z) \end{bmatrix}.$$

In order to simplify the expressions we always assume (without loss of generality) that the filters $H(z)$ and $G(z)$ are properly shifted so that perfect reconstruction is achieved with zero delay and [16]:

$$\det \mathbf{E}_0(z) = -1.$$

Two numerical examples of such filters, which are used for the results of this chapter, are given in Table V-I.

5/3 filter-bank			
Degree in Z	$H(z)$	**Degree in Z**	$G(z)$
1, -3	-0.17677669529664	1,-1	0.70710678118655
0, -2	0.35355339059327	0	-0.35355339059327
-1	1.06066017177982		
9/7 filter-bank			
Degree in Z	$H(z)$	**Degree in Z**	$G(z)$
3, -5	0.03782845550726	3, -3	0.06453888262870
2, -4	-0.02384946501956	2, -2	-0.04068941760916
1, -3	-0.11062440441844	1, -1	-0.41809227322162
0, -2	0.37740285561283	0	0.78848561640558
-1	0.85269867900889		

Table V-I. The 5/3 and 9/7 filter-banks modified so that $\det \mathbf{E}_0(z) = H_0(z)G_1(z) - G_0(z)H_1(z) = -1$ (filter G is linear phase).

Using the Noble identity [15], the single-level filtering-and-downsampling operator can be written as:

$$\mathcal{U}_i^1 = \left[\mathcal{D}\left(z^i U(z)\right)\right]^T \mathcal{D},$$

with $i = \{0,1\}$ denoting the retained polyphase component and $\mathcal{U} = \{\mathcal{H}, \mathcal{G}\}$, $U = \{H, G\}$ respectively. For l decomposition levels ($l \geq 1$), this operator is:

$$\mathcal{U}_p^l = \mathcal{U}_{b_0}^1 \mathcal{H}_{b_1}^1 \ldots \mathcal{H}_{b_{l-1}}^1,\ p = \sum_{i=0}^{l-1} b_i 2^i, b_i = \{0,1\}. \tag{5.1}$$

Definition 1: For a signal $X(z)$ and a perfect-reconstruction analysis filter-bank $U = \{H, G\}$ with $\det \mathbf{E}_0(z) = -1$, we define $\mathbf{w}_p^l(z)$ as the p-phase wavelet subbands of the ODWT of decomposition level $E = l$ ($l \geq 1$) [17] [8] [13], given by:

$$\mathbf{w}_p^l(z) = \left[A_p^l(z) \quad D_p^l(z)\right]^T = \left[\mathcal{H}_p^l \quad \mathcal{G}_p^l\right]^T X(z),\ 0 \leq p < 2^l,$$

with $A_p^l(z)$, $D_p^l(z)$ the low and high-frequency subbands, respectively.

This definition coincides with the ODWT subbands calculated by the cycle-spinning algorithm of Coifman and Donoho [18]: the binary representation of p is actually the binary map of the DWT defined in [18]. In total, the l-level ODWT of a signal $X(z)$ is given by $\boldsymbol{\mathcal{L}}_p^l X(z)$, for every p, $0 \leq p < 2^l$, where:

$$\boldsymbol{\mathcal{L}}_p^l = \begin{bmatrix} \boldsymbol{\lambda}_{p,\mathcal{H}}^l \\ \boldsymbol{\lambda}_{p,\mathcal{G}}^l \end{bmatrix}$$

is a $2 \times l$ matrix operator and:

$$\boldsymbol{\lambda}_{p,\mathcal{U}}^l = \left[\mathcal{U}_{b_0}^1 \quad \mathcal{U}_{b_0+2b_1}^2 \quad \cdots \quad \mathcal{U}_p^l\right].$$

For example, for $l = 1$ we have:

$$\boldsymbol{\mathcal{L}}_p^1 = \mathbf{E}_p(z)\mathcal{D}, p = \{0,1\}.$$

The following lemma establishes the $\boldsymbol{\mathcal{L}}_p^l$ operator recursively.

Lemma 1: The $\boldsymbol{\mathcal{L}}_p^l$ operator applied on signals $\mathbf{x}(z) = \left[X_{l-1}(z) \ \cdots \ X_0(z)\right]^T$ and $X_l(z)$, with $X_q(z)$, $0 \leq q \leq l$, having a sampling rate of 2^{-q}, satisfies:

$$\boldsymbol{\mathcal{L}}_i^1 \left(X_l(z) + \boldsymbol{\lambda}_{p,\mathcal{H}}^l \mathbf{x}(z)\right) = \boldsymbol{\mathcal{L}}_{2p+i}^{l+1} \begin{bmatrix} X_l(z) \\ \mathbf{x}(z) \end{bmatrix}, i = \{0,1\}. \tag{5.2}$$

Proof: By expanding the right part of (5.2) using equation (5.1) and the definition of the $\boldsymbol{\mathcal{L}}_p^l$ operator, we reach the left part as follows:

$$\boldsymbol{\mathcal{L}}_{2p+i}^{l+1}\begin{bmatrix} X_l(z) \\ \mathbf{x}(z) \end{bmatrix} = \begin{bmatrix} \mathcal{H}_i^1 & \mathcal{H}_{i+2b_0}^2 & \cdots & \mathcal{H}_{2p+i}^{l+1} \\ \mathcal{G}_i^1 & \mathcal{G}_{i+2b_0}^2 & \cdots & \mathcal{G}_{2p+i}^{l+1} \end{bmatrix}\begin{bmatrix} X_l(z) \\ \mathbf{x}(z) \end{bmatrix} = \begin{bmatrix} \mathcal{H}_i^1 \\ \mathcal{G}_i^1 \end{bmatrix} X_l(z) + \begin{bmatrix} \mathcal{H}_i^1 \\ \mathcal{G}_i^1 \end{bmatrix} \boldsymbol{\lambda}_{p,\mathcal{H}}^l \mathbf{x}(z)\,. \tag{5.3}$$

■

5.1.2 Problem Description

In this section, we briefly present two methods for the CODWT under the video coding scenarios of Chapter IV. The first method is based on the IDWT followed by the low-band shift (LBS) method [1, 19]. The LBS can be seen as a specific implementation technique of the "à-trous" algorithm [8] [7], where the different ODWT subbands are produced and stored according to the retained polyphase components. The second method represents our proposed approach.

Figure V-1(a) shows an example of the one-dimensional ODWT for three decomposition levels starting from an input signal $X(z)$. This figure facilitates the description of the LBS method [1]. Initially, the input signal $X(z)$ is decomposed in two subband sets $A_0^1(z), D_0^1(z)$ & $A_1^1(z), D_1^1(z)$ by retaining separately the even and odd polyphase components of the non-decimated decomposition respectively, or equivalently, by performing two wavelet decompositions: one to the zero-shifted and one to the unit-shifted input signal respectively. Each of the low-frequency subbands $A_0^1(z)$ and $A_1^1(z)$ is further analyzed in the same manner, while the high-frequency subbands $D_0^1(z)$ and $D_1^1(z)$ are the outputs of the first decomposition level. This process is repeated successively, yielding the ODWT representation from the input signal $X(z)$ (see Figure V-1(a)). The subbands $A_0^3(z)$ and $D_0^l(z)$, $l = 1,2,3$ represent the critically-sampled (complete) DWT of three decomposition levels, while the subbands $A_i^3(z), D_i^l(z)$, $1 \le l \le 3$, $1 \le i < 2^l$ represent the calculated ODWT. Hence, for the CODWT based on this method, the signal $X(z)$ has to be reconstructed by performing the IDWT to the subbands $A_0^3(z)$, $D_0^l(z)$, followed by the LBS.

Notice that the subbands $A_i^3(z), D_i^l(z)$ shown in Figure V-1(a) stem from the classical ODWT decomposition scheme of [1], which is equivalent to the "à-trous" algorithm [8]. The difference is that, at every level, the subbands of Figure V-1(a) must be interleaved in order to produce the non-decimated ODWT obtained with the algorithm of [8]. As a result, any subband $D_i^l(z)$ in the ODWT of Figure V-1(a) is the i-th polyphase component of the non-decimated ODWT of level l [8] [7].

In the two-dimensional case, the ODWT can be constructed in the same manner as in Figure V-1(a), by applying the LBS method on the input-subband rows and on the columns of the results. Hence, to facilitate the description, we focus in the following analysis on the one-dimensional case, with the extension in two dimensions following the row-column approach.

In a progressive resolution-refinement (resolution scalable) framework, the additional constraint that the subband coding and decoding occurs in a bottom-up manner is imposed: the coarsest-resolution subbands of the DWT are processed independently (subbands $A_0^3(z), D_0^3(z)$) and for every higher-resolution level l with $l = 2,1$, the subband $D_0^l(z)$ is additionally processed. In the case of closed-loop systems these subbands are decoded at pre-defined base-quality levels. In this way, for each target resolution-level $l = 3,2,1$, the encoder uses the same reference frames as the decoder will be

able to create at the client side, and no drift occurs across resolutions [5, 20] [21] [3]. This can be seen as an extension of the base-layer concept used in quality-progressive closed-loop video coding [22]; although the creation of the reference frames at the base-layer potentially lowers the performance of the subset of decoders that progressively process additional quality layers (or resolution levels in our case), this guarantees drift-free operation.

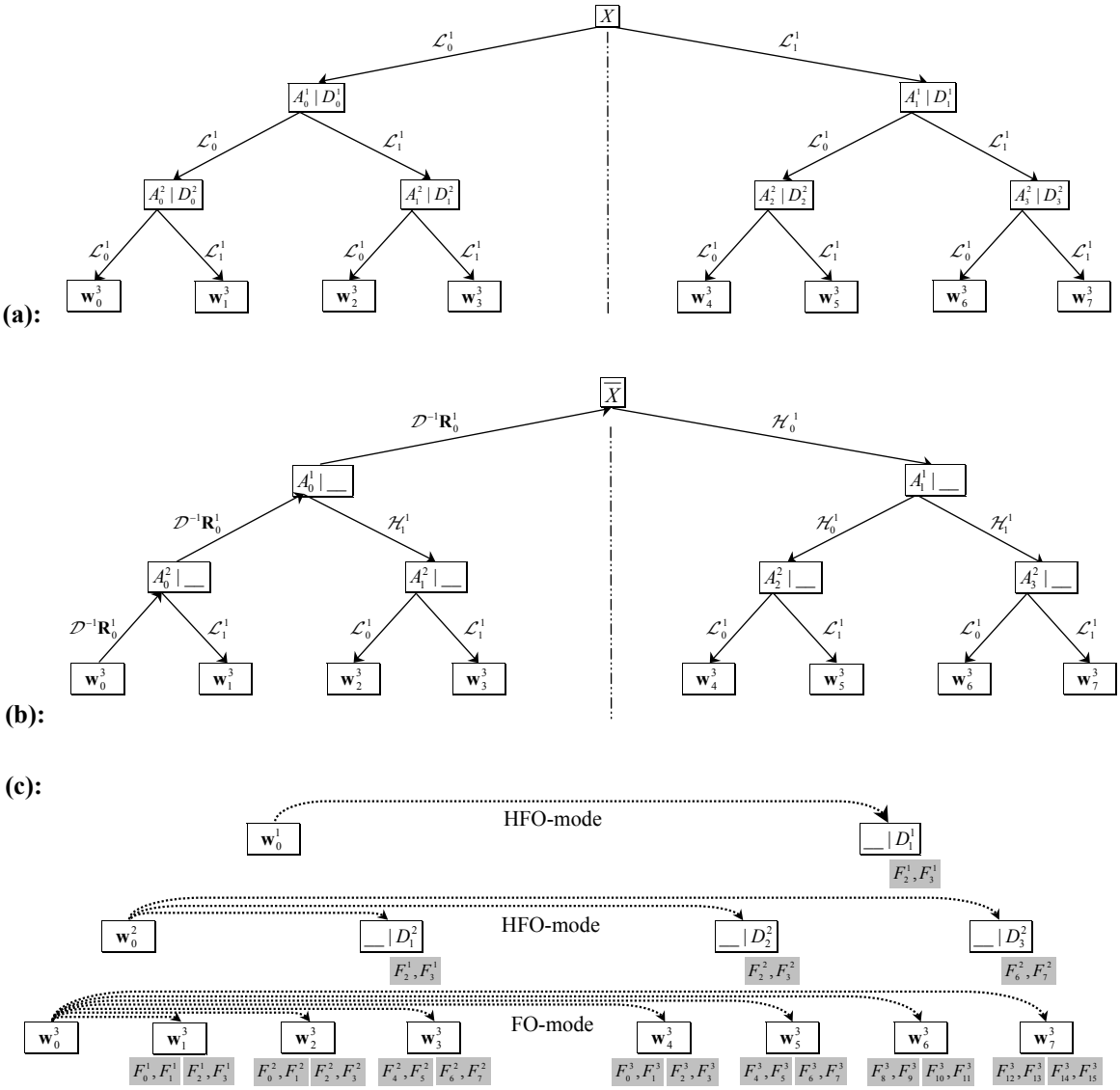


Figure V-1. (a): The ODWT construction of three levels starting from the input signal X. A number of one-level discrete wavelet transforms are performed that retain the even or odd samples of the non-decimated transform ($\mathcal{L}_i^1$ with $i = 0,1$ respectively) (b): The level-by-level ODWT construction (example for level three) using the conventional multi-rate IDWT and LBS approach that perform a set of inverse and forward transforms. (c): The level-by-level ODWT construction for all three levels using the proposed approach.

Under such a resolution-scalable coding framework, the CODWT based on the LBS method is readily adaptable to perform a *level-by-level* construction of the ODWT representation (denoted by LL-LBS), starting from the subbands of the critically-sampled transform of each decoded level. This process is illustrated in Figure V-1(b). Starting from the DWT subbands $\mathbf{w}_0^3(z)$ (coarsest resolution level), three single-level inverse wavelet transforms are performed. Subsequently, from the reconstructed signal $(\overline{X}(z))$, all the subbands $\mathbf{w}_i^3(z)$, $1 \le i < 8$ are constructed by performing the single-level forward transforms shown in Figure V-1(b). It is important to notice that, since in this case the subbands $D_0^2(z), D_0^1(z)$ are not available, the reconstructed signal $\overline{X}(z)$ and the subbands $\mathbf{w}_i^3(z)$ are only an approximation of $X(z)$ and of the original ODWT of level three respectively, shown in Figure V-1(a). However, in this resolution-scalable scenario, given the common information available at both the encoder and decoder sides, this ODWT representation is the best possible approximation for the current resolution level. Finally, if higher-resolution versions of the input data are required, the ODWT construction by the LL-LBS is repeated for the finer resolution levels ($l = 2$ or $l = 1$, depending on the target resolution).

Notice that, in coding applications, for every decomposition level l, the low-frequency subbands $A_j^l(z)$, $1 \le j < 2^l$ are part of the calculated ODWT only if level l is the coarsest-resolution of the decomposition (i.e. if $l = 3$ in the example of Figure V-1(c)). We define this construction that generates the signals:

$$\mathbf{w}_{\mathrm{ODWT}}^l(z) = \begin{bmatrix} \mathbf{w}_1^l(z) \\ \vdots \\ \mathbf{w}_{2^l-1}^l(z) \end{bmatrix},$$

as the *full-overcomplete* transform-production mode (FO-mode). In all the other cases (levels $l = 2,1$ of Figure V-1(c)), the critically-sampled DWT consists of the subband $D_0^l(z)$ and hence, in coding applications only the high-frequency ODWT subbands $D_1^l(z),\ldots,D_{2^l-1}^l(z)$ need to be calculated; this case is defined as the *high-frequency overcomplete* transform-production mode (HFO-mode). The difference between the FO and HFO modes is illustrated in Figure V-1(c).

Additionally, Figure V-1(c) presents the proposed alternative approach to the LL-LBS method for the level-by-level CODWT of level three, two and one. In this approach, the CODWT uses a set of prediction filters [11, 12] [14], denoted as $F_j^i(z)$, $1 \le i \le 3, 0 \le j < 2^{i+1}$, which are convolved with the subbands $A_0^i(z), D_0^i(z)$ to calculate the ODWT representation of each level. Notice that for levels two and one of Figure V-1(c), the ODWT construction occurs in the HFO mode, i.e. only the high-frequency subbands of the ODWT are calculated. Additionally, the figure illustrates that by using the prediction-filters, the overcomplete representation of each level is "predicted", as shown with the dotted lines. As a result, no upsampling is performed and no reconstruction of the spatial-domain signal $X(z)$ is required. The mathematical derivation of the prediction filters and the proposed CODWT are presented next.

5.1.3 Complete-to-Overcomplete Representations

In this section we present a generic framework for the calculation of the ODWT subbands of decomposition level k as a function of the critically-sampled wavelet decomposition (i.e. the subbands $A_0^k(z), D_0^l(z)$, with $1 \le l \le k$). From this formalism, the level-by-level CODWT of level k, which is especially interesting for applications requiring progressive-refinement in resolution (e.g. [2], [3], [5, 20]), can be found as a special case. Symmetry properties for the prediction filters of every level are also proven. Based on these properties, two efficient schemes for the CODWT calculation are presented.

5.1.3.1 Derivation of the ODWT Subbands of Decomposition Level k from the k-level DWT – the Prediction Filters

We start with by exemplifying the proposed CODWT for levels $E = 1,2$ (Subsection A). Then, Subsection B formulates the CODWT for any arbitrary level $E = k$.

A. Calculation of the ODWT Subbands of Levels $E = 1$ and $E = 2$

Definition 2: We define $\mathbf{F}_0^1(z)$ and $\mathbf{F}_i^2(z)$, $i = \{0,1\}$, as the prediction-filter matrices of levels $E = 1$ and $E = 2$, respectively. They are given by:

$$\mathbf{F}_0^1(z) = \begin{bmatrix} F_0^1(z) & F_1^1(z) \\ F_2^1(z) & F_3^1(z) \end{bmatrix} = \begin{bmatrix} zH_0(z)G_0(z) - H_1(z)G_1(z) & H_1(z)H_1(z) - zH_0(z)H_0(z) \\ zG_0(z)G_0(z) - G_1(z)G_1(z) & H_1(z)G_1(z) - zH_0(z)G_0(z) \end{bmatrix}. \quad (5.4)$$

$$\mathbf{F}_i^2(z) = F_{0,i}^1(z)\mathbf{I} + z^{-(1-i)}F_{0,1-i}^1(z)\mathbf{F}_0^1(z). \quad (5.5)$$

where $F_{0,i}^1(z)$ and $F_{0,1-i}^1(z)$ are the i and $1-i$ polyphase components of filter $F_0^1(z)$, respectively.

Proposition 1: The subbands $\mathbf{w}_1^1(z)$ of the ODWT of level $E = 1$ are given by:

$$\mathbf{w}_1^1(z) = \mathbf{F}_0^1(z)\mathbf{w}_0^1(z). \quad (5.6)$$

while subbands $\mathbf{w}_1^2(z)$, $\mathbf{w}_2^2(z)$, $\mathbf{w}_3^2(z)$ of the ODWT of level $E = 2$ are:

$$\begin{aligned} \mathbf{w}_1^2(z) &= \mathbf{F}_0^1(z)\mathbf{w}_0^2(z) \\ \mathbf{w}_{2+i}^2(z) &= \mathbf{F}_i^2(z)\mathbf{w}_0^2(z) + \mathcal{L}_i^1\left(F_1^1(z)D_0^1(z)\right),\ i = \{0,1\} \end{aligned}. \quad (5.7)$$

Proof: The proof of (5.6) can be derived by performing an inverse transform to subbands $\mathbf{w}_0^1(z)$ followed by a forward wavelet transform that retains the odd polyphase components of the non-decimated decomposition, as shown in Figure V-1(a). In total:

$$\mathbf{w}_1^1(z) = \mathbf{E}_1(z)\mathcal{D}\mathcal{D}^{-1}\mathbf{R}_0(z)\mathbf{w}_0^1(z) = \mathbf{F}_0^1(z)\mathbf{w}_0^1(z)$$

with $\mathbf{F}_0^1(z)$ the prediction-filter matrix given in (5.4).

Applying an inverse wavelet transform to subbands $\mathbf{w}_0^2(z)$ yields:

$$A_0^1(z) = \mathcal{D}^{-1}\left(\mathbf{R}_0(z)\mathbf{w}_0^2(z)\right).$$

Then, as shown in Figure V-1 (a), by performing a forward transform that retains the odd polyphase components of the non-decimated transform we reach the subbands $\mathbf{w}_1^2(z)$ given in (5.7). Additionally, by using a part of equation (5.6), we derive the subband $A_1^1(z)$, as:

$$A_1^1(z) = \left[F_0^1(z) \quad F_1^1(z)\right]\mathbf{w}_0^1(z).$$

The final result is reached by performing a forward wavelet transform retaining the even or odd polyphase components for $i = 0,1$ respectively, and using the Noble identity that exchanges the filtering and downsampling order [15]. Hence:

$$\begin{aligned}\mathbf{w}_{2+i}^2(z) &= \mathbf{E}_i(z)\mathcal{D}A_1^1(z)\\ &= \mathbf{E}_i(z)\left(F_{0,0}^1(z)\mathbf{I} + z^{-1}F_{0,1}^1(z)\begin{bmatrix}0 & z\\ 1 & 0\end{bmatrix}\right)\mathbf{R}_0(z)\mathbf{w}_0^2(z) + \boldsymbol{\mathcal{L}}_i^1\left(F_1^1(z)D_0^1(z)\right)\\ &= \mathbf{F}_i^2(z)\mathbf{w}_0^2(z) + \boldsymbol{\mathcal{L}}_i^1\left(F_1^1(z)D_0^1(z)\right)\end{aligned} \tag{5.8}$$

with $\mathbf{F}_i^2(z)$ the prediction-filter matrices given in (5.5). ■

Proposition 1 hints that in the general case of an arbitrary level k, $k > 1$, the calculation of the ODWT subbands from the k-level DWT involves: (a) the single-rate filtering of the DWT subbands of level k ($\mathbf{w}_0^k(z)$), and (b) the cascade application of filtering-and-downsampling operations to the high-frequency subbands of higher-resolution levels of the DWT ($D_0^{k-1}(z),\ldots,D_0^1(z)$). This intuitive link is mathematically formulated next.

B. Calculation of the ODWT Subbands of Level $E = k$

The generalization of Proposition 1 and the corresponding definitions are given below for an arbitrary level $E = k$.

Definition 3: We define $\mathbf{F}_p^{l+1}(z)$ as the prediction- filter matrices of level $E = l+1$, $1 \le l < k$ with $k > 1$, given by:

$$\mathbf{F}_p^{l+1}(z) = \begin{bmatrix}F_{4p}^{l+1}(z) & F_{4p+1}^{l+1}(z)\\ F_{4p+2}^{l+1}(z) & F_{4p+3}^{l+1}(z)\end{bmatrix} = F_{w(1),b_0}^l(z)\mathbf{I} + z^{-(1-b_0)}F_{w(1),1-b_0}^l(z)\mathbf{F}_0^1(z). \tag{5.9}$$

where $F_{w(1)}^l(z)$ are prediction filters of level $E = l$, the filter subscripts $w(n)$ are defined as:

$$w(n) = 4\left\lfloor\frac{p}{2^n}\right\rfloor,\ 1 \le n \le l, \tag{5.10}$$

and p given by (5.1).

Definition 4: We define $\mathbf{F}_d^l(z)$ as the diagonal matrix of prediction filters, given by:

$$\mathbf{F}_d^l(z) = \mathrm{diag}\left(F_{w(1)+1}^l(z) \;\; F_{w(2)+1}^{l-1}(z) \;\cdots\; F_{w(l)+1}^1(z)\right).$$

Definition 5: We define $\mathbf{d}^{k-1,l}(z)$, $1 \le l < k$ with $k > 1$, as the vector of high-frequency subbands of levels $k-1, k-2, \ldots, k-l$ of the DWT. It is given by:

$$\mathbf{d}^{k-1,l}(z) = \left[D_0^{k-1}(z) \;\; D_0^{k-2}(z) \;\cdots\; D_0^{k-l}(z)\right]^T.$$

Proposition 2: The subbands of the ODWT of level $E = k$, $\mathbf{w}_x^k(z)$, with $1 \le x < 2^k$, are given by:

$$\mathbf{w}_x^k(z) = \mathbf{F}_p^{l+1}(z)\mathbf{w}_0^k(z) + \boldsymbol{\mathcal{L}}_p^l\left(\mathbf{F}_d^l(z)\mathbf{d}^{k-1,l}(z)\right) \quad . \tag{5.11}$$

where x denotes the ODWT-subband index at level k (phase x) and is written as $x = 2^l + p$, while l, p are given as in Definition 3 and $l = \lfloor \log_2 x \rfloor$. In the particular case of $l = 0$ corresponding to $k = 1$ and $x = 1$, we set $p = 0$ and $\boldsymbol{\mathcal{L}}_0^0 \equiv [0 \;\; 0]^T$ (zero operator) as well as $F_A^B(z) \equiv 0$ for any indices $A < 0$ or $B \le 0$. For this case, equation (5.11) becomes Proposition 1.

Proof: The proof is inductive. Initially it is noted that equation (5.11) holds for levels $E = 1$ and $E = 2$, since it corresponds to Proposition 1. Subsequently, assume that (5.11) holds for a particular $E = k$. It is now proven that, given the subbands $\mathbf{w}_0^{k+1}(z)$ (DWT subbands of level $k+1$), if (5.11) holds for $E = k$, it holds also for $E = k+1$.

Let us start by performing an inverse DWT in order to calculate the A_0^k subband in function of subbands $\mathbf{w}_0^{k+1}(z)$:

$$A_0^k(z) = \mathcal{D}^{-1}\left(\mathbf{R}_0(z)\mathbf{w}_0^{k+1}(z)\right). \tag{5.12}$$

The derivation of subbands $\mathbf{w}_1^{k+1}(z)$ follows immediately by performing a forward transform with $\mathbf{E}_1(z)$. Hence we derive (5.11) for the special case of $x = 1$ with k replaced by $k+1$. For the remaining subbands of level $k+1$, we can apply equation (5.11) (involving the filters of the $\mathbf{F}_p^{l+1}(z)$ matrix), since it is true for level $E = k$. We can calculate any subband A_x^k, $1 < x < 2^k$, as:

$$A_x^k(z) = \left[F_{4p}^{l+1}(z) \;\; F_{4p+1}^{l+1}(z)\right]\mathbf{w}_0^k(z) + \boldsymbol{\lambda}_{p,\mathcal{H}}^l\left(\mathbf{F}_d^l(z)\mathbf{d}^{k-1,l}(z)\right). \tag{5.13}$$

As shown in Figure V-1(b) for the example of $k = 2$, in order to calculate the $\mathbf{w}_{2x}^{k+1}(z)$ subbands (even-numbered subbands of level $k+1$), we need to perform a single-level forward transform in equation (5.13), retaining the even samples ("classical" DWT):

$$\begin{aligned}\mathbf{w}_{2x}^{k+1}(z) = {} & \mathbf{E}_0(z)\mathcal{D}\left(F_{4p}^{l+1}(z)A_0^k(z)\right) \\ & + \mathbf{E}_0(z)\mathcal{D}\left[F_{4p+1}^{l+1}(z)D_0^k(z) + \boldsymbol{\lambda}_{p,\mathcal{H}}^l\left(\mathbf{F}_d^l(z)\mathbf{d}^{k-1,l}(z)\right)\right].\end{aligned} \tag{5.14}$$

Based on Lemma 1, equation (5.12) and the Noble identity, the last equation can be written as:

$$\mathbf{w}_{2x}^{k+1}(z) = \mathbf{E}_0(z)\left(F_{4p,0}^{l+1}(z)\mathbf{I} + z^{-1}F_{4p,1}^{l+1}(z)\begin{bmatrix} 0 & z \\ 1 & 0 \end{bmatrix}\right)\mathbf{R}_0(z)\mathbf{w}_0^{k+1}(z) \\ +\boldsymbol{\mathcal{L}}_{2p}^{l+1}\left(\mathbf{F}_d^{l+1}(z)\mathbf{d}^{k,l+1}(z)\right) \qquad (5.15)$$

After simplifications, equation (5.15) becomes equivalent to (5.11) with l replaced by $l' = l+1$ and hence $2 \le l' < k+1$, while the ODWT subband index $2x$ of (5.15) is bounded by $2 \le 2x < 2^{k+1}$. In addition, the resulting prediction- filter matrix of (5.15) is expressed as in (5.9) with $p' = \sum_{i=1}^{l'-1} c_i 2^i$ and $c_i = b_{i-1}$, $c_0 = 0$. Hence, equation (5.15) corresponds to (5.11) for the even-numbered subbands, with k replaced by $k+1$.

In order to calculate the subbands $\mathbf{w}_{2x+1}^{k+1}(z)$ (odd-numbered subbands of level $k+1$), we perform a forward DWT retaining the odd samples, reaching, similarly as before, the following:

$$\mathbf{w}_{2x+1}^{k+1}(z) = \mathbf{E}_1(z)\left(F_{4p,0}^{l+1}(z)\mathbf{I} + z^{-1}F_{4p,1}^{l+1}(z)\begin{bmatrix} 0 & z \\ 1 & 0 \end{bmatrix}\right)\mathbf{R}_0(z)\mathbf{w}_0^{k+1}(z) \\ +\boldsymbol{\mathcal{L}}_{2p+1}^{l+1}\left(\mathbf{F}_d^{l+1}(z)\mathbf{d}^{k,l+1}(z)\right) \qquad (5.16)$$

After simplifications, equation (5.16) becomes (5.11) with l replaced by $l' = l+1$ and hence $2 \le l' < k+1$, while the ODWT subband index $2x+1$ of (5.16) is bounded by $3 \le 2x+1 < 2^{k+1}$. In addition, the resulting prediction-filter matrix of (5.16) is expressed as in (5.9) with $p' = \sum_{i=1}^{l'-1} c_i 2^i$ and $c_i = b_{i-1}$, $c_0 = 1$. As a result, (5.16) corresponds to equation (5.11) for the odd-numbered subbands, with k replaced by $k+1$. One concludes that Proposition 2 is true for the case of $E = k+1$. This means, by induction, that it is true for every E, $E \ge 1$. ■

For each decomposition level k of the DWT, Proposition 2 given in equation (5.11) consists of a *single-rate* and a *multi-rate* calculation part. The first consists of the convolution of the critically-sampled subbands of level k with the prediction filters of the matrix $\mathbf{F}_p^{l+1}(z)$, $0 \le l < k$, while the second consists of the convolutions of the vector of high-frequency DWT subbands of levels $k-1, k-2, .., k-l$ with the diagonal matrix of prediction filters, followed by the successive filtering-and-downsampling with the analysis filters. This result is summarized below.

Proposition 3: The complete-to-overcomplete discrete wavelet transform of any level k is:

$$\mathbf{w}_{\mathrm{ODWT}}^{k}(z) = \begin{bmatrix} \mathbf{P}^1(z) \\ \mathbf{P}^2(z) \\ \vdots \\ \mathbf{P}^k(z) \end{bmatrix}\mathbf{w}_0^k(z) + \begin{bmatrix} \boldsymbol{\mathcal{Q}}^0 \\ \boldsymbol{\mathcal{Q}}^1 \\ \vdots \\ \boldsymbol{\mathcal{Q}}^{k-1} \end{bmatrix}\left(\mathbf{F}_d^{k-1}(z)\mathbf{d}^{k-1,k-1}(z)\right) \quad . \qquad (5.17)$$

with:

$$\mathbf{w}_{\mathrm{ODWT}}^{k}(z)=\begin{bmatrix}\mathbf{w}_{1}^{k}(z)\\ \vdots\\ \mathbf{w}_{2^{k}-1}^{k}(z)\end{bmatrix},\ \mathbf{P}^{l+1}(z)=\begin{bmatrix}\mathbf{F}_{0}^{l+1}(z)\\ \mathbf{F}_{1}^{l+1}(z)\\ \vdots\\ \mathbf{F}_{2^{l}-1}^{l+1}(z)\end{bmatrix},\ \boldsymbol{\mathcal{Q}}^{l}=\begin{bmatrix}\boldsymbol{\mathcal{L}}_{0}^{l} & 0 & \cdots & 0\\ \boldsymbol{\mathcal{L}}_{1}^{l} & 0 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{\mathcal{L}}_{2^{l}-1}^{l} & & \cdots & 0\end{bmatrix}$$

a $2^{l+1}\times(k-1)$ array of operators for every $0\le l<k$, and:

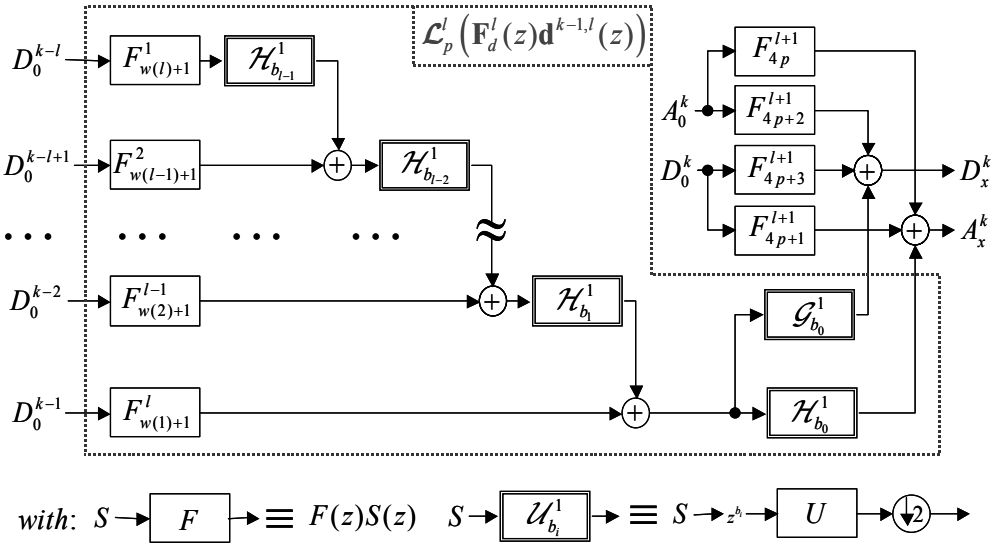
$$\boldsymbol{\mathcal{Q}}^{0}=\boldsymbol{\mathcal{L}}_{0}^{0}=\begin{bmatrix}0\\ 0\end{bmatrix}.$$

5.1.3.2 Properties of the Derived Formulation

A straightforward calculation of any given subband pair $\mathbf{w}_{x}^{k}(z)$ of Proposition 2 is given schematically in Figure V-2. The figure gives also a pseudo-program describing the initialization procedure used to select the appropriate filters involved in the calculation of an arbitrary set of ODWT subbands $\mathbf{w}_{x}^{k}(z)$ of (5.11). The calculation of the multi-rate part of (5.11) shown in Figure V-2 is based on the following derivation:

$$\begin{aligned}\boldsymbol{\mathcal{L}}_{p}^{l}\left(\mathbf{F}_{d}^{l}(z)\mathbf{d}^{k-1,l}(z)\right)=\begin{bmatrix}\mathcal{H}_{b_0}^{1}\\ \mathcal{G}_{b_0}^{1}\end{bmatrix}\Big(F_{w(1)+1}^{l}(z)D_{0}^{k-1}(z)+\mathcal{H}_{b_1}^{1}\left(F_{w(2)+1}^{l-1}(z)D_{0}^{k-2}(z)+\right.\\ \left.+\mathcal{H}_{b_2}^{1}\left(\cdots+\mathcal{H}_{b_{l-1}}^{1}\left(F_{w(l)+1}^{1}(z)D_{0}^{k-l}(z)\right)\cdots\right)\right)\end{aligned}\tag{5.18}$$

which comes from the property that the downsampling and summation operations can be interchanged. Figure V-2 shows that the proposed approach involves the minimum number of downsampling operations and no upsampling is used.



