12-13-2003

Fully Scalable Video Coding Using Redundant-Wavelet Multihypothesis and Motion-Compensated Temporal Filtering

Yonghui Wang

FULLY SCALABLE VIDEO CODING USING REDUNDANT-WAVELET MULTIHYPOTHESIS AND MOTION-COMPENSATED TEMPORAL FILTERING

By

Yonghui Wang

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2003

FULLY SCALABLE VIDEO CODING USING REDUNDANT-WAVELET MULTIHYPOTHESIS AND MOTION-COMPENSATED TEMPORAL FILTERING

By

Yonghui Wang

Approved:

James E. Fowler
Associate Professor of Electrical and Computer Engineering
(Major Professor and Dissertation Director)

Robert J. Moorhead, II
Professor of Electrical and Computer Engineering
(Committee Member)

Nicholas H. Younan
Professor of Electrical and Computer Engineering
Graduate Coordinator of Electrical and Computer Engineering
(Committee Member)

Jörg Meyer
Adjunct Assistant Professor of Computer Science and Engineering
(Committee Member)

A. Wayne Bennett
Dean of the College of Engineering

Name: Yonghui Wang

Date of Degree: December 13, 2003

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Dr. James E. Fowler

Title of Study: FULLY SCALABLE VIDEO CODING USING REDUNDANT-WAVELET MULTIHYPOTHESIS AND MOTION-COMPENSATED TEMPORAL FILTERING

Pages in Study: 122

Candidate for Degree of Doctor of Philosophy

In this dissertation, a fully scalable video coding system is proposed. This system achieves full temporal, resolution, and fidelity scalability by combining mesh-based motion-compensated temporal filtering, multihypothesis motion compensation, and an embedded 3D wavelet-coefficient coder. The first major contribution of this work is the introduction of the redundant-wavelet multihypothesis paradigm into motion-compensated temporal filtering, which is achieved by deploying temporal filtering in the domain of a spatially redundant wavelet transform. A regular triangle mesh is used to track motion between frames, and an affine transform between mesh triangles implements motion compensation within a lifting-based temporal transform. Experimental results reveal that the incorporation of redundant-wavelet multihypothesis into mesh-based motion-compensated temporal filtering significantly improves the rate-distortion performance of the scalable coder. The second major contribution is the introduction of a sliding-window implementation of motion-compensated temporal filtering such that video sequences of arbitrary length may be temporally filtered

using a finite-length frame buffer without suffering from severe degradation at buffer boundaries. Finally, as a third major contribution, a novel 3D coder is designed for the coding of the 3D volume of coefficients resulting from the redundant-wavelet based temporal filtering. This coder employs an explicit estimate of the probability of coefficient significance to drive a nonadaptive arithmetic coder, resulting in a simple software implementation. Additionally, the coder offers the possibility of a high degree of vectorization particularly well suited to the data-parallel capabilities of modern general-purpose processors or customized hardware. Results show that the proposed coder yields nearly the same rate-distortion performance as a more complicated state-of-the-art coefficient coder.

DEDICATION

To my parents, Yunde and Guihua, my wife, Suxia, and my son, Steven.

ACKNOWLEDGMENTS

This dissertation could not have been completed without the contributions and encouragement from many people. I am gratefully indebted to my advisor Dr. James E. Fowler, who was gracious to work with me through some difficult times and was definitely the first person I consulted whenever I was stuck. His attitude toward scientific research made him an excellent role model. I thank also all those who served on my dissertation committee, Dr. Robert J. Moorhead, Dr. Nicolas H. Younan, and Dr. Jörg Meyer, for their encouragement and support. Thanks also go to my fellow students, Dr. Li Hua and Justin T. Rucker, for their selfless help. I am grateful for financial support from the National Science Foundation (NSF), as well as the research facilities provided by the Engineering Research Center at Mississippi State University. Finally, I would like to thank all the members of my family, especially my wife, Suxia, for her love and support as well as those fruitful discussions toward the completion of this dissertation. I thank my son, Steven, for all the happy time he gives.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Video is one of the most powerful forms of multimedia because of the extensive information it delivers. Each video sequence contains substantial visual information, thereby requiring vast resources for storage and communication. Therefore, the compression of video sequences has been the focus of work by many researchers for several decades. Video sequences are highly correlated both temporally and spatially, a fact which makes the compression of video possible. Regarding temporal correlation, because the temporal interval between every two consecutive video frames is very small, the two frames exhibit high similarity. This fact can be seen in Figs. 1.1-1.4. Figs. 1.1 and 1.3 depict slow-motion sequences in which every frame is almost identical to its predecessor. However, even in fast- or complex-motion sequences, such as "Football" in Fig. 1.2 and "Coastguard" in Fig. 1.4, there are large regions which are nearly identical from frame to frame.

To decorrelate a video sequence temporally, modern video-coding systems use motion estimation and motion compensation (ME/MC). Traditional video coding systems, such as H.263 [1, 2] and MPEG-2 [3], use a feedback loop for ME/MC. That is, the system predicts the current frame from a frame that is already encoded, the residual image is transformed and coded, and a set of motion vectors that describe the temporal prediction are delivered to the decoder. Usually, the residual image contains much less energy than the original image, so that the residual image can be coded with significantly fewer bits. However, there is a well known problem with this kind of system structure. Since a closed feedback loop is used, the decoder must assemble its

Figure 1.1: First 8 frames of the “Susie” sequence.

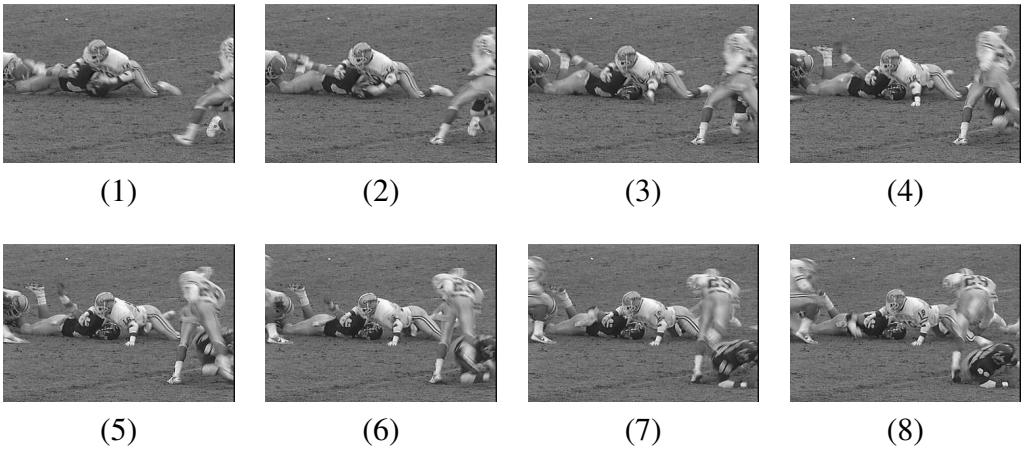


Figure 1.2: First 8 frames of the “Football” sequence.

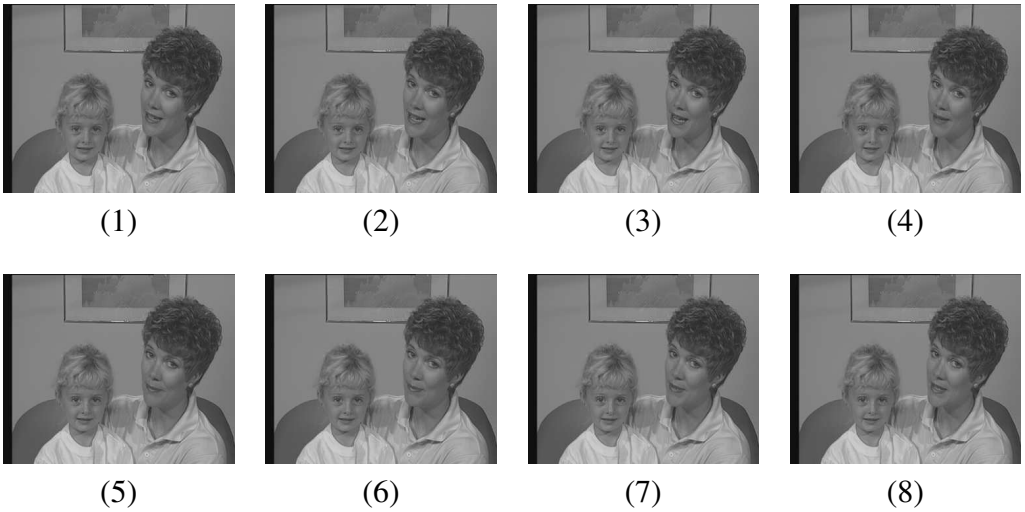


Figure 1.3: First 8 frames of the “Mother & Daughter” sequence.

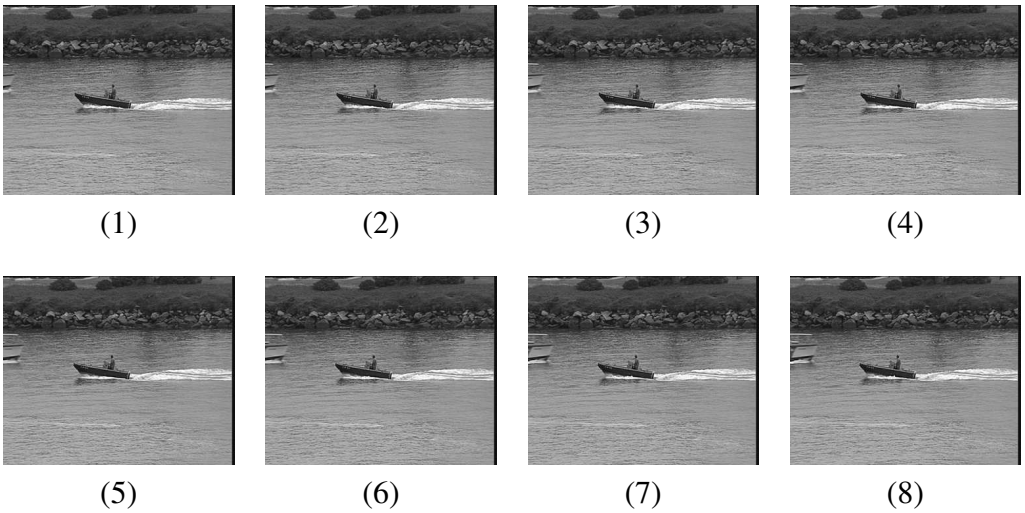


Figure 1.4: First 8 frames of the “Coastguard” sequence.

prediction using the same previously encoded frame, or reference frame, as was used by the encoder. If this reference frame is not available, or only partially available, to the decoder, as may be the case in the event that some communication failure occurred, then the decoder will not be able to correctly decode the current frame. Since the current frame may be subsequently used as a reference for later frames, the decoder may become missynchronized from the encoder, suffering what is known as decoder "drift." Drift often results in seriously degraded video being reconstructed by the decoder and poses an issue of prime importance in many applications, such as the real-time delivery of video over wireless channels which are inherently prone to communication failures.

Drift in predictive-loop coders can also occur in other settings. Of particular current interest is the delivery of video over heterogeneous networks consisting of many users connected to the network via links of varying bandwidth and transmission-error rate. Rather than producing a separate encoding of the video sequence for the specific capabilities of each user's communication link, it is desired to support all possible link capabilities with one single encoding. *Fully scalable* video coding is thus demanded in this case. Scalable video refers to any method that allows partial decoding of the bitstream. According to network conditions, the decoder can receive a scalable bitstream and reconstruct video at a variety of quality levels, spatial resolutions, or temporal resolutions. The degree of scalability offered by the system can vary. Systems that offer a large number of possible qualities or resolutions are said to be highly scalable. A fully scalable video-coding system provides a high degree of fidelity, spatial, and temporal scalability and offers the most flexibility for accommodating diverse network capabilities.

It has been generally recognized that the goal of highly scalable video representation is fundamentally at odds with the traditional ME/MC feedback loop which hinders the achieving of a high degree of resolution, temporal, and fidelity scalability. This is due to

the fact that scalable coding introduces drift into predictive-loop coders. For example, if a decoder is producing video at a reduced spatial resolution, say, at $1/4$ of the original frame size, then the reference frames used in the ME/MC feedback loop are available at only this reduced resolution, rather than at the full resolution that was used by the encoder when initially producing the compressed bitstream. Thus, drift is inherent to any predictive-loop decoder when producing a scaled video representation.

Consequently, the use of 3D transforms, which break the ME/MC feedback loop, are becoming the preferred approach to full scalability, and a number of modern 2D still-image algorithms have been straightforwardly extended to the third dimension (e.g., 3D-SPIHT [4]) by employing separable 3D wavelet transforms. This approach usually involves a wavelet-packet subband decomposition wherein a group of frames is processed with a temporal transform followed by spatial decomposition of each frame. However, without MC, temporal transforms produce low-quality temporal subbands with significant "ghosting" artifacts [5] and decreased coding efficiency. Consequently, there has been significant interest in motion-compensated temporal filtering (MCTF) [5–17] in which it is attempted to have the temporal transform follow motion trajectories.

The work described in this dissertation consists of three major contributions. The first of these is the development of a novel approach to performing MCTF such that temporal filtering takes place within frames that have been already decomposed with a spatial wavelet transform. Experimental results reveal that there is some advantage towards increased scalability over the usual approach in which temporal filtering precedes the spatial transform. However, the salient aspect of this approach to MCTF lies in that we employ multihypothesis motion compensation (MHMC) within the MCTF to combat the uncertainty inherent in estimating motion trajectories for MCTF, thereby achieving rate-distortion performance significantly superior to the usual single-hypothesis MCTF approach. Although multihypothesis has been used

in conjunction with MCTF before (e.g., [12] and [13] propose both spatially and temporally diverse multihypothesis MCTF predictions), in our proposed system, we employ a new class of MHMC—phase-diversity multihypothesis [18, 19]. Specifically, phase-diversity MHMC is implemented by deploying MCTF in the domain of a spatially redundant wavelet transform such that multiple hypothesis temporal filterings are combined implicitly in the form of an inverse transform. While the overwhelming majority of previous MCTF techniques have deployed MCTF in the spatial domain, a few approaches (e.g., [12, 13]) have used the shift invariance of spatially redundant transforms to enable wavelet-domain MCTF. In contrast to these techniques, our redundant-wavelet-multihypothesis (RWMH) approach to MCTF exploits the transform redundancy not only for its shift invariance, but for, more importantly, its potential for superior motion prediction via phase-diversity multihypothesis.

In any practical implementation of temporal filtering, one must be able to process arbitrarily long sequences of frames. The usual approach involves blocking some finite number of frames together into a group of frames (GOF) and restricting temporal filtering to within each GOF. Unfortunately, this approach leads to undesired performance degradation in frames near the GOF boundaries. As the second contribution of this dissertation, we adapt a known solution [15, 20] to this "boundary-effect" problem to the situation in which the temporal filtering is motion compensated.

The method for transforming a video sequence into wavelet coefficients via MCTF is only the first stage in a 3D video-coding system, as the 3D volume of wavelet coefficients must be fed to a 3D coder which will encode the coefficients into a bitstream. It is desired that a 3D coder provide not only efficient rate-distortion performance, but be easily implemented in both software and hardware. As the third major contribution of this dissertation, a novel 3D coder, 3D-Tarp, is proposed. The 3D-Tarp coder explicitly estimates the probability that wavelet coefficients have magnitude greater than a given

threshold, or are "significant" with respect to the threshold, and uses this estimated probability to drive a nonadaptive arithmetic coder. By exploiting the well known Laplacian distribution nature of wavelet-coefficient magnitudes, the 3D-Tarp coder achieves an exceedingly precise estimate of the probability of significance, resulting in efficient coding of the significance information. The 3D-Tarp algorithm is also quite simple, and therefore, easily implemented. Moreover, it can be highly vectorized for implementation in single-instruction-multiple-data (SIMD) architectures.

In the remainder of this dissertation, we describe, in detail, our approach to fully scalable video coding. The discussion is organized as follows. In Chap. II, the relevant literature concerning MCTF is reviewed. Subsequently, redundant wavelet transforms and how they have been used previously for video coding are described in Chap. III. In Chap. IV, we present the first major contribution of the dissertation, our RWMH approach to MCTF which deploys temporal filtering within a redundant wavelet transform. We improve practical performance of this RWMH MCTF approach in Chap. V by eliminating poor performance characteristic to MCTF when deployed in buffers holding a finite number of frames. This sliding-window approach to MCTF is the second major contribution of this dissertation. In Chap. VI, the third major contribution of this dissertation, the 3D-Tarp coder, is presented. Then, finally, in Chap. VII, some concluding remarks are made.

CHAPTER II

MOTION-COMPENSATED TEMPORAL FILTERING

The concept of coding video by grouping several frames together into a 3D volume and employing transforms in the spatial and temporal directions has been explored on and off in the literature for the past several decades. However, temporal transforms for video pose a unique problem that causes 3D video coding to be different from the coding of other 3D data types, such as volumetric medical imagery or multispectral/hyperspectral remotely-sensed imagery. Specifically, motion of an object in time can produce high-frequency coefficients in the temporal transform, even if the object does not vary in shape or gray-level intensity over the temporal interval. Consequently, researchers have, in recent times, sought temporal transforms that track object trajectories so that these transforms successfully temporally decorrelate object pixels regardless of the motion they undergo. Such temporal transforms are called motion-compensated temporal filtering (MCTF).

Two families of approaches to MCTF have arisen thus far. The first family of algorithms combines traditional block-based MC with temporal filtering [8–11, 16, 17]; we refer to these as block-displacement methods. The second family of algorithms uses a lifting realization of the temporal DWT with MC applied along the lifting steps, as proposed by Secker and Taubman in [5, 6]. In this chapter, we describe these two approaches in detail.

2.1 Block-Displacement Methods

First proposed by Ohm [8, 16], then further developed by Woods *et al.* [9, 10], the block-displacement methods have a long history in the area of 3D subband video coding, naturally providing the spatial-resolution as well as frame-rate scalability which are increasingly expected of modern multimedia applications. Although a popular approach to MCTF, block-displacement techniques have traditionally encountered a number of drawbacks. First, the rigid block-motion model fails to capture all aspects of the motion field, leaving significant numbers of pixels "unconnected" between frames; these unconnected pixels are coded separately to the detriment of coding efficiency. Second, it is difficult to achieve sub-pixel accuracy in the MCTF while maintaining invertibility of the temporal transform. Finally, implementation of temporal filters, other than the simple Haar filter, is hindered by the numerous unconnected pixels.

In block-displacement methods, the process of temporal filtering uses block matching. That is, video frames are divided into blocks, and motion vectors of the blocks in the current frame point to the closest matching blocks in the reference frame. If there is no motion, or only pure translational motion, the motion vectors provide a one-to-one mapping between pixels in the reference frame and pixels in the current frame. This one-to-one mapping between frames then provides the trajectory for filtering in the temporal direction for MCTF. However, in more realistic video sequences, motion is usually much more complex, yielding one-to-many mappings for some pixels in the reference frame and no mapping for others, as illustrated in Fig. 2.1. These latter pixels are thus "unconnected." Unconnected pixels can detrimentally affect both overall coding efficiency and subjective video quality since they cannot be directly included in MCTF. In fact, an essential component of any block-displacement method is some solution to the temporal filtering of unconnected pixels.

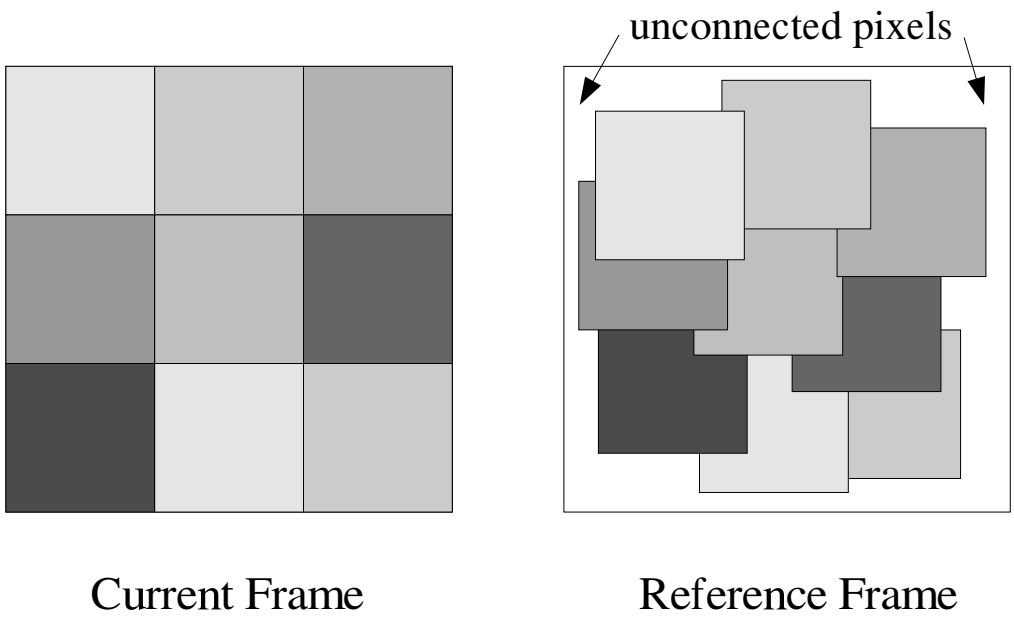


Figure 2.1: In motion estimation of block-displacement methods, blocks in the reference frame corresponding to those in the current frame typically overlap. Thus, some pixels in the reference frame are mapped several times into the current frame while other pixels have no mapping. These latter pixels are "unconnected".

