ADVANCED MOTION MODELING FOR 3D VIDEO CODING

NIKOLA BOŽINOVIĆ

Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

BOSTON UNIVERSITY

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

ADVANCED MOTION MODELING FOR 3D VIDEO CODING

by

NIKOLA BOŽINOVIĆ

B.S., University of Niš, 2000
M.S., Boston University, 2001

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2006

Approved by

First Reader

Janusz Konrad, Ph.D.
Associate Professor of Electrical and Computer Engineering

Second Reader

John W. Woods, Ph.D.
Professor of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute

Second Reader

W. Clem Karl, Ph.D.
Professor of Electrical and Computer Engineering and
Professor of Biomedical Engineering

Fourth Reader

Maja Bystrom, Ph.D.
Associate Professor of Electrical and Computer Engineering

To Mina,
my little star.

Acknowledgments

For the accomplishment of this thesis, I have to thank the following:

First and foremost, I offer my sincerest gratitude to my advisor, Prof. Janusz Konrad, for his vigorous intellectual support, discriminating guidance, and exacting editorial standards, which challenged me to do my absolute best - even when I was quite sure I was already giving it. Working with him during the last six years has been nothing but a source of immense inspiration and pleasure.

Prof. John W. Woods, to whom I am deeply grateful for two reasons: his outstanding contribution to the field in which I worked for better part of my graduate work, and for his insightful comments and suggestions that made my dissertation manuscript so much better.

Prof. Clem Karl and Prof. Maja Bystrom for their stimulating hours of advice and their refinement of the thesis details that would have easily escaped my attention.

My excellent mentors and collaborators during three exciting summers I spent at University of Nice, Prof. Michael Barlaud, and Microsoft Research Asia in Beijing, Dr Feng Wu and Mr. Jizheng Xu. Their inspiring discussions and warm hospitality will be remembered.

My friend and colleague, Mirko Ristivojevic, for flexibility in sharing the same lab space for five years with me, no less than for his great flexibility and brilliance in sharing ideas. I would also like to thank all my other colleagues at Boston University for their shrewd observations and unselfish support.

Dragomir Nikolitch Charitable Fund and Studenica Foundation, for the scholarship they granted me.

Prof. William Oliver, for helping me and my wife find our place at Boston University, and – in the absence of our own families here – for being like a family to us.

My parents Dušan and Olivera, and my brother, Neša, for their relentless belief in me.

Finally, I would like to express my deepest love to my wife, Mina, without whom this accomplishment would not mean nearly as much.

ADVANCED MOTION MODELING FOR 3D VIDEO CODING

(Order No.)

NIKOLA BOŽINOVIĆ

Boston University, College of Engineering, 2006

Major Professor: Janusz Konrad, Ph.D.,

Associate Professor of Electrical and Computer Engineering

ABSTRACT

Driven by new multimedia applications and the growing demand for more flexible and efficient transmission of video, a new approach to video coding has been recently proposed as an alternative to classical hybrid schemes. Instead of sequential frame-based predictive processing, the new approach is based on spatio-temporal 3D transforms, open-loop non-predictive processing, and embedded quantization and coding. This thesis investigates motion modeling for this new coding environment, as well as the impact of such modeling on both coder design and performance.

The first aspect of this thesis deals with video coding based on 3D discrete cosine transform (DCT). We analyze 3D DCT spectrum properties of a globally translating image and show how to use its characteristic footprint for fast and efficient video coding. Previous approaches to 3D DCT video coding have lead to rather modest compression gains due to a limited use of motion characteristics in the transform domain. We develop a coefficient scanning order that adapts to motion, unlike the fixed *zig-zag* scanning of JPEG. We combine this adaptive scanning with a new 3D quantization model to design a low-complexity 3D DCT video coder. The new coder consistently outperforms MPEG-2 both subjectively and objectively (by more than 1.5 dB) at about 25% reduced complexity, while approaching the performance of MPEG-4 (within 0.8 dB) at less than half computational complexity.

The second aspect of this thesis involves the role of motion in emerging video coders

based on 3D discrete wavelet transform (DWT) and motion-compensated temporal filtering (MCTF). Motion invertibility, central to the optimality of lifted MCTF implementation, is first investigated. We introduce a metric for *invertibility error* between two motion fields. We develop advanced motion inversion methods and demonstrate their effectiveness in improving the update lifting step. Experimental results confirm that a better motion inversion, quantified by lower invertibility error, leads to an increase in coding gain up to 0.5 dB over simpler inversion techniques. We propose a new method for *occlusion-aware* modeling and estimation of motion fields and use it to create an adaptive 3D DWT coding structure. Implicit modeling of occluded/uncovered areas, combined with the use of longer wavelet kernels, improves both the prediction and update lifting steps and results in the overall compression gain of up to 1 dB over a non-adaptive coder.

The role of motion in 3D DWT coding motivates our exploration of advanced spatial motion models. We improve the performance of standard deformable triangular meshes through topology modification and an enhanced estimation algorithm. We also introduce a motion model based on hierarchical cubic splines and demonstrate its benefits in terms of motion-compensated prediction over the traditional block-constant model, especially at very low (176×144 and lower) spatial resolutions. We introduce and implement a new hierarchical *mixture motion model* for spatially-scalable motion representation that uses cubic spline-based motion at lower spatial resolutions and variable-size block matching at higher resolutions. Experimental results demonstrate up to 1 dB performance gain of the mixture motion model over the all-block model at lower spatial resolutions without a negative impact on the full resolution performance. Overall, the best scalable results are obtained using either one or two spline-based motion layers at the lowest spatial resolutions of the mixture motion model.

Contents

List of Tables

List of Figures

List of Abbreviations

1-D	.....	One-Dimensional
2-D	.....	Two-Dimensional
3-D	.....	Three-Dimensional
AVC	.....	Advanced Video Coder
CIF	.....	Common Intermediate Format (352×288)
DCT	.....	Discrete Cosine Transform
DFT	.....	Discrete Fourier Transform
DVD	.....	Digital Versatile/Video Disc
DWT	.....	Discrete Wavelet Transform
GOF	.....	Group of Frames
GOP	.....	Group of Pictures
IPTV	.....	Internet Protocol Television
JPEG	.....	Joint Photographic Experts Group
KLT	.....	Karhunen-Lóeve Transform
MCTF	.....	Motion-Compensated Temporal Filtering
MCTP	.....	Motion-Compensated Temporal Prediction
MPEG	.....	Moving Picture Experts Group
PDA	.....	Personal Digital Assistant
PSNR	.....	Peak Signal-to-Noise Ratio
QCIF	.....	Quarter CIF (176×144)
QQCIF	.....	Quarter QCIF (88×72)

Chapter 1

Introduction

The last two decades witnessed an impressive progress in the area of digital image and video processing, with the research impact stretching far beyond academia. Advances in imaging and video technology are fundamentally changing ways in which people communicate visual information. Virtually all facets of daily life are being affected; visual applications are increasingly used at work (corporate video-conferencing, video surveillance), in schools (distance learning, lecture archival), and at home for both personal communication (digital photography and home video, video-phone) and entertainment (digital TV, DVD). Even the cinema is quickly moving towards digital big screen distribution.

The rapid development of modern digital image and video systems can be most easily followed by looking at international compression standardization. Over the years, standardization has played a central role in shaping new multimedia technologies. Standards have also been instrumental in swift consumer adoption of new products and technologies. Their importance stems from the fact that an incredible variety of consumer-level products requires hardware implementation of the underlying coding algorithms - any attempt at mass production would be doomed if specifications were constantly changing. Standards also gained strong support from a large number of competing manufacturers and content providers. They realized early on that lack of a standard industrial platform - "the most of the content working on majority of devices" - would have a catastrophic effect on the entire industry.

Digital photography was the first to get its international standard. The Joint Photographic Experts Group (JPEG) still image coding standard was completed by August 1990; commercial applications using it began to show up in 1991. JPEG found its way to

the core of virtually all image communication systems, from Internet applications, through digital photography and document archiving, to medical imaging. JPEG2000, finalized in 2001, delivers even a richer set of features at higher compression rates.

Early expectations from video coding standards were modest. The first international standard, developed by the Moving Picture Experts Group and named MPEG-1, had the goal of replacing VHS with a digital equivalent, in both quality and storage capacity. The initial modest goals soon gave way to more exciting applications and higher goals. Standards quickly moved from MPEG-1 (and similar H.261, designed primarily for videoconferencing applications) to MPEG-2, which gave birth to one of the most popular consumer electronics products of all times - video DVD. MPEG-2 was also the backbone of digital TV broadcasting, forever changing the TV experience for hundreds of millions of people. MPEG-4 followed in 1999, powering a rapidly increasing video-streaming traffic on the Internet. Throughout the 1990's, the paradigm of video coding was practically synonymous with the concept of temporal motion-compensated prediction, followed by a spatial Fourier-like transform, popularly called *hybrid coding*. The newest state-of-the art video coding standard - Advanced Video Coder - AVC (also known as H.264 or MPEG4 part 10), is not an exception. Some believe that its superb coding efficiency will eventually define the limit of video compression within the hybrid paradigm. Only future will tell if AVC/H.264 will indeed be the last standard in the hybrid coding line; as of now, no work on the next standard (H.265?) is anticipated to start before 2010.

With a major help from standards, consumer video services - not so long ago dominated by non-interactive standard TV broadcasting - are rapidly expanding. Recent consumer-oriented breakthroughs in the video technology world include VoD (Video-on-Demand), TiVo (time-shifting), HD (High-Definition), IPTV (Internet TV), mobile video (Video iPod, PSP2), and video messaging (video phones).

Excellent compression performance of modern coding systems certainly improved or even enabled many of these services, which gives rise to a logical question - why do we need anything better than a state-of-the-art hybrid video coder? To answer this we need

to look beyond pure compression rates and into the advanced features of tomorrow's video applications and various video distribution scenarios.

Many new video services bring along a shift from the classical distribution media and displays. It is this growing variety of distribution links (dedicated cable, best-effort Internet, fading wireless channel) and displays (mobile phones, PDA's, computer monitors, TVs, HDTVs) that propelled highly-scalable video compression to the top of the desirable video features list. At the same time, emerging applications that involve video sensor networks and other low-processing-power mobile devices underline the need for low-complexity video coding solutions. These two issues, namely *scalable video coding* and *low-complexity video coding* have recently come into focus of video coding community for their importance and real-life implications.

The classical hybrid schemes have difficulties handling each of these two problems. The predictive feedback loop, which is at the heart of every hybrid encoder, is ultimately incompatible with the scalability requirement - it requires some information from decoder's side (i.e., target bit-rate and target resolution) in order to operate properly. In contrast, the premise of scalable coding is that the encoder should operate independently from the decoder.

When it comes to complexity, it is important to note that two major driving forces behind the development of hybrid standards were digital TV and DVD. Both these applications are highly asymmetrical, with content being created and encoded at a limited number of highly capable sites. As a result, a real-time implementation of hybrid encoder on a low-power device (e.g., mobile phone, PDA) became very difficult, if not impossible. In addition to problems of scalability and encoding complexity, the increase in web video-traffic (and, to a lesser extent, wireless video applications) brought attention to the issues of error resilience and network-friendliness.

For the above reasons, and in parallel to the development of sophisticated hybrid video coders, the last decade saw a departure from the classical hybrid schemes. Several new approaches were proposed, including multiple-description coding, object-based video coding,

and distributed video coding, but the scheme based on 3D transforms showed the most promise. It is the only new scheme to successfully compete with state-of-the-art hybrid systems in terms of compression performance, while providing desirable features such as scalability and error resilience. The new 3D framework introduces an intriguing paradigm shift: instead of sequential frame-based predictive processing, this approach is based on spatio-temporal 3D transforms, open-loop non-predictive processing, and embedded quantization and coding.

So far, two distinct approaches to 3D video coding have been proposed. In the first approach, a separable transform is applied directly to the original data without motion compensation before the transform coefficients are coded (Natarajan and Ahmed, 1977; Karlsson and Vetterli, 1988; Lee et al., 1997a; Kim and Pearlman, 1997). The other class of algorithms compensates for motion before applying the transform. Motion compensation can be implemented either through filtering along motion trajectories (Ohm, 1994; Choi and Woods, 1999; Pesquet-Popescu and Bottreau, 2001; Xu et al., 2001; Secker and Taubman, 2003) or by projecting all frames onto a reference coordinate system (Taubman and Zakhor, 1994; Wang et al., 1999) and then applying a temporal transform on the sequence of warped frames.

The early implementations of separable 3D video coding were based on the extension of popular two-dimensional discrete cosine transform (DCT) to three dimensions. Although the DCT had already been successfully deployed in both image and video coding, most of the attempts based on 3D DCT have failed to achieve acceptable coding gains, largely due to the lack of understanding of the role of motion.

Our motivation for the investigation of 3D DCT video coding is rooted in the belief that an efficient use of motion is possible in the transform, instead of the original spatio-temporal domain. At the same time, a shift of motion estimation to the transform domain and an efficient DCT implementation should result in a low complexity encoder. Discussion of our contributions to 3D DCT video coding covers the first third of the thesis. Inspired by an earlier result for the Fourier transform case, we investigate clustering of non-zero

DCT coefficients of a uniformly translating image (global, constant-velocity, translational motion). We use the coefficient clustering to develop a transform-domain method for motion estimation. We propose a new approach to video compression based on the 3D DCT applied to a group of frames (GOF), followed by motion-adaptive scanning of DCT coefficients (akin to "zig-zag" scanning in MPEG coders), their adaptive quantization, and final entropy coding.

Our 3D DCT coding design results in a significant improvement over all previously reported 3D DCT-based solutions and a much lower computational complexity than comparable hybrid systems. However, its coding gain is still significantly lower than that of the newest hybrid coders, such as the AVC/H.264. The main obstacle lies in the overly-simplistic transform-domain motion model that leads to small coding gains when the assumption of uniform translation over several frames is severely violated. We, therefore, turn our focus to another multi-frame approach that involves simultaneous temporal processing of a group of frames through the concept of temporal filtering. This concept is closely related to another discrete linear transform - the discrete wavelet transform (DWT).

Unlike the spectral-only analysis of the DFT or DCT, the DWT provides a multi-scale representation of images and video in the space-frequency domain. Wavelets have already proved to be very useful in image processing - wavelet-based JPEG2000 compression standard outperforms (in terms of compression gain) the DCT-based JPEG standard and offers a wider set of features. These include rate/resolution scalability, region-of-interest (ROI) coding, and random data access. In the light of the wavelet image coders success, an extension of DWT to 3D video coding was expected to quickly reach the performance of hybrid coders with the added benefit of scalability.

However, many early attempts, which applied a separable 3D wavelet transform directly to video data (spatio-temporal volume), failed to produce high coding gains. This soon led to a realization that, in order to fully exploit inter-frame redundancies, the temporal part of the transform must compensate the motion. The new paradigm, based on motion-compensated temporal filtering (MCTF), proved to be a viable alternative to hybrid coding

in terms of compression gain, while offering the crucial scalability property (detailed discussion of scalability is presented in Chapter 4). This concept received a significant boost when a lifting wavelet implementation was introduced in the early 2000's, enabling more efficient motion compensation. The potential of wavelet-based video compression technology can be seen in the broad range of applications; while the main focus remains on video delivery over heterogeneous networks or in error-prone environments, exciting possibilities exist in video archiving, video surveillance, and interactive remote video browsing.

The main reason for the success and excellent compression performance of hybrid video coders lies in the advanced motion modeling, developed for years within the standardization bodies. Not surprisingly, motion preserved its crucial role in a new and significantly transformed wavelet video coding scenery. What is surprising is the fact that the block-based motion model is still predominantly deployed, in spite of replacing the locally-supported DCT by the global DWT. The success of block matching, mostly due to its fast and highly-optimized implementation, is certainly remarkable. Nevertheless, we argue that such a discrete, rigid, and low-order motion model is indeed a less-than-perfect match for wavelet video coding. Therefore, we engage in research on advanced motion models and motion-related wavelet video coding solutions. These topics cover the remaining two-thirds of the thesis. We first examine the problem of optimal motion/transform combination before proposing an adaptive structure for the temporal wavelet transform. We investigate solutions for improved spatial motion modeling and better handling of occluded/exposed areas. In the attempt to provide high-quality motion suitable for wavelet video coding, we design a flexible framework that combines desirable features of two motion models: one based on cubic splines and the other block-based. Our *mixture motion model* facilitates a better compensation and higher coding gains by providing continuous motion representation, high-order spatial modeling of motion, and discontinuity modeling. While such an advanced motion modeling increases the computational complexity, it seems inevitable that future video compression algorithms will have to incorporate more sophisticated and effective motion modeling to achieve better compression performance and richer features.

1.1 Outline of the thesis

This thesis deals with various problems of motion modeling in the context of 3D video coding. In chapter 2 we present basics of two popular transforms that are widely used for image and video processing - the Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). In addition, we describe new and efficient lifting implementation of discrete wavelet transform. This chapter also covers two popular motion models based on block-matching and deformable mesh, along with the practical estimation algorithms, and concludes with the overview of a typical transform video coding system.

Chapter 3 deals in its entirety with video coding based on 3D discrete cosine transform (DCT). DCT spectrum of a globally translating image is analyzed and mathematically described. Derived characteristic footprint is used for fast and efficient video coding by means of motion-adaptive coefficient scanning order and newly proposed 3D quantization. This chapter concludes with a presentation of a low-complexity 3D DCT video coder. Experimental results show subjective and objective performance of 3D DCT coder to surpass that of MPEG-2 and approach the performance of MPEG-4.

Chapter 4 discusses fundamental issues of subband video coding including coding structure, scalability, and the concept of motion-compensated temporal filtering (MCTF). Analysis of the motion invertibility problem, central to optimality of popular lifted MCTF implementation, spans over Chapters 4 and 5. To quantify invertibility problem, the novel "invertibility error" metrics is introduced. Also, proposal of increasingly more complex algorithms for motion inversion. New twist on motion estimation algorithm that takes motion invertibility into account and experimental results demonstrating benefits of advanced inversion conclude the chapter.

Chapter 6 highlights the importance of proper treatment of innovation areas in the context of MCTF. Use of longer temporal wavelet kernels makes "innovations-aware" temporal filtering possible. This adaptive structure is based on indirect modeling of occlusion and exposure areas, which are obtained as an integral part of our new motion estimation

method.

Chapters 7 and 8 deal with advanced spatial modeling of motion. We show how modifications to traditional deformable-mesh topology and motion search algorithm can improve coding efficiency. We analyze the problem of spatial scalability and propose to use the so-called *mixture motion model*. This flexible and efficient model combines superior spatial modeling of cubic splines at lower spatial resolutions with efficient variable-size structure of block displacement at higher resolutions. Practical design solutions regarding spatial scalability and experimental results are presented. Finally, Chapter 9 concludes the thesis, and gives pointer to future work in this field.

Chapter 2

Theoretical background and prior work

2.1 Introduction to video compression

Video coding technology is at the heart of modern communication and entertainment systems that over the last few decades had a dramatic impact on modern society. Digital video applications - such as digital TV, off-line video distribution (DVD), or digital home video - are an integral part of everyday life for hundreds of millions of people around the world. New services, like TV-over-Internet (IPTV), peer-to-peer (P2P) video streaming, and a variety of emerging mobile video applications, promise to deliver even more immersive video experience in the new world of digital video.

The success of modern video applications relies heavily on the underlying compression algorithms. Their goal is to reduce the immense amount of visual information captured by video camera to a manageable size so that it can be efficiently stored, transmitted, and displayed. For example, a digital broadcast of uncompressed standard definition (SD) television signal would require a bandwidth of about 250 Mbps[1] (millions of bits per second) - available at every home! This data rate is hardly attainable by consumer-level communication channels. The ongoing transition to High-Definition TV (HDTV) in the US, additionally increases this bandwidth requirement. It is clear that without highly-efficient compression, deployment of modern video technologies would be impossible.

The ultimate goal of every video coding system is to eventually display moving images in a format that can be perceived by viewers. Subjective video evaluation and the related Human Visual System (HVS) research thus play a significant role in the design of coding

[1]For this calculation, we assume that SD signal has spatial resolution of 720×480 pixels, frame rate of 30 frames per second, and uses 8 bits per each of three color channels.

algorithms. Yet, the subjective quality testing requires extensive preparations, equipment, and significant time; it would be very difficult to repeat it each time there is a change in the coding design. That is why virtually all video coding algorithms are being developed under a different constraint, namely minimizing some objective distortion for a fixed compressed-bit budget. This is equivalent to minimizing the number of bits necessary to attain a preset objective quality level of a reconstructed video. The standard objective measure used in video coding is Peak Signal-to-Noise Ratio (PSNR), defined as PSNR $= 20 \log_{10}(\frac{255}{RMSE})$ - scaled logarithmic ratio of the maximum signal value (typically 255 for 8-bit samples) and Root Mean-Squared Error (RMSE) of all pixels in the entire video sequence.

In addition to the compression performance of a video coder (also referred to as rate-distortion or R-D performance) consideration in the overall evaluation may be given to factors such as coding complexity, scalability support, and memory/latency issues. While typically less important than R-D performance, these factors may be very significant in the assessment of a coding technology for the particular application. For example, the rapid growth of mobile video systems puts more focus on efficient *low-complexity* solutions while the ever-increasing variety of displays and transmission channels can be well-served by a *scalable coding system*.

2.1.1 Hybrid video coding

From the early work on digital video in the 80's, majority of research and commercial development focused on schemes combining *predictive* and *transform* coding into efficient *hybrid coding system*.

In a typical hybrid coder, predictive coding - also known as Differential Pulse Code Modulation (DPCM) - is applied first, along the temporal axis. Initially, inter-frame coding used the same-location pixels from the previous frame to predict pixels in the current frame. In the presence of moving objects, this simple temporal prediction fails to achieve satisfying temporal decorrelation of video. More efficient temporal prediction is possible when spatially displaced pixels from the previous frame are used as predictors. The

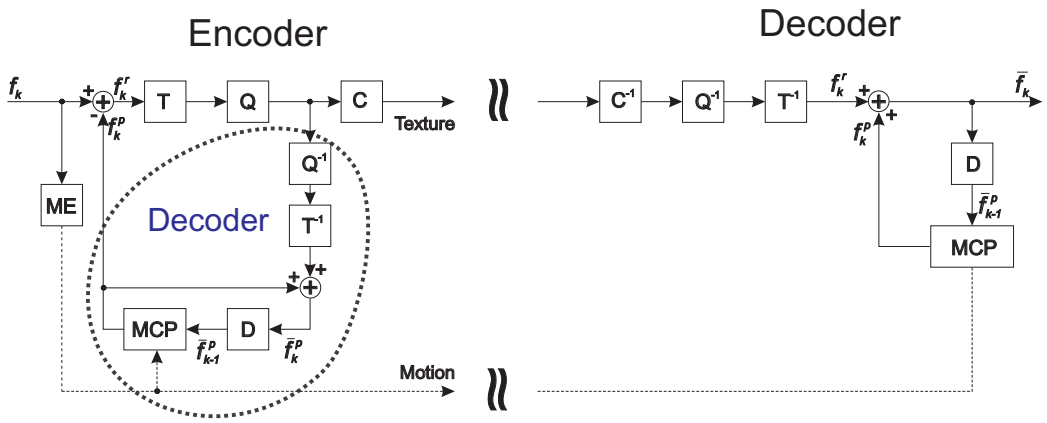


Figure 2·1: Block diagram of a typical hybrid coder.

common name for such motion-adaptive DPCM in the video coding terminology is *motion-compensated temporal prediction* (MCTP). Depending on the quality of motion estimation, excellent temporal decorrelation and corresponding high compression gain can be attained.

In contrast to predictive coding, where signal is coded in the original spatio-temporal domain, transform coding deploys a discrete linear transform to obtain coefficients that are subsequently quantized, encoded, and transmitted. Transform coding has proved very efficient for compression of spatially-correlated sources (Clark, 1985). The choice of transform may vary from traditional transforms with fixed basis functions (e.g., Fourier transform) to newer, signal-adaptive transforms (like Karhunen-Lóeve Transform - KLT). Two most popular linear transforms in image and video processing are Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT), to be discussed in Section 2.3.

After initial motion-compensated temporal prediction, hybrid coder applies a two-dimensio-nal spatial transform. All standard coders use block-based 2D DCT (or its integer approximation, as in AVC/H.264) on the residuals from motion-compensated prediction. Motion vectors used in the prediction stage, quantized transform coefficients, and additional control data are then all entropy coded and transmitted.

The structure of a typical hybrid video coder is shown in Fig. 2·1. The central part of a hybrid encoder is a feedback loop that exploits similarities between two consecutive video

frames. Compression gains mostly come from the reduction in entropy of residual samples when compared to the entropy of original frames. A closer look at the coding model reveals the entire "decoding" structure inside a hybrid encoder. It is important to notice that the prediction is based on quantized frames as the originals are not available at the decoder.

A single encoding cycle starts by forming the residual signal f_k^r from the input video frame f_k and its prediction f_k^p. After the transform (T) and quantization (Q) stages, the feedback loop begins with recreating (now quantized) residual signal. Its addition to the previously used prediction f_k^p results in the decoded frame $\overline{f}_k^p$ - this part of the encoder is identical to the hybrid decoder. In the remaining part of the feedback loop, after a delay stage (D), the decoded frames are used by the Motion-Compensated Prediction (MCP) block to form a new prediction for the next input frame. The motion estimation (ME) block provides motion vectors that control this process. With several modifications, this coding structure is deployed in all video coding standards, from MPEG-1 to AVC/H.264.

2.1.2 3D transform video coding

In contrast to the feedback loop of hybrid schemes, 3D video coding systems rely on the feedforward structure shown in Fig. 2·2. A spatiotemporal transform converts the input video frames into a collection of spatiotemporal coefficients, which are then subject to appropriate quantization and coding. There is a number of ways in which each of these stages may be performed - in this thesis, we primarily deal with the spatio-temporal transform stage.

Very early attempts towards the use of three-dimensional DCT transform were made in the late 70's (Natarajan and Ahmed, 1977; Roes et al., 1977). These and similar approaches have been based on the assumption that when the amount of motion in the video is small, consecutive frames in the video sequence are likely to be highly correlated. While these methods had shown the potential on the conceptual level, they also put forward significant memory requirements for processing 3D volumes of data. High memory and computational requirements of the algorithm proved to be too expensive for successful

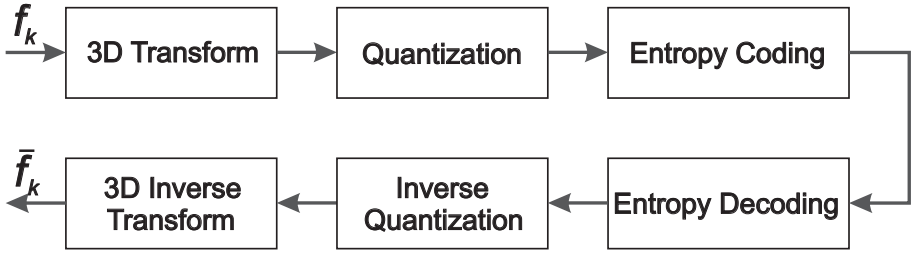


Figure 2·2: Block diagram of a three-dimensional transform coder.

practical implementation at that time.

The use of three-dimensional discrete wavelet transforms for video compression was first proposed in 1988 (Karlsson and Vetterli, 1988) in an attempt to duplicate the gains exhibited by two-dimensional subband structures for image coding. Similarly to the early 3D DCT efforts, no motion compensation was used. Instead, one-dimensional wavelet transform was applied directly to the original video samples straight along the temporal axis - again, under the assumption that the processed samples are highly correlated. Therefore, the low pass subband frames should contain the majority of the signal energy. Further energy compaction was obtained by employing spatial subband decomposition on each of the temporal subbands, resulting in a final 3D subband representation.

Both the DCT and the DWT approaches have been extensively researched over the last decade, resulting in sophisticated 3D video coders that can seriously challenge the performance of hybrid standards.

2.2 Motion modeling and estimation

If video compression is the most important part of every successful consumer video application, its own success directly depends on the efficiency of motion estimation. In this section, we present two most commonly used motion models: block-wise constant model of block-matching and patch-wise affine model of deformable triangular mesh.

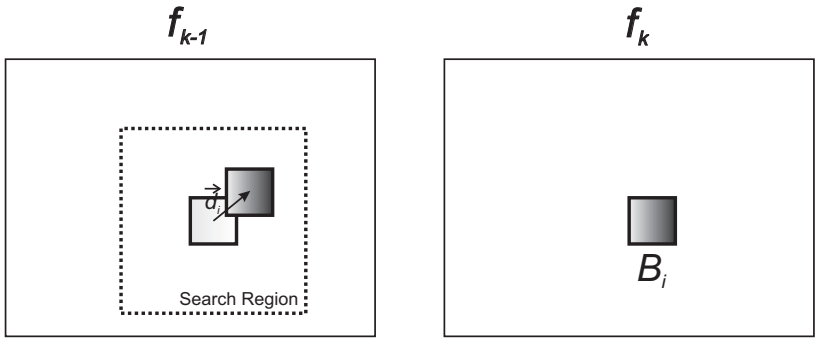


Figure 2·3: Block-matching motion search.

2.2.1 Block-based motion estimation

While the hybrid coding structure (Fig. 2·1) does not set limitations on the motion model, block-based models have been traditionally favored for their natural good match to the block transform of hybrid coders. The simplest block motion model assumes that all pixels within the fixed-size block undergo the same translational motion. It is, therefore, sufficient to use only a single motion vector $\vec{d}$ to describe the displacement of the entire block. This approach works well for as long as the sizes of the blocks that are being matched are relatively small compared to the frame size.

In a typical block-matching algorithm, the current frame f_k is split into fixed-size blocks (Fig. 2·3); for each block, the best match in the previous frame is found, by minimizing a matching criterion. Most of the time, this criterion will either be the Sum of Absolute Differences (SAD) or the Sum of Squared Differences (SSD). Mathematically, block-matching motion estimation for the block B_i that returns the motion vector $\vec{d}_i$ is described as:

$$\vec{d}_i \quad = \quad \text{argmin} \sum_{\vec{x} \in B_i, \vec{d}_i \in SR} C(f_k(\vec{x}) - f_{k-1}(\vec{x} - \vec{d}_i)). \tag{2.1}$$

The distortion measure C is calculated as either absolute value $C(n) = |n|$ (for SAD) or squared value $C(n) = n^2$ (for SSD). In all practical implementations, motion search is limited to a preset "search range" (SR). Its size depends on the nature of motion in the

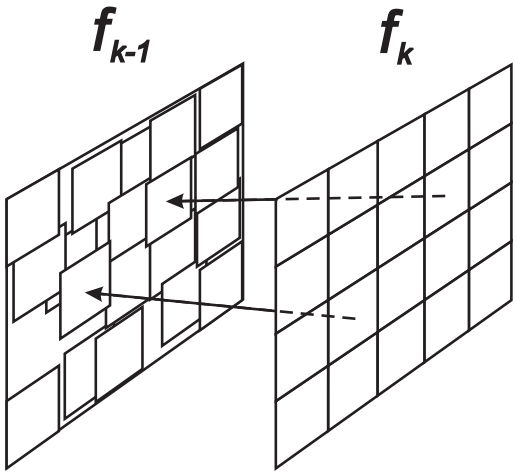


Figure 2·4: Block-based mapping between two consecutive video frames; not all pixels from the reference frame f_{k-1} are used for the prediction of f_k while some are used more than once.

sequence (fast/slow motion) and other design requirements.

The described process is repeated independently for every block in the current frame until all blocks are processed. One effect of such estimation method is that, in general, no one-to-one correspondence exists between the current frame and reference frame, as two adjacent blocks can, and usually will, have different motion vectors. This is illustrated in Fig. 2·4. While every pixel in the current frame f_k has a predictor, not all pixels from the reference frame f_{k-1} are indeed used as predictors. At the same time, some pixels can be used as predictors two or more times.

This fact has more or less significance depending on what temporal processing framework is used. In a typical hybrid coder, motion compensation is purely predictive[2] in nature and neither coding structure nor efficiency is affected. On the other hand, motion-compensated temporal filtering (MCTF) requires access to both motion fields (backward and forward) between the frame pair. This lack of one-to-one correspondence between the pixels of two processed frames has a significant impact on many important MCTF design

[2]Only one motion mapping, typically backward - $M_{k\to k-1}$, is required for proper operation. More on this notation in Chapter 4.

solutions; we will discuss this matter in detail in Chapters 4 through 6.

More recently, in an effort to further improve motion prediction, a modification of the standard block-matching algorithm was proposed in the form of Hierarchical Variable Size Block-Matching (HVSBM). Instead of searching for a single motion vector for the entire 16×16 macroblock, an R-D optimized search has been proposed to find the optimal block-partitioning map (Q_i) of a macroblock B_i and all corresponding motion vectors $\{\vec{d}_i^j\}$. Mathematically, HVSBM is implemented by solving:

$$\begin{aligned} Q_i, \{\vec{d}_i^j\} &= \text{argmin } D_i + \lambda R_i, \\ D_i &= \sum_{\vec{x} \in B_i, \vec{d}_i^j \in SR} C(f_k(\vec{x}) - f_{k-1}(\vec{x} - \vec{d}_i^j(\vec{x}))) \end{aligned} \tag{2.2}$$

for each macroblock B_i, where D_i is the total distortion within a macroblock and R_i represents the cost of coding motion. The regularization parameter λ is used to control the motion rate. First introduced in the MPEG-4 standard, HVSBM is extensively used in both AVC/H.264 and many popular wavelet video coders.

2.2.2 Deformable mesh-based motion estimation

Mesh-based motion models have been shown to be a good alternative to block-based models, such as those used in block matching. In contrast to the block-based model, deformable meshes are capable of capturing more complex spatial affine motion (accounting for rotation, zoom, and shear), in addition to pure translation. In a regular-mesh case, a regular topology is used to partition the current frame. This mesh is subsequently deformed, by *node-point* displacements, into another mesh in the reference frame. This is unlike block matching where each *block* undergoes translational displacement (Fig 2·5).

Two main issues in mesh-based motion estimation are mesh topology and node displacement estimation. Although node topologies can be complex, a very successful approach has been to use triangular patches (Brusewitz, 1990). In this model, displacements of three neighboring nodes define a displacement anywhere within a triangle. For any given location within a patch, this displacement - $d^*(\vec{x})$ - is found by linear interpolation of the

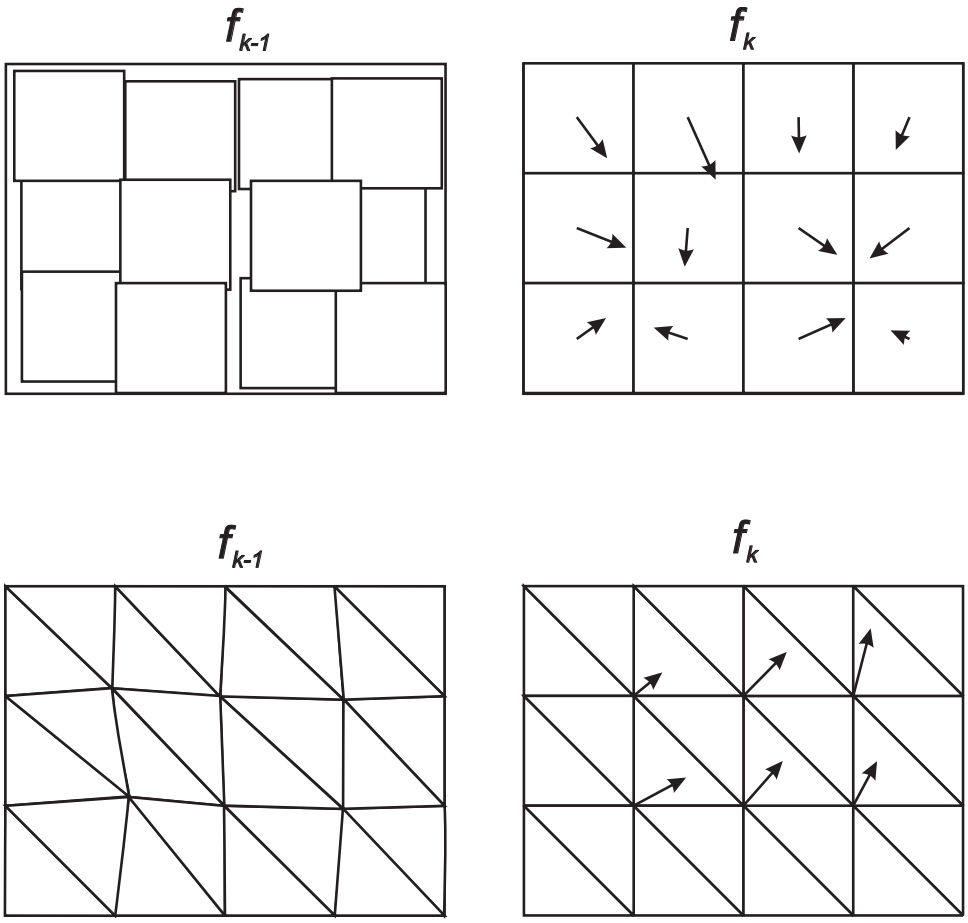


Figure 2·5: Two approaches to motion compensation: a) block displacement of the block-based motion model and b) node displacement of the deformable mesh.

motion vectors at the patch vertices. A triangular mesh can be built from the common square-block partitioning, so that block positions in the current frame are preserved. Mesh nodes can be set at the corners of all square blocks and each block can be divided in half along its diagonal, as shown in Fig. 2·5(b).

As for node displacement estimation, one possibility is an independent estimation of node-point motion vectors, e.g., by maximizing correlation over a small neighborhood of a node. However, since individual nodes are treated independently in this case, motion compensation between node-points is often inadequate. In addition, this approach can cre-

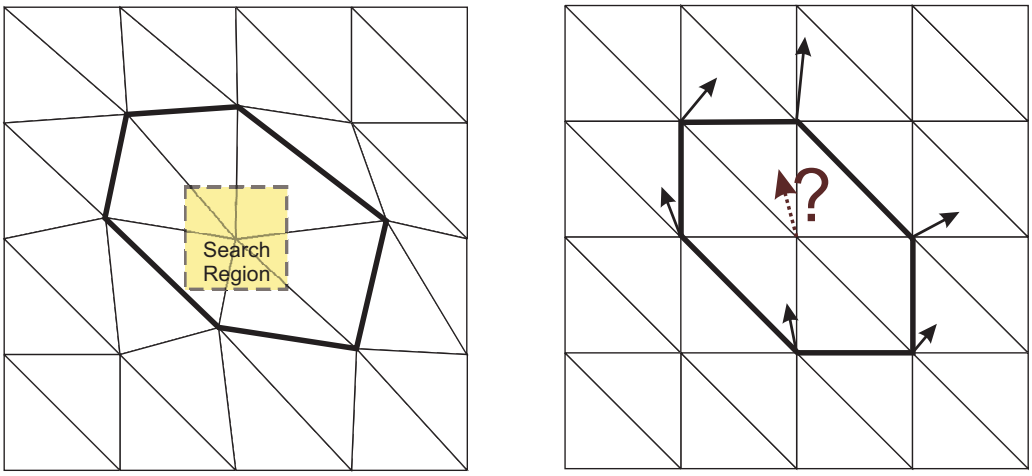


Figure 2·6: Node-point motion estimation using hexagonal refinement.

ate problems with mesh connectivity. To improve this, regularized search-based solutions to node-point motion estimation with triangular and/or quadrilateral regular meshes and spatial transformations have been proposed (Nakaya and Harashima, 1994; Toklu et al., 1996). In particular, Nakaya and Harashima proposed an iterative hexagonal matching procedure where the motion vector at a node point is estimated by local minimization of the prediction error. In each iteration, the motion vector d_i corresponding to the current node i is perturbed in order to minimize local intensity distortion D_i over six triangular patches formed by the current node and its six neighbors (Fig 2·6):

$$D_i = \sum_{j=1}^{6} C_j(f_n(\vec{x}) - f_{n-1}(\vec{x} - \vec{d^*}(\vec{x})),\ \vec{d_i} \in SR,\ \vec{x} \in \cup\, T_j.$$

As mentioned above, in each of the six triangular patches T_j, the motion displacement $\vec{d^*}(\vec{x})$ is computed using linear interpolation from two known node-point motion vectors and the current value of the motion vector $\vec{d_i}$ that is being estimated. It is easy to see that this assures motion continuity between triangular patches.

This single resolution method was later extended (Toklu et al., 1996) by employing

a hierarchy of regular meshes, such that motion estimation on a coarse mesh provides initialization for the next (finer) level of the mesh. For completeness, we note that a number of advanced content-based mesh motion estimation algorithms exist - extensive review is presented in (Altunbasak et al., 2003). However, given typical computational complexity of content-based mesh motion estimation and additional overhead needed for the transmission of mesh topology itself, this approach is not as practical as the regular mesh method for practical compression applications.

2.3 Common transforms

In this section we present two linear transforms that are predominantly used in video coding, the discrete cosine transform (DCT) and the discrete wavelet transform (DWT).

2.3.1 Discrete cosine transform (DCT)

The discrete cosine transform has been a driving force behind practically all image and video coding standards in the last 20 years. First introduced by Ahmed *et al.* in 1974 (Ahmed et al., 1974), it has been used extensively ever since because of its close-to-optimal performance on natural images. In general case, there are 8 types of the discrete cosine transform and also 8 types of the discrete sine transform (DST), as defined by Wang (Wang and Hunt, 1985), where different transform types correspond to different possible symmetrical extensions at both ends of an input sequence. Together, these transforms are often called discrete trigonometric transforms (DTT). Out of these 16 types, the DCT of type 2e ($\mathcal{DCT}_{2e}$) is by far the most popular DTT used in image and video processing, due to its excellent energy compaction properties. One-dimensional $\mathcal{DCT}_{2e}$ is defined as follows:

$$\mathcal{DCT}_{2e}\{u[n]\} = \widetilde{U}[k] = \sqrt{\frac{2-\delta[k]}{N}} \sum_{n=0}^{N-1} u[n] \cos \frac{\pi(2n+1)k}{2N}, \quad k = 0, ..., N-1.$$

In order to compute a multi-dimensional (MD) DCT, 1D transform defined above is applied separately in each direction. An illustration of basis functions of two dimensional 8×8

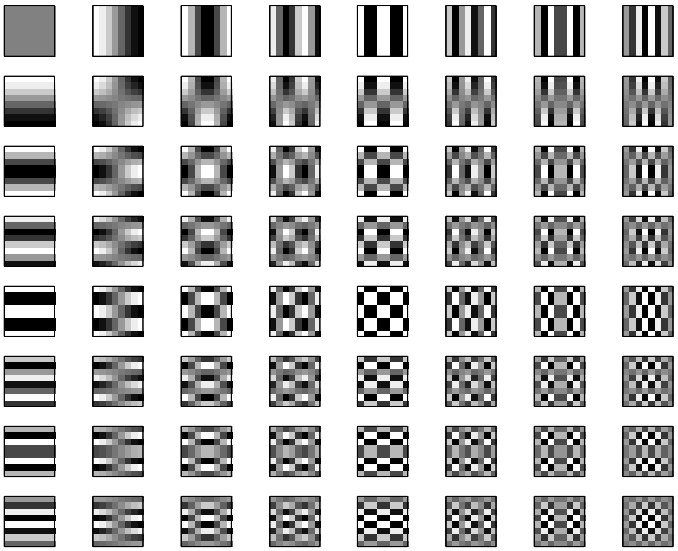


Figure 2·7: 64 basis functions of 8×8 2D DCT.

DCT is shown in Fig. 2·7.

The $\mathcal{DCT}_{2e}$ is closely related to the DFT; its coefficients can be computed by the application of DFT to a symmetrically extended signal using "half-point" symmetric extension - more on symmetrical extensions in Chapter 3. It is straightforward to show (Pitas, 2000) that the coefficients $\widetilde{U}$ of M-dimensional $\mathcal{DCT}_{2e}$ of a signal are related to DFT coefficients U^s of M-dimensional "half-point" symmetric extension of this signal through the following relationship:

$$\widetilde{U}[k_1, k_2, ..., k_M] = \left(\prod_{i=1}^{M} \frac{1}{2} \sqrt{\frac{2 - \delta[k_i]}{N_i}} e^{-j\frac{\pi k_i}{2N_i}} \right) U^s[k_1, k_2, ..., k_M]. \tag{2.3}$$

where $\widetilde{U}[k_1, k_2, ..., k_M] = \mathcal{DCT}_{2e}\{u[n_1, n_2, ..., n_M]\}$ and also $U^s[k_1, k_2, ..., k_M] = \mathcal{F}_{DFT}\{u^s[n_1, n_2, ..., n_M]\}$.

The above relationship is the basis for efficient implementation of the DCT using FFT algorithms (Pitas, 2000). Also, the excellent energy compaction properties of the DCT can be largely explained using this relationship; a symmetrically and periodically extended signal has less high-frequency content than a periodically extended signal, due to the

absence of discontinuities at each period's ends (caused in the latter case, e.g., by different signal values of the first and last signal samples). Note that in the remainder of the thesis when we refer to DCT, we mean DCT of type 2e as defined above.

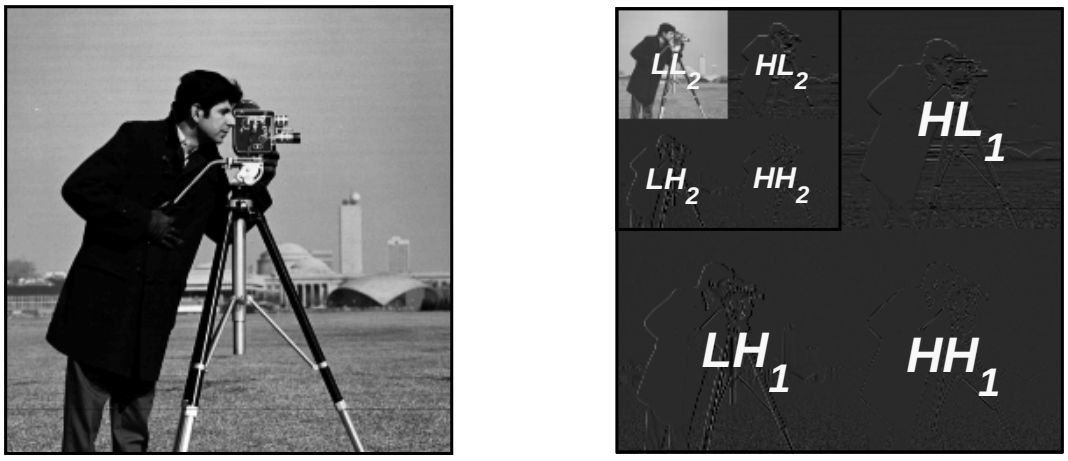
2.3.2 Discrete wavelet transform (DWT)

Wavelet theory fills the void between pure spatio-temporal and pure spectral signal processing. Being well-matched to local stationary properties of natural image and video signals, wavelets have seen rapidly increased usage in image and video compression algorithms as an alternative to the DCT. The most recent still image coding standard JPEG2000 is based on wavelets.

Originally conceived as a continuous transform in the mathematical community, wavelets quickly became a popular signal processing tool among engineers because of the fact that the DWT can be implemented through multi-stage linear subband filtering. For 2D spatial processing, two 1D DWTs are applied separably along x and y axis; extension to 3D is straightforward. An example of two-level 2D Haar DWT with standard subband labels is given in Fig. 2·8.

When wavelets are used for a motion-compensated temporal video processing, a temporal transform is usually referred to as motion-compensated temporal filtering. As a temporal decorrelation method, MCTF directly replaces classical frame-based motion compensation. It is important to note that MCTF can be successfully deployed not only in 3D video coders but also in standard hybrid coders - recently, this has been investigated by MPEG for use in the existing coding standards.

The Haar wavelet (a.k.a. "lazy" or D2 wavelet) is the simplest possible wavelet. Despite its simple structure, Haar wavelet is still frequently used for temporal processing in many newest wavelet-based coders (often in combination with 5/3 Le Gall's DWT). For one-dimensional discrete input signal $x[n]$, high and low pass subbands of the Haar DWT are



(a) (b)

Figure 2·8: a) *Cameraman* image, b) its two-level 2D Haar DWT

obtained using the following analysis equations:

$$\begin{aligned} h[n] &= x[2n+1] - x[2n], \\ l[n] &= \frac{1}{2}x[2n] + \frac{1}{2}x[2n+1]. \end{aligned}$$

These subbands can be used to reconstruct $x[n]$ via the synthesis operation:

$$\begin{aligned} x[2n] &= l[n] - \frac{1}{2}h[n], \\ x[2n+1] &= \frac{1}{2}h[n] + l[n]. \end{aligned}$$

The other popular and widely used wavelet kernel is Le Gall's 5/3 (Gall and Tabatabai, 1988) whose analysis equations are:

$$\begin{aligned} h[n] &= x[2n+1] - \frac{1}{2}(x[2n] + x[2n+2]) \\ l[n] &= \frac{3}{4}x[2n] + \frac{1}{4}(x[2n-1] + x[2n+1]) - \frac{1}{8}(x[2n-2] + x[2n+2]), \end{aligned}$$

Longer DWT kernels are also frequently used in image and video processing, especially for spatial decomposition. For example, Daubachies 9/7 DWT is used as a spatial transform in JPEG2000. For the purpose of temporal processing, Haar and 5/3 kernels remain the

most popular wavelets, mainly because of latency and memory issues. In addition, efficient application of the longer temporal wavelet kernel requires accurate motion tracking over larger number of frames, which itself might be difficult. As the focus of this thesis is mostly on temporal video processing, most of discussions will be limited to Haar and 5/3 DWT.

2.3.3 Lifting implementation of the wavelet transform

In the previous section, both the Haar and 5/3 discrete wavelet transforms were presented in the so-called *transversal form*. In 1996, Sweldens (Sweldens, 1996) showed how every DWT kernel can be factored into lifting steps. Not only is the lifting wavelet implementation more efficient than the transversal; its structure is also much more suitable for the temporal wavelet processing. Lifting guarantees perfect reconstruction of motion-compensated temporal DWT, regardless of particular motion model used. A detailed analysis of equivalence between the two implementations and benefits of using one vs. the other is presented in Chapter 4.

Every analysis lifting stage consists of two distinct steps: *prediction* and *update*. In the prediction stage, odd input samples are *predicted* from even samples to obtain the high-pass subband; the update step *updates* even input samples with samples from this high-pass subband, resulting in low-pass subband. Details on the construction of lifting steps for an arbitrary DWT can be found in (Daubechies and Sweldens, 1998). While, in the general case, such construction might be complex, once lifting steps are found, it is straightforward to demonstrate the equivalence of these two DWT implementations.

For two DWT kernels that are the topic of our investigation, the lifting steps can be written as:

$$\begin{aligned} prediction: \ h[n] &= x[2n+1] - x[2n], \\ update: \ l[n] &= x[2n] + \frac{1}{2}h[n], \end{aligned}$$

for Haar DWT, and:

$$\begin{aligned} prediction: \ h[n] &= x[2n+1] - \frac{1}{2}(x[2n] + x[2n+2]), \\ update: \ l[n] &= x[2n] + \frac{1}{4}(h[n-1] + h[n]), \end{aligned}$$

for 5/3 DWT.

2.4 Overview of prior research

In this section we review recent advances in video coding. We specifically focus on: 1) motion representation, estimation, and compensation and 2) 3D video coding, as these two topics are of particular interest to this thesis.

Over the last 20 years, digital video coding has been almost unanimously identified with the paradigm of hybrid coding (also referred to as frame-based video coding) discussed earlier in this chapter. This comes as no surprise as all video compression systems standardized to date belong to the hybrid coding class. Their development, under the coordination of the Motion Picture Experts Group (MPEG), dealt with different applications, target bit-rates, and quality levels over time. First in this family, MPEG-1 (standardized in 1992) was designed for digital video storage on CD-ROM media, with approximate VHS quality and encoding at 1.5 Mbps. Soon after, MPEG-2 was released with great success. With the goal of offering consumer-level video quality, it supported coding at relatively high bit-rates (from 2 to 20 Mbps, typically 5Mbps) and found application in home entertainment systems, like digital TV and DVD. MPEG-2 was standardized in 1994. In late 1990's, the next coding standard for the fixed and mobile web, MPEG-4, was proposed in an effort to provide more flexible coding, fine-grained scalability, and support for streaming applications and mixed natural/virtual scenery. MPEG-4 became an international standard in the first months of 1999. The AVC/H.264 is the most current, state-of-the-art standard delivering excellent quality across wide bandwidth spectrum, from video conferencing and 3G mobile multimedia to high-definition TV. It offers over 50% bit-rate savings compared to MPEG-2 at high bit-rates and significant coding gain of 3-4dB over MPEG-4 at lower

bit-rates.

Advances in motion estimation and compensation deserve significant credit for this rapid progress. The block-based motion model has been predominantly used in all coders from its inception in the 1980's until modern days. While the basic block-wise constant motion model stayed the same, many changes were introduced into it over time. The demand for more sophisticated prediction drove the block-model development in three directions: 1) towards use of higher accuracy of motion vectors, 2) use of variable block sizes, and 3) deployment of bidirectional prediction.

The early standards (MPEG-1, H.261) performed prediction using full-pixel precision of motion vectors. It was soon realized that motion in natural video sequences is rarely aligned with integer-pel positions; higher motion vector accuracy soon followed. With the rapidly developing DSP chips, enabling real-time implementation of more computationally demanding sub-pixel motion estimation, an extension to half-pel accuracy was implemented in MPEG-2. With the emergence of powerful CPUs, the first real-time software implementations became available in mid-1990's. Another improvement to 1/4 pel accuracy was proposed in MPEG-4 and is currently being used in the AVC/H.264 standard. Some of the newest coders (e.g., MC-EZBC (Chen and Woods, 2004), MSRA coder (Xiong et al., 2005c)) deploy even more accurate motion fields, down to 1/16 pixel precision.