(a)

Initialization of calculation of A_x^k, D_x^k, $1 \le x < 2^k$ *of the ODWT*:

- ***Start.*** *Input the desired x. Calculate*:
  $l = \lfloor \log_2 x \rfloor$, $p = x - 2^l$.
- If l>0:
  - *Find* b_i *from the binary representation of* p:
    $p = \sum_{i=0}^{l-1} b_i \cdot 2^i$.
  - *For* $1 \le n \le l$:
    - *Calculate the filter indices* $w(n)$, *as*:
      $w(m) = 4\left\lfloor \frac{p}{2^n} \right\rfloor$.
    - *Select the filter* $F_{w(n)+1}^{l+1-n}$.
    - *Input the subband* D_0^{k-n}.
- *Select the filters* $F_{4p+\{0,1,2,3\}}^{l+1}$.
- *Input the critically-sampled DWT subbands* A_0^k & D_0^k. ***End***.

(b)

Figure V-2. (a) The straightforward calculation of the CODWT based on Proposition 2. (b) The initialization of the calculation of A_x^k, D_x^k.

In the following parts of this section, we present two symmetry properties for the prediction filters of each decomposition level as well as an efficient calculation scheme of the single-rate and multi-rate parts of Proposition 3.

Proposition 4: For the prediction-filter matrices $\mathbf{F}_p^{l+1}(z)$ of any level $E = l+1$, $1 \le l < k$ of biorthogonal point-symmetric filter-banks, the following properties hold:

$$F_{4m}^{k}(z) = zF_{4(2^{k-1}-m-1)}^{k}(z^{-1}). \tag{5.19}$$

$$F_{4m+1}^{k}(z) = F_{4(2^{k-1}-m-1)+1}^{k}(z^{-1}). \tag{5.20}$$

$$F_{4m+2}^{k}(z) = z^{2}F_{4(2^{k-1}-m-1)+2}^{k}(z^{-1}). \tag{5.21}$$

$$F_{4m+3}^{k}(z) = zF_{4(2^{k-1}-m-1)+3}^{k}(z^{-1}). \tag{5.22}$$

with $0 \leq m < 2^{k-2}$ and $k > 1$.

Proof. Proposition 4 will be demonstrated by mathematical induction. Starting from the first decomposition level, the following relationships are used for the prediction filters:

$$F_{0}^{1}(z^{-1}) = z^{-1}F_{0}^{1}(z) \tag{5.23}$$

$$F_{1}^{1}(z^{-1}) = F_{1}^{1}(z) \tag{5.24}$$

$$F_{2}^{1}(z^{-1}) = z^{-2}F_{2}^{1}(z) \tag{5.25}$$

$$F_{3}^{1}(z^{-1}) = z^{-1}F_{3}^{1}(z) \tag{5.26}$$

$$F_{0}^{1}(z) = -F_{3}^{1}(z) \tag{5.27}$$

$$F_{3}^{1}(z) = -zF_{0}^{1}(z^{-1}) \tag{5.28}$$

The proof of the equations (5.23)–(5.28) follows immediately by:

- deriving the relations between the polyphase components of the filter-bank based on the relationships:

  $$G(z) = G(z^{-1}), zH(z) = z^{-1}H(z^{-1}) \tag{5.29}$$

  which hold for biorthogonal point-symmetric filter-banks [11],

- using these relations in conjunction with the definition of the prediction filters of equation (5.4).

The symmetries of Proposition 4 that correspond to the case $k = 2$ are:

$$F_{0}^{2}(z) = zF_{4}^{2}(z^{-1}) \tag{5.30}$$

$$F_{1}^{2}(z) = F_{5}^{2}(z^{-1}) \tag{5.31}$$

$$F_{2}^{2}(z) = z^{2}F_{6}^{2}(z^{-1}) \tag{5.32}$$

$$F_{3}^{2}(z) = zF_{7}^{2}(z^{-1}) \tag{5.33}$$

To prove this proposition, we use equation (5.23) with the Type-I polyphase components of filter $F_0^1(z)$, yielding $F_{0,0}^1(z^{-1}) = z^{-1}F_{0,1}^1(z)$ and $F_{0,1}^1(z^{-1}) = z^{-1}F_{0,0}^1(z)$. The proof of (5.30)–(5.33) is concluded by replacing filters $F_{\{4,5,6,7\}}^2(z^{-1})$ from their definition given by (5.9) with $l = 1$, and by replacing $F_{0,0}^1(z^{-1})$, $F_{0,1}^1(z^{-1})$ and $F_{\{0,1,2,3\}}^1(z^{-1})$ from the last two equations and (5.23)–(5.26), respectively.

We now assume that Proposition 4 is true for level k and we derive the symmetry relationships for level $k+1$. In the Type-I polyphase components of F_{4m}^k, with $0 \le m < 2^{k-2}$ we can replace the terms $F_{4m}^k(z^{1/2}), F_{4m}^k(-z^{1/2})$ by using (5.19), yielding:

$$F_{4m,0}^k(z) = zF_{4(2^{k-1}-m-1),1}^k(z^{-1}), F_{4m,1}^k(z) = zF_{4(2^{k-1}-m-1),0}^k(z^{-1}) \tag{5.34}$$

with $m = 0,1,..,2^{k-2}-1$ and $k > 1$.

Let us start by proving the symmetry properties for the filters of the form F_{4m}^{k+1} (corresponding to equation (5.19)), with $0 \le m < 2^{k-1}$; these properties are separately derived for m even and m odd. For the case of m even, we define $m = 2j$, with $0 \le j < 2^{k-2}$. Hence, from (5.9) with $l = k$ and from equation (5.28) we obtain:

$$F_{4m}^{k+1}(z) = F_{4j,0}^k(z) + F_0^1(z^{-1})F_{4j,1}^k(z). \tag{5.35}$$

Since $0 \le j < 2^{k-2}$, we can substitute (5.34) in the last equation, obtaining:

$$F_{4m}^{k+1}(z) = z(F_{4(2^{k-1}-j-1),1}^k(z^{-1}) + F_0^1(z^{-1}) \cdot F_{4(2^{k-1}-j-1),0}^k(z^{-1})). \tag{5.36}$$

From the definition of $F_{4m}^{k+1}(z)$ (equation (5.9) with $l = k$) and from the definition of m, (5.36) becomes:

$$F_{4m}^{k+1}(z) = zF_{8(2^{k-1}-j-1)+4}^{k+1}(z^{-1}) \Leftrightarrow F_{4m}^{k+1}(z) = zF_{4(2^k-m-1)}^{k+1}(z^{-1}) \tag{5.37}$$

for $m = 0,2,4,..,2^{k-1}-2$.

For the case of m odd, we define $m = 2j+1$ with $0 \le j < 2^{k-2}$. Thus, from the definition of $F_{4m}^{k+1}(z)$ (equation (5.9) with $l = k$) and from equation (5.28) we obtain:

$$\begin{aligned} F_{4m}^{k+1}(z) = F_{8j+4}^{k+1}(z) = F_{4j,1}^k(z) + F_0^1(z)F_{4j,0}^k(z) \\ \Leftrightarrow F_{4m}^{k+1}(z) = F_{4j,1}^k(z) - zF_3^1(z^{-1})F_{4j,0}^k(z) \end{aligned}. \tag{5.38}$$

Since $0 \le j < 2^{k-2}$, we can substitute (5.34) in equation (5.38) and we get:

$$F_{4m}^{k+1}(z) = z(F_{4(2^{k-1}-j-1),0}^k(z^{-1}) - zF_3^1(z^{-1})F_{4(2^{k-1}-j-1),1}^k(z^{-1})). \tag{5.39}$$

From (5.9) with $l = k$ and from the definition of m, equation (5.39) becomes:

$$F_{4m}^{k+1}(z) = zF_{8(2^{k-1}-j-1)}^{k+1}(z^{-1}) \Leftrightarrow F_{4m}^{k+1}(z) = zF_{4(2^k-m-1)}^{k+1}(z^{-1}) \tag{5.40}$$

for $m = 1,3,5,..,2^{k-1}-1$.

Thus, the combination of (5.37) and (5.40) yields (5.19) with the k replaced by $k+1$.

To prove the symmetry properties for the prediction filters of the form F_{4m+1}^{k+1}, with $0 \le m < 2^{k-1}$, we follow the same rationale as before. Hence, these properties are separately derived for m even and m odd. For the case of m even, we denote $m = 2j$ with $0 \le j < 2^{k-2}$. By using the definition of F_{4m+1}^{k+1} (equation (5.9) with $l = k$) and from (5.24) we obtain:

$$F_{4m+1}^{k+1}(z) = z^{-1}F_1^1(z^{-1})F_{4j,1}^k(z).$$

Since $0 \le j < 2^{k-2}$, we can substitute (5.34) in the last equation, obtaining:

$$F_{4m+1}^{k+1}(z) = F_1^1(z^{-1})F_{4(2^{k-1}-j-1),0}^k(z^{-1}).$$

From equation (5.9) with $l = k$ and from the definition of m we obtain:

$$F_{4m+1}^{k+1}(z) = F_{8(2^{k-1}-j-1)+5}^{k+1}(z^{-1}) \Leftrightarrow F_{4m+1}^{k+1}(z) = F_{4(2^k-m-1)+1}^{k+1}(z^{-1}) \tag{5.41}$$

for $m = 0,2,4,..,2^{k-1}-2$.

For the case of m odd, we denote $m = 2j+1$ with $0 \le j < 2^{k-2}$. Thus, from (5.9) with $l = k$ and from (5.24) we obtain:

$$F_{4m+1}^{k+1}(z) = F_{8j+5}^{k+1}(z) = F_1^1(z)F_{4j,0}^k(z) \Leftrightarrow F_{4m+1}^{k+1}(z) = F_1^1(z^{-1})F_{4j,0}^k(z). \tag{5.42}$$

Since $0 \le j < 2^{k-2}$, we can substitute (5.34) in (5.42), obtaining:

$$F_{4m+1}^{k+1}(z) = zF_1^1(z^{-1})F_{4(2^{k-1}-j-1),1}^k(z).$$

From equation (5.9) with $l = k$ and from the definition of m we obtain:

$$F_{4m+1}^{k+1}(z) = F_{8(2^{k-1}-j-1)+1}^{k+1}(z^{-1}) \Leftrightarrow F_{4m+1}^{k+1}(z) = F_{4(2^k-m-1)+1}^{k+1}(z^{-1}). \tag{5.43}$$

for $m = 1,3,5,..,2^{k-1}-1$.

Thus, the combination of (5.41) and (5.43) yields (5.20) with k replaced by $k+1$.

For the prediction filters of the form F_{4m+2}^{k+1}, with $0 \le m < 2^{k-1}$, we follow exactly the same reasoning as for filter F_{4m+1}^{k+1}, but now their definitions from equation (5.9) (with $l = k$) are used in combination with (5.25) for the even and odd values of m. Equivalently, for the filters F_{4m+3}^{k+1}, $0 \le m < 2^{k-1}$, we follow similar steps as for the proof given for the filters F_{4m}^{k+1}; in this case their definitions from equation (5.9) (with $l = k$) are used in combination with (5.26) for the even and odd values of m. As a result we reach equations (5.21) and (5.22) with k replaced by $k+1$. ■

A practical example of the symmetries of Proposition 4 as well as the derived equations (5.23)–(5.28) is given in Table V-II and Table V-III, where the prediction filters for two popular point-symmetric biorthogonal filter-banks are presented.

Degree in Z	**5/3 filter-bank**			
	$F_0^1(z)$	$F_1^1(z)$	$F_2^1(z)$	$F_3^1(z)$
2	-0.0625	0.03125	-0.1250	0.0625
1	0.5625	-0.5	0.25	-0.5625
0	0.5625	0.9375	-0.125	-0.5625
-1	-0.0625	-0.5		0.0625
-2		0.03125		
Degree in Z	**9/7 filter-bank**			
	$F_0^1(z)$	$F_1^1(z)$	$F_2^1(z)$	$F_3^1(z)$
4	-0.00244140625001	0.00143099204607	-0.00416526737096	0.00244140625001
3	0.02392578125006	-0.00893829770284	0.05562204500420	-0.02392578125006
2	-0.11962890624961	0.09475201933935	-0.18500077367828	0.11962890624961
1	0.59814453124955	-0.32145927076690	0.26708799209208	-0.59814453124955
0	0.59814453124955	0.46842911416863	-0.18500077367828	-0.59814453124955
-1	-0.11962890624961	-0.32145927076690	0.05562204500420	0.11962890624961
-2	0.02392578125006	0.09475201933935	-0.00416526737096	-0.02392578125006
-3	-0.00244140625001	-0.00893829770284		0.00244140625001
-4		0.00143099204607		

Table V-II. The prediction filters of level one for the 5/3 and the 9/7 filter-banks shown in Table V-I.

Degree in Z	5/3 filter-bank			
	$F_0^2(z)$	$F_1^2(z)$	$F_2^2(z)$	$F_3^2(z)$
2	-0.03515625	0.017578125	-0.0703125	0.03515625
1	0.2578125	-0.283203125	0.1484375	-0.3828125
0	0.843750	0.55859375	-0.0859375	0.28125
-1	-0.0703125	-0.33984375	0.0078125	0.0703125
-2	0.00390625	0.048828125		-0.00390625
-3		-0.001953125		
	$F_4^2(z)$	$F_5^2(z)$	$F_6^2(z)$	$F_7^2(z)$
3	0.00390625	-0.001953125	0.0078125	-0.00390625
2	-0.0703125	0.048828125	-0.0859375	0.0703125
1	0.843750	-0.33984375	0.1484375	0.28125
0	0.2578125	0.55859375	-0.0703125	-0.3828125
-1	-0.03515625	-0.283203125		0.03515625
-2		0.017578125		
Degree in Z	**9/7 filter-bank**			
	$F_0^2(z)$	$F_1^2(z)$	$F_2^2(z)$	$F_3^2(z)$
5	-0.00005841255188	0.00003423760266	-0.00009965727597	0.00005841255188
4	-0.00088787078858	0.00064208431103	-0.00116063101768	0.00088787078858
3	0.01174092292788	-0.00325056581524	0.02934202037389	-0.01174092292788
2	-0.06254196166961	0.05005002314451	-0.11091074747671	0.05765914916960
1	0.26671171188334	-0.19238483073456	0.17732657799233	-0.50596952438255
0	0.88179588317802	0.31072164151829	-0.14082618249241	0.31449317932109
-1	-0.12007284164365	-0.24526493865167	0.05464973467748	0.16792440414377
-2	0.02710342407219	0.09377374163572	-0.00869377426124	-0.02710342407219
-3	-0.00403046607971	-0.01586242405270	0.00036249037151	0.00403046607971
-4	0.00023365020752	0.00169389067232	0.00001016910979	-0.00023365020752
-5	0.00000596046448	-0.00014936599745		-0.00000596046448
-6		-0.00000349363292		
	$F_4^2(z)$	$F_5^2(z)$	$F_6^2(z)$	$F_7^2(z)$
6	0.00000596046448	-0.00000349363292	0.00001016910979	-0.00000596046448
5	0.00023365020752	-0.00014936599745	0.00036249037151	-0.00023365020752
4	-0.00403046607971	0.00169389067232	-0.00869377426124	0.00403046607971
3	0.02710342407219	-0.01586242405270	0.05464973467748	-0.02710342407219
2	-0.12007284164365	0.09377374163572	-0.14082618249241	0.16792440414377
1	0.88179588317802	-0.24526493865167	0.17732657799233	0.31449317932109
0	0.26671171188334	0.31072164151829	-0.11091074747671	-0.50596952438255
-1	-0.06254196166961	-0.19238483073456	0.02934202037389	0.05765914916960
-2	0.01174092292788	0.05005002314451	-0.00116063101768	-0.01174092292788
-3	-0.00088787078858	-0.00325056581524	-0.00009965727597	0.00088787078858
-4	-0.00005841255188	0.00064208431103		0.00005841255188
-5		0.00003423760266		

Table V-III. The prediction filters of level two for the 5/3 and the 9/7 filter-banks.

This section is concluded with an extension of the first part of Proposition 4 (equation (5.19)) for all types of perfect-reconstruction filter-banks.

Proposition 5 (extension of Proposition 4): For the prediction-filter matrices $\mathbf{F}_p^{l+1}(z)$ of any level $E = l+1$, $1 \le l < k$ of any perfect-reconstruction filter-bank, the following property holds:

$$F_{4m}^k(z) = zF_{4(2^{k-1}-m-1)}^k(z^{-1}). \tag{5.44}$$

with $0 \le m < 2^{k-2}$ and $k > 1$.

Proof. To prove (5.44) for any perfect-reconstruction filter-bank, it is enough to show that, under conditions for perfect reconstruction that satisfy the general definition of a wavelet filter-bank (Definition 1), the proof demonstrated for (5.19) holds.

In Definition 1 we have assumed zero-delay perfect reconstruction with $\det \mathbf{E}_0(z) = -1$, which can be satisfied with: $G(z) = G(z^{-1})$, $H(z) = z^{-2}H(z^{-1})$ for biorthogonal point-symmetric filter-banks, $G(z) = -z^{-1}G(z^{-1})$, $H(z) = z^{-1}H(z^{-1})$ for biorthogonal half-point symmetric filter-banks, and $G(z) = z^{-1}H(-z^{-1})$ for orthogonal filter-banks. Based on these equations, the proof of (5.30) (first part of the induction for the proof of (5.44)) is concluded for each case by deriving the symmetry relations between the Type-I polyphase components of the analysis filters and using them in combination with the definition of $F_0^1(z)$ from (5.4). For the remainder of the proof, one follows the steps demonstrated by equations (5.34)–(5.40) since they are all based on the validity of (5.30) and the general definition of the prediction filters of (5.9). ■

Corollary 1: For the prediction-filter matrices $\mathbf{F}_p^{l+1}(z)$ of any level $E = l+1$, $1 \le l < k$ of any perfect-reconstruction filter-bank, the following property holds:

$$F_{4m,0}^k(z) = zF_{4(2^{k-1}-m-1),1}^k(z^{-1}),\ F_{4m,1}^k(z) = zF_{4(2^{k-1}-m-1),0}^k(z^{-1}) \tag{5.45}$$

with $m = 0,1,..,2^{k-2}-1$ and $k > 1$.

Based on the validity of Proposition 5, equation (5.45) is the extension of (5.34), for all perfect-reconstruction filter-banks. This equation is very useful for the derivation of one of the efficient calculation algorithms presented in the next section.

5.2 Architectures and Fast Algorithms

In this section, we are focusing on the calculation of the full CODWT, given by Proposition 3, and any subband pair of the ODWT, given by Proposition 2.

5.2.1 Fast Algorithm for the Calculation of the CODWT

Starting with the calculation of the CODWT, although the straightforward design of Figure V-2 provides the complete transform if it is used for all the subbands of a given resolution level, it fails to exploit the symmetry properties demonstrated by Proposition 4 and Proposition 5. As a result, with respect to the required arithmetic operations, it is not a good implementation solution.

It is easy to observe that all the derived symmetry properties of the previous section refer to the single-rate part of Proposition 3. As a result, in the following we separate our discussion to the design of an architecture for the single-rate and the multi-rate part of Proposition 3.

5.2.1.1 Calculation of the Single-rate Part of Proposition 3

Based on Corollary 1, having the same input signal, the convolutions with filters $F^l_{4m,0}(z)$, $F^l_{4m,1}(z)$, $m = 0,1,..,2^{l-2}-1$, $l > 1$, can be reused to produce the convolutions with the filters $F^l_{4(2^{l-1}-m-1),1}(z)$, $F^l_{4(2^{l-1}-m-1),0}(z)$, respectively. This is achieved by using a lattice structure (LS) such as the one depicted in Figure V-3. This is of great practical importance since the utilization of this symmetry in the prediction filters of (5.9) leads to the reduction of the necessary arithmetic operations for the proposed CODWT. As an example, based on Corollary 1, if we expand the form of the prediction filters of (5.9) in the general expression of (5.17) for $k = 3$, with the setting $\mathbf{d}_0^{2,2}(z) = \mathbf{0}$ (single-rate calculation) we reach the following result:

$$\begin{aligned}
\mathbf{w}_1^3(z) &= \mathbf{F}_0^1(z)\mathbf{w}_0^3(z), \\
\left[\mathbf{w}_2^3(z) \quad \mathbf{w}_3^3(z)\right] &= \left[\mathbf{w}_0^3(z) \quad \mathbf{w}_1^3(z)\right]\begin{bmatrix} F_{0,0}^1(z) & zF_{0,0}^1(z^{-1}) \\ F_{0,0}^1(z^{-1}) & F_{0,0}^1(z) \end{bmatrix}, \\
\left[\mathbf{w}_4^3(z) \quad \mathbf{w}_5^3(z) \quad \mathbf{w}_6^3(z) \quad \mathbf{w}_7^3(z)\right] &= \left[\mathbf{w}_0^3(z) \quad \mathbf{w}_1^3(z)\right]\begin{bmatrix} F_{0,0}^2(z) & zF_{4,0}^2(z^{-1}) & F_{4,0}^2(z) & zF_{0,0}^2(z^{-1}) \\ F_{4,0}^2(z^{-1}) & F_{0,0}^2(z) & F_{0,0}^2(z^{-1}) & F_{4,0}^2(z) \end{bmatrix}.
\end{aligned} \tag{5.46}$$

The set of equations (5.46) shows that the single-rate calculation of the proposed CODWT is simplified to the convolutions with filters $\mathbf{F}_0^1(z)$ and filters $F_{0,0}^1(z)$, $F_{0,0}^2(z)$, $F_{4,0}^2(z)$ with the use of a set of lattice structures of Figure V-3. To demonstrate this in practice, based on equation (5.46), Figure V-4(a) shows an architecture for the calculation of the ODWT subbands of level three under the single-rate construction for resolution scalability. In the figure, a set of three processor elements (PE) is used for the production of subbands $\mathbf{w}_i^3$, $1 \le i < 8$. The definition of a PE is given in Figure V-4(b); it contains four lattice structures such as the one shown in Figure V-3. In the general case of decomposition level k, the architecture of Figure V-4(a) uses $2^{k-1}-1$ PEs for the production of the $\mathbf{w}_{\text{ODWT}}^k$ subbands.

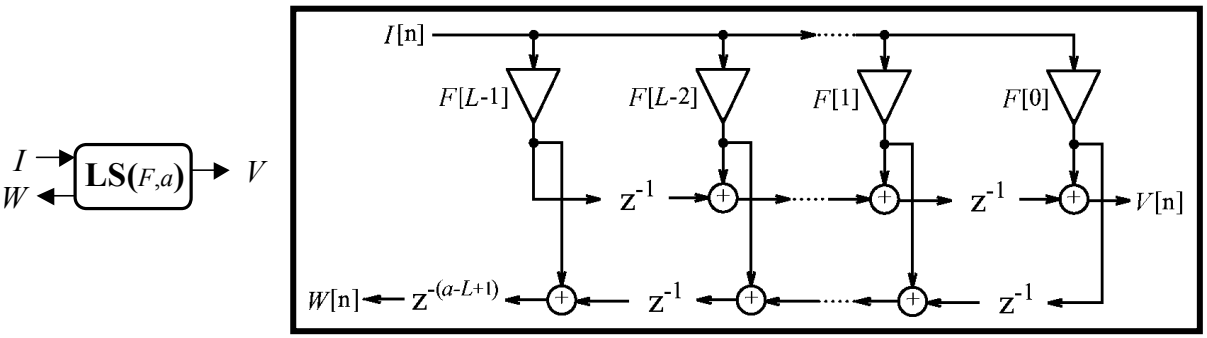
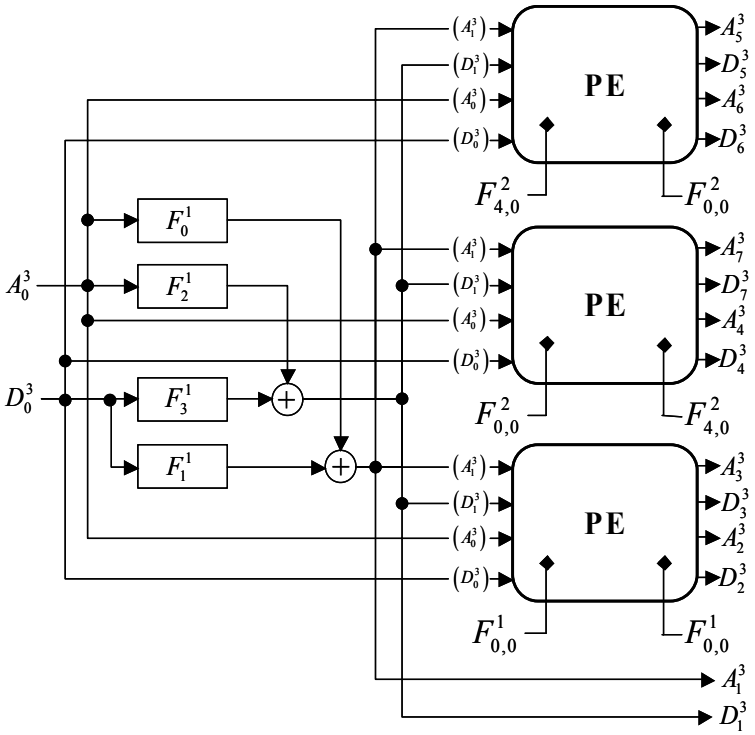
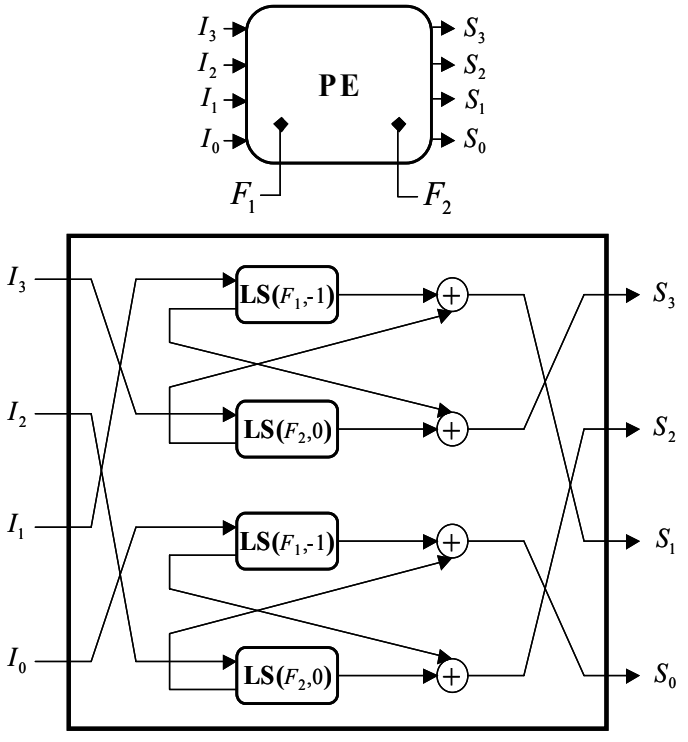


Figure V-3. A lattice structure that implements the two convolutions $V(z) = F(z)I(z)$ and $W(z) = z^{-a}F(z^{-1})I(z)$ for a filter F with L taps using L multipliers.



(a)



(b)

Figure V-4. (a) An architecture for the calculation of the single-rate part of equation (5.17). An example with $k = 3$ is given, where the ODWT is computed using subbands A_0^3, D_0^3 and 3 processing elements (PE). (b) The design of a PE.

5.2.1.2 Calculation of the Multi-rate Part of Proposition 3

Concerning the multi-rate part of the $\mathbf{w}_{\mathrm{ODWT}}^{k}$ given in Proposition 3, for each level k, the calculation of the single-rate part is augmented by each of the terms $\mathcal{L}_p^l\left(\mathbf{F}_d^l(z)\mathbf{d}^{k-1,l}(z)\right)$, $0 \le l < k$. The explicit form of these terms was given in (5.18); it consists of the convolutions $F_{4m+1}^{l-i+1}(z)D^{k-i}(z)$, $m = 0,1,..,2^{l-2}-1$, $1 \le i \le l$, i.e. the elements of $\mathbf{F}_d^l(z)$ and $\mathbf{d}^{k-1,l}(z)$ followed by filtering-and-downsampling with the analysis filter-bank. Figure V-5 presents an efficient architecture for the multi-rate part of Proposition 3 for $k = 3$. In the figure, the recursive definition of filters $F_{w(i)+1}^{l-i+1}(z)$ given by (5.9) and the symmetry property of Corollary 1 are used for the reduction of the necessary multiplications. Specifically, the lattice structure of Figure V-3 is used for the production of the results of the convolutions with filters $F_1^2(z)$, $F_5^2(z)$. In general, the figure demonstrates that a recursive calculation is used: for each subband $D_0^l(z)$, $1 \le l < k$, a new stage is inserted with filter $F_1^1(z)$ and $2^{l-1}-1$ lattice structures (left side of the figure); also, $2^l - 2$ low-pass filtering-and-downsampling operations are applied to subband $D_0^{l-1}(z)$, $1 < l < k$. In addition, $2^k - 2$ analysis filter-banks are used for $D_0^{k-1}(z)$.

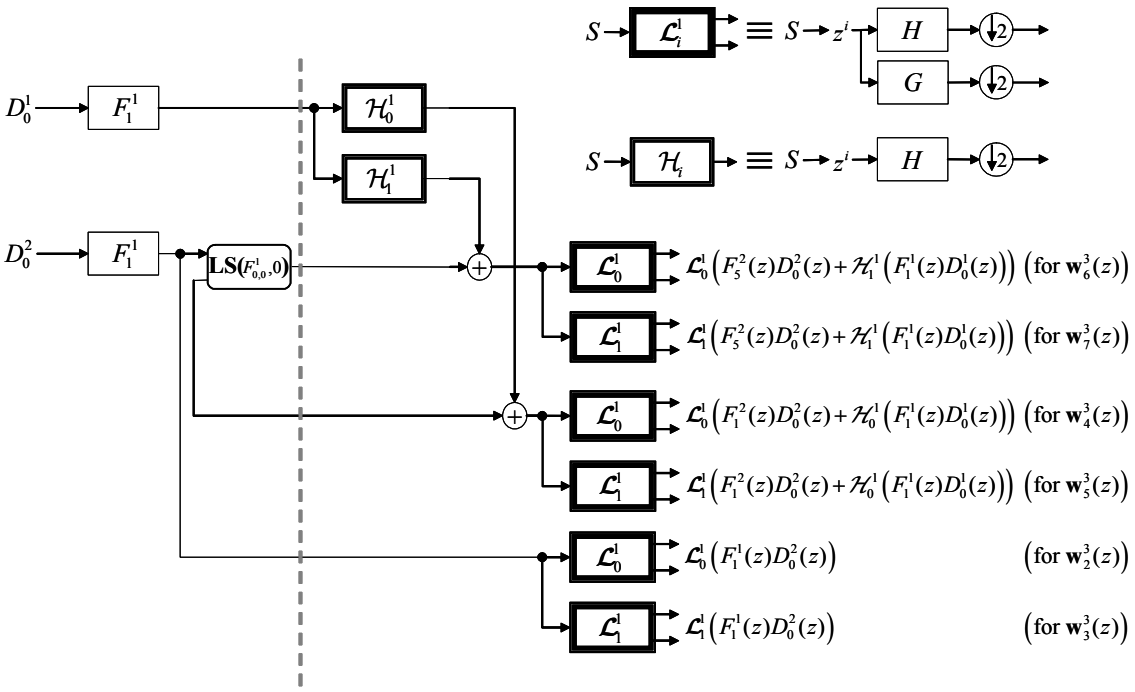


Figure V-5. An architecture for the calculation of the multi-rate part of equation (5.17). An example with $k = 3$ is given.

5.2.2 Fast Algorithm for the Single-rate Calculation of Proposition 2 for Biorthogonal Point-symmetric Filter-banks

Although the algorithm of the previous subsection provides a fast computation of the CODWT by reusing results from other phases based on the architectures of Figure V-4 and Figure V-5, this algorithm may be considered to be complex to realize in a system with restricted resources, such as an

embedded processor. This is due to the fact that, in order to allow maximum hardware utilization, the architecture of single-rate calculation of Figure V-4 demands a large number of filter kernels operating in parallel. Moreover, as shown by Figure V-4, for each input pair of samples, a number of phases is produced simultaneously, which demands a relatively-large number of registers to store the intermediate results of the calculation.

As demonstrated in [11], if one focuses on biorthogonal point-symmetric filter-banks, the prediction-filter symmetries of Proposition 4 can be directly used to produce the low- or high-frequency subbands of pairs of ODWT phases. Although less intermediate results are reused for such a computation scheme, this has the potential of producing a fast algorithm that utilizes only four filter-kernels in parallel. Since this scenario is targeted towards platforms with limited hardware resources, we are not going to discuss the multi-rate part of the CODWT; if it is deemed necessary to be used by a specific application, the architecture of Figure V-5 of Subsection 5.2.1.2 can be used for this case as well.