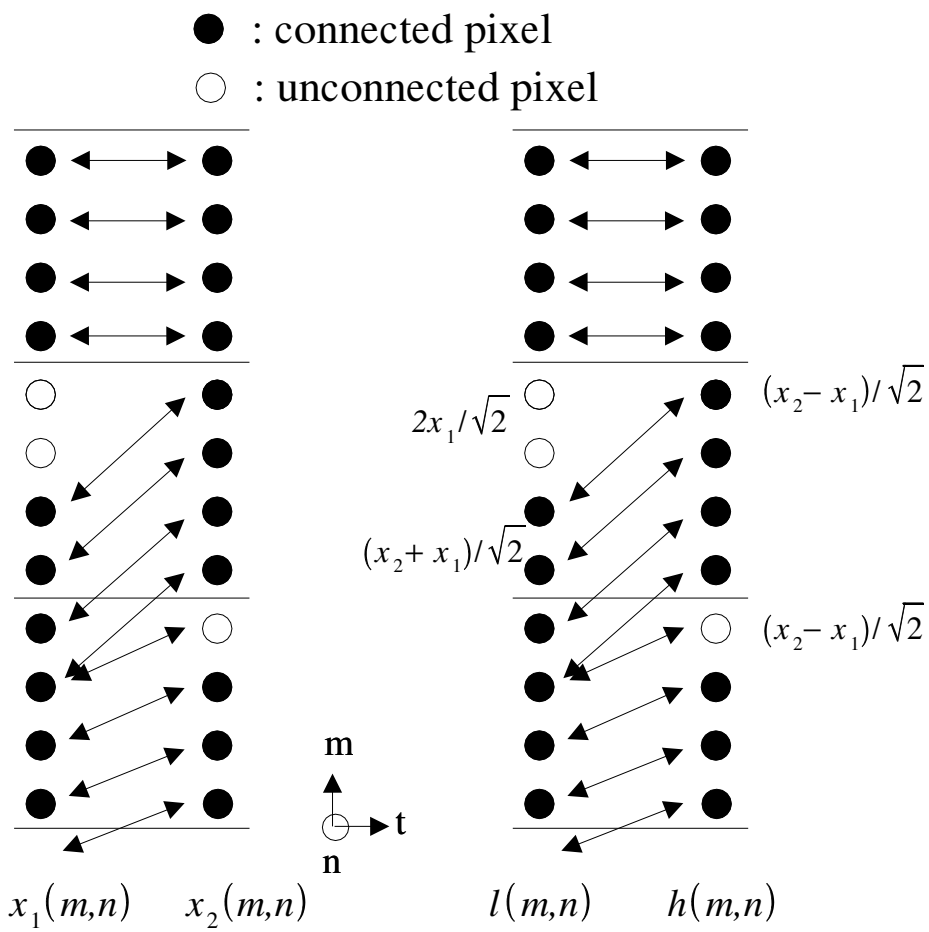


Figure 2.2: MCTF based on the Haar filter. $x_1(m,n)$, $x_2(m,n)$, $l(m,n)$, and $h(m,n)$ are reference, current, lowpass, and highpass frames, respectively. 2D frames are illustrated as 1D signals to facilitate visualization. Integer-pixel accurate MC is assumed. Adapted from [9].

Additionally, since filters with long impulse responses must contend with many consecutive frames, unconnected pixels make temporal filters other than Haar difficult to implement. Therefore, attention is usually focused exclusively on the Haar filter in these methods. In one scale of Haar decomposition, two frames (reference and current) are decomposed into one lowpass frame and one highpass frame. In [9], for an unconnected pixel in the reference frame, the original value is inserted into the same location of the temporal lowpass frame as illustrated in Fig. 2.2. This procedure can be summarized as follows. For connected pixels, MCTF with the Haar filter is employed as:

$$l(m-\bar{d}_m, n-\bar{d}_n) = \frac{1}{\sqrt{2}}\tilde{x}_2(m-\bar{d}_m+d_m, n-\bar{d}_n+d_n) + \frac{1}{\sqrt{2}}x_1(m-\bar{d}_m, n-\bar{d}_n), \tag{2.1}$$

$$h(m,n) = \frac{1}{\sqrt{2}}x_2(m,n) - \frac{1}{\sqrt{2}}\tilde{x}_1(m-d_m, n-d_n). \tag{2.2}$$

For unconnected pixels, an *ad hoc* approximation to Haar filtering is used:

$$l(m,n) = \frac{2x_1(m,n)}{\sqrt{2}}, \tag{2.3}$$

$$h(m,n) = \frac{1}{\sqrt{2}}x_2(m,n) - \frac{1}{\sqrt{2}}\tilde{x}_1(m-d_m, n-d_n), \tag{2.4}$$

where $l(m,n)$ and $h(m,n)$ are temporal lowpass and highpass frames, respectively; $x_1(m,n)$ is the reference frame, $x_2(m,n)$ is the current frame, and (d_m, d_n) is the motion vector. $\bar{d}_m$ and $\bar{d}_n$ are nearest integers to d_m and d_n, respectively, while $\tilde{x}_1$ and $\tilde{x}_2$ are interpolated values if motion vectors are sub-pixel accurate; otherwise, integer-pixel accuracy is assumed and $\bar{d}_m = d_m$, $\bar{d}_n = d_n$, $\tilde{x}_1 = x_1$, and $\tilde{x}_2 = x_2$.

The synthesis procedure can be derived from the above analysis equations. For connected pixels we have:

$$x_1(m-d_m, n-d_n) = \frac{1}{\sqrt{2}} l(m-d_m, n-d_n) - \frac{1}{\sqrt{2}} h(m,n), \tag{2.5}$$

$$x_2(m,n) = \frac{1}{\sqrt{2}} l(m-d_m, n-d_n) + \frac{1}{\sqrt{2}} h(m,n), \tag{2.6}$$

while for unconnected pixels we have:

$$x_1(m,n) = \frac{1}{\sqrt{2}} l(m,n), \tag{2.7}$$

$$x_2(m,n) = \frac{1}{\sqrt{2}} l(m-d_m, n-d_n) + \frac{1}{\sqrt{2}} h(m,n). \tag{2.8}$$

Since the temporal transform is invertible, the original frames can be perfectly reconstructed, only for integer-pixel-accurate MC wherein $\tilde{x}_1 = x_1$, $\tilde{x}_2 = x_2$, $d_m = \bar{d}_m$, and $d_n = \bar{d}_n$. However, for sub-pixel-accurate MC, because of the imperfect spatial interpolation (e.g. $\tilde{x}_1 \neq x_1$, $\tilde{x}_2 \neq x_2$), only lossy reconstruction can be achieved. Thus, the block-displacement methods suffer from a lack of temporal-transform invertibility when sub-pixel accuracy is employed.

Although recent advances have alleviated some of the ill effects of unconnected pixels in block-displacement techniques (i.e., resorting to a lifting implementation of the temporal transform greatly facilitates sub-pixel accuracy [6, 10, 11, 17], while bidirectional prediction improves the coding efficiency of unconnected pixels [10]), the block-based model remains limited in its ability to adequately capture motion for MCTF.

2.2 Lifting Implementation of Temporal Transforms

It has been recognized that a lifting implementation of the temporal transform permits MC schemes more general than block displacement to be implemented in an

easily inverted fashion [6]. For example, let $x_1(m,n)$ and $x_2(m,n)$ be two consecutive frames of a video sequence, and let $W_{i,j}$ denote the operator that maps frame i onto the coordinate system of frame j through the particular MC scheme of choice. Ideally, we would want $W_{1,2}\left[x_1\right](m,n) \approx x_2(m,n)$. Haar-based MCTF would then be implemented via lifting as:

$$h(m,n) = \frac{1}{2}\left(x_2(m,n) - W_{1,2}\left[x_1\right](m,n)\right), \tag{2.9}$$

$$l(m,n) = x_1(m,n) + W_{2,1}\left[h\right](m,n), \tag{2.10}$$

where $l(m,n)$ and $h(m,n)$ are the lowpass and highpass frames, respectively, of the temporal transform [6]. This formulation, which can be easily extended to more complicated filters, permits any motion model to be used since the lifting decomposition is easily inverted as:

$$x_1(m,n) = l(m,n) - W_{2,1}\left[h\right](m,n), \tag{2.11}$$

$$x_2(m,n) = 2h(m,n) + W_{1,2}\left[x_1\right](m,n). \tag{2.12}$$

For one scale of lifting-based Haar MCTF, we have two motion fields, the forward field $W_{2,1}$ and the backward field $W_{1,2}$ for every pair of frames, or one field per frame. It can be shown that, for multiple scales of temporal transformation, we have approximately two fields per frame [5].

The lifting-based MCTF process can be easily extended to longer wavelet filters, such as the biorthogonal 5-3 filter (i.e., linear lifting). Let $x_{2k}(m,n)$ and $x_{2k+1}(m,n)$ denote even- and odd-frame subsequences of the original video sequence, respectively.

The 5-3 lifting steps can be summarized as follows [6],

$$h_k(m,n) = x_{2k+1}(m,n) - \frac{1}{2}\left(W_{2k,2k+1}[x_{2k}](m,n) + W_{2k+2,2k+1}[x_{2k+2}](m,n)\right), \tag{2.13}$$

$$l_k(m,n) = x_{2k}(m,n) + \frac{1}{4}\left(W_{2k-1,2k}[h_{k-1}](m,n) + W_{2k+1,2k}[h_k](m,n)\right), \tag{2.14}$$

while the inverse transform is

$$x_{2k}(m,n) = l_k(m,n) - \frac{1}{4}\left(W_{2k-1,2k}[h_{k-1}](m,n) + W_{2k+1,2k}[h_k](m,n)\right), \tag{2.15}$$

$$x_{2k+1}(m,n) = h_k(m,n) + \frac{1}{2}\left(W_{2k,2k+1}[x_{2k}](m,n) + W_{2k+2,2k+1}[x_{2k+2}](m,n)\right). \tag{2.16}$$

Each highpass frame is actually the residual from a bi-directional motion-compensated prediction of the two neighboring original video frames along the motion trajectories. It is important to note that the number of motion-mapping operators grows by a factor of two compared to that of the Haar filter.

The use of traditional block matching to provide the motion fields $W_{i,j}$ for lifting-based MCTF is problematic. Although block-based ME can produce motion mappings $W_{i,j}$ and $W_{j,i}$ (e.g., [10]), unconnected pixels are not accounted for in the MCTF in this case, and, thus, the backward $W_{i,j}$ and forward $W_{j,i}$ motion mappings are not inverses of one another. Consequently, the unconnected pixels must be processed separately from the MCTF, and either both motion mappings must be transmitted to the decoder, or one mapping must be approximately estimated from the other. If it is desired that the MCTF must be invertible, then the encoder must code more than one motion field per frame. Specifically, for the Haar transform, two fields per frame are required; for the 5-3 transform, four fields per frame are needed. In order not to overly burden the

bitstream with motion-vector overhead, one would prefer to not increase this overhead beyond that for traditional hybrid coders like H.263 and MPEG-2, namely, one motion field per frame. In order to provide the requisite invertibility of the motion fields, it has been proposed that mesh-based MC be used [5, 6].

2.3 Lifting MCTF with Meshes

In recent years, mesh-based ME/MC (e.g., [21–25]) has been proposed to exploit geometries more general than traditional rectangular blocks, in order to provide more accurate modeling of object motion in video. The simplest mesh structure, the triangle mesh, has been applied using both regular (e.g., [21]), as well as irregular (e.g. [22–25]), geometry; in the latter case, irregular triangle meshes are typically explicitly adapted to image content, while regular triangle meshes are typically constructed by subdividing traditional block geometries irrespective of image content. In either case, the ME process involves the tracking of triangle vertices, or "control points," from one frame to another, while MC involves the mapping of the contents of each triangle from one frame to another using an affine transform.

2.3.1 Mesh-based MC/ME and Affine Transforms

Although it is possible to iteratively optimize control-point motion to globally adapt the mesh from one frame to the other [21], a simpler ME strategy, which is illustrated in Fig. 2.3, involves centering a small block about each control point and estimating motion of the block from one frame to the next in a process akin to traditional block-based ME [23–25]. In either case, after control-point motion is determined, an affine transform is constructed for each triangle based on the positions of its three control points in both frames. The transform is then applied to map each pixel in the triangle into the other

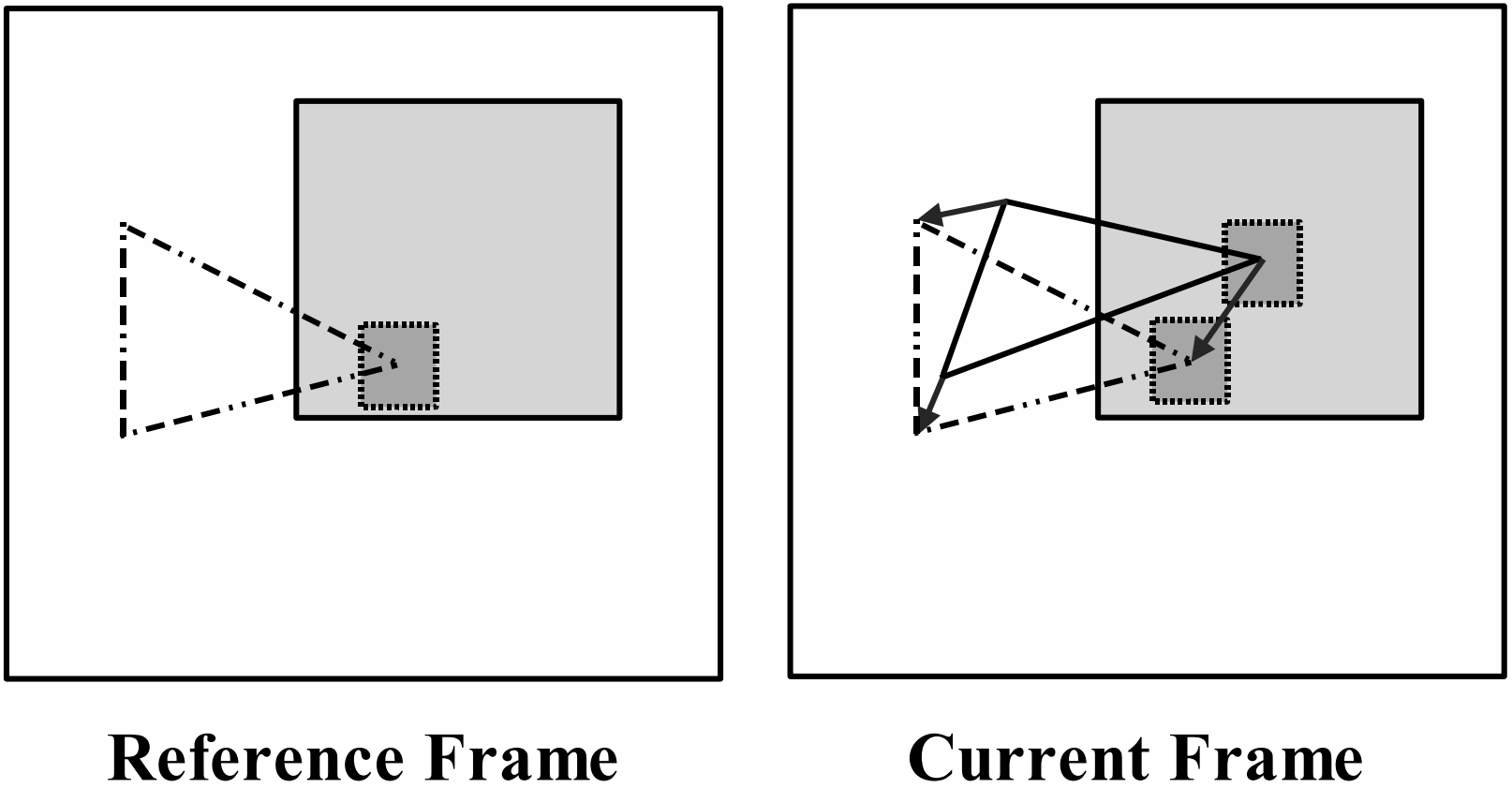


Figure 2.3: A simple mesh ME strategy. A small block centered about each control point is used to estimate motion of the control point from one frame to the next.

frame, while some form of interpolation, often bilinear interpolation, is used to resolve the subpixel positions inevitably produced by the mapping [23–25].

Affine transforms are widely used in computer graphics. In homogeneous coordinates, affine transforms can represent translation, rotation, and scaling. Consequently, an affine transform can map a point inside one triangle to point inside another triangle. The affine transform is a vector-matrix equation,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.17}$$

where x and y are the coordinates of a coefficient in a triangle in the current frame, x' and y' are the corresponding coordinates in the reference-frame triangle, and a_1, a_2, a_3, b_1, b_2, and b_3 are the six parameters of an affine transform that is determined for each pair of current- and reference-frame triangles independently [22]. To determine the transform parameters, we evaluate (2.17) for each of the three vertices of the triangle in the current frame using the known relation between the current- and reference-frame vertices,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}, \tag{2.18}$$

to yield six equations in six unknowns. This procedure is illustrated in Fig. 2.4. Once the parameters of the transform are determined, it is applied to a coefficient location in the current frame to determine the corresponding location in the reference frame, from which a prediction of the coefficient is determined. Bilinear interpolation is employed to calculate predictions for locations that lie off the RDWT-coefficient grid in the reference frame.

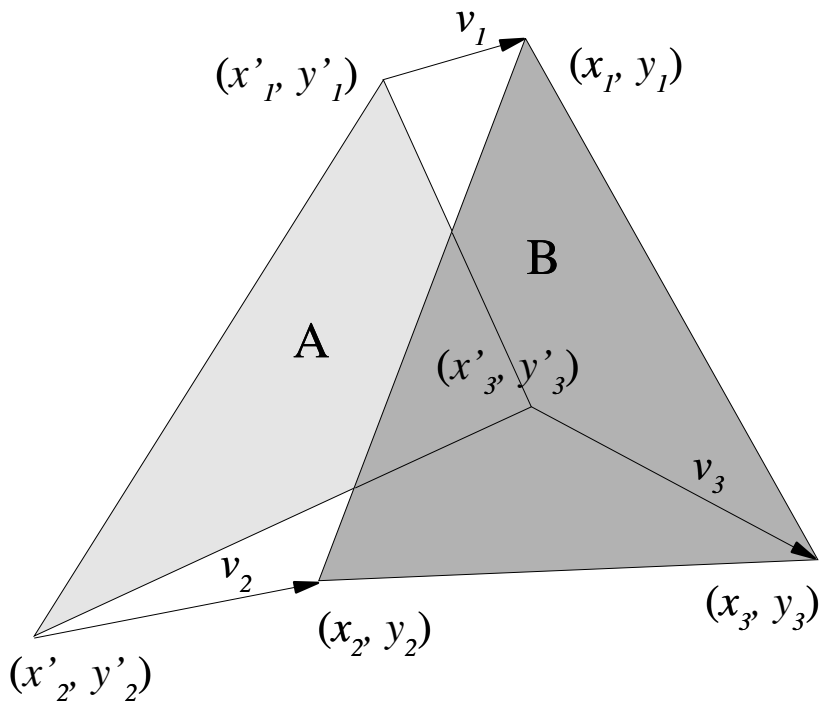


Figure 2.4: Affine transformation. $\triangle A$ is a triangle in reference frame. $\triangle B$ is a triangle in current frame. Motion vector $v_i = (\Delta_{x_i}, \Delta_{y_i})$. The affine transformation parameters a_1, a_2, a_3, b_1, b_2, and b_3 between these two triangles can be calculated using all the known values.

2.3.2 Motion Fields for Mesh-based MCTF

In [5, 7], it is proposed to use triangular mesh-based MC in lifting MCTF in order to circumvent the limitations of the block-displacement motion model. Since mesh-based geometries can account for local expansions and contractions in the motion field, unconnected pixels do not occur. Consequently, the forward motion field, $W_{j,i}$, and the backward motion field, $W_{i,j}$, are inverses of each other and only one field need be encoded. In [5], this fact is used to significantly reduce the number of encoded motion fields, and further reduction comes from the observation that, in a multiple-scale temporal transform, the motion fields at each transform scale closely resemble one another. Specifically, in [5], a regular triangular mesh is formed in one frame by dividing square blocks along their diagonals, and motion vectors are estimated for each vertex of the mesh. By combining this mesh-based MCTF with embedded spatial coding of the temporally transformed frames, the system of [5] achieves full temporal, resolution, and fidelity scalability with motion overhead on the order of one motion field per frame.

For example, motion fields for two temporal transform scales with the 5-3 filter are illustrated in Fig. 2.5. If the ME model were block-based, the discontinuous motion fields would make F_{11} different from the inverse of B_{11}. However, unlike block-based ME, mesh-based ME can track more complex motion to yield continuous motion fields without the unconnected-pixel problem, thereby producing motion fields that are perfectly invertible. Thus, as indicated in Fig. 2.5, only two motion fields, F_{12} and B_{21}, need be coded. The forward mapping F_{21} is recovered by inverting of B_{21}, while concatenating F_{12} and B_{21} produces B_{11}. Finally, B_{12} and F_{11} are the inverses of F_{12} and B_{11}, respectively. Consequently, only F_{12} and B_{21} need to be coded in order to recover all six motion fields, given that the motion fields are continuous and invertible.