In all video coders up to MPEG-2, motion compensation was performed using macroblocks of fixed size, typically 16×16. As an improvement, variable block sizes were introduced in MPEG-4 for better local motion modeling and adaptation to object boundaries. In the AVC/H.264 specification, the size of motion blocks ranges from 16×16 down to 4×4, with 7 different sizes - 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4.

Finally, instead of prediction from a single frame (P-frames) or from a fixed linear combination of two frames (as in B-frames), multi-hypothesis approach has been proposed (Flierl et al., 2002). Coupled with a multi-frame prediction (that uses a buffer of previous frames to find the best predictors), it provided significant coding gains for different

scenarios[3] at the cost of increased complexity. Extensions of this method found way into the AVC/H.264 standard, most notably thorough generalized B pictures (Flierl and Girod, 2003).

Concurrently with the development of industry-driven video standards, various alternative concepts were explored over the last 20 years. From the pool of "non-standard" schemes, 3D video coding concept emerged as the most serious challenger to hybrid video coding.

Early 3D video coding attempts focused on applying the still image transform coding concept to video coding, by simply extending the popular discrete cosine transform to three dimensions. Most of the works (Natarajan and Ahmed, 1977; Roes et al., 1977; Yeo and Liu, 1995; Westwater and Furht, 1996; Chan and Siu, 1997; Lee et al., 1997a) used the separable 3D DCT directly on the original data (without motion compensation) before coding the obtained coefficients. Various coefficient scanning (plane-by-plane, 3D zig-zag, parabolic) and quantization strategies were proposed. Also, variable-size data cubes were used to adaptively match the video content and align volume boundaries with spatio-temporal discontinuities (Furht et al., 2003). Other algorithms used simple motion activity estimators to classify each data volume as either no-motion, low-motion, or high motion - different strategies were then used for coding of each case (Lee et al., 1997b), but none of the proposed 3D DCT solutions truly accounted for image sequence motion. While computationally very efficient, these coders failed to compete with contemporary hybrid video coders in terms of the coding performance.

Following in the footsteps of DWT-based still image coders, wavelet video coders (Karlsson and Vetterli, 1988) emerged in the late 1980's. All video coders deploying wavelets can be divided into three classes: 1) hybrid-like MCTP+2D DWT coders, 2) 2D DWT + MCTP coders, and 3) 3D DWT coders with motion-compensated temporal filtering. We briefly discuss the first two schemes before moving on to the 3D DWT scheme that is of

[3]Especially for scenes with rapid repetitive flashing, back-and-forth scene cuts, or uncovered background areas.

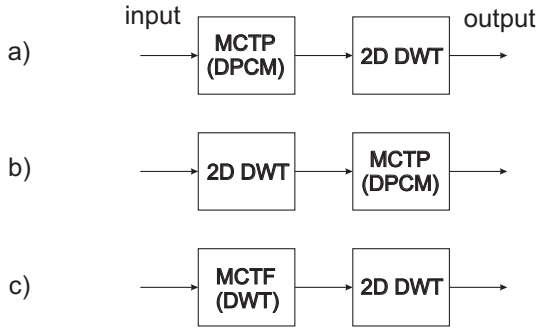


Figure 2·9: Video coders employing the discrete wavelet transform.

the greatest interest to our work.

Within the MCTP+2D DWT concept (Zhang and Zafar, 1992; Marpe and Cycon, 1999), two-dimensional wavelet transform is applied to the motion-compensated residual frame, instead of the DCT (Fig. 2·9a). Because of motion compensation, pixels in the residual frames are less spatially correlated than in the original frames and the spatial wavelet transform is not very effective in representation of those residual frames. Thus, the MCTP+2D DWT method does not offer noticeable improvement over standard MC-DCT approaches and, more importantly, does not offer new features (e.g., scalability) that would justify its use.

The second approach, 2D DWT+MC, applies 2D DWT to each original video frame before performing motion compensation in the wavelet domain (Fig. 2·9b). Due to the fact that commonly used wavelet transforms are often critically sampled, shift-variance associated with these transforms limits the effectiveness of motion compensation. For example, a single pixel shift between two consecutive video frames in the spatial domain results in a different shift between corresponding high-subbands of these two frames, which prevents effective motion compensation in wavelet domain and lower coding efficiency of this method.

Finally, in the third case of separable 3D wavelet video coding (Karlsson and Vetterli, 1988; Kim and Pearlman, 1997; Kim et al., 2000; Xu et al., 2002), a temporal-domain wavelet transform is usually applied before 2D wavelet transform on each resulting video

frame (Fig. 2·9c). This scheme is typically referred to as "t+2D". With a 3D wavelet representation, it is possible to achieve coding scalability in rate, quality, and resolution. However, due to the object motion in the scene, spatially co-located pixels across frames are usually misaligned. Thus, in order to take full advantage of the temporal-domain wavelet transform and hence achieve high coding efficiency, wavelet filtering has to be performed along motion trajectories. Because of the problems involved with the coupling of motion estimation and the wavelet transform, the bottleneck of the efficient 3D wavelet video coding lies in motion estimation and compensation.

In an attempt to incorporate motion estimation into 3D wavelet video coding, Ohm (Ohm, 1994) proposed to use block-matching - similar to that used in standard video coders - for temporal video filtering. The important contribution of Ohm's scheme was a coding design that supported perfect reconstruction of motion-compensated Haar wavelet filtering. This was achieved by paying special attention to connected/unconnected pixels. However, Ohm's scheme failed to achieve perfect reconstruction with motion alignment at any sub-pel resolution and exhibited modest compression gains. We note that this classic scheme has been only recently generalized to handle sub-pel motion vector accuracies (Hsiang et al., 2004).

Concurrent to Ohm's efforts, Taubman and Zakhor (Taubman and Zakhor, 1994) pre-distorted the input video sequence by translating frames relative to one another before the wavelet transform so as to compensate for camera pan motion. Later, Wang *et. al* (Wang et al., 1999) used the mosaicing to warp each video frame into a common coordinate system and applied a shape-adaptive 3D wavelet transform on the warped video. Both of these schemes adopt a global motion model that is not capable of enhancing the temporal correlation among video frames in sequences with local motion.

An important development for wavelet-based image and video coding was the discovery of *zerotrees* for coding of 2D DWT coefficients. The zerotree approach is still commonly used as it achieves high performance and a progressive bit-stream by exploiting the inter-subband dependency among insignificant wavelet coefficients. This is mainly achieved by

reorganizing such coefficients in space-scale trees. The zerotree idea was first introduced by Shapiro's embedded zerotree wavelet (EZW) (Shapiro, 1993) and then reformulated with the set partitioning in hierarchical trees (SPIHT) algorithm (Said and Pearlman, 1996). Besides the zerotree concept, other techniques that exploit statistical dependencies in the wavelet domain have been recently proposed. For example, the embedded block coding with optimized truncation (EBCOT) algorithm (Taubman, 2000), adopted in the JPEG2000 standard (JPEG2000, 2000), combines layered block coding, R-D optimization, and context-based arithmetic coding in an efficient and highly scalable way. Moreover, with the embedded zero-block coding (EZBC) algorithm (Choi and Woods, 1999), the positive aspects of quad-tree partitioning and context modeling of wavelet coefficients are well combined, too.

The breakthrough in wavelet video coding came in 2001, when the concept of motion-compensated lifted DWT was introduced (Pesquet-Popescu and Bottreau, 2001; Luo et al., 2001; Secker and Taubman, 2001); floodgates for a dramatic increase in research activity were open. The importance of lifting is that it guarantees a perfect reconstruction of MC-DWT for an arbitrary motion model or accuracy. As a consequence, advanced motion estimation algorithms developed for hybrid coders could be easily deployed in the new context. The first implementation of sub-pixel (half-pel) motion compensation in Haar lifted DWT by Pesquet-Popescu and Bottreau (Pesquet-Popescu and Bottreau, 2001) demonstrated the advantage of lifting over classical transversal processing. Soon after, the first 5/3 biorthogonal wavelet filter with half-pel motion accuracy was presented (Luo et al., 2001). Secker and Taubman used both the Haar and 5/3 filters in combination with 1/2-pel resolution motion estimation (Secker and Taubman, 2001). Numerous follow-up works (Secker and Taubman, 2003; Chen and Woods, 2004; Flierl and Girod, 2004) demonstrated the substantial advantage of 5/3 over Haar filter for temporal wavelet transformation.

Inspired by the promising results of wavelet video coders and lively research activity in both academic and industrial environment, MPEG formed an ad-hoc group for exploration of wavelet video coding in 2004. Its focus has been on the efficient implementation capable

of competing with the AVC/H.264. The group has investigated various design issues, such as use of 9/7 or longer filters, ultra-accurate motion estimation (down to 1/16 pel), adaptive and low-complexity implementation, scalable motion coding, and spatial scalability problem. Most important contributions to the recent wavelet video development appeared in (Chen and Woods, 2002; Ohm, 2002; Secker and Taubman, 2003; Chen and Woods, 2004; Golwelkar, 2004).

Several popular wavelet-based video coding platforms have been made available to the public; they are used as testbeds by the wide community of researchers. Motion-Compensated Embedded Zero-Block Coder (MC-EZBC) (Chen and Woods, 2002), proposed by Chen and Woods, has become widely used for its excellent performance. In the original MC-EZBC, motion estimation was performed on each pair of frames in a hierarchical fashion down to 1/8-pel accuracy before going through a motion-compensated lifting-based Haar transform. Additional levels of temporal wavelet decomposition were performed on the low-pass frames by following the same procedure. In early versions of MC-EZBC coder, the Haar filter did not fully exploit the long-term correlation across video frames. In addition, motion vectors at each temporal decomposition layer were estimated and coded independently without exploiting the cross-layer correlations. A newer and improved MC-EZBC (Golwelkar, 2004) makes use of the 5/3 filters in conjunction with improved motion vector coding with special attention on spatial scalability performance. Recent versions of MC-EZBC perform on-par with the AVC/H.264 standard for some sequences.

Another popular coder (currently used as a verification model in MPEG's wavelet video coding efforts) was developed by Microsoft Research Asia (Xu et al., 2001). It is based on Barbell lifting (Xiong et al., 2004), energy distributed update steps (Feng et al., 2004), and 3D Embedded Subband Coder with Optimized Truncation - 3D ESCOT (Xu et al., 2001). In addition to Microsoft, notable contributors to this coder include RWTH Aachen, Samsung, Panasonic, ENST-Paris, and University of Brescia, Italy.

We end this chapter with a few words on the outlook for 3D video coding. As with

all other new technologies, various non-technical issues play important roles in adoption rate of the new systems - combined with objective performance, they will determine the outcome and ultimate success of 3D video approach. With the existing video infrastructure, investment in MPEG-2 technology, and aggressive push for the deployment of AVC/H.264 across the industry, many believe that 3D video coders will face an uphill battle before finding their place in industry-supported applications. The lesson of JPEG2000, which still struggles to generate strong industrial friction more than 5 years after its induction and despite the obvious technological advantage over JPEG, is very important. It is clear that external industrial factors, such as licensing fees, cost of technology shift, or increased computational complexity, are indeed as important when it comes to adoption as the technology itself; only time will tell what is the future of 3D video technologies.

Chapter 3

Motion analysis and video coding in 3D DCT domain

3.1 Introduction

In the early days of separable 3D video coding, extending the popular 2D discrete cosine transform (DCT) to three dimensions was in the focus of many research efforts. While computationally efficient, most 3D DCT schemes failed to match the performance of contemporary hybrid coders, largely due to the lack of understanding of the motion role. In this chapter, we demonstrate that efficient low-complexity 3D DCT video coding is indeed possible - its key is in the effective use of motion in the transform domain.

We start this chapter by revisiting the well known result on motion modeling in the Fourier transform domain in Section 3.2. In Section 3.3, we show how, similar to the DFT case, non-zero DCT coefficients of a uniformly translating image (global, constant-velocity, translational motion) aggregate around a folded plane in the spatio-temporal "frequency"[1] space. Based on this observation, we develop a method to estimate motion in a 3D block of pixels in Section 3.4. We then propose a new approach to video compression based on 3D DCT applied to a group of frames in Section 3.5. Building blocks of our coder are: motion-adaptive scanning of DCT coefficients (akin to "zig-zag" scanning in MPEG coders), adaptive coefficient quantization, variable-size volume processing, and entropy coding. We base our new coefficient scanning pattern on geometry of a spectrum model for uniformly trans-

[1]Note that although the notion of frequency applies to the FT domain and is not accurate in the DCT domain, we nevertheless will use this notion in the DCT context from now on with the understanding that a high "frequency" in the DCT domain corresponds to a high number of zero crossings of image intensity per unit distance.

lating image and demonstrate its advantage over all previously proposed scans, such as the 3D zig-zag scan (Yeo and Liu, 1995). Our coding design also incorporates new 3D quantization table (*quantization volume*), which accounts for spatio-temporal contrast sensitivity of the human visual system. Section 3.6 presents experimental results showing that the motion-adaptive scan consistently achieves better energy compaction than "motion-blind" scans. We finally compare compression performance and computational complexity of our coder to that of other 3D DCT coders and standard hybrid coders.

3.2 Interpretation of uniform image translation in the Fourier domain

A translational model is the most frequently used motion model in video coding; motion is assumed to be translational between each two consecutive frames, which works well in frame-based predictive coders of the MPEG coding family. A straightforward extension of this model to 3D yields motion that is uniform over the entire group-of-frames (GOF). While inevitably leading to a less sophisticated motion representation, the assumption of uniform translation is certainly reasonable for shorter temporal support of the processed 3D volume.

The analysis of a uniform translational motion in the Fourier transform domain was first presented by Jacobson and Wechsler (Jacobson and Wechsler, 1987). Let $u_0(x_1, x_2)$ be the continuous intensity of a still image as a function of continuous spatial coordinates (x_1, x_2). Similarly, let $u(x_1, x_2, x_3)$ be the continuous intensity function of a time-varying image, where x_3 denotes the time coordinate. A uniform translation of intensities u_0 at all spatial coordinates with constant velocity $[v_1, v_2]^T$ results in a time-varying image function:

$$u(x_1, x_2, x_3) = u_0(x_1 - v_1 x_3, x_2 - v_2 x_3). \tag{3.1}$$

Assuming, for the purpose of the Fourier-transform case, that the image u_0 has dimensions which are infinite, it is straightforward to show that the following relationship holds:

$$u_0(x_1 - v_1 x_3, x_2 - v_2 x_3) = [u_0(x_1, x_2)\delta(x_3)] * \delta(x_1 - v_1 x_3, x_2 - v_2 x_3), \tag{3.2}$$

where "$*$" denotes continuous 3D convolution operator while $\delta(\cdot)$ is the Dirac impulse (1D or 2D, depending on the number of arguments). Applying the continuous-space Fourier transform (CSFT) to each term of the above convolution one obtains (Jacobson and Wechsler, 1987):

$$\mathcal{F}\{u(x_1, x_2, x_3)\} = U_0(f_1, f_2)\delta(f_1 v_1 + f_2 v_2 + f_3). \tag{3.3}$$

Clearly, the Fourier spectrum of a uniformly-translating image is limited to the plane $f_1 v_1 + f_2 v_2 + f_3 = 0$ in the spatio-temporal frequency space (f_1, f_2, f_3). Such a unique energy footprint permits identification of the direction and amplitude of motion. Motion estimation methods based on velocity polling functions (Jacobson and Wechsler, 1987) or directional motion-sensitive spatio-temporal filters (Heeger, 1988) have been proposed to this effect. However, if the motion is not translational or velocity varies in time, no clear plane can be identified in the frequency space. The larger the departure from the translational nature of motion, the more fuzzy is the spectral-occupancy plane (more ambiguity).

In practice, the time-varying image u is sampled, and the CSFT needs to be replaced by the discrete-space Fourier transform (DSFT), a multi-dimensional equivalent of the discrete-time Fourier transform (DTFT) (Oppenheim et al., 1999). Let $u[n_1, n_2, n_3]$ be an image sequence, i.e., a sampled time-varying image. Similarly to the continuous case, we can construct the sequence $u[n_1, n_2, n_3]$ by uniformly translating a still continuous image $u_0(x_1, x_2)$ by 2D displacement (d_1, d_2) between each two consecutive frames, and then sampling it. Thus, we have $u[n_1, n_2, n_3] = u_0(n_1 - d_1 n_3, n_2 - d_2 n_3)$. Note, that although u_0 is a continuous function, it is sampled at points $(n_1 - d_1 n_3, n_2 - d_2 n_3)$. By replacing the continuous convolution with its discrete-time equivalent (convolution sum), the CSFT – with the DSFT, and again assuming an infinite-size image (to ignore boundary effects), it follows that:

$$\mathcal{F}_{DSFT}\{u[n_1, n_2, n_3]\} = U_0(f_1, f_2)\delta(f_1 d_1 + f_2 d_2 + f_3), \tag{3.4}$$

where $\mathcal{F}_{DSFT}$ denotes the DSFT, and $U_0(f_1, f_2) = \mathcal{F}_{DSFT}\{u_0(n_1, n_2)\}$. Obviously, due to the periodic nature of the DSFT, this spectral plane is also periodic. A similar relationship holds for the DFT and circular convolution applied to a finite-size image u_0, except that Kronecker delta replaces the Dirac impulse while integer indexes k_1, k_2, k_3 replace frequencies f_1, f_2, f_3:

$$\mathcal{F}_{DFT}\{u[n_1, n_2, n_3]\} = U_0[k_1, k_2]\delta[k_1 d_1 + k_2 d_2 + k_3]. \tag{3.5}$$

Note that $\delta[k_1 d_1 + k_2 d_2 + k_3]$ above is a sampled version of $\delta(f_1 d_1 + f_2 d_2 + f_3)$ from equation (3.4). In other words, it describes samples of the plane $f_1 d_1 + f_2 d_2 + f_3 = 0$ at discrete frequencies (k_1, k_2, k_3). In general, for integer values of d_1 and d_2 the *density* of these samples will be higher, whereas for non-integer values it will be lower. In fact, for non-rational displacements or for such (d_1, d_2) that $k_1 d_1 + k_2 d_2 + k_3 = 0$ only for (k_1, k_2, k_3) outside of the image sequence domain, there may be no plane samples in the DFT domain except $(k_1, k_2, k_3) = (0, 0, 0)$.

In Fig. 3·1, we illustrate the spectral behavior of FT, DSFT, and DFT in the 2D case, i.e., the case of a translating 1D intensity profile. Such a moving profile forms a 2D image whose various transforms are graphically depicted in Fig. 3·1. Fig. 3·1(a) illustrates a 2D version of equation (3.3), showing that the spectrum of a signal undergoing uniform translation is limited to a line of spectral occupancy in the FT space. If the 2D image formed by the moving profile is sampled on a lattice (Dubois, 1985), its DSFT[2] leads to a spectrum shown in Fig. 3·1(b); the spectrum of the underlying continuous signal is now replicated on the reciprocal lattice. In this paper, we consider only orthogonal lattices (rectangular sampling grids). Since the DFT is a sampled version of the DSFT (assuming the region of support of the DFT equals that of the signal on which the DSFT is computed), Fig. 3·1(c) shows the same, although sampled, spectral content as Fig. 3·1(b). Note, that

[2]Note that the "granular" appearance of the spectral line in Figs. 3·1.a and 3·1.b, despite the continuous nature of frequency in FT and DSFT, is due to the fact that the FT was simulated by DFT of a high-resolution signal and the DSFT – by adding together periodic repetitions of the FT.

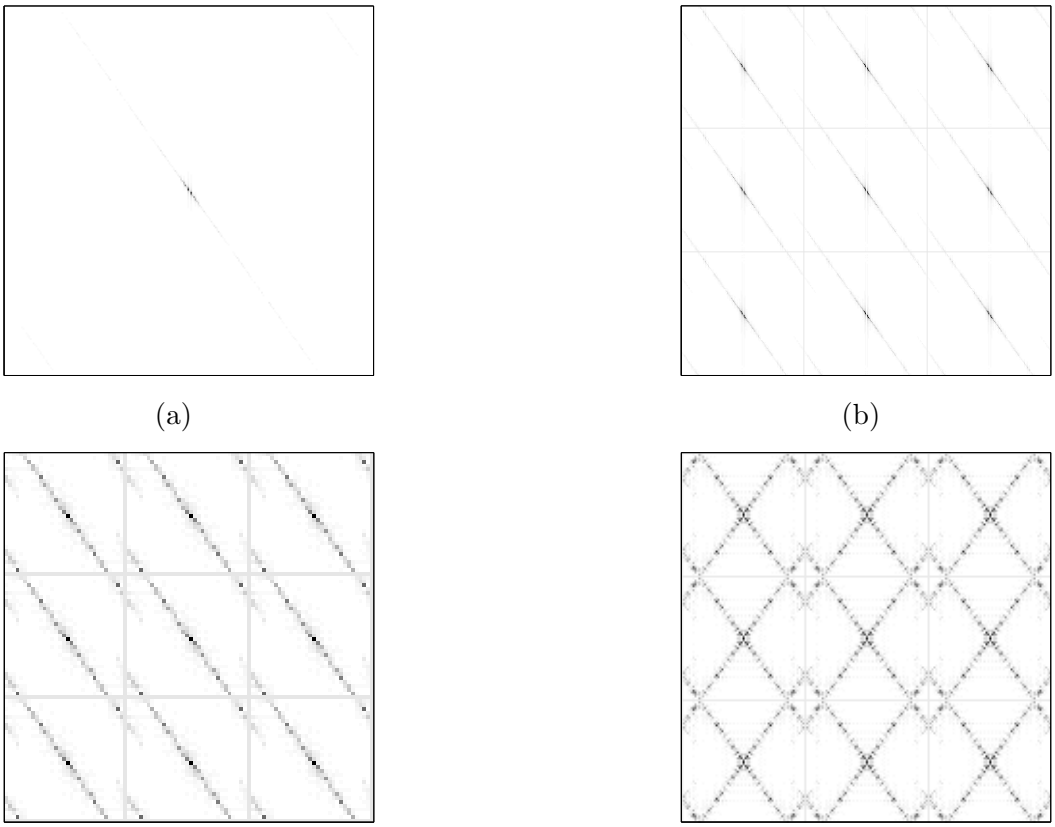


Figure 3·1: Illustration of the spectrum of a translating profile for 2D case (1D signal under translation): (a) FT; (b) DSFT; (c) DFT; and (d) DFT of a symmetrically-extended 2D signal.

in Fig. 3·1.c the DFT was periodically repeated in order to parallel the periodic nature of the DSFT; the DFT is really one period of the periodic structure shown in this figure. In both cases (DSFT and DFT) the spectrum extending beyond a unit cell of the reciprocal lattice (the square around each spectrum) finds way into neighboring unit cells; in any unit cell more than one spectral line can be present. Although this is not aliasing (the spectra do not overlap), correct reconstruction of the continuous signal is possible only with special narrowband diagonal filters; any separable 2D filter will introduce reconstruction errors. Finally, Fig. 3·1(d) shows the DFT of a symmetrically-extended 2D signal (horizontal and vertical symmetric extension), again periodically repeated to show similarity to the DSFT.

The fact that the spectrum stencil is thinner than in Fig. 3·1(c) is due to the fact that because of the symmetric extension the transformed signal has twice as many samples in each direction, and so has its DFT. It is well-known that the DCT is closely related to the DFT of a symmetrically-extended signal. Thus, the characteristic "folding" of the spectrum at unit cell boundaries in Fig. 3·1(d) can be also expected in the DCT. This effect is clearly due to the symmetric extension. We will explain all this in detail in the next section.

3.3 Interpretation of uniform image translation in the DCT domain

The discrete cosine transform introduced in Section 2.3.1 has been a driving force behind all image and video coding standards in the last 20 years, with the exception of wavelet-based JPEG2000. In addition to the excellent energy compaction, another reason for the ubiquitousness of $\mathcal{DCT}_{2e}$ is its relationship to the DFT; its coefficients can be computed by the application of DFT to a symmetrically extended signal using "half-point" symmetric extension[3]. If we define $u^s[m,n]$ to be a "half-point" symmetric extension of the original 2D signal $u[m,n]$:

$$u^s[m,n] = \begin{cases} u[m,n] & 0 \le m \le M-1, 0 \le n \le N-1 \\ u[2M_1-m-1,n] & M \le m \le 2M-1, 0 \le n \le N-1 \\ u[m,N-n-1] & 0 \le m \le M-1, N \le n \le 2N-1 \\ u[2M_1-m-1,2N-n-1] & M \le m \le 2M-1, N \le n \le 2N-1, \end{cases} \tag{3.6}$$

it is straightforward to show (Pitas, 2000) that the coefficients $\widetilde{U}$ of the M-dimensional $\mathcal{DCT}_{2e}$ of a signal are related to DFT coefficients U^s of the M-dimensional "half-point"

[3] In "full-point" symmetric extension of a 1D signal, the symmetry axis coincides with the last (or first) sample of the signal, whereas in "half-point" extension it is shifted by 1/2 of the sampling period beyond the signal. In practice, this means that while in the former case the last sample of the signal is not repeated in its symmetric extension, in the latter case – it is repeated. An MD symmetric extension can be computed by independent application of a 1D symmetric extension in each direction.

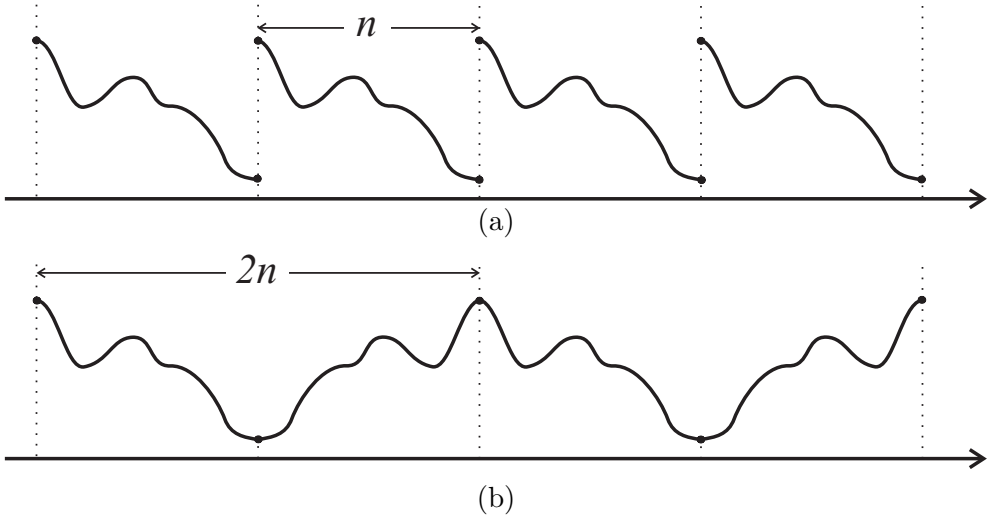


Figure 3·2: The periodicity implicit in one-dimensional (a) DFT, (b) DCT.

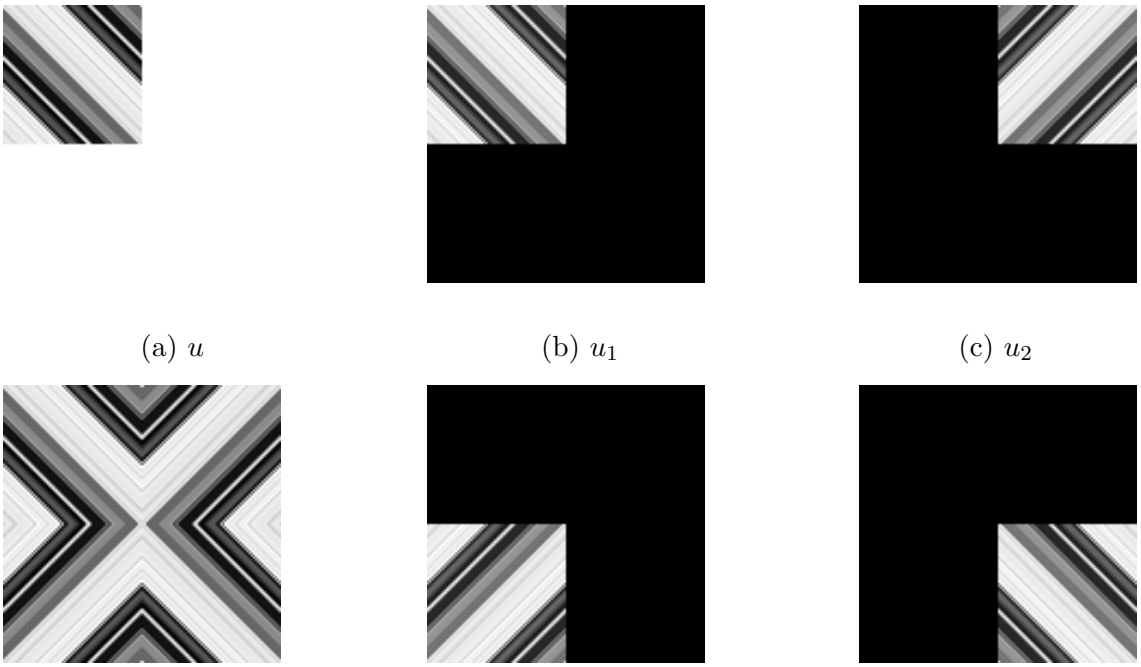
symmetric extension of this signal through the following relationship:

$$\widetilde{U}[k_1, k_2, ..., k_M] = \left(\prod_{i=1}^{M} \frac{1}{2} \sqrt{\frac{2 - \delta[k_i]}{N_i}} e^{-j\frac{\pi k_i}{2N_i}} \right) U^s[k_1, k_2, ..., k_M]. \tag{3.7}$$

In this equation, $\widetilde{U}[k_1, k_2, ..., k_M] = \mathcal{DCT}_{2e}\{u[n_1, n_2, ..., n_M]\}$ and $U^s[k_1, k_2, ..., k_M] = \mathcal{F}_{DFT}\{u^s[n_1, n_2, ..., n_M]\}$.

The above relationship leads to an efficient implementation of the DCT using FFT algorithms. The excellent energy compaction properties of the DCT can be largely explained using this relationship; a symmetrically and periodically extended signal has less high-frequency content than a periodically extended signal, due to the absence of discontinuities at each period's ends (caused in the latter case, e.g., by different signal values of the first and last signal samples). This is illustrated in Fig. 3·2.

Our goal now is to derive an analytical expression for the DCT of an image undergoing uniform translational motion. Since we already know the relationship between the DCT of a signal and the DFT of its symmetric extension, we need to find the relationship between DFT of the symmetric extension and DFT of the underlying moving pattern u_0 (3.5). Below, we give an outline of the derivation for the 2D case; details for both 2D and 3D



(a) u (b) u_1 (c) u_2

(d) u^s (e) u_3 (f) u_4

Figure 3·3: Original $N \times N$ signal u and its $2N \times 2N$ symmetric extension u^s ((a) and (d), respectively) that can be decomposed into the sum of four signals: (b) u_1; (c) u_2; (e) u_3; and (f) u_4.

case can be found in the Appendix at the end of this chapter.

Let $u[n_1, n_3]$ $(0 \leq n_1, n_3 \leq N-1)$ be a 2D signal, shown in Fig. 3·3(a), constructed by translation of 1D intensity profile $u_0[n_1]$. Since the DCT can be computed as the DFT of a symmetrically extended signal using (3.7), we first construct a "half-point" 2D symmetric extension $u^s[n_1, n_3]$ $(0 \leq n_1, n_3 \leq 2N-1)$ of the signal $u[n_1, n_2]$ (Fig. 3·3(d)). Note that the signal $u^s[n_1, n_3]$ can be decomposed into a sum of four signals: $u_i[n_1, n_3]$, $i = 1, ..., 4$, where each signal u_i is obtained through a symmetric half-point mirror of the original signal $u[n_1, n_3]$ and/or suitable zero padding. Since the four signals u_i are related to u by symmetry and/or zero padding, we can express the sum of their DFTs in terms of the DFT of u. We note here that zero padding results in increased spatial resolution of the DFT. Finally, using the formula for DFT of a translating image (3.5) and the DCT/DFT relationship (3.7), we can write the expression for DCT that depends only on the DFT of

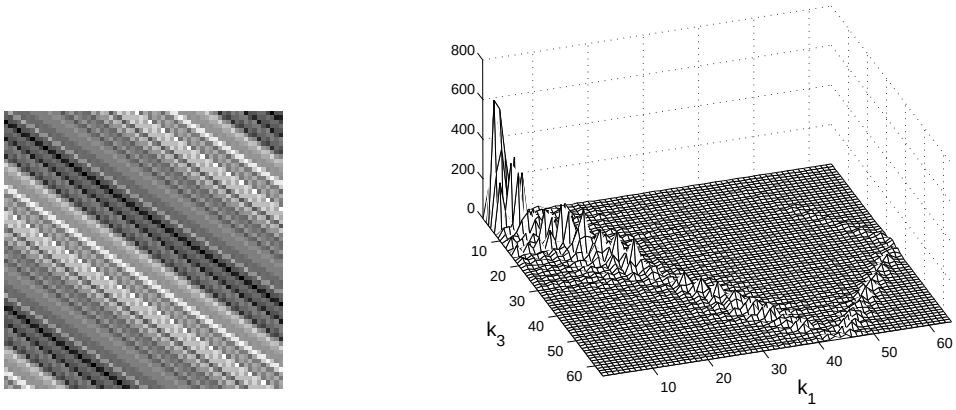
the 1D intensity profile and displacement d (for details please see the Appendix):

$$\begin{aligned}\widetilde{U}[k_1,k_3] &= \frac{\sqrt{(2-\delta[k_1])(2-\delta[k_3])}}{2N}|U_0[\frac{k_1}{2}]| \\ &\Big(\delta[(k_1d_1+k_3)/2]\cos(\Phi_0[k_1/2]-\pi\frac{k_1+k_3}{2N})+ \\ &\delta[(k_1d_1-k_3)/2]\cos(\Phi_0[k_1/2]-\pi\frac{k_1-k_3}{2N})\Big) \quad 0\le k_1,k_3\le N-1,\end{aligned} \tag{3.8}$$

where $\widetilde{U}[k_1,k_3]=\mathcal{DCT}_{2e}\{u[n_1,n_3]\}$ while $|U_0[k_1]|$ and $\Phi_0[k_1]$ are the magnitude and phase of the DFT of the 1D intensity profile $u_0[n_1]$, respectively, i.e., $\mathcal{F}_{DFT}\{u_0[n_1]\}=U_0[k_1]=|U_0[k_1]|e^{j\Phi_0[k_1]}$. The corresponding expression for the 3D case can be found in the Appendix.

It is clear from the above equation that the DCT spectrum of signal $u[n_1,n_3]$ is a sum of ridges $\delta[(k_1d_1+k_3)/2]$ and $\delta[(k_1d_1-k_3)/2]$, each modulated by a cosine function dependent on the phase of u_0. Since $\delta[(k_1d_1-k_3)/2]$ is a horizontal (or vertical) mirror of $\delta[(k_1d_1+k_3)/2]$, the sum of them causes the apparent spectral line folding within a unit cell (Fig. 3·1.d). This apparent folding is, in fact, a repeated spectrum extending from a neighboring unit cell. Although at the points of intersection of the two spectra aliasing occurs, since we are interested in the identification of spectral occupancy this should not be a significant issue. Similarly, in the 3D case (see the Appendix) there are four Kronecker deltas: $\delta[k_1d_1+k_2d_2+k_3]$, $\delta[-k_1d_1+k_2d_2+k_3]$, $\delta[k_1d_1-k_2d_2+k_3]$, $\delta[k_1d_1+k_2d_2-k_3]$. Since the last three are horizontal, vertical and temporal mirrors of the first plane, respectively, an apparent spectral folding occurs in 3D as well.

We verified this observation experimentally on a natural 1D intensity profile (2D case derived above) and a 2D image (3D case) each undergoing synthetic displacements. Fig. 3·4 shows the result for the 2D case; 1D intensity profile $u_0[n_1]$ is shifted by d_1=1.5 pixels between each two consecutive positions of n_3 (considered to be the time index), thus creating an intensity image with diagonal bands as seen in Fig. 3·4(a) (linear interpolation was used in order to generate subsequent rows from the 1D intensity profile in the case of fractional-pixel displacement). This can be thought of as an image sequence of a 1D image (single line). After applying 2D separable DCT, this image results in the spectrum



(a) (b)

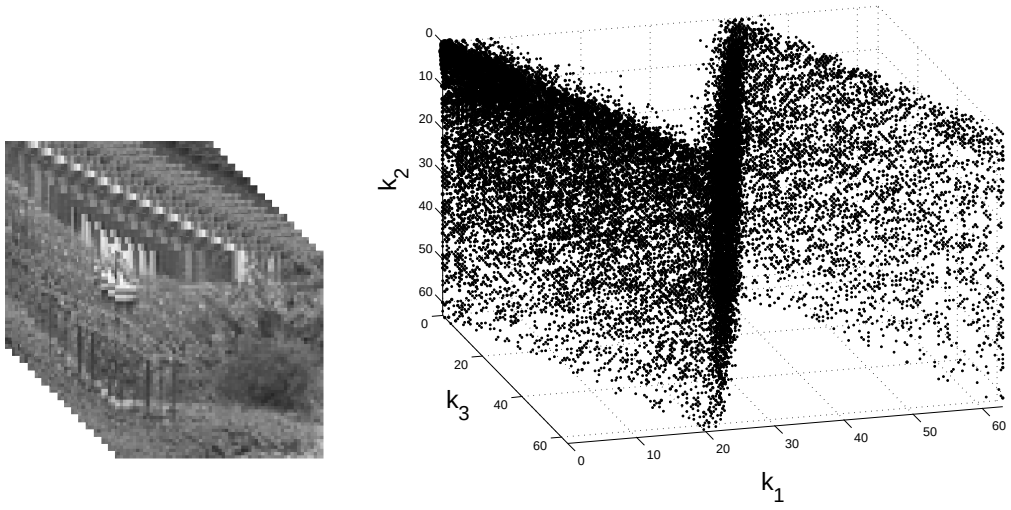
Figure 3·4: (a) Intensity image $u[n_1, n_3]$ of a 1D intensity profile $u_0[n_1]$ ((64 pixels) uniformly translating by 1.5 pixels between each consecutive two rows; and (b) its 2D DCT transform.

depicted in Fig. 3·4(b). Note the clearly visible spectral folding. Based on our previous analysis, it is clear that the "folding tail" of a 2D DCT "occupancy line" is caused by an aliasing from neighboring Voronoi cells in the DCT space.

We have performed a similar experiment for the 3D case: $u_0[n_1, n_2]$ is a still image (Fig. 3·5.a) that is shifted by $[d_1, d_2]$=[3, 0] between each two consecutive positions n_3. This creates an image sequence to which we apply 3D DCT. In Fig. 3·5(b), coefficients of the 3D DCT of this sequence are shown; the smallest coefficients have been removed by thresholding to improve visualization. Again, note the clearly visible folding of the plane along the k_1 axis.

3.4 Motion estimation in DCT domain

The characteristic energy footprint in the DCT domain for a translating image can be used to compute the direction and amplitude of motion. Suppose that a spatial block of pixels over a number of frames is considered. By applying the 3D DCT to this sequence of blocks, an analysis of the DCT coefficient energy can be performed to find a constant-velocity translational motion. One possibility is to search for a plane that passes through the highest concentration of large-energy 3D DCT coefficients; a logical choice is to seek



(a) (b)

Figure 3·5: (a) First 16 frames of sequence $u[n_1, n_2, n_3]$ $(64 \times 64 \times 64)$ which is formed by uniform [3,0] translation of a still frame $u_0[n_1, n_2]$; and (b) thresholded coefficients of its 3D DCT transform.

orientations that maximize the cumulative energy of coefficients in close proximity of the plane. Alternatively, one can minimize a complementary criterion, such as the cumulative energy of coefficients farthest away from the plane, for example:

$$\min_{\vec{\phi}} \sum_{\vec{k}} (\widetilde{U}[\vec{k}])^2 d(\vec{\phi}, \vec{k}), \tag{3.9}$$

where $d(\vec{\phi}, \vec{k})$ is the distance between a DCT coefficient at $\vec{k} = (k_1, k_2, k_3)^T$ and the dominant-energy plane whose orientation is described by a 2D vector of parameters $\vec{\phi}$. Note, that since this plane passes through the origin of the coordinate system, two parameters are enough to describe its orientation. In order to achieve minimum, $\vec{\phi}$ must be such that all coefficients distant from the plane described by $\vec{\phi}$, i.e., with large $d(\vec{\phi}, \vec{k})$, have small energy (small $(\widetilde{U}[\vec{k}])^2$). The remaining coefficients, likely to have large energy, will then be located close to this plane. Note that the above minimization allows us to estimate motion up to the sign only. After finding the motion direction and magnitude, we verify which motion vector (out of the four possibilities: $[\pm d_1, \pm d_2]$) is indeed optimal (i.e., gives

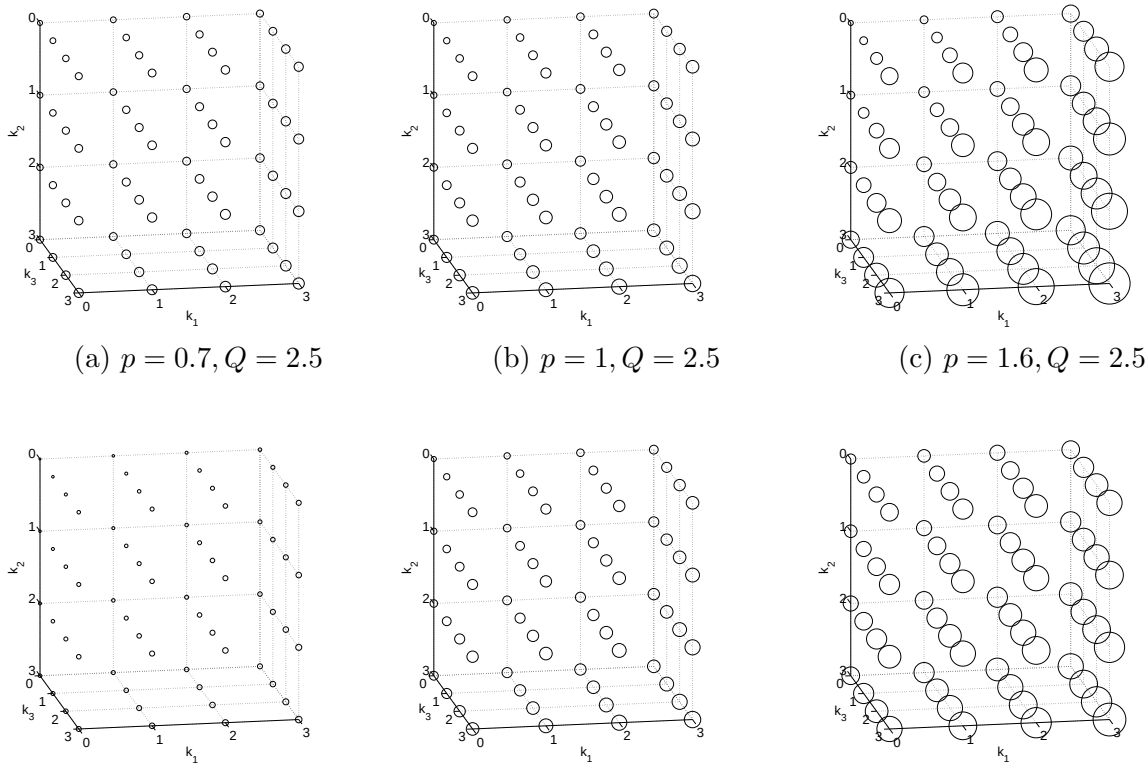
the best prediction).

Motion estimation based on minimization (3.9) is an alternative to block matching across multiple frames; either method can be used in the 3D DCT coder that we describe in the next section. However, although the plane-orientation and motion-direction representations are equivalent, this does not mean that an estimation method in one domain will give the same result as estimation in the other domain. This is due to the fact that different cost criteria and different optimization strategies may be used in each case. One remark is in order here. While the proposed motion model is not any more restrictive spatially than block motion models, temporally it is more restrictive since it a assumes constant velocity over the temporal support of the DCT.

In the next section, we discuss video coding in 3D DCT domain that revolves around the idea of coding only significant 3D DCT coefficients close to the dominant-energy plane. The orientation of this plane can be represented explicitly through parameters $\vec{\phi}$ or implicitly through a vector in the space-time domain, and can be efficiently communicated from the encoder to the decoder.

3.5 3D DCT video coding

We have shown thus far that 3D DCT spectrum of a uniformly translating image is limited to a folding plane. Although this observation is true for the idealized case of a global, constant-velocity translation only, an approximate relationship is expected to hold for sufficiently small $x - y - t$ volumes of a video sequence. Therefore, this model of the 3D DCT spectrum can be used for motion estimation in the transform domain. In this section, we propose to exploit the discovered planar concentration of 3D DCT coefficients for the purpose of compression by means of suitable DCT coefficient quantization, scanning, volume adaptation, and entropy coding.



(a) $p = 0.7, Q = 2.5$ (b) $p = 1, Q = 2.5$ (c) $p = 1.6, Q = 2.5$

(d) $p = 1, Q = 1$ (e) $p = 1, Q = 2.5$ (f) $p = 1, Q = 5$

Figure 3·6: Examples of several $4 \times 4 \times 4$ quantization volumes (quantization step is proportional to the marker size). Top row: the effect of varying p for a constant $Q = 2.5$; Bottom row: the effect of varying Q for a constant $p = 1$.

3.5.1 3D DCT coefficient quantization

Quantization is the fundamental step in achieving lossy compression. As standard 2D JPEG-like quantization tables are insufficient for compression using 3D DCT (Lee et al., 1997b) (i.e., a single table cannot account for spectral variations along the temporal frequency axis), a logical solution would be to use multiple JPEG-like tables. Instead, we extend the idea of JPEG/MPEG (2D) quantization table to *quantization volume* (3D table). Similarly to a quantization table, quantization volume must account for properties of the human visual system (HVS). In the 2D case, reduced spatial contrast sensitivity of HVS at high horizontal and vertical frequencies (Pearson, 1975) is the primary factor in selecting larger quantization step for higher k_1 and k_2. In the 3D case, the temporal

contrast sensitivity also needs to be considered. Since the spatio-temporal contrast sensitivity is considered not to be a separable function (Pearson, 1975), we need to consider a non-separable quantization model. This model should increase quantization step when contrast sensitivity is reduced. In particular, the step should increase with an increase of k_1, k_2 or k_3, increase even more for joint frequencies $k_1 - k_2$, $k_1 - k_3$, $k_2 - k_3$, and finally be the largest for $k_1 - k_2 - k_3$. Note, that in saying this we are ignoring the secondary effect of reduced spatial and temporal contrast sensitivity for very low frequencies (spatially – below 3 cycles/degree, and temporally – below 8Hz (Pearson, 1975)). Such an effect could be included in the quantization step model at the expense of added model complexity.

Based on the above considerations we propose the following quantization volume:

$$q[\vec{k}] = \lfloor Q(1 + k_1^p + k_2^p + k_3^p) \rfloor, \tag{3.10}$$

where $q(\vec{k})$ is the quantization step for coefficient at position $\vec{k}$, Q is a parameter similar to the Q-factor in JPEG quantization, and $\lfloor x \rfloor$ denotes the nearest integer less than or equal to x. By changing the parameter p, we are able to control the rate of decay within the quantization volume. This, in turn, controls the quality of the resulting image sequence (after quantization and inverse transform) and its bit-rate; higher values of p result in lower image quality but also lower bit-rate. Thus, in addition to Q, the parameter p is another element of rate control; the rate control here is necessarily more complex than in JPEG/MPEG since it involves two parameters. Typical values of p range from 0.7 to 1.6. Illustrations of several $4 \times 4 \times 4$ quantization volumes are presented in Fig 3·6, for various Q's and p's. In all examples, quantization step at $[k_1, k_2, k_3]$ is directly proportional to the area of the circle symbol at that location.

3.5.2 3D DCT coefficient scanning order

The purpose of scanning in any compression scheme is to order the quantized coefficients into a vector suitable for subsequent entropy coding and transmission. The order of coefficient scan has a significant impact on the overall compression performance. From the point

of view of an entropy coder, optimal scan would ideally order all quantized coefficients in a decaying amplitude order, which would also result in the longest zero-run at the end of the scan. Obviously, this scan is not at all practical, as its geometry is highly complex; in order to be used in the decoder, it would require transmission of a large number of coefficient coordinates. Therefore, we set three requirements for a practical scanning method. A good scanning order should:

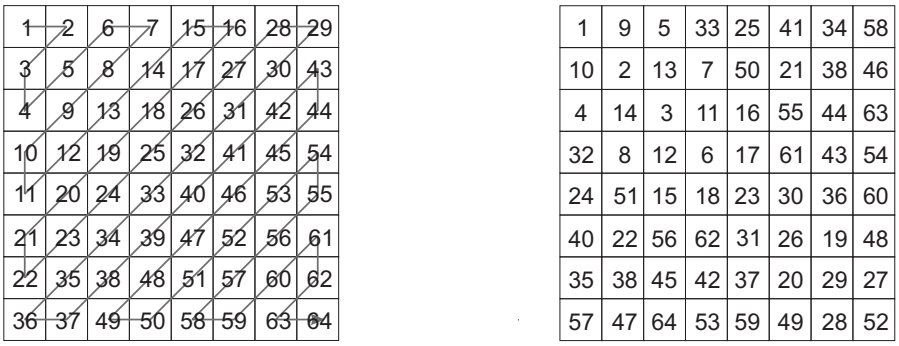
1. be predefined or dependent on a small number of parameters so that either no or only small rate overhead is incurred,

2. group zero-valued coefficients into clusters so that a run-length coder can be applied as efficiently as possible,

3. provide a very long zero-run of coefficients at the end so that a terminating code can save many bits.

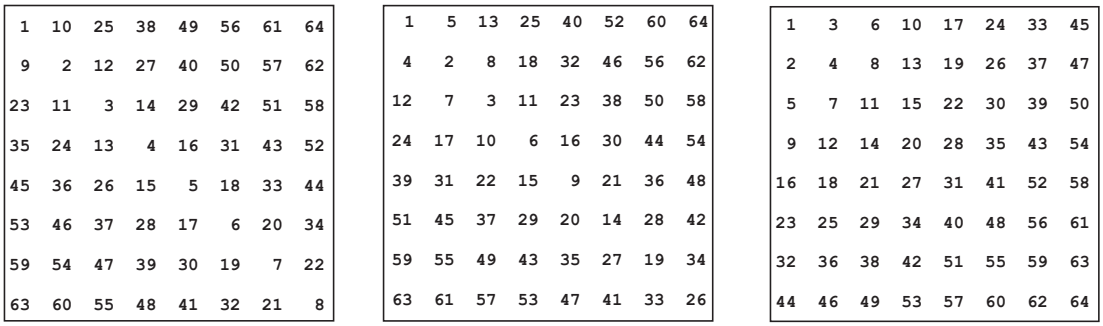
The most popular scanning method extensively used in both image and video DCT coding is the zig-zag scan, illustrated in Fig. 3·7(a). For the coding of still images or motion-compensated prediction error, zig-zag scan fits all the requirements listed above, and is adopted for use in both JPEG and MPEG coding standards. However, in the context of 3D DCT coding, this scan fails to properly capture motion-adaptive DCT spectrum. To illustrate this, in Fig. 3·7(b), we show the "reference" two-dimensional scan for the 8×8 block created by shifting the intensity profile $u_0[n_1]$ from the Figure 3·4 by 1 pixel/line. The numbers in the matrix denote the order in which coefficient at that position would be scanned based purely on its amplitude.

In prior work on 3D DCT coding, various scanning orders of DCT coefficients have been proposed, such as plane-by-plane scanning, 3D extension of the zig-zag scan (Yeo and Liu, 1995), and a more complex, parabolic scan (Lee et al., 1997b). However, none of these scan patterns adapts to motion, i.e., they all neglect the fact that significant coefficients tend to group along a plane defined by the dominant motion in the video sequence. This leads to sub-optimal coefficient ordering and shorter zero-runs, which, in turn, reduces efficiency



(a) (b)

Figure 3·7: Coefficient scanning for 8×8 block; a) Zig-zag scan, b) Optimal(reference) scan for a sample profile $u_0[n_1]$ (Fig. 3·4), undergoing translational shift with $d_1 = 1$.



(a) (b) (c)

Figure 3·8: Two-dimensional scans calculated from (3.11) for 8×8 block and $\vec{\phi} = 1$; a) $\lambda = 0$, b) $\lambda = 0.3$, c) $\lambda = 100$.

of the zero-run-length coding and of the overall compression. Therefore, we propose an adaptive scanning pattern that takes into account properties of the 3D DCT spectrum of a video sequence. As we have shown, this spectrum is related to motion in the sequence, and thus the proposed scan pattern is motion-adaptive.

From the analysis of 3D DCT spectrum in Section 3.3 we can conclude that, for a sufficiently small 3D volume of pixels in an image sequence, the further away a 3D DCT coefficient is from the folding plane (Fig. 3·5) the smaller should be its magnitude. At the same time, typical spectrum of a camera-acquired (i.e., natural rather than synthetic) image sequence has the amplitude rolling off at higher spatio-temporal frequencies. Based

on these two arguments, we propose to order coefficients based on the following metric:

$$C(\vec{k}) = d(\vec{\phi}, \vec{k}) + \lambda \|\vec{k}\|, \tag{3.11}$$

where $d(\vec{\phi}, \vec{k})$ is the earlier-defined distance function accounting for the spectral properties in the case of pure translation, $\|\vec{k}\|$ is the distance from the origin and λ is a weighting parameter used to adjust the balance between these two distance terms. After plane orientation $\vec{\phi}$ (motion direction) has been estimated, the metric $C(\vec{k})$ is calculated for all discrete 3D frequencies $\vec{k}$, and coefficients are ordered based on the value of $C(\vec{k})$: coefficients with smaller values are scanned earlier, while those with larger values are pushed towards the end.