The symmetries of Proposition 4 indicate that for point-symmetric biorthogonal filter-banks, for the calculation of the subbands $A_i^k, D_i^k,\ 1 < i < 2^k,\ k > 1$, we can derive half of the corresponding F-filters as the time-inverses of the other half of the filters under some shifts. Specifically, the filters are time-inversed in groups of four filters that lay in a "mirror" fashion in the group of prediction filters. Thus, by looking at the indices of (5.19)–(5.22), the first four F-filters are related with the last four, the second four F-filters with the penultimate four, and so on. This is also numerically shown in the two examples of Table V-III where we display the prediction filter taps for the biorthogonal point-symmetric 5/3 and 9/7 filter-banks for $k = 2$. One can exploit this simple symmetry to reduce the number of required multiplications.

In this case, one can also explore the additional approach of approximating the prediction filters by setting to zero all taps that are smaller than a threshold [11]. In this way, the size of the filters of each level is reduced, while a good approximation of the final result is obtained. We note that this technique cannot be applied in the LBS approach since the lifting coefficients and the taps of the biorthogonal filter-banks do not have magnitudes below the chosen thresholds, which were experimentally chosen for the prediction-filters method so that adequate precision for very high-bitrate coding is obtained in the ODWT domain: a PSNR above 50 dB was imposed in every ODWT subband constructed with the single-rate prediction-filters method in comparison to the construction with the LL-LBS algorithm. This was verified on a set of JPEG-2000 test images. An experimental evaluation of this algorithm against the LL-LBS revealed that computational reductions can be obtained under thresholding, which in some cases can be significant, i.e. reductions of about 40-50% in the number of required multiplications [11]. The reader is referred to [11] for further details on the computational reductions with this algorithm.

5.3 Complexity Analysis and Experimental Evaluation

We evaluate the computational requirements of the proposed architecture of subsections 5.2.1.1 and 5.2.1.2 versus the architecture of the conventional approach that uses the IDWT and LBS algorithm for the production of the $\mathbf{w}_{\text{ODWT}}^k(z)$ subbands. In addition to that, by focusing on practical

applications [5, 20] [2] [3], we examine the computational complexity and delay for the transform part of a system that encodes/decodes using the CODWT in a resolution-scalable framework. The computational complexity is formulated based on the number of required multiplications and additions. Furthermore, the one-dimensional case of an N-sample signal is analyzed, with the extension to two dimensions following immediately.

5.3.1 Computational Complexity for the calculation of the $\mathbf{w}_{\mathrm{ODWT}}^{k}(z)$ Subbands

5.3.1.1 General

For a one-level wavelet transform of an N-sample signal, the computational complexity can be expressed as:

$$\Lambda_C(N) = \Upsilon_C(N) = \left(\tfrac{N}{2}\mathrm{I}_{\mathrm{multiply}}\right)\mathrm{Cost}_{\mathrm{multiply}} + \left(\tfrac{N}{2}\mathrm{I}_{\mathrm{add}}\right)\mathrm{Cost}_{\mathrm{add}},$$

where $\Lambda_C(N)$ and $\Upsilon_C(N)$ denote the complexity of a decomposition and reconstruction of N input samples respectively, $\mathrm{Cost}_{\mathrm{multiply}}$ and $\mathrm{Cost}_{\mathrm{add}}$ express the implementation complexity of one multiplication and addition respectively, and $\mathrm{I}_{\mathrm{multiply}}$, $\mathrm{I}_{\mathrm{add}}$ are factors that denote the number of multiplications and additions for each application of the filter-bank to the input. These factors depend on the implementation technique (convolution or lifting [23]). For example, by using the classical lifting factorization proposed in [23] for the 9/7 filter-bank, we have $\mathrm{I}_{\mathrm{multiply}} = 4$, $\mathrm{I}_{\mathrm{add}} = 8$ instead of $\mathrm{I}_{\mathrm{multiply}} = 9$ and $\mathrm{I}_{\mathrm{add}} = 14$ that is achieved with convolution[1]. However, the gain in multiplications and additions of the lifting implementation comes only when both the low and high-frequency subbands are used for the reconstruction, or conversely when both are produced from a one-level decomposition [23]. Hence, for a decomposition where only the low-frequency (average) or the high-frequency (detail) subband is produced, the computational complexity of the transform of an N-sample signal under a convolution-based or lifting-based implementation is:

$$\Lambda_A(N) = \left(\tfrac{N}{2}\mathrm{f}_{H,\mathrm{multiply}}\right)\mathrm{Cost}_{\mathrm{multiply}} + \left(\tfrac{N}{2}\mathrm{f}_{H,\mathrm{add}}\right)\mathrm{Cost}_{\mathrm{add}},$$

$$\Lambda_D(N) = \left(\tfrac{N}{2}\mathrm{f}_{G,\mathrm{multiply}}\right)\mathrm{Cost}_{\mathrm{multiply}} + \left(\tfrac{N}{2}\mathrm{f}_{G,\mathrm{add}}\right)\mathrm{Cost}_{\mathrm{add}},$$

respectively, where the factors $\mathrm{f}_{H,\mathrm{multiply}}$, $\mathrm{f}_{G,\mathrm{add}}$ denote the computations (multiplications, additions) performed by one application of each analysis filter. In general, if T_U denotes the number of taps of analysis filter $U = \{H,G\}$, $\mathrm{f}_{U,\mathrm{multiply}} = \left\lfloor \frac{T_U+1}{2} \right\rfloor$ if U is a symmetric analysis filter and $\mathrm{f}_{U,\mathrm{multiply}} = T_U$ otherwise. Also, in all cases, $\mathrm{f}_{U,\mathrm{add}} = (T_U - 1)$. Similarly, for a reconstruction from the low-frequency (average) subband only, the complexity for the production of an N-sample output signal is $\Upsilon_A(N) = \Lambda_D(N)$.

[1] Note that we shall use lifting factorizations without scaling factors in order to minimize the arithmetic complexity. In a coding system, the multiplications with these factors can be implicitly performed by embedding the scaling factors in the quantization tables applied to each subband at each resolution level.

5.3.1.2 Computational Complexity for the Conventional Multi-rate Approach for the Production of Subbands $\mathbf{w}_{\mathrm{ODWT}}^{k}(z)$

As explained in Subsection 5.1.2, the conventional multi-rate approach for the production of subbands $\mathbf{w}_{\mathrm{ODWT}}^{k}(z)$ consists of the inverse DWT followed by the LBS algorithm. For a decomposition in k levels, k single-level inverse transforms should be performed to reconstruct the input signal X. Then, by performing a number of forward transforms that retain the even or odd polyphase components of the non-decimated transform, all the ODWT subbands of level k are produced. Thus, the complexity of the construction in the full-overcomplete mode, where both the low and high-frequency ODWT subbands of level k are produced, is:

$$\begin{aligned}\mathrm{LBS}(\mathbf{w}_{\mathrm{ODWT}}^{k}) &= \left[\Upsilon_C(\tfrac{N}{2^{k-1}}) + \Upsilon_C(\tfrac{N}{2^{k-2}}) + \cdots + \Upsilon_C(N)\right] \\ &\quad + \left[(2-1)\Lambda_A(N) + \ldots + (2^{k-1}-1)\Lambda_A(\tfrac{N}{2^{k-2}}) + (2^{k}-1)\Lambda_C(\tfrac{N}{2^{k-1}})\right], \\ &= N\left[\mathrm{I}_{\mathrm{multiply}}\left(2-2^{1-k}\right) + \mathrm{f}_{H,\mathrm{multiply}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{multiply}} \\ &\quad + N\left[\mathrm{I}_{\mathrm{add}}\left(2-2^{1-k}\right) + \mathrm{f}_{H,\mathrm{add}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{add}}\end{aligned} \tag{5.47}$$

where $\mathrm{LBS}(\mathbf{s})$ denotes the computational complexity for the production of the vector of signals $\mathbf{s}$ using the conventional approach based on the IDWT and LBS.

In the case of the LL-LBS algorithm described in Subsection 5.1.2, the higher-resolution high-frequency subbands of the critically-sampled DWT are not available, i.e. $\mathbf{d}_0^{k-1,k-1}(z) = \mathbf{0}$. Hence, the complexity of this case is:

$$\begin{aligned}\mathrm{LLLBS}(\mathbf{w}_{\mathrm{ODWT}}^{k}) &= \left[\Upsilon_C(\tfrac{N}{2^{k-1}}) + \Upsilon_A(\tfrac{N}{2^{k-2}}) + \cdots + \Upsilon_A(N)\right] \\ &\quad + \left[(2-1)\Lambda_A(N) + \ldots + (2^{k-1}-1)\Lambda_A(\tfrac{N}{2^{k-2}}) + (2^{k}-1)\Lambda_C(\tfrac{N}{2^{k-1}})\right] \\ &= N\left[\mathrm{I}_{\mathrm{multiply}} + \mathrm{f}_{G,\mathrm{multiply}}\left(1-2^{1-k}\right) + \mathrm{f}_{H,\mathrm{multiply}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{multiply}} \\ &\quad + N\left[\mathrm{I}_{\mathrm{add}} + \mathrm{f}_{G,\mathrm{add}}\left(1-2^{1-k}\right) + \mathrm{f}_{H,\mathrm{add}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{add}}\end{aligned} \tag{5.48}$$

In the high-frequency overcomplete mode, where only the high-frequency subbands of level k are produced, the computational complexity can be calculated similarly as before; the results are:

$$\begin{aligned}\mathrm{LBS}(\mathbf{d}_{\mathrm{ODWT}}^{k}) &= N\left[\left(\mathrm{I}_{\mathrm{multiply}} + \mathrm{f}_{G,\mathrm{multiply}}\right)\left(1-2^{-k}\right) + \mathrm{f}_{H,\mathrm{multiply}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{multiply}} \\ &\quad + N\left[\left(\mathrm{I}_{\mathrm{add}} + \mathrm{f}_{G,\mathrm{add}}\right)\left(1-2^{-k}\right) + \mathrm{f}_{H,\mathrm{add}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{add}}\end{aligned}, \tag{5.49}$$

$$\begin{aligned}\mathrm{LLLBS}(\mathbf{d}_{\mathrm{ODWT}}^{k}) &= N\left[\mathrm{I}_{\mathrm{multiply}}2^{-k} + \mathrm{f}_{G,\mathrm{multiply}}\left(2-3\cdot 2^{-k}\right) + \mathrm{f}_{H,\mathrm{multiply}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{multiply}} \\ &\quad + N\left[\mathrm{I}_{\mathrm{add}}2^{-k} + \mathrm{f}_{G,\mathrm{add}}\left(2-3\cdot 2^{-k}\right) + \mathrm{f}_{H,\mathrm{add}}\left(k-2+2^{1-k}\right)\right]\mathrm{Cost}_{\mathrm{add}}\end{aligned} \tag{5.50}$$

5.3.1.3 The Proposed Method

As explained in Subsection 5.2.1.1, Figure V-4(a) demonstrates the calculation of the single-rate part of Proposition 3. For each processor element of Figure V-4(a), if the applied filters have T_{F_1}, T_{F_2} non-zero coefficients, $2(T_{F_1}+T_{F_2})$ multiplications and $4(T_{F_1}+T_{F_2})-4$ additions are performed within the PE for each set of 4 inputs; notice that this holds for $k>1$, as for $k=1$ there are no PEs. Using Figure V-4(a), the computational complexity of the single-rate part of the proposed

CODWT for the two different constructions, i.e. the FO and the HFO modes mentioned in Subsection 5.1.2, is given by:

$$\begin{aligned}\mathrm{PF}_{\mathrm{SR}}\left(\mathbf{w}_{\mathrm{ODWT}}^{k}\right) = \frac{N}{2^{k}}\Bigg[\left\lfloor\frac{T_{F_0^1}+1}{2}\right\rfloor + T_{F_1^1} + T_{F_2^1} + \left\lfloor\frac{T_{F_3^1}+1}{2}\right\rfloor + 2\sum_{l=2}^{k}\sum_{i=0}^{2^{l-2}-1}\left(T_{F_{4i,0}^{l-1}} + T_{F_{2^l-4-4i,0}^{l-1}}\right)\Bigg]\mathrm{Cost}_{\mathrm{multiply}} \\ + \frac{N}{2^{k}}\Bigg[\sum_{i=0}^{3}\left(T_{F_i^1}-1\right) + 2 + 4\sum_{l=2}^{k}\sum_{i=0}^{2^{l-2}-1}\left(T_{F_{4i,0}^{l-1}} + T_{F_{2^l-4-4i,0}^{l-1}} - 2\right) + 4\left(2^{k-1}-1\right)\Bigg]\mathrm{Cost}_{\mathrm{add}}\end{aligned}, \tag{5.51}$$

$$\begin{aligned}\mathrm{PF}_{\mathrm{SR}}\left(\mathbf{d}_{\mathrm{ODWT}}^{k}\right) = \frac{N}{2^{k}}\Bigg[T_{F_2^1} + \left\lfloor\frac{T_{F_3^1}+1}{2}\right\rfloor + \sum_{l=2}^{k}\sum_{i=0}^{2^{l-2}-1}\left(T_{F_{4i,0}^{l-1}} + T_{F_{2^l-4-4i,0}^{l-1}}\right)\Bigg]\mathrm{Cost}_{\mathrm{multiply}} \\ + \frac{N}{2^{k}}\Bigg[\sum_{i=2}^{3}\left(T_{F_i^1}-1\right) + 1 + 2\sum_{l=2}^{k}\sum_{i=0}^{2^{l-2}-1}\left(T_{F_{4i,0}^{l-1}} + T_{F_{2^l-4-4i,0}^{l-1}} - 2\right) + 2\left(2^{k-1}-1\right)\Bigg]\mathrm{Cost}_{\mathrm{add}}\end{aligned}, \tag{5.52}$$

where $\mathrm{PF}_{\mathrm{SR}}(\mathbf{s})$ is the computational complexity of the single-rate part of equation (5.17) for the production of the vector of signals $\mathbf{s}$ and $T_{F_i^l}$ is the number of non-zero coefficients of filter F_i^l. In equations (5.51), (5.52), the number of multiplications for the prediction filters of level one is reduced by taking advantage of the symmetry property of (5.23), which holds also for filter $F_3^1(z)$ since $F_0^1(z) = -F_3^1(z)$ from the definition of (5.4). Furthermore, if one restricts the complexity analysis to biorthogonal filter-banks, the symmetries of (5.24) and (5.25) can be used to further reduce the multiplications of (5.51), (5.52).

From the description of Subsection 5.2.1.2 and Figure V-5, the computational complexity of the multi-rate part of Proposition 3 for the general case of k decomposition levels can be deduced. Below, it is expressed for each case as a sum of the required computational complexity for each subband $D_0^l(z)$, $1 \le l < k$:

$$\begin{aligned}\mathrm{PF}_{\mathrm{MR}}\left(\mathbf{w}_{\mathrm{ODWT}}^{k}\right) = N\Bigg[\sum_{l=1}^{k-1}\left(2^{-l}\left(T_{F_1^1} + \mathrm{f}_{\mathrm{LS,multiply}}(l) + \mathrm{f}_{H,\mathrm{multiply}}\mathrm{f}_{\mathrm{an}}(l)\right)\right) + 2^{-k}\mathrm{I}_{\mathrm{multiply}}\mathrm{f}_{\mathrm{an}}(k)\Bigg]\mathrm{Cost}_{\mathrm{multiply}} \\ + N\Bigg[\sum_{l=1}^{k-1}\left(2^{-l}\left(T_{F_1^1} - 1 + \mathrm{f}_{\mathrm{LS,add}}(l) + \left(\mathrm{f}_{H,\mathrm{add}} + 1\right)\mathrm{f}_{\mathrm{an}}(l)\right)\right) + 2^{-k}\mathrm{I}_{\mathrm{add}}\mathrm{f}_{\mathrm{an}}(k)\Bigg]\mathrm{Cost}_{\mathrm{add}}\end{aligned}, \tag{5.53}$$

$$\begin{aligned}\mathrm{PF}_{\mathrm{MR}}\left(\mathbf{d}_{\mathrm{ODWT}}^{k}\right) = N\Bigg[\sum_{l=1}^{k-1}\left(2^{-l}\left(T_{F_1^1} + \mathrm{f}_{\mathrm{LS,multiply}}(l) + \mathrm{f}_{H,\mathrm{multiply}}\mathrm{f}_{\mathrm{an}}(l)\right)\right) + 2^{-k}\mathrm{f}_{G,\mathrm{multiply}}\mathrm{f}_{\mathrm{an}}(k)\Bigg]\mathrm{Cost}_{\mathrm{multiply}} \\ + N\Bigg[\sum_{l=1}^{k-1}\left(2^{-l}\left(T_{F_1^1} - 1 + \mathrm{f}_{\mathrm{LS,add}}(l) + \left(\mathrm{f}_{H,\mathrm{add}} + 1\right)\mathrm{f}_{\mathrm{an}}(l)\right)\right) + 2^{-k}\mathrm{f}_{G,\mathrm{add}}\mathrm{f}_{\mathrm{an}}(k)\Bigg]\mathrm{Cost}_{\mathrm{add}}\end{aligned}, \tag{5.54}$$

where, for each subband $D_0^l(z)$, the multiplication and addition-related complexity of the used lattice structures is $\mathrm{f}_{\mathrm{LS,multiply}}(l) = \sum_{i=1}^{l-1}\sum_{j=0}^{2^{i-1}-1} T_{F_{4j,0}^i}$, $\mathrm{f}_{\mathrm{LS,add}}(l) = 2\sum_{i=1}^{l-1}\sum_{j=0}^{2^{i-1}-1}\left(T_{F_{4j,0}^i} - 1\right)$ for $l > 1$, with

$\mathrm{f}_{\mathrm{LS,multiply}}(1)=0$, $\mathrm{f}_{\mathrm{LS,add}}(1)=0$. Also, $\mathrm{f}_{\mathrm{an}}(l)=2^{l}-2$ is the number of the used low-pass analysis filters.

Note that, as shown in (5.17), the multi-rate calculation occurs for $k>1$. For this case, the total complexity of the final result of (5.17) is given by:

$$\mathrm{PF}(\mathbf{w}_{\mathrm{ODWT}}^{k})=\mathrm{PF}_{\mathrm{SR}}(\mathbf{w}_{\mathrm{ODWT}}^{k})+\mathrm{PF}_{\mathrm{MR}}(\mathbf{w}_{\mathrm{ODWT}}^{k})+N(2-2^{1-k})\mathrm{Cost}_{\mathrm{add}}, \tag{5.55}$$

and,

$$\mathrm{PF}(\mathrm{d}_{\mathrm{ODWT}}^{k})=\mathrm{PF}_{\mathrm{SR}}(\mathrm{d}_{\mathrm{ODWT}}^{k})+\mathrm{PF}_{\mathrm{MR}}(\mathrm{d}_{\mathrm{ODWT}}^{k})+N(1-2^{-k})\mathrm{Cost}_{\mathrm{add}} \tag{5.56}$$

for the FO and HFO construction, respectively. The last terms correspond to the complexity of the additions between the multi-rate and single-rate part of the transform. As in the case of the single-rate calculation, if one restricts the complexity analysis to biorthogonal filter-banks, the relation $F_1^1(z^{-1})=F_1^1(z)$ that holds for this category can be used to further reduce the multiplication complexity of (5.53), (5.54).

For a numerical comparison between the two approaches, by using equation (5.55) for the proposed CODWT and equation (5.48) for the conventional IDWT+LBS approach, we identified that both methods have comparable arithmetic complexity under a convolution-based implementation. This is shown in Table V-IV, where numerical examples of the multiplication and addition requirements are presented for the levels $k=1,2,3$ using the proposed and the conventional approaches. Both cases have been normalized by the number of input samples (pixels) N. The use of lifting reduces the computational requirements for both approaches; however we generally found that lifting factorizations tend to benefit less the proposed approach since they are only used in the analysis filters of the multi-rate part of the transform.

9/7 filter-bank [24] [16]			$\mathrm{I}_{\mathrm{multiply}}=9, \mathrm{I}_{\mathrm{add}}=14$	
Construction of decomposition level k:	**Proposed (Mpp)**	**IDWT+ LBS (Mpp)**	**Proposed (App)**	**IDWT+ LBS (App)**
1	8.5	9	15	14
2	15.25	16	27	25
3	23.625	22	40.5	34.5
7/5 filter-bank [25, 26]			$\mathrm{I}_{\mathrm{multiply}}=7, \mathrm{I}_{\mathrm{add}}=10$	
1	6.5	7	11	10
2	11.75	12.5	20	18
3	18.125	17.25	29.5	25
4/4 filter-bank [23]			$\mathrm{I}_{\mathrm{multiply}}=4, \mathrm{I}_{\mathrm{add}}=6$	
1	6	4	7	6
2	11.5	8	14.25	10.5
3	18.75	12	21.25	14.25

Mpp: Multiplications per pixel (average), App: Additions per pixel (average)

Table V-IV. Multiplications and additions for the CODWT of various levels under a convolution-based implementation. The total number of operations is normalized with the number of input pixels.

On the other hand, as shown in Table V-V, the proposed approach exhibits a *practical* reduction in computation times in comparison to the conventional method for "ANSI-C" implementations running on a personal computer with an Intel Pentium IV processor. Both convolution-based and lifting-based implementations have been tested. The experiments were carried out for an SD-resolution video sequence (720x480 pixels, 3 color channels, 100 frames) with the 9/7 filter-bank. The row-column implementation was used for both methods. Table V-V demonstrates that the proposed approach runs, on average, 35.83% faster in comparison to the conventional approach when both employ a lifting-based implementation. Profiling of the execution revealed that, although the proposed method performs (on average) more instruction-related operations in the case of lifting-based implementations, a significantly-better utilization of the cache memory is achieved in comparison to the IDWT+LBS approach. This is attributed to the fact that, by using the proposed calculation scheme, the proposed CODWT utilizes the same input to produce a number of intermediate results; for example, Figure V-4(a) demonstrates that the single-rate part of all the subbands of level three is produced by using $A_0^3(z), D_0^3(z), A_1^3(z), D_1^3(z)$. As a result, the proposed CODWT achieves a significantly-higher locality in the processing. This leads to the speedup in computation times reported in Table V-V.

Construction of decomposition level k:	**Convolution based** Speedup of proposed vs. LBS	**Lifting based** Speedup of proposed vs. LBS
1	77.12 %	69.02 %
2	38.54 %	33.31 %
3	12.43 %	5.15 %
Average:	42.70 %	35.83 %

Table V-V. Percentage of speedup in execution time of the proposed method versus the conventional approach (9/7 filter-bank) in a Pentium IV processor.

5.3.2 The Computational Complexity of each Method under the Application Framework

For the CODWT under the application framework of Subsection 5.1.2, a system that supports a total of k decomposition (resolution levels) performs the CODWT in full-overcomplete mode for level k and in high-frequency overcomplete mode for levels $k-1, k-2, .., l$, with l indicating the output resolution of a certain (de)coder [5, 20] [2] [3]. The *total* CODWT complexity for a (de)coder that ceases the processing at level l $(1 \le l \le k)$ is denoted as $\mathrm{LLLBS}^k(l)$, $\mathrm{PF}_{\mathrm{SR}}^k(l)$ for the LL-LBS and the prediction-filters method respectively. Based on (5.48), (5.50)–(5.52), we have:

$$\mathrm{LLLBS}^k(l) = \begin{cases} \mathrm{LLLBS}(\mathbf{w}_{\mathrm{ODWT}}^k), \text{ if } l = k \\ \mathrm{LLLBS}(\mathbf{w}_{\mathrm{ODWT}}^k) + \sum_{j=l}^{k-1} \mathrm{LLLBS}(\mathbf{d}_{\mathrm{ODWT}}^j), \text{ if } 1 \le l < k \end{cases} \tag{5.57}$$

$$\mathrm{PF}_{\mathrm{SR}}^k(l) = \begin{cases} \mathrm{PF}_{\mathrm{SR}}(\mathbf{w}_{\mathrm{ODWT}}^k), \text{ if } l = k \\ \mathrm{PF}_{\mathrm{SR}}(\mathbf{w}_{\mathrm{ODWT}}^k) + \sum_{j=l}^{k-1} \mathrm{PF}_{\mathrm{SR}}(\mathbf{d}_{\mathrm{ODWT}}^j), \text{ if } 1 \le l < k \end{cases} \tag{5.58}$$

It is important to notice that the results of this section are applicable also for the two-dimensional case of a $C \times R$ input image since both techniques can be applied in a separable manner along the rows and columns of the input. Specifically, the two-dimensional implementation of the proposed CODWT for any level k consists of:

- applying the prediction-filters row-wise to the two-dimensional DWT subbands using the calculation scheme of Figure V-4(a) and Figure V-5, thereby producing the ODWT subbands with phase $(0, i)$, $1 \leq i < 2^k$ (two-dimensional phase [3] [2]);

- the column-wise filtering of the resulting subbands with the same one-dimensional scheme.

Hence, to produce the $2^k \times 2^k$ ODWT subbands of level k, the complexity associated with the row-wise filtering is $(R/2^k) \times \mathrm{PF}(k)$. In addition, the complexity of the column-wise filtering is $C \times \mathrm{PF}(k)$. In the same manner, the separable application of the one-dimensional LBS approach indicates that the complexity is $(R/2^k + C) \times \mathrm{LBS}(k)$. As a result, the complexity reduction offered by the proposed approach is $1\text{-}\frac{\mathrm{PF}(k)}{\mathrm{LBS}(k)}$ in both one-dimensional and two-dimensional cases.

In resolution-scalable coding, the comparison between the lifting-based implementations of the proposed CODWT and LL-LBS approaches is given in Table V-VI. The case of a coding system that supports a maximum of $k = 4$ decomposition (resolution) levels and decodes to any output resolution-level $l, 1 \leq l \leq k$ is assumed. In this scenario, as explained in Subsection 5.1.2, decoding a resolution level l requires that the CODWT is performed for resolution-levels $k, k-1, \ldots, l$. These results show that, in resolution-scalable coding, the proposed CODWT achieves notable complexity reductions in comparison to LL-LBS. Furthermore, we find that significant reductions in the computation time are experimentally observed. This is shown in Table V-VII, where we assume a resolution-scalable scenario with a HDTV video sequence processed in four (dyadically-reduced) resolutions, ranging from 1920×1088 pixels $(l = 1)$ to 240×136 pixels $(l = 4)$. The proposed CODWT exhibits an average reduction of 78.45% in computation time for lifting implementations.

Table V-VI presents a numerical comparison between the two approaches using equations (5.57) and (5.58) for the LL-LBS and the proposed approach, respectively. Again, the results have been normalized by the number of input samples (pixels) N. The case of a coding system that supports a maximum of $k = 4$ decomposition (resolution) levels and (de)codes to any level l, $1 \leq l \leq 4$ is used. In this scenario, as explained in Subsection 5.1.2, decoding a resolution level l requires that the CODWT is performed for resolution-levels $k, k-1, \ldots, l$. One orthogonal and two biorthogonal filter-banks were chosen in order to cover all types of filter-banks that are used in image and video coding literature. The achieved reductions of the proposed approach are reported in comparison to the LL-LBS using a lifting-based implementation. In every case the proposed approach reduces significantly the required computations. Furthermore, we find that, in practice, significant reductions in the computation time are experimentally observed. This is shown in Table V-VII, where we assume a resolution-scalable scenario with a HDTV video sequence processed in four (dyadically-reduced) resolutions, ranging from 1920×1088 pixels $(l = 1)$ to 240×136 pixels $(l = 4)$. The proposed CODWT exhibits an average reduction of 78.45% in computation time for lifting implementations.

CODWT stops at level l :	Multiplications		Reduction (%)	Additions		Reduction (%)
	Proposed (Mpp)	LL-LBS (Mpp)		Proposed (App)	LL-LBS (App)	
9/7 filter-bank, ($I_{mult} = 4,\ I_{add} = 8$) [24] [16]						
1	23.81	35.75	**33.39**	52.75	71.5	**26.22**
2	19.81	32.25	**38.57**	38.25	64.5	**40.70**
3	15.81	25.5	**37.99**	27.5	51	**46.08**
4	11.06	15.125	**26.86**	17.625	30.25	**41.74**
7/5 filter-bank, ($I_{mult} = 3,\ I_{add} = 6$) [25, 26]						
1	17.8125	25.75	**30.83**	37.25	51.50	**27.67**
2	14.8125	23.25	**36.29**	26.75	46.5	**42.47**
3	11.8125	18.5	**36.15**	19	37	**48.65**
4	8.3125	11.125	**25.28**	12.125	22.25	**45.51**
4/4 filter-bank, ($I_{mult} = 2,\ I_{add} = 5$) [23]						
1	13.25	30	**55.83**	20.75	33.75	**38.52**
2	10.25	27	**62.04**	14.25	29.75	**52.10**
3	7.75	21	**63.10**	9.5	23.25	**59.14**
4	5.25	12	**56.25**	5.625	14	**59.82**

Mpp: Multiplications per pixel (average), App: Additions per pixel (average)

Table V-VI. Multiplication and addition budget for the CODWT in a resolution-scalable codec with four decomposition levels. Three representative filter-banks (of varying complexity) used in lossy image and video coding are presented.

CODWT stops at level l :	Speedup of proposed vs. LL-LBS using convolution	Speedup of proposed vs. LL-LBS using lifting
1	82.33 %	77.78 %
2	82.58 %	78.20 %
3	84.95 %	81.08 %
4	82.00 %	76.75 %
Average:	82.96 %	78.45 %

Table V-VII. Example of speedup in computation times for the CODWT under resolution-scalable construction; 100 frames of a HDTV color video sequence (1920×1088 pixels) were used with the 9/7 filter-bank.

An interesting observation stemming from Table V-VI is that the computational complexity of both techniques scales-up with the number of decoded resolution levels, a result that is highly useful for complexity-scalable video coding.

The significant complexity reductions of Table V-VI and Table V-VII that are achieved with the proposed approach in resolution scalable scenarios can be attributed to the following two aspects:

- A single-rate calculation is used. Hence, the series of inverse and forward wavelet transforms required by the conventional multi-rate approach is avoided.
- The scheme of Subsection 5.2.1.1 reduces the complexity based on the filter symmetries of Subsection 5.1.3.2.

As a result, the proposed approach becomes *increasingly* more efficient in comparison to the conventional approach as the resolution-level l increases. Interestingly, this tendency is reversed at the coarsest resolution level where the computational-reduction percentage offered by the proposed approach is usually decreased (Table V-VI and experimental results of Table V-VII). This phenomenon occurs since the ODWT of the coarsest resolution is constructed in FO-mode (i.e. both the low and high-frequency ODWT subbands are created), while the ODWT of the lower-resolution levels are constructed in HFO-mode (i.e. only the high-frequency ODWT subbands are created).

5.3.3 Calculation Delay of each Method under the Application Framework

Assuming the classical point-symmetric or periodic extension for the signal edges, consider that the two methods are implemented in a system where one application of a filter (or filter-bank) on a set of input samples requires a_{LBS} processing cycles for the LL-LBS method and a_{PF} processing cycles for the prediction-filters method. To diminish the side effects of scheduling algorithms for the multiple filtering operations, we assume the case of high parallelism, where one filter per required convolution is present (similar to the system of [27]). In this way, every filtering application initiates as soon as sufficient input is present [27]. Furthermore, to facilitate the description, the delay resulting from the storage or retrieval of intermediate results is not taken into account.

Starting with the LL-LBS method, k single-level inverse transforms and $\sum_{l=1}^{k}(2^l-1)$ single-level forward transforms are performed for the production of the subbands of the ODWT of level k; see the example in Figure V-1(b) for $k=3$. After an initiation latency (denoted as $L_{\mathrm{init,LL\text{-}LBS}}(k)$), the first coefficients of the reconstructed input signal $\overline{X}$ are produced. Then, all filter-kernels of every level work in parallel to perform the inverse and forward transforms. This is equivalent to a cascade connection of the inverse and forward recursive pyramid algorithm of Vishwanath [27].

The total time required for the completion of the calculation of all subbands is determined by the delay for the production of the signal with the maximum length, since during this process the highest number of *consecutive* filter applications occurs [27]. The signal with the maximum length produced by the LL-LBS is the reconstructed input-signal $\overline{X}$. The production of $\overline{X}$ requires $\frac{N}{2}$ applications of filter $\tilde{H}$ (synthesis low-pass filter). Additionally, after the production of all samples of $\overline{X}$, the

completion of the forward transforms at each level requires an additional number of filtering operations that will cause an additional number of processing cycles, denoted as $L_{\text{comp,LL-LBS}}(k)$. Hence, the total delay of the LL-LBS system for the production of all the subbands of decomposition level k for an N-point input signal is:

$$L_{\text{LL-LBS}}(k,N) = L_{\text{init,LL-LBS}}(k) + \frac{N}{2} a_{\text{LBS}} + L_{\text{comp,LL-LBS}}(k) \tag{5.59}$$

For the prediction-filters method, the application of filters $F_0^1,\ldots,F_3^1$ to A_0^k, D_0^k can be initiated in parallel for the calculation of subbands A_1^k, D_1^k. After this initiation, which in this case requires $L_{\text{init,PF}}(k)$ processing cycles, the PEs that produce the rest of the subbands of level k (if $k>1$) can also be applied in parallel by reusing the coefficients of subbands A_1^k, D_1^k, as seen in Figure V-4(a). As a result, the required delay for the completion of the process is:

$$L_{\text{PF}}(k,N) = L_{\text{init,PF}}(k) + \frac{N}{2^k} a_{\text{PF}} \tag{5.60}$$

Equations (5.59), (5.60) show that, in systems that can achieve a high-degree of hardware parallelism, the delay of the proposed CODWT for resolution level k is expected to be proportional to $\frac{N}{2^k} a_{\text{PF}}$, while the LL-LBS approach achieves a delay proportional to $\frac{N}{2} a_{\text{LBS}}$. Examples of the delay ratios between the two approaches under this high-parallelism scenario are given in Table V-VIII. Since it is difficult to quantify the actual ratio between the factors a_{PF} and a_{LBS} without measurements from a realization of the two approaches in a custom-hardware design, we resort to report the delay-reduction gains offered by the proposed approach under three assumptions: $a_{\text{LBS}} = u_c \cdot a_{\text{PF}}$, with $u_c = \{0.5,\ 1,\ 2\}$, corresponding to a "pessimistic", "average" and "optimistic" case for the relative efficiency in hardware implementation of the proposed method versus the conventional approach.