As originally proposed, the system of [5, 7] uses a uniform, regular triangle mesh resulting from the dividing of the frame into square blocks and the splitting of each

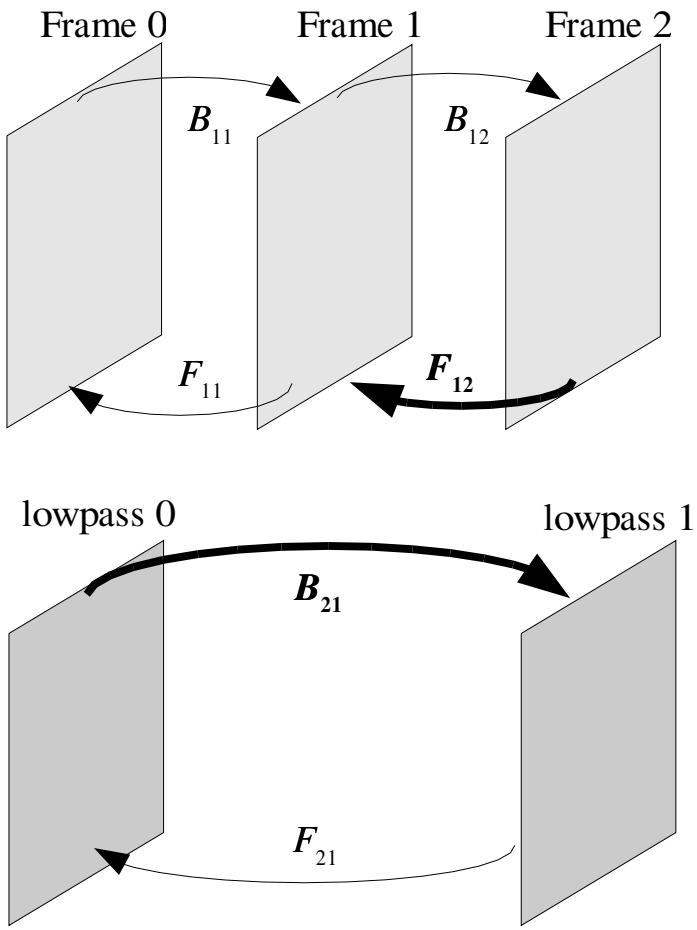


Figure 2.5: Motion fields for two scales of temporal transform with the 5-3 filter. F_{ij} denotes the jth forward mapping in the ith transform scale and B_{ij} is the corresponding backward mapping. Adapted from [5].

block along its diagonal. The control points of this uniform mesh are tracked from one frame to the next via the iterative hexagonal-refinement optimization of [21]. In our investigations to follow in Chap. IV, we also use this regular triangle-mesh structure; however, we opt for the simpler block-based ME strategy of [23] to determine control-point motion.

2.4 Motion Estimation with Subpixel Accuracy

In video sequences, all pixels are assumed to lie at integer positions; however, objects do not necessarily move in integer-pixel displacements. Consequently, modern ME/MC algorithms often employ motion vectors with resolution increased to subpixel accuracy. This increased accuracy typically yields a better prediction of motion at the cost of a higher motion-vector rate overhead. In many cases, however, a net gain in rate-distortion performance is achieved.

In subpixel-accurate ME/MC, motion vectors often point to positions in between integer pixels, thereby requiring some form of interpolation to resolve subpixel values. The simplest approach is perhaps bilinear interpolation, and extensive literature has shown that a simple bilinear interpolation achieves good performance for half-pixel accuracy in traditional block-based ME/MC systems. However, to further increase accuracy beyond half-pixel in these systems, bilinear interpolation does not typically improve performance since the additional motion-vector overhead usually outweighs the potential reduction in distortion [26]. As a result, more sophisticated interpolation filters replace bilinear interpolation in many modern systems; for example, quarter-pixel accuracy is achieved in the block-based ME/MC procedure of MPEG-4 using an 8-tap filter [26].

In block-based ME/MC, increasing the resolution of a motion vector affects the accuracy with which every pixel in the block is predicted. However, for triangle-

mesh ME/MC, rate-distortion gain due to subpixel-accuracy is difficult to foresee, since increased resolution of control-point motion may have only a modest effect on the affine transform itself. Additionally, the affine transform already interpolates to subpixel accuracy in the interior of the triangle even when the control points move with integer displacement. However, we will see later that using half-pixel accuracy does offer rate-distortion benefits in the triangle-mesh ME/MC which we use in the 3D video coder we propose in Chap. IV. This coder combines the concept of MCTF as described above with a form of multihypothesis ME/MC that employs diversity of phase in redundant wavelet expansions to improve MC performance. We overview the general ways redundant wavelet transforms have been used previously in video coding, as well as detail the specifics of the phase-diversity approach to MC, in the next chapter before proposing our 3D redundant-wavelet-multihypothesis (3D-RWMH) coder in Chap. IV.

CHAPTER III

THE REDUNDANT DISCRETE WAVELET TRANSFORM AND ITS USE IN VIDEO CODING

With the rise of wavelet transforms in still-image coding, there has been great interest in exploiting the proven coding efficiency of these transforms for video. However, the interaction between motion estimation/motion compensation (ME/MC) and the discrete wavelet transform (DWT) has posed some difficulties. The most straightforward approach to incorporating wavelet transforms into video is to replace the discrete cosine transform (DCT) widely employed in video-compression systems with the DWT, such that ME/MC takes place in the spatial domain, and the DWT is applied to the resulting residual image. However, this simple approach suffers from blocking artifacts which are exacerbated if the DWT is applied, as is usual, on the whole video frame rather than block-by-block. The alternative is to have ME/MC take place in the wavelet domain. Wavelet-domain ME/MC eliminates the inefficiency associated with blocking artifacts; moreover, resolution-scalable coding, such that a single coded bitstream can be decoded at multiple frame sizes, becomes possible. However, the fact that the usual critically sampled DWT, used ubiquitously in image-processing applications, is shift variant greatly hinders the ME/MC in the wavelet domain. Recent efforts for wavelet-domain ME/MC have consequently focused on using the redundant discrete wavelet transform (RDWT). The RDWT is a redundant, or overcomplete, transform which, in essence, eliminates the downsampling associated with the DWT to provide shift invariance.

In this chapter, we review the RDWT and how it has been incorporated into video coders in recent literature. While most techniques use the RDWT to provide shift invariance, our own work in this area [18, 19] additionally makes use of the multiple phases of the RDWT to improve ME/MC prediction. We review this redundant-wavelet approach to multihypothesis ME/MC in Sec. 3.3.

3.1 The Redundant Discrete Wavelet Transform (RDWT)

The RDWT can be considered to be an approximation to the continuous wavelet transform that removes the downsampling operation from the traditional critically sampled DWT to produce an overcomplete representation. The shift-variance characteristic of the DWT arises from its use of downsampling, while the RDWT is shift invariant since the spatial sampling rate is fixed across scale. The RDWT has also been called the "undecimated DWT," the "overcomplete DWT," and the *algorithme à trous*. The reader is referred to [27] for greater detail on the RDWT, its implementations, and its relation to the critically sampled DWT.

There are several ways to implement the RDWT, and several ways to represent the resulting overcomplete set of coefficients. As we will see in the next section, the most popular coefficient-representation scheme employed in RDWT-based video coders is that of a "coefficient tree," as illustrated in Fig. 3.1 for a 1D signal. This tree representation is easily created by employing filtering and downsampling as in the usual critically sampled DWT; however, all sets, or "phases," of downsampled coefficients are retained and arranged as "children" of the signal that was decomposed. The process is repeated on the lowpass bands of all nodes to achieve multiple decomposition scales. It is straightforward to see that each path from root to leaf in the RDWT tree constitutes a distinct critically sampled DWT, and there are 2^J such critically sampled DWTs in a J-scale decomposition. An alternative, and equivalent, implementation of the RDWT

tree representation comes from employing consistent subsampling phase and shifting the lowpass bands by one sample to generate children in the tree. Indeed, this "low-band-shift" [28] method has been a popular implementation for the RDWT-based video coders we consider in the next section.

The tree representation of RDWT coefficients makes for convenient identification of critically sampled DWTs within the overcomplete coefficient set. However, the *algorithme à trous* implementation yields perhaps the most natural coefficient representation. In this implementation, decimation following wavelet filtering is eliminated, and, for each successive scale of decomposition, the filter sequences themselves are upsampled, creating "holes" of zeros between nonzero filter taps. As a result, the size of each subband resulting from an RDWT decomposition is exactly the same as that of the input signal, as is illustrated for a 1D signal in Fig. 3.2. The advantage of this "spatially coherent" representation is that each RDWT coefficient is located within its subband in its spatially correct position. As illustrated in Fig. 3.2, by appropriately subsampling each subband of an RDWT, one can produce exactly the same coefficients as does a critically sampled DWT applied to the same input signal. In fact, in a J-scale 1D RDWT, there exist 2^J distinct critically sampled DWTs corresponding to the choice between even- and odd-phase subsampling at each scale of decomposition. Additionally, the coefficients at a given scale in the tree representation of the RDWT can be appropriately "interleaved" to produce the subbands of the spatially coherent representation; i.e., the two representations consist of exactly the same coefficient values.

The situation is similar for 2D decompositions implemented with separable 1D transforms. In this case, each node in the tree representation would have four children corresponding to the choice of even and odd subsampling in both the horizontal and

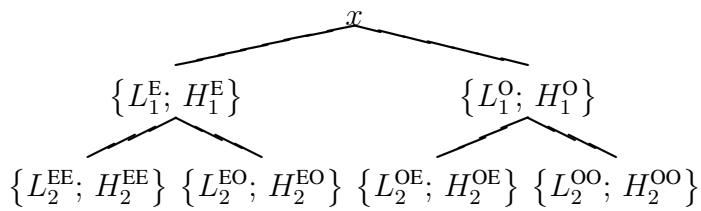


Figure 3.1: Tree representation of a two-scale RDWT of 1D-signal x. Approximation and detail coefficients at scale j are L_j and H_j, respectively. E indicates even-phase subsampling; O indicates odd-phase subsampling. A path from root to leaf indicates a distinct critically sampled DWT; a J-scale RDWT consists of 2^J such DWTs.

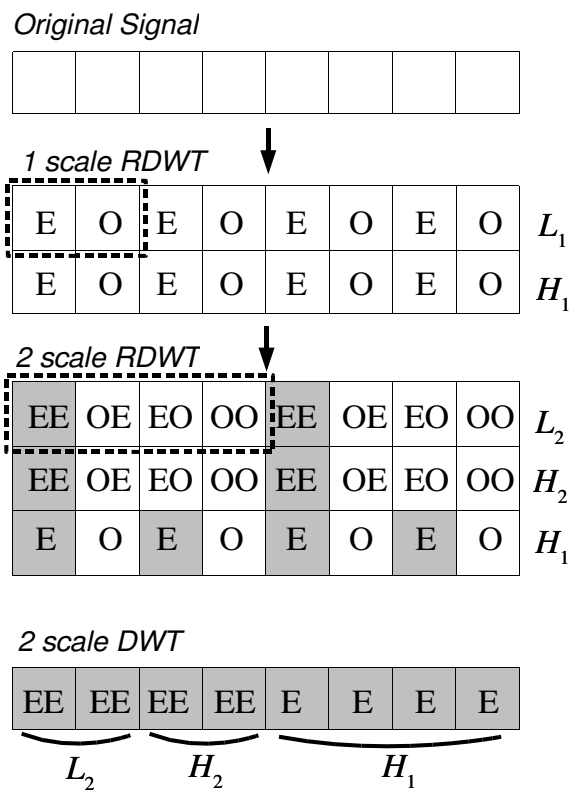


Figure 3.2: Spatially coherent representation of a two-scale RDWT. Coefficients retain their correct spatial location within each subband. Gray coefficients indicate the subsampling pattern necessary to recover one of the 2^J critically sampled DWTs.

vertical directions, and a J-scale 2D RDWT consists of 4^J distinct critically sampled DWTs. A 2D RDWT is illustrated in Figs. 3.3 and 3.4.

The RDWT is a perfectly reconstructing transform. To invert the RDWT, one can simply independently invert each of the constituent critically sampled DWTs and average the resulting reconstructions together. However, this implementation of the inverse RDWT incurs unnecessary duplicate synthesis filterings of the highpass bands; thus, one usually alternates between synthesis filtering and reconstruction averaging on a scale-by-scale basis in practical implementations (see [29]). The final reconstruction of this latter implementation, however, is identical to that produced by the conceptually simpler former approach.

3.2 Motion-Estimation/Motion-Compensation (ME/MC) Using The RDWT

In recent literature, the RDWT has been used extensively within the traditional hybrid video-coding architecture that features a motion compensated-prediction (MCP) feedback loop. Later, in Chap. IV, we introduce a system that employs the RDWT in the alternative 3D MCTF architecture. However, first we review those systems proposed with the traditional MCP loop.

The majority of prior work concerning RDWT-based video coding originates in the work of Park and Kim [28], in which the system shown in Fig. 3.5 was proposed. In essence, the system of Fig. 3.5 works as follows. An input frame is decomposed with a critically sampled DWT, and the resulting wavelet-domain coefficients are partitioned into blocks. Each block consists of all the coefficients in the DWT that correspond to a particular spatial-domain block in the original image, and thus includes coefficients from all subbands at all scales. A full-search block-matching algorithm then computes motion vectors for each wavelet-domain block; the system uses as the reference for this search an RDWT decomposition of the previous reconstructed frame. Since these reconstructed

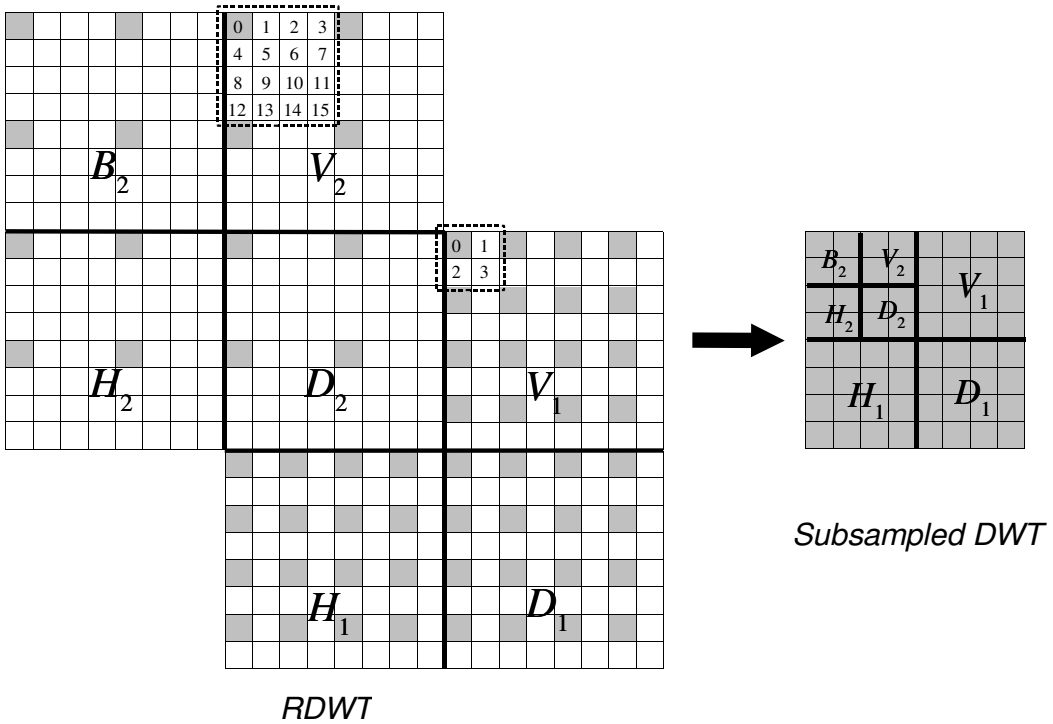


Figure 3.3: Spatially coherent representation of a two-scale 2D RDWT. Coefficients retain their correct spatial location within each subband, and each subband is the same size as the original image. B_j, H_j, V_j, and D_j denote the baseband, horizontal, vertical, and diagonal subbands, respectively, at scale j. This figure shows that subsampling recovers one of the 4^J critically sampled DWTs.

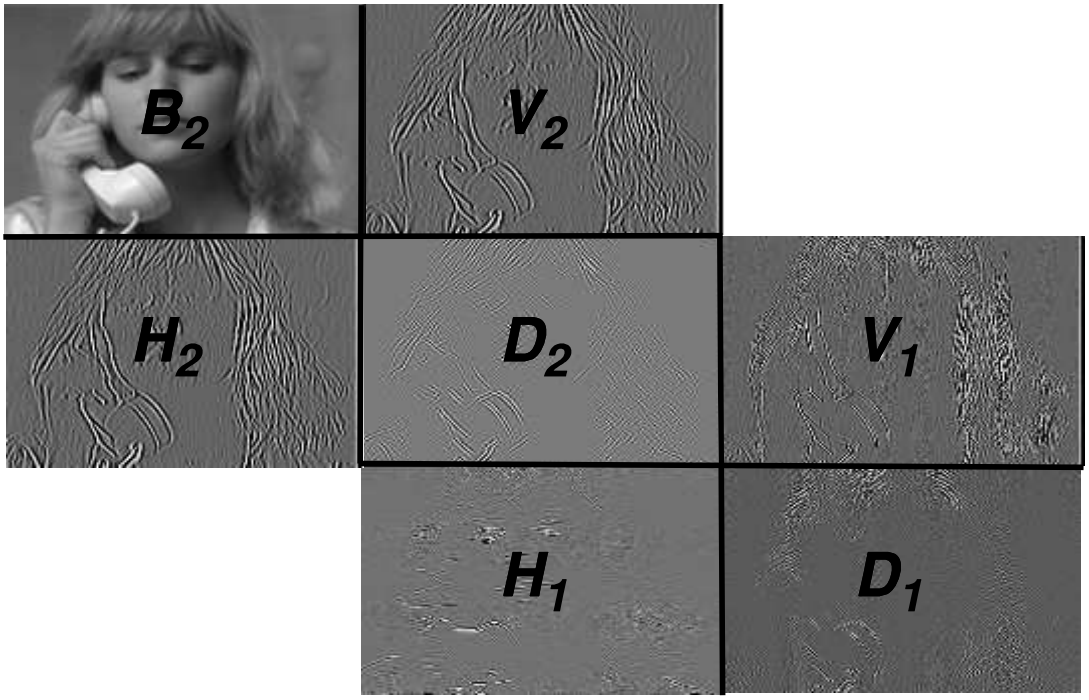


Figure 3.4: An example of an RDWT applied to an image. B_j, H_j, V_j, and D_j denote the baseband, horizontal, vertical, and diagonal subbands, respectively, at scale j.

RDWT coefficients are arranged in the tree representation as described above, the ME procedure of this system, which is shown in Fig. 3.6, amounts to identifying, for each block of the current frame, a particular critically sampled DWT in the reference-frame tree (a root-to-leaf path), and a displacement within that DWT. Transmission of a single motion vector per block suffices to convey all of this motion information to the decoder. A suitable cross-scale distortion metric that averages distortions incurred in each subband is used to drive the ME search. In summary, a single critically sampled DWT of the current frame is predicted in a block-by-block manner from a wavelet-domain reference frame wherein all phases are retained. By using such an overcomplete expansion of the reference frame, the best-matching block from all possible phases is obtained, and the shift-variant nature of the critically sampled DWT is overcome.

Subsequent work has further refined the system depicted in Fig. 3.5. In particular, in [30–32], multiple motion vectors are transmitted for each current-frame block by estimating motion in each subband independently, while the system of [33] employs interpolation between the coefficients in distinct root-to-leaf paths of the RDWT tree to enable motion compensation to be performed with sub-pixel accuracy. Additionally, resolution-scalable video coders [31, 32, 34, 35] have been devised that constrain the ME/MC procedure to process each scale of the wavelet decomposition independently.

Our own prior work [24, 25, 36] in this area departs significantly from the RDWT-based video-coding architecture originating in [28] and shown in Fig. 3.5. Specifically, we have designed a system which adapts the triangle-mesh ME/MC technique described in Sec. 2.3 to the wavelet domain by replacing the block-based ME/MC of [28] with triangle-mesh ME/MC. This modification to the system of [28] necessitates that we employ all phases of an RDWT of the current input frame to estimate motion rather than a single critically sampled DWT of the current frame. The encoder of our redundant-wavelet-triangle-mesh (RWTM) video-coding system [24, 25, 36] is depicted in Fig. 3.7.

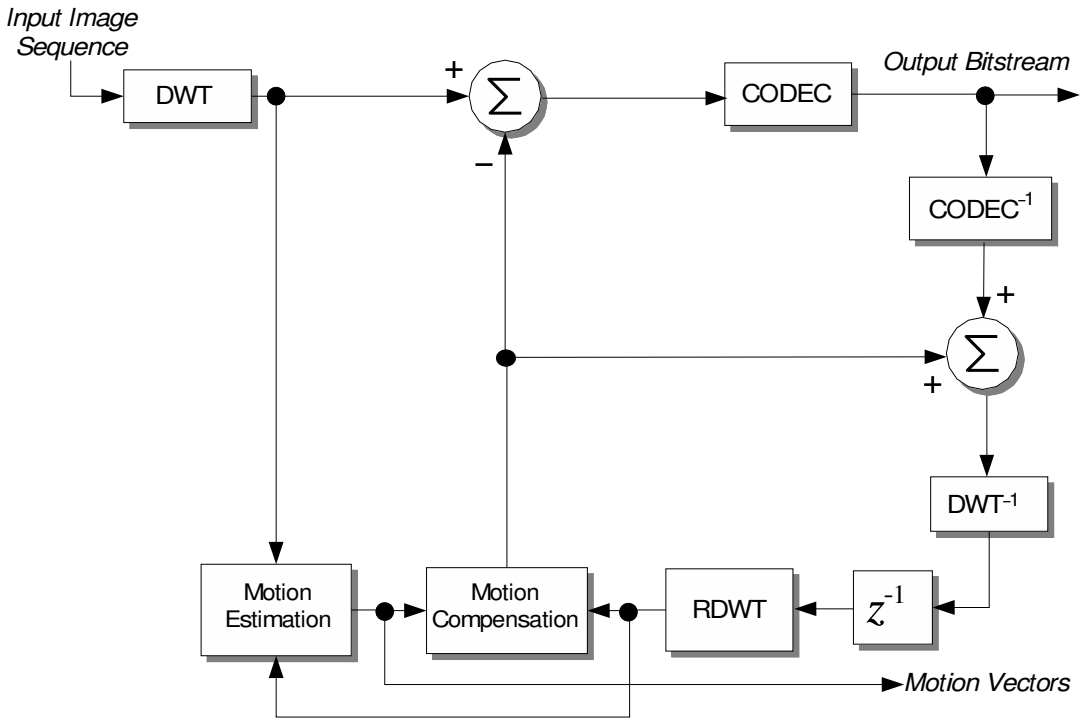


Figure 3.5: The RDWT-based video coder of [28]. z^{-1} = frame delay, *CODEC* is any still-image coder operating in the critically-sampled-DWT domain.