The value of λ is set based on how accurately the motion in the image sequence is modeled by linear translation. For example, if motion in the currently encoded volume is very close to the assumed model (e.g., in the case of still background or camera pan), the coefficients will be more compactly grouped around the dominant plane. In this case, there is more confidence in the first term of $C(\vec{k})$ and λ should be smaller. If motion in the volume is very different from the assumed uniform translational motion, large coefficients will be more dispersed around the plane of dominant motion. In that case, using a larger value of λ is more appropriate. In the limit, as $\lambda \to \infty$, this model becomes motion-independent sphere-by-sphere scan.

A two-dimensional example of such coefficient ordering is given in Fig. 3·8. For the uniform translation of $d_1 = 1$ pixel/line, quantized coefficients are scanned in the order defined by a 2D version of the cost function (3.11). For $\lambda = 0$ (Fig. 3·8(a)), the scanning method relies exclusively on motion, while large λ value leads to a motion-ignorant scan similar to a zig-zag scan (Fig. 3·8(c)). A correct selection of parameter λ (more on this in the Section 3.6.2) leads to the best result - the scan shown in Fig. 3·8(b) is obtained for $\lambda = 0.3$ and closely matches the reference scan from Fig 3·7(b). Note that, although complicated, the scan from Fig. 3·8(b) is still defined by only two parameters - motion parameter $\vec{\phi}$ and control parameter λ. If these two are known at the decoder, the scan

order can be easily reproduced and used for block reconstruction.

Our proposal for content-adaptive scan raises another question - how significant is the scan-related bit rate overhead? Recall that pre-defined scans, such as the zig-zag scan, incur no rate overhead. In our approach to video coding, although the trajectory of the scan may be very complex, it is still uniquely defined by a very small set of parameters (2D vector $\vec{\phi}$) from which the scan can be exactly reconstructed at the decoder. Even if we consider no quantization and entropy coding of $\vec{\phi}$, and we use for each component of this 2D vector a wasteful 32-bit floating-point representation, for a 16×16×8 coding volume we will face less than 0.01 bits per pixel of rate overhead. Once quantization of $\vec{\phi}$ and entropy coding are considered, this becomes a negligible component of the overall rate.

3.5.3 3D DCT support adaptation

A significant decision that affects both performance and computational complexity of any transform coder is related to subimage (block) size. In most video applications, images are divided into blocks so that the correlation between adjacent blocks is significantly reduced; typically, blocks' sizes are selected to be power-of-two, as this simplifies the computation of block transforms. It has been shown that the DCT transform performs best for block sizes up to 16×16; larger block sizes not only increase computational complexity but also fail to produce significant gains. In Fig. 3·9, the impact of block size on two-dimensional transform reconstruction error is illustrated - it is clear that the DCT curve flattens as the block size increases beyond 16×16. The most popular block sizes used in still image coding are 8×8 and 16×16.

The degree with which a simple translational motion model over multiple frames can accurately portray the true underlying motion depends on the $x - y - t$ volume of data to which 3D DCT is applied, i.e., 3D DCT support volume. For small volumes, the model is more accurate, but the associated coding overhead increases. To the contrary, for large volumes the coding overhead is reduced but the motion model may not be accurate enough. To address this issue, we propose to adapt the size of 3D DCT to its motion content as

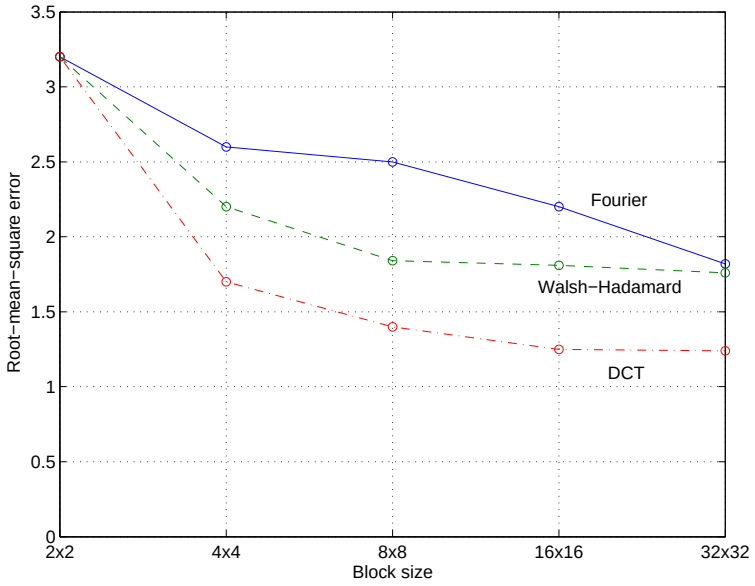


Figure 3·9: Reconstruction error versus block size. After the block-transform, one quarter of coefficients is retained and used for reconstruction.

follows. If a certain percentage of coefficients (in our experiments we used a fixed threshold of 2%) closest to the dominant plane contains more than t% of the total volume energy, we consider the motion in this volume to be very accurate and apply 3D DCT to this volume. If energy of these coefficients is smaller than t%, we split the current volume spatially into four sub-volumes (spatial quad-tree) and repeat the plane fitting step in each sub-volume. We repeat the process until either the energy of coefficients nearest to the plane is larger than the set threshold, or the volume size reaches some predefined minimum. The threshold t can be effectively used, along with quantization parameters Q and p and scanning parameter λ, for rate control of the proposed coder.

Regarding adaptive temporal support for the DCT, we observed relatively small gains when using it in our experiments, at a significantly higher computational cost (primarily related to the detection of optimal temporal volume boundaries). Therefore, we limit the 3D DCT support adaptation to the spatial dimensions. Even without a full temporal support adaptation, the detection of global temporal boundaries (scene cuts) and corresponding alignment of pixel volumes is still necessary for efficient coding of multi-scene sequences.

3.5.4 Alternatives to DCT-based motion estimation

In addition to DCT-domain motion estimation, the motion parameters $\vec{\phi}$ can be alternatively estimated as a standard motion vector in the space-time domain. The most popular way to achieve this has been block matching, the standard approach to motion estimation in video coding standards. In order to compare it with our proposed DCT domain motion estimation, we implemented exhaustive-search block matching with 1/4-pixel precision for each two consecutive frames of the pixel volume considered. The final dominant motion was calculated as a temporal average of all the estimated frame-to-frame motion vectors.

Block matching is not the only alternative to plane fitting - phase correlation is a fast and simple alternative for estimating inter-image displacements (Thomas, 1987; Wang et al., 2002). In the case of pure translation, the method produces a single peak on the phase correlation surface. If the motion departs from a uniform translation (e.g., affine motion) or in the case of multiple moving objects, an ill-defined peak (single peak with "flat" slopes) or multiple peaks will result.

The main advantage of phase correlation is that it naturally provides a quantitative measure of how well the estimated motion matches the assumed translational model - the existence of a single dominant peak can be used as indicator of an accurate estimate of translational motion. More precisely, we compare the fraction of total phase-correlation surface energy contained in the main peak with a pre-selected threshold and use it to drive a volume splitting process.

Finally, it is worth noting that both plane fitting and phase correlation methods provide additional information about fidelity of computed motion estimates, which can be used to control other coding parameters such as the λ (for scanning order).

3.5.5 Entropy coding

The adaptively scanned and quantized DCT coefficients usually contain significant runs of consecutive zeros. To take advantage of such runs, we first apply run-length coding. For additional coding gain of both coefficients and motion vectors, we employ the popular

context-adaptive binary arithmetic coder - CABAC (Marpe et al., 2003). The CABAC offers a new approach to entropy coding that utilizes time-varying statistics of symbols produced by a source coder for bit rate reduction. CABAC was originally developed within MPEG for the AVC/H.264.

In CABAC, the input is first binarized (any non-binary-valued symbol, like transform coefficient or motion vector, is converted into a binary code). A proper context model is then selected, and arithmetic encoding of each bin is performed according to the selected probability. Finally, the context model is updated based on the actual coding value - the encoding of the next bin makes use of new and updated model probabilities.

3.5.6 Computational complexity

In this section, we discuss computational complexity of the proposed 3D DCT-based coder. A direct comparison with standard video coders in terms of execution time is unfortunately not possible since our 3D DCT-based video coder (presented in more detail in Section 3.6) is completely implemented in Matlab. This makes it difficult to directly compare the encoding time of our 3D DCT coder with optimized C implementations of MPEG-2 and MPEG-4. However, an informative comparison can be obtained if the total number of operations required in the two coding architectures is analyzed.

The main computational difference between our 3D DCT coder and standard hybrid coders lies in the motion estimation block. Where hybrid coders require a motion vector per block between each pair of frames, only one motion vector per group of blocks (GOB) is needed in the 3D DCT approach. Typically, the processed 3D volumes are of length 8 or 16 - this translates to approximately a ten-fold reduction in the number of required motion vectors and results in a significant speed up in motion estimation. In addition, as the dominant translation is required to detect the maximum-occupancy plane, well-known fast motion estimation methods (e.g., phase correlation (Thomas, 1987)) can be used to significantly improve the overall encoding speed.

In a typical hybrid coding system, motion estimation consumes between 50% and 90%

of the total encoding time, depending on the complexity of search algorithm used. In order to illustrate the impact of reduced motion estimation cost on the overall encoding time, we assume that 75% of encoding is spent on motion estimation. A comparable 3D DCT coder with fixed $16 \times 16 \times 8$ volumes (GOF of size 8) would introduce an eight-fold reduction in motion estimation time and reduce the total encoding time to one third (i.e., decrease of encoding time by about 67%).

While the 3D DCT approach reduces complexity of the motion estimation block, additional complexity is now involved in the entropy coding block. The most computationally involved part of our coder is motion-adaptive scanning. We more closely look at two options for its calculation: *on-line* and *off-line*.

In the on-line approach, every time the dominant motion in the current volume is estimated, the cost function (3.11) is calculated. Pixel positions ($\vec{k}$) in the volume are then ordered and scanned, based on this cost function. This obviously requires a large number of operations, although a buffer of recently-used scans can be used to reduce coding time.

An off-line calculation of scan order seems much more practical. For a predetermined range of motion vectors, all possible scanning orders are calculated off-line and made available to both the encoder and the decoder. Fast look-up tables are then used for coefficient ordering every time motion of the current volume is estimated. An identical look-up table is used in the decoder for inverse scan and volume restoration. For a search range of ± 8 pixels and 1/2 pixel motion accuracy, there is a total of 1089 scan orders. Assuming that storage of each scan requires 4kB,[4] the total memory required to store all scan orders is about 4MB. We also note that smaller motion vectors occur more frequently as the result of motion estimation than larger estimates. If memory is limited, only scans corresponding to small motion vectors can be precalculated and stored, while other scans are calculated "on-line".

In our approach, all multidimensional DCT operations are separably implemented and

[4]This assumption is based on a $16 \times 16 \times 8$ volume (total of 2048 pixels) and two bytes per pixel for storage of each scan-order number.

(a) (b)

Figure 3·10: Motion vector field for frame #5 of the MPEG-4 sequence *"Stefan"* using 1/4 pixel precision: (a) block matching (16×16), and (b) plane fitting in the DCT domain ($16 \times 16 \times 8$).

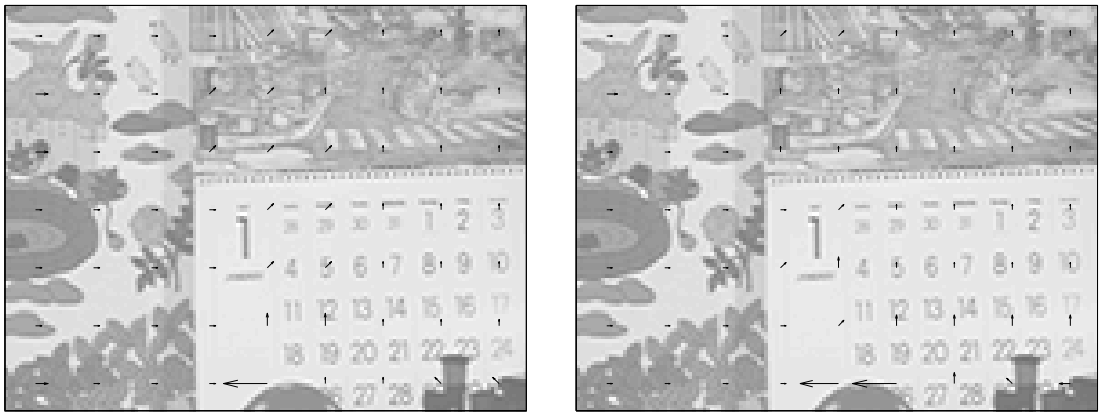
the DCT itself introduces no significant memory strain, as the largest input to a DCT block is of length 16. For the storage of the currently encoded data volume, we use at most 4kB, as the largest 3D volume processed in our coder is of size $16 \times 16 \times 16$.

In conclusion, if the memory is not severely constrained, we believe that implementing off-line scanning approach would be most beneficial computationally is highly recommended. In that case, the computational time of the entropy coding block in a 3D DCT coder approaches that of fixed scanning order of MPEG-4. Coupled with aforementioned savings in motion estimation, the estimated computational complexity of the 3D DCT-based approach stands at about 40 to 50% of complexity of MPEG-4 simple profile using the same 1/4 pixel motion accuracy. As MPEG-2 main profile is less computationally involved than MPEG-4 (among other things, it employs 1/2 motion compensation accuracy), we estimate the computational savings of 3D DCT coder at about 25% compared to MPEG-2.

3.6 Experimental results

3.6.1 Motion estimation

We have implemented the proposed motion estimation algorithm in 3D DCT domain (3.9) by means of exhaustive search in the $\vec{\phi}$'s state space implicitly defined by motion vectors



(a) (b)

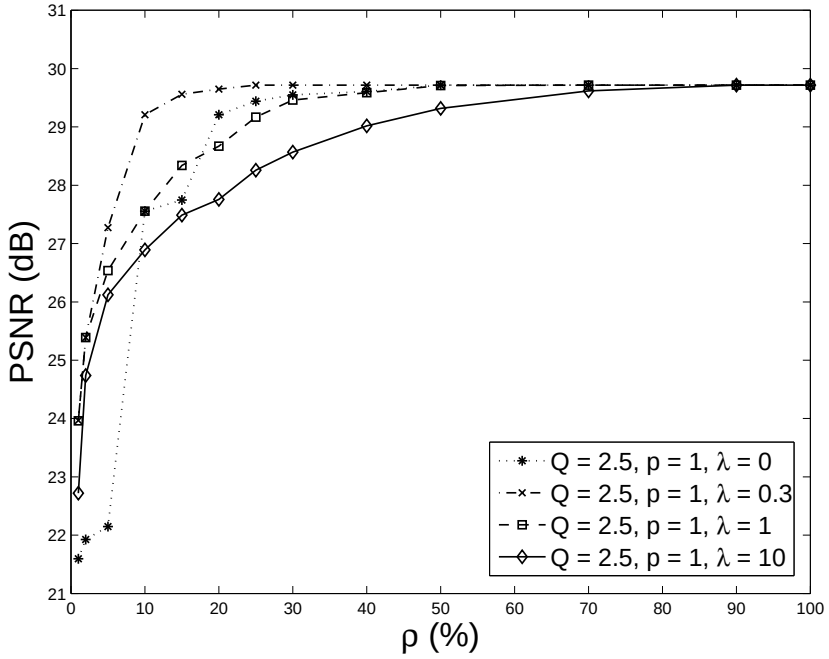
Figure 3·11: Motion vector field for frame #8 of *"Mobile and Calendar"* using 1/4 pixel precision: (a) block matching (16×16), and (b) plane fitting in the DCT domain ($16 \times 16 \times 8$).

with 1/4-pixel precision in the range of ± 8 pixels in both directions. Plane fitting in the DCT domain was performed on fixed-size coefficient volumes ($16 \times 16 \times 8$).

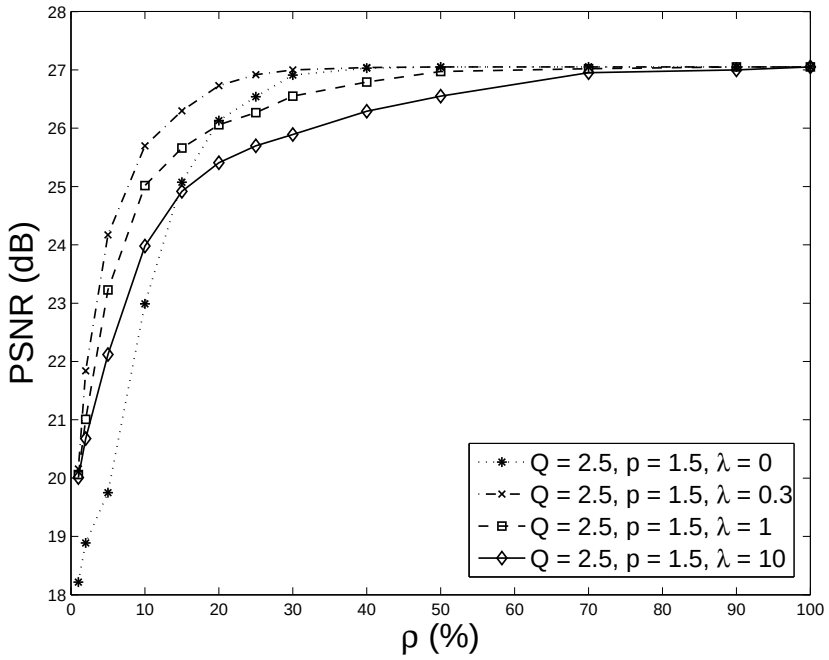
Fig. 3·10 shows displacement field estimated from the QCIF-resolution MPEG-4 test sequence *"Stefan"* using block matching applied to frame pairs (16×16 blocks), as well as our plane-fitting method. In Fig. 3·11, we show similar results for QCIF-resolution *"Mobile and Calendar"* sequence. Note that while the block matching result is obtained by measuring the displacement between two frames only, the plane fitting method finds the displacement that explains image dynamics over 8 frames. Clearly, the DCT-derived motion vectors quite accurately render motion in both sequences even with the imposed constancy constraint over the group of frames (in this case 8). The result will not be as accurate should a significant velocity change take place over time.

3.6.2 Quantization and scanning order

In order to calibrate the proposed scanning and quantization, we have first studied the impact of parameters λ, Q and p on the DCT coefficient restriction error. This error describes compaction properties of various scans, and is defined as the difference between the original image and a reconstructed image obtained by keeping only a fraction ρ of



(a)



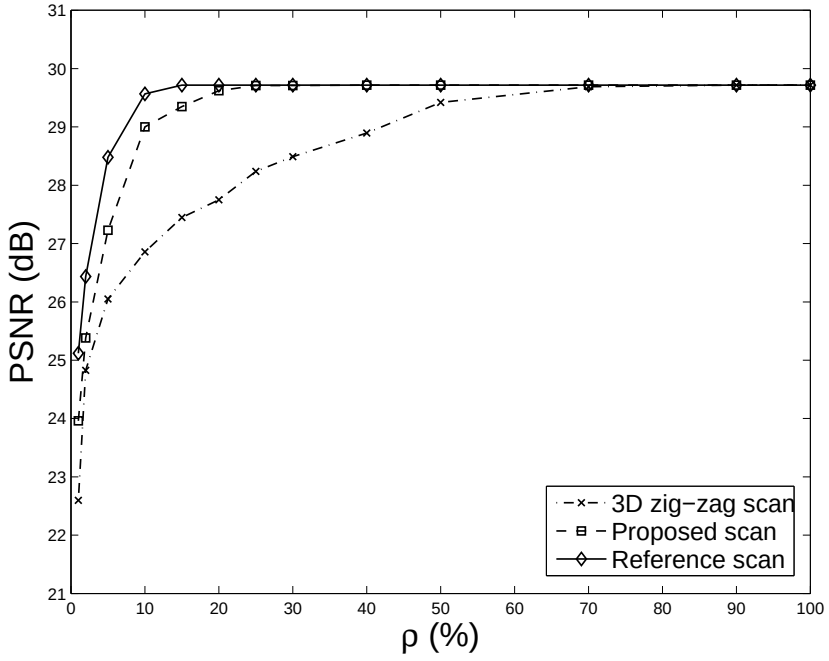
(b)

Figure 3·12: Coefficient restriction error expressed as PSNR for the proposed scan with different values of parameter λ: (a) synthetic sequence with globally translational motion $[d_1, d_2] = [0, 1]$, (b) MPEG-4 test sequence *"Stefan"*.

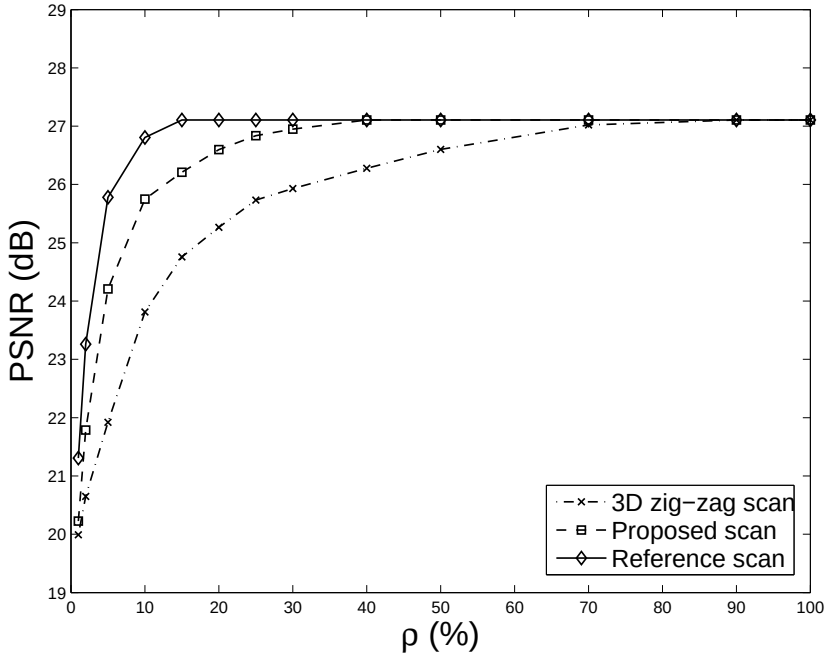
quantized and ordered DCT coefficients. Clearly, this error is strongly influenced by the ordering of coefficients; larger coefficients scanned earlier will result in faster saturation of the error. It should be noted, however, that there is no direct relationship between the coefficient restriction error and image coding error. Although in both cases coefficients are quantized, in the coefficient restriction error they are gradually set to zero starting from the end of the scan, whereas in the coding error they are entropy coded (Section 3.5.5).

In Fig. 3·12 we show the dependence of the coefficient restriction error on ρ for different values of λ for the proposed scan on a 16-frame QCIF-resolution natural-texture sequence (Fig. 3·5.a) with synthetic motion, as well as for a 32-frame QCIF-resolution natural MPEG-4 test sequence (*"Stefan"*). Plane fitting was used for motion estimation in the case of natural sequence, while in the synthetic sequence motion vector value is known. In the synthetic case, we used moderate quantization (Q=2.5, p=1) leading to a good-quality reconstruction even at 5:1 compression (ρ=20%). For the natural test sequence, we used the same Q but faster rate of decay (Q=2.5, p=1.5), in order to obtain similar histogram of the quantized coefficients to that in the synthetic case. From the results, it is clear that the best performance is attained in each case for λ=0.3; more confidence is given to the distance from the dominant-energy plane than to the distance from the origin. This was to be expected when motion is well defined. We have also obtained similar graphs for finer and coarser quantization with the difference that for coarser quantization the PSNR curves attain saturation faster and at lower PSNR value. We conclude that, although an optimal (from the point of view of coefficient restriction error) λ can always be found, the range of good performance (loss of less than 1dB from the best λ) is quite large: basically any λ between 0.2 and 0.5 works very well. Interestingly, λ=0 results in a larger error since coefficients are ordered exclusively based on their distance from the dominant plane, thus disregarding the low-pass spectrum of typical video.

Having found a good range for the parameter λ, we have tested the proposed scan order against other methods described in the literature. We compared our scan with a 3D extension of the zig-zag scan (Yeo and Liu, 1995), and with the optimal ordering, i.e.,



(a)



(b)

Figure 3·13: Coefficient restriction error expressed as PSNR for different scan orders: (a) synthetic sequence with globally translational motion $[d_1, d_2] = [0, 1]$, Q=2.5, p=1, λ=0.3, (b) MPEG-4 test sequence *"Stefan"*, Q=2.5, p=1.5, λ=0.3.

based on the magnitude of the actual 3D DCT coefficients to be encoded. The latter scan is used only for reference as it is the limit on how well we can perform the ordering of coefficients, but it is not a practical method since coordinates of all coefficients need to be transmitted thus significantly increasing the bit rate, as described in the Section 3.5.2. In Fig. 3·13(a), we show results for the synthetic sequence. The 3D DCT is performed on $16 \times 16 \times 16$ blocks, and the same moderate coefficient quantization as before (Q=2.5, p=1) is applied prior to scanning. Note that for PSNR of 28dB we need only about 10% of the DCT coefficients for the proposed scan, and about 25% of coefficients for the 3D zig-zag scan. The reference scan requires about 5% of coefficients. In Fig. 3·13(b), we show results for the test sequence *"Stefan"* (QCIF), with the 3D DCT performed again on $16 \times 16 \times 16$ blocks, and the quantization parameters Q=2.5, p=1.5. Motion estimation was performed using plane fitting. Again, a restriction to the first 13% of coefficients in the proposed motion adaptive scan results in PSNR of about 26dB, while about 35% of coefficients are needed to achieve the same PSNR with the 3D zig-zag scan. The reference scan requires only about 6% of the coefficients. Clearly, the proposed scan significantly outperforms the 3D zig-zag scan and is fairly close to the reference scan in both cases.

Note that the proposed scan also results in much longer zero runs when compared to the 3D zig-zag scan. For example, the average length of the final zero-run for the $16 \times 16 \times 16$ block of the *"Stefan"* sequence encoded with quantization parameters Q=2.5, p=1.5 was 439 with 3D zig-zag scan, 2112 with the proposed scan and 3173 with the reference scan. This suggests that significant gain should be expected from subsequent run-length coding.

3.6.3 Video compression

We now move to the analysis of compression performance of our 3D DCT coder. As pointed out in Section 3.5.3, we limit the temporal DCT support to 16 frames, while we allow spatial DCT support to adapt starting at 16×16 pixels and ending at 4×4 pixels. A temporal scene-cut detection is required for multi-scene video coding in order to prevent GOF running across non-correlated frames. Depending on the current volume size and

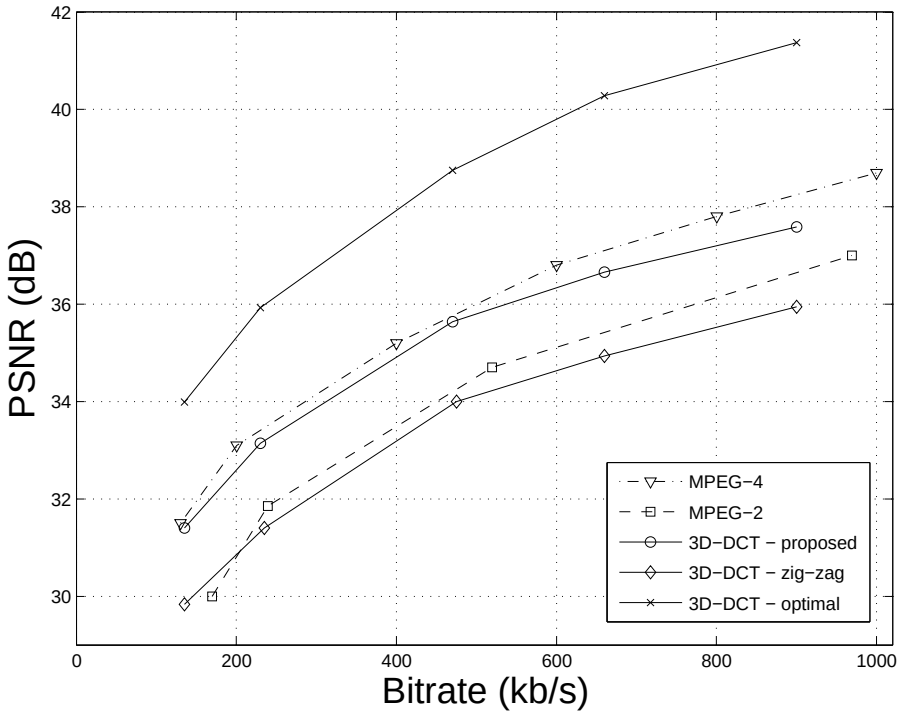


Figure 3·14: Rate-distortion performance of the 3D-DCT coder with optimal, 3D zig-zag and proposed coefficient scans in comparison with MPEG-4 and MPEG-2 coders - *Foreman* sequence

target bit-rate, typical values of the volume size adaptation threshold t are in the range of 15-40%. This threshold is kept fixed for the entire sequence based on the target bit rate, and used to adaptively select 3D DCT size as described in Section 3.5.3.

In Fig. 3·14 and 3·15, we compare the proposed coder for three scans: optimal (reference), 3D zig-zag and the proposed scan, with two standard DCT-based coders, MPEG-2 (main profile) and MPEG-4 (simple profile). For our compression tests, we use CIF resolution *"Foreman"* and *"Mobile and Calendar"* sequences at 30Hz.

We can see that, as expected, in both sequences the optimal scan outperforms all other scans. Also, our proposed scan outperforms the 3D zig-zag by at least 2 dB across all rates. For the *"Foreman"* sequence, the proposed 3D-DCT coder easily outperforms the MPEG-2 coder and is close to the MPEG-4 coder, especially at lower bitrates. For *"Mobile and Calendar"*, MPEG-4 outperforms our coder by less than 1 dB, and our coder still outperforms MPEG-2 by about 2 dB. The very good performance of our coder, especially

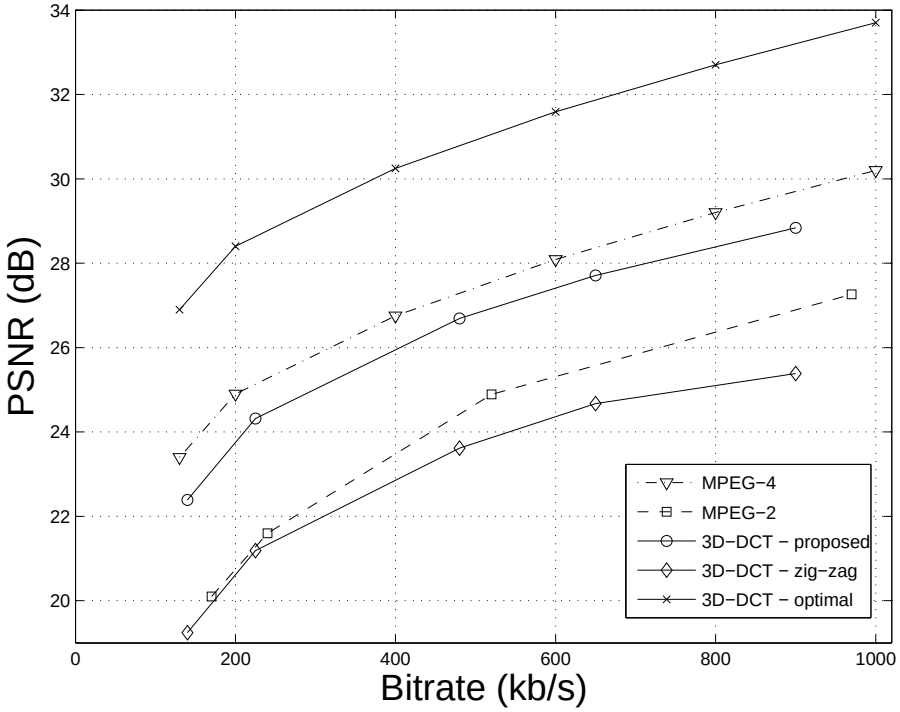


Figure 3·15: Rate-distortion performance of the 3D-DCT coder with optimal, 3D zig-zag and proposed coefficient scans in comparison with MPEG-4 and MPEG-2 coders - *Mobile and Calendar* sequence.

at lower rates, can be credited to the small motion overhead that is typically five times smaller than that of MPEG.

In addition, longer zero-runs are observed at low bitrates, as a result of effective motion-adaptive scanning that follows after coarse quantization. We finally notice that at higher bit-rates the gap between the MPEG-2 and 3D DCT coders decreases, due to a relatively poor motion rendering in the 3D DCT coder.

The average coding performance of the proposed 3D DCT coder relative to both MPEG standards has been summarized in Table 3.1 using the method proposed by Bjontegaard for measuring the average difference between two rate-distortion curves (Bjontegaard, 2001). This method first fits a third-order polynomial through points on each curve. Then, integrals of these two polynomial functions are calculated and the average PSNR gain/loss is computed as the difference between the two integrals divided by the integration interval (160-1000 kbps in our experiments). The average bit-rate and PSNR gain/loss between the

Table 3.1: Average coding gain/loss of the proposed 3D DCT coder in comparison with MPEG-2 and MPEG-4 for *"Foreman"*, *"Mobile and Calendar"*, *"Stefan"*, *"MIT"* and *"Coastguard"* computed using the method proposed by Bjontegaard (Bjontegaard, 2001).

	MPEG-2	MPEG-4
Bit rate change at fixed PSNR	-22.9%	+14.2%
PSNR change at fixed bit rate	+1.83 dB	-0.71 dB

proposed coder and both MPEG-2 and MPEG-4 have been computed using this approach for the following test sequences: *"Foreman"*, *"Mobile and Calendar"*, *"Stefan"*, *"MIT"* and *"Coastguard"*. Clearly, for a given PSNR our coder requires about 23% lower rate than MPEG-2, but 14% higher rate than MPEG-4. For a given rate, our coder outperforms MPEG-2 by 1.83 dB while performing about 0.71 dB below MPEG-4.

Finally, a visual comparison of our proposed coder with MPEG-2 and MPEG-4 is presented in Fig. 3·16. Our coder produces artefact-free image that is clearly better than MPEG-2 (notice how MPEG-2 coder produces artifacts around player's right foot (*Stefan*) and noise in the calendar *(Mobile and Calendar)*. At the same time, the performance of 3D DCT coder is visually indistinguishable from MPEG-4, despite about 0.7 dB objective loss.

3.7 Summary and conclusions

In this chapter, we have studied translational motion properties in the DCT domain. We have shown that translational, constant-velocity motion results in restricted spectral occupancy in the 3D DCT domain; the spectrum is limited to a folding plane. Based on this spectral footprint, we have proposed a motion estimation method based on plane fitting to high-energy DCT coefficients. We have shown that this method performs well and is comparable to block matching. We have also exploited the spectral footprint in video coding; we have proposed DCT coefficient quantization and motion-adaptive scanning, as

Stefan CIF 30Hz, decoded frame #60

(a) MPEG-2

(b) 3D-DCT

(c) MPEG-4

Mobile CIF 30Hz, decoded frame #241

(d) MPEG-2

(e) 3D-DCT

(f) MPEG-4

Figure 3·16: Left column: visual comparison of frame #60 from *Stefan* CIF sequence encoded at 768 kbps: (a) MPEG-2 (27.63 dB), (b) proposed 3D-DCT (28.94 dB), (c) MPEG-4 (29.71 dB). Right column: visual comparison of frame #241 from *Mobile and Calendar* CIF sequence encoded at 256 kbps: (d) MPEG-2 (21.69 dB), (e) proposed 3D-DCT (24.34 dB), (f) MPEG-4 (25.05 dB).

well as DCT support adaptation, based on this footprint. Complemented with suitable entropy coding, the proposed coder performs very well; for lower rates it clearly outperforms MPEG-2, although it is outperformed by MPEG-4. Visually our coder produces sequences very similar to MPEG-4 at lower bit rates, while outperforming MPEG-2 at higher bit rates.

In the next chapter, we investigate another video compression method based on a multiframe approach and motion-compensated temporal filtering (MCTF). In MCTF coders, the discrete wavelet transform assumes the role of the DCT as the decorrelating transform of choice.

APPENDIX

We consider the 2D case first. Let $u[n_1, n_3]$ $(0 \le n_1, n_3 \le N-1)$ be a 2D signal constructed by translation of 1D intensity profile $u_0[n_1]$ and let $U[k_1, k_3]$ $(0 \le k_1, k_3 \le N-1)$ be its discrete Fourier transform.[5] Let u_1 be defined as a zero-padded version of u, and let u_2, u_3 and u_4 be horizontal, vertical and horizontal/vertical symmetric "half-point" mirrors of u, with the remainder padded out with zeros:

$$u_1[n_1, n_3] = \begin{cases} u[n_1, n_3] & 0 \le n_1, n_3 \le N-1 \\ 0 & \text{otherwise} \end{cases}$$

$$u_2[n_1, n_3] = \begin{cases} u[2N-1-n_1, n_3] & N \le n_1 \le 2N-1, 0 \le n_3 \le N-1 \\ 0 & \text{otherwise} \end{cases}$$

$$u_3[n_1, n_3] = \begin{cases} u[n_1, 2N-1-n_3] & 0 \le n_1 \le N-1, N \le n_3 \le 2N-1 \\ 0 & \text{otherwise} \end{cases}$$

$$u_4[n_1, n_3] = \begin{cases} u[2N-1-n_1, 2N-1-n_3] & N \le n_1, n_3 \le 2N-1 \\ 0 & \text{otherwise} \end{cases}$$

With the above definitions, the symmetrically extended (type 2e) version of u, namely u^s as defined in (3.6), can be decomposed as follows:

$$u^s[n_1, n_3] = u_1[n_1, n_3] + u_2[n_1, n_3] + u_3[n_1, n_3] + u_4[n_1, n_3].$$

Note that by using the zero-padding theorem, we can express the DFT of each of those four signals as interpolated (by factor of 2) DFT of the corresponding non-padded signals, i.e., $U_1[k_1, k_3] = U[k_1/2, k_3/2]$, where $U_i[k_1, k_3] = \mathcal{F}_{DFT}\{u_i[n_1, n_3]\}$. Since, by definition,

[5] We use (n_1, n_3) indices rather than (n_1, n_2) in order to maintain consistency with the 3D notation used in the main body of this article, where n_3 denotes a temporal index.

u_1, u_2, u_3, u_4 are mirrors of one another, it is easy to show that the 2D DFT of u^s is:

$$\begin{aligned} U^s[k_1,k_3] &= U[\frac{k_1}{2},\frac{k_3}{2}] + U[-\frac{k_1}{2},\frac{k_3}{2}]e^{j\pi k_1/N} + \\ & \quad U[\frac{k_1}{2},-\frac{k_3}{2}]e^{j\pi k_3/N} + U[-\frac{k_1}{2},-\frac{k_3}{2}]e^{j\pi(k_1+k_3)/N}. \end{aligned}$$

Using now the relationship (3.5) to express $U[k_1,k_3]$ as a function of $U_0[k_1]$ (DFT of intensity profile $u_0[n_1]$): $U[k_1,k_3] = U_0[k_1]\delta[k_1d_1 + k_3]$, and also the magnitude/phase representation for U_0, i.e., $U_0[k_1] = |U_0[k_1]|e^{j\Phi_0[k_1]}$ it is easy to see that:

$$\begin{aligned} U^s[k_1,k_3] &= |U_0[\frac{k_1}{2}]|\delta[\frac{k_1}{2}d_1 + \frac{k_3}{2}]e^{j\Phi_0[\frac{k_1}{2}]} + \\ & \quad |U_0[-\frac{k_1}{2}]|\delta[-\frac{k_1}{2}d_1 + \frac{k_3}{2}]e^{j\pi k_1/N}e^{j\Phi_0[-\frac{k_1}{2}]} + \\ & \quad |U_0[\frac{k_1}{2}]|\delta[\frac{k_1}{2}d_1 - \frac{k_3}{2}]e^{j\pi k_3/N}e^{j\Phi_0[\frac{k_1}{2}]} + \\ & \quad |U_0[-\frac{k_1}{2}]|\delta[-\frac{k_1}{2}d_1 - \frac{k_3}{2}]e^{j\pi(k_1+k_3)/N}e^{j\Phi_0[-\frac{k_1}{2}]}. \end{aligned}$$

Since the input signal $u_0[k]$ is real, and thus $|U_0[k]| = |U_0[-k]|$ and $\Phi[k] = -\Phi[-k]$, we can finally write the DFT of signal's 2D symmetric extension as:

$$\begin{aligned} U^s[k_1,k_3] = 2|U_0[\frac{k_1}{2}]|\ e^{j\pi\frac{k_1+k_3}{2N}}\Big(&\delta[(k_1d_1+k_3)/2]\cos(\Phi_0[k_1/2] - \pi\frac{k_1+k_3}{2N}) + \\ &\delta[(k_1d_1-k_3)/2]\cos(\Phi_0[k_1/2] - \pi\frac{k_1-k_3}{2N})\Big), \\ & 0 \le k_1,k_3 \le 2N-1. \end{aligned}$$

The final DCT of $u[n_1,n_3]$ is obtained from the above DFT as a scaled version (3.7) of one quarter of $U^s[k_1,k_3]$ as follows:

$$\begin{aligned} \widetilde{U}[k_1,k_3] &= \frac{\sqrt{(2-\delta[k_1])(2-\delta[k_3])}}{2N}|U_0[\frac{k_1}{2}]| \\ & \quad \Big(\delta[(k_1d_1+k_3)/2]\cos(\Phi_0[k_1/2] - \pi\frac{k_1+k_3}{2N}) + \\ & \quad \delta[(k_1d_1-k_3)/2]\cos(\Phi_0[k_1/2] - \pi\frac{k_1-k_3}{2N})\Big) \quad 0 \le k_1,k_3 \le N-1. \end{aligned}$$

It is clear from the above equation, that DCT of the original signal is a sum of modulated ridges: $\delta[(k_1d_1 + k_3)/2]$ and its horizontal mirror $\delta[(k_1d_1 - k_3)/2]$. The addition of this mirror is exactly what explains the observed spectral folding.

In the 3D case, we consider $u[n_1, n_2, n_3]$ $(0 \le n_1, n_2, n_3 \le N-1)$ to be a 3D signal constructed by translation of image $u_0[n_1, n_2]$. Let $U[k_1, k_2, k_3]$ $(0 \le k_1, k_2, k_3 \le N-1)$ be its discrete Fourier transform. Similarly to the 2D case, the symmetrically extended 3D signal $u^s[n_1, n_2, n_3]$ can be written as a sum of zero-padded $u[n_1, n_2, n_3]$ and its seven "half-point" mirrors (horizontal, vertical, temporal, horizontal-vertical, horizontal-temporal, vertical-temporal and horizontal-vertical-temporal). By applying 3D DFT to $u^s[n_1, n_2, n_3]$, and using the zero-padding theorem, it easy to show that:

$$\begin{aligned} U^s[k_1,k_2,k_3] &= U[\frac{k_1}{2},\frac{k_2}{2},\frac{k_3}{2}] + U[-\frac{k_1}{2},\frac{k_2}{2},\frac{k_3}{2}]e^{j\pi k_1/N} + \\ &\quad U[\frac{k_1}{2},-\frac{k_2}{2},\frac{k_3}{2}]e^{j\pi k_2/N} + U[\frac{k_1}{2},\frac{k_2}{2},-\frac{k_3}{2}]e^{j\pi k_3/N} + \\ &\quad U[-\frac{k_1}{2},-\frac{k_2}{2},\frac{k_3}{2}]e^{j\pi (k_1+k_2)/N} + U[-\frac{k_1}{2},\frac{k_2}{2},-\frac{k_3}{2}]e^{j\pi (k_1+k_3)/N} + \\ &\quad U[\frac{k_1}{2},-\frac{k_2}{2},-\frac{k_3}{2}]e^{j\pi (k_2+k_3)/N} + U[-\frac{k_1}{2},-\frac{k_2}{2},-\frac{k_3}{2}]e^{j\pi (k_1+k_2+k_3)/N}. \end{aligned}$$

Similar to the 2D case, using the relationship (3.5), i.e., $U[k_1, k_2, k_3] = U_0[k_1, k_2]\delta[k_1d_1 + k_2d_2 + k_3]$, the magnitude/phase representation for U_0, and assuming that u_0 is real, the final DCT of $u[n_1, n_2, n_3]$ can be shown to be:

$$\begin{aligned} \widetilde{U}[k_1,k_2,k_3] &= \frac{\sqrt{(2-\delta[k_1])(2-\delta[k_2])(2-\delta[k_3])}}{4\sqrt{N^3}} \\ &\quad |U_0[\frac{k_1}{2},\frac{k_2}{2}]|\Big(\delta[(k_1d_1+k_2d_2+k_3)/2]\cos(\Phi_0[k_1/2,k_2/2] - \pi\frac{k_1+k_2+k_3}{2N}) + \\ &\qquad \delta[(k_1d_1+k_2d_2-k_3)/2]\cos(\Phi_0[k_1/2,k_2/2] - \pi\frac{k_1+k_2-k_3}{2N})\Big) + \\ &\quad |U_0[\frac{k_1}{2},-\frac{k_2}{2}]|\Big(\delta[(k_1d_1-k_2d_2-k_3)/2]\cos(\Phi_0[k_1/2,-k_2/2] - \pi\frac{k_1-k_2-k_3}{2N}) + \\ &\qquad \delta[(k_1d_1-k_2d_2+k_3)/2]\cos(\Phi_0[k_1/2,-k_2/2] - \pi\frac{k_1-k_2+k_3}{2N})\Big) \\ &\qquad\qquad 0 \le k_1,k_2,k_3 \le N-1. \end{aligned}$$

Clearly, the DCT of a translating image is a sum of four cosine-modulated planes: the

original plane $\delta[k_1 d_1 + k_2 d_2 + k_3]$ and its three mirrors. The addition of the three mirrors is exactly what explains the observed 3D spectral folding.

Chapter 4

Subband video coding: Beyond 3D DCT

4.1 Introduction

Our new 3D DCT coding design, presented in Chapter 3, results in significant coding improvement over all previously reported 3D DCT-based solutions. It also offers lower computational complexity than a comparable hybrid system. However, the compression performance of our 3D DCT coder is still about 1 dB below that of the MPEG-4, and, by extrapolation, about $2-4$ dB below the newest coding standard, AVC/H.264, mainly due to the overly-simplistic motion modeling and smaller coding gain when assumption of the uniform translation over several frames is severely violated. Despite this, the idea of simultaneous processing of group of frames (as opposed to two-frame temporal prediction) deserves further attention. In anticipation of better temporal decorrelation, we pursue advanced multi-frame processing, by means of *temporal filtering*. This concept is closely related to another discrete linear transform - the discrete wavelet transform (DWT).

Long before wavelets were proposed to handle temporal decorrelation in video coders, they had already proved to be useful for still image coding. A strong research activity in wavelet image coding during the 1990's resulted in the JPEG2000 compression standard, based entirely on the DWT. As most of the wavelet-based still image coding solutions proved to be effective in handling the spatial part of the 3D wavelet transform, our work focuses on the remaining temporal wavelet transform. When adaptively steered using estimated motion, this transform is called *motion-compensated temporal filtering* (MCTF). Section 4.2 discusses fundamentals of wavelet video coding with focus on the scalability features. The new temporal processing block and its structure are presented in detail in

Section 4.3 for multi-level motion-compensated Haar and 5/3 DWTs, as both of these kernels are used extensively for temporal decorrelation of video. MCTF is first analyzed in the original *transversal* form (Section 4.3.1); another implementation, called *lifting*, is subsequently introduced in Section 4.3.2. Properties of both MCTF implementations, including the transform invertibility, are discussed. Finally, sections 4.4 and 4.5 provide an insight into how motion and temporal wavelet transform interact. The experimental verification of the most important theoretical result concludes the chapter.

4.2 Wavelet video coder: design and properties

Since their introduction in late 80's (Karlsson and Vetterli, 1988), significant research progress has been made in the area of 3D DWT video coders. This section lays out basic concepts of motion-compensated wavelet video coding. The structure of a typical 3D DWT coder is identical to that of a more general 3D transform coder (Fig. 2·2). In contrast to hybrid systems, a 3D DWT encoder does away with the prediction loop creating the so called feedforward (as opposed to feedback) structure. In its most popular form, the 3D DWT transform is separably implemented as a one-dimensional temporal (T) wavelet transform followed by the two-dimensional spatial (S) transform (Figure 4·1). Entropy coding (E) of quantized transform coefficients completes the encoder. These steps are simply reversed in the decoder (E^{-1}, S^{-1}, and T^{-1}) to obtain the reconstructed video. As the temporal transform precedes the spatial transform in the encoder, this structure is often referred to as "t+2D." Multiple levels of both temporal and spatial transform may be used (in our example, two levels of each transform are shown). The existing solutions for spatial wavelet transform, developed for still image coding, are relatively easily adjusted to handle spatial part of a 3D transform. Most of the time, 3 to 5 stages of Daubechies 9/7 wavelet transform (Antonini et al., 1992) are used for spatial processing.

Despite the departure from the hybrid coding paradigm, modern 3D DWT coders also include many of the algorithms originally developed for hybrid coders. For example, motion estimation (e.g., hierarchical variable-size block matching - HVSBM) and entropy

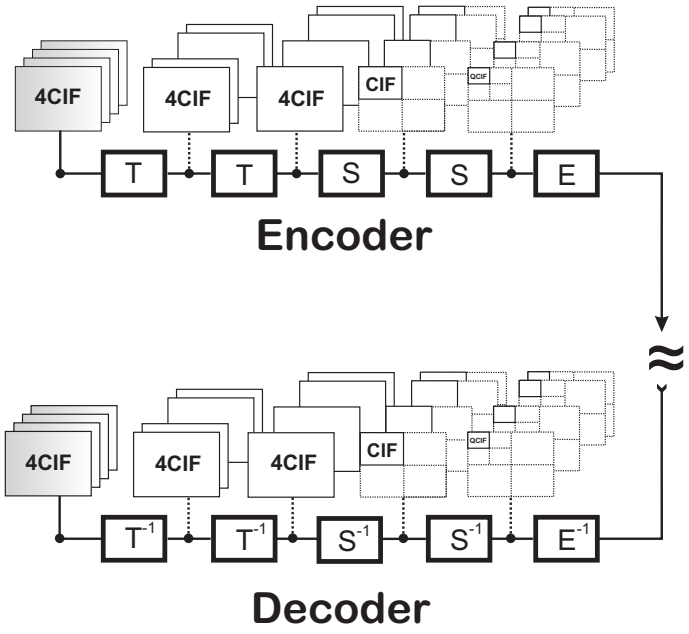


Figure 4·1: Separable implementation of the 3D DWT transform in the wavelet video encoder; "t+2D" coding structure is shown, with two levels of temporal and two levels of spatial DWT.

coding (e.g., context-adaptive binary arithmetic coding - CABAC), have been initially developed for the H.264/AVC standard and later successfully implemented in the context of 3D subband coding (Marpe et al., 2003; Golwelkar and Woods, 2003). Alongside formidable compression performance, 3D-DWT coders exhibit higher computational complexity and increased memory requirements compared to H.264/AVC. The latest wavelet efforts within MPEG have divided the focus equally between compression performance and computational complexity. As discussed earlier in Chapter 2, only a reasonably complex video coding scheme providing high compression gains will stand a chance of successfully competing with the H.264/AVC.

4.2.1 Scalability of wavelet video coders

The feedforward structure of a 3D DWT coder (Figure 4·1) naturally supports highly scalable video coding. Video scalability generally denotes a capability of coding scheme to easily extract and decode video from once-encoded bitstream at varying quality, frame-

rate, or resolution, respectively called *quality, temporal, and spatial* scalability. In addition to these three, other types of scalability exist. Few examples are: *complexity scalability* (which allows for the adaptation to decoding resources) and *object-based scalability* (where the bitstream is layered according to the importance of objects in a scene). Ideally, different types of scalability are combined, according to specific application requirements, in order to provide scalability along multiple dimensions.

Quality scalability (also referred to as PSNR or rate scalability) is natively supported and easily implemented in wavelet video coders for two main reasons: one is the open-loop non-predictive structure of the wavelet encoder, while the other is bit-plane coding of spatio-temporal subbands using embedded zero-tree coders (Shapiro, 1993; Said and Pearlman, 1996; Taubman, 2000; Hsiang and Woods, 2000). Decoding of an embedded bitstream can be stopped at any point with nearly optimal reconstruction PSNR. For a long time, PSNR scalability has been used as a synonym for scalability in general. Many wavelet video coders relied exclusively on rate scalability (provided through bit-plane coding) - other scalability types were rarely supported or included in the design.

Temporal scalability allows for efficient frame-rate adjustment of the decoded video. It is typically implemented by terminating temporal synthesis of motion-compensated temporal DWT at a desired temporal resolution (target frame-rate). In the example of Figure 4·1, this corresponds to removing the leftmost T^{-1} synthesis block from the decoder. With standard dyadic structure of the DWT, practical temporal scalability implementations are limited to a "factor of two" reductions in frame-rate. Recently, more general M-band temporal filter-banks were proposed (Tillier and Pesquet-Popescu, 2004), providing different frame-rate decimation factors (e.g., 3).