For the two-dimensional processing of a $C \times R$ image, as explained before, both methods can be implemented via two identical one-dimensional systems used row-wise and column-wise. However, instead of processing all the rows and consequently processing the columns, the filtering in each direction can be interleaved and the column processing begins after an initiation latency so that enough coefficients exist column-wise for the mirroring and for the initiation of the filter-applications required for every method. Hence, the comparison of both methods in terms of delay for the two-dimensional CODWT follows the result of the one-dimensional case.

Level k	$\frac{L_{\text{LL-LBS}}(k,512)}{L_{\text{PF}}(k,512)}$ **with** $u_c = 0.5$	$\frac{L_{\text{LL-LBS}}(k,512)}{L_{\text{PF}}(k,512)}$ **with** $u_c = 1$	$\frac{L_{\text{LL-LBS}}(k,512)}{L_{\text{PF}}(k,512)}$ **with** $u_c = 2$
1	0.51	1.01	2.03
2	0.96	1.92	3.84
3	1.78	3.56	7.11
4	2.99	5.98	11.96

Table V-VIII. Examples of the ODWT delay-reduction gains offered by the proposed approach (FO-mode, 9/7 filter-bank) for various decomposition levels of a 512-sample signal. Three indicative cases for the ratio $u_c = a_{\text{LBS}}/a_{\text{PF}}$ are used.

5.4 Conclusions

In this chapter, a new framework for the construction of the overcomplete DWT starting from the subbands of the critically-sampled decomposition was presented. The proposed framework has inherent advantages in comparison to the conventional approach since it consists of a direct transform from the complete to the overcomplete DWT, using the minimum number of downsampling operations and no upsampling. For resolution-scalable video coding applications that utilize a level-by-level construction of the ODWT, it is demonstrated that the proposed CODWT has significant implementation advantages over the conventional approach because it offers (a) significant computation savings, and (b) a single-rate calculation that can provide a scalable reduction in the transform-production delay. These features lead to inherent computational scalability in comparison to the conventional approach.

References

[1] H.-W. Park and H.-S. Kim, "Motion Estimation Using Low-Band-Shift Method for Wavelet-Based Moving-Picture Coding," *IEEE Transactions on Image Processing*, vol. 9, pp. 577-587, 2000.

[2] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. v. d. Schaar, J. Cornelis, and P. Schelkens, "In-band Motion Compensated Temporal Filtering," *Signal Processing: Image Communication*, vol. 19, pp. 653-673, 2004.

[3] X. Li, "Scalable Video Compression via Overcomplete Motion Compensated Wavelet Coding," *Signal Processing: Image Communication*, vol. 19, pp. 637-651, 2004.

[4] S. Cui, Y. Wang, and J. E. Fowler, "Mesh-based Motion Estimation and Compensation in the Wavelet Domain using a Redundant Transform," presented at IEEE International Conference on Image Processing (ICIP), Rochester, NY, USA, 2002.

[5] C. Mayer, "Motion compensated in-band prediction for wavelet-based spatially scalable video coding," presented at Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Hong-Kong, CN, 2003.

[6] J. Skowronski, "Pel-recursive Motion Estimation and Compensation in Subbands," *Signal Processing: Image Communication*, vol. 14, pp. 389-396, 1999.

[7] S. G. Mallat, *A wavelet tour of signal processing*. San Diego: Academic Press, 1998.

[8] H. Sari-Sarraf and D. Brzakovic, "A shift-invariant discrete wavelet transform," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2621-2626, 1997.

[9] M. J. Shensa, "The discrete wavelet transform: Wedding the A Trous and Mallat Algorithms," *IEEE Transactions on Signal Processing*, vol. 40, pp. 2464-2482, 1992.

[10] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens, and J. Cornelis, "A new method for complete-to-overcomplete discrete wavelet transforms," presented at 14th International Conference on Digital Signal Processing, DSP2002, Santorini, Greece, 2002.

[11] Y. Andreopoulos, A. Munteanu, G. V. d. Auwera, J. Cornelis, and P. Schelkens, "Single-rate calculation of overcomplete discrete wavelet transforms for scalable coding applications," *Signal Processing*, to appear.

[12] Y. Andreopoulos, A. Munteanu, G. V. d. Auwera, P. Schelkens, and J. Cornelis, "Complete-to-Overcomplete Discrete Wavelet Transforms: Theory and Application," *IEEE Transactions on Signal Processing*, to appear.

[13] X. Li, "New results on phase shifting in the wavelet space," *IEEE Signal Processing Letters*, vol. 10, pp. 193-195, 2003.

[14] G. Van der Auwera, A. Munteanu, P. Schelkens, and J. Cornelis, "Bottom-up Motion Compensated Prediction in the Wavelet Domain for Spatially Scalable Video Coding," *IEE Electronics Letters*, vol. 38, pp. 1251-1253, 2002.

[15] G. Strang and T. Nguyen, *Wavelets and filter banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[16] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure Applied Mathematics*, vol. 45, pp. 485-560, 1992.

[17] X. Li, L. Kerofsky, and S. Lei, "All-phase motion compensated prediction in the wavelet domain for high performance video coding," presented at IEEE International Conference on Image Processing (ICIP), Thessaloniki,Greece, 2001.

[18] R. R. Coifman and D. L. Donoho, "Translation invariant de-noising," in *Wavelets and Statistics: Lecture Notes in Statistics 103*, A. Antoniadis and G. Oppenheim, Eds. New York: Springer-Verlag, pp. 125-150.

[19] H. S. K. Kim and H. W. Park, "Wavelet-based moving-picture coding using shift-invariant motion estimation in wavelet domain," *Signal Processing: Image Communication*, vol. 16, pp. 669-679, 2001.

[20] C. Mayer and S. Albert, "Scalable video coding with in-band prediction," presented at Proceedings of SPIE Electronic Imaging - Image and Video Communications and Signal Processing, Santa Clara, US, 2003.

[21] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens, and J. Cornelis, "Scalable Wavelet Video-Coding with In-Band Prediction - Implementation and Experimental Results," presented at IEEE International Conference on Image Processing (ICIP), Rochester, New York, USA, 2002.

[22] H. M. Radha, M. v. d. Schaar, and Y. Chen, "The MPEG-4 Fine-grained Scalable Video Coding for Multimedia Streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, pp. 53-68, 2001.

[23] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, pp. 247-269, 1998.

[24] ISO/IEC, "JPEG 2000 image coding system," ISO/IEC JTC1/SC29/WG1, IS 15444-1, December 2000 2000.

[25] M. Boliek, "JPEG-2000 Part II Final Committee Draft," ISO/IEC JTC1/SC29/WG1 (JPEG), FCD 15444-2, Dec. 2000.

[26] C. Brislawn, "Interim report on (Part 2) core experiment CodEff07. 7-tap/5-tap Filter Bank option," ISO/IEC JTC1/SC20/WG1 (JPEG), N1761, July 2000.

[27] M. Vishwanath, "The Recursive Pyramid Algorithm for the Discrete Wavelet Transform," *IEEE Transactions on Signal Processing*, vol. 42, pp. 673-676, 1994.

Chapter VI

COMPLEXITY MODELING ASPECTS FOR WAVELET-BASED SCALABLE VIDEO CODING

COMPLEXITY is often the deciding factor that determines the extent to which a certain algorithm can be deployed in a real-world multimedia communication infrastructure. For example, in video compression systems, MPEG standards always define a set of profiles and levels; one of the purposes of this separation is to divide the algorithmic parameter space into segments that will produce coding architectures of varying complexity. Thus, in one of the recent MPEG standards, namely the Advanced Video Coder (or ITU-T H.264) [1], several motion-estimation tools are standardized, whose usage is recommended only under certain platform implementation capabilities [2]. Similarly, in the area of scalable video coding (which is the focus of this dissertation), one of the purposes of extracting substreams that represent lower frame-rates and resolutions is to accommodate the complexity profile of a certain decoding device in the best possible manner. Finally, it is worth mentioning that the maximum-allowable latency in the transmission of compressed video content is always determined as a function of the transmission channel parameters and the complexity of the algorithm.

As a result, in this chapter we are concerned with the important topic of establishing complexity models for scalable video coding. We note that several related research results on the complexity of practical video coders are already published in the literature (e.g. see [2] [3] [4] [5]). Since complexity modeling for video coders usually involves a large number of parameters due to the interplay of several algorithmic-related factors, we are limiting the analysis of this chapter according to the following criteria:

- we are concerned primarily with wavelet-based scalable coding systems;

- the target implementation platforms are always considered to be programmable processors rather than custom hardware designs;

- our analysis is always theoretical and remains platform independent, i.e. specific implementation capabilities that can be used in a certain device to speedup the algorithm realization are not taken into account.

The first restriction limits the scope of the presented material into a specific type of scalable video coding systems, which are however showing significant promise for future video communication systems. On the other hand, the second and third restrictions aim at providing results that can be generally applicable to a wide range of systems, by selecting the model parameters to suit the implementation platform. In addition, our choice of programmable processors targets mainly special purpose Digital Signal Processors (DSPs) currently forming attractive solutions for cost-effective designs with a rapid development-cycle. It appears that the flexibility in fine tuning algorithmic parameters through software, combined with the ease in upgrading the implementation to newer (and faster) processor designs has already shifted a significant share in the multimedia implementation market to DSP-based systems [6].

The following section defines complexity in a pragmatic manner that fits the framework of this dissertation and furthermore introduces the particular modeling topics that will be discussed in the remainder of this chapter.

6.1 Introduction to Complexity Modeling – Novel Aspects treated in this Dissertation

In digital signal compression, one can define complexity as the number of operations performed by a generic machine that realizes the coding algorithm. The most generic definition of such a device is a Universal Turing Machine (UTM). It was recently shown that by defining complexity in terms of the shortest program of a UTM (Kolmogorov complexity), rate-distortion theory can be transformed into complexity-distortion theory [7]; moreover, the two theories appear to produce asymptotically the same results [7].

Although this definition can be useful for studying complexity under a pure theoretical framework, under the pragmatic constraints set in the previous subsection, complexity can be defined as the metric that determines the total number of arithmetic and memory operations for the realization of a coding algorithm in a generic computer architecture, such as a basic pipelined RISC machine [8]. In this way, similar to several contributions found in the literature [9] [10] [11], one can establish the total arithmetic complexity by profiling the software that realizes the algorithm of interest for the total number of additions and subtractions, multiplications and, potentially, divisions. In addition, the memory complexity can be assessed as the total number of memory accesses performed under the execution of the algorithm in a generic RISC machine. Since today's systems typically use a layered memory hierarchy that separates local (on-chip) from non-local (off-chip) memory, it is important that some generic assumptions for this hierarchy are taken into account when establishing the memory complexity. The scope of these assumptions is always selected to reflect the large majority of today's

practical architectures. Although it may not be possible to directly predict the practical execution time of an algorithm based on these metrics, such an analysis allows the relative comparisons of two algorithms in terms of complexity; additionally, under certain conditions, it is possible to map these generic complexity metrics into real complexity metrics for a given platform, and obtain a reliable metric for the execution time of a given video coder [9] [11] [12].

The two following sections focus on arithmetic and memory complexity as defined previously, and propose models that analyze video coding realizations with respect to their expected performance. In particular, Section 6.2 analyzes the memory complexity of the discrete wavelet transform module found in the scalable coding algorithms of this dissertation. As mentioned before, generic assumptions are made for the memory hierarchy of the target architectures in order for the results to match reality, and a widely-used memory structure that separates the memory layers into cache levels is used. A cache level is a small intermediate high-speed memory situated in between the functional units of the processor and main memory. Since memory-related complexity becomes a bottleneck in a practical system realization only under certain limits, constraints are set on the cache memory sizes in the analysis of Section 6.2; these constraints reflect the vast majority of implementation architectures found in the literature.

Although Section 6.2 focuses on a specific module, the following section (6.3) presents a holistic approach for determining on-the-fly complexity metrics for entire video coding systems, based on generic complexity metrics that are established in a methodological manner for any chosen software realization of a motion-compensated video coder. In terms of practical applications, the focus is given on the decoding complexity and a recently proposed streaming architecture is used to communicate hint information for the complexity metrics to each decoder receiving the compressed bitstream. For each case, experimental results are presented in order to validate the proposed models and the conclusions drawn from the presented results are outlined.

6.2 High-level Cache Modeling for Two-Dimensional Discrete Wavelet Transform Implementations

JPEG-2000 [13] and MPEG-4 Visual Texture Coding [14] are the two new standards that base part of their compression efficiency on discrete wavelet transforms for the coding of images and textures. The standardization effort of JPEG-2000 showed that the largest part of the complexity of a wavelet-based image coding system is associated to the memory organization for the production of the transform coefficients [15]. In several cases, a solution that involves tiling of the input signal is chosen in order to reduce the complexity when dealing with large amounts of input data. Thus, the term "image" in this section refers to the currently-processed tile, with the tile-size ranging typically from 128×128 to 1024×1024 pixels.

The features that are mainly investigated by researchers are the different image traversal algorithms for the band-bursty production of the wavelet coefficients [16] [17] [18] [19] [20, 21] [22] and the efficient coupling of such transform-production methods (producers of information) with coding algorithms (consumers of information) [17] [21] [15]. The transform-production methods that are the most

appropriate for coding applications can be classified in two major categories, based on the coefficient-production schedule.

The first category is based on an image traversal that leads to a strictly breadth-first production of wavelet coefficients (SBF methods) [23]. This is usually referred as the row-column wavelet transform (RCWT) [24] since, for every wavelet-decomposition level, the input data are filtered-and-downsampled horizontally and vertically in a sequential manner. In this way, the complete set of coefficients of the current level is produced before the initiation of the calculations for the next level; hence, this category has a strictly breadth-first production schedule. The second category consists of image traversals that lead to roughly depth-first production of wavelet coefficients (RDF methods) [11]. The two main techniques of this category are the local (or block-based) wavelet transform (LWT) [18, 20] and the line-based wavelet transform (LBWT) [16]. They use a block or a line-based traversal respectively, to input the image and produce block or line parts of the transform subbands of all levels. With minor adaptations, both methods can produce the transform subbands of all levels in a streaming manner, thus a dyadically-decreased number of subband lines is produced for all decomposition levels.

A third category that is based on the traversal that leads to a strictly depth-first production of wavelet coefficients [19] is not examined here because, as demonstrated by [22], it demands a vastly parallel architecture that is usually not achievable in software implementations in programmable processors.

In the implementation arena, rapid advances in the area of (embedded) instruction-set processors have turned these systems into attractive solutions for the realization of real-time multimedia applications. This is explainable by the flexibility offered by such architectures, which allows the implementation of several coding algorithms on the same hardware, and also by time-to-market reasons. Typically, the best performance for data-dominated applications such as the DWT, is expected from the method that most efficiently reuses the data in the processor caches and maximally reduces the off-chip memory accesses to avoid additional delays and energy dissipation [25]. As a result, the data-related cache misses determine the efficiency of each method with respect to throughput and power.

Following the above indications, this section analyzes the data-cache performance of various DWT-production approaches on instruction-set platforms and not the arithmetic or instruction-related complexity of these applications. Although the typical target architecture is generic, it is based on popular (embedded) instruction-set video or multimedia processors (e.g. Philips TriMedia, TMS320C6x but also Pentium MMX), as these are state-of-the-art in real-time multi-dimensional signal-processing applications [6]. The data and instruction-memory multilevel hierarchies consist of a number of caches. We denote by D-Cache and I-Cache, respectively, the data and instruction cache memory or level (hierarchy) one. In addition, we denote by L2-Cache the cache memory of level two. This cache can be a separable configuration of data and instruction cache memories, or a joint configuration of both. The data transfer and storage organization is left to hardware. Thus, the instructions or data enter the processor core after passing through the cache-hierarchy [26]. The typical path followed is: Main Memory→L2-Cache→I-Cache or D-Cache→Processor. When the instruction or data do not exist at the cache of a certain level, a miss occurs, in which case the processor waits until a block of instructions or data is fetched from the upper-level cache (or the main

memory), causing a delay to the execution not related to the computational complexity of the executed program. Following the classical 3-C model of the cache misses, the latter can be separated into capacity, compulsory and conflict misses [26].

This section first presents single-processor software designs for all the transform-production methods (Subsection 6.2.1). Based on them, an analytical model is proposed in Subsection 6.2.2 that allows for the prediction of the expected number of data-cache misses in a generic memory hierarchy. The validity of the proposed equations is bounded by a set of constraints for the cache characteristics. However, the linking between the transform production and the coding consumption is not studied here, since this would specialize the presented results for a specific application. In order to verify the theoretical framework and compare the proposed software designs, results are presented in Subsection 6.2.3 from simulations and from two real platforms. It must be noted however that the theoretical analysis does not aim to precisely predict the number of misses in a specified cache architecture, because various system-dependent operations such as data-input from storage or image-retrieval media or memory allocation operations are expected to produce fluctuations in the final results. Nevertheless, using the theoretical results of this section, one can approximately calculate the relative performance of the presented methods with respect to the cache utilization and, more importantly, analytically determine the advantages and disadvantages of every approach.

It is interesting to notice that general work on matrix computations refers explicitly to the effects of different traversal schedules on the input of two-dimensional matrices in a cache memory (subsection 1.4.7 of [27]). Moreover, experimental work reporting on the efficiency of a JPEG-2000 implementation for the DWT in a cache-based memory hierarchy can be found in [28]. An analysis for block-based DWT implementations in programmable architectures has been independently reported in [29], where the authors identify the input block-size that leads to optimum utilization of the system cache. The cache behaviour of different schedules under a generic memory hierarchy was also studied in [15]. Moreover, the work presented in this section was published in [23] [30]; our presentation in the following parts of this section is based on [30].

6.2.1 The different Wavelet Transform-production Approaches

This section analyzes the different production schedules that will be used for the memory-related comparisons. The analysis is structured in two sections, depending on the coefficient-production schedule. A pictorial description for each proposed design is presented, which facilitates the analysis of the cache-miss penalties. Since these penalties are caused by the input of data in the processor caches, the description concentrates on the copy activities between the different components and not on the detailed description of the filtering processes, because the latter actions are localized between the internal (on-chip) data-cache and the processor core and it is expected that they do not cause important fluctuations in the data-related miss penalties.

6.2.1.1 The design of strictly breadth-first methods - The RCWT approach

These methods are based on the batch processing of all rows and columns of the input of every decomposition level. Since the filtering is not interleaved for each row or column, a lifting-scheme

implementation is assumed [31], so that the minimum number of arithmetic operations is required for the transform production. Hence, a number of passes (S) are performed through every row or column of the input array and for every pass one predict-and-update step is performed in the input data [31]. For example, the 5/3 and 9/7 filter-banks are used in [13], which require one and two predict-and-update steps respectively through the input row or column. Hence, $S = 1$ for the 5/3 and $S = 2$ for the 9/7 filter-bank. All steps that are simple scaling operations can be applied during the final storage of the results and are not counted as additional passes. It must be noted that we always assume that the dyadic (Mallat) decomposition is applied [24] to the input image; this means that for each row or column-decomposition, the output coefficients are reordered so that the low-frequency coefficients are stored on the left part of the output, while the high-frequency coefficients are stored on the right part.

Without targeting a specific architecture, the designs of the SBF category can be studied in three different cases [23] [30], depending on the pursued optimizations. Apart from the pictorial presentation of this section, the reader is referred to [23] for detailed pseudocode for the various cases. The basic memory component used is array IMG that contains the input image and the final result after every decomposition level. This array requires $N_1 \times (N_2 \cdot c_p)$ bytes of memory, where c_p denotes the number of bytes that represent one wavelet coefficient in memory, and N_1, N_2 denote the size of input data rows and columns, expressed in pixels. Additionally, one extra array is utilized, named TMP_IMG. The size of this array varies, depending on the specific case and the value of S. Below, a short description is given for the three distinct cases of the application of a discrete wavelet decomposition at level l, with $0 \le l < L$, where L denotes the total decomposition levels. Due to the dyadic (reordered) organization of the coefficients in IMG, only the upper-right $(N_1 \cdot 2^{-l}) \times (N_2 \cdot 2^{-l})$ elements of the array are used for level l:

- Case I: Minimum memory implementation. In this case, TMP_IMG is a one-dimensional array containing $0.5 \cdot \max\{N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}\} \cdot c_p$ bytes. The application of the transform in one row can be seen in Figure VI-1 for level l. As demonstrated by the left part of Figure VI-1, this case applies the predict-and-update steps of the lifting scheme directly in the elements of IMG and replaces the results in-place (grey dots). After the end of each row or column processing, a reordering procedure is applied to the newly produced coefficients of the current row or column of IMG with the aid of TMP_IMG array by extracting all the high-frequency coefficients (white dots), shifting all low-frequency coefficients (black dots) into neighbouring positions and then storing back the contents of TMP_IMG (Reordering of Figure VI-1).

- Case II: Minimum number of memory accesses for the transform production. In this case, as shown in Figure VI-2, if $S = 1$ (lower part of the figure) TMP_IMG is a two-dimensional array with $(N_1 \cdot 2^{-l}) \times (N_2 \cdot 2^{-l})$ elements of c_p bytes each. If $S > 1$ (upper part of Figure VI-2) TMP_IMG is a one-dimensional array with $\max\{N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}\}$ elements. The application of the transform in this case for the l-th decomposition level is pictorially explained in Figure VI-2 and leads to the minimum amount of array access/copy operations [32].

- Case III: Maximum locality in the processing. As shown in Figure VI-3, initially each new row or column of IMG is copied in TMP_IMG, which is a one-dimensional array occupying $\max\{N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}\}$ elements. All the subsequent predict-and-update passes occur locally in TMP_IMG. The results of the last pass are directly stored back into IMG in reordered form as seen from the right part of Figure VI-3.

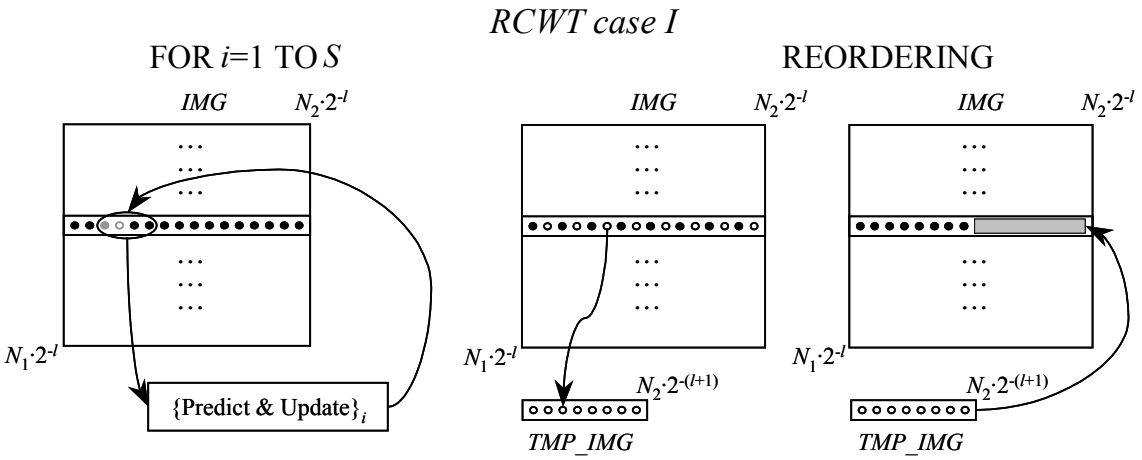


Figure VI-1. Application of RCWT case I. Only the row-by-row processing is shown, a similar scheme applies for the columns.

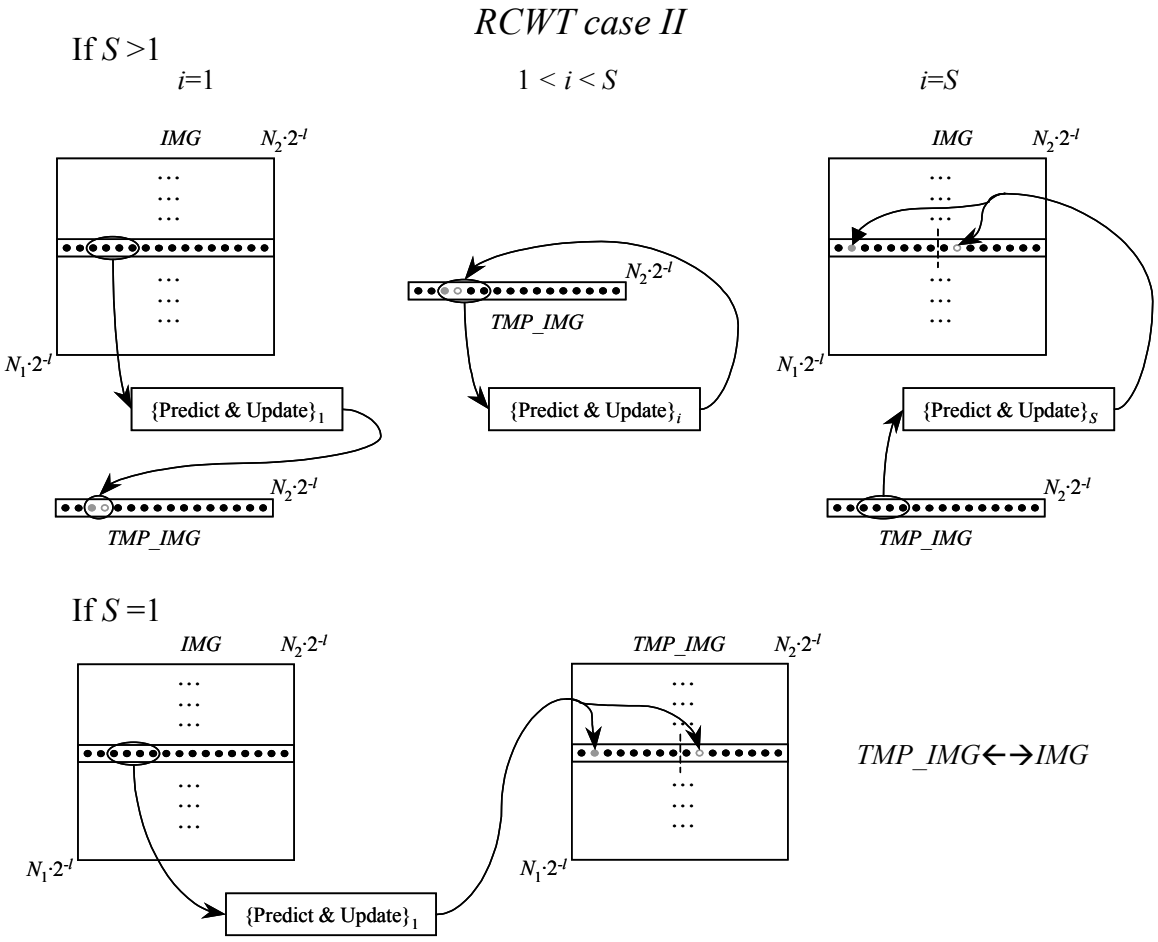


Figure VI-2. Application of RCWT case II. Only the row-by-row processing is shown, a similar scheme applies for the columns. Operation $TMP\_IMG \leftarrow\rightarrow IMG$ denotes the exchange of memory pointers of IMG and TMP_IMG [23].

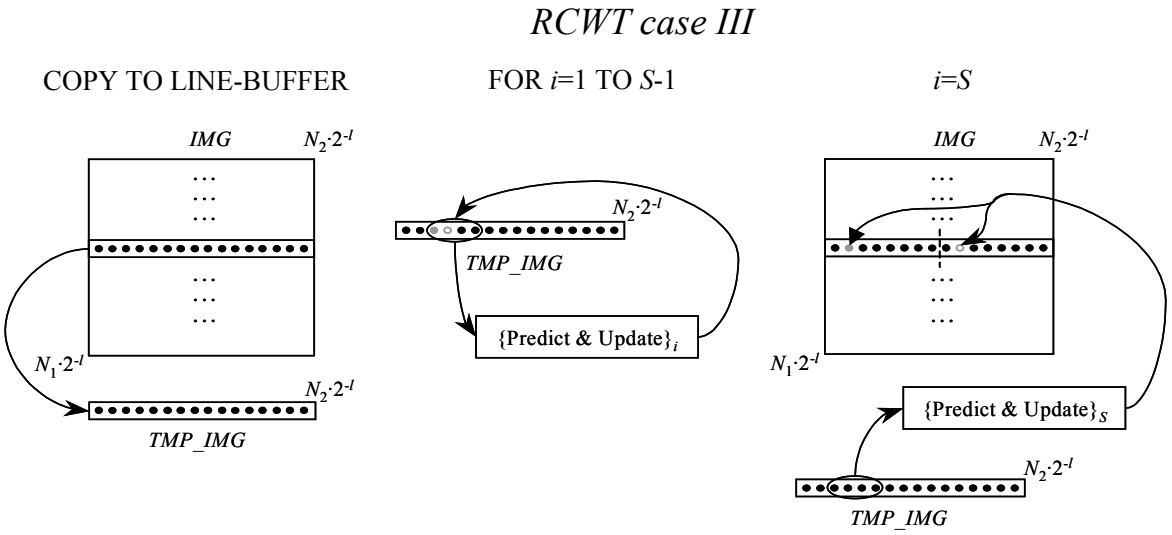


Figure VI-3. Application of RCWT case III. Only the row-by-row processing is shown, a similar scheme applies for the columns.

6.2.1.2 The design of roughly depth-first methods - The LWT and LBWT approaches.

These methods are based on the separation of the input of every decomposition level into non-overlapping components, the processing of these components using some of the previously produced low-frequency coefficients (or image samples for level $l = 0$) and the update of these coefficients with some of the newly-produced low-frequency coefficients.

The outputs of every level (high-frequency coefficients) are usually driven to a compression engine according to the selected compression algorithm. Typically, the outputs are grouped in parent-children trees [20] or intra-subband blocks [13]. However, as mentioned in the previous section, the link with a specific coding system is not studied in this chapter.

The input components are blocks for the LWT method (block-based) and lines for the LBWT method (line-based). Since usually a software implementation targets a single-processor system, the single-processor architecture of [20] is selected. However, unlike [20], the presented design allows the implementation of both methods with the same architecture, due to the flexibility of a software implementation, which supports different memory configurations. Thus, the line-based architecture presented in this section can be seen as the asymptotic case of the block-based scheme, when the block width becomes equal to the image width. A detailed pseudocode for the software implementation of the methods of this category is presented in [23]. The LWT design that is presented in this section has been successfully used in a coding system in [21], where the reader can also find details about the initialization and finalization phenomena at the image borders.

The processing performed by the RDF methods is shown pictorially in Figure VI-4 for decomposition level l. The used arrays follow the naming conventions of [20] and [21] to emphasize the relationships with that architecture. Thus, IPM denotes the inter-pass memory, where each new input component is written. In general, IPM is a two-dimensional array occupying $(X \cdot 2^L) \times (Y \cdot 2^L \cdot c_p)$ bytes, with $X, Y \in \mathbb{Z}_+$ parameters of the design; for the LBWT, IPM has a static width of $N_2 \cdot c_p$ bytes. Since the filtering is interleaved on the columns for every decomposition level (and also on the rows for the block-based architecture), the intermediate results are stored in the overlap memory, denoted as OM_ROWS and OM_COLS in Figure VI-4. For biorthogonal filter-banks with $(2M+1)/(2M-1)$ taps, OM_ROWS is a three-dimensional array (decomposition level/row of IPM/overlap coefficient) that occupies $2^{L+1} \cdot (2M-1) \cdot c_p$ bytes (maximally) and OM_COLS is a three-dimensional array (decomposition level/column of current level/overlap coefficient) that occupies maximally $2N_2 \cdot (2M-1) \cdot c_p$ bytes. The actual filtering is performed using a small one-dimensional array of $(2M+1) \cdot c_p$ bytes, called filtering-FIFO (FF) to emphasize the inherent functionality, and is convolution based.

As seen in the right part of Figure VI-4, for every row of IPM, first the coefficients of the corresponding row of OM_ROWS are inserted in FF, followed by each new pair of coefficients of the current row of IPM. After the filtering, the resulting low and high-frequency coefficients are written in-place in IPM. When the filtering of the current row of IPM is completed, the results remaining in FF are written back into the current row of OM_ROWS for the processing of the neighbouring block. A similar scheme applies for the columns, with the aid of OM_COLS. It must

be noted that for the line-based approach the row filtering is not split in successive parts, thus only OM_COLS is utilized. In addition, lifting-based approaches can be considered as well [15], if the memory sizes of the various buffers are adapted to the specific data-dependencies of the lifting-based filtering.

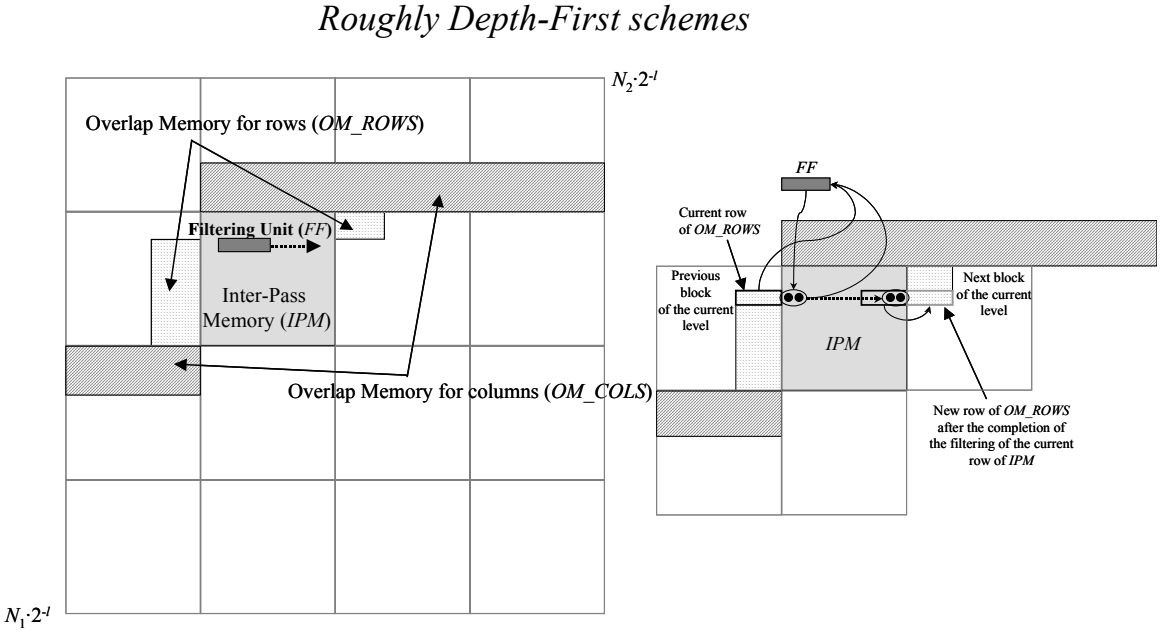


Figure VI-4. Illustration of the processing of one input component for the RDF methods.