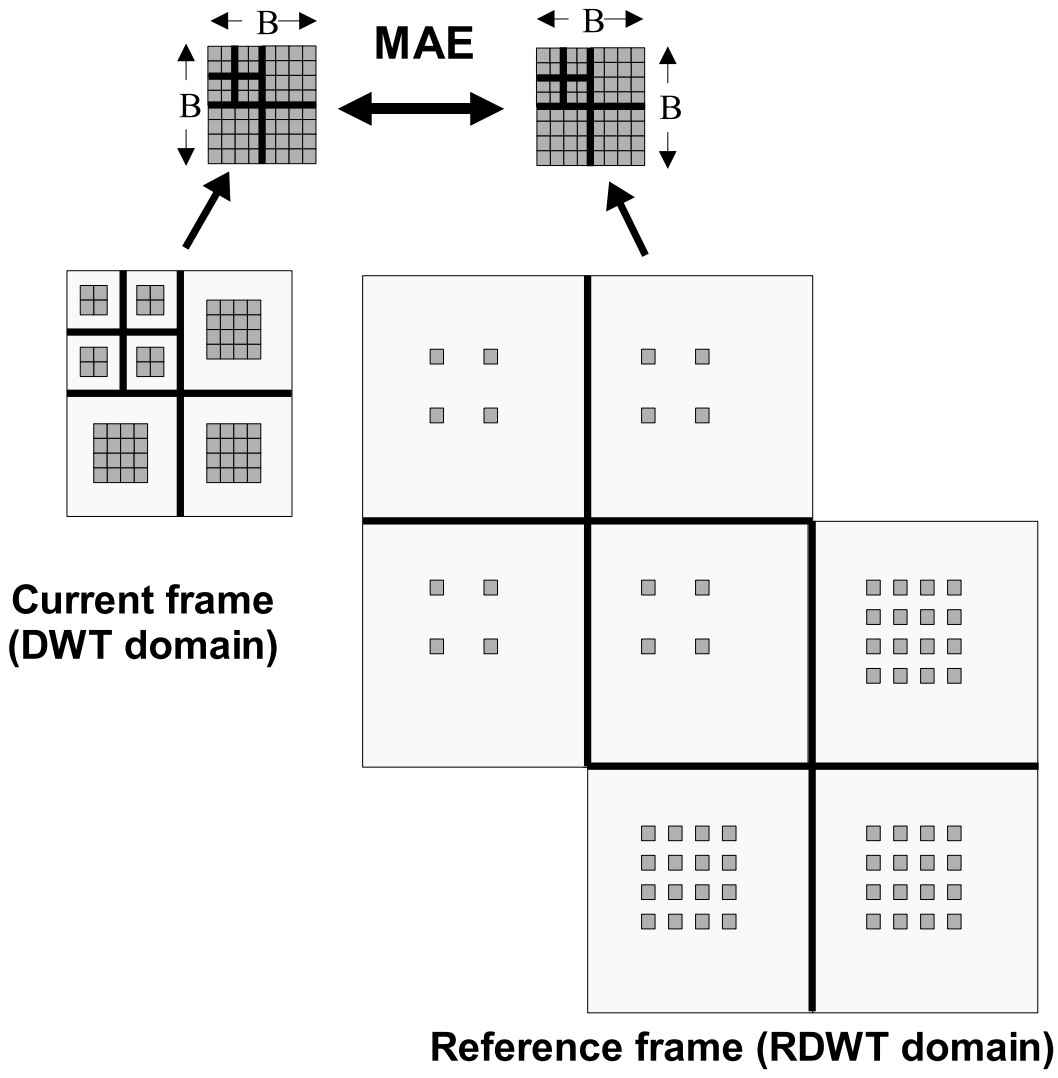


Figure 3.6: The ME procedure of [28], spatially coherent representation. B is the block size for block-matching ME.

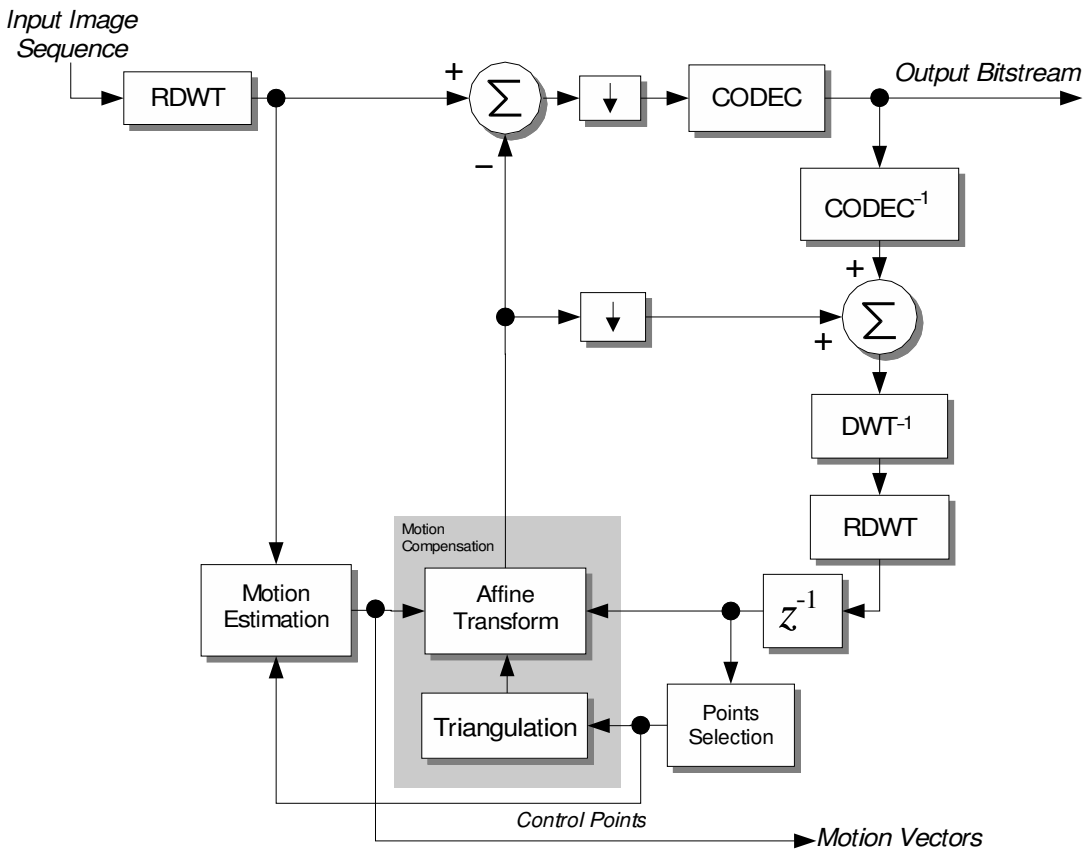


Figure 3.7: The RWTM coder of [24, 25, 36].

The input image is first transformed using an RDWT, and control points, located on most salient image edges, are identified in the previous reference frame. The motion of these control points from the reference frame to the current frame is estimated in the RDWT domain, and motion vectors are transmitted to the decoder to allow it to track control-point motion. MC is accomplished by first using a triangulation algorithm to generate a triangle mesh on the control points in the reference frame and then using affine transformations to predict, subband by subband, triangles in the current frame from triangles in the reference frame. Residing in the RDWT domain, the motion-compensated residual is itself redundant; consequently, it is downsampled before coding. The final encoding step consists of a wavelet-domain still-image coder—we have used SPIHT [37], but any wavelet-domain still-image coder would suffice. The identification of image edges in this system is accomplished through use of a "correlation mask" calculated across RDWT subbands.

The above systems exploit the redundancy of the RDWT solely to provide shift invariance, thereby permitted ME/MC to take place in the wavelet domain. The only exception is the system of [24, 25, 36], which additionally uses the RDWT to aid tracking the ME/MC mesh in time. In the next section, we present an additional use of RDWT redundancy—multihypothesis prediction.

3.3 Multihypothesis Motion Compensation (MHMC) and the RDWT

Multihypothesis motion compensation (MHMC) [38] forms a prediction of pixel $s(x, y)$ in the current frame as a combination of multiple predictions in an effort to combat the uncertainty inherent in the motion-estimation process. Assuming that the combination of these hypothesis predictions is linear, we have that the prediction of $s(x, y)$ is

$$\tilde{s}(x, y) = \sum_i w_i(x, y)\tilde{s}_i(x, y), \tag{3.1}$$

where the multiple predictions $\tilde{s}_i(x, y)$ are combined according to some weights $w_i(x, y)$. A number of multihypothesis techniques for MC have been proposed over the last decade. One approach to MHMC is to implement multihypothesis prediction in the spatial domain; i.e., the predictions $\tilde{s}_i(x, y)$ are culled from spatially distinct locations in the reference frame. This class of MHMC includes fractional-pixel MC [39] and overlapped block motion compensation (OBMC) [40, 41]. Another approach is to deploy MHMC in the temporal domain by choosing predictions $\tilde{s}_i(x, y)$ from multiple reference frames. Examples of this class of MHMC are bidirectional prediction (B-frames) as used in MPEG-2 and H.263 and long-term-memory motion compensation (LTMMC) [42]. Of course, it is possible to combine these two classes by choosing multiple predictions that are diverse both spatially and temporally [43]. Note that the calculation of (3.1) in the decoder must be identical to that in the encoder; consequently, it will be necessary to transmit the weights $w_i(x, y)$ to the decoder as side information in the case that the weights are not fixed or not determinable from information already possessed by the decoder. Although implementation dependent, B-frames and LTMMC typically incur this additional side-information burden while fractional-pixel MC and OBMC do not.

In [18, 19, 36], we proposed a new class of MHMC by extending the multihypothesis-prediction concept into the transform domain. Specifically, we perform ME/MC in the RDWT domain, and use multiple predictions that are diverse in transform phase. An interesting aspect of our approach is that low-resolution information is inherently predicted with a greater number of hypotheses, which corresponds to the greater difficulty inherent in estimating motion in signals with spatially low resolution. Additionally, since the weighting of the individual predictions is carried out implicitly in the form of an inverse transform, no side information need be sent to the decoder.

Our redundant-wavelet multihypothesis (RWMH) technique works as follows. In all prior RDWT-based coders [24, 25, 28, 30–35], the coefficients of a critically sampled DWT of the current frame are motion compensated using an RDWT of the reference frame. However, if an RDWT of the current frame were to be used instead, there would be 4^J distinct critically sampled DWTs for the current frame. Using Fig. 3.3 as an example, in existing techniques, only the gray coefficients are motion compensated, i.e., only one critically sampled DWT of the current frame is involved in ME/MC procedure. These techniques are thus single-hypothesis methods. We could have used the coefficients of any other critically sampled DWT in the ME/MC process to have obtained another version of the motion fields and residual coefficients. In fact, each of the critically sampled DWTs within the RDWT of the current frame will view motion from a different perspective, and can be independently motion-compensated. Consequently, if motion is predicted in the RDWT domain and all DWTs participate in ME/MC, the inverse RDWT can construct a multihypothesis prediction in the form of (3.1). Specifically, for a J-scale RDWT, the reconstruction from DWT i of the RDWT is $\tilde{s}_i(x, y)$, $0 \leq i < 4^J$, while $w_i(x, y) = 4^{-J}$, $\forall i$. Below, we describe our RWMH video-coding system [18, 19, 36] that performs MHMC in precisely this fashion.

The encoder of our RWMH video-coding system is depicted in Fig. 3.8. The current and reference frames are transformed into RDWT coefficients, and both ME and MC take place in this redundant-wavelet domain. In a J-scale RDWT decomposition, each $B \times B$ block in the original spatial domain corresponds to $3J+1$ blocks of the same size, one in each subband. The collection of these co-located blocks is called a set. In the ME procedure, which is shown in Fig. 3.9, block matching is used to determine the motion of each set as a whole. Specifically, a block-matching procedure uses a cross-subband distortion measure that sums absolute differences for each block of the set similar to the cross-subband ME procedure of [28].

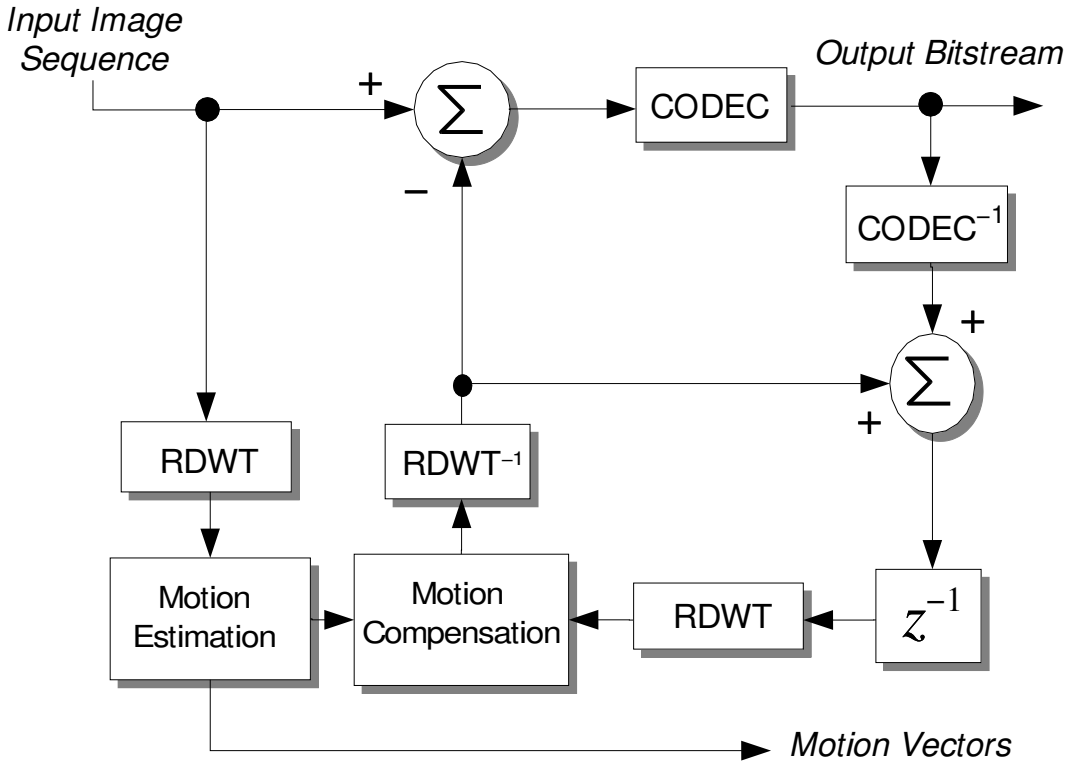


Figure 3.8: Block diagram of the RWMH video-coding system [18, 19, 36]. *CODEC* is any still-image coder.

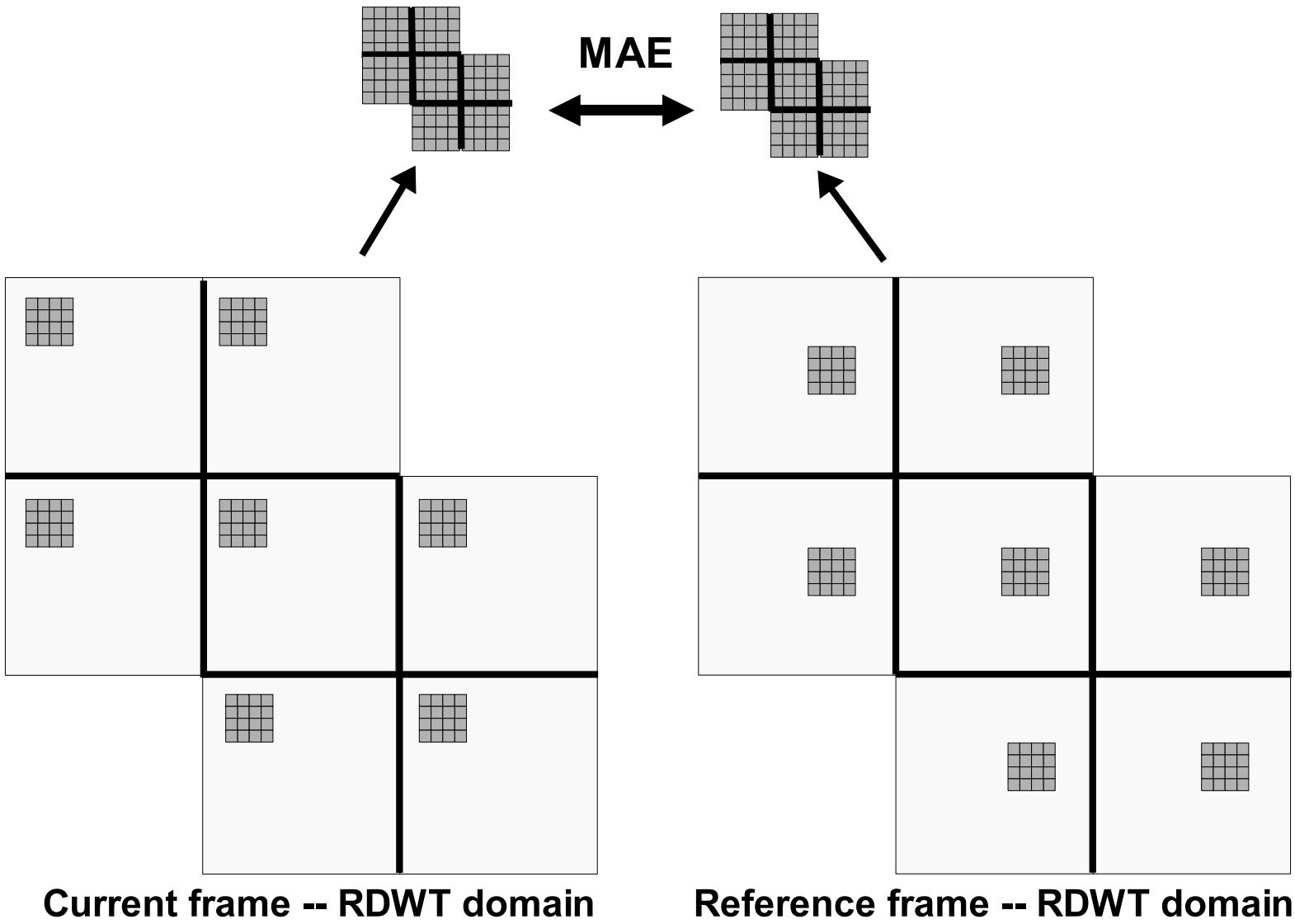


Figure 3.9: The ME procedure of [18, 19, 36].

After the ME search has determined motion vectors for each set, a motion-compensated frame is created in the RDWT domain using the same motion vector for each block of the set. The inverse RDWT is performed on this RDWT-domain motion-compensated frame, combining the multiple phases into a spatial-domain multihypothesis prediction. This spatial-domain prediction is subtracted from the current frame, and the residual is coded. This final encoding step consists of a still-image coder; for the experiments below, we use SPIHT [37], but any still-image coder, wavelet-based or otherwise, would suffice.

We demonstrate that our RWMH system yields significant performance improvement over the system of [28], which is a single-phase equivalent to our RWMH system. In the system of [28], ME is executed within the RDWT domain; however, only a single critically sampled DWT is predicted, and the ME is optimized to that single phase. Average PSNR figures for fixed bit rate are tabulated in Table 3.1, and frame-by-frame PSNR profiles for two sequences are shown in Figs. 3.10 and 3.11. In these results, "RDWT Block" refers to the system of [28]. In addition, "Spatial Block" refers to block-based ME/MC in the spatial domain, the traditional method employed in video-coding standards, followed by an entire-image DWT and then SPIHT coding of the DWT coefficients. In these results, we code grayscale sequences with the first frame intra-coded (I-frame), while all subsequent frames use ME/MC (P-frames). All wavelet transforms (DWT and RDWT) use the popular 9-7 biorthogonal filter with symmetric extension, and all ME/MC methods use integer-pixel accuracy.

These results illustrate that multihypothesis prediction in the form of our RWMH system achieves at least a 0.4-dB gain over single-phase prediction. For sequences with complex motion, our RWMH system achieves even larger performance gains. For example, RWMH exhibits a gain of nearly 1 dB over the system of [28] and a gain of over 2 dB over the spatial-domain system for the "Football" sequence.

Table 3.1: Distortion averaged over all frames of the sequences, from [18, 19, 36].

	PSNR (dB)		
	Spatial Block	RDWT Block	RWMH
Football†	26.3	27.9	28.6
Susie	36.0	37.4	37.8
Mother & daughter	40.2	40.8	41.2
Coastguard	28.1	28.9	29.5

Rate is 0.25 bpp except †, which is 0.5 bpp.