Finally, the importance of spatial video scalability increased significantly over the last few years. The dramatic proliferation of visual displays, from cell phones, through video iPods, PDAs, and notebooks, to high-quality HDTV screens, has raised the demand for a scheme capable of decoding a scalably-encoded video at a range of supported video resolutions and with high quality. Spatial scalability is typically achieved by termination

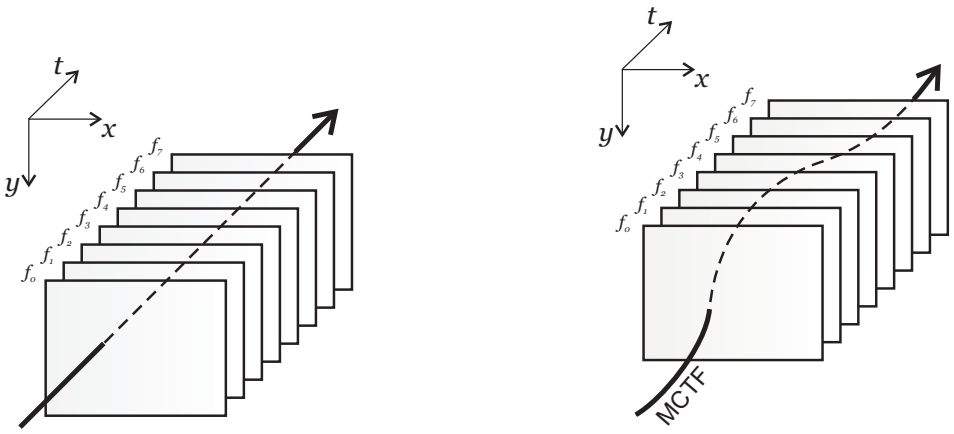
of spatial synthesis at the desired spatial resolution level followed by motion-compensated temporal synthesis on the reduced-resolution frames. We devote special attention to the spatial scalability problem in Chapter 8.

One important problem related to the evaluation of scalability performance is that of a proper reference selection, against which the coding performance should be measured. This is especially true for the case of spatially and temporally scalable decoding. For example, the dynamic range of coefficients in the low frame-rate sequence (approximation temporal subband) is larger than that of the original frames. The dynamic range also varies depending on the number of temporal decomposition levels and on the design of temporal filters used for analysis, which automatically requires the use of different PSNR definition. The poor reference choice and its variation between different encoders may result in a very good PSNR but low subjective quality. Most of the time, the reconstruction at the highest available bitrate (nearly lossless coding) is used as a reference for that particular encoder. Visual quality of the reconstructed approximation subband might also be used for comparison, which still remains very subjective (as artifacts may differ depending on the encoder settings).

4.3 Motion compensated temporal filtering (MCTF)

Motion-compensated temporal filtering is a novel approach to temporal video decorrelation that exploits excellent energy compaction properties of the DWT. In Section 2.3.2, we already introduced two wavelet kernels, Haar and LeGall 5/3, for the case of one-dimensional input. These two wavelets are frequently used in the MCTF context for one-dimensional temporal filtering of three-dimensional video data. The main idea of the MCTF is illustrated in Fig. 4·2; instead of simple temporal filtering along the t axis (Fig. 4·2(a)), temporal filtering along motion trajectory is performed (Fig. 4·2(b)) for maximal temporal decorrelation.

We now introduce MCTF notation and terminology. Let f_k denote the k-th frame of an image sequence, and let $\vec{x}$ denote spatial position of a pixel in this frame. Also, let $M_{k\to l}(\vec{x})$



(a) (b)

Figure 4·2: Temporal filtering: (a) Without motion compensation, temporal filtering is simply performed along the time axis. (b) For maximum temporal decorrelation, filtering should follow motion trajectory.

denote the motion transformation that maps frame k onto the coordinate system of frame l, as shown in Fig. 4·3(a)[1]. Specifically, motion vector field that represents a mapping $M_{k\to l}(\vec{x})$ is anchored (tails of the motion vectors) in frame l and referenced (arrow-heads of the motion vectors) in frame k. In the discrete case, this usually corresponds to a mapping of possibly non-grid positions from the frame k to grid positions of frame l.

At this point, we also define terms *"forward"* and *"backward"* motion fields, which were intuitively used before. If the current video frame has a reference in the temporally preceding frame (Fig. 4·3(b)), we call the corresponding motion field *"backward"* and denote it as $M_k^B(\vec{x})$. On the other hand, if the current frame uses future frame as a reference, we use the term *"forward"* motion field and denote it as $M_k^F(\vec{x})$ (Fig. 4·3(c)). It is obvious that both motion vector fields originate in the current frame and have their tails aligned with the grid points, whereas vector arrow-heads might point to grid or non-grid positions (depending on the particular motion model used) in either the previous frame (for $M_k^B(\vec{x})$) or the subsequent frame (for $M_k^F(\vec{x})$). The final part of our MCTF terminology includes

[1]For now, we do not make any assumption on the specific model for the mapping $M_{k\to l}(\vec{x})$, except that all pixels in frame f_k are mapped onto the frame f_l. Later in this text, several popular motion models, such as block- and mesh-based models, are analyzed in this context.

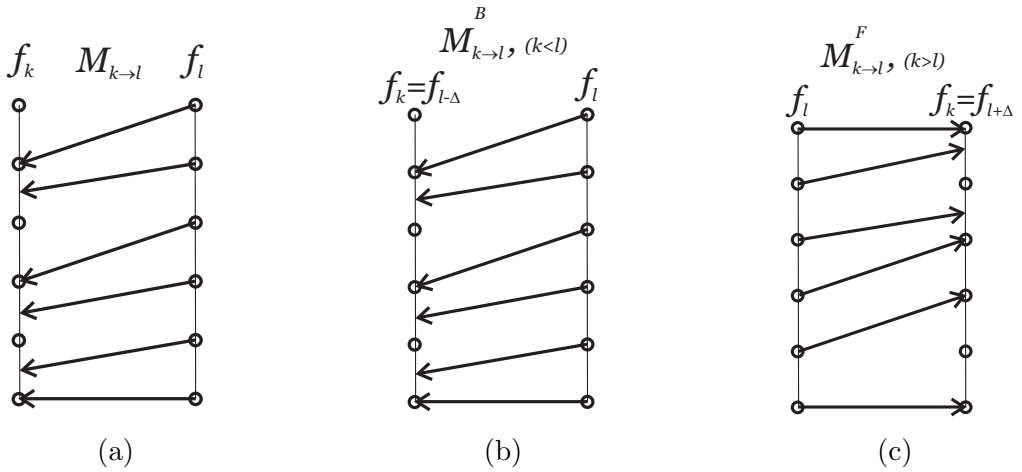


Figure 4·3: (a) Motion mapping $M_{k\to l}$; (b) "Backward" motion field; (c) "Forward" motion field.

terms *"direct"* and *"reverse"* motion; we use them to denote two motion mappings between a single frame-pair (one "backward" and one "forward"). The first motion field (the direct field) is directly estimated, and the second motion field (the reverse field) is typically derived (inverted) from the direct field.[2]

4.3.1 Transversal MCTF implementation

A single stage of the original, *transversal* temporal DWT decomposition/reconstruction process is shown in Fig. 4·4. Input video frames are filtered using low and high pass filters and subsequently temporally subsampled. In multi-resolution analysis, this procedure is repeated on the resulting low pass temporal subband l_k. Two filters, G and H, are typically selected from the class of *quadrature mirror filters*.

Analysis equations for transversally implemented temporal Haar transform without motion compensation are similar to the one-dimensional example from Section 2.3.2. Using our notation, these can be straightforwardly written as follows (for the sake of simplicity,

[2]Note that we can not simply use terms "backward" and "forward" motion to describe such a motion pair; as we will later see for the case of bidirectional prediction, "forward" fields are equally often directly estimated as "backward" fields.

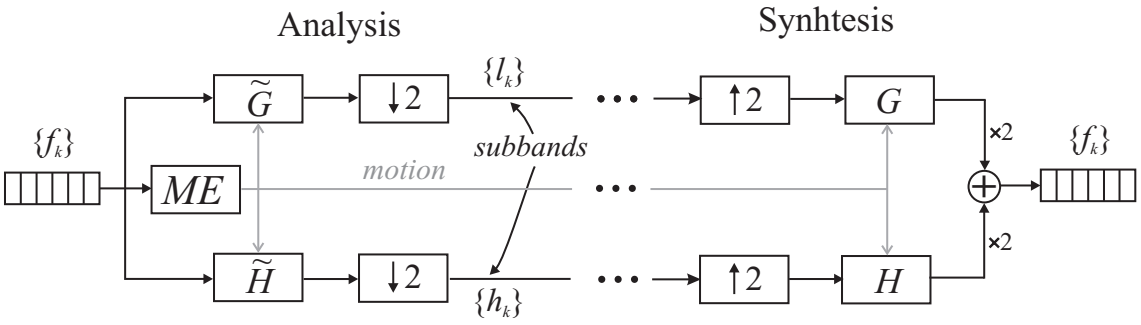


Figure 4·4: A single step of (motion-compensated) wavelet analysis/synthesis - transversal implementation.

we leave out the scaling coefficients):

$$\begin{aligned} h_k[\mathbf{x}] &= f_{2k+1}[\mathbf{x}] - \widetilde{f}_{2k}[\mathcal{M}_{2k\to 2k+1}(\mathbf{x})] \\ l_k[\mathbf{x}] &= f_{2k}[x] + \frac{1}{2}\widetilde{h}_k[\mathcal{M}_{2k+1\to 2k}(\mathbf{x})], \end{aligned} \tag{4.1}$$

In the decoder, the original video frames are easily reconstructed using the synthesis equations:

$$\begin{aligned} \hat{f}_{2k}[\vec{x}] &= l_k[\vec{x}] - \frac{1}{2}h_k[\vec{x}], \\ \hat{f}_{2k+1}[\vec{x}] &= \frac{1}{2}h_k[\vec{x}] + l_k[\vec{x}]. \end{aligned} \tag{4.2}$$

Similarly, in the case of non-motion-compensated 5/3 biorthogonal wavelet, transversal analysis steps are:

$$\begin{aligned} h_k[\vec{x}] &= f_{2k+1}[\vec{x}] - \frac{1}{2}(f_{2k}[\vec{x}] + f_{2k+2}[\vec{x}]), \\ l_k[\vec{x}] &= \frac{3}{4}f_{2k}[\vec{x}] + \frac{1}{4}(f_{2k-1}[\vec{x}] + f_{2k+1}[\vec{x}]) - \frac{1}{8}(f_{2k-2}[\vec{x}] + f_{2k+2}[\vec{x}]), \end{aligned} \tag{4.3}$$

while synthesis equations for the 5/3 DWT can be written as:

$$\begin{aligned} \hat{f}_{2k}[\vec{x}] &= l_k[x] - \frac{1}{4}(h_{k-1}[\vec{x}] + h_k[\vec{x}]), \\ \hat{f}_{2k+1}[\vec{x}] &= \frac{3}{4}h_k[\vec{x}] - \frac{1}{8}(h_{k-1}[\vec{x}] + h_{k+1}[\vec{x}]) + \frac{1}{2}(l_k[\vec{x}] + l_{k+1}[\vec{x}]). \end{aligned} \tag{4.4}$$

In order to maximize energy compaction of the temporal DWT transform, filtering should be performed along motion trajectories, leading to motion-compensated transver-

sal Haar and 5/3 equations. In the general case of sub-pixel motion mapping, motion-compensated data samples are obtained through the process of spatial interpolation - such samples are denoted by $\bar{f}$. The analysis equations for the Haar case are:

$$\begin{aligned} h_k[\vec{x}] &= f_{2k+1}[\vec{x}] - \bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x})), \\ l_k[\vec{x}] &= \frac{1}{2} f_{2k}[\vec{x}] + \frac{1}{2}\bar{f}_{2k+1}(M_{2k+1\to 2k}(\vec{x})), \end{aligned} \tag{4.5}$$

while for the 5/3 case:

$$\begin{aligned} h_k[\vec{x}] &= f_{2k+1}[\vec{x}] - \frac{1}{2}(\bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x})) + \bar{f}_{2k+2}(M_{2k+2\to 2k+1}(\vec{x}))), \\ l_k[\vec{x}] &= \frac{3}{4} f_{2k}[\vec{x}] + \frac{1}{4}(\bar{f}_{2k-1}(M_{2k-1\to 2k}(\vec{x})) + \bar{f}_{2k+1}(M_{2k+1\to 2k}(\vec{x})))- \\ &\quad \frac{1}{8}(\bar{f}_{2k-2}(M_{2k-2\to 2k}(\vec{x})) + \bar{f}_{2k+2}(M_{2k+2\to 2k}(\vec{x}))). \end{aligned} \tag{4.6}$$

As illustrated in Fig. 4·3, we use $M_{k\to l}$ to denote a motion filed that maps reference frame f_k (and its potentially non-grid intensities) onto the integer grid of current frame f_l. In equations above, for example, $M_{2k\to 2k+1}$ denotes that f_{2k} is a reference frame, and that f_{2k+1} is the current (anchor) frame. We use this notation throughout the thesis.

The inclusion of sub-pel motion into transversal DWT, while improving the energy compaction compared to non-motion-compensated MCTF (4.1 and 4.3), also leads to a very significant problem, namely loss of transform invertibility. As a result, the reconstruction error is non-zero even when the quantization step is entirely skipped. This is of great concern, especially for the high quality reconstruction performance, as the number of cascaded temporal filtering steps must be limited to prevent the error buildup. The cause of this error is found in the non-ideal spatial interpolation required for sub-pel motion compensation. To demonstrate this, we take a look at the synthesis equations for the Haar case (similar result holds in the 5/3 case):

$$\begin{aligned} \hat{f}_{2k}[\vec{x}] &= l_k[\vec{x}] - \frac{1}{2}\bar{h}_k(M_{2k+1\to 2k}(\vec{x})), \\ \hat{f}_{2k+1}[\vec{x}] &= \frac{1}{2} h_k[\vec{x}] + \bar{l}_k(M_{2k\to 2k+1}(\vec{x})). \end{aligned} \tag{4.7}$$

Note that for sub-pixel motion precision, we have to use interpolated values (denoted by $\bar{h}$

and $\bar{l}$) of high and low subbands calculated earlier. Substituting for h_k and l_k, we get the following reconstruction of even and odd frames, respectively:

$$\hat{f}_{2k}[\vec{x}] = \frac{1}{2} f_{2k}[\vec{x}] + \frac{1}{2}\bar{f}_{2k+1}(M_{2k+1\to 2k}(\vec{x})) - \frac{1}{2}\bar{f}_{2k+1}(M_{2k+1\to 2k}(\vec{x}))+ \frac{1}{2}\overline{\bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x}))}(M_{2k+1\to 2k}(\vec{x})) \tag{4.8}$$

$$\hat{f}_{2k+1}[\vec{x}] = \frac{1}{2} f_{2k+1}[\vec{x}] - \frac{1}{2}\bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x})) + \frac{1}{2}\bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x}))+ \frac{1}{2}\overline{\bar{f}_{2k+1}(M_{2k\to 2k+1}(\vec{x}))}(M_{2k+1\to 2k}(\vec{x})). \tag{4.9}$$

In both equations, the second and third elements of the sum cancel out. However, the last element of the sum requires interpolation of data previously obtained through another interpolation.[3] Therefore, when sub-pixel motion is used, any non-ideal interpolation (using finite-support interpolation filter) will inevitably lead to the loss of perfect reconstruction. Even if forward and backward motion fields are perfectly matched (i.e., true inverses of one another), no perfect reconstruction is possible. This is best explained using the one-dimensional example in Fig.4·5. Original data (open circles) is interpolated (triangles) at half-point positions using linear interpolation (Fig.4·5(a)). The reconstruction (solid circles) from such interpolated data fails to match the original sample values (Fig.4·5(b)), even when exact interpolation positions[4] - integer points in our case - are available. When cubic interpolation is used (Fig.4·5(c)), the reconstruction error decreases but it is not completely eliminated (Fig.4·5(d)).

Due to this loss of perfect reconstruction, the originally proposed MCTF method (Ohm, 1994) was limited to integer-pel accuracy motion and translational block-matching motion. Ohm's approach basically implements (4.5).[5] Because of the independent motion of adjacent blocks, block-derived motion trajectories may in practice overlap. This means that

[3] In our notation, first interpolation is denoted with a small bar symbol; the obtained frame is then interpolated again, which is marked by a large bar symbol.

[4] Simulating ideally matched backward and forward "motion shifts".

[5] Although in his original proposal subbands were referenced differently - the high subband was aligned with the reference motion-predicted frame instead of the current frame.

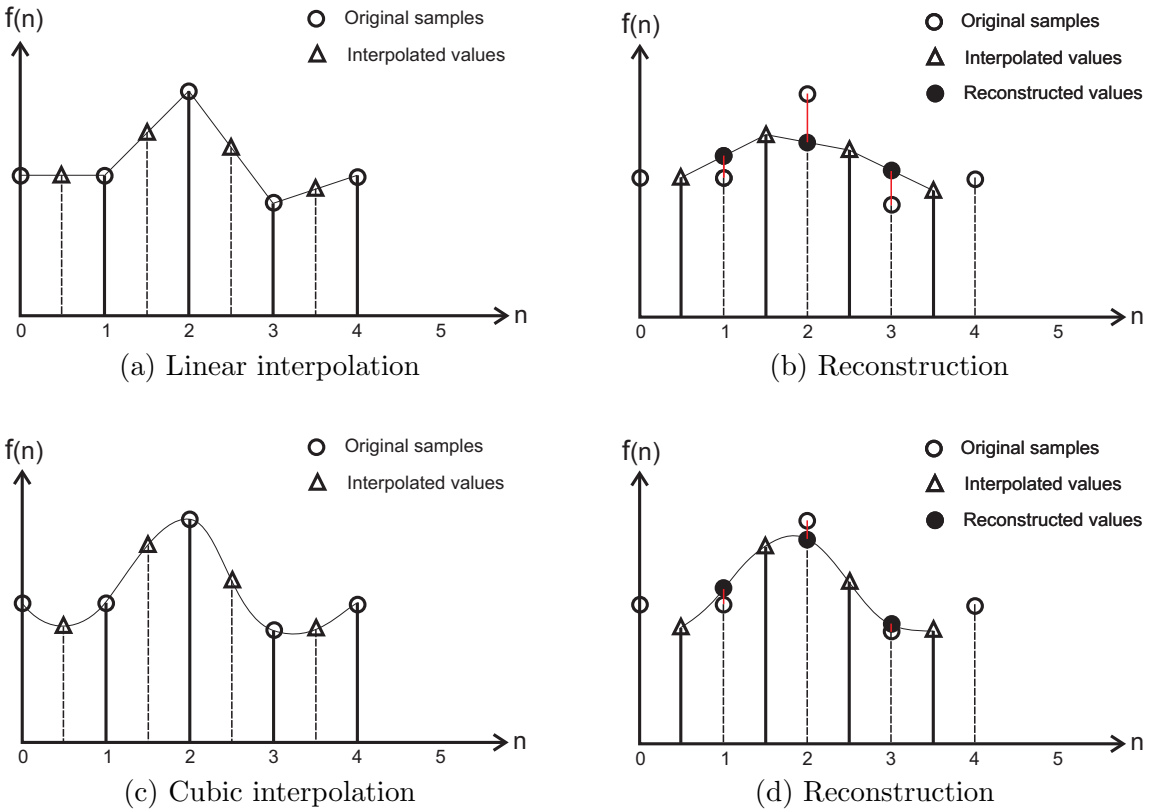


Figure 4·5: Loss of perfect reconstruction due to interpolation of data already obtained through interpolation. Reconstruction error is larger for linear interpolation (top row) than for cubic interpolation (bottom row).

some "disconnected" pixels, mostly found at the block boundaries, will be used for MCTF more than once, while other pixels will not be used at all. Only those pixels that do not fall into either of these two categories (the so-called "connected" pixels) can be processed by simple motion-compensated Haar filtering of (4.5). To that end, "inverted" motion vectors are derived by a simple change of motion direction.

Disconnected pixels, however, require special treatment. In Ohm's approach, the so-called "uncovered" pixels, aligned with the low-subband, are simply copied directly into the lowpass frame. This is a reasonable thing to do from the compression point of view, since the statistics of the original sample values and lowpass coefficients are very similar. However, the same approach should not be used for the so-called "covered" pixels in the highpass frame, as that would dramatically increase the energy of the high subband and

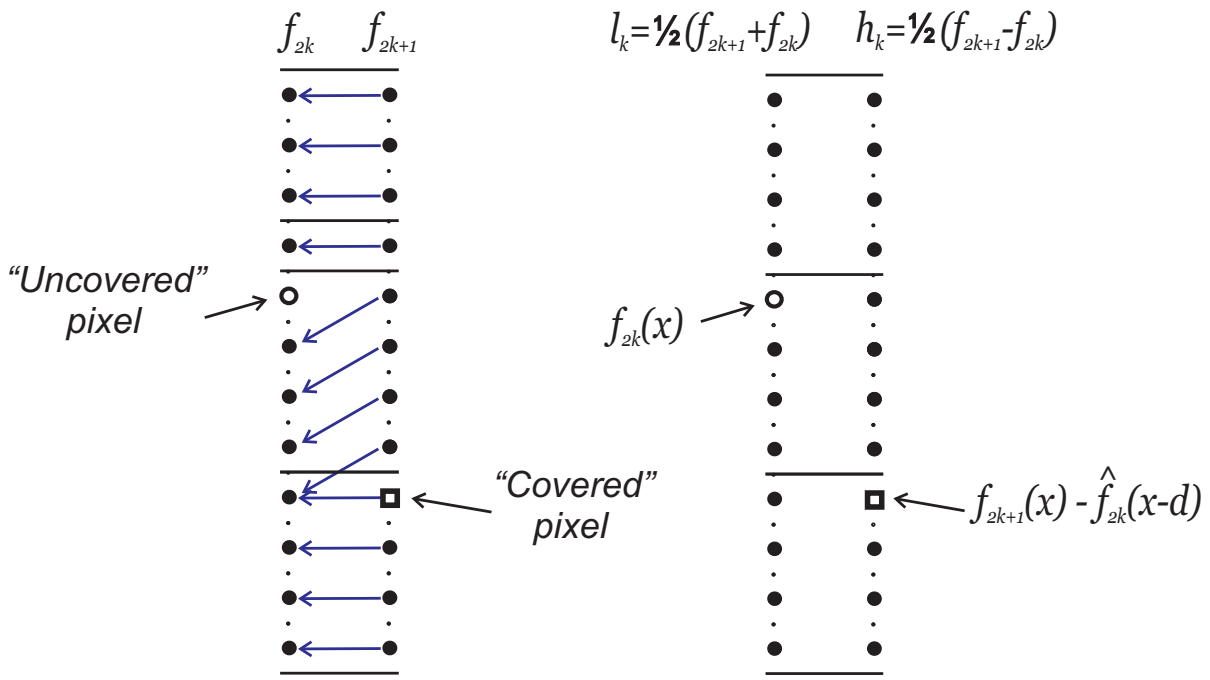


Figure 4·6: Problem of disconnected pixels in block displacement schemes.

result in coding losses. Instead, a spatial prediction of the covered pixels is used. As long as the correct classification procedure is followed by the decoder, there is no need for transmission of connected/disconnected classification map.

Choi and Woods (Choi and Woods, 1999) later improved Ohm's proposal, by realigning highpass subbands in a more natural way with the current (typically odd) video frames. Also, spatial prediction of the covered pixels was avoided and replaced with a simple temporal prediction. Their approach, that is commonly used in many current DWT-based coders, is illustrated in Fig. 4·6. However, the problem of transform invertibility limited their approach to integer-pel motion mapping too.

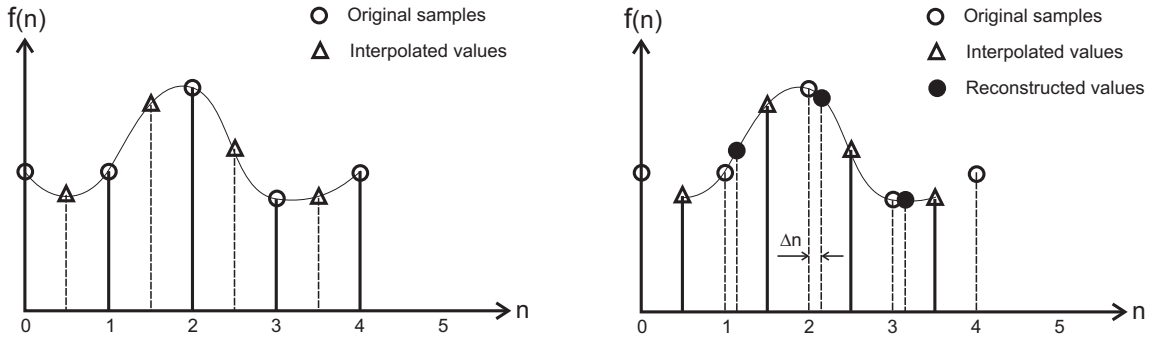
Eventually, half-pel block-based motion was integrated into transversal MCTF scheme by Hsiang and Woods (Hsiang and Woods, 1999), using a technique originally developed for interlaced/progressive scan conversion. This method creates a higher resolution lattice from pixels along each block's motion trajectory, and applies temporal filtering over this higher density lattice. Extension to even higher motion accuracies using this approach

is possible, although cumbersome. Quantitatively, the error caused by the non-invertible transform ranges from 50 dB (for two levels of temporal decomposition) to 33 dB (four levels) (Hsiang et al., 2004). This error is also sequence dependent, and varies with the choice of a particular spatial interpolating kernel.

Taking a different route, another MCTF implementation was developed by Taubman and Zakhor (Taubman and Zakhor, 1994), through invertible translational spatial warping of video frames prior to the application of separable 3D-DWT. The spatial warping operation results in image sequence where spatial features are temporally aligned, which increases the temporal correlation of a video sequence subjected to 3D DWT. In practice, the alignment is performed independently over groups of frames. These schemes are particularly well-suited to exploiting camera pan, where the scene motion consists essentially of global translation. In the work of Taubman and Zakhor, the frame warping is also referred to as "camera pan compensation".

Neither of the described methods, however, can correctly account for motion more complicated than translation, regardless of the order of kernel used for interpolation. The scenario of a locally expanding or contracting motion field essentially corresponds to a non-uniform resampling of the video frame. Consider the example of a contractive motion field between two frames whose discrete pixel samples are obtained by sampling the underlying "continuous" image sequence at the Nyquist rate. As features become closer together in the "zoomed-out" frame, certain high spatial frequencies are inevitably lost when such motion mapping is applied in the discrete image domain. This violation of the Nyquist sampling criterion prevents transversal MCTF implementation from deploying advanced spatial motion models and still remaining invertible.

Going back to our analysis of the transversal DWT, so far we assumed that both required motion mappings, $M_{2k+1\to 2k}$ and $M_{2k\to 2k+1}$, are perfect inverses of each other. This helped us in pointing out to the problems of spatial interpolation and sampling, and their effects on the non-invertibility of the transform. While improved spatial interpolation helps in reducing coding loss, we have seen that not even the ideal interpolation can guarantee



(a) Sinc interpolation (b) Reconstruction

Figure 4·7: Loss of perfect reconstruction due to non-invertible motion mappings.

a perfect reconstruction in the areas of expansive or contracting motion. Another scenario where perfect reconstruction is not attainable is that of a forward/backward motion field pair that is not perfectly matched. This is illustrated in Fig. 4·7. Reconstruction error in this example is caused by the discrepancy between the original and final sampling point, i.e., by the motion mismatch.

For the case of non-expansive motion and ideal interpolation, it can be shown from (4.8) and (4.9) (Konrad, 2004), that sufficient conditions for perfect reconstruction of motion-compensated Haar temporal filtering are:

$$\begin{aligned} \mathcal{M}_{2k\to 2k+1}(\mathcal{M}_{2k+1\to 2k}(\vec{x})) &= \vec{x}, \\ \mathcal{M}_{2k+1\to 2k}(\mathcal{M}_{2k\to 2k+1}(\vec{x})) &= \vec{x}. \end{aligned} \tag{4.10}$$

The above equations state that motion transformation must be invertible. If the motion transformation $\mathcal{M}_{k\to l}$ is invertible, then the Haar MCTF obeys perfect reconstruction. Even if motion is not invertible, perfect reconstruction may still hold for certain sequences. Note, however, that if we require that perfect reconstruction condition be *always* satisfied, regardless of image sequence $\{f_k\}$, i.e., regardless of its content, then motion invertibility is the *necessary and sufficient* condition in the case of the MCT Haar transform.

Similarly, the sufficient conditions for perfect reconstruction in the 5/3 MCTF case

(under the assumption of sinc interpolation and local translation) are (Konrad, 2004):

$$\begin{aligned}
\mathcal{M}_{2k\to 2k-1}(\mathcal{M}_{2k-1\to 2k}(\vec{x})) &= \vec{x},\\
\mathcal{M}_{2k\to 2k+1}(\mathcal{M}_{2k+1\to 2k}(\vec{x})) &= \vec{x},\\
\mathcal{M}_{2k+1\to 2k}(\mathcal{M}_{2k\to 2k+1}(\vec{x})) &= \vec{x},\\
\mathcal{M}_{2k+1\to 2k+2}(\mathcal{M}_{2k+2\to 2k+1}(\vec{x})) &= \vec{x},
\end{aligned} \tag{4.11}$$

plus a number of conditions of the general form:

$$\begin{aligned}
\mathcal{M}_{k\to n}(\mathcal{M}_{n\to l}(\vec{x})) &= \mathcal{M}_{k\to l}(\vec{x}),\\
\mathcal{M}_{k\to n}(\mathcal{M}_{n\to l}(\vec{x})) &= \mathcal{M}_{k\to m}(\mathcal{M}_{m\to l}(\vec{x})).
\end{aligned} \tag{4.12}$$

While the first set of conditions (4.11) states that motion must be invertible, the second set states that composition of motion operators $\mathcal{M}_{k\to n}$ and $\mathcal{M}_{n\to l}$ must be a valid motion operator. This will be the case when the domain of operator $\mathcal{M}_{k\to n}$ is a subset of the range of the operator $\mathcal{M}_{n\to l}$. Note that, for $k = l$, motion composition (4.12) implies motion invertibility.

It is now clear that three strong conditions must hold for the perfect reconstruction of transversally implemented MCTF in the most general case: 1) purely translational motion, which does not produce loss of high spatial frequencies due to motion-compensated resampling, 2) full invertibility of a motion pair, and 3) ideal (sinc) interpolation of video samples.

While being sufficient, these three conditions are not necessary. As described earlier, in a particular case of transversal Haar MCTF using block matching, solutions may exist that guarantee a perfect reconstruction even when the last two of these conditions are violated (Ohm, 1994; Choi and Woods, 1999).

Transversal DWT formulations have been extensively used for temporal wavelet processing from the early days of subband video coding. Still, the number of restrictions set forth by the transform invertibility problem has limited the efficiency of this method. Fortunately, this changed in 2001, when a new tool in the form of *lifting* MCTF implementation

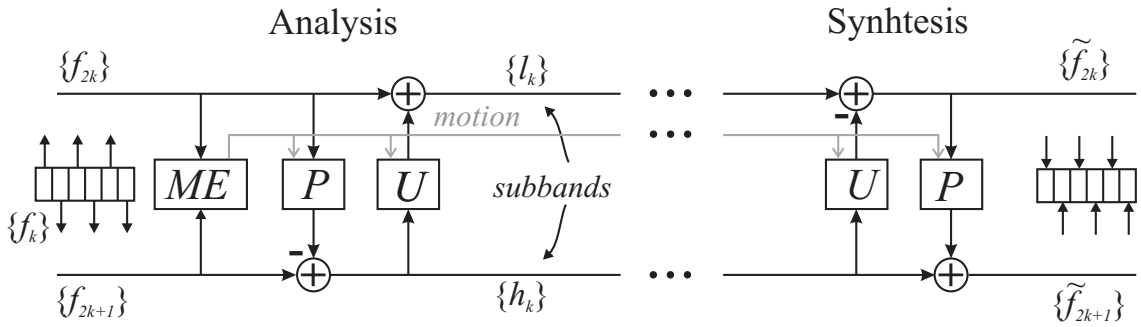


Figure 4·8: Transversal implementation of motion-compensated subband decomposition

was introduced into video coding.

4.3.2 Lifting MCTF implementation

The lifted wavelet transform (Sweldens, 1996) offers an alternative implementation of the wavelet transform overcoming most of the above problems. Its importance lies in the fact that the invertibility of the lifted wavelet transform is not compromised when the input samples undergo non-linear operation, as we will see in a moment. For this reason, lifting has become a preferred method for temporal DWT video analysis (Pesquet-Popescu and Bottreau, 2001; Luo et al., 2001; Secker and Taubman, 2001), as it can easily handle arbitrary motion compensation. Lifting also leads to computational speed-up when compared to the standard implementation (for kernels longer then Haar) and offers significant reduction in memory requirements (Daubechies and Sweldens, 1998).

A general design of lifted MCTF is illustrated in Fig 4·8. The analysis stage consists of two successive steps: prediction (P) and update (U). In the prediction step, odd frames are predicted from (motion-compensated) even frames. The prediction error forms the corresponding high-pass subband, traditionally aligned with odd input frames. In the update step, the "approximation" subband is then obtained by updating even frames with a scaled linear combination of high-subband samples, effectively forming low-pass temporal subband.

Using lifting implementation, analysis and synthesis steps of Haar DWT can be ex-

pressed as:

$$\begin{aligned} analysis\ \ (prediction): &\qquad h_k[\vec{x}] = f_{2k+1}[\vec{x}] - f_{2k}[\vec{x}] \\ analysis\ \ (update): &\qquad l_k[\vec{x}] = f_{2k}[\vec{x}] + \frac{1}{2}h_k[\vec{x}] \end{aligned} \tag{4.13}$$

$$\begin{aligned} synthesis\ \ (inv.\ update): &\qquad \hat{f}_{2k}[\vec{x}] = l_k[\vec{x}] - \frac{1}{2}h_k[\vec{x}] \\ synthesis\ \ (inv.\ prediction): &\qquad \hat{f}_{2k+1}[\vec{x}] = h_k[\vec{x}] + f_{2k}[\vec{x}]. \end{aligned} \tag{4.14}$$

The analysis equations of the lifted 5/3 DWT can be written as follows:

$$\begin{aligned} prediction: &\quad h_k[\vec{x}] = f_{2k+1}[\vec{x}] - \frac{1}{2}(f_{2k}[\vec{x}] + f_{2k+2}[\vec{x}]) \\ update: &\quad l_k[\vec{x}] = f_{2k}[\vec{x}] + \frac{1}{4}(h_{k-1}[\vec{x}] + h_k[\vec{x}]). \end{aligned} \tag{4.15}$$

Similar to the Haar case, the inverse update and prediction steps complete the synthesis:

$$\begin{aligned} inv.\ update: &\qquad \hat{f}_{2k}[\vec{x}] = l_k[\vec{x}] - \frac{1}{4}(h_{k-1}[\vec{x}] + h_k[\vec{x}]) \\ inv.\ prediction: &\quad \hat{f}_{2k+1}[\vec{x}] = h_k[\vec{x}] + \frac{1}{2}(f_{2k}[\vec{x}] + f_{2k+2}[\vec{x}]). \end{aligned} \tag{4.16}$$

By carrying out the temporal decomposition without motion compensation, however, significant energy of the transformed signal will remain in the high subband, which is not desirable from compression point of view. To reduce energy in the high subband and improve compression, motion is incorporated into the temporal lifting steps. For the Haar case, a set of analysis equations commonly used in many 3D-DWT coders is:

$$\begin{aligned} prediction: &\quad h_k[\vec{x}] = f_{2k+1}[\vec{x}] - \bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x})) \\ update: &\quad l_k[\vec{x}] = f_{2k}[\vec{x}] + \frac{1}{2}\bar{h}_k(M_{2k+1\to 2k}(\vec{x})), \end{aligned} \tag{4.17}$$

while the 5/3 motion-compensated analysis steps are:

$$\begin{aligned} prediction: h_k[\vec{x}] &= f_{2k+1}[\vec{x}] - \frac{1}{2}(\bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x})) + \bar{f}_{2k+2}(M_{2k+2\to 2k+1}(\vec{x}))) \\ update: l_k[\vec{x}] &= f_{2k}[\vec{x}] + \frac{1}{4}(\bar{h}_{k-1}(M_{2k-1\to 2k}(\vec{x})) + \bar{h}_k(M_{2k+1\to 2k}(\vec{x}))). \end{aligned} \tag{4.18}$$

In the above equations, $\bar{f}$ denotes interpolated value. Synthesis steps are easily derived; we present them here only for motion-compensated lifted Haar DWT:

$$\begin{aligned} \textit{inverse update}: \quad & \hat{f}_{2k}[\vec{x}] = l_k[\vec{x}] - \frac{1}{2}\bar{h}_k(M_{2k+1\to 2k}(\vec{x})) \\ \textit{inverse prediction}: \; & \hat{f}_{2k+1}[\vec{x}] = h_k[\vec{x}] + \hat{f}_{2k}(M_{2k\to 2k+1}(\vec{x})). \end{aligned} \tag{4.19}$$

In stark contrast to transversal implementation, no new interpolation is required in either of the two synthesis steps. More accurately, the synthesis interpolation step can perfectly reproduce all the results of the analysis interpolation step, for as long as the same interpolating kernels are used. This fact guarantees perfect reconstruction regardless of the order of interpolating kernel or motion accuracy. Furthermore, forward and backward motion mappings need not to be matched to one another. Finally, expanding or contracting motion is longer an obstacle. Amazingly, even the use of completely arbitrary motion fields, while inevitably degrading the compression efficiency, would still result in a perfectly invertible transform! While the above result was based on a discussion of the Haar DWT, similar conclusions also hold for the 5/3 DWT and longer kernels.

The use of 5/3 and longer kernels for MCTF, however, leads to two alternatives in the way motion estimation is performed: *unidirectional* or *bidirectional* (Luo et al., 2003; Golwelkar, 2004). In the unidirectional approach (Fig. 4·9(a)), just like in basic predictive coding, all motion vector fields are *backward* and anchored in every video frame. This approach permits the direct use of readily available motion estimation algorithms. In the bidirectional case, the estimated motion fields always originate at the frames aligned with the high temporal subband (typically, odd frames), and alternate in pointing backward and forward, as shown in Fig. 4·9(b). This is discussed further in Chapter 5.

4.4 Interpretation of the lifted MCTF

In the previous section, we have seen that the lifted MCTF does away with the problem of transform invertibility. We now turn our focus to interpretation of this potent temporal decorrelation method. By comparing (4.1) and (4.13) for the Haar case, and (4.3) and

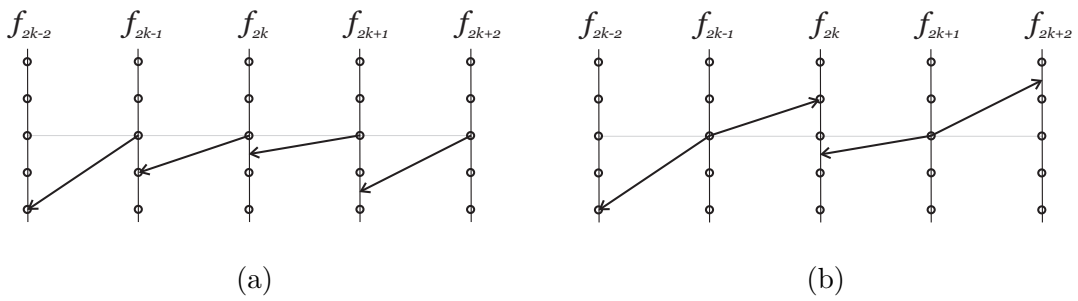


Figure 4·9: Motion estimation for MCTF: a) unidirectional; b) bidirectional.

(4.15) for the 5/3 case, it is trivial to show that lifting is equivalent to transversal filtering implementation in the absence of motion. However, maximum energy compaction requires for the DWT to be used along motion trajectories. Filter banks used for the subband decomposition are usually designed under a certain optimality constraint (Vetterli and Kovacevic, 1995). As we have already seen, in order to preserve the transform invertibility, these filters had to be modified (Ohm, 1994) in all areas where motion is not purely translational (e.g., boundaries of a disjoint blocks). In practice, lifting is used instead, as the perfect reconstruction is not compromised by motion compensation. However, when lifting steps include motion compensation, the question arises as to what kind of subband decomposition the lifting really implement, and how this is related to original (and optimized) transversal filter banks. This problem was initially brought to attention and analyzed in (Secker and Taubman, 2003; Konrad, 2004).

When there are no local expansions or contractions in the video sequence, interpretation of lifting in the context of motion trajectory filtering is pretty straightforward. Assuming ideal sinc interpolation, it can be shown (Secker and Taubman, 2003; Konrad, 2004) that in order for lifting to truly match the optimal filter bank implementation (i.e., to implement filtering along motion trajectories), the employed motion mappings must be invertible. More specifically, the following relation has to hold in the general case of longer wavelet

kernels (Konrad, 2004):

$$\begin{aligned} M_{2(k-m)\to 2k}(\vec{x}) &= M_{(k-i-j)\to 2(k-i)+1}(M_{2(k-i)+1\to 2k}(\vec{x})), \\ M_{2(k-m)+1\to 2k+1}(\vec{x}) &= M_{(k-i-j)+1\to 2(k-i)}(M_{2(k-i)\to 2k+1}(\vec{x})), \end{aligned} \tag{4.20}$$

where k, i and j are temporal frame indices and $m = i + j$. These conditions require, in general, that motion composition be well-defined, and, for some values of m (e.g., $m = 0$ in (4.20)), that motion invertibility hold. This result demonstrates that only in the context of motion mappings that satisfy the above conditions can the lifting framework be truly understood as being equivalent to subband decomposition along the motion trajectories. This is not the case when $M_{k\to i}$ and $M_{i\to k}$ are not inverses of one another, despite preserved transform invertibility.

In the more complex case of expanding or contracting motion, there is no clear way of fully understanding the behavior of the lifting MCTF. Still, if video frames in the neighborhood of a shrinking/expanding frame are modeled as a sum of a two sequences (Secker and Taubman, 2003), one containing low and the other containing high spatial frequencies, the interpretation of filtering along motion trajectories might still be used for the "low-pass" video sequence. For this to be valid, properly designed motion-compensating interpolation filters should be used. The range of spatial frequencies for which the above interpretation of lifting works will depend on the interpolation order, as well as on the nature and level of motion activity (i.e. the severity of expansion/contraction). Lifted MCTF of the remaining "high-pass" component does not have a similar interpretation, but this component is usually significantly less important than the base (low-pass) component. This is especially the case when video frames are spatially smooth, or when motion in the sequence is close to translation.

4.5 Importance of motion trajectories (to invert or not to invert?)

Referring to equations (4.17) and (4.18), it can be seen that for each forward motion warping operator, $M_{2k\to 2k+1}$, the reverse warping operator, $M_{2k+1\to 2k}$, is also required. We

have just seen that the process of motion-compensating the lifting steps is equivalent to applying the temporal DWT along an underlying set of motion trajectories, as long as the these two motion fields are inverses of one another. Commonly used discontinuous motion models such as block-based models cannot be strictly inverted, so some sort of approximation must be employed. Later, in Chapter 7, we introduce two advanced motion models, deformable mesh-based and cubic spline-based models, which do not require significant approximations for inversion.

Clearly, there are two choices for the selection of a motion field pair. One is to directly estimate one motion field involved in lifting, and then derive the other one, using some inversion method (or its approximation). A more detailed discussion of this topic will follow in Chapter 5).

Another approach is based on the obvious extension of predictive schemes: independent optimization of the parameters of each individual motion mapping, with respect to a displaced frame difference measure. In this case, $M_{2k\to 2k+1}$ and $M_{2k+1\to 2k}$ are obtained through independent forward and backward motion estimation. Regardless of the motion model used, such estimation will not generally lead to two motion mappings being inverses of one another. Even in the absence of modeling or estimation errors, discrepancies between $M_{2k\to 2k+1}$ and $M_{2k+1\to 2k}$ can be expected in areas of occlusion and exposure. In such areas, the relationship between successive frames cannot be truly described in terms of a set of motion trajectories.

In order to compare these two approaches, we computed the reconstruction peak signal-to-noise ration (PSNR) obtained by using inverted block-based motion, with that obtained by estimating every motion field independently. For inversion, we adopted a rudimentary inversion method that simply reverses the sign of the motion field (we later refer to this as "neighbor-frame-copy" inversion in Section 5.4.2). In these experiments, we use the lifted Haar and 5/3 motion-compensated temporal transform, at texture (no motion) bit-rate of 1024 kbps. Tables 4.1 and 4.2 show results for 1 and 3 levels of temporal decomposition, respectively. Fixed size 16×16 blocks with quarter-pel motion accuracy and JPEG2000

Table 4.1: Independent estimation vs. motion-inversion: luminance PSNR performance [dB] at 1024 kbps (motion rate not included), single level of temporal decomposition

Sequence	*Coastguard*		*Foreman*		*Stefan*	
Lifted MCTF	Haar	5/3	Haar	5/3	Haar	5/3
Ind. estimation	31.12	33.13	32.67	34.71	26.71	29.18
Inversion	(+0.11)	(+0.04)	(+0.08)	(+0.02)	(+0.08)	(+0.04)

Table 4.2: Independent estimation vs. motion inversion: luminance PSNR performance [dB] at 1024 kbps (motion rate not included), three levels of temporal decomposition

Sequence	*Coastguard*		*Foreman*		*Stefan*	
Lifted MCTF	Haar	5/3	Haar	5/3	Haar	5/3
Ind. estimation	34.27	35.93	36.63	39.03	29.75	32.04
Inversion	(+0.34)	(+0.15)	(+0.29)	(+0.10)	(+0.32)	(+0.14)

derived subband coder were used to encode three standard CIF-resolution test sequences: "Coastguard", "Foreman", and "Stefan".

We can see that motion inversion always outperforms the baseline case of two independently-estimated motion fields. This confirms that, unlike in hybrid coding schemes, prediction error is not the only metrics that plays role in the determination of current motion estimate. To the contrary, this result suggests that the overall coding performance also improves when "direct" and "reverse" motion fields are "well matched", i.e., closer to being inverses of each another. Also, note that the presented results do not include the cost of motion information, thus fairly comparing temporal decorrelation of the two methods. Including the motion cost would further reduce the performance achieved with independently-estimated motion fields. This is because there is no need to transmit motion parameters that are determined by inversion of another motion field, but independently-estimated motion fields must each be encoded and transmitted.

These results confirm our previous theoretical discussion and interpretation of the lifting

MCTF. From the compression point of view, it is beneficial to apply wavelet filters along known set of motion mappings (even when they are not perfectly matched to the real motion activity), than to predict and update video frames using independently-estimated motion fields.

4.6 Conclusions

In this chapter, we have set up the stage for motion analysis in the context of wavelet video coders. We have introduced motion-compensated temporal filtering as a powerful method for temporal video decorrelation and discussed technical limitations of motion-compensated transversal temporal DWT.

In order to stay invertible, practical implementations of this transform were either based on ad-hoc techniques (Ohm, 1994), designed exclusively for full-pixel block-matching, or used global motion frame warping, ignoring local expansions and contractions (Taubman and Zakhor, 1994). Neither of these approaches corresponds to a true subband decomposition along motion trajectories in the case of non-translational motion, which is common in the majority of real-life video sequences. The employment of advanced (sub-pel) local motion in the transversal DWT framework is limited due to the loss of perfect reconstruction and error caused by non-ideal interpolation.

The motion-compensated temporal lifting transform overcomes these problems. Virtually any motion may be used without the loss of transform invertibility. However, in order for the lifting transform to truly implement wavelet filtering along motion trajectories, and thus achieve maximum coding gain, two motion mappings used for lifting must be "well matched" i.e., be inverses of each another.

When motion mapping is not invertible, there is no clear way to understand the behavior of lifting transform in the context of temporal filtering, despite the preserved transform invertibility. Our experimental results confirm that the compression performance benefits from enforcing set motion trajectories (through inversion), compared to independently-estimated backward/forward motion field pair.

In the next chapter, we will investigate the motion invertibility problem in detail and show how the coding gain could be increased by improving inversion, even for motion model that is not fully invertible (e.g., block-matching). Motion inversion will be also discussed in Chapter 6, in the context of efficient coding of backward-forward motion mapping pairs.

Chapter 5

Invertible motion for wavelet video coding

5.1 Introduction

Early in the thesis (Section 2.4), we pointed out the benefits using highly accurate motion estimation (e.g., 1/4 pixel) for the compression efficiency of a video coder. In the previous chapter, we have also seen that better motion prediction alone does not lead to maximum coding gains of wavelet video coders. We have compared two methods for obtaining motion for the update lifting step and showed that motion inversion consistently outperforms direct estimation in terms of the overall coding gain. This is a very important practical confirmation of a theoretical result showing that only in the context of invertible motion mapping will the lifted DWT truly implement filtering along motion trajectories (Secker and Taubman, 2003; Konrad, 2004).

This chapter starts with a detailed analysis of the update lifting step in Section 5.2; motion inversion plays a central role in this problem. Obviously, not all motion models (including the most popular block-based model) can be perfectly inverted. We therefore propose a new metric for measuring the motion invertibility error (Section 5.3). We then propose several advanced motion inversion methods in Section 5.4 and use the invertibility error to evaluate their coding performance in Section 5.5. As better motion inversion results in higher coding gain, we also propose to incorporate invertibility into the process of motion estimation (Section 5.6) by introducing additional invertibility factor in the typical cost function (2.2.1). We discuss how this modification of the motion estimation algorithm affects both coding performance and computational complexity. At the end of the chapter, we discuss an alternative "truncated" wavelet transform, which requires no

motion inversion as no MC filtering is used in the update step (Section 5.7). The impact of motion overhead on the selection of the best temporal DWT kernel is also discussed. Experimental results comparing various motion/transform combinations are shown in the Section 5.8. Section 5.9 concludes this chapter.

5.2 The importance of motion invertibility

The issue of motion invertibility is specific to wavelet video coding, as no inversion of motion is required in hybrid coding schemes. Before a further invertibility analysis, we point out to the main difference between the two coding schemes, which lies in the way they handle temporal decorrelation. Hybrid coders use *motion-compensated temporal prediction* (MCTP), while video frames are decorrelated using *motion-compensated temporal filtering* (MCTF) in wavelet coders. In order to obtain high and low temporal subbands, a *pair* of motion mappings (backward and forward) between each two consecutive frames is required. In the lifting context, these two motion mappings are used one after another in two lifting stages, prediction and update.

As we have seen in Section 4.3.2, the prediction lifting step of a wavelet coder closely follows the motion prediction stage of a hybrid coder. The goal of prediction is to minimize the bit rate needed to represent the motion-compensated residual. Since this bit rate is closely related to the energy of the prediction error (i.e., high-band signal), one can use this energy instead, in order to find the best motion vectors.

Motion-compensation, however, does not end with the prediction step in the MCTF framework. In the update step, a motion field (with direction opposite to that used in prediction) controls the update of original video frames with samples from the high band. The existence of two motion fields, representing motion between the same two frames but in opposite temporal directions, enables different options when estimating motion and using it in MCTF:

1. independent estimation of both prediction and update motion fields,

2. direct estimation of the prediction motion field and derivation of the update motion field (through motion inversion),

3. estimation of the prediction field and omission of MCTF in the update step (low-pass band is, in this case, generated by completely skipping the filtering part).

The first method produces motion fields each of which guarantees the lowest prediction error. However, as we have seen in Section 4.5, this does not necessarily lead to better compression performance. One reason for this is motion overhead; the other, less obvious, lies in the inherent mismatch between this motion compensation approach and the structure of the wavelet video coder.

Many wavelet video coders rely on the second method, in which one motion field is estimated and the other one is derived, using motion inversion. Such schemes introduce no additional motion overhead and achieve good coding performance. On the negative side, motion inversion requires additional computational complexity, which can vary from negligible (in the case of trivial inversion) to very high. Depending on the inversion quality, this process can result in a set of well-defined motion trajectories, benefiting the coding performance.

Finally, the third approach calls for an omission of the filtering part in the update step. In the absence of low-pass temporal filter, this transform is sometimes referred to as the "truncated wavelet transform". The low-pass temporal subband is formed by simply subsampling original video frames. Such an approach makes sense from the coding perspective because the statistics of original frames are similar to those of the temporal low band. Still, the use of truncated kernel introduces significant aliasing and reduces the maximum achievable compression gain. This is especially true for high quality, full frame-rate coding; indeed this approach is most commonly used in low bit-rate, low-complexity implementations. More detailed discussion of truncated wavelet filtering is presented in Section 5.7.