6.2.2 Calculation of the Data-related Cache Penalties

6.2.2.1 Constraints of the presented Analysis

For every different method of Subsections 6.2.1.1 and 6.2.1.2, the expected number of D-Cache misses varies, depending on the cache organization of the target execution-platform. For simplicity in the presented analysis, the case of a fully-associative data-cache is selected; in this way however, the conflict misses of an n-way set-associative cache are excluded from the description [26]. In the case of a miss due to lack of space in the cache (capacity miss), the LRU replacement strategy [26] is assumed, where the cache block that was least-recently used is replaced with the new input block. For each of the different approaches of the DWT, the boundary effects (initialization and finalization phenomena) are ignored so as to facilitate the description. In addition, the I-Cache behaviour will not be discussed because, as the experimental results show (see Table VI-I for an example) that for the systems of interest, the I-Cache misses are much less frequent than the data-related cache-misses. Apart from the algorithm parameters, the important cache parameters must be defined. We denote s_{L2}, s_D the cache size of the L2 (data-cache) and D-Cache respectively, and b_{L2}, b_D the corresponding blocksizes. The blocksize represents the number of sequential bytes that are inserted in every cache

from the higher level of the memory hierarchy whenever an access to that level is performed [26]. In general, due to paging and burst-mode effects of the SDRAM chips, the number of bytes transferred in every RAM access is not always fixed [26]. Thus, if a more detailed analysis is desired, these effects should be incorporated in the model.

We target realistic cache implementations where only a part of the input image can be buffered in the L2-Cache and also, in some cases, only a fraction of the currently-processed input (row, column or block, line) can fit in the D-Cache. This is because, even in future designs of custom DSPs that will target wavelet-related compression applications, the minimization of the on-chip cache will be a critical issue that determines performance and cost. Thus, a solution where the entire input image can fit in the cache is not expected to be feasible. In order to theoretically define the cases for which the analysis of this section is applicable, we set the following constraints:

$$2N_2 \cdot c_p < s_{\mathrm{D}} < N_1 \cdot b_{\mathrm{D}} \tag{6.1}$$

$$2N_1 \cdot b_{\mathrm{L2}} < s_{\mathrm{L2}} < N_1 \cdot N_2 \cdot c_p \tag{6.2}$$

where $b_{\mathrm{D}} > 2\left(N_2/N_1\right) \cdot c_p$ so that (6.1) is valid. These constraints come from the way the utilized two-dimensional arrays are stored in memory. Typically, a two-dimensional array is allocated in memory as a sequence of rows. Thus, when accessing elements in the row direction, sequential memory accesses are performed that in fact are grouped together since the cache organization of every level always inputs b_{L2} or b_{D} sequential bytes. However, when accessing elements in the column direction, these elements are not sequential in the physical memory and so, for accessing k sequential coefficients of one column, $k \cdot b_{\mathrm{L2}}$ or $k \cdot b_{\mathrm{D}}$ bytes will be copied in the L2 or D-Cache due to the unavoidable grouping. Having this in mind, the constraint of (6.1) means that the D-Cache is sufficiently large to hold input coefficients that correspond to two image rows ($2N_2 \cdot c_p$ bytes), but not large enough to sustain the coefficients of one entire image column ($N_1 \cdot b_{\mathrm{D}}$ bytes). The constraint of (6.2) means that the L2-Cache is sufficiently large to hold input coefficients that correspond either to two portions of b_{L2} sequential columns when each portion is coming from a different two-dimensional array, or to $2b_{\mathrm{L2}}$ sequential columns when they come from the same array. In addition, an upper constraint is set; the L2-Cache is always smaller than the original image size, otherwise only compulsory misses for the initial image input are expected in the L2-Cache. In general, the constraints of (6.1) and (6.2) are chosen so that the lower bound certifies that no exponential increase happens in the data-related misses, while the upper bound certifies that the data-related misses are not negligible in comparison to the instruction-related misses.

As seen from the description of this section, we focus on a two-level cache architecture. It must be noted however that the presented theoretical analysis is invariant to the number of levels; it only depends on the cache characteristics of the system. This is a natural consequence from the fact that the behaviour of the cache memories is invariant to the specific level they belong to, since they are designed to operate independently, ignoring the upper and lower-level memory organization [26]. In this way, our theoretical model can cover both a two-level and a one-level cache hierarchy, if the equations for the prediction of the misses in level two are ignored. As a result, the vast majority of processors found in the market is covered by our description.

6.2.2.2 The expected data-related cache misses.

Under the constraints of (6.1) and (6.2), the data-related cache misses can be estimated. For all the presented methods, one iteration through the input data consists of the input of an image component (row, column or block), the application of the filtering process to that component and the production and storage of the results. Since the various methods perform similar operations, such as sequential accessing along a row or a column of a two-dimensional matrix, the calculation of the various data-related cache misses, was made using some templates for the most typical cases, specifically adapted for each method.

For the various cases that follow, the term processing means the input of the specific array elements in the cache, the application of the filtering (for the convolution-based implementation) or a number of predict-and-update steps (for the lifting-scheme implementation), and the storage of the final results. The notation $A \gg B$ indicates that A is up to one order of magnitude larger than B, while $A \approx B$ denotes that A is comparable or smaller than B.

Template 1 (T1): Row-by-row processing of a two-dimensional array of $R \times (C \cdot c_p)$ bytes in a cache of s_c bytes with blocksize b_c.

Constraint:
$$s_c \gg C \cdot c_p \tag{6.3}$$

Expected misses:
$$\mathrm{T1}(R, C, b_c) = (R \cdot C \cdot c_p) \big/ b_c \tag{6.4}$$

Proof. See Appendix.

Template 2 (T2): Column-by-column processing of a two-dimensional array after the completion of the row-by-row processing. The array has $R \times (C \cdot c_p)$ bytes and is inserted in a cache of s_c bytes with blocksize b_c.

Constraint:
$$s_c \approx R \cdot C \cdot c_p \;\&\; s_c \gg R \cdot b_c \tag{6.5}$$

Expected misses:
$$\mathrm{T2}(R, C, b_c, s_c) = \sum_{i=1}^{(C \cdot c_p)/b_c} (W + \sum_{j=1}^{i} a_j) \tag{6.6}$$

with:

$$W = (R \cdot C \cdot c_p - s_c) \big/ (C \cdot c_p) \tag{6.7}$$

$$a_j = \begin{cases} b_c \left(W + \sum_{u=1}^{j-1} a_u \right) \big/ (C \cdot c_p - b_c), & \text{if } W + \sum_{u=1}^{j-1} a_u \le R \\ 0, & \text{otherwise} \end{cases} \tag{6.8}$$

Proof. See Appendix.

Template 3 (T3): Column-by-column processing of a two-dimensional array after the completion of the row-by-row processing. The array has $R \times (C \cdot c_p)$ bytes and is inserted in a cache of s_c bytes with blocksize b_c. Each column is processed k times (k passes per column).

Constraint: $$s_c \approx R \cdot b_c \tag{6.9}$$

Expected misses: $$\mathrm{T3}(R, C, k, s_c) = f_{\mathrm{T3}} \cdot R \cdot C \tag{6.10}$$

with

$$f_{\mathrm{T3}} = \begin{cases} 1, \text{ if } s_c > R \cdot b_c \\ k, \text{ otherwise} \end{cases} \tag{6.11}$$

Proof. See Appendix.

The interested reader can find in detail the explanation for the expected misses cause by the various templates in the Appendix (Subsection 6.5). In short, template T1 gives the expected misses from a row-by-row processing of a two-dimensional array when a large cache is concerned, i.e. the L2-Cache. Template T2 gives the expected misses from a subsequent column-by-column processing of the input two-dimensional array, and template T3 shows the misses from the column-by-column processing when a small cache is concerned, i.e. the D-Cache.

The following two subsections analyze our findings.

A Data misses in the L2-Cache.

Beginning from the strictly breadth-first (SBF) production methods, as shown in Figure VI-1 – Figure VI-3, all cases will process sequentially (in the row or column direction) the elements of array IMG in the L2-Cache. Additionally case II with $S = 1$ will input the elements of array TMP_IMG (two-dimensional array), since the complete array does not fit in the L2-Cache due to the constraint of (6.2). In all cases where TMP_IMG is a one-dimensional array, it is assumed that, after the initial input, no additional misses occur from this array, since it is reused in every row or column processing and hence it is expected to remain in the cache. This case is simplified to the row-by-row and column-by-column processing of a two-dimensional array with the exception of case II with $S = 1$, where another two-dimensional array is inserted in the cache at the same time. By applying the template T1 for the rows, the misses due to the row filtering of level l are:

$$\begin{aligned} SBF\_row\_miss_{\mathrm{L2}}(l) &= f_S \cdot f_{\mathrm{L2}} \cdot \mathrm{T1}(N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}, b_{\mathrm{L2}}) \\ &= f_S \cdot f_{\mathrm{L2}} \cdot N_1 \cdot N_2 \cdot 2^{-2l} \cdot c_p / b_{\mathrm{L2}} \end{aligned} \tag{6.12}$$

where:

$$f_S = \begin{cases} 2, \text{ if case\_II } \& \ S = 1 \\ 1, \text{ otherwise} \end{cases}, \quad f_{\mathrm{L2}} = \begin{cases} 0, \text{ if } l > 0 \ \& \ f_S \cdot N_1 \cdot N_2 \cdot c_p \cdot 2^{-2(l-1)} < s_{\mathrm{L2}} \\ 1, \text{ otherwise} \end{cases} \tag{6.13}$$

The factor f_S of (6.12) distinguishes that sole case where two two-dimensional arrays are inserted in the cache. The factor $N_1 \cdot 2^{-l}$ represents the number of rows of the level l. Every row of this level has $N_2 \cdot 2^{-l} \cdot c_p$ bytes. The f_{L2} factor is a bound that checks whether the output of the previously produced decomposition level fits in the L2-Cache. If so, then no additional misses are accounted for. Otherwise, all low-frequency coefficients are assumed non-resident in the cache.

For the L2-Cache misses of the column processing, after the completion of the row processing of level l, the template T2 is applicable; thus, the number of L2-Cache misses for the processing of level l is:

$$\begin{aligned} SBF\_column\_miss_{\mathrm{L2}}(l) &= \mathrm{T2}(N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}, b_{\mathrm{L2}}, s_{\mathrm{L2}}) \\ &= \sum_{i=1}^{\frac{fS \cdot N_2 \cdot 2^{-l} \cdot c_p}{b_{\mathrm{L2}}}} (W + \sum_{j=1}^{i} a_j) \end{aligned} \tag{6.14}$$

where the f_S factor, as before, determines the total amount of coefficients inserted, according to the specific case and the number of passes (S) through the newly-inserted row or column. The factors W and a_j are defined by equations (6.7) and (6.8).

For the roughly depth-first (RDF) methods the calculation of the data misses in the L2-Cache is simpler. For modest values of X, Y, based on the constraint of (6.1), after the compulsory cache-misses caused by the row-by-row processing of each new block (template T1), the column-by-column processing is not expected to cause any additional misses since all the data for the current block are assumed resident in the cache. Hence, template T2 does not apply and no misses are expected during this procedure. The interested reader can define exactly the set of values of X, Y according the specific case where the constraint of (6.5) is not satisfied (and thus the template is not applicable). Essentially this means that s_c must be larger than $R \cdot C \cdot c_p$, where $s_c = s_{\mathrm{L2}}$ and $R = X \cdot 2^L$ and $C = Y \cdot 2^L$ (for the block-based). In this case, for the block-based method:

$$\begin{aligned} RDF\_miss_{\mathrm{L2}} &= \left[N_1 / (X \cdot 2^L)\right] \cdot \left[N_2 / (Y \cdot 2^L)\right] \cdot \mathrm{T1}(X \cdot 2^L, Y \cdot 2^L, b_{\mathrm{L2}}) \\ &= N_1 \cdot N_2 \cdot c_p / b_{\mathrm{L2}} \end{aligned} \tag{6.15}$$

The expected number of misses of the line-based method is given also from (6.15) if one replaces $Y \cdot 2^L$ with N_2 (the line width), since the input-block of the line-based method consists of entire rows. It must be noted that both the LWT and LBWT methods are tuned to a streaming data input; hence, the image data are collected in a sequence of 2^L rows either directly in IPM (LBWT method), or in a row-buffering memory of size 2^L image rows (LWT method), overwriting the previously inserted data. Since this memory is expected to be resident in the L2-Cache, if the image-input procedures are not taken into account in the total cache-miss budget, the expected misses in the L2-Cache for the RDF methods are then obtained by replacing by 2^L in equation (6.15):

$$RDF\_miss_{\mathrm{L2,DIRECT\_INPUT}} = 2^L \cdot N_2 \cdot c_p / b_{\mathrm{L2}} \,. \tag{6.16}$$

Equation (6.16) is useful because it excludes the image-input procedures, which are system-dependent functions. Thus it models the extreme (fictitious) case where the input data are written directly in the

L2-Cache from the input and not from the main memory. The actual experimental results are expected to be between the numbers reported by (6.15) and (6.16), depending on how much the image-input procedures are taken into account by profiling tools that produce these results.

B Misses in the D-Cache.

For the strictly breadth-first methods, the misses occurring in the D-Cache due to the row processing can be calculated using template T1 since the constraint of (6.1) assures the applicability of this template for all cases. Thus, the D-Cache misses for the row processing of level l are simply:

$$SBF\_row\_miss_{\mathrm{D}}(l) = \mathrm{T1}(N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}, b_{\mathrm{D}}) = f_S \cdot N_1 \cdot N_2 \cdot 2^{-2l} \cdot c_p / b_{\mathrm{D}} \qquad (6.17)$$

with f_S defined by equation (6.13). The misses during the column processing are more complex, since, as shown by the constraint of (6.1), an entire input column does not always fit in the D-Cache.

For case I (see Figure VI-1), if each column of IMG of decomposition level l does not fit in the D-Cache, the application of the lifting scheme in S passes can be calculated using template T3. The reordering procedure will cause an additional pass per column of IMG and the input of TMP_IMG for every column. These misses can be calculated by templates T3 and T1, respectively. Furthermore, as shown in the right part of Figure VI-1, the last step of the reordering requires another pass through the last half of every column, hence these additional misses (if any) are calculated with template T3. Summarizing:

$$\begin{aligned} SBF\_caseI\_col\_miss_{\mathrm{D}}(l) &= \mathrm{T3}(N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}, S{+}1, s_{\mathrm{D}}) + N_2 \cdot 2^{-l} \cdot \mathrm{T1}(1, 0.5N_1 \cdot 2^{-l}, b_{\mathrm{D}}) \\ &\quad + f_{\mathrm{D}a} \cdot \mathrm{T3}(N_1 \cdot 2^{-l}, 0.5N_2 \cdot 2^{-l}, 1, s_{\mathrm{D}}) \\ &= (f_{\mathrm{T3}} + 0.5\, c_p / b_{\mathrm{D}} + 0.5 f_{\mathrm{D}a}) \cdot N_1 \cdot N_2 \cdot 2^{-2l} \end{aligned} \qquad (6.18)$$

where f_{T3} is defined in (6.11) and $f_{\mathrm{D}a}$ checks if half a column fits in the cache:

$$f_{\mathrm{D}a} = \begin{cases} 1, \text{ if } N_1 \cdot 2^{-l} \cdot b_{\mathrm{D}} > 2s_{\mathrm{D}} \\ 0, \text{ otherwise} \end{cases} \qquad (6.19)$$

For case II (see Figure VI-2), if one pass is performed ($S = 1 \Rightarrow f_S = 2$) then one column from both IMG and TMP_IMG (two-dimensional) arrays of the current decomposition level must enter the D-Cache, causing compulsory misses that can be calculated with template T3. Otherwise, one column of the current level and TMP_IMG array (one-dimensional) must enter the D-Cache and templates T3 and T1 are used (in this case $f_S = 2$). Hence, the misses in the D-Cache from the column filtering of level l are:

$$\begin{aligned} SBF\_caseII\_col\_miss_{\mathrm{D}}(l) &= f_S \cdot \mathrm{T3}(N_1 \cdot 2^{-l}, N_2 \cdot 2^{-l}, S, s_{\mathrm{D}}) \\ &\quad + (2 - f_s) \cdot N_2 \cdot 2^{-l} \cdot \mathrm{T1}(1, N_1 \cdot 2^{-l}, b_{\mathrm{D}}) \\ &= N_1 \cdot N_2 \cdot 2^{-2l} \cdot \left[f_S \cdot f_{\mathrm{T3}} - (f_S - 2) \cdot c_p / b_{\mathrm{D}} \right] \end{aligned} \qquad (6.20)$$

For case III (see Figure VI-3), for the filtering of each column of the current decomposition level, the column itself and TMP_IMG must enter D-Cache (copy operation of Figure VI-3). The compulsory misses of this operation are calculated with templates T3 and T1 respectively. The application of $S-1$ predict-and-update steps (middle of Figure VI-3) does not cause any additional misses since the constraint of (6.1) certifies that TMP_IMG is expected to reside in the D-Cache. Finally, the last predict-and-update step (last pass) will directly store the final results in binary-tree (reordered) form as shown in Figure VI-3 and the capacity misses expected from this operation are calculated with template T3. Summarized:

$$\begin{aligned} SBF\_caseIII\_col\_miss_{\mathrm{D}}(l) &= 2\cdot \mathrm{T3}(N_1\cdot 2^{-l}, N_2\cdot 2^{-l}, 1, s_{\mathrm{D}}) \\ &\quad + N_2\cdot 2^{-l}\cdot \mathrm{T1}(1, N_1\cdot 2^{-l}, b_{\mathrm{D}}) \\ &= N_1\cdot N_2\cdot 2^{-2l}\cdot (2 + c_p/b_{\mathrm{D}}) \end{aligned} \tag{6.21}$$

For the roughly depth-first methods, the expected number of D-Cache misses can be calculated based again on the templates presented before. For simplicity we study the case where $X = Y = 1$. For the LWT method, all the memory components used for the filtering procedures of every level are expected to reside in the D-Cache after their initial input, except the OM_COLS array, which is the only array that does not completely fit in the cache according to the constraint set in (6.1). For the initial input of every image block in the IPM, the compulsory misses in the D-Cache can be calculated with the application of template T1. The additional misses occurring in the D-Cache are due to the copying of the coefficients stored in OM_COLS to FF, as seen in the right part of Figure VI-4; for every decomposition level l, this procedure consists of sequential processing along a row of OM_COLS of level l, hence again template T1 can be utilized. In total, the following D-Cache misses are expected for the block-based method:

$$\begin{aligned} RDF\_LWT\_miss_{\mathrm{D}}(l) &= N_1\cdot N_2\cdot 2^{-2L}\cdot \left[2^L\cdot \mathrm{T1}(1, 2^L, b_{\mathrm{D}}) + \mathrm{T1}(1, 2M-1, b_{\mathrm{D}})\cdot \sum_{l=1}^{L} 2^l\right] \\ &= N_1\cdot N_2\cdot 2^{-2L}\cdot \left[2^{2L}\cdot c_p/b_{\mathrm{D}} + (2^{L+1}-2)\cdot (2M-1)\cdot c_p/b_{\mathrm{D}}\right] \end{aligned} \tag{6.22}$$

The value of M depends on the used filter-bank taps. Usually a biorthogonal $(2M+1)/(2M-1)$ filter-bank is used with $M = 2$ or $M = 4$ [13].

For the line-based method, IPM consists of entire rows of the input image and for the row processing, due to the constraint of (6.1), only the two most-recently accessed rows of IPM are expected to reside in the D-Cache. The coefficients of every level are skewed in the IPM in order to perform in-place storage of the filtering results. For every decomposition level l, for the row filtering, template T1 is used for the D-Cache miss calculation and for the column filtering, template T3 gives the expected misses from the column input of IPM, while template T1 estimates the misses from the copying of the elements of OM_COLS of the current level to FF. In total, for L decomposition levels, the expected D-Cache misses are:

$$
\begin{aligned}
RDF\_LBWT\_miss_{\mathrm{D}}(l) = N_1 \cdot 2^{-L} \cdot \Bigg[& \sum_{l=1}^{L} \mathrm{T1}(2^l, N_2, b_{\mathrm{D}}) + \sum_{l=1}^{L} \mathrm{T3}(2^L, N_2, 1, s_{\mathrm{D}}) \\
& + \mathrm{T1}(1, 2M-1, b_{\mathrm{D}}) \cdot \sum_{l=1}^{L} 2^l \Bigg] \\
= N_1 \cdot 2^{-L} \cdot \Big[& N_2 \cdot \left(c_p / b_{\mathrm{D}} \right) \cdot (2^{L+1} - 2) + L \cdot 2^L \cdot N_2 \\
& + (2M-1) \cdot \left(c_p / b_{\mathrm{D}} \right) \cdot (2^{L+1} - 2) \Big]
\end{aligned}
\tag{6.23}
$$

The presented theoretical analysis follows [30] and leads to equations similar to the ones reported in [23], with a few simplifications in the calculations. Using the formulas of equations (6.12) – (6.16) for the L2-Cache and (6.17) – (6.23) for the D-Cache, the data-related cache-misses can be found for the dyadic wavelet decomposition of an $N_1 \times N_2$ image in L decomposition levels for any typical wavelet filter-bank. For the filter operations, either the lifting-scheme approach can be used for the strictly breadth-first approaches (hence S passes will be applied for every input row or column), or a classical convolution-based implementation can be utilized for the roughly depth-first methods. As mentioned before, the convolution-based filtering was chosen so as to facilitate the generality of the presented design for the roughly depth-first methods without considering filter-specific lifting dependencies.

6.2.3 Experimental Validation of the Proposed Model

To experimentally check which traversal method appears to be the most efficient one with respect to the minimization of the data-related cache penalties, the different approaches illustrated in Subsection 6.2.1 were implemented using ANSI-C and were compiled and simulated under the same conditions. All source codes were platform-independently optimized for speed.

The cache-miss penalties were measured using the tools of [33] in a typical 64-bit simplescalar processor architecture. The specified cache organization simulated a two-level, separable instruction and data cache with the LRU replacement-strategy. Each of the two cache levels was considered fully associative, with 128 and 2048 blocks for the first and second level respectively. This configuration satisfies the constraints of (6.1) and (6.2) for the majority of input image sizes. A block size of 64-bytes was selected for both levels. Figure VI-5 shows the results for the 9/7 filter-bank with floating-point precision for the wavelet coefficients ($S = 2$, $M = 4$, $c_p = 4$), while Figure VI-6 shows the results for the 5/3 filter-bank with fixed-point precision ($S = 1$, $M = 2$, $c_p = 2$). All results are averages of the execution for 4 to 6 decomposition levels, which correspond to typical settings for image and texture compression applications. Instead of absolute values, the theoretical curves have been uniformly scaled to the maximum experimental point of every graph and all measurements are shown as a percentage of the maximum measurement, so as to facilitate the comparisons. As discussed in the previous section, the streaming-input design of the RDF methods is expected to have a linear increase of the data-related cache misses when the image size increases (equation (6.15)). However, since the image I/O from/to the storage medium is not measured by the simulation tools, the experimental results follow equation (6.16) instead. An abnormality is seen in the L2-Cache misses of the LBWT method, where the experimental points do not follow the theoretical curve; instead an exponential increase is observed. This difference is explained from the fact that for the LBWT method, for large image sizes and 5 or 6 decomposition levels, the IPM size becomes larger than the

L2-Cache size; thus the assumption that the complete *IPM* is resident in the L2-Cache does not hold any more and the theoretical calculations fail to comply with the actual measurements. This is especially noticeable for the 9/7 filter-bank (Figure VI-5) since there the *IPM* is two times larger than for the 5/3 filter-bank (Figure VI-6).

To show the relative comparison between instruction and data-related cache misses, in Table VI-I the average misses for the various system caches are displayed for the SBF category for a variety of cases. The ratios of Table VI-I show that, under the constraints of (6.1) and (6.2), the classical implementation of the wavelet transform is a data-dominated application; in the vast majority of cases, the misses in the I-Cache are 1-4 orders of magnitude less than the ones reported for the D-Cache. These facts validate our assumptions that the data-related cache penalties play a dominant role in the transform execution-speed.

To show the effectiveness of the different approaches in real systems, first the Intel-Pentium processor [34] was selected since it has a typical, widely used, superscalar architecture. The processor cache is a two-level four-way set-associative cache. The first level consists of separate instruction and data cache of size 8 Kb each, while the second level has a joint organization of the instruction and data cache with total size 256 Kb. The blocksize is 32 bytes. As a result, significant differences exist between this cache organization and the cache model of the simplescalar processor that was used for the verification of the theoretical analysis. In the Pentium cache, apart from the capacity misses and compulsory misses, the total miss-rate comes also from conflict misses. In addition, the data-bank conflicts reflect the effects of the multiple Functional Units, because they refer to conflicts in the D-Cache caused by simultaneous accesses to the same data-bank by the execution-pipelines (U and V pipes [34]).

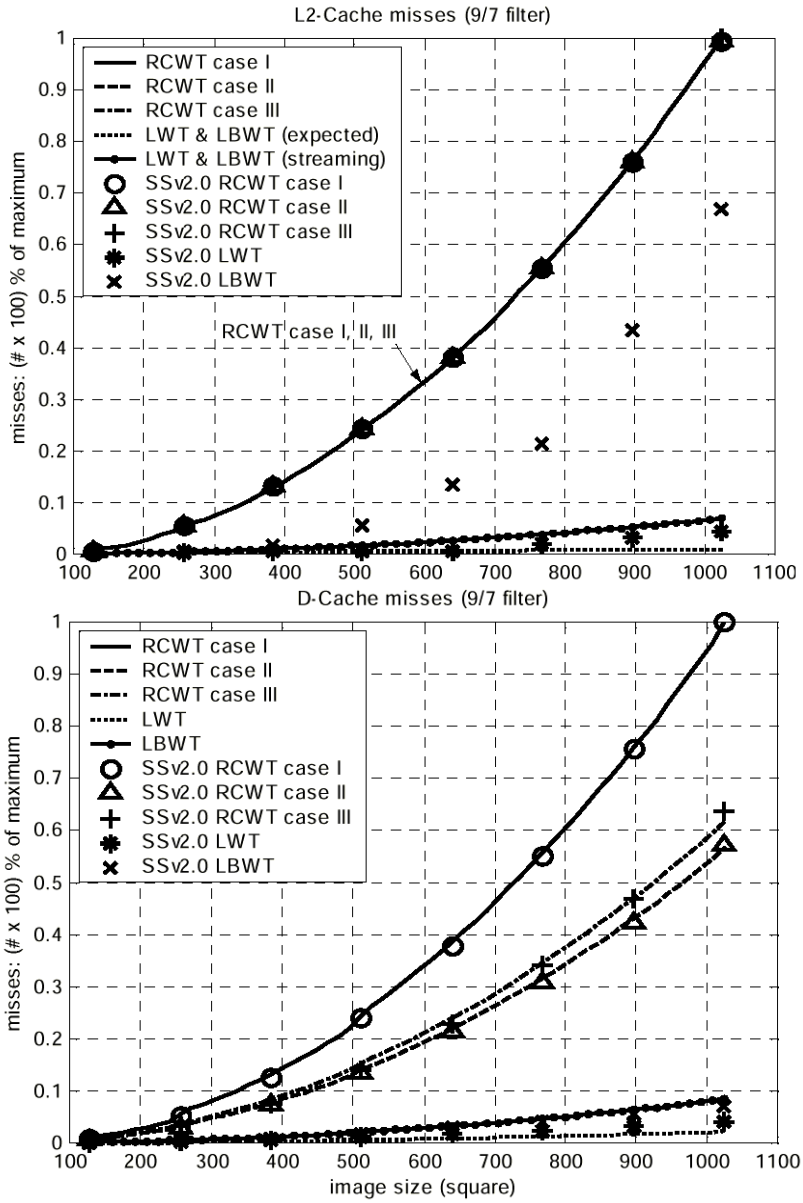


Figure VI-5. Theoretical (lines), and experimental (points) data-related cache misses measured with simulator tools, for the 9/7 filter-bank. SS denotes the SimpleScalar simulator toolset.

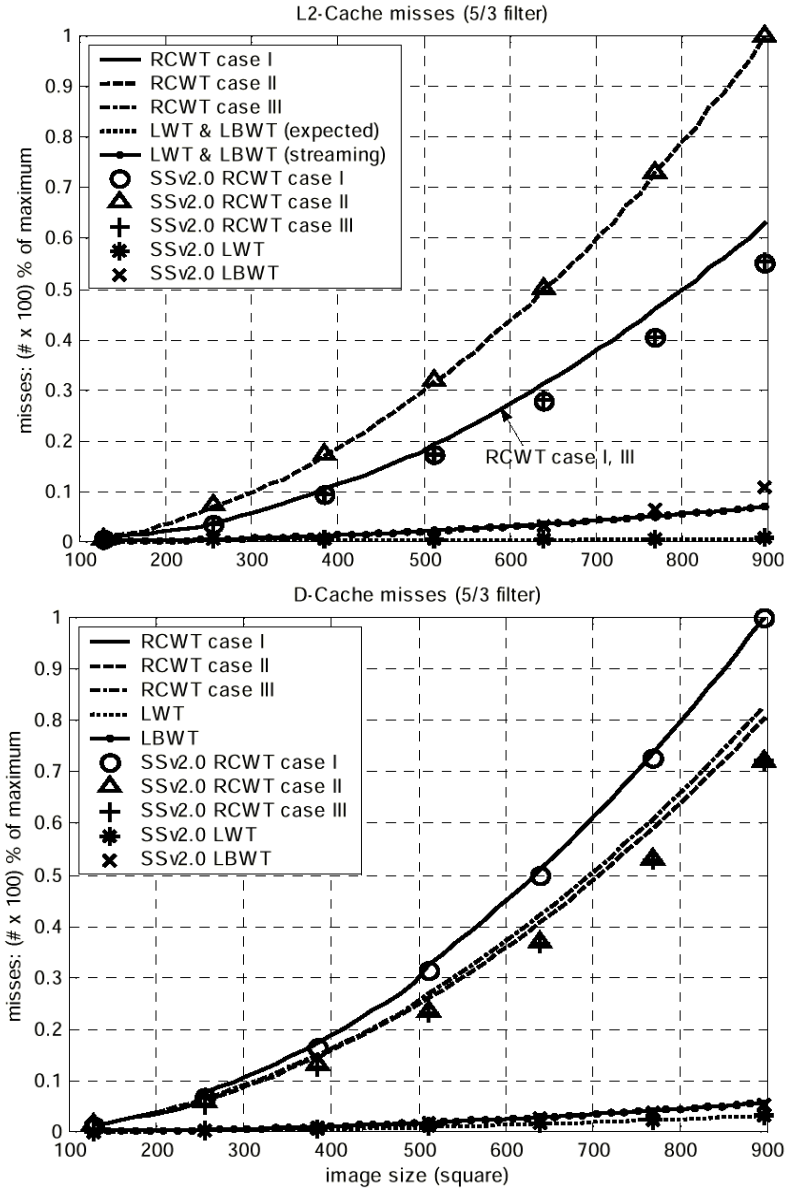


Figure VI-6. Theoretical (lines), and experimental (points) data-related cache misses measured with simulator tools, for the 5/3 filter-bank. SS denotes the SimpleScalar simulator toolset.

Strictly Breadth-First Methods
(average for cases I, II, III with 5 decomposition levels)

Image size	D-Cache	I-Cache	**Ratio**	L2-Cache (data)	L2-Cache (instruction)	**Ratio**
128×128	37559	1174	**32:1**	1465	753	**1.95:1**
256×256	236156	1187	**199:1**	17475	765	**23:1**
512×512	1050275	1188	**884:1**	74905	766	**98:1**
1024×1024	4483127	1242	**3610:1**	305505	783	**390:1**

Table VI-I. Cache misses for the SBF category, as measured with the tools of [33] for various image sizes. The 9/7 filter-bank was used.

In Figure VI-7, the experimental results in the Pentium architecture are displayed as cache-miss rate or conflict-miss rate versus image-size, decomposition-levels graphs. In addition, the theoretical calculations of this case, based on the equations of Subsection 6.2.2, are shown in Figure VI-8. All the experimental results of Figure VI-7 were produced by sampling the processor registers during the dedicated, single-process execution using the tool of [35] for increased accuracy. Only the specific portions of the source codes that produce the transform are profiled, in order to avoid any system-dependent I/O procedures.

For the D-Cache, if one takes into account the data-bank conflicts shown in Figure VI-7 (which will also produce stall cycles), then the theoretical (predicted) ordering of the methods agrees with what is seen in the experiments, although not numerically (as for the simplescalar processor). In addition, in the RDF category, both methods appear to minimize the bank conflicts, mainly because the localization of the processing into separate small memory components (OM_ROWS, OM_COLS, FF, IPM) helps during the parallel execution, since the simultaneously executed commands are expected to access memory arrays that are usually resident in different (non-conflicting) banks.

For the L2-Cache, the theoretical calculations of Figure VI-8 are not in complete agreement with the experimental results of Figure VI-7 with respect to the ordering of the methods for the cache misses, mainly because this cache is a joint organization of instruction and data caches, thus the miss penalties are affected also by the instruction-related behaviour of every method. Nevertheless, in general, the RDF category again appears to minimize the measured penalties in comparison to the SBF methods.

In order to experimentally validate the link between the data-related cache misses and the obtained throughput, Table VI-II shows throughputs in an Intel Pentium III 700Mhz/Win2000. Although the RDF methods were found to have increased instruction cache misses in comparison to the SBF approaches, the reduction of the data-related cache misses in the LWT and LBWT methods practically doubles the achieved throughput, especially at large images.