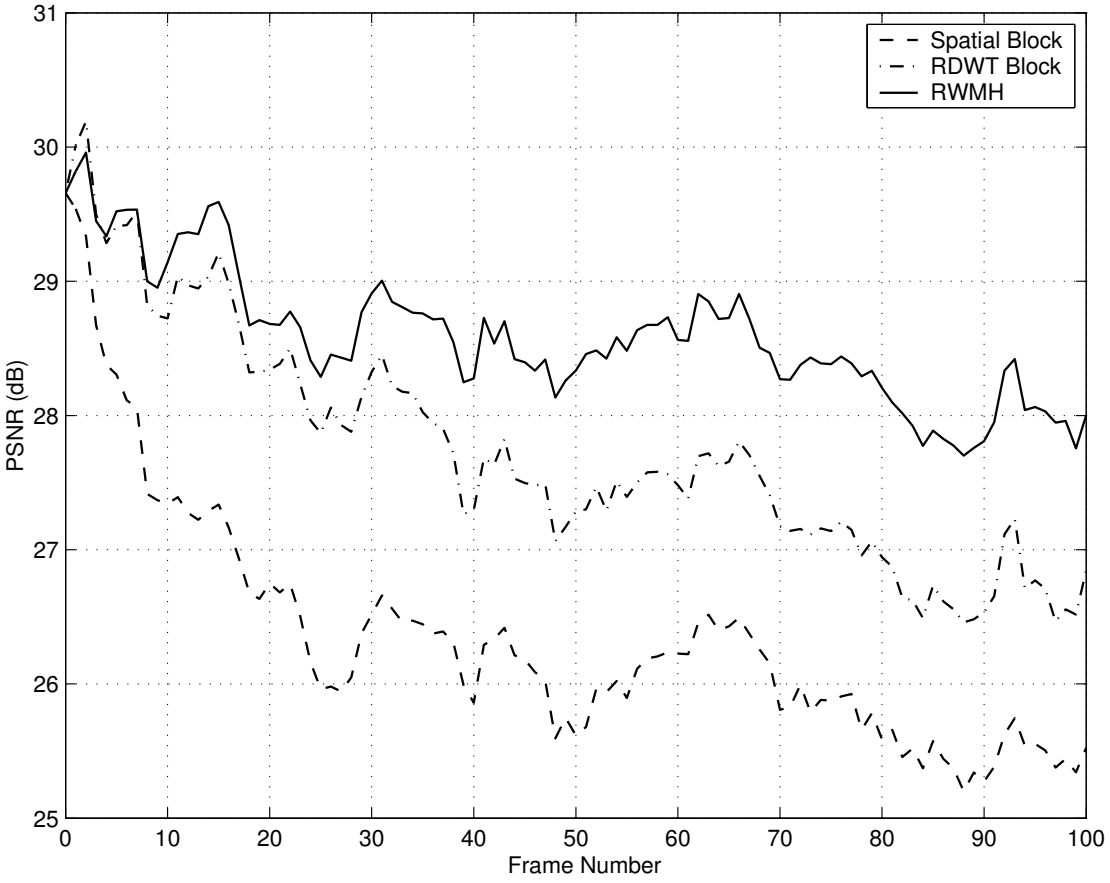


Figure 3.10: Frame-by-frame PSNR for "Football" at 0.5 bpp (1.3 Mbps), from [18, 19, 36].

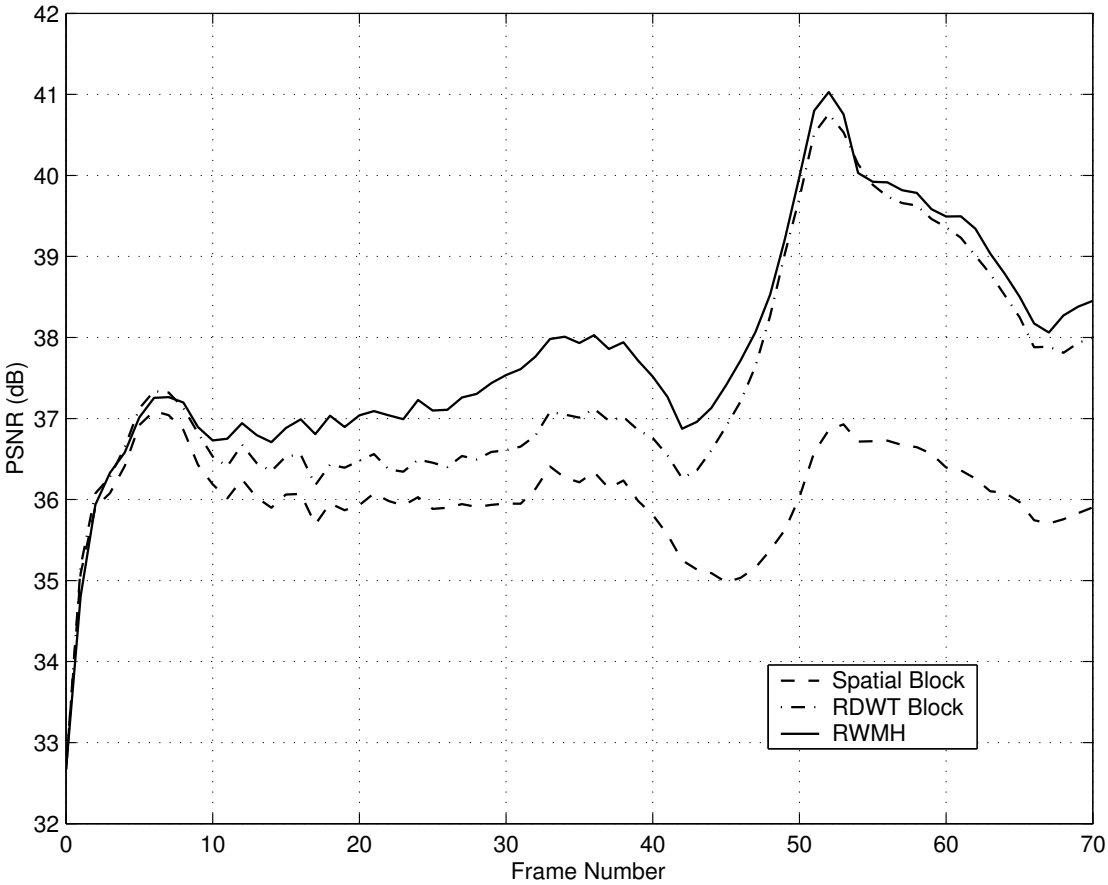


Figure 3.11: Frame-by-frame PSNR for "Susie" at 0.25 bpp (644 kbps), from [18, 19, 36].

CHAPTER IV

THE 3D-RWMH SYSTEM

As described in Sec. 3.3, a number of multihypothesis techniques for MC have been proposed in the past, including fractional-pixel-accurate MC, B-frames, OBMC, and multiple reference frames. These techniques employ multiple predictions that are diverse spatially or temporally to improve the overall predictive ability of the system. In [18, 19, 36], we introduced a new class of MHMC in which the multihypothesis-prediction concept is extended into the RDWT-transform domain, and uses multiple predictions that were diverse in transform phase. This redundant-wavelet-multihypothesis (RWMH) approach was outlined in Sec. 3.3. In the RWMH paradigm, each one of the critically sampled DWTs of an RDWT "views" motion from a different perspective and thus forms an independent hypothesis of the true motion of the video sequence, while the inverse RDWT combines these multiple hypotheses into a single prediction. In the system of Fig. 3.8, this prediction is incorporated into the MC feedback loop of a hybrid video-coding architecture employing block-based ME/MC. In this chapter, we introduce the RWMH concept into the MCTF framework to eliminate the MC feedback loop and produce a fully scalable 3D video coder. The discussion in this chapter elaborates on our previous publications [44, 45] in which the 3D-RWMH system was first developed.

4.1 System Overview

The encoder of our 3D-RWMH video-coding system, depicted in Fig. 4.1, first performs a spatial RDWT on each frame and then performs MCTF in the redundant-

wavelet domain. This is in contrast to many prior MCTF techniques [5–11] in which MCTF takes place in the spatial domain. Since MCTF is performed in the RDWT subbands, it is overcomplete spatially; consequently, before coding the temporal subbands, we remove this spatial redundancy by performing an inverse spatial RDWT on each frame. In essence, each RDWT phase in each frame can be considered to have viewed the MCTF from a different perspective and thus forms an independent hypothesis about the temporal filtering taking place. The inverse spatial RDWT implicitly combines these hypotheses into a multihypothesis estimate of what the true temporal filtering should be. After the inverse spatial transform, the temporally transformed frames are coded by a suitable 3D coder. In our experiments, we use 3D-SPIHT [4], but other coders are possible.[1] In our 3D-RWMH system, motion is tracked using a triangular mesh deployed in each of the subbands of the RDWT decomposition of each frame. Since all RDWT subbands are the same size, the same triangle mesh is used for all subbands of a frame, as described below.

4.1.1 Motion Estimation and Compensation

We first describe the ME/MC procedure of the 3D-RWMH system assuming integer-pixel accuracy; we consider the case of subpixel-accurate ME/MC below. The ME procedure for the 3D-RWMH system starts by identifying a uniform, regular triangular mesh in the first frame of the sequence by dividing the frame into square blocks and splitting each block along its diagonal. Motion into the next frame is estimated by centering a small block at each vertex in the first frame and finding the best matching block in the second frame, following the method outlined in Sec. 2.3.1. Motion of the control points from the second frame to the third frame is tracked in this same manner,

[1] For many 3D coders, such as 3D-SPIHT, a spatial forward DWT (not shown in Fig. 4.1) is applied to each frame following the spatial inverse RDWT of the 3D-RWMH system, since the coefficients resulting from 3D-RWMH are in the DWT domain in only one dimension (the temporal dimension).

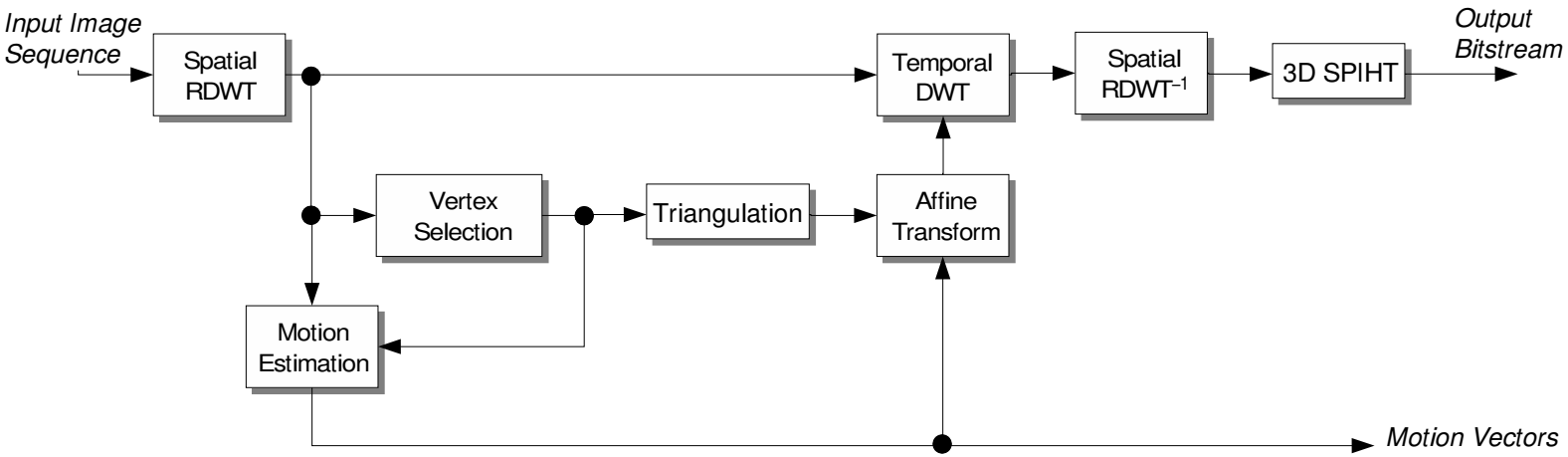


Figure 4.1: Block diagram of the 3D-RWMH video-coding system.

and so on to subsequent frames. If the motion vectors have integer-pixel accuracy, the control points in every frame reside on the integer-pixel grid.

We search for the motion of the control points of the mesh by minimizing a distortion metric that spans across all subbands of the RDWT decomposition, as we did in [24, 25]. Specifically, the motion vector, (Δ_x, Δ_y), for control point (x, y) in the reference frame is the vector in the search window about (x, y) in the current frame that minimizes the mean absolute error (MAE),

$$\mathrm{MAE}(x, y, \Delta_x, \Delta_y) = \frac{1}{B^2} \sum_{m=-\lfloor B/2 \rfloor}^{\lfloor B/2 \rfloor} \sum_{n=-\lfloor B/2 \rfloor}^{\lfloor B/2 \rfloor} \mathrm{AE}(x+m, y+n, \Delta_x, \Delta_y). \tag{4.1}$$

The absolute error (AE) is

$$\begin{aligned}\mathrm{AE}(x, y, \Delta_x, \Delta_y) = 2^{-J/2} \left| B_J^{cur}(x+\Delta_x, y+\Delta_y) - B_J^{ref}(x, y) \right| + \\ \sum_{j=1}^{J} 2^{-j/2} \Bigg\{ \left| V_j^{cur}(x+\Delta_x, y+\Delta_y) - V_j^{ref}(x, y) \right| + \\ \left| H_j^{cur}(x+\Delta_x, y+\Delta_y) - H_j^{ref}(x, y) \right| + \\ \left| D_j^{cur}(x+\Delta_x, y+\Delta_y) - D_j^{ref}(x, y) \right| \Bigg\},\end{aligned} \tag{4.2}$$

where *cur* and *ref* denote subbands from the current and reference frames, respectively, and B_j, H_j, V_j, and D_j are the baseband, horizontal, vertical, and diagonal subbands, respectively, at scale j. We assume block size B is odd. In the search, motion vectors are chosen from a window of size $W > 0$ such that $-W \leq \Delta_x, \Delta_y \leq W$. The evolution of a regular triangle mesh over frames is illustrated in Fig. 4.2.

For an N-frame video sequence, this ME process results in $N - 1$ motion fields regardless of the temporal filter used, as illustrated in Fig. 4.3. We note that this is the same number of motion fields produced by a traditional coder with a MC feedback

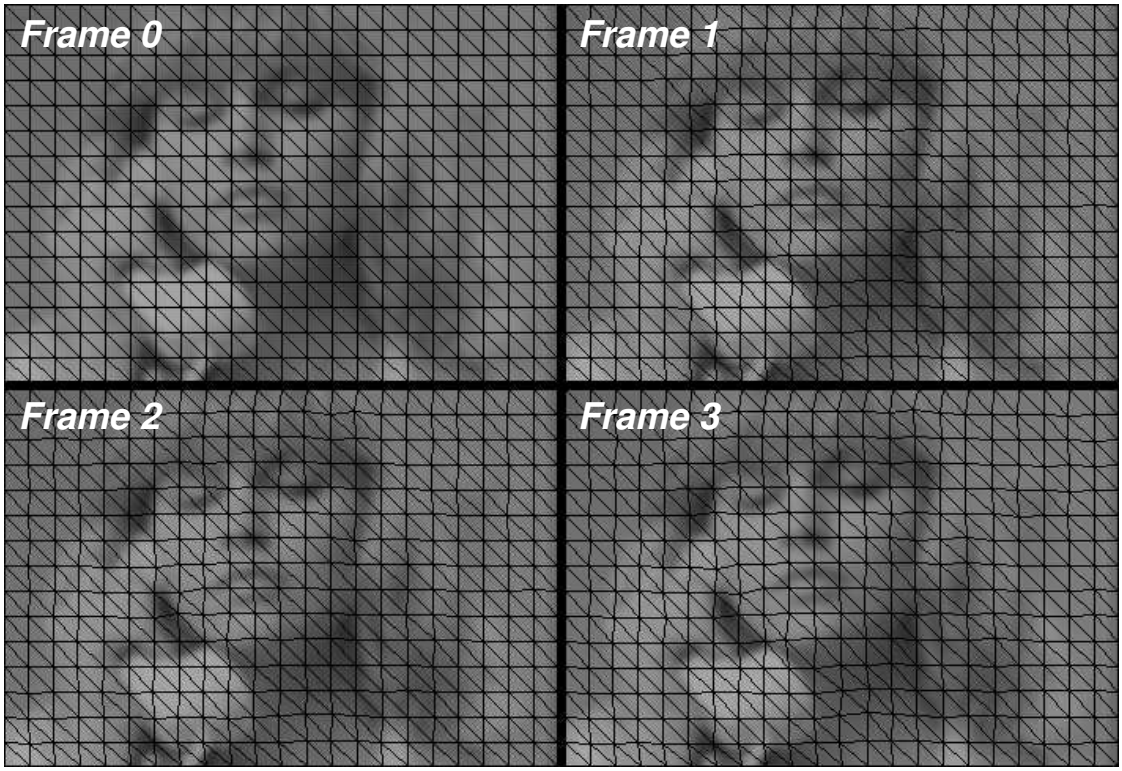


Figure 4.2: The regular triangle mesh in the first frame and its evolution over subsequent frames. Only the basebands of the frames are shown.

loop. Since each of the $N-1$ motion fields are mesh-based and thus completely invertible, forward and backward motion fields between each pair of frames can be calculated from these $N-1$ fields. Using these forward and backward motion fields, affine transforms between the triangles of each pair of frames are used to implement a motion-compensated lifting-based filtering in the temporal direction. This temporal filtering proceeds by mapping each triangle in a reference frame into the current frame using an affine transform as described in Sec. 2.3.1; this affine mapping is performed for each triangle in each subband separately. Bilinear interpolation between the surrounding four integer-pixel locations is used to resolve subpixel positions produced by the affine mapping.

We can construct affine transforms between any two frames by concatenating motion fields from the set of $N-1$ motion fields produced by the above ME process. Multiple-scale temporal transforms are thereby supported since these concatenated motion fields can be used for affine transforms at the higher temporal decomposition scales, as is illustrated in Fig. 4.3.

We have found it beneficial to periodically "reset" the triangle mesh rather than allow the ME of the control points to continue indefinitely. Using this periodic "reset" operation, we can avoid ME inaccuracies accumulation. Specifically, we track control-point motion for N' frames and then reset the triangle mesh to the initial uniform mesh (again by diagonally splitting square blocks). We repeat this procedure for the next N' frames. Fig. 4.3 illustrates the motion-tracking and mesh-resetting procedure for $N' = 4$ and $N = 8$. A single triangle mesh is used for all subbands of the RDWT; this is possible since each RDWT subband has the same size. MC proceeds by mapping each triangle in the reference frame into the current frame using an affine six-parameter model as described in Sec. 2.3.1; this affine mapping is performed for each triangle in each subband separately.

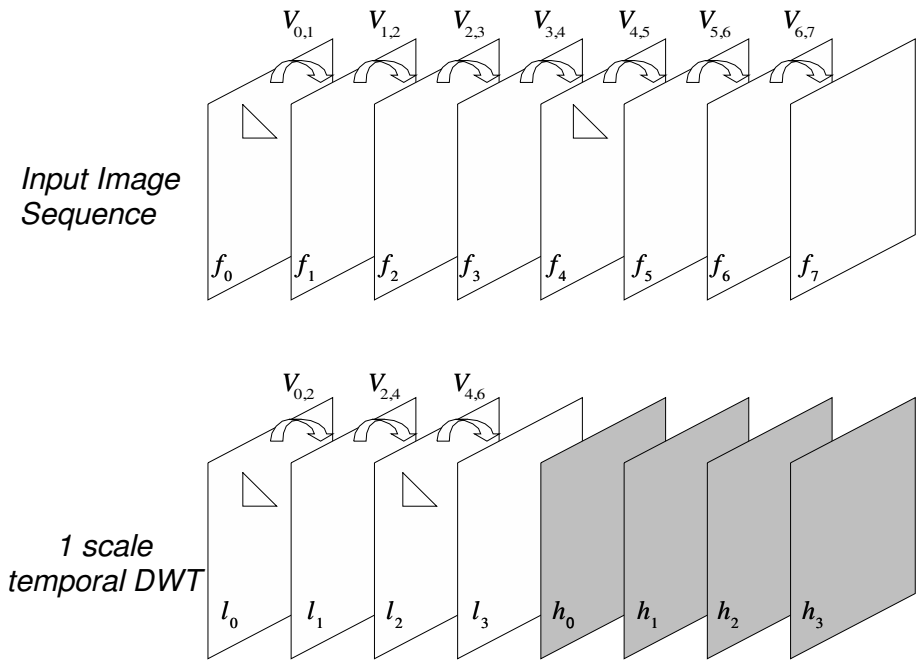


Figure 4.3: Motion fields for MCTF in the 3D-RWMH system for $N = 8$ and $N' = 4$. A small triangle in a frame indicates where the triangle mesh is reset to the uniform mesh. Field $V_{i,j}$ maps frame i to frame j; concatenated fields are $V_{0,2} = V_{0,1} + V_{1,2}$, $V_{2,4} = V_{2,3} + V_{3,4}$, and $V_{4,6} = V_{4,5} + V_{5,6}$.

4.1.2 Half-pixel Accuracy

The above ME procedure assumes integer-valued motion vectors, while MC in the form of the affine transform employs interpolation between integer-pixel values to resolve subpixel positions arising in the mapping. In this section, we describe modifications to the above ME/MC approach in order to accommodate half-pixel accuracy.

When the motion-vector resolution is increased to half-pixel accuracy, the motion-vector search is first carried out as described above for integer-pixel accuracy. Then, the eight neighboring locations at a distance of $\left(\pm\frac{1}{2}, \pm\frac{1}{2}\right)$ from the best match location are searched to refine the motion vector to half-pixel resolution. We note that, in the case that a motion vector points to a location on the half-pixel grid in one frame, the initial integer-valued search for the motion of the control point into the next frame involves half-pixel locations. In this case, x and y in (4.1) and (4.2) will refer to half-pixel locations while Δ_x and Δ_y will be integer-valued. The subsequent refinement search will involve both integer- and half-pixel locations.

Recall that, for ME/MC involving triangle meshes, subpixel accuracy is invoked in the affine-transform mapping of the MC process. In the half-pixel 3D-RWMH system, we use values on the half-pixel grid as the basis of the interpolation of the affine mapping. Specifically, the affine-transform mapping from one triangle to another uses bilinear interpolation applied to the four nearest locations on the half-pixel grid. In practice, we achieve subpixel accuracy for both ME and MC by interpolating the entire RDWT subband by factor 2 both horizontally and vertically. Afterward, ME of the control points and MC with the affine transform are carried out as if on the integer-pixel grid, and the resulting residual subbands are downsampled to their original size.

We have investigated two methods for producing values on the half-pixel grid—simple bilinear interpolation and more sophisticated filter-based interpolation. For

the latter approach, the interpolation filter we use is the 8-tap filter from [10] with coefficients {-0.0105, 0.0465, -0.1525, 0.6165, 0.6165, -0.1525, 0.0465, -0.0105}. Specifically, this 1D FIR filter is applied both horizontally and vertically on the integer-pixel values to generate values on the half-pixel grid. Recall that, in our 3D-RWMH system, the ME and MC processes are carried out on each RDWT subband individually; thus each RDWT subband is interpolated independently.

Motion-vector information for the motion fields resulting from the ME process is transmitted to the decoder. In our system, the H.261 variable-length-code (VLC) table for motion-vector data (MVD) is used for coding the integer part of the motion vectors, while the fractional part of the vectors is sent by appending a single-bit code to each Huffman codeword.