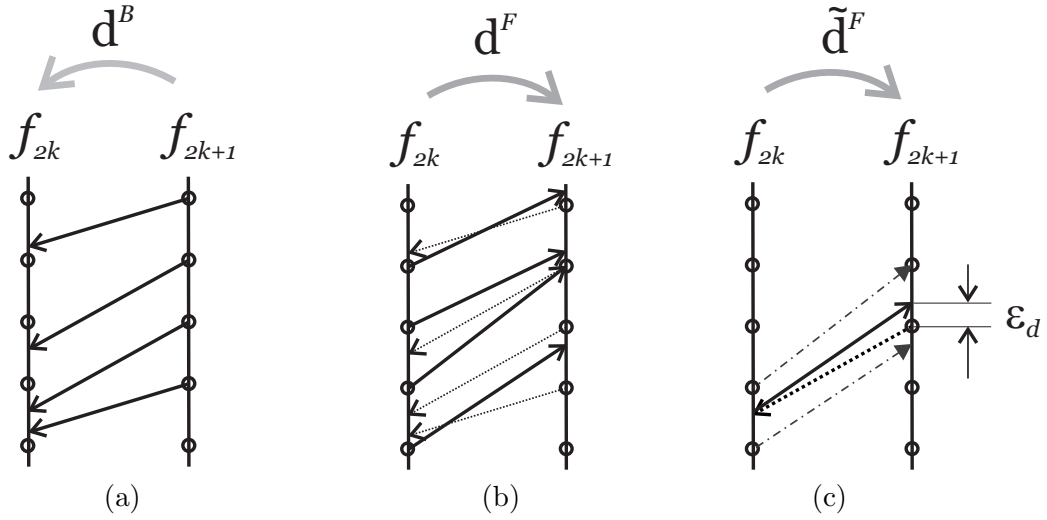


Figure 5·1: Illustration of motion invertibility error. In solid lines: (a) Backward motion $\mathbf{d}^B$; (b) Forward motion $\mathbf{d}^F$; (c) Interpolated motion $\bar{\mathbf{d}}^F$, derived from $\mathbf{d}^F$. Invertibility error at a pixel in f_{2k+1} is defined as the distance between the tail of a motion vector anchored at that pixel and the tip of the corresponding vector $\bar{\mathbf{d}}^F$.

5.3 Motion invertibility error metrics

Ideally, two motion fields (forward and backward) that are required for a single step of MCTF should both match a unique set of motion trajectories.[1] In this ideal case, we say that these two motion vector fields are perfect inverses of each another. In any practical situation, however, due to imperfections of motion modeling and estimation algorithms, the trajectory set defined by one motion field will often fail to match that represented by another field. In order to objectively measure how far one motion field is from being the perfect inverse of another field, we propose the following *invertibility error*:

$$\epsilon_d = \sum_{\vec{x}} |\mathbf{d}^B(\vec{x}) - \bar{\mathbf{d}}^F(\vec{x} + \mathbf{d}^B(\vec{x}))|. \tag{5.1}$$

where $\mathbf{d}^F = [d_x^F, d_y^F]^T$ and $\mathbf{d}^B = [d_x^B, d_y^B]^T$ are forward and backward motion vectors, respectively, defined for every integer pel position in the respective anchoring frame. $\bar{\mathbf{d}}$

[1] If motion rendering is accurate, this set of trajectories will closely match the true motion of the real-world objects.

denotes interpolation of x and y components of $\mathbf{d}$ at non-grid positions.

To the best of our knowledge, our work is the first to propose a comprehensive framework for quantitative analysis of forward/backward motion field relationship in the context of video coding. Prior related work on this topic has been done primarily in the field of stereo matching (Izquierdo, 1997) and applied to the problems of occlussion/uncovered area detection and intermediate view reconstruction. However, the invertibility error metric (5.1) is a novel concept and the original contribution of this dissertation.

For the example in Fig. 5·1, this error measures the sum of departures of points in frame f_{2k+1} when each of them is projected onto frame f_{2k} using the backward motion field and then forward-projected onto frame f_{2k+1} using the (estimated or derived) forward motion field. A pair of motion fields being perfect inverses of each other would result in zero error ϵ_d. Note that the invertibility error is defined without any assumptions on the nature of motion fields involved - it can be computed for any two motion mappings $\mathbf{d}^B$ and $\mathbf{d}^F$. When both motion fields are known (e.g., independently estimated), the task of calculating the invertibility error is straightforward, as it only requires interpolation of motion field $\mathbf{d}^F$. This interpolation belongs to a simpler, regular-to-irregular interpolation category (Glassner, 1995).

It is important to note that, because of this interpolation step, the motion invertibility error introduced in (5.1) is not uniquely defined. This metric depends on the choice of particular interpolation kernel used for regular-to-irregular interpolation discussed above, in addition to motion fields $\mathbf{d}^B$ and $\mathbf{d}^F$. In the most general case, there is no prior that would suggest the use of a particular interpolation model - in reality, once these two motion fields are known, any interpolating kernel can be used to obtain a valid measure of motion invertibility. For our experiments presented in Section 5.5 we selected to use the popular cubic spline interpolator (Unser, 1999), as it presents a nice combination of good performance and manageable computational complexity.

In most practical coding scenarios, however, one motion field (e.g., forward) is derived from another (e.g., backward) using motion inversion. When fractional-pel motion accuracy

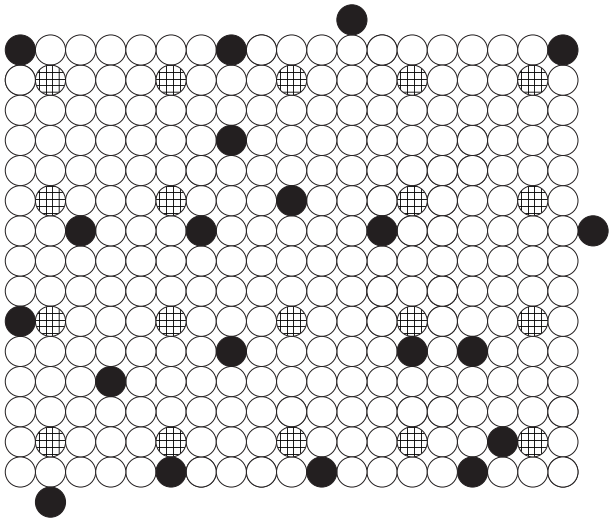


Figure 5·2: Example of irregular-to-regular motion-field interpolation: reconstruction of motion vectors at regular positions (hashed) is based on the knowledge of vectors at irregular positions (black).

is used, tips of motion vectors from the direct field are not aligned with the grid in the reference frame f_{2k}. A derivation of the reverse field is equivalent to the process of irregular-to-regular interpolation, illustrated in Fig. 5·2. This example is constructed for quarter-pixel motion precision - it illustrates the fact that, based on the knowledge of motion components at irregular grid points (black), we need to recover motion at the regular grid points (hashed). This is the reason why, in practical wavelet video coding context, the motion interpolation problem is not trivial. The relation to Fig. 5·1 is that black disks correspond to arrowheads of motion vector field $\mathbf{d}^B$ and hashed disks correspond to tails of motion vector field $\mathbf{d}^F$. In the next section, we propose several methods for practical inversion of motion fields that do not have one-to-one mapping property (Section 2.2.1) and cannot be trivially inverted. A prime example of such motion is, naturally, ubiquitous block-based motion model.

5.4 Motion inversion algorithms

In the case of integer-pel accurate motion, the standard (ad-hoc) approach to generating motion for the update step (Ohm, 1994; Choi and Woods, 1999) was earlier presented in

Section 4.3.1. It suggests the use of reversed motion vectors at the positions where they are well defined (simple-connected pixels). If a reverse motion vector is not defined at all (unconnected pixel), this method inserts the sample value of this unconnected pixel into a corresponding low subband. For multiply-connected pixels, the decision on what motion trajectory should be used for filtering in the update step is based on scan order; the first encountered pixel that uses the current multiply-connected pixel as a predictor is used as a reference in the update step. With the introduction of fractional-pel motion, nearest-neighbor interpolation of a "direct" motion field was proposed (Choi and Woods, 1999; Chen and Woods, 2002). This method copies a motion vector from the nearest non-grid position less than half-pel away from the current (on grid) pixel, inverts its sign, and uses it in the update step. For unconnected and multiply-connected pixels, the same strategies as before are used.

In order to investigate the effect of the forward-backward motion field coupling on coding performance, we now look at different ways of meaningful inversion of a motion field. This will also help us exploit the inversion techniques presented here in efficient joint coding of a motion field pair between the same two frames/subbands. For that reason, and to focus solely on the effects of motion inversion, inversion over the entire image domain is performed (no intra pixel replacement is used). Our compression assumption is that a wavelet coder will benefit from better matched (in terms of invertibility error ϵ_d) motion field pairs. To test this, we analyze progressively more complex methods for motion field inversion.

5.4.1 Motion inversion based on collinearity assumption

One of the simplest methods for motion inversion is based on the so-called "collinear-extension" (Valentin et al., 2003). It assumes collinearity between the forward and backward motion vectors originating at the same frame, as illustrated in Fig 5·3. This corresponds to the assumption of constant-velocity motion over three frames. This inversion method supports only unidirectional motion estimation, as it requires motion vectors that

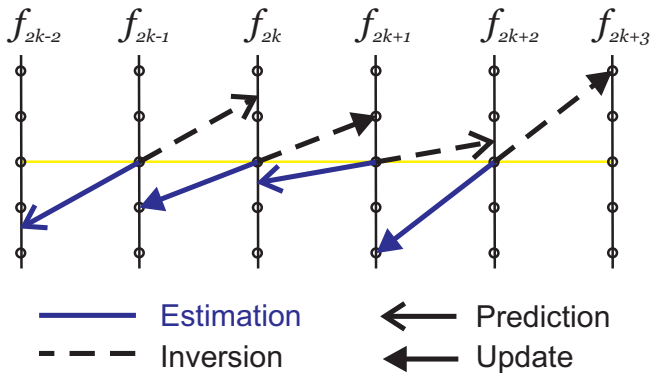
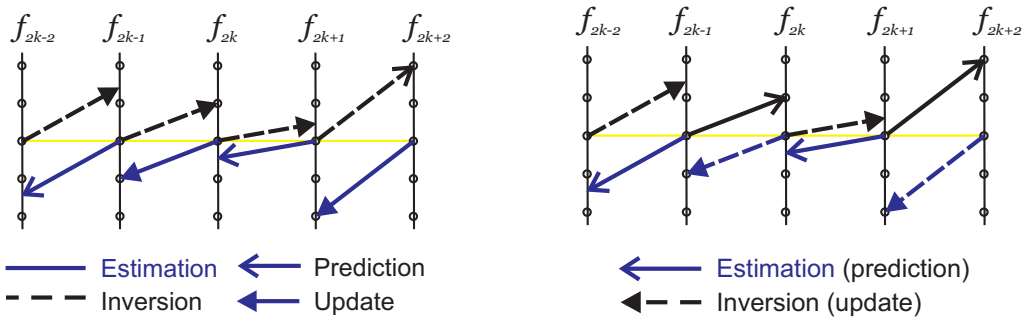


Figure 5·3: "Collinear-extension" motion inversion



(a) (b)

Figure 5·4: Neighbor-frame-copy motion inversion: (a) unidirectional case, (b) bidirectional case.

are anchored in each frame in order to function properly. Practically, this inversion changes the sign of a backward motion field anchored at f_n and assigns the result to the forward motion field anchored at the same frame f_n (note that collinear inversion is not practical for use in conjunction with the Haar DWT for the obvious reasons). This simple inversion is computationally very inexpensive, but not capable of modeling more realistic variable-velocity motion.

5.4.2 Motion inversion based on neighbor-frame-copy method

Another relatively simple motion inversion technique called "neighbor-frame-copy" (NFC), is illustrated in Fig. 5·4. This method "copies" the motion field of a neighboring frame -

as opposed to the same frame in the collinear method - with the opposite sign (Božinović et al., 2004). There are two subclasses of this method: one where motion is estimated in the unidirectional fashion (Fig. 5·4(a)) and the other with bidirectionally estimated motion (Fig. 5·4(b)). Solid lines represent motion vectors that are directly estimated from input frames using prediction error criterion, while dashed lines show vectors that are obtained through inversion. Similarly, the open arrowheads represent motion vectors used in the prediction step and closed arrowheads denote vectors that are used for the update step. As with the collinear method, the computational complexity of this inversion method is very low. It is clear from Fig. 5·4 that NFC inversion does away with motion constancy requirement. However, this inversion method is also more sensitive to errors in regions with large displacements, since the motion is assumed to be spatially uniform over the length of the motion vector. For this reason, we can expect collinear inversion to be better suited to image sequences with spatially fast-varying but temporally uniform motion. On the other hand, sequences with temporal acceleration and spatially slow-varying motion are expected to benefit from using neighbor-frame-copy inversion.

Both of these techniques compute only a coarse inverse motion field due to a simple inversion process. A proper inversion should project, through motion compensation, all grid points from the current image to the plane of the reference image, and then reverse the sign of each motion vector. As discussed above, the projected grid is irregular and some form of irregular-to-regular data interpolation is needed.

5.4.3 Nearest-neighbor motion inversion

Our nearest-neighbor motion inversion method (Božinović et al., 2004) is based on the standard ad-hoc approach (Choi and Woods, 1999) for the update step motion derivation. For all motion vectors defined on an irregular grid in the target frame, we compute the vectors (processing one motion coordinate at a time) at regular grid locations using nearest-neighbor interpolation (Fig. 5·5(a)). First, each irregular location is mapped to the nearest pixel and the associated motion vector is copied there (the pixel becomes "occupied"). In

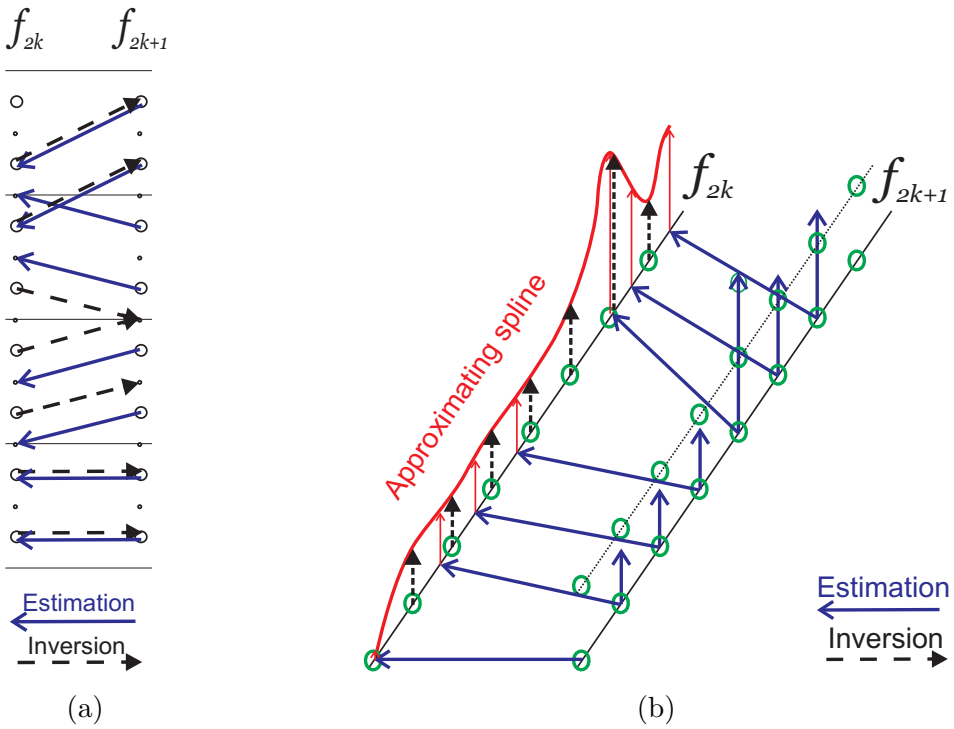


Figure 5·5: "(a) Nearest-neighbor" motion inversion, (b) Spline-based motion inversion.

contrast to the method of Choi and Woods, all "unoccupied" pixels are visited successively and assigned the motion vector from their nearest "occupied" pixel. The procedure is iterated until all pixels become "occupied". This way, no intra-substitution is required and the sole effect of motion inversion can be observed. Details of this procedure are described in a recent technical report (Zhao, 2004).

5.4.4 Spline-based motion inversion

As the nearest-neighbor interpolation is known to have suboptimal performance, we propose to use an advanced irregular-to-regular interpolation method based on spline approximation (Vázquez et al., 2005). For one-dimensional case, the spline-based inversion is illustrated in Fig. 5·5(b). Although based on cubic splines, this method is not an interpolation method since it uses a prior term related to the curvature of the computed surface for each motion component. Due to this prior, the resulting motion-component surface need not pass through the original data points induced by motion-compensated projection. Similar to

the nearest-neighbor, we apply this method twice: once for the x and once for the y component of motion vectors. Both the nearest-neighbor and spline-based inversion are more computationally involved than the first two techniques of collinear and NFC inversion.

5.4.5 Alternatives to motion inversion for the update step

Few newer solutions have been recently proposed for the the implementation of the MC update step. These methods do not aim at directly inverting the motion field; instead, the reconstruction error (after the synthesis step) is analyzed, under the assumption of additive quantization noise. The problem of finding the best reference for the update step is then solved by minimizing such a reconstruction error. This approach was independently proposed by Girod and Han (Girod and Han, 2005) and Tillier *et al.* (Tillier et al., 2004). It has been shown that the optimal update of multiply-connected pixels in the Haar case should use the weighted average of all pixels connected to this pixel as a reference. However, the proposed solution does not provide a similar simple answer in the case of longer wavelet filters or when motion different from block-matching is used. It rather relies on very large matrix inversion, which leads to extremely high computational complexity (Girod and Han, 2005). This limits practical applications of this method to low-resolution sequences.

Although seemingly different from our approach, the new proposal can also be interpreted in the light of motion field inversion. Locations of multiply-connected pixels correspond to areas of high density of motion vector "arrow-heads" from the "direct" motion field. Where smoothing-spline interpolation calculates the best corresponding inverse motion field and uses a single motion vector for the reverse trajectory (*motion vector averaging*), the approach of both (Girod and Han, 2005) and (Tillier et al., 2004) calls for averaging of intensities at multiply-connected pixels (*pixel intensity averaging*).[2] Depending on the application and available computational resources, one or the other approach might be used.

[2]This is similar to the comparison of motion field interpolation and pixel intensity interpolation of overlapped block motion compensation (OBMC).

5.5 Experimental results - motion inversion

In this section, we compare all proposed motion inversion methods. Compression settings (previously described in Section 4.5) between different coding runs are kept constant, except for the motion inversion method. We also make sure that the identical directly-estimated prediction step motion field is used in all cases. For the experiments comparing different inversion methods, we have used our own implementation of the 3D subband coder based on JPEG2000 with FSBM and half-pixel motion accuracy. These results expand on those already presented in Tables 4.1 and 4.2, adding more advanced inversion methods. In fairness to the methods that estimate both forward and backward motion fields, motion bit-rate is excluded from the reported bit budget. This allows us to focus solely on the quality of subband decomposition, without the bias of a different motion overhead.

In Table 5.1, we present luminance PSNR performance for *Coastguard* (300 frames), *Foreman* (300 frames), and *Stefan* (300 frames) CIF sequences, after a single temporal 5/3 decomposition, and average texture bit-rate of 512 kbps (similar results were observed across the range of bit-rates). Motion bit-rate is not included in the overall bit budget. Rows in the Table 5.1 denote different inversion methods (except for "5/3 Ind", where motion fields are estimated independently). Rows 2-4 show results for "neighbor-frame-copy" (NFC) inversion in the unidirectional case as well as "collinear-extension" and "neighbor-frame-copy" inversions in the bidirectional case, respectively. These three methods of inversion were described earlier and depicted in Figs. 5·3 and 5·4. The last two rows present results for nearest-neighbor (NN) inversion and spline-based approximation. Along with the coding gains, we also show inversion error values (ϵ_d), introduced earlier. It is clear that there exists a strong correlation between this measure and the coding performance.

We can see that virtually any method of motion inversion outperforms two independently-estimated motion fields. This confirms that the overall coding performance improves when the direct and reverse motion fields are well-matched, i.e., when they are closer to being inverses of each another.

Table 5.1: Comparison of different motion-inversion methods. Luminance PSNR performance [dB] at 512 kbps (motion rate not included), single level of 5/3 temporal decomposition. All sequences are of CIF resolution at 30Hz.

Configuration	*Coastguard*		*Foreman*		*Stefan*	
	PSNR	ϵ_d/pixel	PSNR	ϵ_d/pixel	PSNR	ϵ_d/pixel
Ind	33.12dB	0.24	34.71dB	0.54	29.18	0.43
Col	(+0.04)	0.21	(+0.02)	0.43	(+0.03)	0.22
NFC-Uni	(+0.03)	0.20	(+0.01)	0.40	(+0.03)	0.19
NFC-Bi	(+0.04)	0.19	(+0.02)	0.39	(+0.04)	0.17
NN	(+0.07)	0.08	(+0.05)	0.24	(+0.07)	0.17
Spline	(+0.09)	0.06	(+0.07)	0.19	(+0.11)	0.13

In Table 5.2, we show the PSNR performance for all three sequences at the total average rate of 1024 kbps (motion rate included). All motion vectors are losslessly encoded; the motion overhead ranges from 21% to 29%. We again notice the increase in PSNR for more accurate inversions of block-based motion fields. Two independently estimated motion fields are jointly coded (this is denoted as 5/3 Jnt) in order to reduce motion bit-rate. We can see that the coding gap increases even more between the inversion and "independent estimation" case, as the effects of worse subband decomposition and the increased motion overhead both reduce the overall coding gain of 5/3 Jnt approach.

These results prove that the problem of finding the best motion for both lifting MCTF steps is not straightforward. We confirmed our previous finding from Section 4.5 that motion inversion outperforms independent motion estimation. Furthermore, a better inversion can improve coding gain, at the cost of higher computational complexity of the inversion algorithm.

As mentioned earlier, the motion fields used in the prediction lifting step were kept the same in all inversion scenarios. These motion fields were optimized strictly for the prediction step. Knowing that their inverses will be used for the update step, we may ask the question if this is the right thing to do. The reason is that a non-restrictive motion

Table 5.2: Comparison of different motion-inversion methods. luminance PSNR performance [dB] at 1024kbps (motion rate included), three levels of temporal decomposition

Configuration	*Coastguard*		*Foreman*		*Stefan*	
	PSNR	ϵ_d/pixel	PSNR	ϵ_d/pixel	PSNR	ϵ_d/pixel
5/3 Jnt	32.93dB	0.26	36.91dB	0.54	30.17	0.37
5/3 Col	(+1.23)	0.20	(+0.74)	0.43	(+0.61)	0.21
5/3 NFC-Uni	(+1.19)	0.18	(+0.73)	0.40	(+0.63)	0.17
5/3 NFC-Bi	(+1.30)	0.16	(+0.79)	0.39	(+0.64)	0.16
5/3 NN	(+1.37)	0.07	(+0.84)	0.24	(+0.77)	0.14
5/3 Spline	(+1.52)	0.06	(+1.03)	0.19	(+1.11)	0.12

estimation in the prediction stage may lead to a highly irregular motion estimate, whose inversion might be more difficult and the resulting update-step motion-compensation less efficient. To this end, we suggest a motion estimation algorithm that searches for the best motion in *both* prediction and update step, simultaneously minimizing prediction error and invertibility metric.

5.6 Inversion-aware motion estimation

In this section, we introduce a new motion estimation algorithm that is *inversion-aware*. More formally, for the case of fixed-size block matching, we modify the standard motion estimation cost function (2.2.1) to include the invertibility error:

$$\vec{\hat{d}}_i \;=\; \min_{\vec{d}_i} \sum_{\vec{x}\in B_i} C(f_n(\vec{x}) - f_{n-1}(\vec{x}-\vec{d}_i)) + \lambda\, \varepsilon_d. \tag{5.2}$$

The motion estimation algorithm relying on this cost function requires two passes. In the first pass, standard non-regularized motion estimation is performed. In the second pass, a modified exhaustive motion search on macro-block basis is repeated - this time, the inverse field is also calculated for each forward field and invertibility error is added to the cost function. A motion vector that minimizes the modified cost function (5.2) is selected.

This process is repeated for each macro-block until the entire frame is processed.

As the invertibility error of block-based motion fields occurs exclusively at block boundaries, the two-step process describe above can be simplified. Instead of using invertibility error directly, we can indirectly reduce it by improving global motion smoothness. If the two neighboring blocks have similar motion vectors, the invertibility error at the block boundary is certain to be small. Therefore, global regularization of the motion search algorithm improves motion invertibility. Obviously, overly strong regularization hurts motion prediction performance and a good balance is needed.

In the context of variable-size block matching, the regularization approach to reducing invertibility error becomes even more obvious. In the HVSBM, better prediction relies on greater local variation of the motion field; in contrast, lower invertibility error is the result of more similar neighboring motion vectors. Therefore, the motion-rate regularization parameter λ can be used to to indirectly control the invertibility error. In our compression experiments presented in later chapter, we found that the best results in MCTF coder are obtained using λ somewhat higher than in a comparable hybrid coder.

We performed extensive experiments using inversion-aware motion estimation, both in the context described in the previous section, and in conjunction with the occlusion-aware lifting scheme presented in the next chapter. While we have observed small gains when fixed-size block matching was used (averaging 0.3dB), gains of the inversion-aware motion estimation were minimal (less than 0.05dB) when used with HVSBM (implemented with the block sizes ranging from 4×4 to 16×16). Overall, such small gains, and the introduced computational penalty of the two-pass method, advise against its further employment. Nonetheless, advanced inversion methods and the reduction of invertibility error will play a major role in predictive (joint) coding of a motion-field pair, required for the implementation of occlusion-aware MCTF. This topic is further discussed in Section 6.5.1.

5.7 Temporal wavelet kernels for MCTF

The ability to perform subband decomposition along motion trajectories raises a question as to which filter banks are the most effective for this task. Historically, initial efforts have been largely confined to the Haar transform. Early results (Ohm, 1994; Taubman and Zakhor, 1994) even reported no benefits from using longer filters, but extensive follow-up research indicated that consistently superior compression performance can be achieved with the 5/3 transform (Secker and Taubman, 2003; Golwelkar, 2004). The biorthogonal 5/3 kernel presents a nice blend of good decomposition performance, limited complexity, and manageable processing delay. Its drawbacks are increased motion overhead (doubled from the Haar's case) and higher latency. Kernels longer than 5/3 were also proposed and implemented (Golwelkar, 2004) without producing significant, if any, coding gains.

5.7.1 The issue of motion bitrate overhead

Motion bitrate overhead is an important issue related to the choice of DWT for temporal filtering. In the prediction step of a single-level temporal Haar decomposition shown in Fig. 5·6, one motion field per each frame-pair is needed (solid vectors). The update step also requires one motion vector field per each pair of frames. As already mentioned, in order to prevent excessive increase in motion bitrate, motion vectors used in the update step (dashed vectors) are typically derived, and not estimated. Thus, for a single level of Haar DWT temporal decomposition, the number of motion fields needed is one half of that required in standard hybrid coding. However, since temporal multi-resolution analysis (MRA) requires additional motion mappings between low-subband frames (the lower subbands in Fig. 5·6), the total number of motion vector fields turns out to be comparable for the Haar DWT case and temporal prediction case.

In the 5/3 case, it is clear from Fig. 5·7 that, because of the bidirectional prediction, the temporal 5/3 DWT requires twice as many motion vectors then the Haar DWT. Depending on the application, this motion overhead can jeopardize coding performance. This is especially true at lower bitrates and under the assumption of lossless coding of motion. Also,

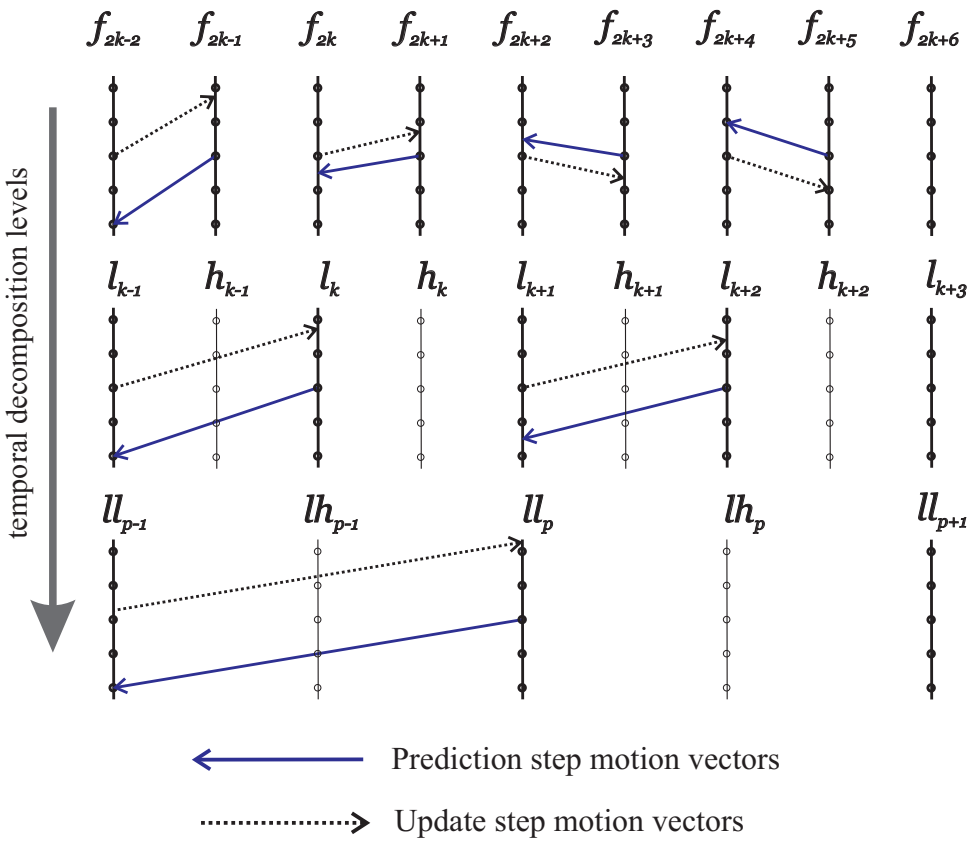


Figure 5·6: Temporal subband decomposition and motion vectors required for analysis for Haar DWT.

the computational complexity of motion inversion becomes a problem in low-complexity applications.

5.7.2 The 1/3 DWT - Truncated 5/3 wavelet kernel

In the attempt to combine good bidirectional prediction of the 5/3 DWT and low motion overhead of the Haar DWT, an alternative wavelet kernel has been recently proposed (Luo et al., 2001; André et al., 2004). It consists of the same high-pass filter as the standard 5/3 DWT, associated with a simple downsampling operation instead of low-pass filtering and downsampling:

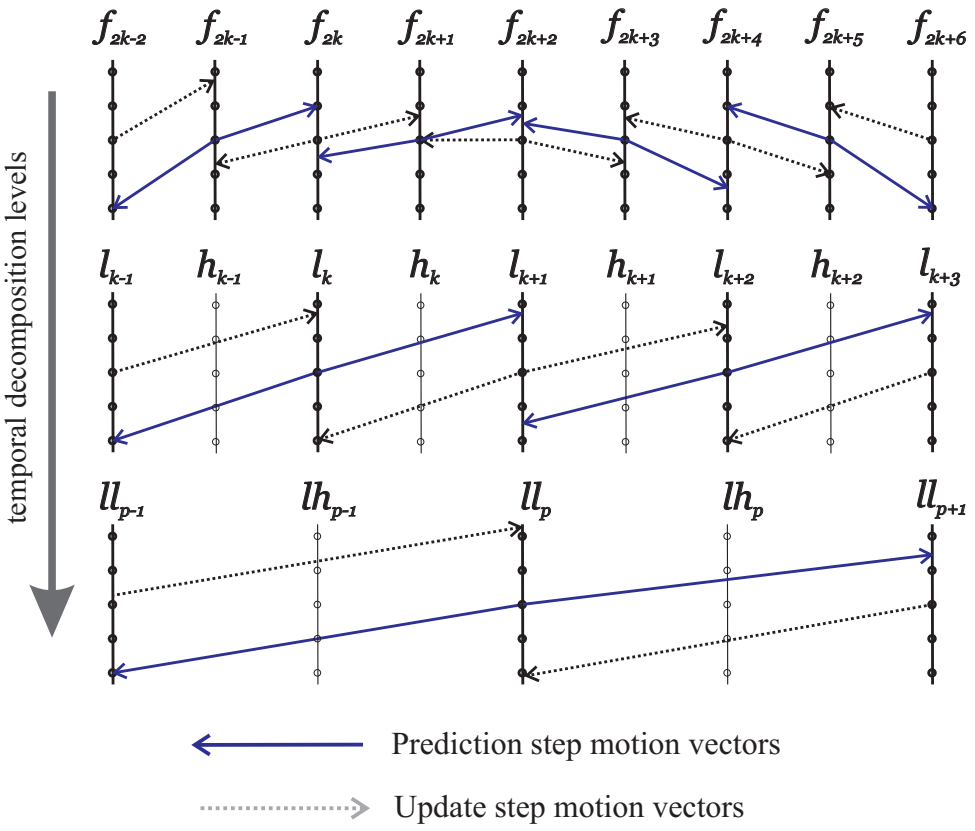


Figure 5·7: Temporal subband decomposition and motion vectors required for analysis for 5/3 DWT.

$$\begin{aligned} h_k[x] &= f_{2k+1}[x] - \frac{1}{2}(\tilde{f}_{2k}[M_{2k\to 2k+1}(x)] + \tilde{f}_{2k+2}[M_{2k+2\to 2k+1}(x)]) \\ l_k[x] &= f_{2k}[x]. \end{aligned} \tag{5.3}$$

The lowpass and highpass analysis filters in this case have transfer functions $\hat{G}(z) = 1$ and $\hat{H}(z) = 1 + \frac{1}{2}(z + z^{-1})$, which have one and three taps, respectively. Accordingly, this transform is sometimes referred to as "1/3 transversal DWT" or the "truncated" 5/3 DWT.

This temporal analysis approach has been particularly popular in fast coders, since no motion compensation is needed for the (trivial) update step. The main shortcoming of the 1/3 wavelet transform is a significant overlap of temporal frequency subbands - this introduces aliasing due to the lack of lowpass filtering (unless the original signal is

band-limited to half its Nyquist frequency, which, in general, is not the case). If, on the other hand, motion is inaccurately computed (or incapable of properly capturing true scene motion in the occlusion/exposure areas) in standard 5/3 DWT, and then used for low-pass filtering in the update step, undesirable "ghosting" effect may be visible in the low-pass subsequence. These subjectively-disturbing artifacts can, in fact, be more annoying than the error induced by additional aliasing of 1/3 DWT. In addition, these artifacts might be especially undesirable in applications requiring high reconstruction quality at lower temporal resolution (e.g., video surveillance).

5.8 Experimental results

In this section, we compare the Haar, 5/3 and 1/3 wavelet kernels for MCTF. In collaboration with the I3S laboratory of University of Nice (André et al., 2004; Božinović et al., 2004), we developed a 3D motion-compensated subband coder. We typically use three to five levels of temporal decomposition; embedded coding of these temporal subbands is performed with JPEG2000-based subband coder. The motion is computed using a standard 16×16 fixed-size block matching, with half-pel motion accuracy. The resulting vectors are losslessly encoded using JPEG-LS.

We start with the analysis of a full frame-rate coding performance of 30 fps CIF sequences *Foreman* and *Stefan*. In Tables 5.3 and 5.4 we compare compression results of Haar, 1/3, and 5/3 DWTs. For the results in the first three rows, lossless coding of all necessary estimated motion fields required for implementation of these DTWs is performed. The fourth row presents results of a coder that, in the update step, uses an "inverse" of motion estimated in the prediction step of the 5/3 transform. The inversion is performed using the "nearest-neighbor" method, described in Section 5.4.3. We denote two different configurations using the 5/3 kernel as "5/3 Ind" (5/3 DWT and independent estimation and coding of motion vectors) and "5/3 Inv" (5/3 DWT with inverted motion field in the update step).

We can notice a good performance of the Haar DWT at lower bitrates, which can be

Table 5.3: Luminance PSNR performance [dB] of various DWT kernels for 30Hz CIF *Stefan* sequence.

DWT	256kbps	512kbps	1024kbps	2048kbps
Haar	25.42	28.03	31.39	34.45
1/3	25.37	27.92	31.12	34.89
5/3 Ind	24.76	27.61	30.94	34.97
5/3 Inv	25.22	28.51	32.04	35.76

Table 5.4: Luminance PSNR performance [dB] of various DWT kernels for 30Hz CIF *Foreman* sequence.

DWT	256kbps	512kbps	1024kbps	1536kbps
Haar	30.17	32.91	35.87	37.75
1/3	29.76	32.81	36.03	38.14
5/3 Ind	25.11	30.69	35.80	37.94
5/3 Inv	29.87	32.99	36.68	38.90

attributed to small motion overhead. Also note that low bit-rate performance of 5/3 DWT configurations is relatively poor compared to both the Haar and 1/3 DWT due to significant motion overhead. Improvement in coding performance of 5/3 kernel is noticeable with the increase of decoding bit-rate, where the share of motion bit-rate in the total bit-budget decreases.

At all but the lowest bit-rate (256kbps), the "5/3 Inv" exhibits superior performance. Its gain over the Haar configuration is, as expected, small at medium bitrates (e.g., only 0.08 dB at 512 kbps for *Foreman* sequence). The reason for such a small difference might be that the motion occupies significant part of bit-budget at lower bitrates which benefits Haar DWT.

At higher bit-rates, "5/3 Inv" consistently outperforms all other configurations, by

Table 5.5: Luminance PSNR performance [dB] of 1/3 and 5/3 DWT kernels for 15Hz CIF *Stefan* sequence.

DWT	256kbps	512kbps	1024kbps	2048kbps
1/3	24.33	28.11	32.54	37.32
5/3 Inv	24.05	28.70	33.04	37.85

typically more then 1dB, which should be attributed to the improved spectral localization of longer analysis filters. We also point out the fact that the prediction step in this case benefits from two motion compensation operators unlike in the Haar case, where only one reference is used for prediction. This use of two motion mappings helps reduce the effects of model error in either motion mapping, in a similar manner to the bidirectional motion compensation commonly employed in hybrid coding. Finally, at full frame rate decoding (30Hz) the 1/3 DWT performs similarly to Haar DWT at all bitrates.

The performance of the 1/3 DWT, however, needs to be evaluated at different levels of temporal resolutions (frame-rates) or employed in adaptive manner in order to fully assess its coding potential. Despite coding loss of 1/3 DWT at high bitrates and full frame rate, if the motion model fails to capture the scene activity (e.g., occluded/uncovered regions), ghosting artifacts may arise in the low-pass temporal frames, as a result of the update steps (i.e., low-pass filtering). In that case, lower frame rate video might be reconstructed at higher quality and without visible artifacts with no motion compensation in the update step. We compare performance of 1/3 with 5/3 for reduced frame-rate *Stefan* sequence in Table 5.5; we can verify that, at reduced frame rates, the 1/3 DWT more closely follows the performance of 5/3 DWT.

5.9 Conclusions

In this chapter, we shed a significant amount of light on motion processing for the update lifting step. We proposed new motion invertibility error and showed how better inversion

of motion estimated in the prediction lifting step can improve the overall coding results. We proposed incorporating invertibility into the structure of motion estimation block. Finally, we compared performance of the three most popular wavelet kernels for the MCTF. We showed that the truncated 5/3 wavelet can improve the quality of reduced frame-rate sequences in spatial regions where the motion cannot be accurately modeled. This comes at the price of a lower full frame-rate compression performance. The next chapter demonstrates how improved results can be obtained by selectively applying the update step. In addition, our new scheme opens a possibility for fully-adaptive lifting, where the prediction lifting step is also modified. The key issue with this approach is the robust detection of spatial regions where the motion mapping does not represent real motion.

Chapter 6

Occlusion-aware wavelet video coding

6.1 Introduction

In the previous chapter, we investigated the problem of improving the update lifting step through better motion inversion. We now turn our focus to the improvement of the prediction lifting step. We investigate how the combination of occlusion-driven update and occlusion-driven prediction leads to fully adaptive motion-compensated temporal filtering and improves compression performance. We start this chapter with the analysis of simple prediction of the Haar DWT (Section 6.2). We discuss the main obstacles of occlusion-based prediction compared to occlusion-adaptive update in the Haar's case. Section 6.3 extends the discussion to the 5/3 temporal DWT. We propose a solution where all motion fields involved in both lifting steps are directly *estimated*, instead of some of them being indirectly *derived*. The proposed iterative algorithm for such motion estimation is described in detail in Section 6.4, and discussed in the context of fully-adaptive, occlusion-driven lifting (Section 6.5). Attention is given to efficient motion coding and advanced motion inversion is used for the predictive motion compression. Experimental results are presented in Section 6.6. Section 6.7 concludes the chapter.

6.2 Prediction lifting step - the Haar DWT

Motion compensation is a standard method for improving video compression - instead of original video frames, motion-compensated prediction error is coded and transmitted. In the Haar case, this prediction error corresponds to the high-pass temporal subband. The compression goal of minimizing the number of bits required for subband representation is

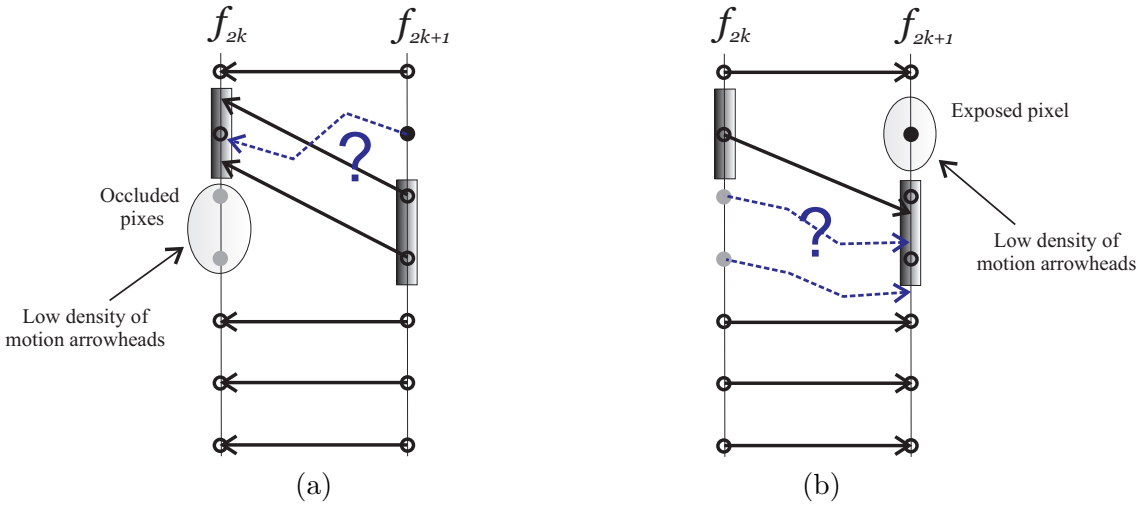


Figure 6·1: Problem of occluded and exposed areas in the Haar MCTF: a) Prediction step; b) Update step.

equivalent to minimizing the energy of the high-subband. In practice, this energy is used to find the best motion vector fields. the mean absolute error (MAE) or mean squared error (MSE) metric, and any popular motion vector search algorithm might be used.

In the case of Haar kernel, a single option for motion prediction exists, as illustrated in Fig. 6·1(a). Typically, the reference frame (denoted with f_{2k}) immediately precedes the current frame (f_{2k+1}). Motion vectors from the field $\mathcal{M}_{2k\rightarrow 2k+1}(\mathbf{x})$ (solid vectors in Fig. 6·1(a)) are used to predict the so-called "well-connected" pixels. The conventional approach to motion prediction (ISO/IEC JTC1 IS 13818-2 (MPEG-2), 1994) and the original wavelet video coding proposals (Ohm, 1994; Choi and Woods, 1999) do not treat "exposed" pixels (solid circles in frame f_{2k+1} differently from other pixels in the current frame. In reality, however, these exposed pixels are indeed without a meaningful reference. In practical coding scenarios, motion vectors corresponding to exposed pixels are assigned based not on true correspondences, but rather on local motion in the immediate neighborhood of the current "exposed" pixel, as illustrated by a dashed the motion vector in Fig. 6·1(a)).

In the Haar case, the only possible remedy is a spatial prediction of such exposed pixels (Wiegand et al., 2003). This approach, however, is limited to spatial prediction of the entire block to which the exposed pixel belongs. In other words, individual exposed pixels cannot

be predicted without incurring a high signaling cost. Since exposed areas in real video sequences are rarely co-aligned with block boundaries, the effectiveness of this approach is often limited.

From Fig. 6·1(a), it is clear that by analyzing (backward) motion $\mathcal{M}_{2k\rightarrow 2k+1}(\mathbf{x})$, anchored (defined on the grid) in current frame f_{2k+1}, one can attempt to detect occluded areas of frame f_{2k}. Indeed, this approach is effectively exploited in conventional update schemes (Choi and Woods, 1999; Chen and Woods, 2002). Due to the lack of appropriate reference, original pixel values are substituted directly into the low subband at every "occluded" pixel position. This method prevents emergence of annoying artifacts in the low temporal subbands (due to erroneous update), at the cost of introduced aliasing. The observation we make here, and that is central to the following discussion, is that the detection of occluded/uncovered pixels is possible only in the reference, and not in the current frame of the motion estimation step. In other words, complete occlusion-adaptive update should be driven by the analysis of "motion-vector tips distribution" in the reference frame.

When the direction of motion estimation is reversed (i.e., when motion $\mathcal{M}_{2k+1\rightarrow 2k}(\mathbf{x})$) is *estimated*), we can use the density distribution of its motion arrow tips in frame f_{2k+1}, in the similar manner (details of this procedure are given in Section 6.4). This can be subsequently used for proper labeling of "exposed" pixels in frame f_{2k+1}. Again, the key fact here is that a direct estimation of motion (i.e., true pixel intensity matching, and not inversion) must be used to drive this process.

When dealing with these newly exposed pixels (solid circle in frame f_{2k+1} of Fig. 6·1(b)), there is no reason to assume that they occur less frequently then unconnected (occluded) pixels in the update step. They should be treated in a different manner from other "well-connected" pixel. The use of smaller blocks (i.e. finer local motion modeling) may reduce the number of unconnected pixels and improve the overall prediction. This, however, does not solve the fundamental problem of inappropriate motion modeling - regardless of (variable-size) motion model or its accuracy, covered/uncovered pixels simply do not have a proper match in the reference frame, and should be treated differently. To make things

worse, not even a correct labeling of exposed pixels can enable the use of similar "update-step" solution - the classical approach of "intra" pixel replacement in prediction subband is simply inefficient from the coding point of view.

At the end of this section, we also note that all boundary pixels (leaving or entering the scene) fall into the category of occluded/uncovered pixels. For example, pixels leaving the scene might be treated as occluded areas while pixels entering the scene can be categorized as newly uncovered pixels. Depending on the nature of sequence at hand, they can account for a significant percentage of the total number of pixels, especially in the presence of camera pan. In Chapter 7, we will demonstrate how a even a simple modification of motion topology at frame boundaries can lead to moderate compression gains.

6.3 Prediction lifting step - the 5/3 DWT

The number of options for improving motion-compensated prediction increases in the case of the 5/3 DWT. The goal of the prediction step remains the same: to minimize the energy of the high temporal subband. However, unlike Haar's single-reference prediction, two predictions are now available, as depicted in Fig. 6·2(a). We already demonstrated in Chapter 5 that the bidirectional approach to 5/3 prediction outperforms the unidirectional prediction, as it directly minimizes the high-subband energy. Throughout the remainder of the thesis, we use the bidirectional approach to 5/3 prediction exclusively.

To perform prediction in a single temporal decomposition step of the 5/3 case, both motion fields $M_{2k\rightarrow 2k+1}$ and $M_{2k+2\rightarrow 2k+1}$ are required. The simplest and most commonly used method for bidirectional prediction is an independent estimation of both motion fields, using one of the usual distortion measures (SAD or SSD):

$$\begin{aligned} M_{2k\rightarrow 2k+1} &= \operatorname{argmin} \sum_{\vec{x}\in B} \mathcal{C}\Big[f_{2k+1}(\vec{x}) - \bar{f}_{2k}(M_{2k\rightarrow 2k+1}(\vec{x}))\Big] \\ M_{2k+2\rightarrow 2k+1} &= \operatorname{argmin} \sum_{\vec{x}\in B} \mathcal{C}\Big[f_{2k+1}(\vec{x}) - \bar{f}_{2k+2}(M_{2k+2\rightarrow 2k+1}(\vec{x}))\Big], \end{aligned} \tag{6.1}$$

Here, $\mathcal{C}$ denotes the selected distortion measure, and B is a current macro-block.

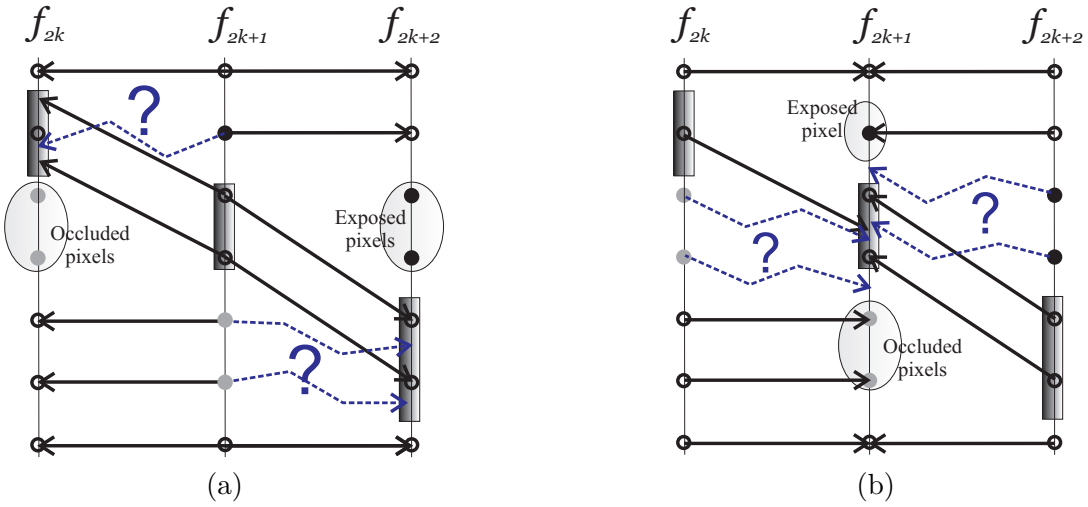


Figure 6·2: Problem of occluded and exposed areas in the 5/3 MCTF: a) Prediction step; b) Update step. Note that we use terms *occluded* and *exposed* pixels with respect to time direction and not motion vector direction (that might be opposite).

Interestingly, the 5/3 DWT provides an opportunity for even better prediction; instead of independently estimating the two required motion mappings, their joint estimation might lead to further energy reduction of the high-pass subband. The joint estimation of both motion fields can be formulated, for example, as follows:

$$(M_{2k\to 2k+1}, M_{2k+2\to 2k+1}) = \operatorname{argmin} \sum_{\vec{x}\in B} \mathcal{C}\Big[f_{2k+1}(\vec{x}) - \frac{\bar{f}_{2k}(M_{2k\to 2k+1}(\vec{x})) + \bar{f}_{2k+2}(M_{2k+2\to 2k+1}(\vec{x}))}{2}\Big]$$

Obviously, the optimal solution of this problem requires an exhaustive search of the entire state space for both motion fields. As this is an extremely computationally intensive task, a sub-optimal solution needs to be found. One of the most successful (Pau et al., 2004) iteratively solves for one motion field while keeping the other constant. This iterative estimation leads to better prediction and benefits the overall coding gain - improvements of up to 0.4 dB over independent search were reported (Pau et al., 2004).

Nevertheless, this modification fails to address the problem of inadequate motion modeling of uncovered or occluded areas. As illustrated in Fig. 6·2(a), in the prediction step of

the 5/3 lifting, some pixels in the current frame f_{2k+1} do not have a valid match in one of the reference frames (f_{2k} and f_{2k+2}). For that reason, those pixels should not be predicted at all from an inappropriate reference. Standard motion estimation algorithms neglect this fact and assign motion vector to each current frame pixel, regardless of the existence of its true reference.

It is therefore obvious that some sort of detection of the exposed/occluded areas is required for the fully-adaptive lifting, and that an interaction with the motion model is a natural way to selectively apply both of the lifting steps. As occlusion/uncovered regions can be reliably detected only in the reference (range) and not in the current frame (domain) of each motion estimation step,[1] this requires that motion be *directly estimated in both directions within each frame pair*. This observation serves as the foundation of our new iterative motion estimation method. Using estimated motion, we derive an occlusion label field, c^O and exposure label field, c^E, in each of the video frames. These label fields are used to adaptively drive both steps of the lifting scheme, as explained in the next section.

6.4 Exposure/occlusion detection and motion estimation

In order to estimate a pair of backward/forward motion fields (e.g., $\mathcal{M}_{2k\rightarrow 2k+1}(\mathbf{x})$ and
$\mathcal{M}_{2k+1\rightarrow 2k}(\mathbf{x})$) between the two video frames f_{2k+1} and f_{2k}, we propose the following iterative algorithm. We model the motion using hierarchical variable-size block matching (HVSBM), with block sizes varying from 16×16 (macro-block size), down to 4×4, similar to AVC/H.264 (Wiegand et al., 2003; Schafer et al., 2003). Without loss of generality, we first analyze the problem of motion estimation between a frame pair; this simplifies our notation as some temporal indices can be omitted and easily deducted from the context.

In the first step, the backward motion field $\mathcal{M}_{2k\rightarrow 2k+1}(\mathbf{x})$ is estimated for each macro-block (MB) in the current frame. This is done by minimization of the conventional rate-

[1] This is explained earlier in Section 6.2.