In a second experiment we used a typical Very Long Instruction Word (VLIW) DSP architecture, namely the TriMedia TM1 processor [36] introduced by Philips. TriMedia is a 32-bit dedicated media processor for high performance multimedia applications that deal with high quality video and audio. A summary of the key features of the TM1 architecture is given in Table VI-III. The experimentally measured cache misses are reported in Figure VI-9 using the simulator tools of the processor [36]. Moreover, the theoretical predictions from the proposed model can be seen in Figure VI-10. Similarly as before, the theoretical (predicted) ordering of the methods agrees with what is seen in the experiments, although not numerically. Finally, the relative performance of the three basic transform production schedules in this processor is seen in Table VI-IV, where "RC" represents the RCWT case III algorithm (taken as an average case), "LB" represents the LBWT and "BB" represents the block-based LWT coefficient production schedule; for each case, we display in the columns of Table VI-IV the scheme that achieved the highest throughput. In total, the LBWT had the best performance in 11 cases, the LWT in 11 cases, while the RCWT outperformed the other methods in only 2 cases. This validates the assumption that the best-performing algorithms in terms of cache misses appear to produce the highest throughput.

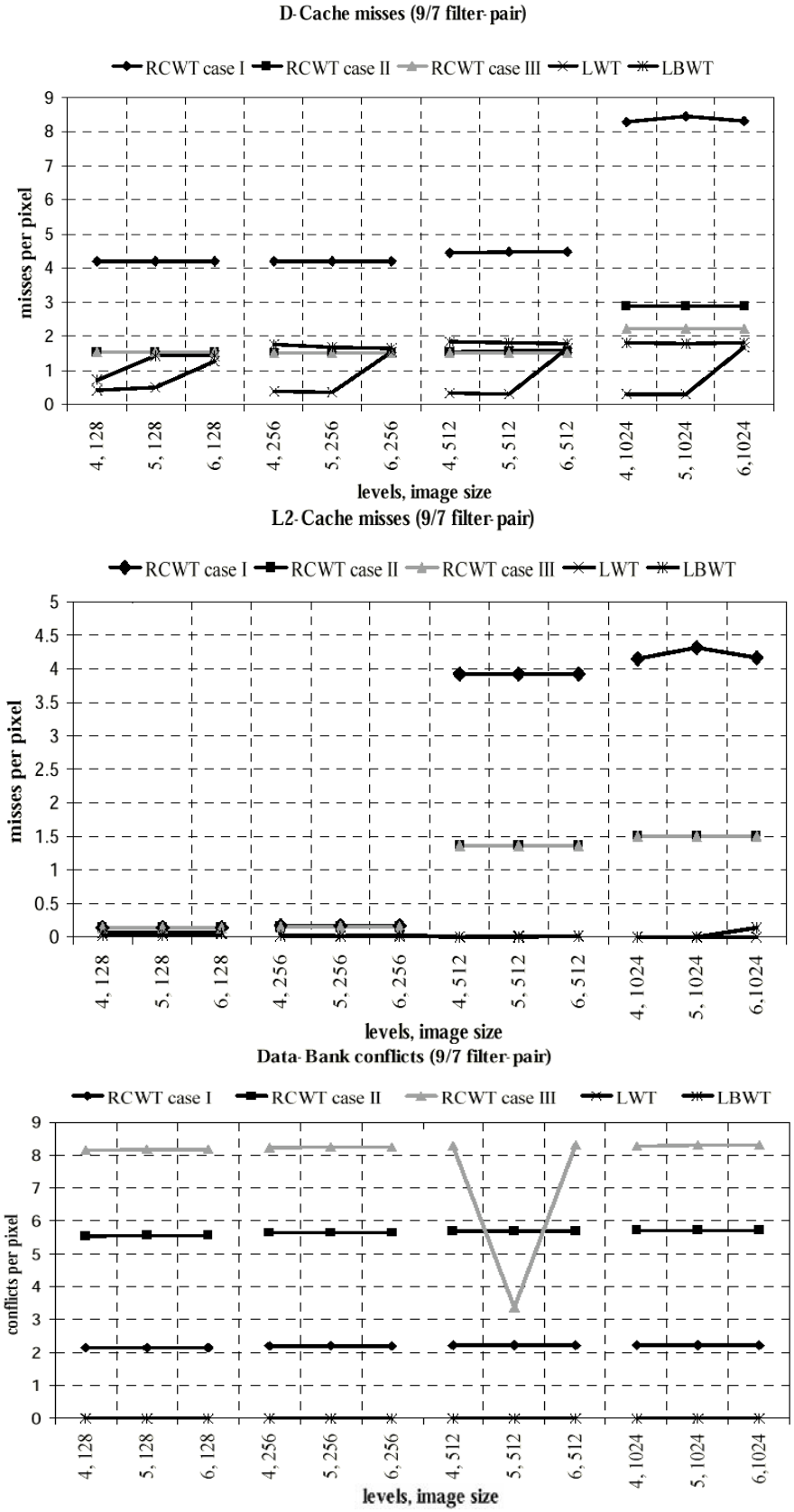


Figure VI-7. Experimental data-related penalties per pixel in the Intel Pentium platform. The 9/7 filter-bank was used.

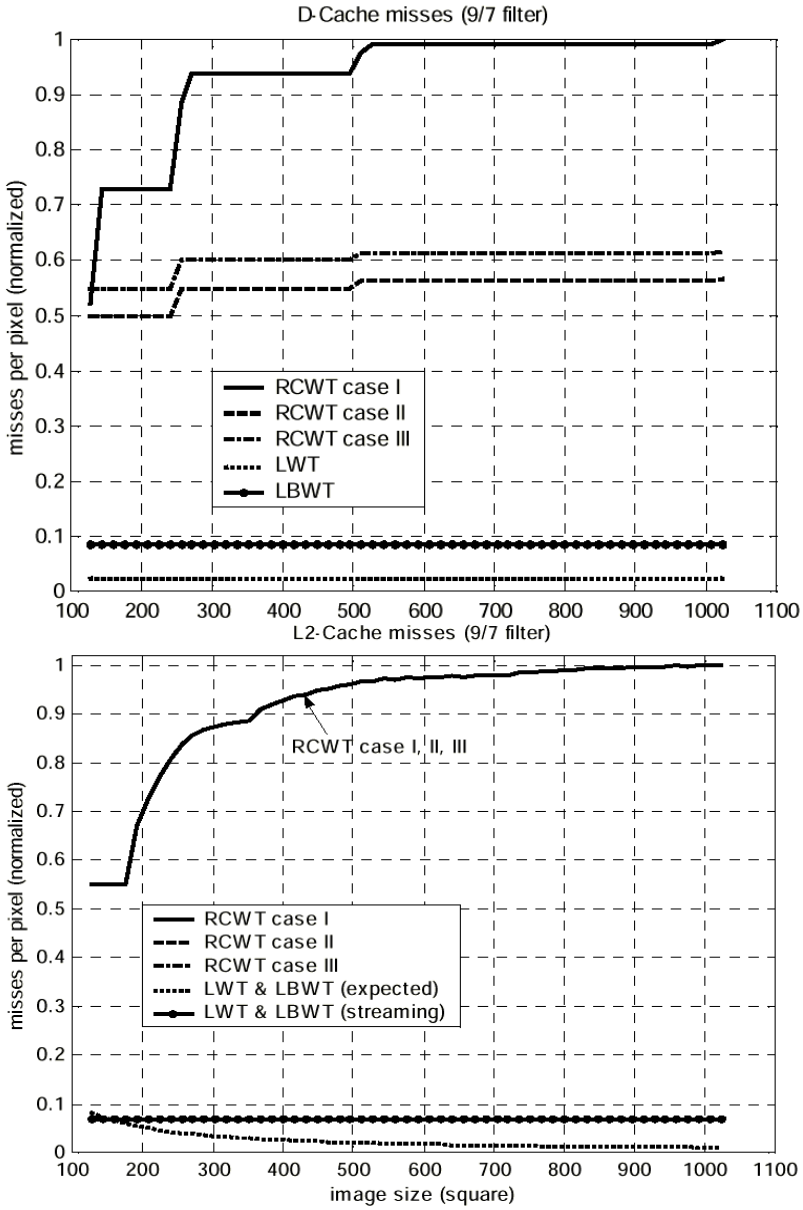


Figure VI-8. Theoretical calculations based on the fully-associative model for the application of the 9/7 filter-bank with the Intel-Pentium cache-characteristics (cache size and blocksize). For each image size, the normalized average miss ratio (misses per pixel) from the theoretical predictions for $L = 4,5,6$ is presented.

Image size	RCWT case I	RCWT case II	RCWT case III	LWT	LBWT
512x512	1.075	1.841	1.887	3.369	2.455
1024x1024	0.940	1.700	1.600	3.558	2.831
2560x2048	0.926	1.690	1.759	3.506	2.952

Table VI-II. Average throughput in Mpixels/sec for 4 to 6 levels (10 runs per method/image size/level).

Feature	Value	Feature	Value
Clock frequency	100 MHz	SW controlled part of data cache	Up to 8 kBytes
Maximum performance	4 giga operations per second	On chip instruction cache size	32 kBytes
Number of parallel data paths	27	Off-chip main memory size range	0.5 MBytes to 64 MBytes
Maximum number of data paths activated by the same instruction	5	External highway width and speed	32 bits - 400 MBytes/second
Register file size/bandwidth	128 32-bit registers with 15 read and 5 write ports	Packed instruction width	32 bits
On chip data cache size/bandwidth	16 kBytes/dual port	Decompressed instruction width	220 bits

Table VI-III. Major architecture features of TriMedia TM1.

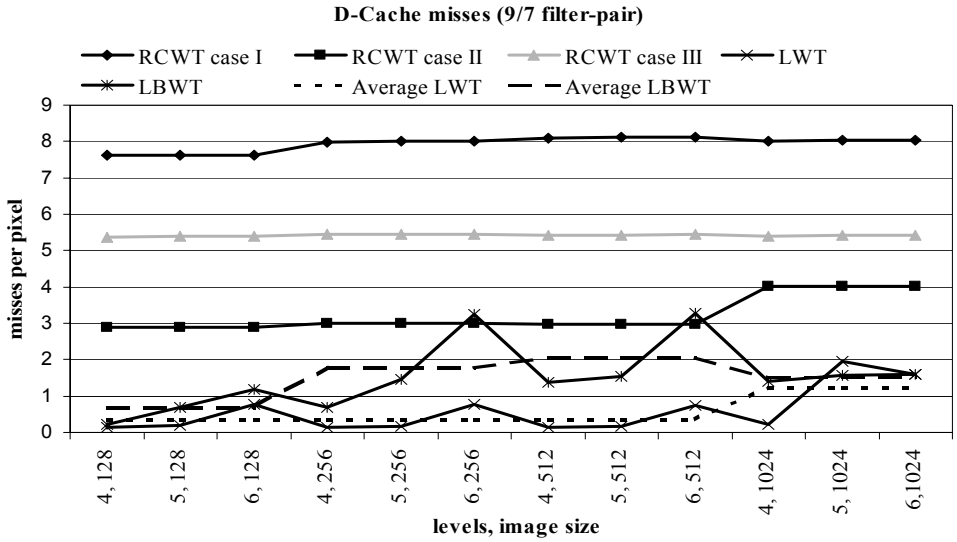


Figure VI-9. Experimental data-related cache misses in the Philips TriMedia TM1 with the 9/7 filter-bank.

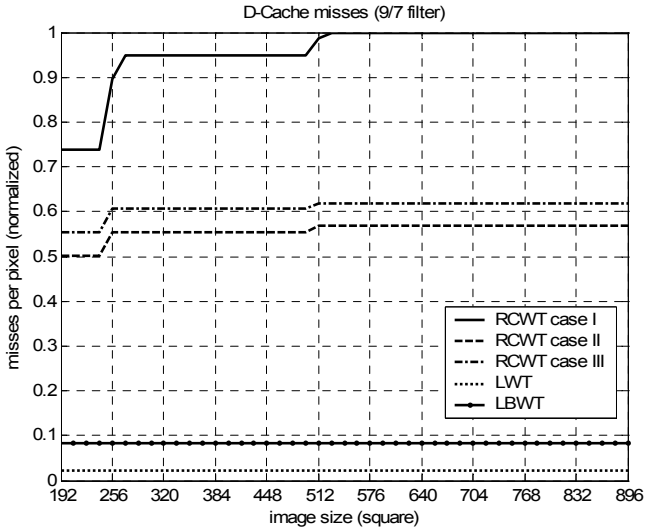


Figure VI-10. Theoretical calculations based on the fully-associative model for the application of the 9/7 filter-bank (average for $L = 4,5,6$) with the Philips TriMedia TM1 cache-characteristics (cache size and blocksize).

Image size	# of Decomposition levels	Best implementation for 5/3 filter-bank	Best implementation for 9/7 filter-bank
128×128	4	LB	LB
128×128	5	LB	LB
128×128	6	LB	LB
256×256	4	LB	LB
256×256	5	LB	BB
256×256	6	BB	BB
512×512	4	LB	RC
512×512	5	LB	BB
512×512	6	BB	BB
1024×1024	4	BB	RC
1024×1024	5	BB	BB
1024×1024	6	BB	BB

Table VI-IV. Best wavelet transform implementation option for all possible sets of image size, number of decompositions levels and filter-bank in terms of highest throughput. The results are inferred from the experiments on a VLIW DSP (Philips TriMedia TM1).

6.2.4 Concluding Summary for the Proposed Cache-modeling Approach

In this section, it has been shown that the data-related cache penalties are expected to dominate the execution of the two-dimensional multilevel DWT in typical programmable platforms since they are by far more frequent than the instruction-related cache penalties. In Table VI-V, we list the conclusions drawn from the presented experiments with respect to the software design and the cache efficiency of the various DWT-production methods. From the relative comparisons shown in Table VI-V, one can identify the various trade-offs for the software design of DWT systems. Apart from quantitative results, a theoretical framework was also proposed for the analytical estimation of the data-related misses, which can be utilized for high-level parametrical estimation of the DWT implementation-efficiency in a variety of cache architectures. This was verified by simulation results in a conflict-free cache model and partly by experimental results from (a) a superscalar processor with a two-level, set-associative cache; (b) a VLIW DSP with a single-level, set-associative cache.

Coefficient-Production Scheme		**Strictly Breadth-First (RCWT)**			**Roughly Depth-First (RDF)**	
Software Implementation		**Case I**	**Case II**	**Case III**	**LWT**	**LBWT**
C-code design complexity		low	low	low	high	high
Average assembly-code size in two systems: superscalar (Pentium) / VLIW (TriMedia)		72	72	72	100	86
Expected gain from platform-dependent optimizations		low	low	medium	high	high
Cache Behaviour: Simulator [33] % Average→ 5/3,9/7, 4-6 levels		**Case I**	**Case II**	**Case III**	**LWT**	**LBWT**
Level-one Cache	I-Cache accesses	64	53	61	100	74
	Instruction-related misses	12	12	12	100	65
	D-Cache accesses	100	78	93	94	82
	Data-related misses	100	63	66	4	6
prediction-accuracy of proposed model		very good	very good	very good	very good	very good
Level-two Cache	Instruction-related misses	54	54	54	100	65
	Data-related misses	28	100	20	1	10
prediction-accuracy of proposed model		very good	very good	very good	good	**constraint violation**
Cache Behaviour: Real Platform [34] % Average→ 5/3,9/7, 4-6 levels		**Case I**	**Case II**	**Case III**	**LWT**	**LBWT**
Level-1 Cache	Instruction-related misses	2	2	1	100	6
	Data-related misses	100	47	29	9	29
model validation – prediction of ordering of the methods with respect to the data-cache penalties		correct				
Level-2 Cache	Misses **Note**: joint I and D-Cache	100	40	38	0.1	1
Model validation – prediction of ordering of the methods with respect the cache misses		partially correct, not fully verified				

Table VI-V. Relative comparisons of the various methods of the two categories with respect to cache efficiency and design flexibility. Image sizes of 128×128 up to 1024×1024 are considered and the reported numbers are percentages that are normalized to the maximum measurement.

6.3 Complexity Modeling of Transform-based Video Decoders

The second modeling topic of this chapter concerns system complexity of video decoders. Since links are established in this section with MPEG-21 terminology [37] and we present an application example within a streaming scenario, we shall use the term "decoder" interchangeably with the terms "terminal" and "receiver". The work presented in this section was published in [12]; consequently, our presentation in this section follows [12].

Traditionally, multimedia compression and streaming have been studied within the rate-distortion theory framework that defines tradeoffs between information rate and distortion. Non-scalable bitstream switching, adaptive rate scaling, transcoding, scalable coding, distortion-optimized packet scheduling, network-adaptive source/channel coding, multiple description coding, etc. [38], have been developed to address real-time adaptation of multimedia content at the server or on-the-fly at a proxy, based on the network conditions. However, in most cases, these network-centric approaches neglect the user experience, as well as the capabilities and resource constraints of the receiver (e.g. display size, processing power, battery-life etc.).

Part 7 of the MPEG-21 standard, entitled Digital Item Adaptation, has defined a set of description tools for adapting multimedia based on the user characteristics, terminal capabilities, network characteristics and natural environment characteristics [39] [37]. We introduce a new complexity model that can assist the MPEG-21 DIA engine within a rate-distortion-complexity (R-D-C) framework to *explicitly* consider the resources available at the receiver. These can include hardware resources such as memory, processors, functional units, instruction and data memory bandwidths and limits on the power dissipation.

The focus of this section is not on system-specific complexity or power optimization since these topics have already been thoroughly studied for different multimedia codecs (e.g. see [2] [3] for software implementations of the AVC codec and [4] for Motion JPEG2000). Instead, the novelty of our approach is twofold. Firstly, in this section we introduce the concept of "virtual" decoding complexity and determine a general R-D-C framework that can be easily applied to a variety of existing and upcoming image and video compression schemes. Secondly, unlike the MPEG-4 Video Complexity Verifier (VCV) [5] [14] that determines whether the decoding resources fit within a certain profile (which corresponds to maximum allowable decoder resources), we consider *average* decoding complexities estimated using a model-based approach that considers the decoding algorithm implementation, as well as the transmission bit-rate and content characteristics. While worst-case bounds on complexity are extremely important for dedicated hardware implementations (e.g. application-specific integrated circuits), they are not very meaningful for next generation programmable architectures that can support multi-fidelity algorithms by allowing dynamic resource allocation. Examples of such architectures include energy-adjustable processors with dynamic frequency and voltage scaling and reconfigurable architectures.

In the proposed framework, the server generates a platform-independent model quantifying a set of Generic Complexity Metrics (GCMs) for decoding/streaming. Within the R-D-C framework the GCMs are linked with different operational R-D points.

The (pre-computed) GCMs are then mapped into Real Complexity Metrics (RCMs) that explicitly consider the specific terminal architectures and available resources. Consequently, within a multimedia streaming scenario, compressed bitstreams are correspondingly adapted at the server, proxy or receiver based on the determined RCMs.

This section is organized as follows. Subsection 6.3.1 presents a general methodology for constructing the R-D-C framework. We also introduce the GCM concept, which enables the definition of common (generic) complexity metrics across different classes of receivers. Subsections 6.3.2 and 6.3.3 illustrate how the proposed complexity model can be used for multimedia adaptation based on network characteristics and terminal capabilities. In Subsection 6.3.4, indicative simulation results and experiments using the proposed R-D-C framework are presented. Finally, Subsection 6.3.5 concludes the proposed approach and discusses future research topics.

6.3.1 Rate-Distortion-Complexity Framework

Recently, rate-distortion theory was extended to complexity-distortion theory and the complexity scalability of several simple algorithms (e.g. searching algorithms) has been investigated [7]. To enable on-the-fly adaptation within the MPEG-21 DIA framework, a practical R-D-C framework is required that relates the various operational R-D points (corresponding to different sub-streams) to their corresponding decoding complexity. Our focus within this subsection will be on modeling the generic complexity of multimedia decoding algorithms that does not consider the specific receiver features, capabilities and instantaneous resources.

6.3.1.1 Generic Modeling of Complexity

In order to represent at the server side different receiver architectures in a generic manner, we will deploy a concept that has been successful in the area of computer systems, namely, a virtual machine. We assume an abstract receiver referred to as Generic Reference Machine (GRM). This is representative of the computation and resource models of the receiver architectures in use. The GRM can be viewed as a basic pipelined RISC machine [8]. Assuming the GRM as the target receiver, we will develop an abstract complexity measure to quantify the decoding complexity of multimedia bitstreams.

The key idea of the proposed paradigm is that the same bitstream will require/involve different resources/complexities on various receivers. Given the number of factors that influence the complexity of the receiver, it is impractical to determine at the server side the *specific* (real) complexity for every possible receiver architecture. Consequently, we adopt a *generic complexity model* that captures the abstract/generic complexity metrics (GCMs) of the employed decoding or streaming algorithm depending on the content characteristics and transmission bit-rate. GCMs are derived by computing

the average number of times the different GRM-operations are executed. We consider a simple GRM that supports the following operations[1]:

$$op = \{add, multiply, assign\} \tag{6.24}$$

In DIA, the AdaptationQoS tool [39] defines adaptation units (AUs) as a group of video macroblocks, an entire video frame, a certain resolution of a frame, a group of pictures etc. The GCMs necessary for the decompression and streaming can be transmitted to the receiver at different granularities (e.g. for each functional unit, sub-unit etc.) and for varying-size AUs [39]. In general, finer granularity allows better control of the adaptation, but this may come at the expense of an increased communication and computational overhead.

Several illustrative examples on how R-D adaptation can be performed in the AdaptationQoS framework can be found in [39]. In this section, we develop models to be used in this framework for complexity adaptation. Although our experiments will be based on a scalable video coder, it is important to notice that the proposed models can apply more broadly, e.g., to conventional non-scalable video coding schemes [40] [1], as well as multiple description coding schemes. The difference is that, to create a set of alternative bitstreams resulting in different rate, distortion and complexity tradeoffs, multiple encodings/transcodings of the same content are typically required, which incurs higher computational load for the server.

We assume that each video group of pictures is partitioned into N independently-coded adaptation units. For example, to provide efficient resolution scalability, the maximum size of an AU is usually bounded to be an entire resolution level of a given intra- or inter-frame in the GOP. Let the set of AUs that correspond to the decoded resolution and frame rate of a GOP be denoted as $\{b_1, b_2, \ldots, b_N\}$. Each independent AU b_i, $1 \le i \le N$, is associated with a set of rate-distortion points $\left\{R_i^{j(i)}, D_i^{j(i)}\right\}$ with $j(i)$ indicating the corresponding bitstream-adaptation point. Within the AdaptationQoS tool of MPEG-21 DIA, the rate-distortion points are termed as *dependent IOPins* [41] since they depend on the number of permissible decoding parameters. This dependency stems from the fact that feasible adaptation points, which are referred to as *free IOPins* [41], can be derived at different spatio-temporal resolutions.

An optimization that aims to minimize the overall distortion in the GOP under a rate-constraint $R_{\max}$ can be stated as:

$$\{j_r^*(i), \lambda_r^*\}_{\forall b_i} = \arg\min_{j(i),\lambda} \left\{ \sum_{i=1}^{N} \left(D_i^{j(i)} + \lambda \cdot R_i^{j(i)} \right) \right\} : R_{GOP} \le R_{\max} \tag{6.25}$$

The Lagrangian multiplier λ must be adjusted until the value $\lambda = \lambda_r^*$ is found where the rate corresponding to the selected points $j_r^*(i)$ is (approximately) equal to $R_{\max}$.

---

[1] More sophisticated GRMs can be defined to facilitate better mapping of GCMs to architecture-dependent resources, e.g. different data- and memory types, word lengths, etc. However, this involves more complex R-D-C modeling, GCM to RCM mapping and bitstream adaptation mechanisms.

A Proposed Generic Complexity Model

To each possible bitstream adaptation point formulated by the set of solutions of (6.25), an associated complexity metric can be defined. We illustrate here how a generic complexity model can be built for video decoders employing motion compensation and transform coding. Similar models can also be built for alternative coding and/or streaming algorithms.

The proposed framework is inspired by the work of He and Mitra [40] on rate-control for image and video compression. Our approach models the expected decoder complexity with an AU-level granularity based on source (video-data) characteristics as well as implementation-related features. For a transform-based motion-compensated video coding scheme, the complexity of decoding is primarily determined by:

- the intra- and inter-frame decoding and inverse transform;
- the motion-compensation process[2].

For each AU b_i, we define the following complexity-function variables, corresponding to these decoder operations:

- the percentage of non-zero (decoded) transform coefficients, denoted by $p_T(i)$;
- the percentage (per pixel) of decoded motion vectors out of the maximum number of possible motion vectors (hypotheses) provided by the utilized motion model, denoted by $p_M(i)$.

The purpose of these variables is not to encapsulate the input source characteristics but rather to represent the *underlying mechanism* based on which the input source characteristics can vary. It was already shown that a model that depends on these variables can be generic and yet efficiently predict the variations of the compressed source data, e.g., see the model of [40], as well as the use of non-zero transform coefficients in the VCV buffer model of [5].

Our motivation behind the choice of p_T comes from the fact that, in the decoding process, particular processing operations are activated only when a significant (non-zero) coefficient is found. For example, during the decoding process of wavelet-based embedded codecs, for each processed bitplane, the transform areas with insignificant (zero) coefficients are either skipped or processed in a uniform way by performing a fixed operation such as decoding a non-significance symbol. On the other hand, more complex operations such as sign decoding, list management and coefficient refinement operations occur at the areas where coefficients are found to be significant. We refer to the JPEG-2000 algorithm [42] as a practical test case. Similar observations hold for the decoding of other wavelet-based and DCT based image-coding schemes.

---

[2] Although the decoding of the motion-vector information also contributes in the complexity profile of the decoding process, with the exception of extremely low bitrates, the effect of this process in the complexity of practical video decoders can be considered to be negligible.

Concerning motion information, in state-of-the-art video coding systems motion is adaptively estimated using a system with variable block-size multihypothesis prediction [43], as explained in Chapter III and Chapter IV. An example of such a system is pictorially demonstrated in Figure VI-11. Depending on the motion characteristics, the motion estimation algorithm partitions each macroblock adaptively and a variable number of motion vectors are assigned to the pixels belonging to each macroblock area. However, due to complexity and bandwidth limitations, there always exists a maximum number of motion vectors that can be associated with each pixel in a given frame. Consequently, at the pixel-level, the motion vectors associated with each error frame represent a percentage p_M of the maximum number of the vectors that represent the maximum-density motion field. As a result, the number of arithmetic operations and memory accesses during motion compensation depends on p_M.

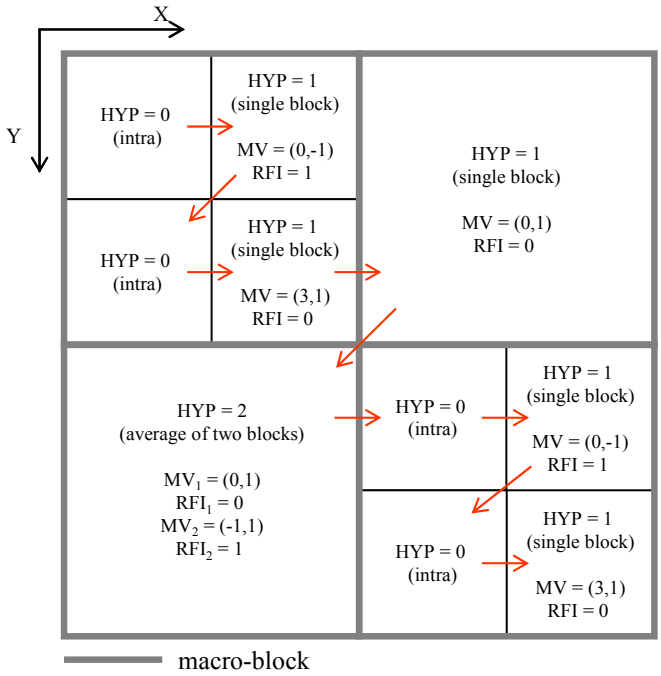


Figure VI-11. Motion estimation with variable block-size multihypothesis prediction. Notations: HYP is the number of motion-vectors (hypotheses) used for the prediction of each subblock ($HYP = 0$ denotes intra-prediction); MV_h, $1 \le h \le HYP$, is the h-th motion vector; RFI_h represents an index to the fractional-pixel interpolated position for the h-th motion vector.

For each AU b_i, the proposed complexity functions can now be defined as $W^{op}(p_T(i), p_M(i))$, with the superscript op given in (6.24). This estimation of $W^{op}(p_T(i), p_M(i))$ is based on a decomposition in a set of basis functions that characterize the complexity of the different parts of the motion-compensated wavelet decoding system. These basis functions can be considered as high-level descriptors of the complexity characteristics of each part of the entire decoding system. In particular, we introduce the texture-related complexity basis-function, $\mathcal{T}(p_T(i))$, and three motion-related complexity basis-functions, $M_F(p_M(i))$, $M_S(p_M(i))$ and $M_Z(p_M(i))$, which relate to the operation

modes associated with the motion-compensation process. Their precise definitions will be given in Subsection B.

For a given AU b_i, the complexity functions $W^{op}(p_T(i), p_M(i))$ are formulated as:

$$\begin{aligned} W^{op}(p_T(i),p_M(i)) = & A^{op}(p_T(i)) \cdot \mathcal{T}(p_T(i)) + B^{op}(p_M(i)) \cdot \mathcal{M}_{\mathcal{F}}(p_M(i)) \\ & + C^{op}(p_M(i)) \cdot \mathcal{M}_{\mathcal{S}}(p_M(i)) \\ & + D^{op}(p_M(i)) \cdot \mathcal{M}_{\mathcal{Z}}(p_M(i)) \end{aligned} \tag{6.26}$$

where $A^{op}(p_T(i))$ is the texture-related complexity-decomposition coefficient, and $B^{op}(p_M(i))$, $C^{op}(p_M(i))$, $D^{op}(p_M(i))$ are the motion-related complexity-decomposition coefficients. For each AU of the input video, the complexity basis functions provide a high-level estimation of the data-dependent computational resources that are required for the processing of its associated bitstream. They are varying solely based on the source characteristics. On the other hand, the texture- and motion-related complexity-decomposition coefficients are dependent on the decoding algorithm and its implementation architecture. Once the algorithm (and its implementation) is fixed for a given transmission scenario they can be determined offline by using a number of training sequences and profiling results from the specific decoder implementation architecture. Finally, if intra-frame coding algorithms are considered (e.g. Motion JPEG), the motion-related complexity decomposition coefficients and basis functions are annulated and hence, $\forall i : p_M(i) \equiv 0$.

Based on (6.26), the complexity-decomposition functions for a GOP consisting of N AUs is given by:

$$W^{op}_{GOP}(p_T,p_M) = \sum_{i=1}^{N} W^{op}(p_T(i),p_M(i)) + E^{op}(\overline{p}_T, \overline{p}_M) \tag{6.27}$$

where $E^{op}(\overline{p}_T, \overline{p}_M)$ can be interpreted as the basic decoding complexity per GOP, which varies based on its average texture and motion content, represented by:

$$\overline{p}_T = \frac{1}{N}\sum_{i=1}^{N} p_T(i), \overline{p}_M = \frac{1}{N}\sum_{i=1}^{N} p_M(i).$$

In typical scenarios, this factor represents the cumulative effect in complexity of decoding operations that are almost data-independent, such as the inverse transform (inverse DCT or inverse DWT). Conceptually, equation (6.27) is analogous to a frequency-decomposition of a signal. In this case, $E^{op}(\overline{p}_T, \overline{p}_M)$ can be seen as the "DC" component of the decomposition.

B Basis-functions for the Complexity Decomposition

The texture-related complexity basis-function $\mathcal{T}(p_T(i))$ of an AU b_i is the function associated with the number of operations needed to decode a transform representation with a percentage of $p_T(i)$ non-zero coefficients. Following [40], we define $\mathcal{T}(p_T(i))$ as:

$$\mathcal{T}(p_T(i)) = \frac{1}{X_l \cdot Y_l}\sum_{k=0, c_k(p_T(i))\neq 0}^{X_l \cdot Y_l} (\lfloor \log_2 |c_k(p_T(i))| \rfloor + 2) \tag{6.28}$$

where $c_k(p_T(i))$ represents the decoded value of the k-th transform coefficient of an AU that reconstructs $X_l \times Y_l$ pixels of the input video frame. When decoding stops at resolution l (with $l = 0$ being the original resolution), we have:

$$X_l \cdot Y_l = 4^{-l} X_{or} \cdot Y_{or}$$

where the original frame has $X_{or} \times Y_{or}$ pixels. Equation (6.28) shows that the decoded transform-coefficient values c_k are a function of $p_T(i)$. In embedded coding, the coefficient values c_k of each AU b_i are a function of the number of decoded (integer or fractional) bitplanes $q(i)$. In typical transform-coding schemes, $p_T(i)$ is monotonically-increasing with $q(i)$. As a result, there is a one-to-one mapping between $p_T(i)$ and $q(i)$. Hence, c_k is also a function of $p_T(i)$. We note here that in the generic case of non-embedded coding, e.g. quantization of DCT coefficients, equation (6.28) can be applied as well [40].

Equation (6.28) determines the sum of the magnitudes and sign information of the non-zero transform coefficients. This is representative for the complexity variations associated with classical transform-based coding schemes.

As explained before, the complexity of motion compensation for each AU b_i is expected to be proportional to the percentage $p_M(i)$ of the existing motion vectors per pixel. This is modelled by $M_F(p_M(i))$, which is defined as:

$$M_F(p_M(i)) = v \cdot p_M(i) \cdot 4^{-l} \tag{6.29}$$

where v represents the maximum number of motion vectors (hypotheses) associated with each of the $X_l \times Y_l$ pixels of the decoded AU b_i (see Figure VI-11).

The complexity of motion compensation is also related to the number of motion vectors associated with fractional-pixel positions. Fractional-pixel accurate motion compensation typically involves interpolation operations with a filter kernel approximating the sinc function, hence requiring additional processing operations. This is modelled by the $M_S(p_M(i))$ function, defined as:

$$M_S(p_M(i)) = M_F(p_M(i)) \cdot s(p_M(i)) \tag{6.30}$$

where $s(p_M(i))$ is the percentage of existing vectors associated with a fractional-pixel position. For each AU b_i, we define this percentage as a function of $p_M(i)$ since, in our experimentation over a large set of sequences encoded using different settings, it was found that, for block-based motion estimation, this percentage is monotonically increasing with $p_M(i)$. Finally, the additional complexity of advanced techniques using overlapped block motion compensation or deblocking is modelled by the $M_Z(p_M(i))$ function. The application of such techniques strongly relates to the local differences between neighbouring motion vectors. Specifically, the definition of $M_Z(p_M(i))$ is based on the following steps:

- The motion vectors are organized in an one-dimensional array. All the macroblocks are scanned in a raster order, while motion vectors inside the macroblock are parsed in a zigzag scan order, as shown with the arrows of Figure VI-11.