4.1.3 Experimental Results

We use the "Susie" (64 frames), "Football" (96 frames), "Mother & Daughter" (96 frames), and "Coast Guard" (96 frames) sequences in our experiments. The first two sequences have a spatial resolution of 352×240 pixels while the last two sequences are 352×288. All sequences are grayscale and have a temporal sampling of 30 frames/sec. (noninterlaced).

In the 3D-RWMH system, the spatial RDWT uses the popular 9-7 biorthogonal filter, while the temporal filtering uses either the lifting 5-3 biorthogonal filter of (2.13), (2.14), (2.15), and (2.16), or the lifting Haar filter of (2.9), (2.10), (2.11), and (2.12). The spatial transform uses symmetric extension at the image boundaries, while the temporal transform uses symmetric extension at each end of the video sequence. Both the spatial and temporal transforms use a decomposition of $J = 3$ levels. For control-point ME, a block size of $B = 17$ is used, and motion vectors are searched in a window of size $W = 15$. The triangle mesh is reset to the uniform mesh every $N' = 4$ frames; this

uniform mesh is the result of diagonal subdividing of 16×16 blocks. Since 3D-SPIHT [4]—the core compression engine in the 3D-RWMH system—produces an embedded coding, the sequence is coded at exactly the specified target rate.

Initially, we assume ME/MC with integer-pixel accuracy. We compare the rate-distortion performance of the 3D-RWMH system to an equivalent spatial-domain MCTF system. Specifically, we compare 3D-RWMH using the 5-3 temporal filter (denoted "3D-RWMH-53"), 3D-RWMH using the Haar temporal filter (denoted "3D-RWMH-Haar"), and a spatial-domain technique (denoted "SD-MCTF-53"). The spatial-domain system performs MCTF in the spatial domain and then subsequently employs a critically sampled spatial transform and embedded coding. In this system, a triangle-mesh ME procedure identical to that of the 3D-RWMH-53 coder is employed, and, like the 3D-RWMH-53 system, temporal decomposition takes place with 5-3 biorthogonal lifting with symmetric extension. This SD-MCTF-53 system is essentially a single-hypothesis version of the 3D-RWMH-53 coder and corresponds roughly to the system of [5], except that the ME process is somewhat different, and 3D-SPIHT, rather than JPEG-2000, is used to code the wavelet coefficients. From the results of Figs. 4.4-4.5 and Table 4.1, we see that the incorporation of multihypothesis into MCTF yields significant improvement in rate-distortion performance in comparison to single-hypothesis MCTF. Additionally, the longer 5-3 temporal filter outperforms the Haar filter in the 3D-RWMH system.

We compare the 3D-RWMH-53 system to the RWMH system described in Sec. 3.3. From Figs. 4.6-4.7 and Table 4.2, we see that 3D-RWMH-53, which features MCTF and is fully scalable, outperforms RWMH, which features an ME/MC feedback loop and its associated difficulties concerning scalability.

Using the 5-3 temporal filter in 3D-RWMH, we compare half-pixel accuracy using interpolation with the 8-tap filter from [10] to bilinear interpolation in Fig. 4.8. We see that the 8-tap filter significantly outperforms bilinear interpolation for half-pixel

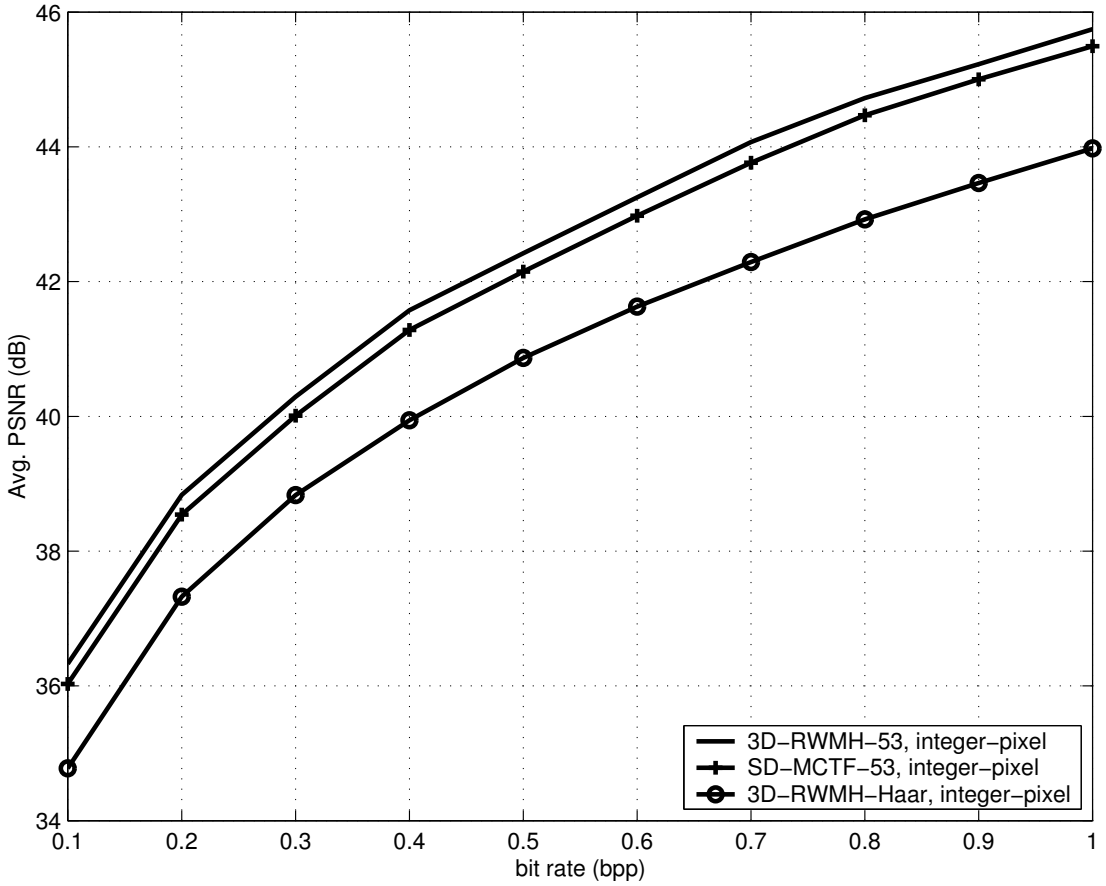


Figure 4.4: Rate-distortion performance for "Susie."

Table 4.1: Distortion averaged over all frames of the sequences.

	PSNR (dB)		
	3D-RWMH-Haar integer-pixel	SD-MCTF-53 integer-pixel	3D-RWMH-53 integer-pixel
Football†	28.3	29.3	29.7
Susie	38.8	40.0	40.3
Mother & daughter	42.6	44.3	44.8
Coastguard	29.7	32.0	32.1

Rate is 0.3 bpp except †, which is 0.5 bpp.

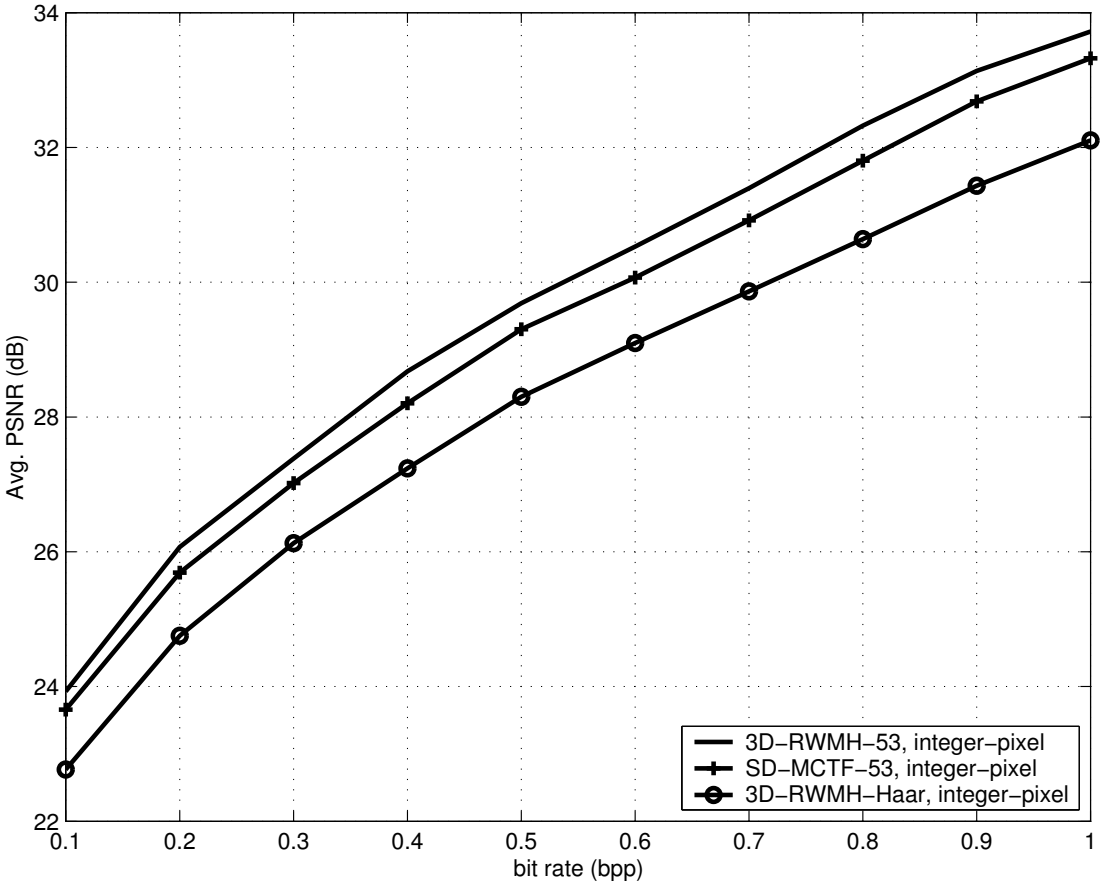


Figure 4.5: Rate-distortion performance for "Football."

Table 4.2: Distortion averaged over all frames of the sequences.

	PSNR (dB)	
	3D-RWMH-53, integer-pixel	RWMH
Football†	29.7	28.6
Susie	40.3	38.5
Mother & daughter	44.8	41.9
Coastguard	32.1	29.5

Rate is 0.3 bpp except †, which is 0.5 bpp.

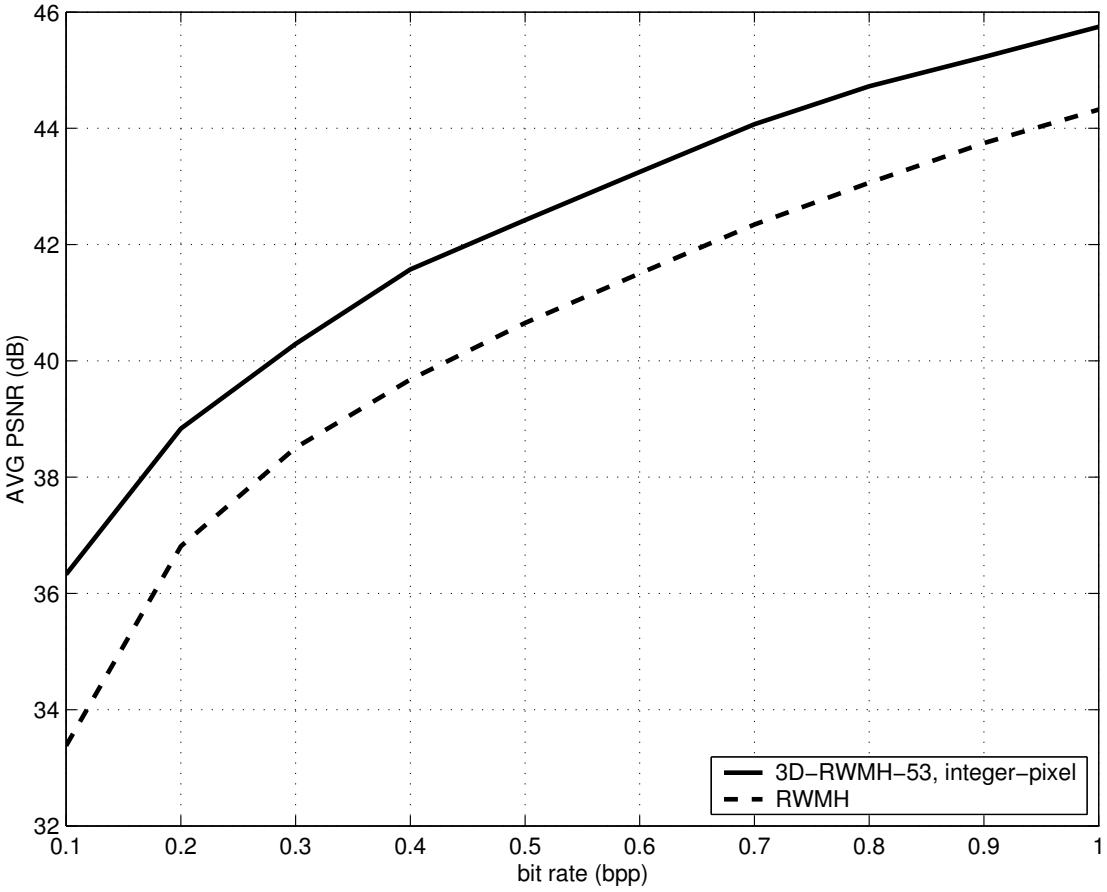


Figure 4.6: Rate-distortion performance for “Susie.”

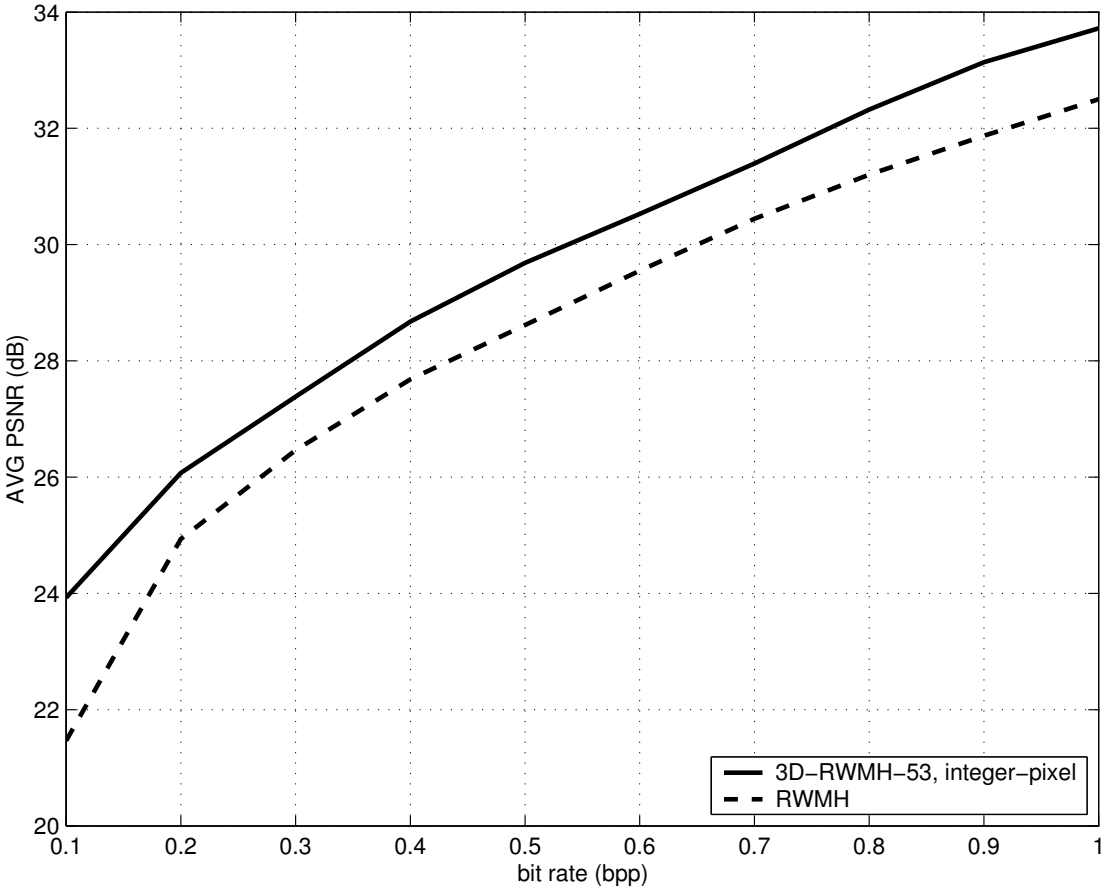


Figure 4.7: Rate-distortion performance for “Football.”

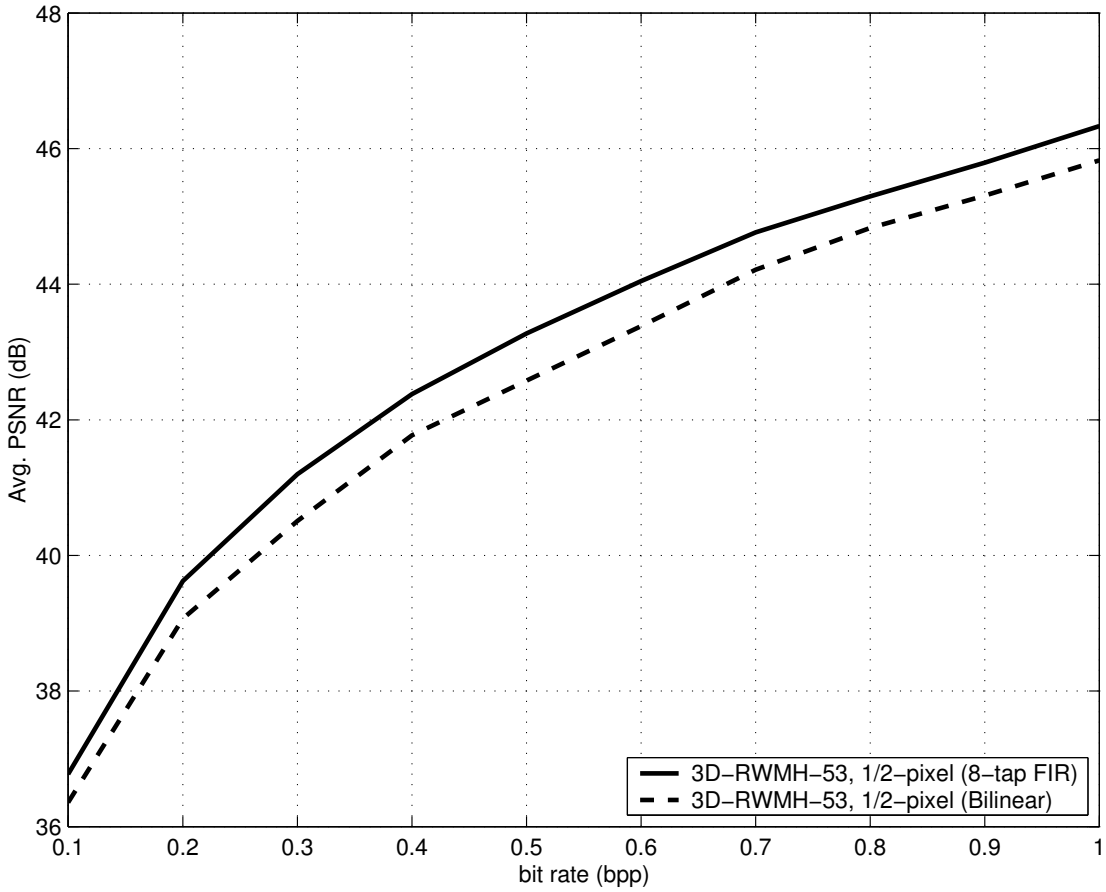


Figure 4.8: Rate-distortion performance for "Susie" for 3D-RWMH with bilinear and filter-based interpolation.

accuracy; subsequently, we use the 8-tap filter exclusively to produce half-pixel values.

As a final body of results, we compare the 3D-RWMH-53 system with half-pixel accuracy to the bidirectional MC-EZBC system from [10]. Bidirectional MC-EZBC is a 3D video coder employing traditional block-based MCTF in the spatial domain and represents perhaps the state-of-the-art of such coders. Temporal filtering is essentially a bidirectional version of the Haar filter, with a lifting implementation providing $\frac{1}{8}$-pixel accuracy for ME/MC and appropriate measures to compensate for "unconnected" pixels. Experimental results are presented in Figs. 4.9-4.12 and in Table 4.3. From these results, we see that 3D-RWMH-53 with half-pixel accuracy outperforms MC-EZBC. In particular, a gain on the order of 0.3 to 0.5 dB over MC-EZBC is seen for all sequences considered.

4.2 Scalability of 3D-RWMH

As stated in Chap. I, a fully scalable video-coding system should provide a high degree of fidelity, spatial, and temporal scalability. In this chapter, we demonstrate that the 3D-RWMH-53 system is a fully scalable video-coding system. Additionally, we present some experimental results comparing the scalability of 3D-RWMH-53 to that of other systems. A 3D embedded, wavelet-based coder, such as 3D-SPIHT or the 3D-Tarp coder to be presented in Chap. VI, is used in 3D-RWMH systems to code the 3D transformed coefficients. 3D-SPIHT, 3D-Tarp, and similar coders give high compression performance, allow fully progressive transmission, and provide an embedded bitstream. These qualities make for a video-coding system with a high degree of fidelity scalability. Consequently, in the remainder of this section, we consider only spatial and temporal scalabilities of 3D-RWMH systems.

Table 4.3: Distortion averaged over all frames of the sequences.

	PSNR (dB)		
	3D-RWMH-53 integer-pixel	MC-EZBC 1/8-pixel	3D-RWMH-53 1/2-pixel
Football†	29.7	29.6	30.1
Susie	40.3	40.9	41.2
Mother & daughter	44.8	45.2	45.5
Coastguard	32.1	32.1	32.4

Rate is 0.3 bpp except †, which is 0.5 bpp.