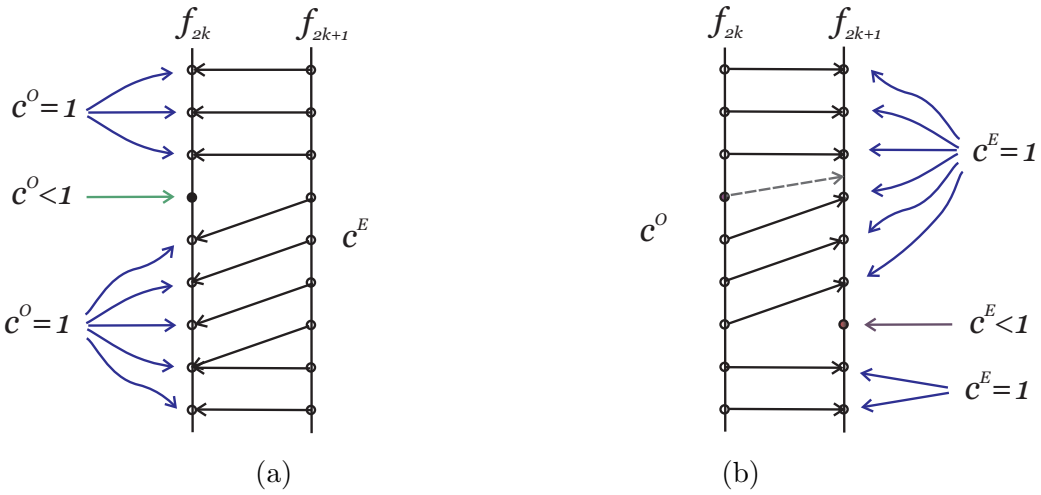


Figure 6·3: Iterative motion estimation and occlusion/exposure detection: a) Backward ME and occlusion detection. Label field c^E is initialized to 1 in the first iteration and dynamically changed during the estimation process; b) Forward ME and exposure detection. There is no need to explicitly initialize occlusion field c^O, as its first estimate is already known after the first half of the first iteration.

distortion cost function on a macro-block level:

$$J_B = \sum_{\mathbf{x} \in MB} c^E(\mathbf{x}) |f_{2k+1}(\mathbf{x}) - f_{2k}(\mathcal{M}_{2k \to 2k+1}(\mathbf{x}))| + \lambda R_B. \tag{6.2}$$

The subindex B in J_B denotes the backward direction of motion, and R_B denotes the total rate used for the coding of backward motion vectors and associated block partition map for macro-block MB. This initial motion estimation is illustrated in Fig. 6·3(a). The $c^E(\mathbf{x})$ is a label at position $\mathbf{x}$ in the coordinates of frame f_{2k+1}, and takes a value between zero and one ($0 \leq c^E(\mathbf{x}) \leq 1, \forall \mathbf{x}$). As previously mentioned, its goal is to label all newly-exposed pixels in frame f_{2k+1} as "unconnected", since they have no good reference in frame f_{2k}. Before the first iteration is started, however, c^E is unknown. That is why we initially set $c^E(\mathbf{x}) = 1, \forall \mathbf{x}$ (all pixels are assumed to be "connected"). From our previous discussion, since c^E is defined over the coordinates of f_{2k+1}, it can be estimated only by analyzing the distribution of the forward motion field $\mathcal{M}_{2k+1 \to 2k}$, anchored in another frame (f_{2k}). This also explains the iterative nature of our algorithm (illustrated in Fig. 6·4).

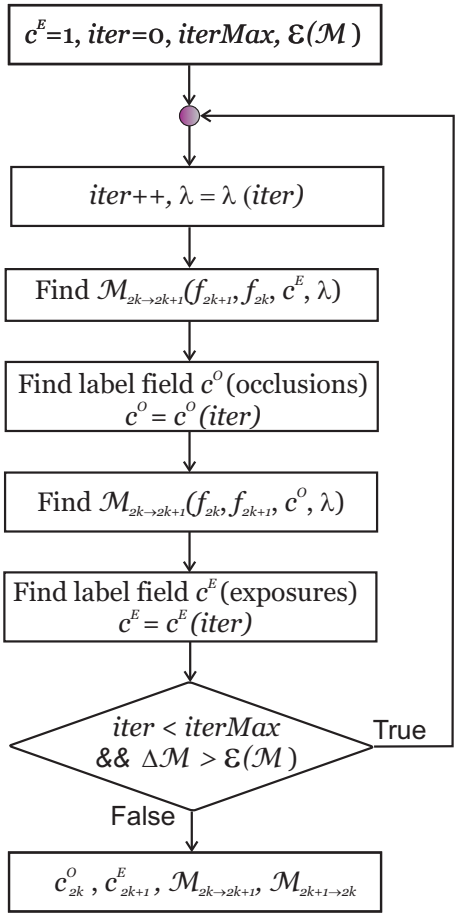


Figure 6·4: Iterative algorithm for detection of occlusion/exposure regions

The estimated backward motion field $\mathcal{M}_{2k\to 2k+1}$ can now be used to form an initial label field c^O, supported on the coordinates of frame f_{2k}. For this, we use the approach inspired on the occluded/exposed area detection method proposed by Ince and Konrad (Ince and Konrad, 2005). In brief, we calculate the density field of a motion field arrow-tips and threshold it to obtain positions of occluded/uncovered areas (i.e., all pixels with motion-arrow-tip density lower than a certain threshold are declared as "innovations").

Next, the forward motion field $\mathcal{M}_{2k+1\to 2k}(\mathbf{x})$ is estimated by minimizing:

$$J_F = \sum_{\mathbf{x}\in MB} c^O(\mathbf{x}) |f_{2k}(\mathbf{x}) - f_{2k+1}(\mathcal{M}_{2k+1\to 2k}(\mathbf{x}))| + \lambda R_F, \tag{6.3}$$

where c^O is the occlusion label field just derived in frame f_{2k}; R_F is the rate associated

with the forward motion field. Estimation of the forward motion field $\mathcal{M}_{2k+1\rightarrow 2k}(\mathbf{x})$ and subsequent derivation of label field of exposed pixels (c^E) completes the first iteration of the estimation algorithm.

In the remainder of the estimation process, J_B and J_F are minimized iteratively one after the other, with the interleaved estimation of c^E and c^O. This estimation process is stoped when either the estimated motion fields reach convergence or when the maximum number of iterations is achieved (typically 5). Most of the time, 2 to 4 iterations are sufficient for accurate detection of occluded/uncovered areas.

Obviously, the confidence in the accuracy of label fields c^O and c^E increases with the number of iterations. For that reason, occluded/uncovered pixels detected earlier do not have to be assigned a label 0 immediately. Instead, we decided to gradually reduce the value of the initial c^E/c^O label. As the iterations progress, occluded/uncovered labels of both $c^O(\mathbf{x})$ and $c^E(\mathbf{x})$ are calculated (using heuristic formula) as $c(\mathbf{x})_i = \max\{1-(\frac{i}{5})^2, 0\}$, with i being the iteration number. Naturally, the "connected" pixels maintain the label $c^E(\mathbf{x}) = c^O(\mathbf{x}) = 1$ at all times. After the final iteration, all $c^E(\mathbf{x})$ and $c^O(\mathbf{x})$ labels different from 1 are set to zero.

The novelty of this approach lies in the fact that it allows for a different contribution of "connected" and "unconnected" areas to the "live" motion estimation cost function through (monotonically decreasing) labels of fields $c^E(\mathbf{x})$ and $c^O(\mathbf{x})$. One of the main reasons for such a design is the well understood problem of ill-defined reference for un-connected pixels. Their unwanted contribution to the total distortion can now be suppressed with appropriate c^E and c^O labels. When the iteration number increases, the probability of correct occlusion/exposure labeling increases too - this further justifies the use of monotonically decreasing labels c^E and c^O.

6.5 Occlusion-adaptive lifting for 5/3 temporal wavelet filtering

We now complete the description of our fully-adaptive lifting based on occlusion/exposure detection. The bi-directional frame referencing of the 5/3 lifted DWT allows adaptive

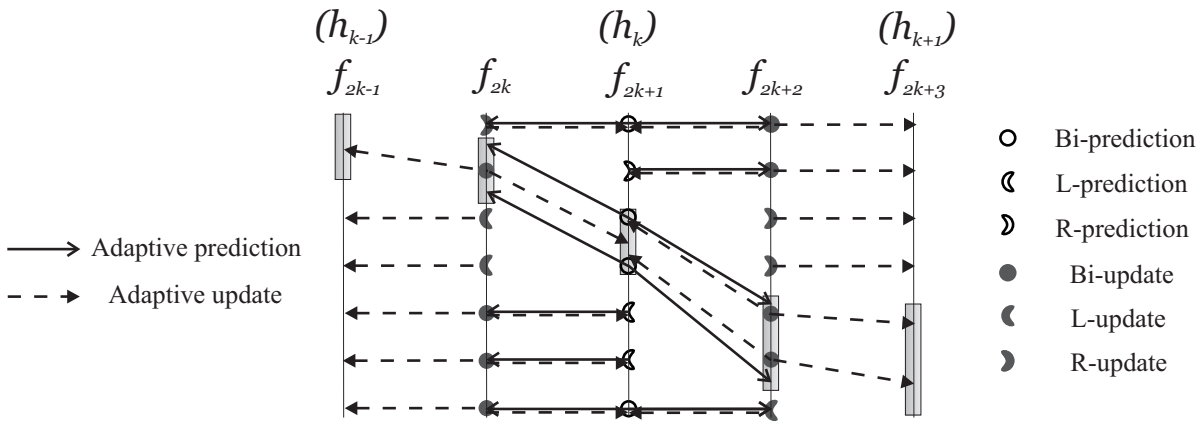


Figure 6·5: Adaptive lifting

prediction and update paths based on previously labeled occluded/exposed areas. This is illustrated in Fig 6·5, where only motion vectors that are actually used in the prediction step (solid vectors) and update step (dashed vectors) are included. In our simplified example, an object is expanding and moving down through the 5 represented frames. Clearly, frame f_{2k+1} is predicted from frames f_{2k} and f_{2k+2}, while frames f_{2k} and f_{2k+2} are updated from the corresponding high subbands. All prediction-step pixels in this example can be partitioned into one of the following categories: bi-predicted, left-predicted, and right-predicted. Similarly, all update-step pixels can be categorized as bi-updated, left-updated, and right-updated.

After both backward and forward motion fields within each frame pair have been estimated, the adaptive lifting for 5/3 DWT can be performed as follows:

$$\begin{aligned} h_k[\mathbf{x}] &= f_{2k+1}[\mathbf{x}] - \frac{w_P(\mathbf{x})}{2}\Big(c^E_{2k+1}(\mathbf{x})\widetilde{f}_{2k}[\mathcal{M}_{2k\to 2k+1}(\mathbf{x})] + c^O_{2k+1}(\mathbf{x})\widetilde{f}_{2k+2}[\mathcal{M}_{2k+2\to 2k+1}(\mathbf{x})]\Big) \\ l_k[\mathbf{x}] &= f_{2k}[\mathbf{x}] + \frac{w_U(\mathbf{x})}{4}\Big(c^E_{2k}(\mathbf{x})\widetilde{h}_{k-1}[\mathcal{M}_{2k-1\to 2k}(\mathbf{x})] + c^O_{2k}(\mathbf{x})\widetilde{h}_k[\mathcal{M}_{2k+1\to 2k}(\mathbf{x})]\Big), \end{aligned}$$

where w_P=1 for bi-prediction, 2 for uni-prediction, and 0 for intra-prediction. Similarly, w_U=1 for bi-update, 2 for uni-update, and 0 for intra update. In experiments presented later in this chapter, the "intra" prediction mode is handled by using a separate pixel-based spatial prediction. Typically, the number of such pixels is very small, justifying the use of

extra (pixel-based) signalization. The subscripts in c^E and c^O refer to the corresponding frames where the labels are defined.

This occlusion-adaptive lifting can significantly reduce the occurrence of "unconnected" pixels; in order for a pixel to be classified as "unconnected" in this new scheme, it has to be "unconnected" at both sides. This can happen in one of only two cases: (1) at the beginning or at the end of a GOP (where a reference from one side does not exist and a pixel is "unconnected" at the other side), or (2) in the case of a pixel that has reference neither in the previous nor in the following frame. The latter case assumes an existence of a pixel that is occluded immediately after it was exposed; this scenario is rare, but it can also occur due to complicated scene motion or failure in motion estimation.

From the coding efficiency point of view, it is crucial to note that the occlusion/exposure labels c^O and c^E need not to be separately coded and transmitted. Instead, they can be simply *derived* in the decoder from the received motion fields, by repeating the procedure from the encoder. This way, the information about occluded/uncovered areas is embedded into the motion information itself. The penalty is that both backward and forward motion fields need to be transmitted in order for the fully-adaptive lifting to work. Thus, joint coding of these two motion fields (Section 6.5.1) must be used to keep the motion bit-rate overhead low.

A similar idea to the one of adaptive lifting structure presented in this chapter was recently independently proposed by the group at the University of Aachen, Germany (Rusert et al., 2004). Within the optimization scheme for locally-adaptive MCTF using 5/3 lifting and HVSBM motion model, Rusert *et. al* proposed to use different frame and block modes for adaptive prediction, while keeping the conventional approach to the update step (Choi and Woods, 1999), which includes nearest-neighbor motion inversion and special treatment of "unconnected" pixels. The distinguishing feature of our approach is the iterative motion estimation of the backward and forward motion fields between each two frames, which indirectly models both occluded and exposed areas.

6.5.1 Joint coding of motion fields

As no gain comes for free, we are now required to code two seemingly independent motion fields for each frame pair. However, a detailed examination reveals that, due to a coupled and iterative estimation, these two motion fields are largely correlated. Indeed, they both closely follow a unique trajectory set for all well-connected pixels; for this reason, these two fields are very close to being inverses of each another in these areas. This leads to their efficient joint coding. For the best results, high-quality motion inversion is required; the spline-based motion inversion developed in Section 5.4.4 is used to that end. In our implementation, the first motion field (typically the one used for prediction step) is coded using a standard motion coding procedure similar to that of AVC/H.264. This consists of median prediction followed by context-adaptive binary arithmetic coding (CABAC). The spline-based inverse (Section 5.4.4) of this motion field is used as a predictor for coding of the second motion field, with the final prediction error arithmetically coded.

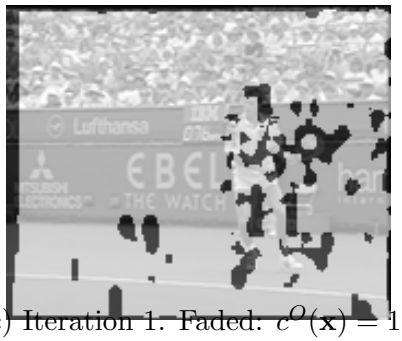
Additional increase in motion compression efficiency can be obtained if we observe that, in our approach, the motion vectors assigned to the "unconnected" pixels (i.e., in regions where $c^E(\mathbf{x}) = 0$ or $c^O(\mathbf{x}) = 0$) are not used for MC lifting (Fig. 6·5). Although all motion vectors still have to be transmitted, those corresponding to occluded/uncovered areas can be freely adjusted to minimize the total motion bit-rate, as long as there is no impact on the (derived) label fields c^E and c^O.

6.5.2 Possible extensions towards motion invertibility

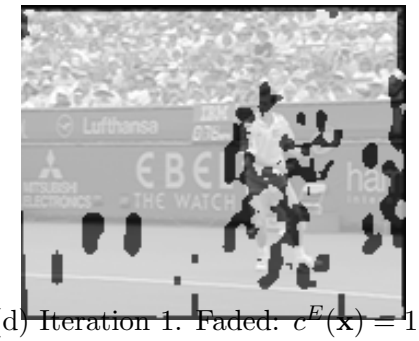
The occlusion-driven motion estimation algorithm presented above can be modified to include the cost of invertibility (Section 5.6). Our extensive experimentation, however, advised against its use and showed that benefits of the occlusion-adaptive lifting and invertibility error minimization mostly overlap. Only small compression gains are observed (on the order of 0.01 dB), not justifying the increased computational cost.

(a) *Stefan* QCIF, frame #7

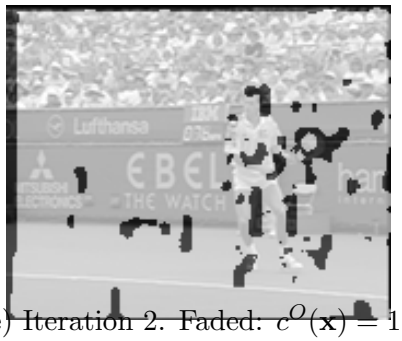
(b) *Stefan* QCIF, frame #8



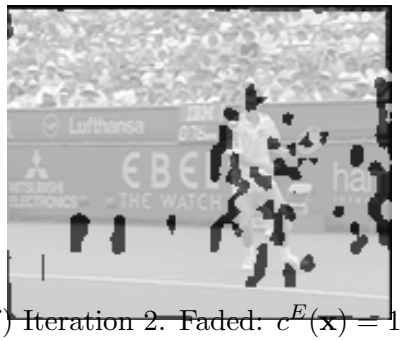
(c) Iteration 1. Faded: $c^O(\mathbf{x}) = 1$



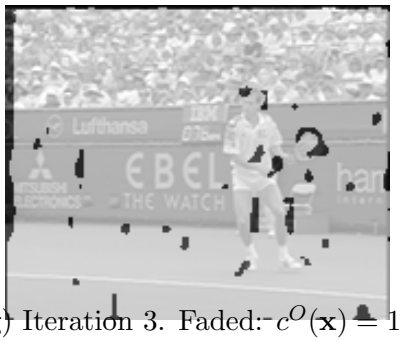
(d) Iteration 1. Faded: $c^E(\mathbf{x}) = 1$



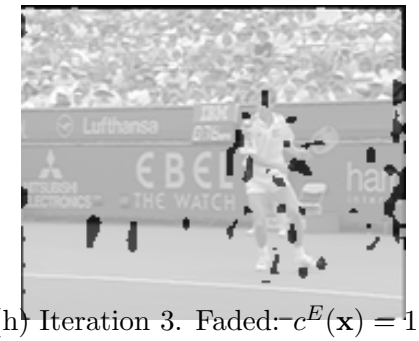
(e) Iteration 2. Faded: $c^O(\mathbf{x}) = 1$



(f) Iteration 2. Faded: $c^E(\mathbf{x}) = 1$



(g) Iteration 3. Faded: $c^O(\mathbf{x}) = 1$



(h) Iteration 3. Faded: $c^E(\mathbf{x}) = 1$

Figure 6·6: Top row: original frames #7 and #8 from *Stefan* QCIF sequence. Rows 2-4: Occlusion (left column) and exposure (right column) label fields $c^O(\mathbf{x})$ and $c^E(\mathbf{x})$ through different motion estimation passes.

6.6 Experimental results

In this section we present results using our proposed, occlusion-adaptive lifting scheme with occlusion/exposure detection. Motion is modeled using HVSBM with block sizes ranging from 16×16 down to 4×4. Typically, each iteration starts with large motion rate regularization parameter $\lambda = 64$; this facilitates estimation of smoother motion fields with less inter-block local variation. With increasing iteration number (one iteration consists of two independent motion estimations), the value of λ is decreased (typically by 10, so that the fifth iteration ends with $\lambda = 24$). Such an approach allows a quick detection of the most likely occlusion/exposure regions at the start (using coarser motion estimate), which is visible in the examples from Figs. 6·6 and 6·7. The estimated motion is then fine-tuned once the initial estimates of c^E and c^O are already available.

In Fig. 6·6, we show the progressive estimation of label fields c^E and c^O for the *Stefan* QCIF sequence. The top row displays two original frames (numbers 7 and 8). The second row shows the field of occluded labels (c^O) (overlayed over the reference frame) on the left. Faded (lighter) areas denote "connected" pixels, with c^O=1. Similarly, the field of exposed labels (c^E), overlayed over the current frame, is displayed on the right. As previously described, occluded/uncovered pixels in label fields c^E and c^O are not immediately set to zero; "soft-labeling" is used instead. For example, after this first iteration, "innovation" labels have a value of 0.96. Labels c^E and c^O are actively used for subsequent motion estimation runs, as described in the previous section. With increasing iteration number, the confidence in accurate detection of occluded and exposed areas increases, and c^E and c^O in non-faded areas in Fig. 6·6 are finally set to zero. The algorithm ends either when the maximum number of iterations is reached, or when the motion estimate between two iterations does not change by more than a preset threshold.

First, we can notice accurate detection of both occluded and exposed areas around the moving object. In the *Stefan* sequence, the player's legs move from left to right; the occluded area to the right from both of his legs is appropriately detected and labeled.

(a) *Flower Garden* CIF, frame #233

(b) *Flower Garden* CIF, frame #234

(c) Iteration 1. Faded: $c^O(\mathbf{x}) = 1$

(d) Iteration 1. Faded: $c^E(\mathbf{x}) = 1$

(e) Iteration 2. Faded: $c^O(\mathbf{x}) = 1$

(f) Iteration 2. Faded: $c^E(\mathbf{x}) = 1$

(g) Iteration 3. Faded: $c^O(\mathbf{x}) = 1$

(h) Iteration 3. Faded: $c^E(\mathbf{x}) = 1$

Figure 6·7: Top row: original frames #233 and #234 from *Flower Garden* CIF sequence. Rows 2-4: Occlusion (left column) and exposure (right column) label fields $c^O(\mathbf{x})$ and $c^E(\mathbf{x})$ through different motion estimation passes.

Table 6.1: R-D performance (luminance PSNR [dB]) - *Stefan* (QCIF, 30Hz)

Rate [kbps]	128	256	384	512	768
Adaptive 5/3 lift. MCTF	26.11	27.29	30.05	31.99	33.94
Non-adaptive 5/3 lift. MCTF	26.40	27.31	29.54	31.17	32.84
H.264/AVC	27.44	28.35	30.95	32.70	34.65

Also, as the head of the racquet moves up, pixels around it become occluded, which is also accurately detected. Second, due to the global camera pan from left to right, most of the pixels along the left frame boundary "disappear" in-between two frames. This is a special form of occlusion that is correctly detected. Our adaptive lifting algorithm uses this information to prevent filtering along these pixels in the update step. Similarly, we can visually confirm that the estimated field of "exposed" pixels c^E (right column) matches the exposed image areas relatively well.

Similar progressive results are shown in Fig. 6·7 for the CIF resolution image sequence *Flower Garden*. In the first row, original input frames 233 and 234 are shown. Between these two frames, the tree in the foreground moves quickly from right to left, while the entire frame also shifts to the left, due to the global camera motion. Again, our indirect estimation of occluded and exposed regions works well, correctly labeling pixels that are being occluded by the tree and exposed from behind the tree. Also, occluded/exposed pixels around a few vertical branches in the top left corner of the image are correctly labeled. Pixels disappearing from the scene are detected as occluded along the left border of the reference frame (left column); those entering the scene from the right are also appropriately labeled.

We use detected occlusion/exposure areas to adaptively drive motion-compensated temporal filtering, as described in the previous section. Typically, three levels of motion-compensated temporal decomposition are used to obtain temporal subbands. These are

Table 6.2: R-D performance (luminance PSNR [dB]) - *Mobile and Calendar* (CIF, 30Hz)

Rate [kbps]	384	512	768	1024	1536
Adaptive 5/3 lift. MCTF	28.32	29.60	31.50	33.12	35.43
Non-adaptive 5/3 lift. MCTF	28.30	28.74	30.87	32.11	34.35
H.264/AVC	28.56	29.22	31.40	32.87	35.70

Table 6.3: R-D performance (luminance PSNR [dB]) - *Flower Garden* (CIF, 30Hz)

Rate [kbps]	384	512	768	1024	1536
Adaptive 5/3 lift. MCTF	26.44	27.54	30.31	32.07	34.40
Non-adaptive 5/3 lift. MCTF	26.56	27.02	29.75	31.23	33.44
H.264/AVC	27.14	28.03	30.87	32.66	35.08

then decomposed over 5 resolution levels using biorthogonal CDF 9/7 wavelets (Taubman and Marcellin, 2002). The resulting spatio-temporal coefficients are encoded using the 3D-ESCOT from Microsoft Research Asia coder (Xu et al., 2001; Xiong et al., 2005c). Although the number of motion fields to be coded doubles, a relatively small increase in motion bitrate is observed (up to only 17%), due to efficient joint coding of motion fields. This overhead is small enough to justify the application of the algorithm at higher bit-rates, where motion occupies a smaller relative portion of the total bit budget.

The rate-distortion results in Tables 6.1 - 6.4 compare the occlusion-adaptive method with the non-adaptive implementation of our scalable coder, and also with a non-scalable AVC/H.264 codec. The AVC/H.264 implementation used for comparison was the JM 9.2 reference software (AVC/H.264, 2004) (main profile, 5 reference frames, GOP size 15, IBBP frame structure, rate control enabled).

We observe that the occlusion-adaptive lifting approach constantly outperforms the

Table 6.4: R-D performance (luminance PSNR [dB]) - *Football* (CIF 30Hz)

Rate [kbps]	384	512	768	1024	1536
Adaptive 5/3 lift. MCTF	28.56	30.34	31.94	33.43	35.83
Non-adaptive 5/3 lift. MCTF	28.89	30.43	31.76	32.96	35.11
H.264/AVC	30.01	31.19	33.10	34.68	36.92

non-adaptive coder, with typical gains ranging between 0.5 dB and 1.2 dB, depending on the sequence. It is clear that this gain is smaller at lower bitrates (due to a higher motion overhead of occlusion-aware coder) and grows when total decoding bit-rate increases. At the same time, our occlusion-adaptive lifting coder shows performance comparable (within 1 dB) to non-scalable H.264, with the added scalability feature.

6.7 Conclusions

In this chapter, we analyzed the role of motion in motion-compensated temporal lifting with the special emphasis on the the prediction lifting step. We proposed a new, fully-adaptive occlusion-driven temporal lifting scheme, based on 5/3 DWT and indirect detection of occluded/exposed image areas. Motion fields included into prediction and update steps are iteratively estimated and jointly coded. Our approach leads to increase in overall coding performance of up to 1.3 dB when compared to a non-adaptive coder.

Chapter 7

Advanced spatial motion modeling

7.1 Introduction

While the design of efficient spatio-temporal transforms dominated research in the arena of wavelet video coding over the last few years, significantly less attention has been devoted to advancing spatial motion modeling in the same context [1]. The dominance of block models in the hybrid context can be credited to its good match to a block decorrelating transform (in addition to relatively low complexity). Yet, there is no obvious reason why block matching should be favored in the wavelet video coding context. To the contrary, motion models that can better support the global nature of the DWT might be more suitable for this task. In addition, the motion invertibility discussion (Chapters 4 and 5) suggests that wavelet video coding benefits from using invertible motion; block-based motion is inherently non-invertible and therefore requires special handling, as seen in Chapter 5.

Regarding spatial modeling of motion, the rigid translational model of block-matching leaves plenty of room for improved spatial modeling. This is especially true when more complex motion is present in the scene (e.g., zoom in/out, rotation).

Advanced spatial motion models are also better suited for motion modeling at low resolutions (used in spatially-scalable coding, Chapter 8). It is a well-demonstrated fact that block matching performs quite well at higher spatial resolutions; the assumption of locally-homogeneous motion of neighboring pixels, which are usually part of the same object, is fairly well founded. However, at reduced spatial resolutions, neighboring pixels originate from physical points further apart and are less likely to have identical (or similar)

[1] It is most likely that the increased computational burden related to advanced motion models prevented them from being considered for standardization.

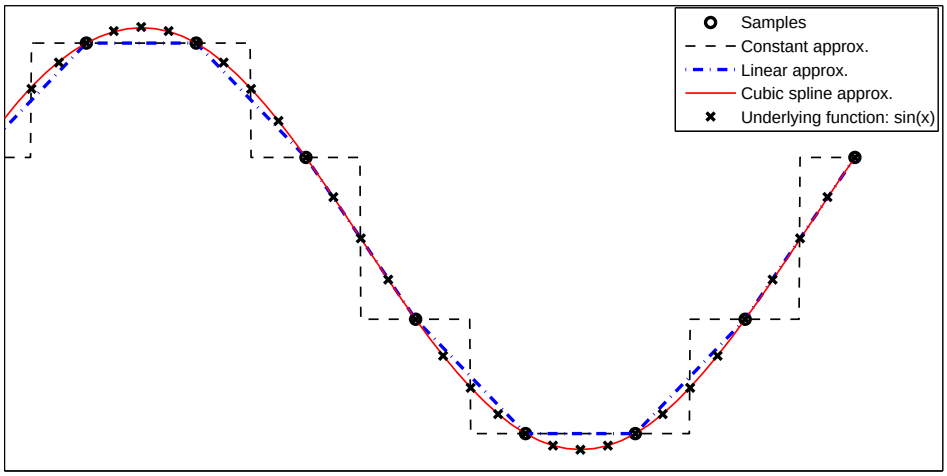


Figure 7·1: Comparison of various interpolators of a sine function (input samples are $\pi/4$ apart). Benefits of higher order models are obvious, especially at low sampling rates (low-resolution data).

underlying physical motion. Clearly, a simple piecewise-constant motion model is not likely to accurately capture the variations in a motion field at coarser spatial scales. In a simple 1D illustration of this phenomenon (Fig. 7·1), three different interpolators of a sine function from relatively sparse samples ($\pi/4$ apart) are compared. The superior performance of higher-order models over a piecewise-constant model is obvious.

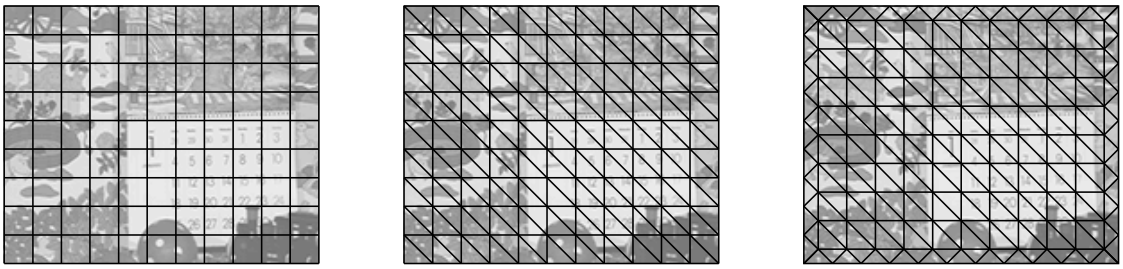
In this chapter, we focus on two advanced spatial motion models. The first model is based on deformable triangular mesh and implements spatially affine motion model. This model was successfully introduced to wavelet video coding by Secker and Taubman (Secker and Taubman, 2001) and its basics were presented in Chapter 2. We start with briefly revisiting mesh-based motion estimation in Section 7.2 and then propose two modifications to the standard approach. We first aim to improve handling of frame boundaries by changing the standard triangular mesh topology (Section 7.2.1). We then propose an improvement to classic motion estimation of Nakaya and Harashima (Nakaya and Harashima, 1994) in Section 7.2.2, which modifies the raster-scan order in which the motion of the mesh nodes is estimated. Combined, these two modifications lead to up to 0.5dB improvement over the standard mesh-based approach.

In the second part of this chapter, we introduce the general framework for hierarchical motion modeling based on splines (Section 7.4). Depending on the choice of basis and control-node topology, both block-based and mesh-based models can be analyzed in this context as zeroth and first-order splines, respectively. To maximize prediction performance, we propose to use a third-order (cubic) splines for motion modeling, as they have been shown to be a good tradeoff between complexity and approximation performance (Unser, 1999). The hierarchical spline framework is especially well suitable to support spatial scalability performance of wavelet video coders. In Section 7.5 we demonstrate the excellent prediction performance of spatially advance spline based motion models especially at low resolutions. Section 7.6 concludes the chapter.

7.2 Mesh-based motion estimation for wavelet video coding

Mesh-based motion model and motion estimation were introduced in Chapter 2. We briefly recall that mesh-based model is capable of capturing spatially affine motion. That means it can account for rotation, zoom, and shear, in addition to pure translation of block-matching. The estimation algorithm of hexagonal-matching (Nakaya and Harashima, 1994) is iterative in nature; in every pass, the motion vector at each node point is estimated by local minimization of the prediction error over six triangular patches (Fig 2·6). This approach is necessarily related to higher computational cost; nevertheless, following the significant increase in available computational power in recent years, advanced spatial motion models (like the mesh-based models) are, deservingly, receiving more attention.

Typically, mesh-based motion estimation algorithms (Nakaya and Harashima, 1994) are constrained in such a way that connectivity of the mesh is preserved. The mesh motion field then provides continuous "one-to-one" mapping between frames, which naturally solves "non-connectivity" problems discussed in previous chapters. On the other hand, it is well-known that motion estimation between two images fails whenever parts of the scene get occluded or newly exposed due to, for example, object motion. This happens in majority of real-life video sequences; therefore, most of the time some parts of images



(a) (b) (c)

Figure 7·2: Node-point topologies: a) fixed-size block-matching, b) standard triangular mesh, c) proposed modified triangular mesh.

indeed have undefined (underlying) motion. We saw how this problem can be addressed using discontinuous block-matching for innovation-driven temporal filtering in Section 6.5. In the connected mesh-based model, however, there is no similar way to handle the occluded/exposed areas.

Although the problem of occlusions and uncovering can be partially addressed using content-adaptive meshes (where mesh topology adapts to image content), this can significantly increase the already high computational cost related to mesh-based motion estimation. In addition, it can introduce rate-overhead problems (as topology needs to be communicated to the decoder). For this reason, we propose to improve the coding performance of a regular mesh-based model in the presence of innovations for the particular case of occluded and exposed areas occurring at image boundaries.

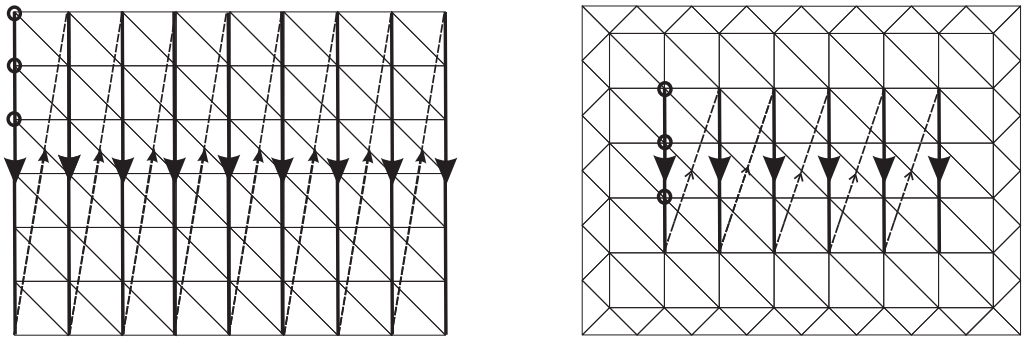
7.2.1 Enhanced boundary handling

Whenever a camera moves and/or objects leave or enter the field of view, frame features appear or disappear. Significant discrepancies between the current and reference frames along the frame boundaries may lead to a reduced compression gain. In order to address this, an improved mesh-based motion compensation is needed. Important practical constraints on this new motion compensation are regular mesh topology (that is well suited to compression applications) and acceptable computational complexity of the modified algorithm.

The standard approach to mesh topology has been to use triangular patches, where, through a suitable model, displacements of three neighboring nodes define displacements anywhere within a triangle. A triangular mesh can be built from the common square-block partitioning thus preserving block positions in the reference frame; mesh nodes can be set at the corners of all square blocks and each block divided in half along its diagonal (Fig. 7·2(b)). This configuration introduces a slight increase in the number of motion vectors needed to represent the complete motion field as compared to a block-based representation. If original video frames consist of $M \times N$ blocks, then MN motion vectors are needed for motion representation in standard block-based approach while $(M+1)(N+1)$ vectors are necessary in a triangular mesh motion representation. For the typical block size of 16×16 pixels and standard video resolutions, the increase in uncompressed motion information is 5% (ITU.R-601 - 720×480 resolution), 11% (SIF resolution) and 21% (QCIF resolution).

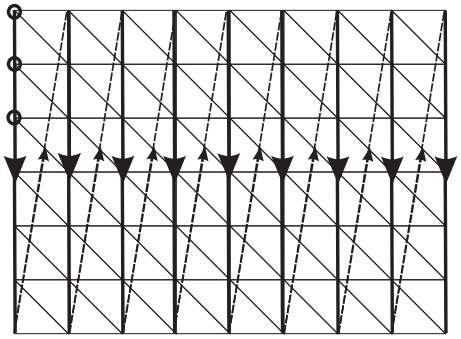
Our proposed triangular mesh topology shifts mesh node-points by half of the inter-node distance toward inside of the frame while constructing a double-density mesh at the frame boundary (Fig. 7·2(c)). The motivation for this new topology is clear - smaller patches can allow for more accurate motion modeling at frame boundaries, particularly in the presence of global motion. Moreover, the proposed modification in mesh topology prevents the motion outliers (that are likely to occur at boundaries of a typical video sequence) to affect larger patches that are now removed from the frame boundary.

With the modified mesh topology, the number of nodes needed for complete mesh representation is now $(M+2) \times (N+2) - 4$, which represents a slight increase in motion information to transmit as compared to the block-based and standard triangular mesh configurations. For example, at SIF resolution the increase in the total number of nodes is approximately 22% as compared to block-based motion and 10% in comparison with standard triangular mesh configuration. However, smoother motion fields with fewer outliers should compress better, thus reducing the negative impact of the increased number of nodes. For typical video bit-rates this amounts to less than 5% increase in the total

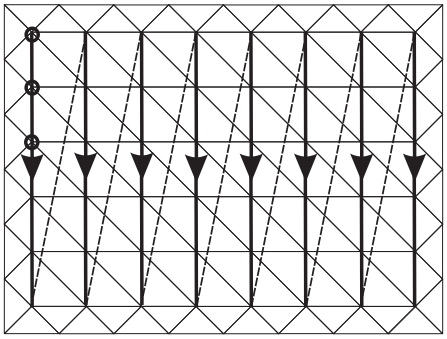


(a) Regular mesh - Iteration #1

(b) Modified mesh - Iteration #1



(c) Regular mesh - Iteration #2



(d) Modified mesh - Iteration #2

Figure 7·3: Difference in motion estimation order between standard scheme (left column) and proposed scheme (right column). The first three nodes in each iteration are denoted with circles.

bit-rate compared to block-based motion.

7.2.2 Enhanced mesh-based motion estimation

Unreliable motion estimation at frame boundaries is also a reason to modify the existing sequential search-based node-point motion estimation algorithms. Since in the iterative hexagonal refinement algorithm (Nakaya and Harashima, 1994) displacements of neighboring nodes affect each other, erroneous vectors at image boundary may adversely affect neighboring nodes that are located away from the boundary. Motion of mesh nodes is typically estimated in the raster scan order; this means that potentially significant number of erroneous motion vectors along the frame border can have an early impact on the accuracy of motion estimation away from the border.

In order to minimize this effect, we propose to start the iterative process of hierarchical hexagonal refinement on triangular patches closer to the image center (Fig. 7·3). More nodes that are closer to the frame boundary are included in subsequent iterations. This way, more reliable node-point displacements are calculated earlier and potential propagation of erroneous estimates from the frame boundary is avoided. This process helps in dealing with occlusion-prone (at the boundary) cases like global camera pan or zoom-in. Also note that constraints imposed on motion vectors by standard block-matching algorithms before MPEG-4 (e.g., restriction on motion vectors pointing to the interior of the frame) may often result in motion vector outliers. This inhomogeneity of motion field in the mesh case can then have a significant negative impact on the both the motion overhead and the prediction performance.

7.3 Experimental results - mesh-based motion

In this section we analyze the performance of our wavelet video coder when mesh-based motion is used. All experiments are performed using our own subband coder (Božinović et al., 2004). We use three levels of the motion-compensated 5/3 lifting transform, with motion estimated at 1/4 pixel accuracy. Temporal subbands are spatially transformed and codded using our JPEG2000-based coder. We compare the proposed approach using modified mesh with standard mesh and block-matching (inversion is performed using nearest-neighbor approach, Section 5.4.3). To assure a fair comparison, we use fixed-size (16×16) block matching, as only fixed-size meshes are used.

We begin with the analysis of the motion bitrate overhead. We estimate motion for the first 150 frames of *Mobile and Calendar* and *Football* sequences (CIF resolution, 30Hz) and encode it losslessly using JPEG-LS directly on arrays of horizontal and vertical motion components. All required motion fields from three temporal decomposition levels are encoded independently, i.e., no temporal cross-resolution prediction was used.[2] Average

[2]Although this approach is sub-optimal from point of view of mesh-based motion coding (Secker and Taubman, 2003), it is a good indicator of relative motion overhead introduced by different motion models.

Table 7.1: Motion bitrate (kbps) for *Mobile and Calendar* and *Football* sequences encoded at CIF resolution (30Hz). Relative increase in number of nodes and motion bitrate are also shown.

Sequence	*Mobile and Calendar*			*Football*		
Motion model	Rate(kbps)	% nodes	% bitrate	Rate(kbps)	% nodes	% bitrate
Block matching	134	100%	100%	179	100%	100%
Mesh	142	+10%	+6%	187	+10%	+4%
Modified mesh	151	+20%	+13%	196	+20%	+9%

motion bit-rates for *Mobile and Calendar* sequence are 134 kbps, 142 kbps and 151 kbps for block-matching, standard mesh and modified mesh methods, respectively. This presents 6% and 13% increase in motion bit-rate of mesh-based motion compared to standard fixed-size block matching. The motion overhead increases less than expected from the growing number of nodes, which is due to the smoothing effect that regularization has on the estimated node motion. In the case of *Football*, this effect is even more visible because of very dynamic motion in the sequence. Average motion bit-rates increase from 179 kbps for block-matching, to 187 kbps in the case of standard mesh (4% increase), and to 196 kbps (9% increase) in the case of modified mesh.

Figures 7·4 and 7·5 show rate-distortion performance of our coder when three different motion configurations are used. We show average luminance PSNR for first 128 frames of two CIF sequences: *Flower Garden*, *Mobile and Calendar*. Both sequences are characterized by strong global (camera) motion, with a significant amount of boundary occlusion effect. We see that the modified mesh outperforms regular mesh by 0.1 - 0.4dB on average, while the gain over the fixed-size block motion model is more significant (up to 1.3dB at high rates). Note that these gains are possible despite increased motion bitrate (Table 7.1), which is due to better motion modeling and/or boundary handling. At very low bitrates, the difference between the block and mesh models is less visible, as losslessly coded motion occupies a significant portion of the overall bit budget. When the decoding bitrate in-

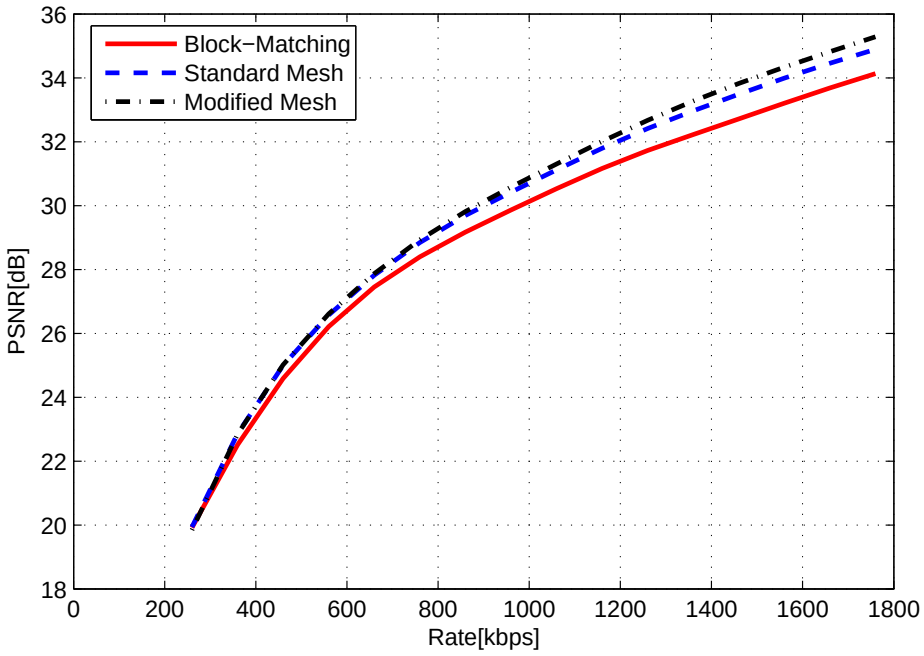


Figure 7·4: Rate-distortion performance comparing block-matching, mesh, and modified mesh; *Flower Garden* sequence at CIF resolution, 30Hz.

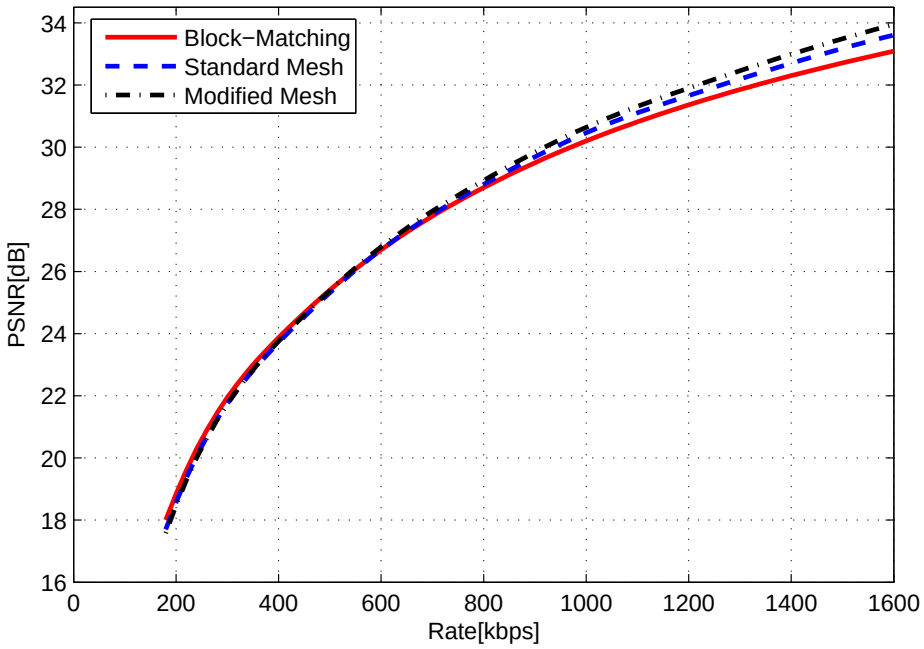


Figure 7·5: Rate-distortion performance comparing block-matching, mesh, and modified mesh; *Mobile and Calendar* sequence at CIF resolution, 30Hz.

creases, the significance of motion overhead is reduced, and the better motion modeling of mesh leads to more substantial gains. We observed similar results, although with somewhat smaller coding gain of modified mesh configuration,[3] in sequences than exhibit small or no camera motion. This reinforces our claim that advanced spatial modeling and invertibility support are indeed beneficial to compression performance of wavelet video coders.

7.4 Spline-based motion modeling

In previous section, we have seen how mesh-based model improves spatial motion modeling. With triangular meshes, the order of the motion model increases from zero (block-wise constant) to one (patch-wise planar, in both x- and y-coordinate). The improved coding gain of the mesh model naturally leads to the next question: can we improve the prediction and compression performance by using even higher order of spatial motion models? In this section, we present the general framework for motion modeling based on splines. Depending on the spline order and spline control node topology, both block-based and mesh-based motion models can be interpreted within this framework. However, our main interest lies in investigating potential benefits of even higher-order spatial motion models. We in particular propose to use third-order (cubic) parametric splines, as they have been shown to offer a good compromise between accurate spatial modeling and manageable computational complexity. In Fig. 7·6, we show B-splines (Unser, 1999) of degrees 0 to 3.

The spline motion model attempts to keep the best properties of block- and mesh-based models, while eliminating their drawbacks. Block-based models can capture sharp transitions in the motion field, at the expense of oversimplified uniform translational modeling of unit patches. Triangular meshes, on the other hand, are capable of supporting higher-order parametric affine (first-order or planar) motion model. This well-understood model is solidly founded in the physical camera movement and its effect on the perceived scene motion. When applied locally, the affine model can also support more complex motion of the scene objects (e.g., rotation, object approaching, or receding). However, a connected

[3] This is expected, as most of the gains of modified mesh come from improved boundary handing.

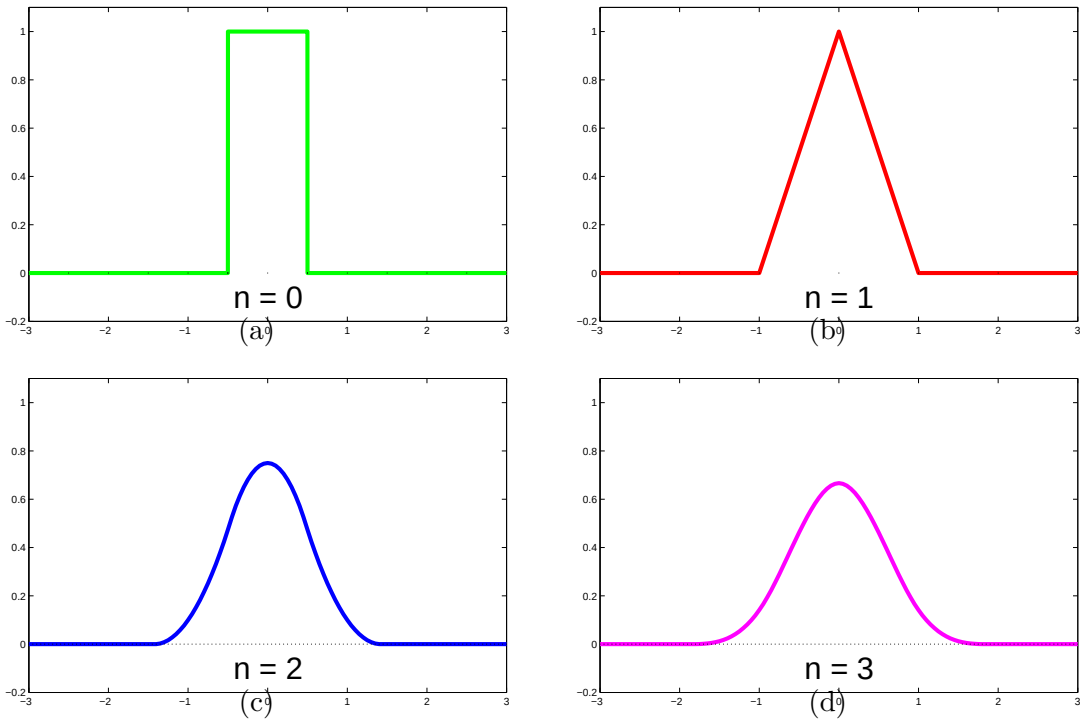


Figure 7·6: The centered B-splines of degree 0 to 3.

mesh representation can "over-smooth" the scene motion, and bilinear interpolation on triangles is not always capable of modeling sharp transitions in the motion field. It is here that we find a major motivation for our research on higher-order spatial models that can preserve a continuous motion representation and improve transition handling at the same time.

Similar to other motion models used for video compression, the spline-based model falls in between local (pixel-based) and global (frame-based) models. The spline-based motion model can be viewed as a local parametric model, since the motion within each spline patch is defined by a finite number of control vertices. Parametric splines naturally provide continuous motion representation. In Sections 4.4 and 4.5, we have seen that only in the context of continuous mapping with well-defined inverse we can interpret the temporal lifting-implemented MC-DWT as operating along motion trajectories. When the node topology of the model is preserved, continuous models exhibit a desirable invertibility property. In addition, it has been shown that the composition of continuous motion fields

at different temporal scales can facilitate compact motion representation and reduce motion overhead (Secker and Taubman, 2003).

Our practical approach to spline-based motion flow estimation revisits, to a certain degree, the work of Szeliski and Shum (Szeliski and Shum, 1996), which was carried out in the context of image registration. In practice, most of their work was implemented for the case of a bilinear spline basis; we extend this model to more advanced cubic splines. We replace the node topology of hierarchical spline basis with the one more suitable to spatially scalable wavelet-based video compression applications; the estimated spline-based motion fields are then used for video compression in this context. In another notable work on the advanced multi-scale motion representation, Moulin et al. (Moulin et al., 1997) have proposed an integrated system with motion vectors represented using hierarchical basis functions in the form of linear splines.

In our discussion, we limit the temporal motion modeling to piece-wise linear trajectories, i.e., a single motion mapping defines motion between two frames only. In this case, the motion field anchored in the current frame f_k is independent from fields defined at all other times. Under this assumption, we can study the two-frame problem without any loss of generality. If the current and reference intensity images (respectively, $f_k(x, y)$ and $f_m(x, y)$) are known,[4] the standard motion prediction error measure, i.e., *sum of squared differences* (SSD) can be written as:

$$J(\{u_i, v_i\}) = \sum_i \Big(f_k(x_i, y_i) - f_m(x_i - u_i,\ y_i - v_i) \Big)^2. \tag{7.1}$$

The standard approach to motion estimation is then defined as a minimization problem over displacement fields $u_i \triangleq u(x_i, y_i)$ and $v_i \triangleq v(x_i, y_i)$ representing x and y components of a motion mapping $\mathcal{M}_{l \to k}$. We model these two motion fields as two-dimensional spline functions controlled by a smaller number of spline coefficients $\hat{u}_j$ and $\hat{v}_j$, which lie on a coarser spline control grid. This is illustrated in Fig.7·7; we use $\mathbf{d} = [u, v]^T$ to denote a

[4] We sometimes also use the vector form to denote spatial coordinates as $\mathbf{x} = [x, y]^T$.