- For every motion vector $v_m = (x_m, y_m)$ that points to reference frame r_m, where m is the index in the 1-D array, $(1 \le m \le M_F(p_M(i)) \cdot X_l \cdot Y_l)$ calculate:

$$d_m = (|x_m - x_{m-1}|, |y_m - y_{m-1}|)$$

  where we preset $v_0 = (0,0)$.

- Let $t_m = \{1: \text{ if } \|d_m\| \ge Q \text{ or } r_m \ne r_{m-1};\ 0: \text{ otherwise}\}$, where Q represents a pre-determined threshold.

  We define $M_Z(p_M(i))$ as:

$$M_Z(p_M(i)) = \sum_{m=1}^{M_F(p_M(i)) \cdot X_l \cdot Y_l} t_m \tag{6.31}$$

The basis functions of (6.28) – (6.31) can be calculated dynamically during encoding for each AU b_i. In order to avoid any increase in the encoding computational complexity, statistical methods could be adopted to calculate these basis functions based on their observed properties, following techniques similar to the ones used in [40]. In practice, the increase in encoder complexity was found to be negligible in comparison to the motion estimation complexity and the direct calculation of the basis functions provides a higher modeling accuracy. Having pre-determined values for the complexity-decomposition coefficients, the generated basis-function values for each adaptation point are used in (6.26) to estimate the GCMs.

6.3.2 Estimation of the Complexity Decomposition Coefficients

In this section, we outline the practical method we used to estimate the values of the complexity-decomposition coefficients of equation (6.26) for each op of (6.24). This estimation is based on a training algorithm. For this purpose, we chose seven CIF video sequences ("Coastguard", "Foreman", "Stefan", "Tempete", "Paris", "Football", "News" – first 48 frames of each, which correspond to the temporal dependencies required for the MCTF operation of the first GOP in the utilized codec – SDMCTF codec used in Chapter IV with the "best mode" settings). These sequences were chosen as representative of a large variety of content. For each AU b_i, $1 \le i \le N$, of each sequence, the values of the basis functions $\mathcal{T}(p_T(i))$, $\mathcal{M}_{\mathcal{F}}(p_M(i))$, $\mathcal{M}_{\mathcal{S}}(p_M(i))$, $\mathcal{M}_{\mathcal{Z}}(p_M(i))$ were calculated for a representative set of values of $p_T(i)$, $p_M(i)$. In particular, we chose two sets of values for $p_T(i)$, $p_M(i)$, denoted by $p_T^e = \{0.01, 0.02, 0.03, 0.04, 0.05\}$ and $p_M^e = \{0.001, 0.002, 0.003, 0.004\}$, respectively. The range of these sets was selected so that the corresponding bitrate for the texture and motion-vector information fits in the practical bitrate regimes of CIF video sequences. Naturally, the use of more sequences for the training stage and a high granularity in the regime of practical values of $p_T(i)$, $p_M(i)$ helps in increasing the accuracy of the complexity-modeling process. In practice, the estimation of the values of the complexity-decomposition coefficients was performed by the following process.

The encoding algorithm was executed multiple times for each sequence, each time enforcing the number of motion vectors that corresponded to each AU b_i to be equal to a pre-determined value

that corresponds to the specific value that is of interest, i.e. for every i we select one p_M^e and set $p_M(i) = p_M^e$.

The bitstream extraction and decoding occurred by selectively decoding a number of bitplanes for the texture information of each AU b_i that corresponds to the specific value that is of interest, i.e. for every i we select one p_T^e and set $p_T(i) = p_T^e$.

In this way, for each possible pair $\{p_T^e, p_M^e\}$ from the set of values that spans the bitrates of interest, based on (6.26), (6.27), we have:

$$W_{GOP}^{op}(p_T^e, p_M^e) = N \cdot W^{op}(p_T^e, p_M^e) + E^{op}(p_T^e, p_M^e) \tag{6.32}$$

The actual values of $W_{GOP}^{op}(p_T^e, p_M^e)$ were (off-line) measured by profiling the software implementation of the video decoder for the set of operations defined by (6.24). Hence, the experimentally-derived GCMs $W_{GOP}^{op}(p_T^e, p_M^e)$ depend on the software implementation of the decoder.

Based on the profiling results and on the complexity basis functions (which were calculated using equations (6.28) – (6.31) with $p_T(i) = p_T^e$ and $p_M(i) = p_M^e$ for every i), the values of $A^{op}(p_T^e)$, $B^{op}(p_M^e)$, $C^{op}(p_M^e)$, $D^{op}(p_M^e)$ and $E^{op}(p_T^e, p_M^e)$ of equation (6.27) were determined using multiple linear regression. In particular, for each pair of permissible values $\{p_T^e, p_M^e\}$ and for each op defined in (6.24), we establish the linear system:

$$\mathbf{W}_{GOP}^{op} = \mathbf{T}^{op} \cdot \mathbf{C}^{op} + \mathbf{E}^{op} \tag{6.33}$$

where $\mathbf{W}_{GOP}^{op} = \left[W_{GOP,seq(1)}^{op}(p_T^e, p_M^e) \quad \ldots \quad W_{GOP,seq(7)}^{op}(p_T^e, p_M^e) \right]^T$ is the 7×1 vector containing the total number of op operations for each sequence $seq(j)$, $1 \le j \le 7$, of the training set, measured by profiling the decoder software; $\mathbf{T}^{op}$ represents the 7×4 matrix of the estimated basis functions $\mathcal{T}_{seq(j)}(p_T^e)$, $\mathcal{M}_{\mathcal{F},seq(j)}(p_M^e)$, $\mathcal{M}_{\mathcal{S},seq(j)}(p_M^e)$, $\mathcal{M}_{\mathcal{Z},seq(j)}(p_M^e)$ for the complexity decomposition for all the sequences $seq(j)$ and is defined as:

$$\mathbf{T}^{op} = \begin{bmatrix} \mathcal{T}_{seq(1)}(p_T^e) & \mathcal{M}_{\mathcal{F},seq(1)}(p_M^e) & \mathcal{M}_{\mathcal{S},seq(1)}(p_M^e) & \mathcal{M}_{\mathcal{Z},seq(1)}(p_M^e) \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{T}_{seq(7)}(p_T^e) & \mathcal{M}_{\mathcal{F},seq(7)}(p_M^e) & \mathcal{M}_{\mathcal{S},seq(7)}(p_M^e) & \mathcal{M}_{\mathcal{Z},seq(7)}(p_M^e) \end{bmatrix} \tag{6.34}$$

for each op and $\{p_T^e, p_M^e\}$; $\mathbf{C}_{GOP}^{op} = \left[A^{op}(p_T^e) \quad B^{op}(p_M^e) \quad C^{op}(p_M^e) \quad D^{op}(p_M^e) \right]^T$ is the 4×1 vector containing the complexity decomposition coefficients that we want to estimate, which depend on p_T^e or p_M^e as explained before; finally, $\mathbf{E}^{op} = \left[E^{op}(p_T^e, p_M^e) \quad \ldots \quad E^{op}(p_T^e, p_M^e) \right]^T$ is the 7×1 vector containing the "mean" (DC) complexity decomposition coefficient $E^{op}(p_T^e, p_M^e)$. In order to provide the best estimation of $\mathbf{C}_{GOP}^{op}$ and $\mathbf{E}^{op}$ based on multiple linear regression [27], we have:

$$\begin{bmatrix} \mathbf{C}_{GOP}^{op}(p_T^e, p_M^e) \\ E^{op}(p_T^e, p_M^e) \end{bmatrix} = \left(\left[\mathbf{T}^{op} \quad \mathbf{1} \right]^T \left[\mathbf{T}^{op} \quad \mathbf{1} \right] \right)^{-1} \left[\mathbf{T}^{op} \quad \mathbf{1} \right]^T \mathbf{W}_{GOP}^{op} \tag{6.35}$$

where $\mathbf{1} = \left[1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \right]^T$ is a column vector with unitary entries that is used to incorporate $\mathbf{E}^{op}$ in the estimation of (6.35).

Once computed for each op and each $\{p_T^e, p_M^e\}$, the values of the complexity decomposition coefficients are kept in lookup tables for the actual GCM estimation during the bitstream adaptation process.

To summarize, the previously-described off-line training stage only involves the adjustment of the number of motion vectors and the number of significant coefficients of each frame to predetermined values that correspond to each pair $\{p_T^e, p_M^e\}$. The first task is typically accomplished by iteratively adjusting the Lagrangian-based motion-vector pruning scheme of the encoder (see the advance motion estimation algorithm of Chapter IV for details). The latter is easily performed at the bitstream extraction stage.

We note here that the profiling of the decoder software implementation is generic and can typically be performed using automated analysis tools (e.g. [44] for the memory-related operations). This is an important aspect since, in this way, if a different decoder software-implementation is to be used, the off-line training stage can be re-performed with the new software and different sets of complexity decomposition coefficients can be derived and used in the model.

Finally, the usage of the proposed model during the actual bitstream adaptation process in a streaming server is straightforward. For each AU b_i, depending on the $\{p_T(i), p_M(i)\}$, the complexity-decomposition functions are calculated (equations (6.28) – (6.31)). Then, the corresponding complexity decomposition coefficients are found by using the lookup tables and rounding the values $\{p_T(i), p_M(i)\}$ to the closest lookup-table entry. At this stage, using equations (6.26) and (6.27) for the current GOP, the model-based estimation of the GCMs takes place.

6.3.3 A Practical Scenario For Complexity-Driven Adaptation

In this section, we show how the proposed R-D-C framework allows for dynamic adaptation to network characteristics and terminal capabilities. As mentioned in the introduction, the mapping of GCMs into RCMs can be performed either at the server or the receiver. Figure VI-12 pictorially represents an example of the proposed R-D-C bitstream adaptation.

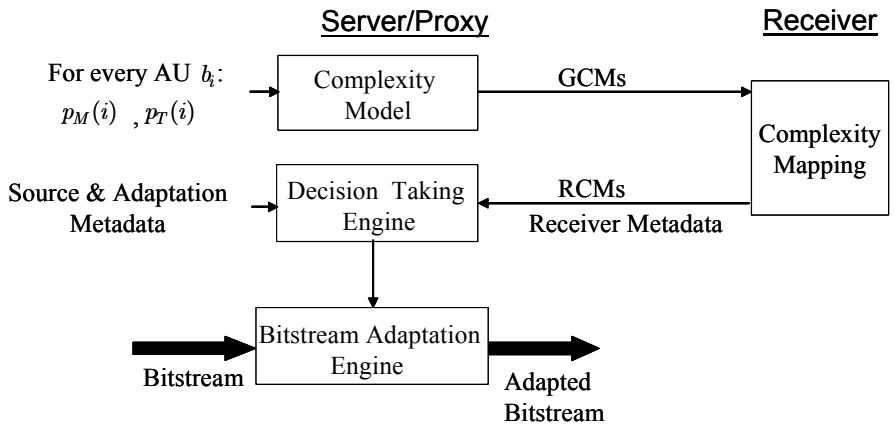


Figure VI-12. An example of R-D-C driven adaptation.

The GCMs $\{W_{op}\}_i^{j(i)}$ are determined at encoding time for each AU, based on the values of $p_M(i)$ and $p_T(i)$. The GCMs are then transmitted to a *complexity mapper*[3], which is aimed at translating the GCMs into RCMs by taking into account a number of factors specific to the receiver architecture and processing platform. These factors include the instruction set of the underlying processor, the data-types supported within the processor, the number of functional units (including hardwired co-processors), the memory hierarchy and input-output processors, the available energy resources and special resources such as FIFOs or other buffers [4]. To determine the available resources, a *resource monitor* can be implemented at the receiver that maintains the current utilization of various resources especially memory, energy etc. The complexity mapper and resource profiler can be implemented in a variety of ways. The reader is referred to [9] [10] [11] for some illustrative examples for this topic. Based on the generated RCMs that entail the number of different operations that can be supported at a specific time, the receiver metadata is generated. Within the DIA standard [39], the generation of such metadata will utilize UED tools such as the CPUBenchmark, PowerCharacteristics, DisplayCapabilities, etc. With such descriptions, complexity adaptation at the server/proxy is enabled. Moreover, other metadata (e.g. MPEG-7 content descriptors) can also be generated to assist the Decision Taking Engine (Figure VI-12) that derives the adapted multimedia bitstream that will be streamed to the receiver.

As a result, for each AU b_i and at each possible adaptation point $j(i)$, the receiver formulates the RCM-based metric as:

$$C_i^{j(i)} = \sum\nolimits_{\forall op} \mathcal{L}\left(\{W^{op}\}_i^{j(i)}\right) \tag{6.36}$$

where $\mathcal{L}(\cdot)$ represents the mapping operation from GCMs to RCMs, which, in the framework of MPEG-21 DIA, involves descriptors from the AdaptationQoS tool. Notice that the RCMs of (6.36) are additive for all the AUs of each GOP, i.e. $C_{GOP} = \sum_{i=1}^{N} C_i^{j(i)}$. Let $C_{\max}$ denote the upper bound for the complexity of the receiver in terms of RCMs per GOP. The adaptation points $j(i)$ for each GOP can be formulated as the parameters to a complexity-constrained optimization problem:

$$\{j_c^*(i), \lambda_c^*\}_{\forall b_i} = \arg\min_{j(i),\lambda} \left\{ \sum_{i=1}^{N} \left(D_i^{j(i)} + \lambda \cdot C_i^{j(i)} \right) \right\} : C_{GOP} \le C_{\max} \tag{6.37}$$

where the Lagrangian multiplier λ must be adjusted until the value $\lambda = \lambda_c^*$ is found where the RCM-based complexity corresponding to the selected bitstream adaptation points $j_c^*(i)$ is equal to the RCM constraint $C_{\max}$. The optimization of (6.37) determines the adaptation points for each AU incurring the minimum distortion under the pre-defined constraint $C_{\max}$.

In general, the solutions of (6.25) and (6.37) (rate and complexity-constrained optimizations, respectively) will produce *different* truncations for each resolution and frame-rate. However, for

[3] As mentioned in the introduction, the complexity mapper can be located at the receiver (as depicted in Figure 2), or at the server/proxy.

practical applications of video streaming, both the rate and complexity constraints must be satisfied, i.e. the solution $\{j^*(i),\lambda^*\}_{\forall b_i}$ must satisfy $C_{GOP} \le C_{\max}$ and $R_{GOP} \le R_{\max}$. The optimization problem is then expressed as:

$$\{j^*(i),\lambda_r^*,\lambda_c^*\}_{\forall b_i} = \arg\min_{j(i),\lambda_r,\lambda_c}\left\{\sum_{i=1}^{N}\left(D_i^{j(i)} + \lambda_r \cdot R_i^{j(i)} + \lambda_c \cdot C_i^{j(i)}\right)\right\} : R_{GOP} \le R_{\max} \text{ and } C_{GOP} \le C_{\max} \tag{6.38}$$

Since the solution of (6.38) involves searching among various values of λ_r,λ_c, the optimization problem can be efficiently solved by first excluding all the truncation points $j_n(i)$ that do not produce monotonically-decreasing slope values (for each b_i). Effectively, a point $j_n(i)$ is excluded if $\lambda_r^{n+1} > \lambda_r^n$ or $\lambda_c^{n+1} > \lambda_c^n$, with $\lambda_r^{n+1},\lambda_c^{n+1}$ defined as:

$$\lambda_r^{n+1} = \frac{D_i^{j_n(i)} - D_i^{j_{n+1}(i)}}{R_i^{j_{n+1}(i)} - R_i^{j_n(i)}},\ \lambda_c^{n+1} = \frac{D_i^{j_n(i)} - D_i^{j_{n+1}(i)}}{C_i^{j_{n+1}(i)} - C_i^{j_n(i)}} \tag{6.39}$$

with $\left\{R_i^{j_{n+1}(i)}, D_i^{j_{n+1}(i)}, C_i^{j_{n+1}(i)}\right\}$ the rate, distortion and complexity estimates of the next valid truncation point ($j_{n+1}(i)$) and λ_r^n, λ_r^n the slope values of $j_n(i)$, defined analogous to (6.39).

Proposed R-D-C based adaptation <u>for each GOP</u> :

Initialization

- Let $adapt = \varnothing$ represent the set of R-D-C optimized adaptation points

Metadata Generation and Transmission

- Establish the set of AUs b_i, $1 \le i \le N$ of the GOP
- Establish the set of bitstream adaptation points and their corresponding R-D-C metadata:

  $\forall b_i : j(i), \left\{R_i^{j(i)}, D_i^{j(i)}, \{W^{op}\}_i^{j(i)}\right\}$

Adaptation Mechanism

- For the current GOP:

  Determine $R_{\max}$ (for eq. (6.25)) based on the network monitor

  Determine $C_{\max}$ (for eq. (6.37)) based on the resource monitor
- For each AU b_i, $1 \le i \le N$, of the GOP:

  Establish $C_i^{j(i)}$ from eq. (6.36) (GCMs-to-RCMs)

  Invalidate all points $j_n(i)$ for which: $\lambda_r^{n+1} > \lambda_r^n$ or $\lambda_c^{n+1} > \lambda_c^n$
- Given $R_{\max}$, find $\{j_r^*(i),\lambda_r^*\}_{\forall b_i}$ from eq. (6.25) (which corresponds to (6.38) with $\lambda_c = 0$)
- Given $C_{\max}$ and $\{j_r^*(i),\lambda_r^*\}_{\forall b_i}$, find $\{j^*(i),\lambda_r^*,\lambda_c^*\}_{\forall b_i}$ from (6.38) with the additional constraint:

  $\forall i : \frac{D_i^{j_p(i)} - D_i^{j^*(i)}}{R_i^{j^*(i)} - R_i^{j_p(i)}} \ge \lambda_r^*$, where $\left\{R_i^{j_p(i)}, D_i^{j_p(i)}\right\}$ correspond to the previous (valid) truncation point $j_p(i)$
- For each AU b_i, $1 \le i \le N$, of the GOP:

  set $adapt = adapt \cup \{j^*(i)\}$

Figure VI-13. Pseudo-code for the optimized R-D-C adaptation.

An algorithm to determine the adaptation point for the AUs of each GOP under joint rate and complexity constraints $R_{\max}$ and $C_{\max}$ is given in Figure VI-13. Unlike previous multimedia streaming approaches that considered only the network limitations, the proposed joint R-D-C optimization may result in the transmission of a bitstream at a lower rate than that provided by the channel, because higher rates streams could not be appropriately decoded by the receiver, thereby unnecessary wasting channel bandwidth.

One possible solution for implementing the above R-D-C adaptation in a streaming scenario is to consider the file format of the media. In [12] [45], an abstraction layer referred to as "multi-track hinting" is introduced, which is an extension of the hinting mechanism that is part of the MP4 file format specification [46]. Multi-track hinting allows structuring compressed video into multiple prioritized sub-streams that can be independently transmitted through multiple channels, as illustrated in Figure VI-14.

This extension to conventional hinting mechanisms provide the flexibility necessary for network- and complexity-adaptive multimedia streaming by adjusting the number and type of transmitted (sub-) streams. Using the multi-track hinting method, each bitstream remains unchanged and it is stored once, but it is virtually divided into multiple sub-streams having different corresponding rates, distortions and complexities [45]. This means that, at the AU-level, the adaptation points $j(i)$ and their corresponding R-D-C metrics $\left\{R_i^{j(i)}, D_i^{j(i)}, \{W_{op}\}_i^{j(i)}\right\}$ are predetermined at the hinting stage, i.e., post encoding, but prior to the actual transmission.

Figure VI-14. The multi-track R-D-C hinting file format.

6.3.4 Experimental Validations

We first present some results that demonstrate the feasibility of the proposed model in predicting generic complexity metrics for a given decoding system. In addition, an example of the R-D-C based adaptation is presented with a prototyped multimedia streaming system.

6.3.4.1 Validation of the Proposed GCM Estimation Model

The proposed model for the GCM estimation of video decoding was tested using the spatial-domain motion-compensated temporal filtering codec used in the experiments of Chapter IV. As explained therein, the chosen decoder implementation combines a number of advanced features such as adaptive temporal decomposition with long temporal filters, variable block-size multihypothesis prediction and update steps, sub-pixel accurate motion compensation and QuadTree Limited coding, which was described in Chapter III. A number of video sequences not belonging to the training set were encoded. From the compressed bitstream of each sequence, several substreams were extracted that correspond to different bitrates. We present results relating to the estimation of the computational complexity, which is quantified with the number of integer additions and multiplications per pixel. A natural extension of the presented results involves estimation of the assignment operations (memory accesses). Some preliminary results concerning the profiling of memory accesses have been presented in [47]. We find that the effect of the memory accesses in the total decoding complexity relates strongly to a certain memory partitioning into local (internal) and non-local (external) accesses. This requires a specific assignment of the memory footprint of the MCTF-based decoder to a certain category of architectures and represents a direction for future research.

Sequence	**Hall Monitor**			**Ice**		
Bitrate/ frame-rate	32kbps 15Hz	64kbps 30Hz	256kbps 30Hz	256kbps 15Hz	512kbps 30Hz	2048kbps 30Hz
$op = multiply$	6 (6)	14 (14)	16 (15)	48 (45)	119 (125)	130 (128)
$op = add$	24 (30)	55 (54)	60 (65)	176 (214)	400 (394)	428 (436)

Table VI-VI. Multiplications and additions per pixel of each GOP of two typical sequences at three different adaptation points. The numbers in parentheses present the prediction based on the proposed GCM model approach. The arithmetic complexity of the inverse DWT is not included, since, under the chosen decoder implementation, it consists of a fixed overhead in terms of additions and multiplications per pixel regardless of the compressed bitstream.

Typical experimental results for the derived GCMs per GOP (software profiling) as well as the model-based GCMs (equations (6.26) – (6.31) with the estimated complexity-decomposition coefficients) are illustrated in Table VI-VI. Notice that the decoder arithmetic complexity can increase by a factor of three for an eighth-fold increase in decoding bitrate. Table VI-VII presents the average model

estimation-error (per GOP) for different sequences. For each sequence, the average error of the model prediction over a set of adaptation points is presented. These points represent an eighth-fold increase over the lowest (base-layer) bitrate. In total, the average error between the experimental and model-based GCMs was found to be approximately 10%. Note that this estimation accuracy is achieved without focusing on the specific MCTF coding algorithm or the texture-coding features of the codec. Moreover the software profiling was performed in a generic manner, without isolating specific implementation features.

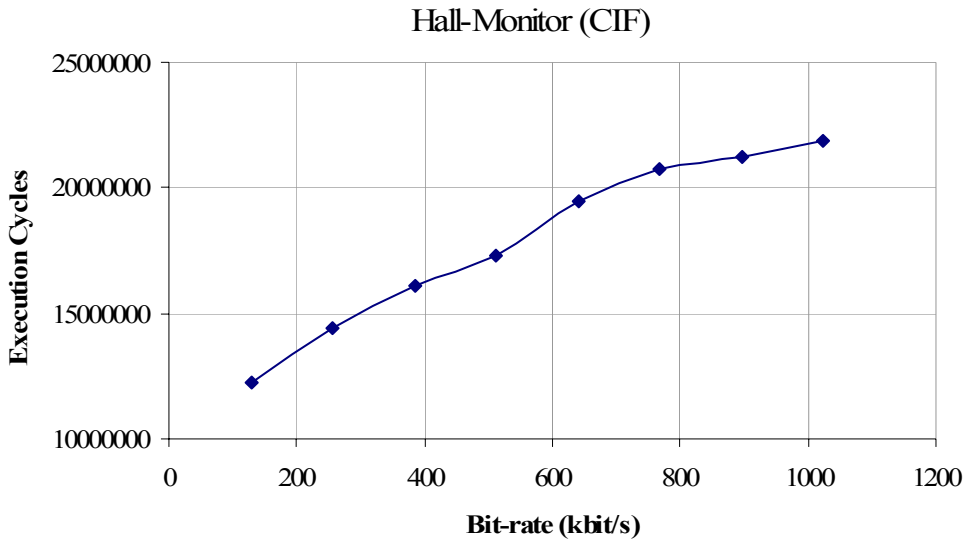
Sequence	**Average error (%)** *multiply*	**Average error (%)** *add*
Hall Monitor	5.7	5.7
Ice	3.5	1.2
Harbour	17.6	13.4
Canoa	13.5	14.3
Total average:	**10.08**	**8.65**

Table VI-VII. Percentage of error between the model prediction and the actual measured additions and multiplications per pixel. For each GOP of every sequence, the average error over a number of adaptation points is presented.

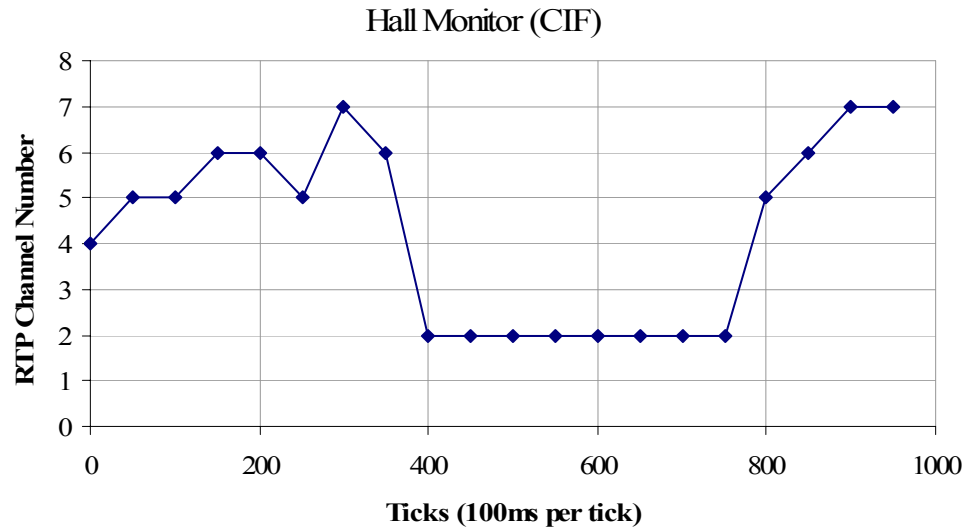
6.3.4.2 R-D-C driven Multimedia Streaming

To present an example of the proposed R-D-C adaptation within an application scenario, we include one set of results with a real-time open-source multimedia streaming system based on the open-source MPEG4-IP software [12]. The results obtained with the proposed R-D-C adaptation mechanism are illustrated in Figure VI-15. In particular, Figure VI-15(a) shows the average-RCM vs. bitrate results associated with eight RTP channels. The RCMs were generated from the model-based GCMs of this paper (see Table VI-VI– "Hall Monitor" sequence) with the methodology described in recent work [48], and are expressed in terms of execution cycles in an Intel Pentium IV processor. Figure VI-15(b) shows the number of channels transmitted to the receiver based on rate and/or complexity constraints. Initially, the adaptation was solely done based on network characteristics information (i.e. rate-distortion constraints) provided by the UEDs. Subsequently, the receiver started another complex application (during the time tick=400 until tick=750), and hence, the complexity of the transmitted bitstream was adapted based on the receiver RCMs.

It was determined that the storage overhead associated with multi-track R-D-C hinting is about 5–15% of the total bitstream, depending on the packet size chosen for a particular hinting scheme [12]. Moreover, it was found [12] that increasing the number of hint tracks only causes negligible increase in overhead due to the fact that, the amount of information generated by hinting is determined by the total number of hinted packets; when the packet size is selected, the total number of packets is also consequently determined independent on whether they are hinted using one track or multiple tracks. Alternatively, if the GCMs are sent to the receiver, the transmission overhead is low (less than 5%): only one number for each *op* is transmitted per AU.



(a)



(b)

Figure VI-15. R-D-C driven multimedia adaptation.

6.3.5 Conclusions And Future Work

The recently standardized MPEG-21 framework enables adaptation based on both terminal capabilities and network conditions. We propose a new and generic rate-distortion-complexity framework that can generate metadata necessary for MPEG-21 Digital Item Adaptation based on the available receiver resources. We illustrate the simplicity and accuracy of the proposed complexity model by predicting the computational complexity of motion-compensated wavelet video coders. The average error between the experimental and model-based generic complexity metrics was found to be

approximately 10%. An example of the proposed network and receiver R-D-C adaptation was presented using a real-time multimedia streaming test-bed. In general, the derived complexity model is generic and can be used for a variety of video coders. As a result, we believe that the proposed complexity-related descriptions could be useful for standardization purposes. Future research could focus on quantifying the extent to which continuous adaptation to the receiver resource complexity can improve the end-to-end performance of multimedia delivery and the battery life of receivers. Another aspect of future research could be to investigate the accuracy of different mappings between GCMs and RCMs for various architectures.

6.4 Summary of the Results of this Chapter

Two models for predicting the complexity of video coding algorithms under realistic conditions have been proposed in this chapter. The two proposals target different aspects of complexity; in particular, the cache modeling of two-dimensional discrete wavelet transforms presented in Section 6.2 targets platforms where the memory bottlenecks are expected to dominate the execution of data-dominated signal processing tools, such as the DWT. The analysis in that section is based on analyzing the algorithmic flow and expressing the expected cache misses in analytical formulations. On the other hand, the complexity modeling framework of motion-compensated video decoders (Section 6.3) is based on a generic decomposition of the arithmetic (and potentially the memory) profile of these systems into a set of basis functions, without specific regards to the details of the algorithms involved.

In the first case, experiments with a simulator of generic pipelined simplescalar architecture revealed that the theoretical framework predicts the expected number of cache misses accurately (Figure VI-5 and Figure VI-6). In the case of real platforms, one can establish a ranking of the different approaches for the memory organization of the DWT calculation that appears to be matching the real measurements (Figure VI-7 and Figure VI-8). Perhaps more importantly, due to the fact that the proposed cache modeling of Subsection 6.2.2.2 is based on the analysis of the data flow within the cache hierarchies for the performance of the DWT, the proposed parameterized equations lead to a deeper understanding of the strengths and weaknesses of each transform-production schedule for the realization of the DWT in programmable platforms.

In the case of the complexity modeling framework of Section 6.3, experiments demonstrate that the proposed paradigm predicts the arithmetic complexity of the chosen video decoding scheme within an accuracy of 10% for a wide variety of video content (Table VI-VII). Due to the fact that no specific fine-tuning of the modeling parameters was performed, the obtained prediction accuracy is a promising result for this category of approaches. Finally, in order to demonstrate a concrete link with practical systems, the proposed framework was utilized within a new streaming architecture and receiver-based complexity adaptation was performed by streaming the model-based generic complexity metrics to the receiver and adapting the decoded bitstream layers according to the estimated complexity (Figure VI-15).

6.5 Appendix

Proof of Template 1: Due to the fact that the processing occurs along a row, all data are taken sequentially from the memory. Thus, the cache blocking strategy will input b_c (useful) bytes simultaneously. Due to the constraint of (6.3) set for this template, all elements of each row fit in the cache. Thus, if each row is processed many times (many passes per row), no additional misses are expected per row. Hence, for a total of $R \cdot C \cdot c_p$ bytes, the compulsory misses reported in (6.4) are expected. ■

Proof of Template 2: The situation after the completion of the row processing can be seen in the left part of Figure VI-16. Due to the sequential processing of the rows, $(R - W)$ rows of the two-dimensional array are expected to be resident in the cache. Of course, the least-recently-accessed row of the array may not be fully stored in the cache, but Figure VI-16 shows the simple case of a number of full rows of the array in the cache in order to facilitate the drawing. The term W can be easily calculated since the total cache size (s_c) is known, and in this way we reach equation (6.7).

The left part of the constraint set for this template (equation (6.5)) assures that almost in all cases $W > 0$. In addition, the right part of the constraint assures that any additional passes per column are not expected to cause misses, since the currently-processed column elements are expected to fit in the cache. The on-cache part of the array is stored into the various cache blocks in portions of b_c bytes, as seen in the example of Figure VI-16. For the input of the first column, the grouping of the cache blocks will cause the simultaneous input of b_c/c_p sequential columns, as the middle of Figure VI-16 shows. These blocks are expected to replace a number of rows of the array that were stored at the least-recently accessed blocks of the cache, due to the LRU replacement strategy. The replaced rows are denoted as a_1. As seen in Figure VI-16, some of the last elements of the first b_c/c_p columns will not be inserted, since they are already on-cache. The number of rows that are replaced can be calculated since the dimensions of the array are known and the number of input blocks of data is $W + a_1$:

$$b_c(W + a_1) = a_1 \cdot C \cdot c_p \Rightarrow a_1 = W \cdot b_c/(C \cdot c_p - b_c) \tag{6.40}$$

Similarly, the right part of Figure VI-16 shows the input of the second group of sequential b_c/c_p columns. Following the same idea, the term a_2 can be calculated as:

$$b_c(W + a_1 + a_2) = a_2 \cdot C \cdot c_p \Rightarrow a_2 = b_c \cdot (W + a_1)/(C \cdot c_p - b_c) \tag{6.41}$$

Generalizing, for the j-th input set of b_c/c_p columns, with $j > 1$ we reach equation (6.8). The constraint set in (6.8) ensures that the total size of the j-th inserted group of columns is smaller than the array height (R). Decimal values for a_j are acceptable in the sense that an incomplete row-replacement was made during the input of the j-th set of b_c/c_p columns.