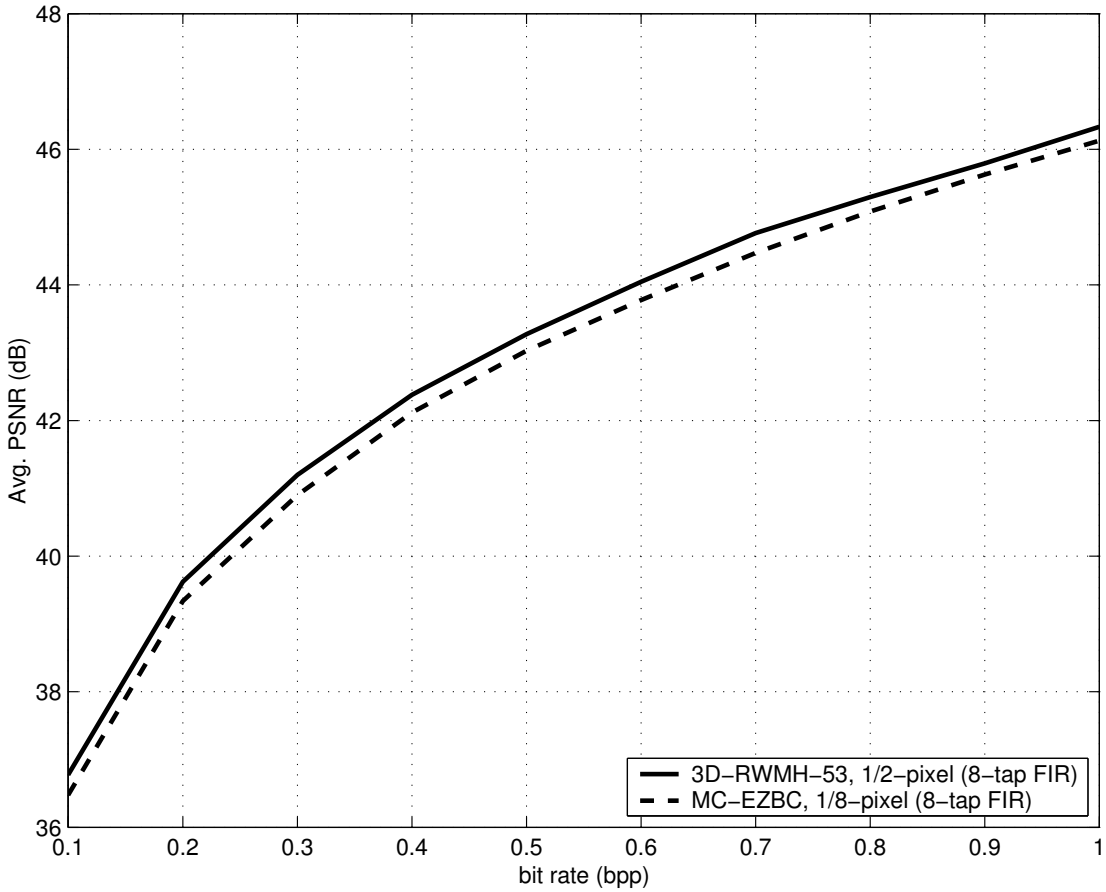


Figure 4.9: Rate-distortion performance for “Susie.”

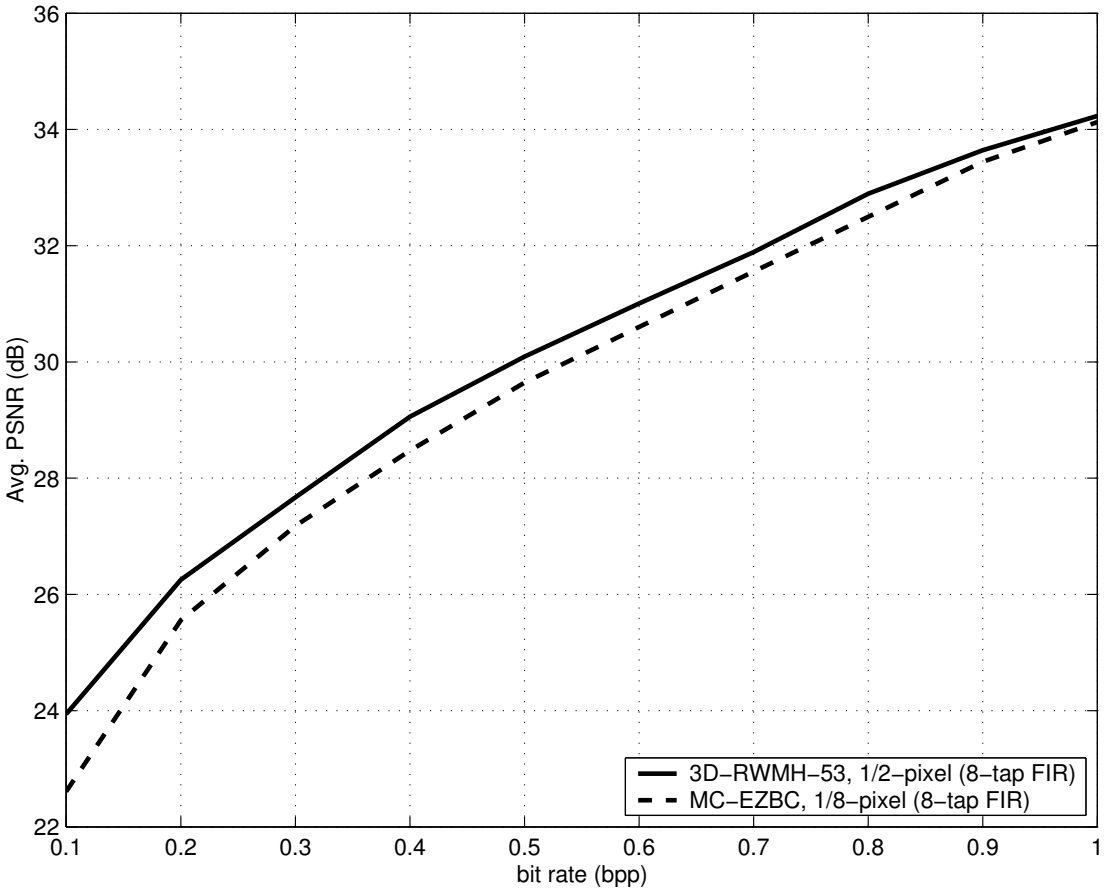


Figure 4.10: Rate-distortion performance for “Football.”

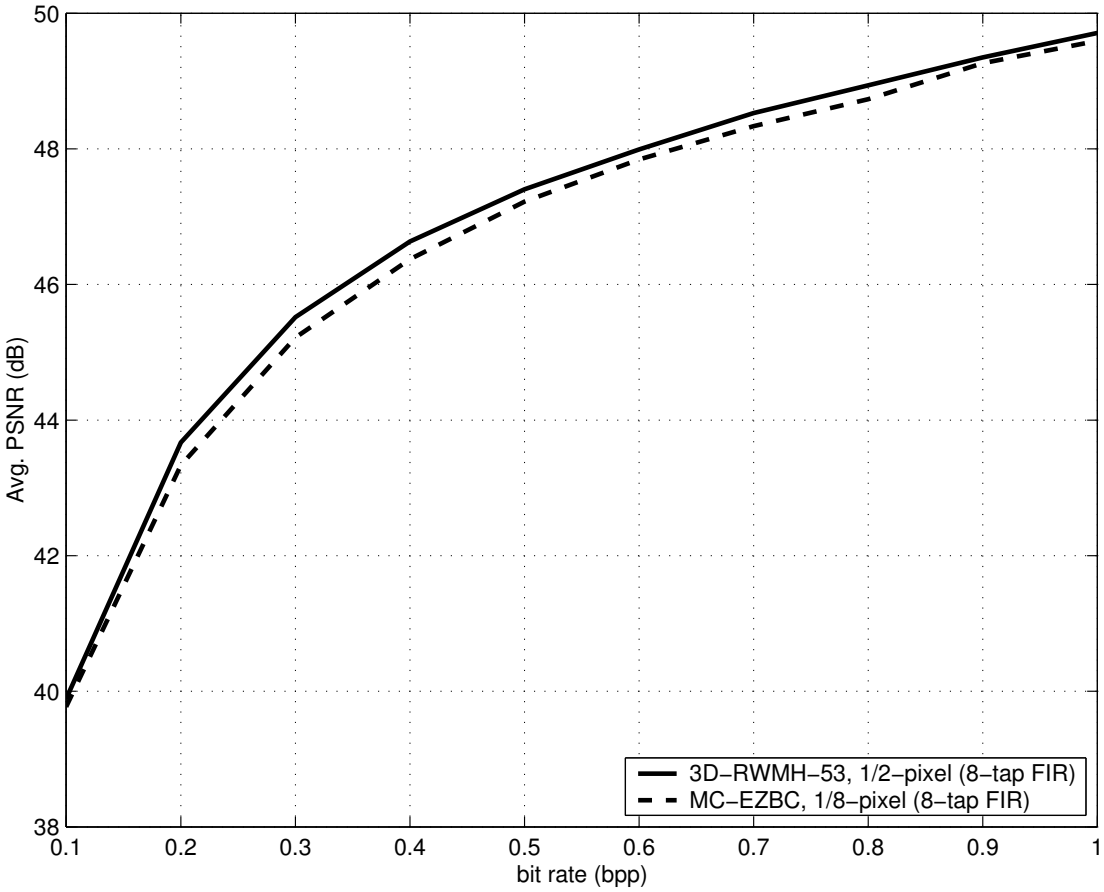


Figure 4.11: Rate-distortion performance for “Mother & Daughter.”

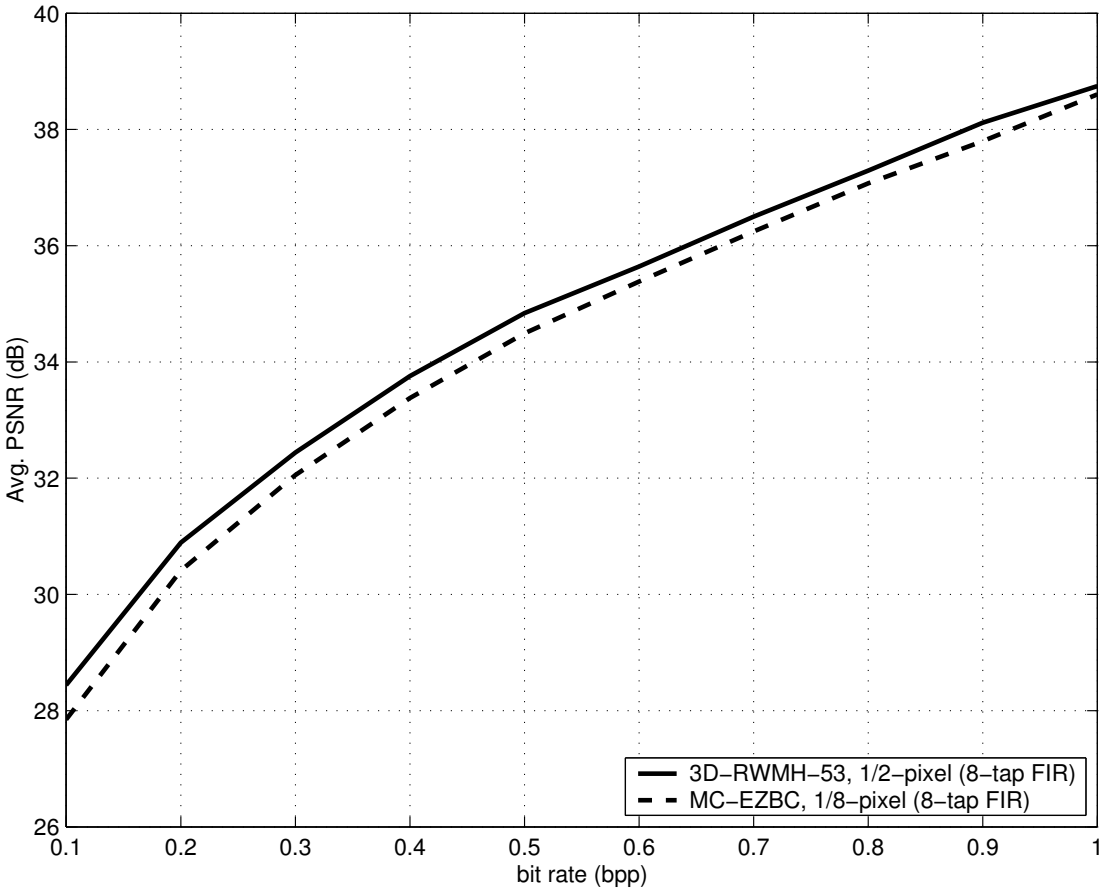


Figure 4.12: Rate-distortion performance for “Coastguard.”

4.2.1 Spatial Scalability

We use the sequences "Susie", "Football", "Mother & Daughter", and "Coastguard" for our experiments. We compare the spatial scalability of 3D-RWMH-53 to that of MC-EZBC. In the experiments, to get a low spatial-resolution video sequence, only the bits necessary for reconstruction at that particular spatial resolution are decoded from the encoded bitstream; that is, only some subsets of the spatial subbands are reconstructed. Then, the inverse MCTF is performed on only the available subbands. Figs. 4.13-4.16 show the experimental results. In these figures, frames of only a quarter of the original size of video images are reconstructed. From the figures, we can see that 3D-RWMH-53 outperforms MC-EZBC for spatial scalability. Individual frames are shown in Figs. 4.17-4.19, from which we can also see that the 3D-RWMH-53 gives better subjective performance.

The issue of determining quality of video reconstructed at reduced spatial resolution is currently a matter of debate within the video-coding community. Unlike the case of full spatial-resolution reconstruction, we do not have "original" frames at reduced spatial resolution to which we can compare our reconstructed frames. Consequently, we must, somewhat arbitrarily, produce a reduced-spatial-resolution version of the original video sequence in order to permit PSNR calculations. In our experiments, we perform a J-scale 2D DWT spatially on each of the original images of a sequence, then retain only the basebands to make a video sequence with size $1/4^J$ of the original size. Alternatively, we could have simply subsampled the original video images. However, this approach does not necessarily represent the most appropriate information for low spatial resolution, particularly in the presence of noise [7].

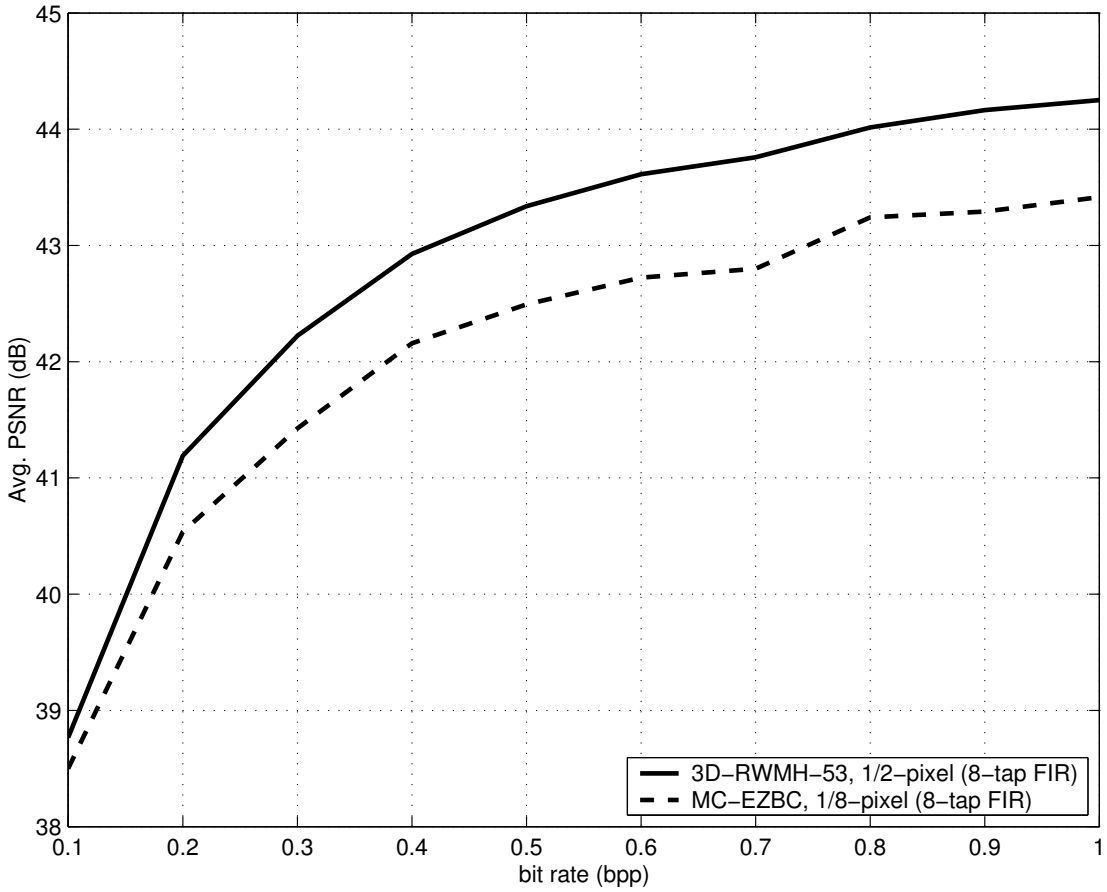


Figure 4.13: Rate-distortion performance of 3D-RWMH-53 for "Susie" decoded at QSIF (176×120). The bit rate is encoding rate.

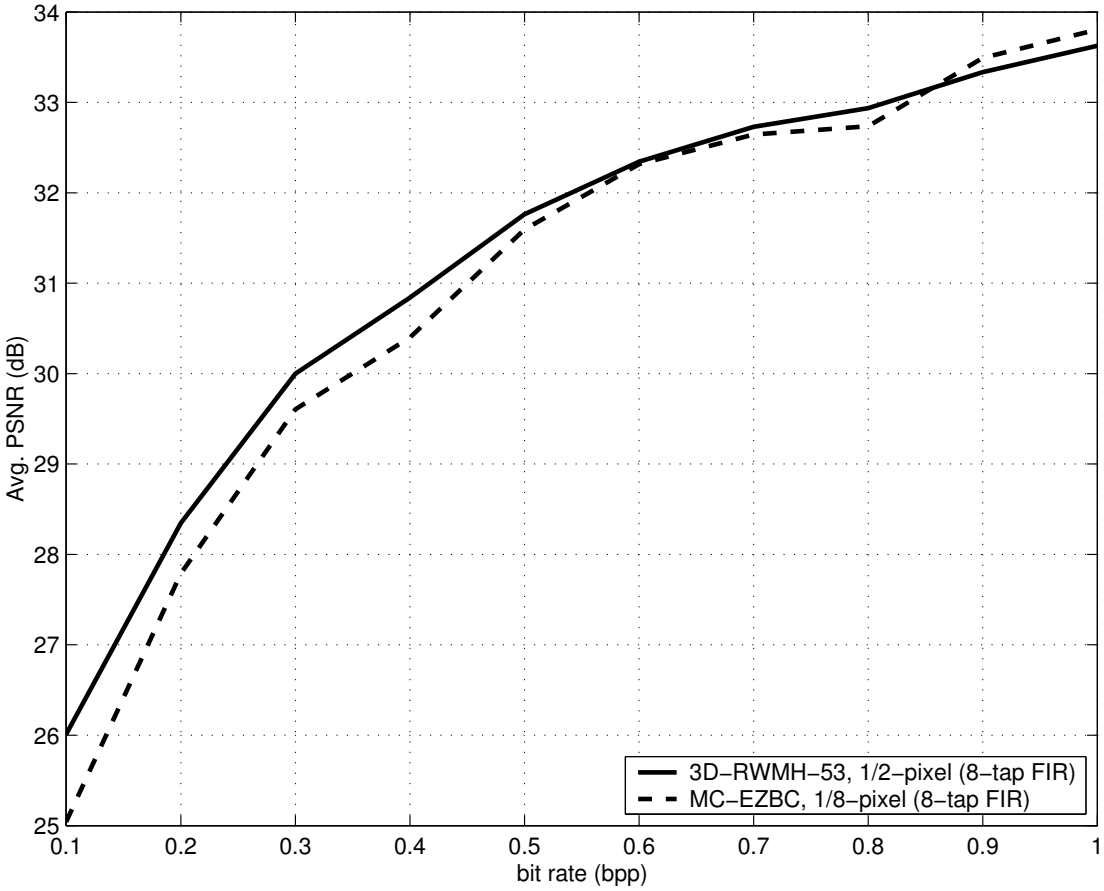


Figure 4.14: Rate-distortion performance of 3D-RWMH-53 for "Football" decoded at QSIF (176×120). The bit rate is encoding rate.