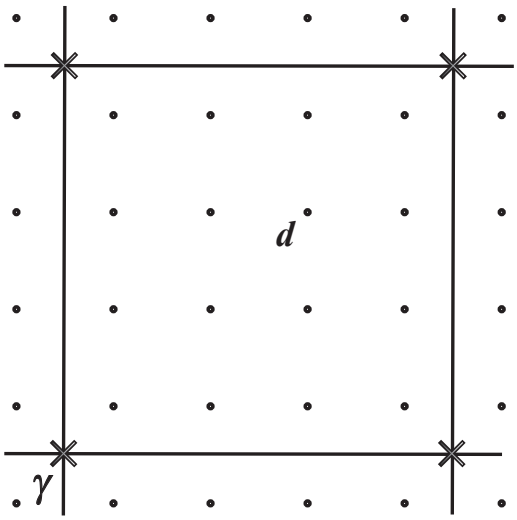


Figure 7·7: Example of the spline control grid; a small number of spline coefficients γ defined at control points j is used to represent values of motion vectors d at pixel positions i.

motion vector and $\gamma = [\hat{u}, \hat{v}]^T$ to denote spline coefficients. Index j is used to denote spline control vertices, while i still refers to pixel indices. The horizontal displacement at pixel i can then be written as:

$$u(x_i, y_i) = \sum_j \hat{u}_j B_j(x_i, y_i) \quad \text{or} \quad u_i = \sum_j \hat{u}_j w_{ij}, \tag{7.2}$$

and similarly for the vertical component of the motion field:

$$v(x_i, y_i) = \sum_j \hat{v}_j B_j(x_i, y_i) \quad \text{or } v_i = \sum_j \hat{v}_j w_{ij}, \tag{7.3}$$

where $B_j(x, y)$ are spline basis function, and are usually non-zero only over a small interval (finite support). We call the $w_{ij} = B_j(x_i, y_i)$ *weights* to emphasize the fact that (u_i, v_i) are known linear combinations of $(\hat{u}_j, \hat{v}_j)$.

In order to recover the local spline-based motion parameters, we need to minimize the cost function (7.1) with respect to the $(\hat{u}_j, \hat{v}_j)$. We do this using a variant of the Levenberg-Marquardt iterative non-linear minimization technique (Press et al., 1992). We here present an overview of the estimation process; detailed algorithm steps can be found

elsewhere (Szeliski and Shum, 1996).

We begin with a single resolution case (non-hierarchical motion representation). In other words, in this scenario, all spline control nodes are located on a single uniform lattice (e.g., nodes denoted with $\times$ in Fig. 7·7). In this case, the gradient of J in (7.1) is first computed with respect to each of the parameters ($\hat{u}_j$ and $\hat{v}_j$):

$$g_j^u = \frac{\partial J}{\partial \hat{u}_j} = 2 \sum_i e_i G_i^x w_{ij},$$
$$g_j^v = \frac{\partial J}{\partial \hat{v}_j} = 2 \sum_i e_i G_i^y w_{ij},$$

where

$$(G_i^x, G_i^y) = \nabla_{\mathbf{x}} \, f_m(\mathbf{x})|_{\mathbf{x}=\mathbf{x_i}-\mathbf{d_i}}$$

is the intensity gradient of the reference frame f_m evaluated at the displaced position of pixel i ($\mathbf{d_i} = [u_i, v_i]^T$), and

$$e_i = f_k(x_i, y_i) - f_m(x_i - u_i, y_i - v_i)$$

is the intensity error at pixel i. The w_{ij} are the sampled values of the spline basis function (7.2).

Algorithmically, we compute the above gradients by first forming the displacement vector (u_i, v_i) at each pixel of the current frame f_k using (7.2) and (7.3),[5] then computing the resampled intensity and gradient values of f_m at $(x_i', y_i') = (x_i - u_i, y_i - v_i)$, computing intensity error at pixel i (e_i), and finally incrementing the g_j^u and g_j^v values of all control vertices affecting that pixel.

For the Levenberg-Marquardt algorithm, we then calculate the approximate Hessian matrix $\mathbf{A}$ (Press et al., 1992) and compute an increment $\Delta\mathbf{u}$ to the current displacement

[5]Before the first iteration, displacement vectors have to be initialized either with zeros or with the result of another (typically faster) initial motion estimation technique.

estimate $\mathbf{u}$ which satisfies:

$$(\mathbf{A} + \lambda\mathbf{I})\Delta\mathbf{u} = -\mathbf{g},$$

where $\mathbf{u}$ is the vector of concatenated displacement estimates $\{\hat{u}_j, \hat{v}_j\}$, $\mathbf{g}$ is the vector of concatenated energy gradients $\{g_j^u, g_j^v\}$, and λ is a stabilization factor which varies over time (Szeliski and Coughlan, 1994a). To solve this large, sparse system of linear equations, we use the well-known preconditioned gradient descent method (preconditioning adjusts the descent direction and accelerates convergence). There are several methods for practical speed-up of the described algorithm. One is to initialize the motion field with the estimate obtained using fast block-matching. We can also run the algorithm in a coarse-to-fine fashion, which helps handling larger displacements. These two approaches can be combined to obtain a maximally computationally efficient solution.

7.4.1 Extension to hierarchical splines

The above algorithm describes the process of calculating spline-based motion parameters at single resolution level. In order to improve efficiency and make the motion model more suitable to spatially-scalable applications, a hierarchical representation is required. Hierarchical basis splines are based on pyramidal data representation. Standard approaches typically rely on Gaussian image pyramids; however, in the context of wavelet video coding, it is beneficial to create the pyramid using the same spatial decomposition wavelets that are later used for coding anyhow. In this scenario, every input frame first undergoes an appropriate spatial wavelet filtering before the motion estimation process is started (for more details on this coding scheme please refer to Chapter 8).

Hierarchical basis splines can be introduced in two formats; one where the total number of nodes in the pyramid is equal to the original number of nodes at the finest level (Szeliski and Coughlan, 1994b), and the other where spline nodes at different resolution levels "overlap" spatially. This is illustrated in Fig. 7·8.

The first approach, with node distribution depicted in Fig. 7·8(a), assumes that all

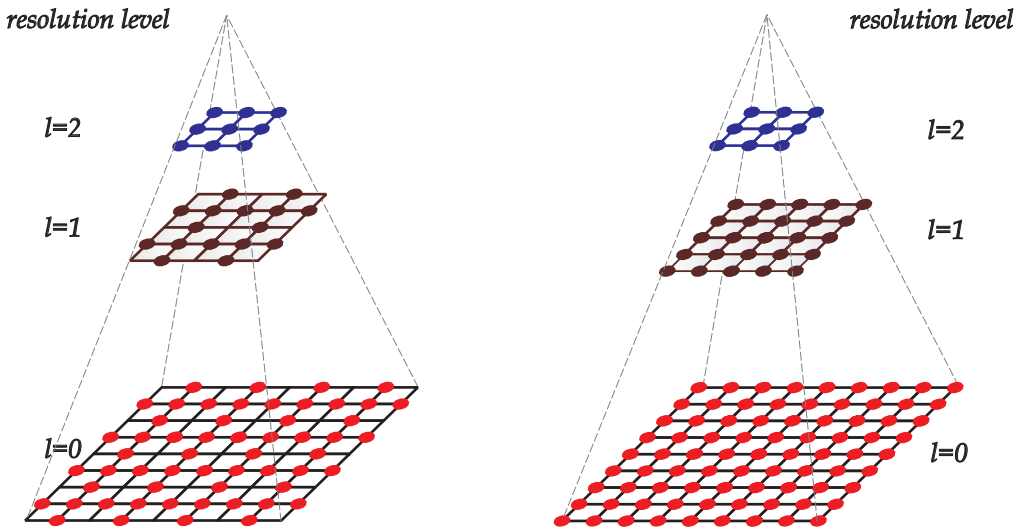


Figure 7·8: Hierarchical basis representation for $L = 3$ pyramid levels. The circles indicate the nodes in hierarchical basis. a) The total number of nodes is equal to the number of variables at the finest ($l = 0$) pyramid level (notice the "missing" nodes at all but coarsest resolution); b) All nodes are populated at each spatial resolution level.

spline parameters across different scales are (iteratively) estimated *at the same time*. While this compact and elegant solution leads to optimal high-resolution motion flow representation, the optimality at low and intermediate resolutions is not guaranteed. Therefore, this approach is not practical from the standpoint of spatially-scalable motion representation and cannot be used for spatially scalable coding.[6] For now, we note that for the best spatially scalable compression performance, motion should ideally be optimized for each supported resolution level.

We, therefore, decided to use node configuration presented in Fig. 7·8(b). In this case, rate-distortion optimized motion representation can indeed be obtained at each spatial level; no subsequent re-estimation of coarse scale parameters is necessary when more pixels from higher-resolution pyramid levels are included in the estimation. This approach is better matched to the spatially-scalable coding structure introduced in Chapter 8.

[6] As discussed in Section 8.4, the MCTFs of spatially-scalable coding scheme are applied independently at each supported spatial resolution.

In addition, our approach supports seamless integration of different spline basis function at various model resolutions.

Using the node topology from Fig. 7·8(b), our motion estimation process could be described as follows. We first estimate the motion at the coarsest (lowest resolution) level of the spatial pyramid, as described in Section 7.4. Once this coarse-resolution motion is known, it is projected (interpolated) onto the next finer resolution level. The minimization process is repeated (using the low-resolution estimate as a predictor) until the final nodal representation is obtained. For compactness, we chose to use a "residual" motion flow representation at each spatial scale; converting from the hierarchical basis representation to the usual nodal (fine-level) motion representation, can then be straightforwardly performed as:

$$u(x_i, y_i) \quad = \quad \sum_{l \in L} \sum_{j \in J_l} \hat{u}_j^l w_{ij}^l. \tag{7.4}$$

Here, J_l is a collection of spline control points at spatial level l, as shown in Fig 7·8(b). This representation obviously has a layered structure; it is relatively easy to show that, with respect to the cost of motion coding, this representation is equivalent to *spatially predictive* coding of motion fields from lower spatial resolutions.

We already mentioned that, with an appropriate node selection at each level of the pyramid, the hierarchical spline model is flexible enough to support different basis function at different hierarchical levels. In this chapter, we use cubic splines (Unser, 1999) at all pyramid levels; a new approach that combines spline-basis of different order will be introduced in Chapter 8. The hierarchical structure of motion estimation algorithm naturally improves the implementation speed; typically only three to four iterations of preconditioned Levenberg-Marquardt algorithm at each resolution are sufficient to obtain high-quality motion estimate. For additional speed-up, the initialization of spline coefficients at the coarsest level can be performed using some fast motion estimation algorithm (e.g. fast block matching).

Finally, at this time we discuss the issue of hierarchical variable-size splines, sometimes

referred to as *quadtree* splines. The variable-size block-matching (VSBM) approach, first introduced in MPEG-4 visual and extended in AVC/H.264, proved to be very popular and successful. Similar concepts were proposed and deployed - with more or less success - to motion models based on triangular meshes (where motion within the patch is obtained using bilinear interpolation within the triangles), and, in similar manner, to block-based bilinear model (Szeliski and Shum, 1996). As we previously mentioned, both the uniform translational model and the affine model can be interpreted in the more general context of spline basis functions, with basis order zero and one, respectively. It is exactly this extremely short support (of one and two inter-nodal distances, respectively) of spline basis functions in these two cases that allows for convenient application of quadtree approach.

To illustrate this, we take a look at the VSBM; the decision to split one macroblock into smaller blocks (effectively introducing new nodes to the spline topology) does not in any way affect the motion estimate in any of the neighboring macroblocks. A similar idea can be implemented, with appropriate node topology, using hierarchical triangular meshes (for details, see Section 9.2.2).

Unfortunately, longer support of advanced cubic spline basis functions (extending to four inter-nodal distances), prevents us from easily embracing the quadtree concept in this higher spatial order case. Any new quadtree node that is introduced on the denser spline-based grid will create undesirable "ripples" in neighboring patches that might not require to be divided on their own. This, in turn, would require re-estimation of the coarse scale spline parameters, which would then affect even more patches and generate a global response to the local change in motion field. For this reason we decided to use only fixed-size node topology for cubic spline basis at all resolution levels. A potential solution to this problem might be to use a high-order spline basis at the lowest spatial resolution (similar to fixed-size macroblocks in AVC/H.264), but then combine it with lower-order (e.g., linear, or zeroth-order) splines that could facilitate efficient quadtree implementation. The problem of hierarchical variable-size (quadtree) splines for the case of higher order basis functions remains an interesting and open topic for future research.

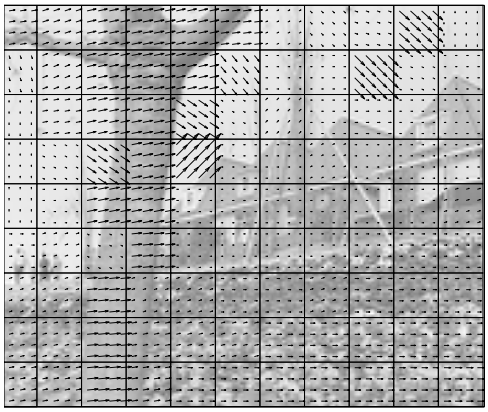
7.5 Experimental results - spline-based motion

In this section, we evaluate motion prediction performance of the introduced spline motion model that is based on cubic basis functions. We compare it with both fixed-size (16×16) block matching (used in MPEG-2), and hierarchical variable-size block matching (used in AVC/H.264). To speed-up the implementation we run our algorithm in coarse-to-fine hierarchical fashion, even when motion is being estimated at a single resolution only.

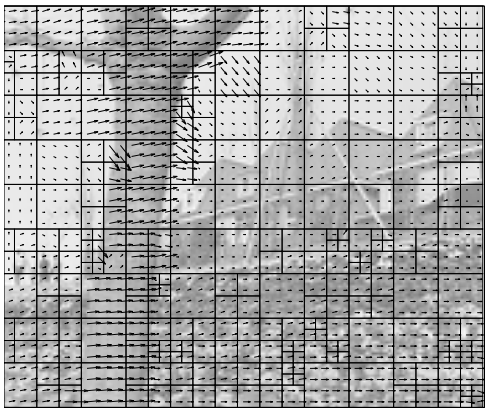
Three motion fields estimated between two frames of *Flower Garden* QCIF sequence (frames #234 and #235) using three different motion models are presented in Fig.7·9. We use these fields to illustrate the most important differences between the discussed models. At the top, the fixed-size block matching motion field (1/4 pixel precision) exhibits visible outliers (the top right corner), and results in the motion-compensated prediction luminance PSNR[7] of 27.48dB. In the middle, we show the motion field estimated using HVSBM (with the same 1/4 pixel precision). This model clearly renders the underlying scene motion more accurately, and it improves the prediction PSNR to 28.39dB. The hierarchical approach helps in removing most of the outliers, while variable size blocks improve prediction around object boundaries. The increased prediction performance is related to a somewhat higher cost of motion coding (because of the larger number of motion vectors and the motion quad-tree map that needs to be encoded). Finally, the spline based flow is shown in Fig.7·9(c). For a fair comparison with both block-based models, spline motion parameters are correspondingly quantized (one way to do this is to quantize the cardinal-form spline coefficients to 1/4 accuracy). We can see that despite the fixed (16×16) node topology of the spline model,[8] its prediction outperforms the HVSMB at PSNR 28.45dB. A hierarchical estimation prevents the occurrence of large outliers and creates a regularized motion field, while high third-order motion model can properly adapt to rapid changes in the motion

[7]Motion prediction PSNR represents a distortion between the current and predicted frame, and it is based purely on motion compensation performance, as no further prediction error is used. We use it purely to compare motion prediction performance; it cannot be directly used to assess coding performance nor it should be mixed with true reconstruction PSNR.

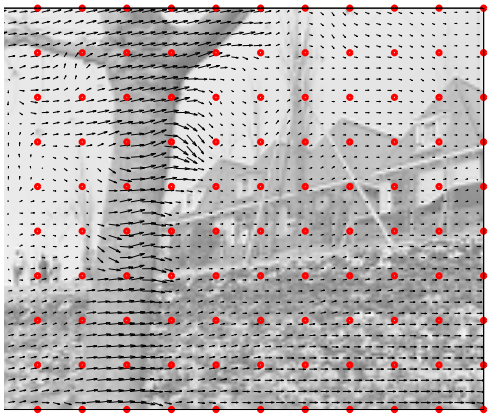
[8]The total number of spline motion parameters is in this case very similar to the number of motion vectors in the fixed-size block-matching case.



(a) FSBM, PSNR = 27.48dB



(b) HVSBM, PSNR = 28.39dB



(c) Spline motion field, PSNR 28.45dB

Figure 7·9: Comparison of three different motion models. *Flower Garden* sequence, frame #235. Motion-prediction luminance PSNR is computed between the current frame and predicted frame.

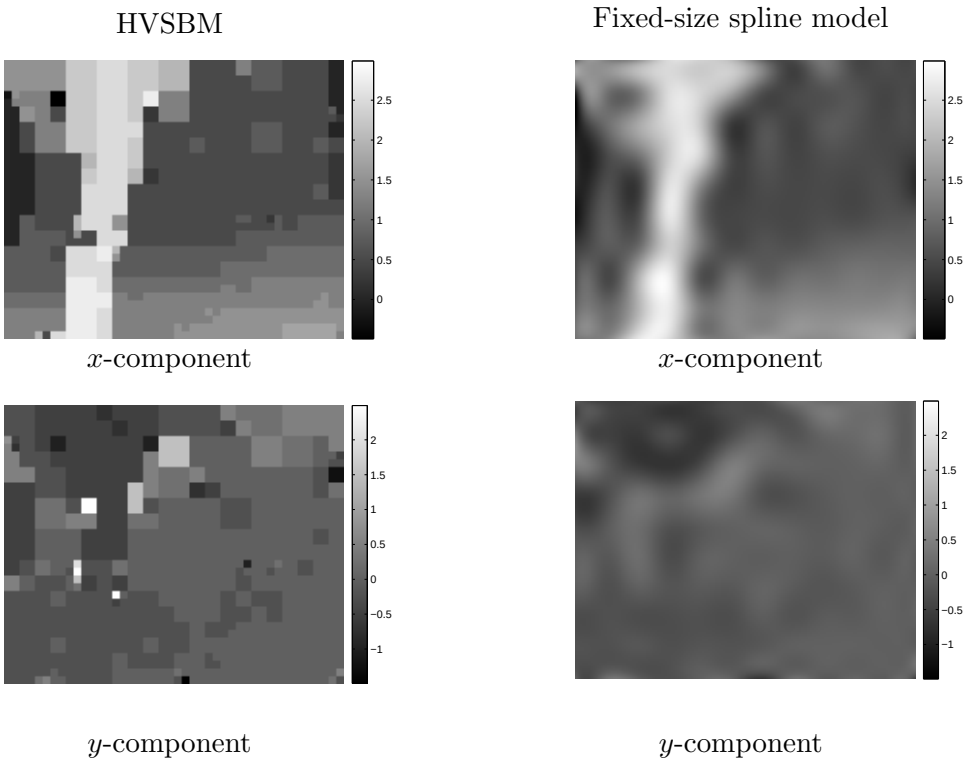


Figure 7·10: x- and y-components of motion fields from Fig. 7·9 as grayscale images; left column: HVSBM; right column: spline-based model.

field (especially visible around the foreground tree).

Fig. 7·9 displays motion vectors on the subsampled grid for visualization purposes. To emphasize the continuous nature of spline-based motion field, we also represent x- and y-components of HVSBM and spline-based motion fields as grayscale images in in Fig. 7·10.

Next, we take a look at motion-compensated prediction performance at three different spatial resolutions: QQCIF, QCIF, and CIF. It is important to note that, in parallel with the increased interest in high-resolution video coding and delivery (e.g., HDTV), the plethora of devices supporting only QCIF or sub-QCIF resolutions (e.g., most multimedia cell phones) emerged recently. They form a wide device and application base for low-resolution coding, providing a strong motivation for future research in the area of compact and efficient motion representation at these spatial scales.

In Tables 7.2, 7.3, and 7.4, a single-resolution cubic-spline model is compared to two block-matching models for a variety of sequences. In order to use 16×16 macro-blocks

Table 7.2: Motion-compensated prediction PSNR(dB) at QQCIF, 30Hz; no prediction error is encoded and used. Average over the first 50 frames.

	No motion	FSBM (16×16)	HVSBM ($\lambda = 34$)	SP16	SP8
Flower Garden	27.66	30.57	31.83	32.04	32.17
Stefan	25.62	29.16	29.79	30.40	30.61
Foreman	30.39	33.86	36.07	36.15	36.37
Mobile	28.31	28.83	29.88	30.34	30.62

Table 7.3: Motion-compensated prediction PSNR(dB) at QCIF, 30Hz; no prediction error is encoded and used. Average over the first 50 frames.

	No motion	FSBM (16×16)	HVSBM ($\lambda = 34$)	SP16	SP8
Flower Garden	21.29	29.09	30.14	30.04	30.23
Stefan	23.18	28.06	28.82	28.91	29.09
Foreman	29.01	34.98	36.23	36.11	36.21
Mobile	25.52	28.23	28.93	28.79	29.13

Table 7.4: Motion-compensated prediction PSNR(dB) at CIF, 30Hz; no prediction error is encoded and used. Average over the first 50 frames.

	No motion	FSBM (16×16)	HVSBM ($\lambda = 34$)	SP16	SP8
Flower Garden	18.00	28.34	30.89	30.25	30.67
Stefan	21.13	28.25	29.13	28.76	28.93
Foreman	27.15	34.15	35.44	34.65	35.19
Mobile	20.36	27.47	28.11	27.76	28.04

at QQCIF resolution, 4 pixels have been cropped from each side of the frame. Also, QQCIF resolution sequences are obtained as an LL spatial subband of QCIF resolution image, after a single level of the CDF 9/7 spatial DWT, typically used for spatial wavelet coding (Taubman and Marcellin, 2002). In each table, the first column shows PSNR of the temporal prediction error without motion compensation. In the next two columns, the prediction performance of fixed-size block matching (block size of 16) and HVSBM ($\lambda = 34$, block size 16×16 down to 4×4) is presented. Finally, the last two columns represent two spline-based motion models; SP16 and SP8 denote cubic spline models with inter-nodal distances of 16 and 8 pixels, respectively.

It is important to note that results presented in Tables 7.2 - 7.4 compare solely the motion-compensated prediction performance of different motion fields, without taking motion bit-rate into account. A full assessment of motion compensation performance is possible only in the context of a complete coding system, which would also take into consideration a true motion bit-rate. These results are presented in the next chapter; for now, in order to justify and motivate a further exploration of spline-based models, we limit our analysis to the number of motion parameters needed to represent a motion field.[9] In this case, motion-compensated prediction using fixed-size block matching (column two) can be effectively compared to that of spline-based SP16 motion field (column four). Both fields use a fixed number of motion vectors per motion field; for a frame size $M \times N$, FSBM produces $\frac{M}{16} \times \frac{N}{16}$ motion vectors, while the number of motion vectors in SP16 motion field is slightly larger at $(\frac{M}{16}+1)\times(\frac{N}{16}+1)$, due to a different node topology, illustrated in Fig. 7·12. This difference should be taken into account when motion prediction results are compared; we can, however, observe that consistent and substantial gains of SP16 over FSBM motion are large enough to justify this increase in the number of motion parameters.[10]

Comparing the motion prediction performance of HVSBM (column three) and SP8

[9]Under the assumption of the same motion accuracy and similar entropy coding in different coding scenarios, the number of motion parameters is closely related to the total motion bit-rate.

[10]For example, at CIF resolution, the number of motion vectors per motion field increases from 396 in FSBM to 437 in SP16 configuration, i.e., by only about 10%.

(column five) is substantially more difficult - the main reason for this is a variable number of motion vectors in HVSBM that depends on the regularization parameter λ. For example, a very large λ would result in the number of motion vectors being identical to the FSBM case (all blocks of size 16×16); on the other hand, $\lambda = 1$ would significantly increase this number and result predominantly in 4×4 blocks. In our experiments, for commonly used value of $\lambda = 34$, the number of motion vectors in HVSBM field was relatively close (most of the time within 15%) to that of fixed-size SP8 configuration. This moderate λ value results in roughly equal number of blocks larger and smaller than 8×8 - a typical motion field for $\lambda = 34$ is presented in Fig. 7·9(b). In conclusion, an informative comparison between block- and spline-based motion fields in terms of motion-compensated prediction is possible between FSBM and SP16 configurations, and HVSBM and SP8 configurations. A more detailed assessment of different motion models in the context of a complete video coder and with motion bit-rate included is given in Section 8.5.

Going back to results in Tables 7.2 - 7.4, we can see that, in terms of motion-compensated prediction error, the spline-based models slightly outperform block-based models at QCIF resolution and more significantly at QQCIF resolution. We can notice that the gain over block-matching is consistently the largest at the lowest (QQCIF) resolution - this is expected in the light of our earlier discussion on advanced spatial motion modeling. In Figs. 7·11 and 7·12, we visually inspect motion fields and corresponding prediction errors. It is clear that when the inter-nodal distance becomes relatively large compared to frame dimensions, the advanced cubic-spline motion model excels and demonstrates superior performance.

However, at CIF resolution (Table 7.4), HVSBM outperforms the fixed-size SP8 configuration; the probable explanation for this is that the use of variable-size blocks at the finest 4×4 level provides a sufficiently localized support for efficient prediction using a simple translational motion model. Until the problem of variable-size cubic splines (discussed in the previous section) gets solved, this result suggests that fixed-size splines might be a good choice for motion representation at very low resolutions. In the next chapter, we show

how different motion fields based on cubic splines and block matching can be combined at different spatial resolutions.

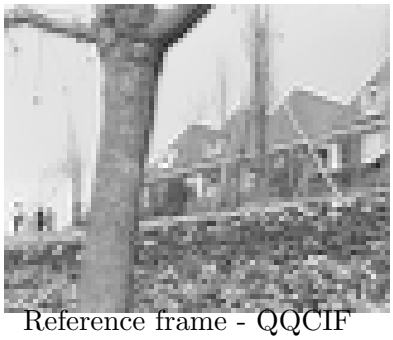
For a more complete evaluation of spline-based motion model, and its comparison with block-matching models, we need to compare the objective compression performance of video coders that take the true encoded motion bitrate into account and use the estimated motion fields for motion-compensated temporal filtering. As we will see in the experimental results section of Chapter 8, the excellent spline-based motion-compensated prediction, together with one-to-one mapping and natural invertibility support of continuous splines, indeed assures an excellent compression performance.

7.6 Conclusions

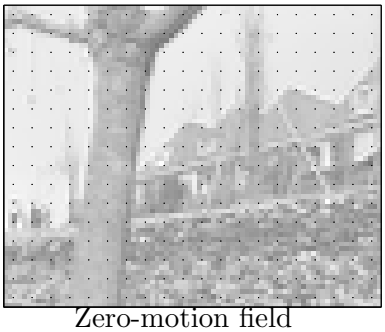
In this chapter, we discussed advantages of two spatially-advanced motion models, one based on deformable triangular mesh, and the other based on cubic splines. Both of these methods show good results in the context of wavelet video coding. Two main reasons for this are better spatial motion modeling (resulting in improved prediction) and explicit invertibility of both schemes (enabling efficient implementation of wavelet filters along well-defined motion trajectories).

Both models, however, incur high computational complexity (when compared to block-matching). In addition, they are not capable of supporting modeling of innovation areas (occlusions and uncovering). For these reasons, in the next chapter, along with a new scheme for spatio-temporal wavelet processing, we propose a scalable and hierarchical motion representation that combines the best properties of various motion models at different spatial scales.

Current frame - QQCIF

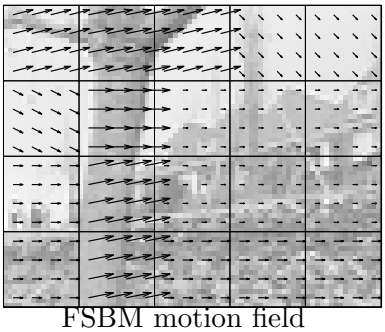


Reference frame - QQCIF



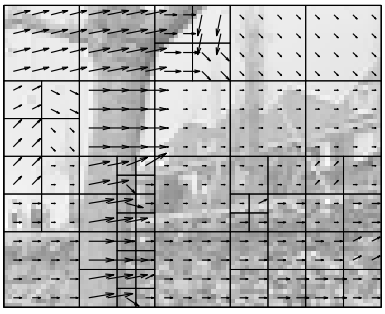
Zero-motion field

No motion, RMSE = 19.18, PSNR = 22.47dB



FSBM motion field

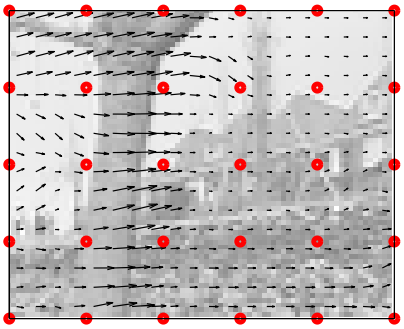
FSBM, RMSE = 8.71, PSNR = 29.32dB



HVSBM motion field

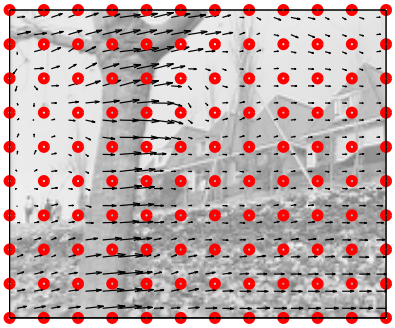
HVSBM, RMSE = 8.07, PSNR = 29.99dB

Figure 7·11: Motion-prediction performance of block-based motion models at QQCIF resolution, *Flower Garden* sequence.



SP16 motion field

SP16, RMSE = 7.89, PSNR = 30.19dB



SP8 motion field

SP8, RMSE = 7.63, PSNR = 30.48dB

Figure 7·12: Continued from Fig. 7·11: spline-based motion fields outperform block-based motion at very low resolutions in terms of motion prediction error.

Chapter 8

Motion modeling for spatial scalability

8.1 Introduction

Spatial video scalability gained a lot of attention recently and became one of the most sought after features in modern video coding systems. The reason for this lies in the dramatic proliferation of visual displays of different resolutions. Demand for simultaneous availability of the video on variety of different devices (e.g., cell phone, video iPod, PDA, notebook, TV, high-definition TV) has increased the interest in video compression schemes capable of decoding a "once-encoded" video at a range of supported video resolutions and with high quality.

Wavelet video coders and their open-loop non-predictive architecture provide an elegant solution to the problem of rate, temporal, and spatial scalability. In Section 8.2, we review the "t+2D" wavelet video coding architecture introduced earlier in Section 4.2 with special emphasis on the design details related to spatial scalability. It has been recently reported that, in the case when motion-compensated temporal filtering is performed along poor or non-existent motion trajectories, this architecture introduces visually disturbing artifacts. One possible solution to this problem includes a modification of the classical "t+2D" scheme to the so-called "2D+t+2D" coding structure, presented in Section 8.3. This structure specifically targets the spatial scalability performance as it allows a tradeoff between energy compaction and the potential for artifacts at lower spatial resolutions. From the perspective of this thesis this structure is interesting as it permits the use of different motion fields for separate MCTFs over spatial subbands of original video signal.

In order to explore this possibility, in Section 8.4 we introduce a framework for spatially-

scalable motion that is well-matched to "2D+t+2D" structure. A spatially-layered motion model provides an effective solution for predictive estimation and coding of motion fields at different spatial scales. In addition, at different resolutions, standard block matching may be combined with higher-order motion models, like that based on cubic splines discussed in the previous chapter. This effectively creates the so-called *mixture motion model*. In Section 8.5, we demonstrated that advanced spatial motion modeling is especially suitable at lower resolutions, where gains of the mixture motion model over block-only hierarchical model can exceed 1 dB. The optimal number of spline-based motion layers in the mixture motion model is heuristically determined, depending on the number of spatial resolution levels and application requirements. Finally, we compare coding efficiency of the "2D+t+2D" scheme to the standard "t+2D" scheme at full spatial resolution (architecture penalty) and analyze the impact of motion overhead introduced by the layered structure of mixture motion model (motion overhead penalty). Section 8.6 concludes the chapter.

8.2 Scalability analysis of "t+2D" wavelet coding structure

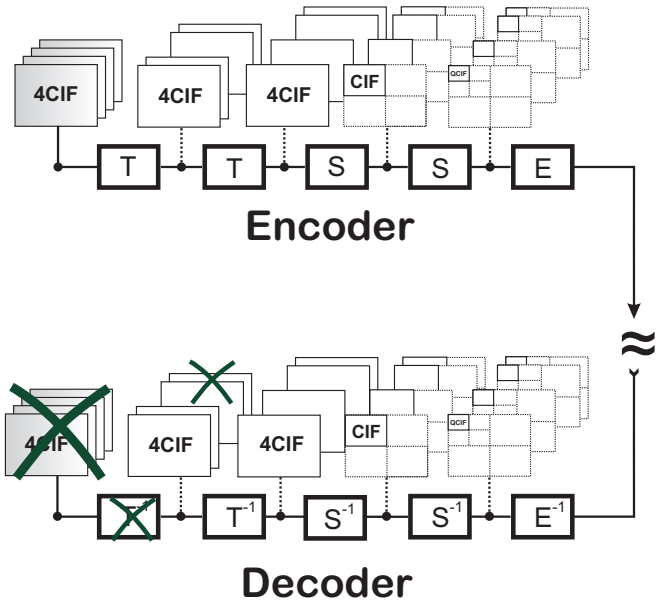
Over the last few years, the issue of spatial scalability has gained particular prominence because of the variety of emerging displays supporting a wide range of resolutions. This is further compounded by the growth of high-definition TV; most of the future video material will be captured using high-resolution video cameras and its presentation on low-resolution devices will require spatial scaling. There are several solutions to this problem. First, video data can be transmitted at the highest spatial resolution and downconverted at the receiver, but this requires a high-bandwidth channel down to the very receiver. Alternatively, the full-resolution bit-stream can be transcoded to a lower resolution (lower rate) at the interface of different-bandwidth networks but this requires complex and costly transcoding-capable network switching gear. Another solution, used notably in video databases, is to encode and store the video material at different spatial resolutions (and, thus, different bit-rates) but this requires more complex data management as well as additional storage capacity. A better solution is to employ spatially-scalable video coding. As detailed in

Section 4.2.1, the goal is to assemble a single bit-stream from which a sub-stream can be extracted and transmitted at lower bit-rate, and subsequently decoded at a lower spatial resolution.

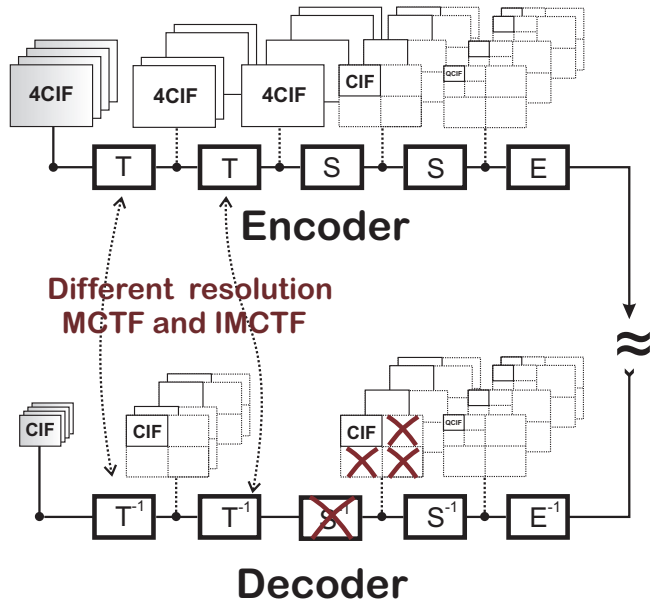
It was recognized early that scalability can be easily embedded into "t+2D" wavelet coding schemes (Fig 4·1). Rate scalability presents the strong point of this and similar wavelet-based schemes; the open-loop non-predictive structure and bit-plane coding of spatio-temporal subbands enable excellent quality scalability performance. Temporal scalability in the "t+2D" architecture is typically implemented by terminating temporal synthesis in the decoder at a desired temporal resolution (target frame-rate), which is illustrated in Fig 8·1(a). As this is the last synthesis block in the "t+2D" decoder, there is no negative impact on the decoded sequence quality (other than loss of temporal resolution).

The issue of spatial scalability is significantly more complicated and it recently received a lot of attention in the wavelet video coding community. The classic "t+2D" wavelet schemes show excellent performance when video is decoded at full spatial resolution, i.e., when the inverse MCTF in the decoder matches the forward MCTF in the encoder. Furthermore, it can be shown that the "t+2D" architecture necessarily maximizes compression efficiency for full-resolution decoding (Mehrseresht and Taubman, 2006). However, structures of this form may lead to artifacts when a decoder attempts to extract video at reduced spatial resolution by omitting one or more levels of spatial synthesis. The block scheme of the "t+2D" decoder operating at reduced spatial resolution is presented in Fig 8·1(b). We can see that the removed spatial synthesis block is followed by the inverse MCTF; in the case of motion failure, a mismatch between the motion in forward and inverse MCTF may introduce visual artifacts.

The main source of visible artifacts is motion failure, i.e., filtering over erroneous or non-existent motion trajectories at the original (highest) spatial resolution in the "t+2D" encoder. This motion failure causes spatial aliasing, present due to the use of finite-length spatial filters (more detail on this topic is presented in the next section), to appear as nonaligned artifacts in reduced spatial resolution sequences (Mehrseresht and Taubman,



(a)



(b)

Figure 8·1: Block diagram of a "t+2D" coding scheme. a) Removal of the last temporal synthesis stage causes no transform or motion mismatch; b) When a spatial synthesis block is removed, subsequent temporal synthesis is affected as the inverse MCTF now operates at a resolution different from forward MCTF.

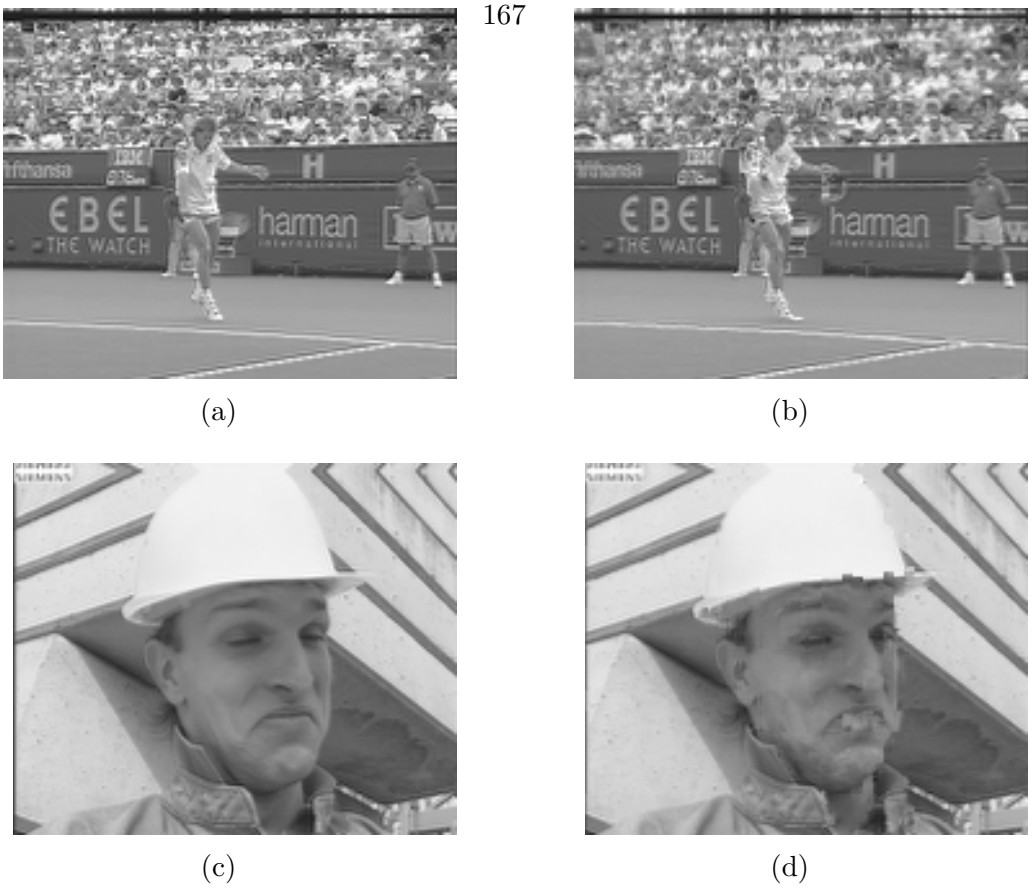


Figure 8·2: Visible artifacts in CIF-encoded/QCIF-decoded sequences *Stefan* (frame #20) and *Foreman* (frame #72), when the "t+2D" scheme from Fig. 8·1(b) is used. Motion for temporal synthesis is obtained by scaling-down motion estimated at CIF-resolution (HVSBM, $\lambda = 34$). a) LL spatial subband of the original CIF *Stefan* frame, b) CIF-encoded/QCIF-decoded *Stefan* frame. Notice the artifacts around player's left arm and right foot. c) LL spatial subband of original CIF *Foreman* frame, d) CIF-encoded/QCIF-decoded *Foreman* frame. Notice visible artifacts across the face.

2006). In the next section, we first illustrate the effect of motion failure on spatially-scalable performance of a "t+2D" coder, before investigating the cause of this effect and possible solutions.

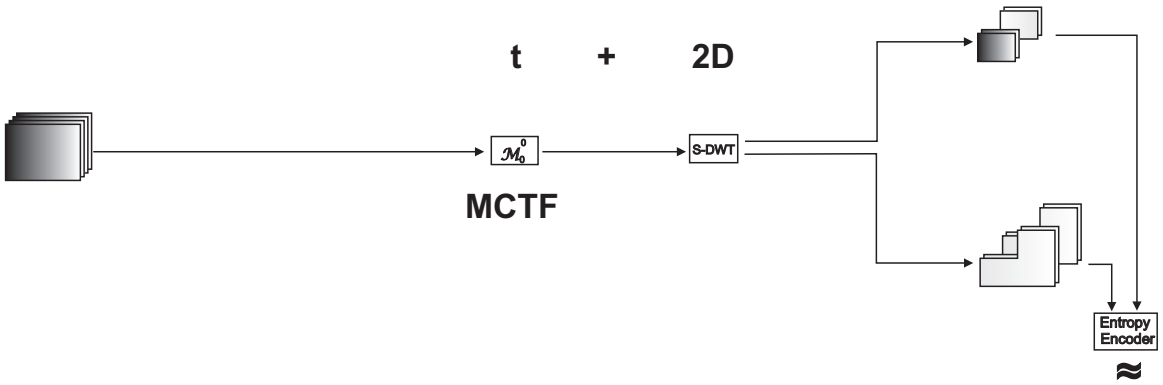
8.2.1 Motion failure and subband leakage

When one or more spatial synthesis stages are dropped, the inverse MCTF operates at a different resolution from that of the direct MCTF during encoding. The motion for inverse filtering at lower resolutions is not directly available - it is instead indirectly generated

from the original-resolution motion, often by simple spatial and amplitude scaling.

In the areas where motion is accurately estimated and properly employed for motion-compensated filtering, no visual distortions will be present in the low-resolution decoded "t+2D" video. However, in areas where the motion model fails, the spatially-reduced output frames exhibits visually annoying artifacts. To illustrate this effect, we show two such examples in Fig. 8·2. To emphasize the artifacts and make them more visible, we applied the MCTF along all motion trajectories, i.e., we considered all pixels in the sequence to be connected. To get a reconstructed frame, we used the "t+2D" structure with three levels of 5/3 MCTF, followed by five levels of spatial DWT. No quantization and entropy coding was used, which means that no additional quantization error is introduced (subband coefficients received by the QCIF-resolution decoder are identical to those coming out from the CIF-resolution encoder). When QCIF video is reconstructed using the approach from Fig 8·1(b), artifacts are clearly visible in the reconstructed sequences in all areas where motion is not accurately captured (e.g., at player's left hand and right shoe for the *Stefan* sequence and in the face in *Foreman* sequence). As no quantization error is present, the artifacts clearly arise from motion failure, affecting both subjective and objective coding performance.

To fully explain the source of these artifacts, the problem of "subband leakage" (Mehrseresht and Taubman, 2006) needs to be understood. To that end, we take a look "under the hood" of a standard "t+2D" encoder, presented in Fig. 8·3(a). Fig. 8·3(b) shows a block-diagram that is equivalent to such a coder. We consider the input to the "t+2D" coder as composed of low- and high-frequency component, by means of a spatial DWT. First, the input video sequence is spatially separated into two sequences. This is done by means of (finite support) spatial DWT analysis, zero-padding of appropriate subbands, and DWT synthesis. These two sequences are then subjected to independent MCTF (both MCTFs use the same motion), and spatial DWT. Due to linearity of all transforms, the resulting spatio-temporal subbands will be the same as if the MCTF were performed on the original sequence. This new interpretation of the coder, however, can shed significant light on the impact of dif-



(a) "t+2D" scheme with one temporal and one spatial decomposition level.

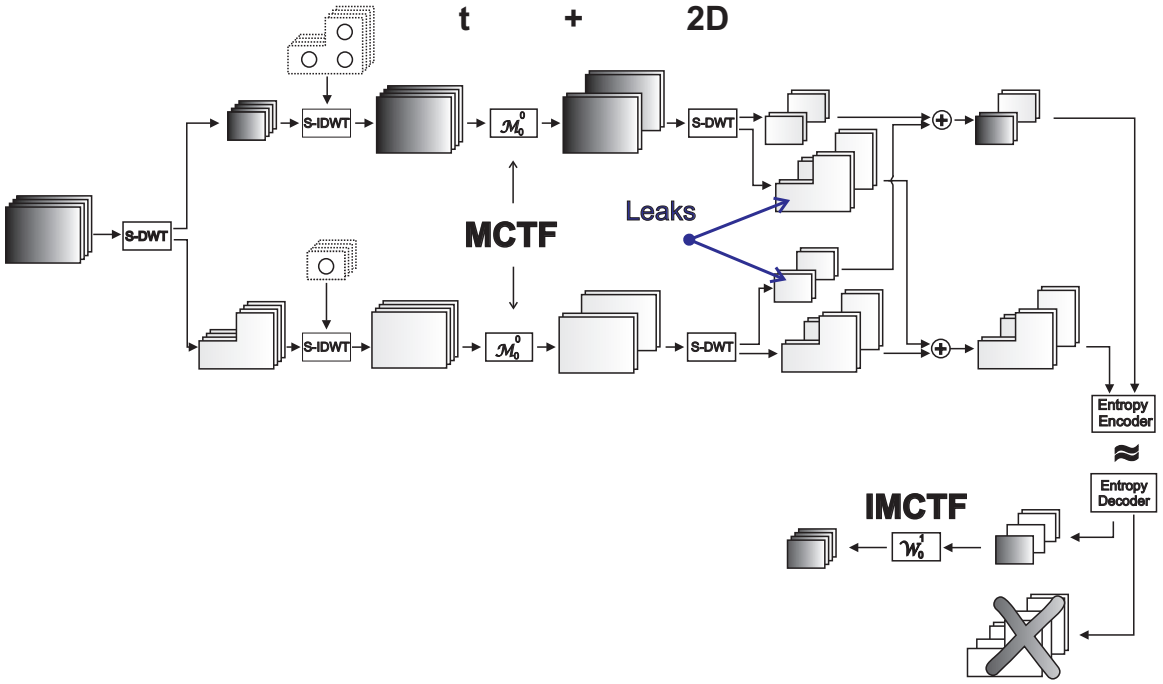


Figure 8·3: (b) Block diagram of the same "t+2D" scheme illustrating the subband leakage phenomenon.

ferent spatial frequencies from the input video sequence on the generated low/high spatial subbands.

Due to the non-ideal spatial filtering in the finite-length 2D DWT step (typically, CDF 9/7 (Taubman and Marcellin, 2002)), spatial aliasing will be present in both high- and low-frequency sequences. It is a well-known fact that an MCTF of image sequence containing low spatial frequencies will result in an output with some high spatial frequencies. An excellent coverage of this topic can be found in (Mehrseresht and Taubman, 2006; Xiong et al., 2005a). Similarly, a motion-compensated temporal filtering of image sequence with high frequencies will create some low spatial frequencies. This effect is referred to as *subband leakage* (Fig. 8·3) and it is caused by critical sampling and non-ideal spatial filtering of the DWT.

The dyadic decimation of the DWT causes the subband leakage problem after one level of temporal analysis to be the most severe when displacements equal an odd number of pixels; it decreases when displacements approach an even number of pixels, and it is zero when displacements corresponds to an even number of pixels. Naturally, the strength of leakage depends on the quality of filters used in 2D DWT. Filters with better frequency discrimination can help reduce the leakage, at the cost of increased computational-complexity. Instead of the widely-used CDF 9/7 (Taubman and Marcellin, 2002), adopted by JPEG2000, several longer filters were proposed to reduce inter-resolution frequency leakage, like CDF 13/11, and CDF 17/11. Although leakage effects are reduced using better spatial filter, the improvement is reported to be relatively small (Mehrseresht and Taubman, 2006). A similar attempt at limiting the amount of spatial aliasing by constructing better spatial filters was proposed by Wu (Wu, 2005). Inspired by MPEG-4 lowpass filtering, two content-adaptive methods for aliasing reduction were designed, effectively improving low-resolution reconstruction by an average of about 2dB.

The well-known "2D+t" schemes provide one solution for breaking this subband dependency; spatial synthesis in the "2D+t" decoder is performed last. When dropped at the decoder, it does not affect other decoding steps. Unfortunately, this approach comes

at the cost of lower coding efficiency due to significantly less efficient motion compensation in the critically-sampled wavelet domain.

Other more recent solutions proposed in the literature that aim at addressing the spatial scalability performance of the "t+2D" coder and reduction of the amount of subband leakage include shift of motion compensation to the overcomplete DWT domain (Andreopoulos et al., 2003), and optimal subband rate allocation (Xiong et al., 2005a). The latter approach preserves a certain share of otherwise completely discarded spatially-high subband; the total bit-budget is split between the coefficients corresponding to low and high spatial frequencies in such a way that the overall reconstruction error is minimized.

8.3 Alternative architecture using parallel MCTF design

While better spatial filters and highly accurate motion estimation (coupled with its careful employment in the MCTF) can be used to significantly reduce the number of visible artifacts, "t+2D" scheme still fails to guarantee an "artifact-free" decoding at reduced spatial resolutions.

In order to overcome this limitation and guarantee artifact-free reconstruction, a modified structure for implementing MCTF was recently proposed under the name of "2D+t+2D" (Mehrseresht and Taubman, 2006) or "in-scale" MCTF (Xiong et al., 2005b). In the "2D+t+2D" approach, several levels of the so-called "pre-S" spatial DWT decomposition are followed by *subband-independent* temporal filtering (Fig. 8·4). Additional levels of spatial 2D-DWT decomposition (used primarily to increase coding efficiency) complete the subband analysis. This structure trades off energy compaction for a removal of artifacts at lower spatial resolutions.

In our notation, we use subscript/superscript index of motion field $\mathcal{M}$ to denote spatial/temporal resolution at which the MCTF is performed. The most important thing to note is that the parallel implementation of MCTFs at different scales eliminates the subband leakage, at the cost of lower energy compaction (mainly due to the sub-optimal

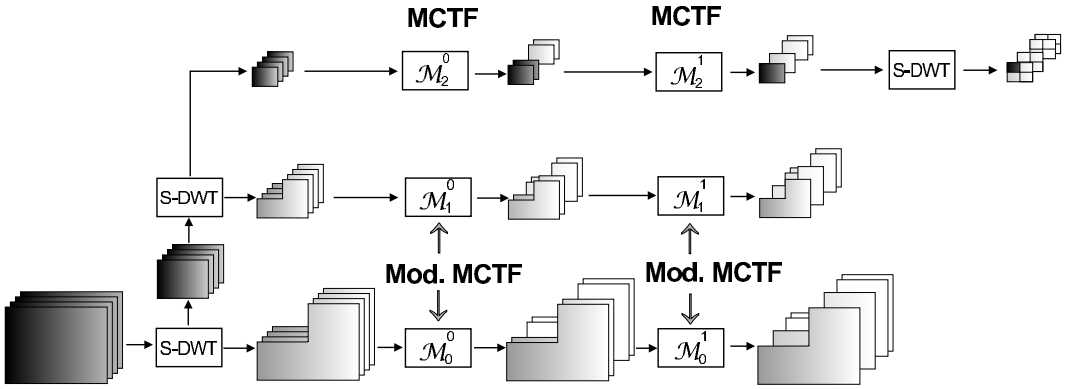


Figure 8·4: "2D+t+2D" encoder supporting three levels of spatial scalability.

motion filtering of low spatial frequencies).[1] It is important to notice that that MCTFs at all but the lowest spatial resolution are not performed in the transform domain - each modified temporal processing block ("Mod.-MCTF" in Fig. 8·4) consists of zero padding, inverse 2D-DWT, MCTF, and forward 2D-DWT.

Most interestingly, this structure opens a possibility for constructing spatially-scalable motion representation, which can be estimated and optimized for each spatial resolution level. In the majority of prior work on this topic, the motion used for MCTF at all spatial scales of an "2D+t+2D" architecture was derived from the motion estimated at full-resolution, by means of scaling and subsampling (Mehrseresht and Taubman, 2006). In the next section, we present a novel framework for spatially-scalable motion representation that is well-matched to the hierarchical structure of spatially-scalable "2D+t+2D" coder.

8.4 Modeling motion for spatial scalability

Scalable coding of motion became an issue relatively recently with the introduction of scalable wavelet video coders. Motion with a degree of coding error, not permitted in

[1]Some form of low-to-high leakage compensation can still be used to communicate the high-frequency subbands generated during IMCTF from lower to higher resolution MCTFs, in order to limit the structure loss at higher resolutions.

hybrid coders, is permitted in wavelet-based coders due to the open-loop nature of the MCTF. Rate scalability of motion has been considered in order to improve compression performance at lower bit-rates (Secker, 2004), by means of shifting a portion of the total bit-budget from motion to texture. However, simple bit-plane coding of motion parameters is highly sub-optimal and not suitable for motion compensation at lower spatial resolutions. In the context of the MC-EZBC coder, Wu (Wu, 2005), proposed a method for generating spatially-scalable motion by layering motion in two or more spatial layers. Still, in Wu's approach, all motion is estimated at the highest resolution only. Below, we present a general framework for the estimation of spatially-scalable motion fields. This framework can be seen as an extension of the model previously introduced in Section 7.5, but more closely analyzed in the context of spatially-scalable video coding and rate-distortion optimization.