Every replacement of a cache block in this case constitutes a capacity miss. Thus, for the input of the first column, $(W + a_1)$ misses are expected. Since s_c is larger than $R \cdot b_c$ (constraint (6.5)), the cache is sufficiently large to fit one group of b_c/c_p columns and no additional misses are expected to occur for the rest $(b_c/c_p - 1)$ columns. Thus every group of columns stays in the cache after it has been

used. This explains the rationale behind the right part of the constraint stated for this template. The total number of groups of columns is $(C \cdot c_p / b_c)$. Hence, under the constraint set, the expected amount of misses, is reported in (6.6). ■

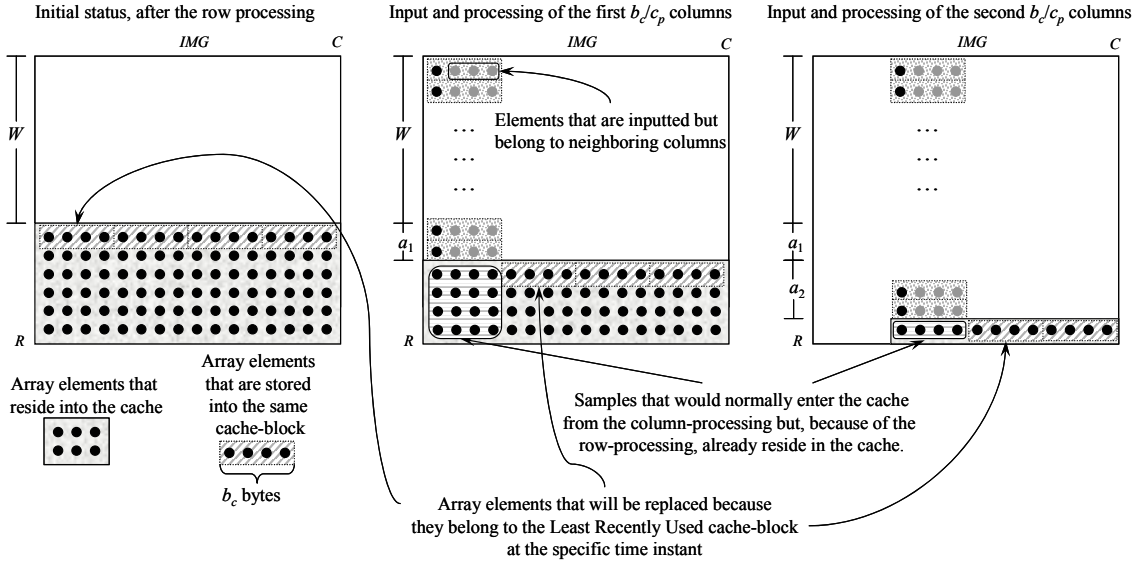


Figure VI-16. The column-by-column input of a two-dimensional array in the cache, after the completion of the row-by-row processing.

Proof of Template 3: This template is similar to the previous one, however, the constraint set for this case shows that it applies for a small cache that has comparable size to a group of input columns. After the row-by-row processing, when one column is inserted, a group of columns comes in the cache as shown in the middle of Figure VI-16. This column causes R misses for one pass. For k passes, if s_c is smaller than $R \cdot b_c$, a total of $k \cdot R$ misses are expected, since this case implies that the entire column does not fit in the cache. If however s_c is larger than $R \cdot b_c$ then, after the first input of each column, no additional misses are expected for the subsequent passes. Thus only R misses are expected for all the k passes. However, since s_c is comparable to $R \cdot b_c$, the subsequent $(b_c/c_p - 1)$ columns of every group will have to be re-inserted (in the vast majority of cases), causing misses as well and as a result a total of $k \cdot R \cdot C$ or $R \cdot C$ misses are expected for the processing of all the groups of columns, depending on the cache size s_c. This is shown in (6.10) and the distinction between the two cases is made with the factor f_{T3}. ■

References

[1] T. Wiegand, "H26L Test Model Long-Term Number 9 (TML-9) draft 0," ITU/TSS/SG16 (VCEG), Document VCEG-N83 D1, December 2001 2001.

[2] S. Saponara, K. Denolf, G. Lafruit, C. Blanch, and J. Bormans, "Performance and complexity co-evaluation of the Advanced Video Coding standard for cost-effective multimedia communications," *EURASIP Journal of Applied Signal Processing*, to appear.

[3] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 704-716, 2003.

[4] M. Ravasi, M. Mattavelli, P. Schumacher, and R. Turney, "High-level algorithmic complexity analysis for the implementation of a motion-JPEG2000 encoder," presented at Workshop on Power and Timing Modeling, Optimization and Simulation, 2003.

[5] J. Valentim, P. Nunes, and F. Pereira, "Evaluating MPEG-4 video decoding complexity for an alternative video verifier complexity model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 1034-1044, 2002.

[6] P. Pirsch, H.-J. Stolberg, Y.-K. Chen, and S. Y. Kung, "Implementation of media processors," *IEEE Signal Processing Magazine*, pp. 48-51, 1997.

[7] D. Sow and A. Eleftheriadis, "Complexity distortion theory," *IEEE Transactions on Information Theory*, vol. 49, pp. 604-608, 2003.

[8] D. A. Patterson and J. L. Hennessy, *Computer Architecture A quantitative approach*: Morgan Kaufmann Publishers, Inc., 1996.

[9] D. H. Albonesi, R. Balasubramonian, S. G. Dropsbo, S. Dwarkadas, F. G. Friedman, M. C. Huang, V. Kursun, G. Magklis, M. L. Scott, G. Semeraro, P. Bose, A. Buyuktosunoglu, P. W. Cook, and S. E. Schuster, "Dynamically tuning processor resources with adaptive processing," *IEEE Computer (special issue on "Power-aware Computing")*, vol. 36, pp. 49-58, 2003.

[10] D. Narayanan and M. Satyanarayan, "Predictive resource management for wearable computing," presented at Proc. Intern. Conf. on Mobile Syst. Appl. and Services, San Fransisco, CA, 2003.

[11] J. Reichel, "Predicting the complexity of signal processing algorithms," presented at Proc. IEEE Int. Conf. on Image Processing, Thessaloniki, GR, 2001.

[12] M. v. d. Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Transactions on Multimedia*, to appear.

[13] M. Boliek, C. Christopoulos, and E. Majani, "JPEG2000 Part I Final Draft International Standard," ISO/IEC JTC1/SC29/WG1, Report September 25, 2000 2000.

[14] ISO/IEC, "Information Technology - Coding of Audio-visual Objects - Part 2: Visual," ISO/IEC JTC1/SC29/WG11, FDIS 14496-1/FDAM1, January 2000.

[15] P. Schelkens, "Multidimensional Wavelet Coding - Algorithms and Implementations," in *Department of Electronics and Information Processing (ETRO)*. Brussel: Vrije Universiteit Brussel, 2001, pp. 212.

[16] C. Chrysafis and A. Ortega, "Line-Based, Reduced Memory, Wavelet Image Compression," *IEEE Transactions on Image Processing*, vol. 9, pp. 378-389, 2000.

[17] E. Ordentlich, D. Taubman, M. J. Weinberger, and G. Seroussi, "Memory efficient scalable line-based image coding," presented at Proceedings of Data Compression Conference, Snowbird, UT, USA, 1999.

[18] G. Lafruit, L. Nachtergaele, J. Bormans, M. Engels, and I. Bolsens, "Optimal memory organization for scalable texture codecs in MPEG-4," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 218-243, 1999.

[19] M. Vishwanath, "The Recursive Pyramid Algorithm for the Discrete Wavelet Transform," *IEEE Transactions on Signal Processing*, vol. 42, pp. 673-676, 1994.

[20] G. Lafruit, L. Nachtergaele, B. Vanhoof, and F. Catthoor, "The Local Wavelet Transform: a memory-efficient, high-speed architecture optimized to a Region-Oriented Zero-Tree Coder," *Integrated Computer-Aided Engineering*, vol. 7, pp. 89-103, 2000.

[21] Y. Andreopoulos, P. Schelkens, N. D. Zervas, T. Stouraitis, C. E. Goutis, and J. Cornelis, "A Wavelet-Tree Image Coding System with Efficient Memory Utilization," presented at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, Utah, USA, 2001.

[22] C. Chakrabarti and C. Mumford, "Efficient realizations of encoders and decoders based on the 2-D discrete wavelet transforms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, pp. 289-298, 1999.

[23] Y. Andreopoulos, P. Schelkens, and J. Cornelis, "Analysis of Wavelet Transform Implementations for Image and Texture Coding Applications in Programmable Platforms," presented at IEEE Workshop on Signal Processing Systems (SiPS), Antwerpen, Belgium, 2001.

[24] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, 1989.

[25] F. Catthoor, S. Wuytack, E. D. Greef, F. Balasa, L. Nachtergaele, and A. Vandecappelle, *Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design*. Dordrecht (The Netherlands): Kluwer Academic Publishers, 1998.

[26] S. K. Przybylski, *Cache and memory hierarchy - A performance-directed approach*: Morgan-Kaufmann (San Fransisco), 1990.

[27] G. H. Golub and C. F. V. Loan, *Matrix computations*, third ed: The Johns Hopkins University Press, 1996.

[28] P. Meerwald, R. Norcen, and A. Uhl, "Cache issues with JPEG-2000 wavelet lifting," presented at Proc. of SPIE Visual Communications and Image Processing, San Jose, CA, 2002.

[29] H. Komi and A. Ortega, "Analysis of cache efficiency in two-dimensional wavelet transform," presented at Proceedings of IEEE International Conference on Multimedia and Expo, paper no TP11.06, 2001.

[30] Y. Andreopoulos, P. Schelkens, G. Lafruit, K. Masselos, and J. Cornelis, "High-level Cache Modelling for 2-D Discrete Wavelet Transform Implementations," *VLSI Signal Processing Systems (special issue on "Signal Processing Systems")*, vol. 34, pp. 209-226, 2001.

[31] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, pp. 247-269, 1998.

[32] Y. Andreopoulos, N. D. Zervas, G. Lafruit, P. Schelkens, T. Stouraitis, C. E. Goutis, and J. Cornelis, "A Local Wavelet Transform Implementation versus an Optimal Row-Column Algorithm for the 2D Multilevel Decomposition," presented at IEEE International Conference on Image Processing (ICIP), Thessaloniki, Greece, 2001.

[33] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *ACM SIGARCH Computer Architecture News*, vol. 25, pp. 13-25, 1997.

[34] "Intel's architecture software developer's manual - Volume 1," Intel Corporation, 1997.

[35] "VTune performance Analyzer v4.5," Intel Corporation, http://developer.intel.com/vtune.

[36] http://www.support.trimedia.philips.com/.

[37] A. Vetro and C. Timmerer, "Digital item adaptation: Overview of standardization and research activities," *IEEE Transactions on Multimedia*, to appear.

[38] M.-T. Sun and A. R. Reibman, *Compressed Video over Networks*, vol. 5. New York - Basel: Marcel Dekker, Inc., 2001.

[39] "Digital item adaptation," ISO/IEC 21000-7 FDIS Part 7: ISO/IEC JTC1/SC29/WG11 (MPEG), n6168, Dec. 2003.

[40] Z. He and S. K. Mitra, "A unified rate-distortion analysis framework for transform coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, 2001.

[41] D. Mukerjee, E. Delfosse, J.-G. Kim, and Y. Wang, "Terminal and network quality of service," *IEEE Transactions on Multimedia*, vol. to appear.

[42] D. Taubman, "EBCOT: Embedded, block coding with optimized truncation," ISO/IEC JTC1/SC29/WG1, N1020R, October 1998 1998.

[43] M. Flierl, "Doctorate Thesis: Video Coding with Superimposed Motion-Compensated Signals." Erlangen: University of Erlangen-Nuremberg, 2003.

[44] http://www.imec.be/atomium.

[45] Q. Li and M. v. d. Schaar, "A flexible streaming architecture for efficient scalable coded video transmission over IP networks," ISO/IEC JTC1/SC29/WG11 (MPEG), m8944, Oct. 2002.

[46] D. Singer, W. Belknap, and G. Franceschini, "ISO media file format specification - MP4 Technology under consideration for ISO/IEC 14496-1:2002 Amd 3," Committee Draft, ISO/IEC JTC1/SC29/WG11 (MPEG), n4270-1, July 2001.

[47] F. Verdicchio, Y. Andreopoulos, T. Clerckx, J. Barbarien, A. Munteanu, J. Cornelis, and P. Schelkens, "Scalable video coding based on motion-compensated temporal filtering: complexity and functionality analysis," presented at Proc. IEEE Intern. Conf. on Image Processing, Singapore, SG, to appear.

[48] G. Lange, M. v. d. Schaar, and V. Akella, "Complexity metric driven energy optimization framework for implementing MPEG-21 scalable video decoders," presented at IEEE International Conference on Accoustics, Speech, and Signal Processing, 2005.

Chapter VII

CONCLUSIONS

FOLLOWING current technological and social trends, today's majority of digital video transmissions around the globe occur via diverse computer network architectures and inhomogeneous end-user terminals. Consequently, a large number of requirements need to be satisfied by modern state-of-the-art source coding algorithms, if these algorithms are to be used in such scenarios. At the source-coding level, these requirements correspond to support for multiple resolutions and frame-rates, and easy adaptability to bandwidth variations of error-prone transmission channels. This adaptability should be provided based on a single compressed bitstream, with every sub-stream producing comparable visual quality to state-of-the-art non-scalable coding operating at the specified bitrate, resolution and frame-rate. Finally, for each operating point, the decoding of each sub-stream should have comparable complexity as non-scalable coding. These requirements constitute the "holy-grail" of scalable video coding research. As in every difficult quest, their complete achievement has been a withstanding problem for a number of years.

7.1 Dissertation Overview and Concluding Remarks

In this dissertation, based on previous research advances in motion-compensated prediction and motion-compensated temporal filtering, we made a number of proposals in the area of wavelet-based scalable video coding. In particular, we introduced in-band motion compensated prediction within closed-loop and open-loop video coding systems. For open-loop systems, in-band motion-compensated temporal filtering was proposed by coupling in-band motion compensated prediction with in-band motion compensated update. In all cases it was found that, with the use of a complete-to-overcomplete wavelet transform for motion estimation and compensation in the wavelet domain, an equivalent system with the conventional closed-loop and open-loop frameworks can be provided in-band. This was shown to have several advantages from the functionality point of view, e.g. for base-layer compatibility with existing MPEG-alike systems. In general, different prediction structures for each resolution level may satisfy different application requirements. Experimental evidence confirmed that performing in-band motion compensated prediction and update provides

substantially-improved resolution scalability versus that provided by conventional systems that perform MCP in the spatial domain. Similar to the way an open-loop video coding architecture provides bitrate scalability at seemingly no cost in compression efficiency versus the equivalent closed-loop system, in-band video coding based on the ODWT appears to provide improved resolution scalability at little or no cost in compression efficiency for the full resolution.

Finally, it was shown that the use of advanced motion estimation benefits both spatial-domain and in-band systems in the same manner. Moreover, objective evaluations, performed by coding a large variety of standard-definition video sequences in a wide range of bitrates, revealed that fully-scalable wavelet-based systems using advanced motion-compensated temporal filtering are (on-average) comparable or superior to the current state-of-the-art in non-scalable coding, i.e. the MPEG-4 Advanced Video Coder.

By focusing on the transform aspects of the proposed in-band video coding architectures, a new framework for the construction of the overcomplete DWT starting from the subbands of the critically-sampled decomposition was presented. The proposed framework has inherent advantages in comparison to the conventional approach since it consists of a direct transform from the complete to the overcomplete DWT, using the minimum number of downsampling operations and no upsampling. For resolution-scalable video coding applications that utilize a level-by-level construction of the ODWT, it was demonstrated that the proposed CODWT has significant implementation advantages over the conventional approach because it offers (a) significant computation savings, and (b) a single-rate calculation that can provide a scalable reduction in the transform-production delay. These features lead to inherent computational scalability in comparison to the conventional approach.

In the last part of this dissertation, complexity aspects of scalable video coding were studied. In particular, two models for predicting the complexity of video coding algorithms under realistic conditions have been proposed. The two proposals target different aspects of complexity; our first proposal consists of a theoretical framework for cache modeling of two-dimensional discrete wavelet transforms and targets platforms where the memory bottlenecks are expected to dominate the execution of data-dominated signal processing tools, such as the DWT. The analysis is based on analyzing the algorithmic flow and expressing the expected cache misses in analytical formulations. Our second proposal consists of a complexity modeling framework of motion-compensated video decoders. The framework is based on a generic decomposition of the arithmetic (and potentially the memory) profile of these systems into a set of basis functions, without specific regards to the details of the algorithms involved.

In the first case, experiments with a simulator of a generic pipelined simplescalar architecture revealed that the theoretical framework predicts the expected number of cache misses accurately. In the case of real platforms, one can establish a complexity ranking of the different approaches for the memory organization of the DWT calculation that appears to be matching the real measurements. Perhaps more importantly, due to the fact that the proposed cache modeling is based on the analysis of the data flow within the cache hierarchies for the performance of the DWT, the proposed parameterized equations lead to a deeper understanding of the strengths and weaknesses of each transform-production schedule for the realization of the DWT in programmable platforms.

In the case of the complexity modeling framework for motion-compensated video decoders, experiments demonstrated that the proposed paradigm predicts the arithmetic complexity of the chosen video decoding scheme within an accuracy of 10% for a wide variety of video content. Due to the fact that no specific fine-tuning of the modeling parameters was performed, the obtained prediction accuracy is a promising result for this category of approaches. Finally, in order to demonstrate a concrete link with practical systems, the proposed framework was utilized within a new streaming architecture and receiver-based complexity adaptation was performed by streaming the model-based generic complexity metrics to the receiver and adapting the decoded bitstream layers according to the estimated complexity.

7.2 Prospective Work

In this dissertation we dealt with a number of topics in the area of video compression. Based on practical requirements, several wavelet-based video coding architectures have been proposed. Starting from the presented results, a number of video coding topics could still be investigated in future research.

One particularly important aspect from a practical point of view, concerns the visual improvement of the results obtained with the proposed codecs versus visually-optimized video codecs such as the MPEG-4 AVC. Visual inspections reveal that, although at the same PSNR level, in many cases the wavelet-based codecs of this thesis tend to produce significantly more blocking artifacts (or ringing) versus AVC. Tools to be considered for this task could include adaptive deblocking filters and post-processing filters.

Another practical aspect concerns the improvement of low-resolution and low frame-rate results of wavelet-based MCTF systems. The proposed in-band MCTF architectures perform a significant step in this direction by improving the visual quality of low-resolution sequences. Nevertheless, different spatial decomposition filters might provide a smoother approximation signal at low resolution, thereby reducing the amount of spatial aliasing observed in some cases in the low-frequency subbands of wavelet-based codecs. In relation to this topic, improvements in the overall compression results are expected if additional research is performed in the still-image coding part of the proposed architectures. Although coders such as the QT-L perform very well for natural images, there are improvements to be obtained with an entropy coding scheme specifically designed based on the statistics of error-frame images.

Finally, an interesting research direction would be to attempt to combine adaptive spatial wavelet decompositions, such as bandelets, with the adaptive temporal filtering techniques proposed in this thesis. This may provide significant gains, especially if it is combined with better entropy coding strategies.

In the area of complexity modeling, there are a number of open questions that could be addressed in future research. Although the cache-modeling strategy appears to provide some insight on the relative performance tradeoffs of different traversal schedules for the discrete wavelet transform, it may be interesting to extend this analysis to other parts of the system, such as the MCTF process. Moreover,

in such a case, the tradeoffs in terms of memory and arithmetic complexity would be significantly higher since entire frames (or wavelet subbands) participate in the temporal filtering process. Finally, with respect to the proposed modeling framework for motion-compensated wavelet decoding, a number of different topics could be addressed, such as: establishment of more basis functions for the complexity decomposition; theoretical investigation of statistical properties of the basis functions and the complexity decomposition coefficients; study of the inter-relationship of the various basis functions (e.g. whether some functions can be considered to be orthogonal to others); validation of the proposed framework for the case of the memory-complexity estimation. In addition, by considering fine-grain sampling of the complexity decomposition variables during the off-line training and establishing the complexity decomposition coefficients based on a larger training set, the limits in the accuracy of the modeling strategy could be investigated.

Finally, an interesting research direction in this area would be to combine the complexity models proposed in this dissertation with parameterized weighing factors in order to take into account current and future technology effects on performance of both the memory hierarchy and the arithmetic and instruction decoding units. In this way, a final ranking of the proposed coding methods with state-of-the-art existing related methods could be established, in terms of speed and power consumption.

ANNEX A

PUBLICATION LIST

Journal Publications

1. J. Barbarien, A. Munteanu, F. Verdicchio, Y. Andreopoulos, J. Cornelis and P. Schelkens, "Motion and texture rate-allocation for prediction-based scalable motion-vector coding," Signal Processing: Image Communication, to appear.
2. Y. Andreopoulos, A. Munteanu, G. Van der Auwera, J. Cornelis and P. Schelkens, "Single-rate calculation of overcomplete discrete wavelet transforms for scalable coding applications," Signal Processing, to appear.
3. M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," IEEE Trans. on Multimedia, to appear.
4. Y. Andreopoulos, A. Munteanu, G. Van der Auwera, J. Cornelis and P. Schelkens, "Complete-to-overcomplete discrete wavelet transforms: theory and applications," IEEE Trans. on Signal Processing, vol. 53, no. 4, pp. 1398-1412, April 2005.
5. D. Turaga, M. van der Schaar, Y. Andreopoulos, A. Munteanu and P. Schelkens, "Unconstrained motion compensated temporal filtering (UMCTF) for efficient and flexible interframe wavelet video coding," Signal Processing: Image Communication, vol. 20, no. 1, pp. 1-19, Jan. 2005.
6. Y. Andreopoulos, A. Munteanu, J. Barbarien, M. van der Schaar, J. Cornelis and P. Schelkens, "In-band motion compensated temporal filtering," Signal Processing: Image Communication (special issue on "Subband/Wavelet Interframe Video Coding"), vol. 19, no. 7, pp. 653-673, Aug. 2004.
7. J. Barbarien, A. Munteanu, F. Verdicchio, Y. Andreopoulos, J. Cornelis and P. Schelkens, "Scalable motion vector coding," IEE Electronics Letters, vol. 40, no. 15, pp. 932-934, July 2004.
8. Y. Andreopoulos, P. Schelkens, G. Lafruit, K. Masselos and J. Cornelis, "High-level cache modeling for 2-D discrete wavelet transform implementations," Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology (special issue on "Signal Processing Systems"), vol. 34, no. 3, pp. 209-226, July 2003.
9. N. D. Zervas, G. P. Anagnostopoulos, V. Spiliotopoulos, Y. Andreopoulos and C. E. Goutis, "Evaluation of design alternatives for the 2-D discrete wavelet transform," IEEE Trans. on Circuits Systems for Video Technology, no. 12, vol. 11, pp. 1246-1262, Dec. 2001.

Selected Conference Publications

1. M. van der Schaar, Y. Andreopoulos and Q. Li, "Real-time ubiquitous multimedia streaming using rate-distortion-complexity models," Proc. of IEEE Global Telecommunications Conference, GLOBECOM'04.
2. J. Barbarien, A. Munteanu, F. Verdicchio, Y. Andreopoulos, J. Cornelis and P. Schelkens, "Scalable motion vector coding," Proc. IEEE International Conf. on Image Processing, ICIP'04, Singapore, SG, to appear.
3. F. Verdicchio, Y. Andreopoulos, T. Clerckx, J. Barbarien, A. Munteanu, J. Cornelis and P. Schelkens, "Scalable video coding based on motion-compensated temporal filtering: complexity and functionality analysis," Proc. IEEE International Conf. on Image Processing, ICIP'04, Singapore, SG, to appear.
4. A. Munteanu, Y. Andreopoulos, M. van der Schaar, P. Schelkens and J. Cornelis, "Control of the distortion variation in video coding systems based on motion compensated temporal filtering," Proc. IEEE International Conf. on Image Processing, ICIP'03, Barcelona, SP, vol. 2, pp. 61-64.
5. J. Barbarien, Y. Andreopoulos, A. Munteanu, P. Schelkens and J. Cornelis, "Motion vector coding for in-band motion compensated temporal filtering," Proc. IEEE International Conf. on Image Processing, ICIP'03, Barcelona, SP, vol. 2, pp. 783-786.
6. Y. Andreopoulos, M. van der Schaar, A. Munteanu, P. Schelkens and J. Cornelis, "Spatio-temporal-SNR scalable wavelet coding with motion compensated DCT base-layer architectures," Proc. IEEE International Conf. on Image Processing, ICIP'03, Barcelona, SP, vol. 2, pp. 795-798.
7. Y. Andreopoulos, M. van der Schaar, A. Munteanu, J. Barbarien, P. Schelkens and J. Cornelis, "Complete-to-overcomplete discrete wavelet transforms for fully-scalable video coding with MCTF," Proc. Visual Comunications and Image Processing, Special Session on Motion-Compensated Wavelet Coding (invited), VCIP'03, Lugano, SW, vol. 5150 (2003), pp. 719-731.
8. J. Barbarien, Y. Andreopoulos, A. Munteanu, P. Schelkens and J. Cornelis, "Coding of motion vectors produced by wavelet-domain motion estimation," Proc. Picture Coding Symposium, PCS'03, Saint Malo, FR, vol. 1, pp. 193-197, April 2003.
9. Y. Andreopoulos, M. van der Schaar, A. Munteanu, J. Barbarien, P. Schelkens and J. Cornelis, "Fully- scalable wavelet video coding using in-band motion compensated temporal filtering," Proc. IEEE International Conf. on Acoustics Speech and Signal Processing, ICASSP'03, Hong Kong, CN, vol. 3, pp. 417-420, March 2003.
10. F. Verdicchio, Y. Andreopoulos, A. Munteanu, J. Barbarien, P. Schelkens, J. Cornelis and A. Peppino, "Scalable video coding with in-band prediction in the complex wavelet transform," Proc. IEEE Advanced Concepts for Intelligent Vision Systems, ACIVS'02, Ghent, BE, vol. 1, pp. 232-238, Sept. 2002.
11. Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens and J. Cornelis "Scalable wavelet video-coding with in-band prediction - Implementation and experimental results," Proc. IEEE International Conf. on Image Processing, ICIP'02, Rochester, US, vol. 3, pp. 729-732, Sept. 2002.

12. Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens and J. Cornelis, "A new method for complete-to-overcomplete discrete wavelet transforms," IEEE Digital Signal Processing, DSP'02, Santorini, GR, vol. 2, pp. 501-504, July 2002.
13. Y. Andreopoulos, K. Masselos, P. Schelkens, G. Lafruit and J. Cornelis, "Cache misses and energy-dissipation results for JPEG-2000 filtering," IEEE Digital Signal Processing, DSP'02, Santorini, GR, vol. 1, pp. 201-209, July 2002.
14. G. Lafruit, Y. Andreopoulos and B. Masschelein, "Reduced memory requirements in modified JPEG2000 codec," Proc. IEEE Digital Signal Processing, DSP'02, Santorini, GR, vol. 1, pp. 219-226, July 2002.
15. Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens and J. Cornelis "Wavelet-based fully-scalable video coding with in-band prediction," Proc. IEEE Benelux Signal Processing Symposium, SPS'02, Leuven, BE, pp. 217-220, March 2002.
16. Y. Andreopoulos, P. Schelkens and J. Cornelis, "Analysis of wavelet transform implementations for image and texture coding applications in programmable platforms," Proc. IEEE Signal Processing Systems, SIPS'01, Antwerp, BE, pp. 273-284, Sept. 2001.
17. Y. Andreopoulos, N. D. Zervas, G. Lafruit, P. Schelkens, T. Stouraitis, C. E. Goutis and Jan Cornelis, "A local wavelet transform implementation versus an optimal row-column algorithm for the 2-D multilevel decomposition," Proc. IEEE International Conf. on Image Processing, ICIP'01, Thessaloniki, GR, vol. 1, pp. 330-333, Oct. 2001.
18. Y. Andreopoulos, P. Schelkens, T. Stouraitis and J. Cornelis "Efficient implementations of a wavelet transform - A roadmap," Proc. IEEE/IEE International Workshop on Systems, Signals and Image Processing, IWSSIP'01, Bucharest, RO, vol. 1, pp. 74-79 June 7-9, 2001.
19. Y. Andreopoulos, P. Schelkens, N. D. Zervas, T. Stouraitis, C. E. Goutis and Jan Cornelis, "A wavelet-tree image coding system with efficient memory utilization," Proc. IEEE International Conf. on Acoustics Speech and Signal Processing, ICASSP'01, Salt Lake City, UT, vol. 3, pp. 1709-1712, May 2001.
20. V. Spiliotopoulos, N. D. Zervas, Y. Andreopoulos, G. Anagnostopoulos and C. E. Goutis, "Quantization effect on VLSI implementations for the 9/7 DWT Filters," Proc. IEEE International Conf. on Acoustics, Speech and Signal Processing, ICASSP'01, Salt Lake City, UT, vol. 2, pp. 1197-1200, May 2001.

Contributions to Standardization Efforts

1. J. Barbarien, A. Munteanu, Y. Andreopoulos, F. Verdicchio, J. Cornelis and P. Schelkens, "Prediction-based scalable motion vector coding," ISO/IEC JTC1/SC29/WG11, m11016, MPEG 69th meeting, Redmond, US, July 2004.
2. Y. Andreopoulos, A. Munteanu, M. van der Schaar, J. Cornelis and P. Schelkens, "Comparison between "t+2D" and "2D+t" architectures with advanced motion compensated temporal filtering," ISO/IEC JTC1/SC29/WG11, m11045, MPEG 69th meeting, Redmond, US, July 2004.
3. Y. Andreopoulos, F. Verdicchio, J. Barbarien, A. Munteanu, M. Van der Schaar and P. Schelkens, "Response to call for proposals on scalable video coding technology," ISO/IEC JTC1/SC29/WG11, m10589, MPEG 68th meeting, Munich, Germany, March 2004.

4. Y. Andreopoulos, J. Barbarien, F. Verdicchio, A. Munteanu, M. van der Schaar, J. Cornelis and P. Schelkens, "Response to call for evidence on scalable video coding," ISO/IEC JTC1/SC29/WG11, m9911, MPEG 65th meeting, Trondheim, Norway, July 2003.
5. J. Barbarien, Y. Andreopoulos, A. Munteanu, P. Schelkens and J. Cornelis, "Coding of motion vectors produced by wavelet-domain motion estimation," ISO/IEC JTC1/SC29/WG11, m9249, MPEG 63rd meeting, Awaji island, Japan, December 2002.
6. Y. Andreopoulos, A. Munteanu, P. Schelkens, M. van der Schaar and J Cornelis, "Control of the distortion variation in motion compensated temporal filtering," ISO/IEC JTC1/SC29/WG11, m9253, MPEG 63rd meeting, Awaji island, Japan, December 2002.
7. Y. Andreopoulos, M. van der Schaar, A. Munteanu, J. Barbarien, P. Schelkens and J. Cornelis "Open-loop, in-band, motion-compensated temporal filtering for objective full-scalability in wavelet video coding," ISO/IEC JTC1/SC29/WG11,m9026, MPEG 62nd meeting, Shanghai, China, October 2002.
8. M. van der Schaar, J. Ye, Y. Andreopoulos, A. Munteanu, "Fully scalable 3-D overcomplete wavelet video coding using adaptive motion compensated temporal filtering," ISO/IEC JTC1/SC29/WG11,m9037, MPEG 62nd meeting, Shanghai, China, October 2002.
9. Y. Andreopoulos, A. Munteanu, G. Van der Auwera, J. Barbarien, P. Schelkens and J. Cornelis "Wavelet-based fine granularity scalable video coding with in-band prediction," ISO/IEC JTC1/SC29/WG11, m7906, MPEG 59th meeting, Jeju island, Korea, March 2002.

ACRONYMS

A

AVC	Advanced Video Coder
AU	Adaptation Unit

B

–

C

CABAC	Context Adaptive Binary Arithmetic Coder
CAVLC	Context Adaptive Variable Length Coding
CODWT	Complete-to-Overcomplete Discrete Wavelet Transform

D

DIA	Digital Item Adaptation
DFT	Discrete Fourier Transform
DPCM	Differential Pulse Code Modulation
DSP	Digital Signal Processor
DTCWT	Dual Tree Complex Wavelet Transform
DWT	Discrete Wavelet Transform

E

EBCOT	Embedded Block Coding with Optimized Truncation
EC	Entropy Coder

F

FFT	Fast Fourier Transform
FGS	Fine Granularity Scalability
FO-mode	Full Overcomplete mode

G

GCM	Generic Complexity Metrics
GOF	Group Of Frames
GRM	Generic Reference Machine

H

HFO-mode	High-Frequency Overcomplete mode

I

IB-MCP	In-Band Motion Compensated Prediction
IB-MCU	In-Band Motion Compensated Update
IBMCTF	In-Band Motion Compensated Temporal Filtering
IBMH-MCP	In-Band Multihypothesis Motion Compensated Prediction
ICWT	Inverse Continuous Wavelet Transform
IDCT	Inverse Discrete Cosine Transform
IDWT	Inverse Discrete Wavelet Transform

J

JVT	Joint Video Team

K

–

L

LBS	Low-Band Shift
LBWT	Line-Based Wavelet Transform
LIP	List of Insignificant Pixels
LIS	List of Insignificant Subtrees
LNC	List of Non-significant Coefficients
LS	Lattice Structure
LSB	Least Significant Bit
LSP	List of Significant Pixels
LWT	Local Wavelet Transform

M

MB	Macroblock
MC-DCT	Motion-Compensated Discrete Cosine Transform

MC-EZBC	Motion-Compensated Embedded Zeroblock Coding
MCP	Motion Compensated Prediction
MCTF	Motion Compensated Temporal Filtering
MCU	Motion Compensated Update
ME	Motion Estimation
MF-MCP	Multiframe Motion Compensated Prediction
MH-MCP	Multihypothesis Motion Compensated Prediction
MH-MCU	Multihypothesis Motion Compensated Update
MH-ME	Multihypothesis Motion Estimation
MPEG	Moving Picture Experts Group
MVC	Motion Vector Coding

N

–

O

OB-MCP	Overlapped Block Motion Compensated Prediction
ODWT	Overcomplete Discrete Wavelet Transform

P

PCRDO	Post Compression Rate Distortion Optimization
PE	Processor Element
PFGS	Progressive Fine Granularity Scalability

Q

QoS	Quality of Service
QT-L	QuadTree Limited

R

RCM	Real Complexity Metrics
RCWT	Row-Column Wavelet Transform
RDF	Roughly Depth First
RL	Refinement List

S

SAD	Sum of Absolute Differences
SAQ	Successive Approximation Quantization
SBD	Subband Decoder

SBF	Strictly Breadth First
SD-MCP	Spatial-Domain Motion Compensated Prediction
SDMCTF	Spatial-Domain Motion Compensated Temporal Filtering
SDTCWT	Single-to-Dual Tree Complex Wavelet Transform
SPIHT	Set Partitioning in Hierarchical Trees
SSD	Sum of Square Differences
STFT	Short-Time Fourier Transform

T

–

U

UDWT	Undecimated Discrete Wavelet Transform
UMCTF	Unconstrained Motion Compensated Temporal Filtering
UTM	Universal Turing Machine

V

VCEG	Video Compression Experts Group
VCV	Video Complexity Verifier
VLIW	Very-Long Instruction Word

W

WWW	World Wide Web

X

–

Y

–

Z

–