Let $\mathbf{M}$ be a set of motion fields at all spatial scales that are supported by a spatially-scalable coder, $\mathbf{M} = \{\mathcal{M}_0, \mathcal{M}_1, ..., \mathcal{M}_{L-1}\}$, where L is the number of spatial scales. In the most general case, the formulation of joint motion estimation across all spatial scales is:

$$\hat{\mathbf{M}} \;=\; \text{argmin } J(\mathbf{M}), \quad J = \sum_{l=0}^{L-1} c_l(D_l + \lambda_l R_l), \tag{8.1}$$

where c_l's are scale-normalization factors, and D_l, λ_l and R_l are the standard distortion, regularization factor, and motion rate at resolution level l (it should be made clear that $\mathbf{M}$ is implicitly included in both distortion and motion rate cost, as discussed in Section 2.2). Obviously, minimizing this cost function simultaneously across all scales is a computationally challenging task. Instead, we propose to solve iteratively, starting from the highest spatial scale[2] $L-1$, a simpler cost function:

$$\hat{\mathcal{M}}_l \;=\; \text{argmin } J_l(\mathcal{M}_l), \quad J_l = D_l + \lambda_l R_l|_{\hat{\mathcal{M}}_{l+1}, \hat{\mathcal{M}}_{l+2}, ..., \hat{\mathcal{M}}_{L-1}}, \tag{8.2}$$
$$l = L-1, ..., 1, \quad \mathcal{M}_L \triangleq 0,$$

where $\hat{\mathcal{M}}_k, k > l$ are motion fields already estimated at scales higher than the current

[2]The highest scale corresponds to the lowest resolution and vice versa.

scale l. R_l is the rate needed to represent motion at scale l (possibly using *prediction* from previously-estimated motion fields). A practical design of this prediction model and its prediction contexts depends on the particular motion models used. In the next two sections, we propose two motion estimation schemes: one employing variable-size block matching (VSBM) at all spatial scales, and the other using a mixture of hierarchical cubic splines (at lower scales) and VSBM.

8.4.1 Spatially-scalable motion estimation using VSBM

In modern video coders, motion vectors obtained by HVSBM are, typically, predictively coded by sequentially following the quad-tree decomposition structure. The prediction residual is then coded using the context-based adaptive arithmetic coding (CABAC). The majority of attempts to generate scalable motion relied on estimating motion at a single (highest) spatial resolution. A large λ was used to generate motion base layer and progressively smaller λ's were used for the refinement of this motion field, but still at the same spatial resolution (Xiong et al., 2005b). In our method, motion is hierarchically estimated at all spatial scales; for the purpose of coding, both spatial and cross-resolution predictors are allowed (Fig. 8·5). This adaptive prediction scheme is deployed at macroblock level. The decision on a particular prediction mode is controlled by the variance of motion estimates and the current depth in partition tree. The most frequently used prediction modes are median, weighted median, and average of $\vec{A}$, $\vec{B}$, $\vec{C}$, $\vec{P}$ (Fig. 8·5), as well as simple prediction using $\vec{P}$. In addition, macroblock partitioning from the next lower resolution is used to initialize the state of four MBs in the current layer; while these initial sub-blocks may be subject to additional partitioning, their subsequent merging is not allowed. In order to speed up motion estimation at higher resolutions, (scaled) motion vectors estimated at lower resolution are used as search candidates.

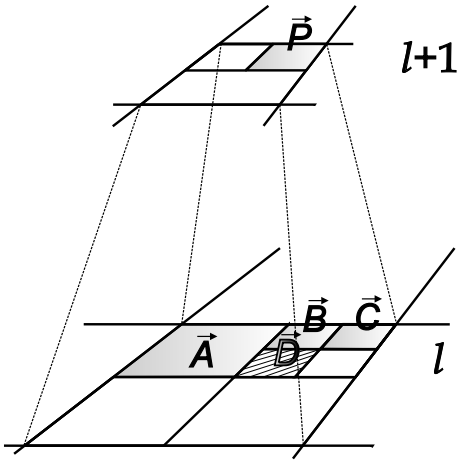


Figure 8·5: Motion prediction: both same-scale ($\vec{A}, \vec{B}, \vec{C}$) and previous-scale ($\vec{P}$) motion vectors are used to predict motion vector of the current block ($\vec{D}$).

8.4.2 Mixture motion model

Most of the current MC-DWT coders rely on block-based motion models inherited from the hybrid coding structure. The excellent performance of block matching in the hybrid context can be easily explained: block-based motion perfectly aligns with block-based decorrelating DCT transform used. In contrast to the hybrid scenario, there is a mismatch between the local support of the block motion model and a global (multi-scale) nature of spatial 2D-DWT. This motivated us to introduce a more flexible motion model based on hierarchical cubic splines in Chapter 7. With more degrees of freedom, this model is bound to result in better motion compensation at high spatial scales than the block-constant model.

We already demonstrated excellent motion-compensated prediction performance of higher-order cubic spline motion fields at low spatial resolutions, like QQCIF and QCIF (Section 7.5). We now create a "mixture motion model" by replacing VSBM with splines at lower layers of the spatially-scalable motion. A small number of spline control nodes can accurately describe motion field at high scale, resulting in improved motion compensation, lower motion rate, and better prediction of motion at the next spatial level.

At the "switch" layer,[3] the motion prediction scheme is similar to that of VSBM (Fig. 8·5). The biggest difference is that the predictor $\vec{P}$ from the previous layer $l+1$ is now derived from a spline-based field and may vary over the MB support. We note that, in contrast to VSBM, the continuous spline model can improve the accuracy of a cross-scale (low resolution) predictor; instead of a simple scaling of motions vectors, a better predictor might be available when the continuous spline is re-evaluated at the current resolution.

8.5 Experimental results

We implemented the motion estimation algorithms for mixture motion model within the framework of the MSRA (Xu et al., 2001; Xiong et al., 2005c) scalable wavelet coder. For VSBM, we used both spatial and cross-scale motion prediction, 16×16 initial macroblocks (with block sizes down to 4×4), and 1/4-pel motion accuracy at each resolution level. For spline-based motion estimation, we used only fixed-size control grid (8×8) within each spatial resolution, for the reasons described in Section 7.4.1. Motion from different temporal resolutions was independently coded; in the future, more efficient joint temporal motion coding should be addressed to maximize coding gain by reducing the overall motion overhead.

In the context of the "2D+t+2D" coding scheme, the new layered motion model (mixture motion model) increases the number of possible parameter combinations in coding configuration. With several additional axes of freedom in the coding setup available, our main focus remains the performance of the mixture motion model, which includes estimation of the optimal number of spline-based motion layers at the top of the spatial pyramid and its comparison (where possible) with the performance of a single-layer HVSBM motion.

In our experiments, we analyze the "2D+t+2D" coder that supports three spatial resolutions: CIF, QCIF, and QQCIF. As inputs, we use CIF sequences, *Foreman* and *Mobile and Calendar*. We vary the number of spline-based motion layers (K), and use the

[3] A switch layer is a spatial layer of motion based on a different model than the next- or previous-scale layer, e.g., block versus spline and spline versus block.

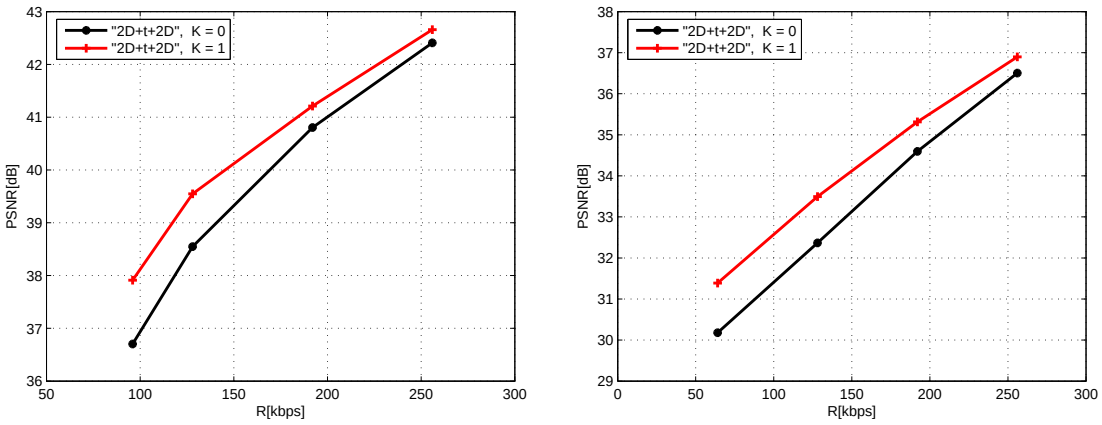


Figure 8·6: Comparison of "2D+t+2D" CIF-encoding/QQCIF-decoding, 15Hz. a) *Foreman*; b) *Mobile and Calendar*.

VSBM for the rest of $L - K$ high-resolution levels. In our practical setting with $L = 3$, the no-splines case corresponds to $K = 0$, while the all spline model is defined by $K = 3$. For comparisons at the highest (CIF) resolution only, we also use a single-layer HVSBM motion estimated at the highest resolution.[4]

Results obtained from spatially-scalable QQCIF decoding of a bitstream produced by the "2D+t+2D" encoder are presented in Fig. 8·6. This is, at the same time, the lowest supported spatial scale of our configuration. There are only two possible choices for motion at this spatial level: VSBM (corresponding to $K = 0$) and spline ($K = 1$). We notice that the spline-based model outperforms the block-based motion model at this very low resolution, typically by more than 1dB. It is important to note that this result, at the same time, completes the comparison of two motion models started in Section 7.5. In Table 7.2, for example, only the motion prediction performance of VSBM and SP8 motion models was compared (columns 3 and 5), without taking into account the cost of motion coding. Only after implementation of both motion estimation algorithms and models in the context

[4]Due to a non-layered structure of this motion field, comparison with methods using a layered motion structure is difficult because of the motion overhead issue. A fair comparison would require using some of the more sophisticated methods for layered-motion generation from the highest-resolution motion, like that of Wu (Wu, 2005).

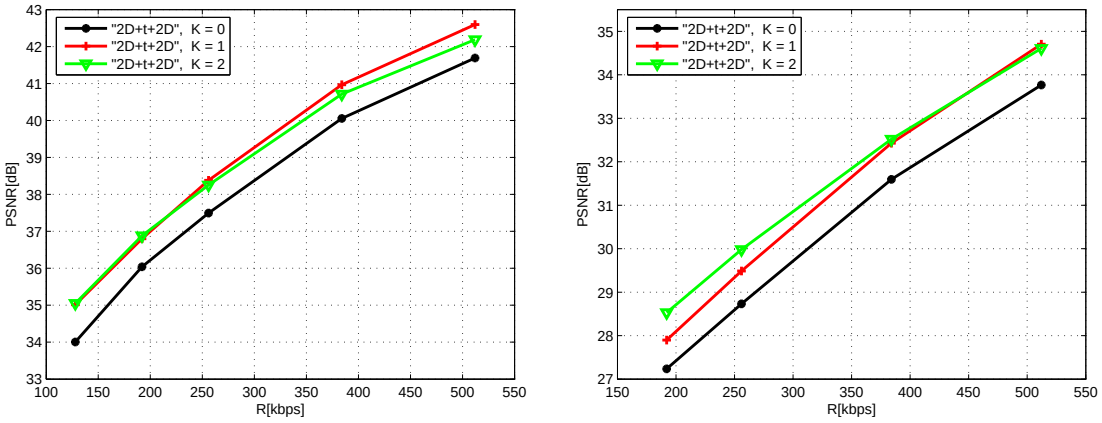


Figure 8·7: Comparison of "2D+t+2D" schemes for CIF-encoding/QCIF-decoding, 15Hz. a) *Foreman*; b) *Mobile and Calendar*.

of a real video coder, as done to obtain results in Fig 8·6, and with the motion bit-rate included, can we indeed claim that splines (with internodal distance of 8) are outperforming the VSBM at QQCIF resolution.

A similar comparison at the intermediate QCIF resolution is presented in Fig. 8·7. At QCIF, the possible options for motion are two-block layers ($K = 0$), one spline and one block ($K = 1$), or two spline layers ($K = 2$). Depending on the sequence, the best performance at QCIF resolution is achieved with one or two spline-based motion layers. Notice that the all-spline model up to QCIF resolution ($K = 2$) performs best at lower bit-rates, while the combination of splines and VSBM ($K = 1$) shows improved performance at higher bit-rates.

In Fig. 8·8 we visually compare a QCIF decoded frame #33 from *Mobile and Calendar* sequence, and draw a similar conclusion. As a reference for the both visual and PSNR comparison we use the "2D+t+2D" natural reference, which is an LL spatial subband of the CIF original video.

Finally, we investigate performance of the mixture motion model in the "2D+t+2D" configuration and the effect of the number of spline-based motion layers at full resolution. We compare high-resolution (CIF) decoding of schemes with $K = 0$, $K = 1$, $K = 2$, or

(a) LL subband of original CIF-resolution frame

(b) $K = 0$, PSNR = 27.18dB

(c) $K = 1$, PSNR = 28.01dB

(d) $K = 2$, PSNR = 28.56dB

Figure 8·8: Comparison of reconstructed frame #33 from *Mobile and Calendar* sequence at 192kbps, "2D+t+2D" CIF-encoding/QCIF-decoding, 15Hz. a) LL subband of CIF-resolution frame; b) $K = 0$ (VSBM); c) $K = 1$ (one spline-base motion layer); d) $K = 2$ (two spline-base motion layers).

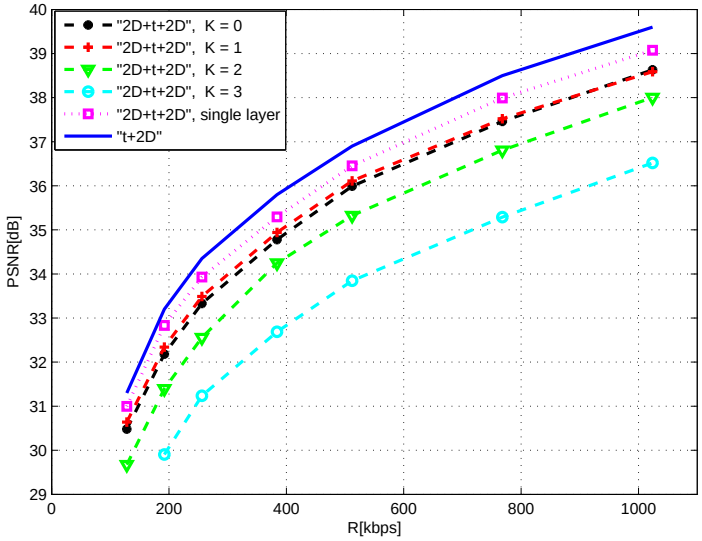


Figure 8·9: R-D performance as a function of the number of spline-based motion layers (K) for *Foreman* sequence, CIF encoding/CIF decoding at 30Hz. The best results for "2D+t+2D" with layered-motion are obtained for $K = 1$.

$K = 3$ levels of spline-based motion (for $K < 3$, VSBM is used at higher resolutions). For both coded sequences, the best results were obtained with $K = 1$ spline layers (Figs. 8·9 and 8·10); this result adds to an already excellent performance of spline-based model at QQ-CIF and QCIF resolutions. We can also notice that full-resolution decoding performance drops when more than one spline motion layers are included at higher spatial resolutions; clearly, the fixed-size spline model is outperformed by variable-size block matching at high resolutions.

There are two explanations for the poor CIF-resolution performance of all-spline model. First, as demonstrated in Section 7.5, variable-size blocks are doing a better job in terms of motion rendering than fixed-size splines at higher resolutions. More generally, even if the variable-size concept is introduced in the spline framework in the future, the smooth spatial model of cubic splines might be too smooth at the highest resolutions and prevent accurate modeling of discontinuities. Certainly, both of these issues deserve further investigation. While we would expect results to improve with the use of variable-topology splines, it is likely that a mixture of spline- and block-based models would still be the most R-D-effective

solution, this time possibly with more spline levels included. More generally, future work should include a larger number of different testing points (at both reduced resolutions and reduced frame rates), in order to determine the optimal motion layer configuration.

To assess the coding loss due to motion overhead of the layered mixture model, we also include results of the "2D+t+2D" scheme using a single-layer high-resolution motion (lower resolution motion is generated by resolution and amplitude scaling). We observe that the loss caused by motion overhead rarely exceeds 1 dB (in the case when zero or one layers of spline-based motion are used), which testifies to the good predictive coding performance of hierarchical motion model and its relatively small motion overhead.

Finally, to this full resolution comparison, we add a "t+2D" coding result using a single spatial layer of HVSBM at the highest spatial resolution.[5] The set of motion fields used is the same as the single-layer motion is the same motion used for a "single-layer" motion in "2D+t+2D" coder. In this case, the difference between the two R-D curves can be seen as a penalty introduced by the "2D+t+2D" architecture at the highest spatial resolution. Clearly, the performance of "t+2D" coder cannot be matched by any of the "2D+t+2D" configurations, due to the aforementioned sub-optimal motion-compensated filtering of low spatial frequencies; depending on the number of spline-based levels, however, it can be kept relatively small (under 1.5 dB).

Although performance of the "2D+t+2D" structure with layered motion suffers at the highest resolution, this coding loss is often not objectionable (Fig. 8·11) and might be acceptable if artifact-free reconstruction at lower spatial resolutions is of primary concern. Naturally, a larger number of supported spatial levels in the "2D+t+2D" scheme would further reduce the full-resolution coding gain; some tradeoff and a good balance should be heuristically determined, based primarily on the coding application.

[5] For the same resolution encoding/decoding, regardless of the quality of motion estimate and filtering, "t+2D" is an optimal configuration.

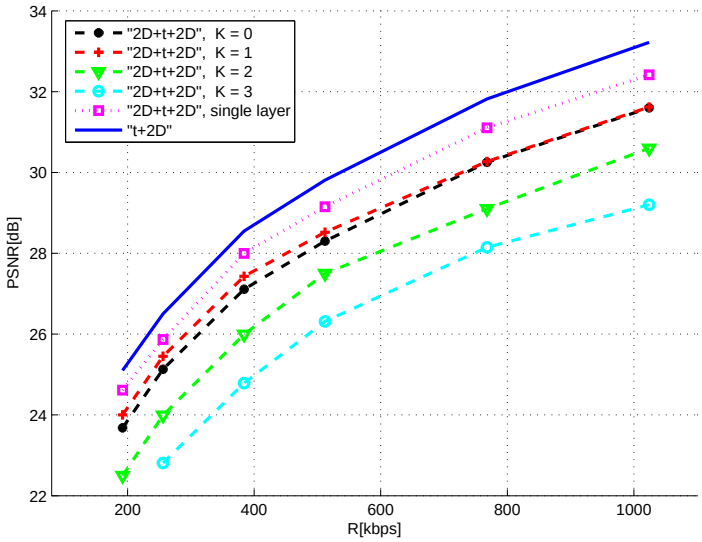


Figure 8·10: R-D performance as a function of the number of spline-based motion layers (K). for *Mobile* sequence, CIF encoding/CIF decoding at 30Hz. The best results for "2D+t+2D" with layered-motion are obtained for $K = 1$.

8.6 Conclusions

In this chapter, we analyzed the spatial scalability performance of wavelet video coders. In the context of recently introduced "2D+t+2D" wavelet video coders, we proposed amd implemented the a new approach to spatially-scalable estimation of motion and introduced a mixture motion model combining cubic-spline motion model with block-based motion model at different supported spatial scales. The improved spatial modeling by cubic splines can lead to a significantly improved R-D performance of "2D+t+2D" schemes (in excess of 1 dB), especially at very low resolutions. We experimentally determined that at least one spline-based motion layer should always be used for the lowest supported spatial resolution; the particular choice of the number of supported spatial resolutions (number of different MCTF levels in "2D+t+2D" structure) and the best number of spline-based motion layers is application dependent and should be determined heuristically.

(a) PSNR = 34.68 dB (b) PSNR = 33.30 dB

Figure 8·11: Comparison of reconstruction at 256kbps, frame #44 from *Foreman* sequence, CIF/CIF encoding decoding, 30Hz. a) "t+2D"; b) "2D+t+2D" using $K = 1$ spline motion levels.

Chapter 9

Conclusions and future directions

This chapter summarizes the contributions of this dissertation and analyzes directions for future research in this area.

The major inspiration for our work has been a substantial improvement observed over the last five years in motion compensation performance of sophisticated rate-distortion optimized block-based motion models. Most often, these motion models are applied in the context of hybrid (block-transform) video coders. It has been clearly demonstrated that most of the coding gains observed in the new generation of hybrid video coders (e.g., AVC/H.264) are indeed a result of better motion handling. At the same time, a new class of open-loop non-predictive video coders emerged, under the common name *3D video coders*. This shift in the spatio-temporal transform and in coding paradigm carries the need to revisit many old and ask some important new questions regarding motion modeling.

The goal of this thesis is to provide answers to these motion-related questions in the context of 3D video coding. The main issues include: possible motion representation and compensation in the frequency, rather than spatial, domain; importance of well-defined motion trajectories and the related motion invertibility problem; issues related to detecting and using occluded/uncovered frame regions for maximum temporal video decorrelation; advanced spatial motion modeling, better suited to capturing a true scene motion (especially when the number of motion-compensating patches per frame is limited); and finally, efficient and flexible motion representation across a number of desired spatial resolutions.

Before we summarize the main findings of this thesis, we offer a high-level view of our approach to motion modeling and compare it to the mainstream R-D optimized motion estimation approach. We would like to make clear that the majority of immediate applica-

tions for the theoretical results and algorithms developed over the course of our research are found in video compression. However, we believe that our approach has an added benefit over the rate-distortion optimization motion search (like the one found in AVC/H.264). With an increasing level of sophistication in R-D motion optimization, the AVC/H.264 approach may result in motion fields that have little or no physical meaning,[1] and should be interpreted purely in a video coding context. Most of the time, we formulate the motion representation and estimation problem in such a way that the true physical motion in a scene is closely modeled, which might be beneficial not only for the purpose of efficient temporal decorrelation but also for the more general scene analysis. While it is certainly a challenging task to match the performance of R-D optimized motion estimation algorithms, we believe that our approach is much more likely to be reused in various image and video processing problems (e.g., video segmentation, occlusion detection) and it therefore has a potential to benefit a wider research community.

There are two major problems addressed in this dissertation - each of our efforts is oriented towards overcoming one of the limiting factors encountered in hybrid video coding systems. In the context of *low-complexity video coding*, we proposed and implemented a 3D DCT-based video coder that can successfully compete with more efficient profiles of the MPEG coding family (MPEG-2 and MPEG-4), sometimes at less than half computational complexity. The main computational savings are obtained by omitting the expensive motion compensation step in spatial domain and applying the fast separable 3D DCT transform directly to input video data. The DCT spectrum obtained in this way is, however, strongly dependent on spatio-temporal motion in the original sequence. For that reason, we first performed a theoretical analysis of spectral properties of a 3D DCT-transformed video and developed a closed form solution for the 3D DCT spectrum in the case of uniform translational motion.

We then proposed to use the resulting characteristic energy footprint for efficient video

[1] The most obvious example for this is the multi-hypothesis approach, where different regions of the same frame may be predicted from completely different temporal references, therefore departing from the idea of motion as a temporally-continuous phenomenon.

coding in the DCT domain. As the first step, we introduced a new 3D quantization model. In order to improve entropy coding of the quantized 3D DCT coefficients, a reduced set of motion parameters is estimated[2] and used for motion-adaptive scanning. This approach results in our proposed coder outperforming all previously proposed 3D DCT-based coders by a wide margin (more than 2 dB). It also compares favorably to standard MPEG-2 video coder at complexity reduced by about 25% and closely matches the performance of MPEG-4 coder at less than half complexity.

To the best of our knowledge, our work on motion analysis for 3D DCT coding provides the most complete theoretical and practical coverage of the topic. In addition to respectable coding performance, the importance of this work also lies in the fact that it the first time it shows theoretically and demonstrates practically why a fixed scan of the transformed 3D DCT coefficients, regardless of the level of its sophistication, cannot provide a viable solution to efficient 3D DCT video coding. As a number of papers are still being published on the topic of 3D DCT coding every year, our theoretical analysis showing both the limits and limitations of the approach might prove beneficial to the wider community of researchers.

The second part of this thesis deals with some of the most important motion-related issues within the 3D DWT/MCTF coding framework. A new paradigm of MC temporal filtering (as opposed to MC prediction used in hybrid schemes), combined with the replacement of the block-based DCT with DWT, requires a special consideration for motion compensation problems. We start by providing a theoretical foundation to the problem of motion invertibility, central to the interpretation of the popular lifting MCTF. We then demonstrate experimentally that factors other than motion-compensated prediction error also play an important role in the coding performance of MCTF-based coders. To quantify this effect, we introduced an objective measure for motion invertibility error, that can be used both as an indicator of coding performance and a regularization factor in the motion estimation process. The existence of motion trajectories (i.e., both forward and backward

[2]This can be done in either spatio-temporal or DCT domain.

motion fields between the same two video frames) opens a door for additional improvement of both prediction and update lifting steps, based on implicit occlusion/uncovered area detection. The implemented approach of "occlusion-aware" motion-compensated temporal filtering can lead to significant (larger than 1 dB) coding gains, and can also facilitate better spatial scalability performance of wavelet video coders, by preventing filtering over non-existent motion trajectories.

The last part of this thesis deals with advanced spatial motion modeling that has a potential for driving future state-of-the art video coding systems. Our extension of deformable mesh motion model includes advanced handling of boundary areas (scene parts leaving or entering a frame) and results in up to 0.5 dB improvement over the regular mesh. Being well-matched to physical camera and object motion, mesh-based models, and their hierarchical extensions in particular, will remain of interest in the future. We also introduced to wavelet video coding a novel framework for motion modeling based on hierarchical cubic splines. We demonstrated that advanced (third-order) motion modeling can significantly improve motion-compensated prediction and, subsequently, the overall coding performance over block-based models, especially at very low spatial resolutions. By combining spline- and block-based motion into a consistent motion framework, we create the so-called "mixture motion model". This motion structure is particularly suitable for flexible and layered motion representation across a broad range of spatial resolutions; high-order motion modeling improves prediction performance at coarsest scales. Combining it with rate-distortion optimized variable-size block model at finer scales may help in achieving a globally efficient solution. We demonstrated that even a simple implementation relying on fixed-size spline motion model can lead to additional coding gain over the block-motion structure.

9.1 Contributions

This thesis investigate motion-related issues in two coding contexts: low-complexity compression (based on 3D DCT) and scalable compression (based on 3D DWT). In the area of low complexity coding, the main contributions of this dissertation are:

- Development of the closed form solution for 3D DCT spectrum of uniformly translating frame sequence.
- Introduction of new 3D DCT quantization model and motion-adaptive scanning of transformed coefficients.
- Development of complete end-to-end video coder based on 3D DCT competitive with MPEG-2 and MPEG-4 hybrid coders at significantly reduced complexity.

The main contributions of this dissertation in the field of scalable wavelet-based video coding are:

- Theoretical analysis of motion invertibility problem and its quantification through a novel motion invertibility measure.
- Introduction of novel spline-based motion inversion and classification of existing motion inversion methods.
- Development of new occlusion-driven temporal filtering based on indirect occluded/exposed area detection.
- Extension of mesh-based motion model to allow better handling of frame boundaries.
- Introduction of spatially-advanced motion modeling in the form of cubic splines to scalable video coding.
- Development of a mixture motion model that is suitable for "2D+t+2D" coding architecture and which combines spline-based and block-based motion models at different spatial resolutions.

9.2 Future work

In this section we briefly describe several directions in which our work on low-complexity and scalable video coding might be extended.

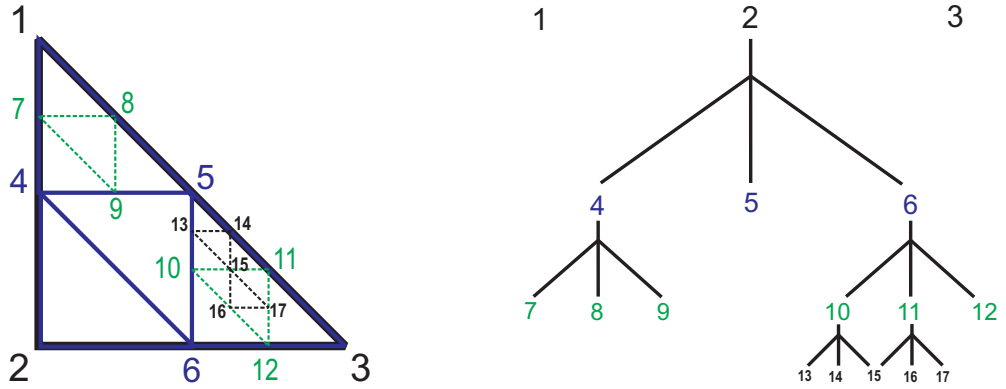


Figure 9·1: Hierarchical variable-size mesh: (a) Example of hierarchical partitioning of a triangular patch; by adding new nodes, hierarchical variable-size mesh model can accurately track motion in the areas of high motion variability (e.g., patch 15-16-17) while in the areas of smooth motion larger patches might be used (4-2-6). (b) "3-tree" ("ternary-tree") corresponding to partitioning in (a).

9.2.1 Extension of 3D DCT coder to include more complex motion modeling

Our current implementation of 3D DCT coder models motion as a uniform translation over the entire GOP. When the length of the GOP is sufficiently small and/or the 3D DCT volume covers the region with motion that is close to a uniform translation, this approach is justified. However, for maximal coding gains, a 3D DCT coder should support more advanced motion trajectories, like temporally-accelerated models of Chahine and Konrad (Chahine and Konrad, 1995). The development of a close-form solution for the case of temporally non-uniform motion, and investigation of appropriate scanning order might additionally improve the performance of our coder.

9.2.2 Extension to variable-size mesh- and spline-based motion models

Impressive coding gains followed the introduction of variable-size blocks into block matching. In order to compete with VSBM and improve its prediction performance, new motion models should also employ non-uniform (variable-size) node topology.

Both variable-size triangular meshes and quadtree splines could provide a convenient

way to use adaptively-sized patches for motion estimation, while maintaining inter-patch continuity. The practical issues on how to actually determine the topology of patches still remain open. Ideally, one would like each patch to cover a region of the image within which a parametric motion model is valid.

One of the many interesting issues related to the "patch-partitioning" problem is illustrated in Fig. 9·1. Similarly to quad-tree partitioning of VSBM, each "split" step in the case of triangular mesh produces four new smaller patches, effectively refining the motion estimate (one example of triangular patch partitioning is given in Fig. 9·1(a)). Unlike quad-tree, however, the corresponding dependency tree now has three leaves in each branch - we therefore call it "3-tree" or "ternary tree" (Fig. 9·1(b)). Similar structures have been investigated in computer science as "B-trees" or "2-3 trees" (Bayer, 1971; Cormen et al., 2001). In addition to the problem of optimized patch partitioning, an efficient representation of these trees would play an important role if motion is to be efficiently coded.

9.2.3 Relaxation of one-to-one mapping constraint for higher-order motion models

Mesh-based motion estimation implicitly includes a strong regularization required to preserve mesh connectivity (i.e., preserve node topology). This can result in excessively smooth motion fields and reduced coding performance. In the future, investigation of a possible relaxation of one-to-one mapping constraint (Section 2.2.2), especially when new nodes in hierarchical variable-size mesh models are introduced, may lead to improved handling of occlusion/uncovered areas. Naturally, discontinuities in the motion field created in such a way should be properly addressed. In addition, new ad-hoc or optimized methods for motion inversion in such cases should be investigated.

9.2.4 Introduction of fine-granularity motion scalability to the mixture motion model

Our current implementation of the mixture motion model supports only a single-layer non-scalable motion representation per spatial-resolution level. In the future, our model

could be improved to support layered and rate-scalable motion with more than one motion layer at each of the spatial resolutions. Although the impact of lossy motion coding on the reconstruction error is highly nonlinear, promising results were reported (Secker and Taubman, 2004; Wu, 2005) using scalable motion. In addition, a significant contribution might be possible in the related rate-distortion optimization selection of the switching layer (spatial layer at which motion model changes, as detailed in Chapter 8).

9.2.5 Joint coding of motion from different temporal resolutions

In order to maximize the compression performance, video coders based on wavelets rely on several levels of motion-compensated temporal filtering. Typically 3 to 5 decomposition levels are used, depending on motion activity in the scene and performance of motion estimation block (e.g., the number of unconnected pixels can be used to control further temporal decompositions). The majority of current video encoders independently compress motion fields from different temporal resolutions, despite an obvious redundance (e.g., motion fields from higher frame rates can be combined to predict slower framerate motion).

9.2.6 Further investigation of scalable coding architectures

The two structures analyzed in Chapter 8, "t+2D" and "2D+t+2D", present the two ends of the spectrum. The "t+2D" scheme maximizes energy compaction and compression efficiency while the "2D+t+2D" structure guarantees "artifact-free" low-resolution reconstruction. One of the interesting opportunities for future research might the design of a system that would balance this tradeoff by providing a solution with close-to-maximum coding gain and significantly reduced potential for low-resolution visual artifacts, even in the case when motion model fails. As we discussed in Chapter 8, in the "t+2D" scheme, all spatial frequencies are MCT filtered using the high-resolution motion, while in the "2D+t+2D" scheme each spatial subband is MCT filtered independently. An architecture combining these two approaches might be able to optimize both visual and objective performance across all spatial scales.

9.2.7 Real-time implementation

Significant portions of our motion estimation algorithms (e.g., spline-based motion estimation) were originally developed in MATLAB; for better performance, improved and optimized port to C is required, and it is mandatory for an integration with the MPEG's reference software. Although computational power of modern video devices continues to rapidly increase, proliferation of portable and low-power devices extends the demand for fast and efficient coder implementation.

References

Ahmed, N., Natarajan, T., and Rao, K. R. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, C-23:90–93.

Altunbasak, Y., Mersereau, R., and Patti, A. (2003). A fast parametric motion estimation algorithm with illumination and lens distortion correction. *IEEE Transactions on Image Processing*, 12(4):395–408.

André, T., Cagnazzo, M., Antonini, M., Barlaud, M., Božinović, N., and Konrad, J. (2004). (N,0) motion-compensated lifting-based wavelet transform. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume III, pages 121–124.

Andreopoulos, Y., Schaar, M. V. D., Munteanu, A., Barbarien, J., Schelkens, P., and Cornelis, J. (2003). Complete-to-overcomplete discrete wavelet transforms for fully-scalable video coding with MCTF. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5150, pages 719–731.

Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I. (1992). Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220.

AVC/H.264 (2004). Reference software: http://iphome.hhi.de/suehring/tml/.

Bayer, R. (1971). Binary B-trees for virtual memory. *ACM-SIGFIDET Workshop 1971*, pages 219–235.

Bjontegaard, G. (2001). Calculation of average PSNR differences between RD-curves. *Document VCEG-M33, VCEG 13th meeting, Austin, TX, USA*.

Božinović, N., Konrad, J., André, T., Antonini, M., and Barlaud, M. (2004). Motion-compensated lifted wavelet video coding: toward optimal motion/transform configuration. In *Signal Processing XII: Theories and Applications (Proceedings of the Twelfth European Signal Processing Conference)*, pages 1975–1978.

Brusewitz, H. (1990). Motion compensation with triangles. In *Proceedings of 3rd International Conference on 64-kbit Coding of Moving Video*.

Chahine, M. and Konrad, J. (1995). Estimation and compensation of accelerated motion for temporal sequence interpolation. *Signal Processing, Image Communication*, 7(4–6):503–527.

Chan, Y.-L. and Siu, W.-C. (1997). Variable temporal-length 3-D discrete cosine transform coding. *IEEE Transactions on Image Processing*, 6:758–763.

Chen, P. and Woods, J. W. (2002). Improved MC-EZBC with quarter-pixel motion vectors. *MPEG document, MPEG2002/M8366*.

Chen, P. and Woods, J. W. (2004). Bidirectional MC-EZBC with lifting implementation. *IEEE Transactions on Circuits and Systems for Video Technology*, 14:1183–1194.

Choi, S.-J. and Woods, J. (1999). Motion-compensated 3-D subband coding of video. *IEEE Transactions on Image Processing*, 8(2):155–167.

Clark, R. J. (1985). *Transform Coding of Images*. Academic Press, Orlando, Florida.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill.

Daubechies, I. and Sweldens, W. (1998). Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, 4(3):247–269.

Dubois, E. (1985). The sampling and reconstruction of time-varying imagery with application in video systems. *Proceedings of the IEEE*, 73(4):502–522.

Feng, B., Xu, J., Wu, F., and Yang, S. (2004). Energy distributed update steps (EDU) in lifting based motion compensated video coding. In *Proceedings of the IEEE International Conference on Image Processing*, volume 4, pages 2267–2270.

Flierl, M. and Girod, B. (2003). Generalized B pictures and the draft H.264/AVC video-compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13:587–597.

Flierl, M. and Girod, B. (2004). Video coding with motion-compensated lifted wavelet transforms. *Signal Processing, Image Communication*, 19:561–575.

Flierl, M., Wiegand, T., and Girod, B. (2002). Rate-constrained multihypothesis prediction for motion compensated video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 12:957–969.

Furht, B., Gustafson, K., Hesong, H., and Marques, O. (2003). An adaptive three-dimensional DCT compression based on motion analysis. In *Proceedings of the ACM Symposium on Applied Computing*, pages 765–768.

Gall, D. L. and Tabatabai, A. B. (1988). Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques video compression. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 761–764.

Girod, B. and Han, S. (2005). Optimal update for motion-compensated lifting. *IEEE Signal Processing Letters*, 12:150–153.

Glassner, A. (1995). *Principles of Digital Image Synthesis, 1st edition.* Morgan Kaufmann, San Francisco, CA.

Golwelkar, A. (2004). *Motion compensated temporal filtering and motion vector coding using longer filters*. PhD thesis, Rensselaer Polytechnic Institute, Electrical, Computer, and Systems Engineering Deptartment.

Golwelkar, A. and Woods, J. (2003). Scalable video compression using longer motion compensated temporal filters. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5150, pages 1406–1416.

Heeger, D. (1988). Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1:279–302.

Hsiang, S.-T. and Woods, J. (1999). Invertible three-dimensional analysis/synthesis system for video coding with half-pixel-accurate motion compensation. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 3653, pages 537–546.

Hsiang, S.-T., Woods, J., and Ohm, J.-R. (2004). Invertible temporal subband/wavelet filter banks with half-pixel-accurate motion compensation. *IEEE Transactions on Image Processing*, 13:1018–1028.

Hsiang, S.-T. and Woods, J. W. (2000). Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 662–665.

Ince, S. and Konrad, J. (2005). Geometry-based estimation of occlusions from video frame pairs. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume II, pages 933–936.

ISO/IEC JTC1 IS 13818-2 (MPEG-2) (1994). Information Technology - Generic Coding of Moving Pictures and Associated Audio.

Izquierdo, E. (1997). Stereo matching for enhanced telepresence in three-dimensional videocommunications. *IEEE Transactions on Circuits and Systems for Video*

Technology, 7(4):629–643.

Jacobson, L. and Wechsler, H. (1987). Derivation of optical flow using a spatiotemporal-frequency approach. *Computer Vision, Graphics, and Image Processing*, 38:29–65.

JPEG2000 (2000). Information technology JPEG 2000 image coding system - part 1: Core coding system. ISO/IEC 15444-1:2000.

Karlsson, G. and Vetterli, M. (1988). Three-dimensional subband coding of video. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1100–1103.

Kim, B. J. and Pearlman, W. A. (1997). An embeded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT). In *Proceedings IEEE Data Compression Conference*, pages 251–260.

Kim, B. J., Xiong, Z., and Pearlman, W. A. (2000). 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*, pages 1374–1387.

Konrad, J. (2004). Transversal versus lifting approach to motion-compensated temporal discrete wavelet transform of image sequences: equivalence and trade-offs. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5308, pages 452–463.

Lee, G., Song, J., and Park, R.-H. (1997a). Three-dimensional DCT/WT compression using motion vector segmentation for low bit-rate video coding. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 456–459.

Lee, M., Chan, R., and Adjeroh, D. (1997b). Quantization of 3D-DCT coefficients and scan order for video compression. *Journal of Visual Communication and*

Image Representation., 8(4):405–422.

Luo, L., Li, J., Li, S., and Zhang, Y.-Q. (2001). Motion compensated lifting wavelet and its application in video coding. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 481–484.

Luo, L., Wu, F., Li, S., and Zhuang, Z. (2003). Advanced lifting-based motion-threading (MT) technique for 3D wavelet video coding. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5150, pages 707–718.

Marpe, D. and Cycon, H. (1999). Very low bit-rate video coding using wavelet-based techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 9:85–94.

Marpe, D., Schwarz, H., and Wiegand, T. (2003). Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13:620–636.

Mehrseresht, N. and Taubman, D. (2006). A flexible structure for fully scalable motion-compensated 3-D DWT with emphasis on the impact of spatial scalability. *IEEE Transactions on Image Processing*, 15:740–753.

Moulin, P., Krishnamurthy, R., and Woods, J. (1997). Multiscale modeling and estimation of motion fields for video coding. *IEEE Transactions on Image Processing*, 6(12):1606–1620.

Nakaya, Y. and Harashima, H. (1994). Motion compensation based on spatial transformations. *IEEE Transactions on Circuits and Systems for Video Technology*, 4:339–356.

Natarajan, T. and Ahmed, N. (1977). On interframe transform coding. *IEEE Transactions on Communications*, 25:1323–1329.

Ohm, J. (1994). Three-dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, 3(5):559–571.

Ohm, J. (2002). Motion-compensated wavelet lifting filters with flexible adaptation. In *Proceedings of International Workshop on Digital Communications*, pages 113–120.

Oppenheim, A., Schafer, R., and Buck, J. (1999). *Discrete-Time Signal Processing*. Prentice-Hall.

Pau, G., Tillier, C., and Pesquet-Popescu, B. (2004). Optimization of the predict operator in lifting-based motion compensated temporal filtering. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5308, pages 712–720.

Pearson, D. (1975). *Transmission and Display of Pictorial Information*. John Wiley & Sons.

Pesquet-Popescu, B. and Bottreau, V. (2001). Three-dimensional lifting schemes for motion compensated video compression. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1793–1796.

Pitas, I. (2000). *Digital Image Processing Algorithms and Applications*. John Wiley & Sons.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical recipes in C: The art of scientific computing*. Cambridge University Press, 2-nd edition.

Roes, J., Prat, W., and Robinson, G. (1977). Interframe cosine transform image coding. *IEEE Transactions on Communications*, 25:1329–1338.

Rusert, T., Hanke, K., and Wien, M. (2004). Optimization for locally adaptive MCTF based on 5/3 lifting. In *Procedings Picture Coding Symposium*.

Said, A. and Pearlman, W. (1996). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250.

Schafer, R., Wiegand, T., and Schwarz, H. (2003). The emerging H.264/AVC standard. *EBU Technical Review*.

Secker, A. (2004). *Motion-adaptive transforms for highly scalable video compression*. PhD thesis, University of New South Wales, School of Electrical Engineering and Telecommunications.

Secker, A. and Taubman, D. (2001). Motion-compensated highly scalable video compression using an adaptive 3D wavelet transform based on lifting. In *Proceedings of the IEEE International Conference on Image Processing*, pages 1029–1032.

Secker, A. and Taubman, D. (2003). Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression. *IEEE Transactions on Image Processing*, 12(12):1530–1542.

Secker, A. and Taubman, D. (2004). Highly scalable video compression with scalable motion coding. *IEEE Transactions on Image Processing*, 13:1029–1041.

Shapiro, J. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462.

Sweldens, W. (1996). The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186–200.

Szeliski, R. and Coughlan, I. (1994a). Hierarchical spline-based image registration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 194–201.

Szeliski, R. and Coughlan, J. (1994b). Spline-based image registration. Technical Report CRL 94/1, Digital Equipment Corporation, Cambridge Research Lab.

Szeliski, R. and Shum, H.-Y. (1996). Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1199–1210.

Taubman, D. (2000). High performance scalable image compression with EBCOT. *IEEE Transactions on Image Processing*, 9:1158–1170.

Taubman, D. and Zakhor, A. (1994). Multirate 3-D subband coding of video. *IEEE Transactions on Image Processing*, 3(5):572–588.

Taubman, D. S. and Marcellin, M. W. (2002). *JPEG2000, Image compression fundamental, standards and practice.* Kluwer academic Publishers Group.

Thomas, G. (1987). Television motion measurement for DATV and other applications. Technical Report 11, BBC Research Deptartment.

Tillier, C. and Pesquet-Popescu, B. (2004). A new 3-band MCTF scheme for scalable video coding. *Proc. of Picture Coding Symposium (PCS).*

Tillier, C., Pesquet-Popescu, B., and van der Schaar, M. (2004). Improved update operators for lifting-based motion-compensated temporal filtering. *IEEE Signal Processing Letters*, 12:146–149.

Toklu, C., Erdem, A., Sezan, M., and Tekalp, A. (1996). Tracking motion and intensity variations using hierarchical 2-D mesh modeling for synthetic object transfiguration. *Graphical Models and Image Processing*, 58:553–573.

Unser, M. (1999). Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38.

Valentin, V., Cagnazzo, M., Antonini, M., and Barlaud, M. (2003). Scalable context-based motion vector coding for video compression. In *IEEE Picture Coding Symposium*, pages 63–68.

Vázquez, C., Dubois, E., and Konrad, J. (2005). Reconstruction of irregularly-sampled images in spline spaces. *IEEE Transactions on Image Processing*, 14(6):713–725.

Vetterli, M. and Kovacevic, J. (1995). *Wavelets and subband coding*. Prentice Hall, Englewood Cliffs, NJ, USA.

Wang, A., Xiong, Z., Chou, P., and Mehrotra, S. (1999). Three-dimensional wavelet coding of video with global motion compensation. In *Proceedings Data Compression Conference*, pages 404–413.

Wang, Y., Ostermann, J., and Zhang, Y. (2002). *Video Processing and Communications*. Prentice Hall.

Wang, Z. and Hunt, B. (1985). The discrete W transform. *Applied Mathematics and Computation*, 16:19–48.

Westwater, R. and Furht, B. (1996). Three-dimensional dct video compression technique based on adaptive quantizers. In *Second IEEE International Conference on Engineering of Complex Computer Systems, Montreal, Canada*.

Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576.

Wu, Y. (2005). *Fully scalable subband/wavelet video coding system*. PhD thesis, Rensselaer Polytechnic Institute, Electrical, Computer, and Systems Engineering Deptartment.

Xiong, R., Wu, F., Xu, J., Li, S., and Zhang, Y.-Q. (2004). Barbell lifting wavelet transform for highly scalable video coding. In *Proceedings Picture Coding Symposium*.

Xiong, R., Xu, J., Wu, F., and Li, S. (2005a). Optimal subband rate allocation for spatial scalability in 3D wavelet video coding with motion aligned temporal filtering. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5960, pages 381–392.

Xiong, R., Xu, J., Wu, F., and Li, S. (2005b). Studies on spatial scalable frameworks for motion aligned 3D wavelet video coding. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5960, pages 189–200.

Xiong, R. Q., Ji, X. Y., Zhang, D. D., Xu, J. Z., Pau, G., Trocan, M., and Bottreau, V. (2005c). Vidwav wavelet video coding specifications. *(ISO/IEC) ISO/IEC JTC1/SC29/WG11 Document M12339*.

Xu, J., abd S. Li, Z. X., and Zhang, Y.-Q. (2002). Memory-constrained 3-D wavelet transform for video coding without boundary effects. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 812–818.

Xu, J., Xiong, Z., Li, S., and Zhang, Y.-Q. (2001). Three-dimensional embedded subband coding with optimized truncation (3-D ESCOT). *Applied and Computational Harmonic Analysis: Special Issue on Wavelet Applications*, 10.

Yeo, B. and Liu, B. (1995). Volume rendering of DCT-based compressed 3D scalar data. *IEEE Transactions On Visualization And Computer Graphics*, 1(1):29–43.

Zhang, Y.-Q. and Zafar, S. (1992). Motion-compensated wavelet transform coding for color video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2:285–296.

Zhao, W. (2004). Motion compensation in temporal discrete wavelet transforms. Technical Report 2004-04, Boston University, Department of Electrical and Computer Engineering.

CURRICULUM VITAE

Nikola Božinović

39 Bay State Rd., Apt. 4F, Boston, MA 02215

nikolab@bu.edu

http://iss.bu.edu/nikolab

Vita

Birth year:	1974.
Birthplace:	Niš, Serbia, Yugoslavia.

Education

Current	**Ph.D. in Electrical and Computer Engineering**, Boston University (GPA: 4.0/4.0). Advisor: Prof. Janusz Konrad; expected graduation: May 2006.
2001	**M.S. in Electrical and Computer Engineering**, Boston University (GPA: 3.97/4.00).
2000	**Dipl.Ing. in Electrical Engineering**, School of Electrical Engineering, University of Niš, Serbia, Yugoslavia (GPA: 9.5/10.0).

Research and work experience

May 2001 -	**Research Assistant**, *Visual Information Processing Laboratory,* Department of Electrical and Computer Engineering, Boston University. Advisor Prof. Janusz Konrad.

Summer 2005	**Visiting Student**, *Microsoft Research Asia*, Beijing. Host Dr. Feng Wu.
Summer 2003, 2004	**Visiting Researcher**, *University of Nice,* Sophia Antipolis, France. Host Prof. Michael Barlaud.
1998 - 2000	**Head of Astronomy Department**, *Petnica Science Center*, Serbia, Yugoslavia.
Fall 1997	**System Engineer**, *DPO*, Muak Lek, Thailand

Honors and Awards

2005	**National Science Foundation**, EAPSI Summer Fellowship
2002 – 2004	**Studenica Foundation/DNCT Fellowship**
2000	**Dean's Fellowship**,*College of Engineering*, Boston University
1995 – 1998	**City of Niš Academic Fellowship**, Serbia, Yugoslavia.
1994	**Ranked** 1^{st} out of more than 3,000 at the Joint Entrance Exam, University of Belgrade

Public Service and Leadership

2000 – 2002	**Treasurer**, Student Association of Graduate Engineers (SAGE), Boston University
1998 – 1999	**President of the Student Union**, University of Niš
1996 – 1997	**Leader and spokesperson**, Student protest against S. Milošević, Serbia
1995 – 1998	**Founder and first officer**, Astronomical Society "Alpha", Nis

Publications

Journal Papers

[1] N. Božinović and J. Konrad. **Motion analysis in 3D DCT domain and its application to video coding.** *Signal Processing, Image Communication*, pages 510–528, Jul 2005.

Conference Papers

[1] N. Božinović and J. Konrad. **Modeling motion for spatial scalability.** *Proceedings IEEE International Conference on Accoustic, Speech, and Signal Processing*, May 2006.

[2] N. Božinović and J. Konrad. **On the importance of motion invertibility in MCTF/DWT video coding.** *Proceedings IEEE International Conference on Accoustic, Speech, and Signal Processing*, vol. II, pp. 49-52, March 2005.

[3] J. Konrad and N. Božinović. **Importance of motion in motion-compensated temporal discrete wavelet transforms.** *Proceedings of SPIE Visual Communications and Image Processing*, vol. 5685, pp. 354-365, Jan. 2005.

[4] N. Božinović, J. Konrad, T. Andrè, M. Antonini, and M. Barlaud. **Motion-compensated lifted wavelet video coding: toward optimal motion/transform configuration.** *in Signal Process. XII: Theories and Applications (Proceedings Twelfth European Signal Processing Conference*, pp 1975-1978, Sep. 2004.

[5] N. Božinović and J. Konrad. **Mesh-based motion models for wavelet video coding.** *Proceedings IEEE International Conference on Accoustic, Speech, and Signal Processing*, vol. III, pp. 141-144, May 2004.

[6] T. Andrè, M. Cagnazzo, M. Antonini, M. Barlaud, N. Božinović, and J. Konrad. **(N,0) motion-compensated lifting-based wavelet transform.** *Proceedings IEEE International Conference on Accoustic, Speech, and Signal Processing*, vol. III, pp. 121-124, May 2004.

[7] N. Božinović and J. Konrad. **Scan order and quantization for 3D-DCT coding.** *Proceedings of SPIE Visual Communications and Image Processing*, vol. 5150, pp. 1204-1215, July 2003.

[8] J. Konrad and N. Božinović. **Interpretation of uniform translational image motion: DCT versus FT.** *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 281-284, Sept. 2002.

Selected Invited Talks

Mar. 2006	*Ricoh Innovations,* Menlo Park (California Research Center)
Nov. 2005	*Mitsubishi Electric Research Labs,* Cambridge (MERL Technology Lab)
Aug. 2005	*Microsoft Research Asia,* Beijing (Internet Media Group)
Jan. 2005	*UC Berkeley,* (Berkeley Audio Visual Signal Processing and Communication Systems)
Sep. 2004	*Rensselaer Polytechnic Institute,* Troy, (Center for Next Generation Video)
May 2004	*Stanford University,* (Information Systems Laboratory)
July 2003	*EPFL,* Lausanne, Switzerland (Signal Processing Institute)
June 2003	*University of Nice,* France (Team CReATIVe)