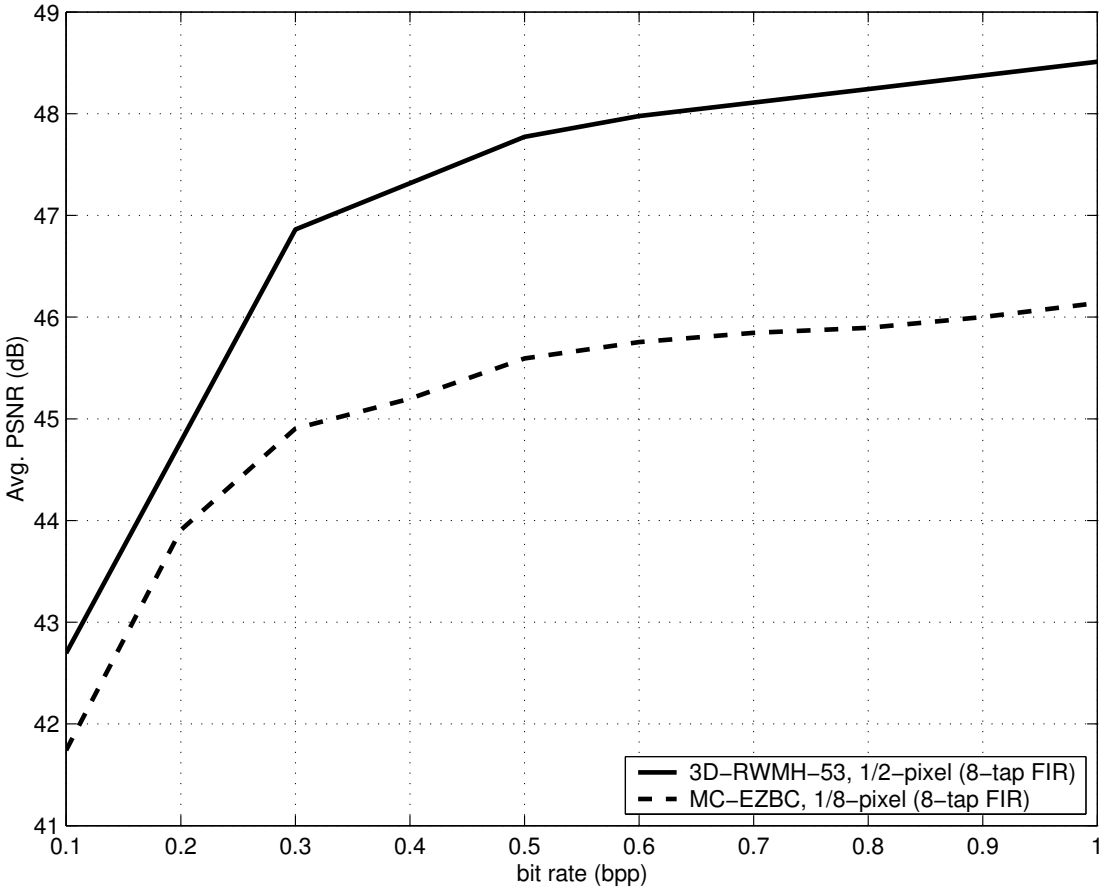


Figure 4.15: Rate-distortion performance of 3D-RWMH-53 for "Mother & Daughter" decoded at QCIF (176×144). The bit rate is encoding rate.

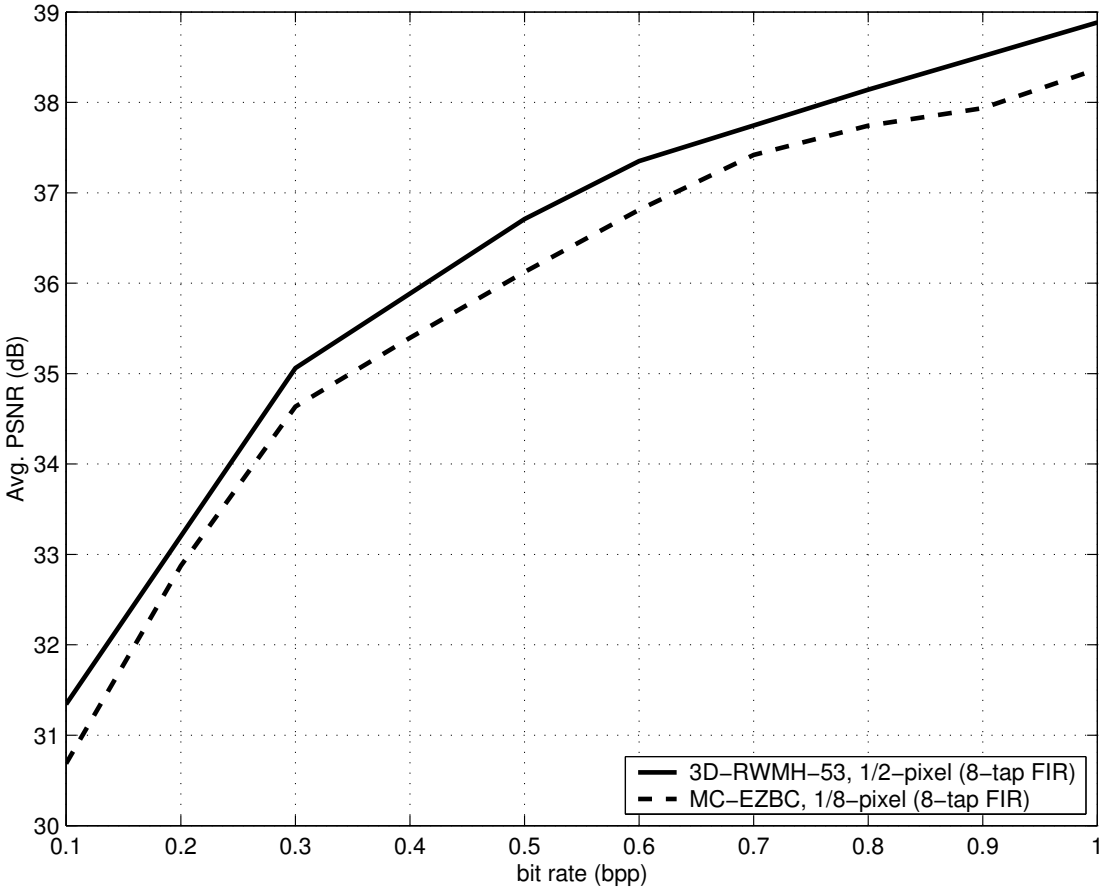


Figure 4.16: Rate-distortion performance of 3D-RWMH-53 for "Coastguard" decoded at QCIF (176×144). The bit rate is encoding rate.

(a)

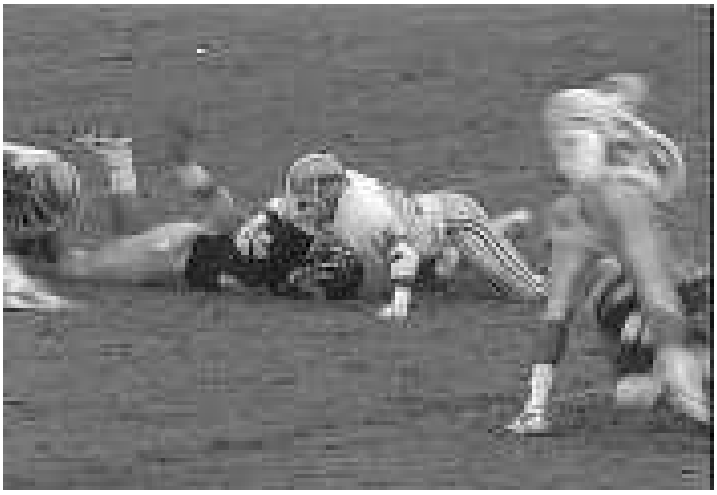
(b)

(c)

Figure 4.17: Original and reconstructed images for frame 5 of "Football" encoded at 0.5 bpp (1.3 Mbps). The images are in SIF size (352 $\times$ 240). (a) Original, (b) MC-EZBC, and (c) 3D-RWMH-53.

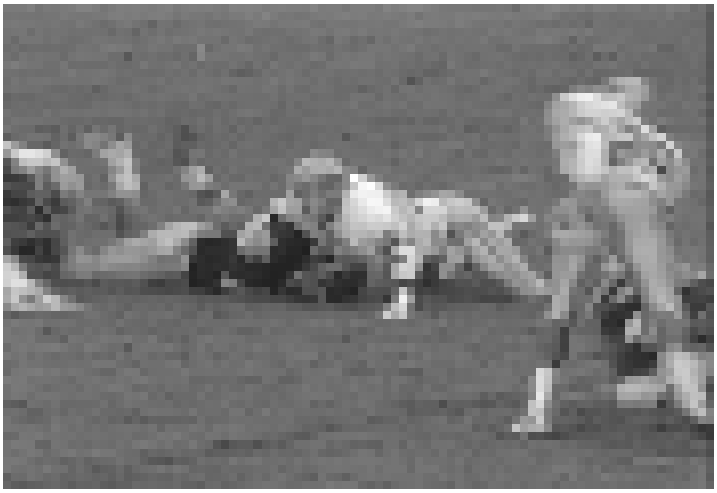
(a)

(b)

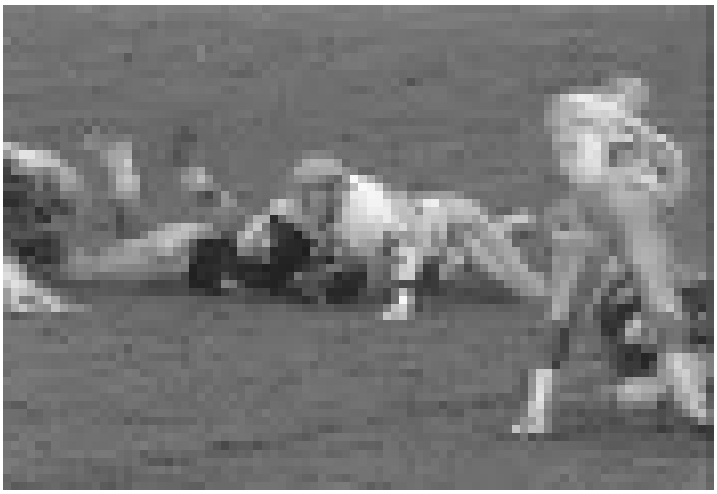


(c)

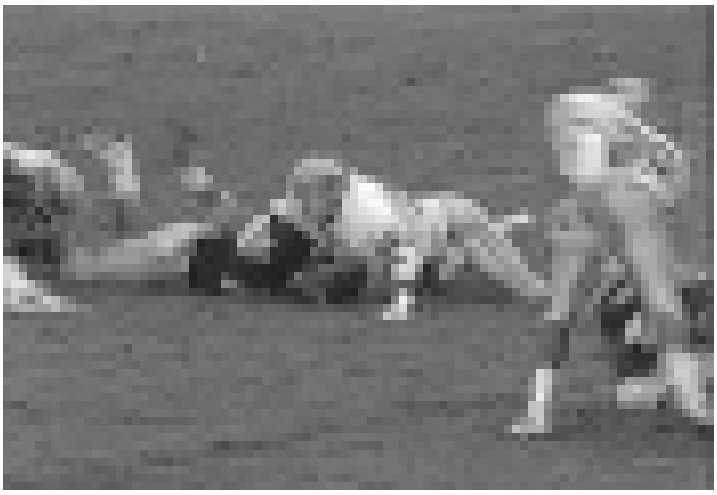
Figure 4.18: Original and reconstructed images for frame 5 of "Football" encoded at 0.5 bpp (1.3 Mbps). The images are in QSIF size (176 $\times$ 120). (a) Original, (b) MC-EZBC, and (c) 3D-RWMH-53.



(a)



(b)



(c)

Figure 4.19: Original and reconstructed images for frame 5 of "Football" encoded at 0.5 bpp (1.3 Mbps). The images are in Q-QSIF size (88 $\times$ 60). (a) Original, (b) MC-EZBC, and (c) 3D-RWMH-53.

4.2.2 Temporal Scalability

Again, we use the sequences "Susie", "Football", "Mother & Daughter", and "Coastguard" for our experiments. These sequences have a frame rate of 30 frames/second. We compare the temporal scalability of 3D-RWMH-53 to that of 3D-RWMH-Haar by decoding only the bits necessary to reconstruct the sequence at a particular reduced temporal resolution; that is, we decode only a subset of the temporal subband frames. Figs. 4.20-4.23 show experimental results for a decoded sequence at 15 frames/second. From the figures, we can see that 3D-RWMH-53 outperforms 3D-RWMH-Haar for temporal scalability due to the fact that 5-3 biorthogonal temporal filtering is essentially a bi-directional motion-compensated prediction, while Haar temporal filtering is single-directional prediction. Some reconstructed images are shown in Fig. 4.24. We can see that 3D-RWMH-53 outperforms 3D-RWMH-Haar in perceptual quality. In Fig. 4.24(b), we can observe significant "ghost" artifacts around the fast moving players, while in Fig. 4.24(c), the "ghost" artifacts are eliminated by the bi-directional 5-3 biorthogonal temporal filtering. The longer temporal filter produces higher quality frames at reduced temporal resolution since the bi-directional prediction is inherently more capable of compensating for rapid motion.

Like the situation previously encountered when evaluating performance at reduced spatial resolution, the method for quantitatively evaluating quality at reduced temporal resolution is an open issue, since the original video sequences exist at only 30 frames/second. However, video sequences are viewed using a "sample-and-hold" approach for each frame; i.e., each frame is displayed for some fixed amount of time. Consequently, it is reasonable to consider the "original" sequence at reduced resolution to be a temporal downsampling of the full-temporal-resolution sequence.

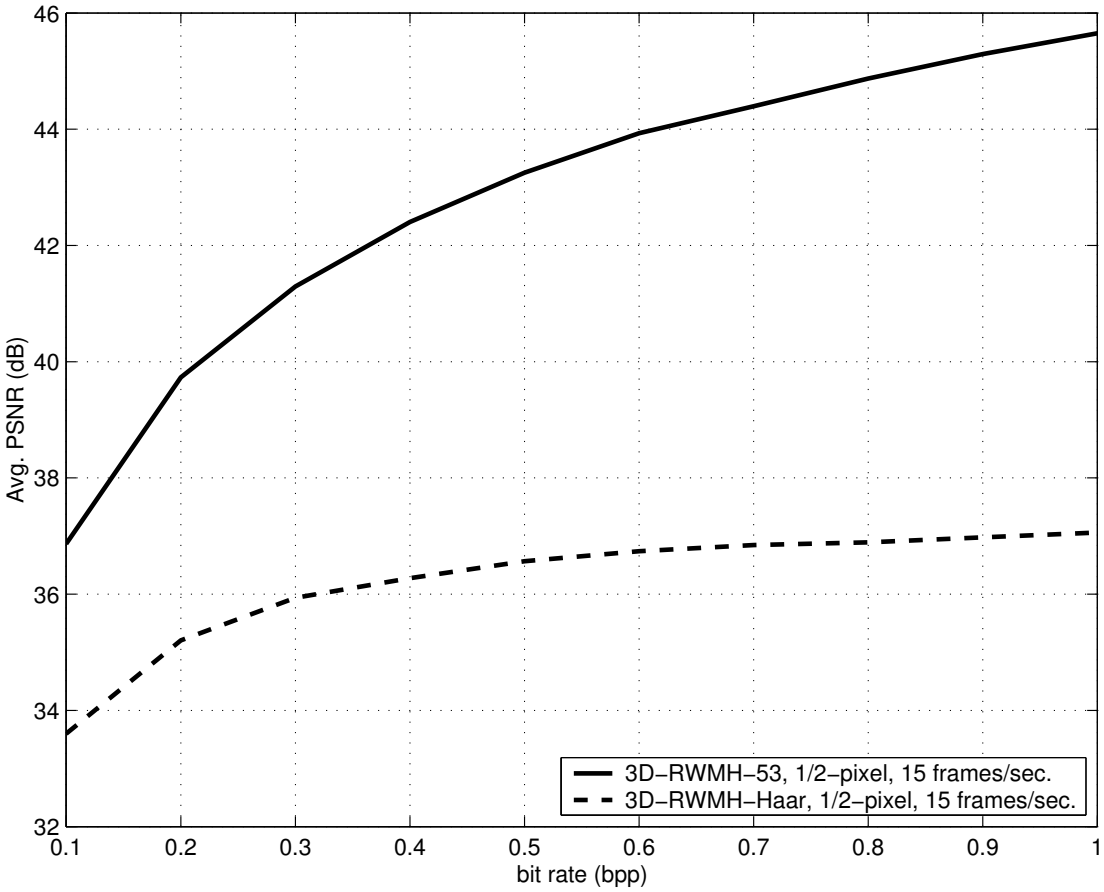


Figure 4.20: Rate-distortion performance of 3D-RWMH-53 for "Susie" decoded at 15 frames/sec. The bit rate is encoding rate.

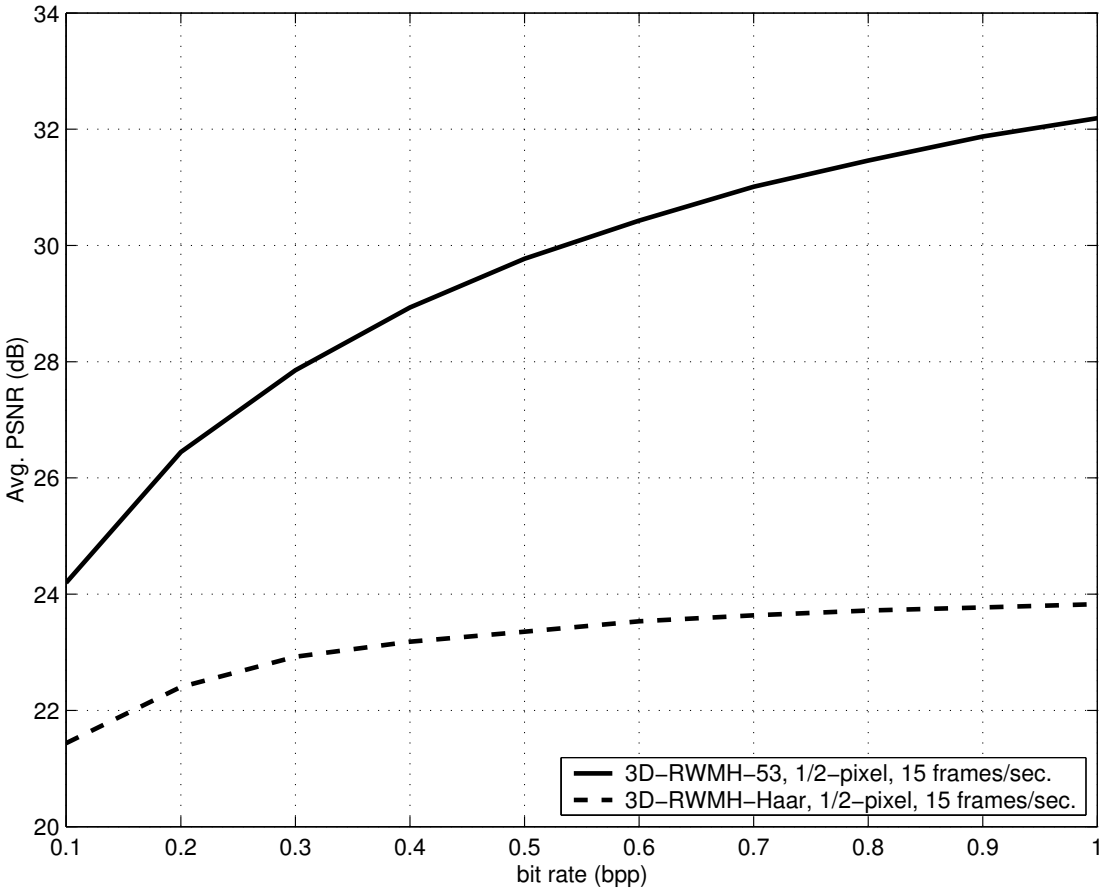


Figure 4.21: Rate-distortion performance of 3D-RWMH-53 for "Football" decoded at 15 frames/sec. The bit rate is encoding rate.

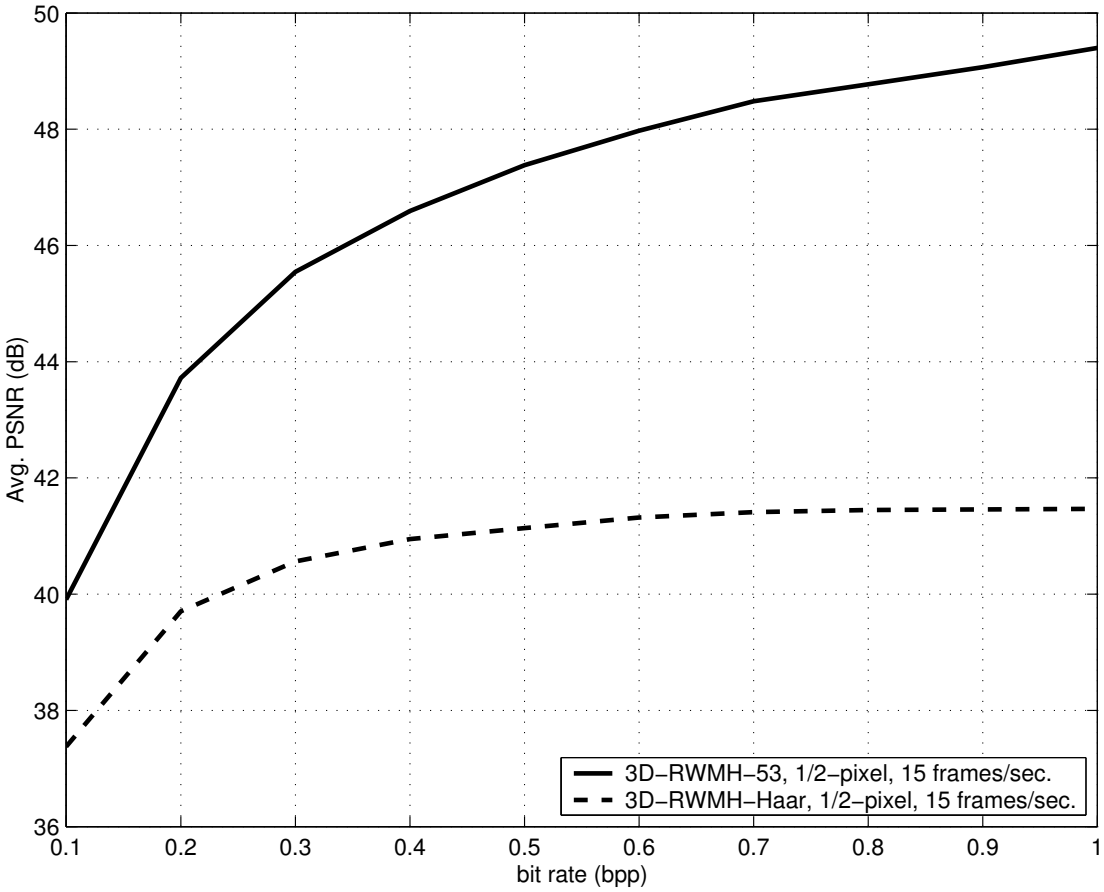


Figure 4.22: Rate-distortion performance of 3D-RWMH-53 for "Mother & Daughter" decoded at 15 frames/sec. The bit rate is encoding rate.

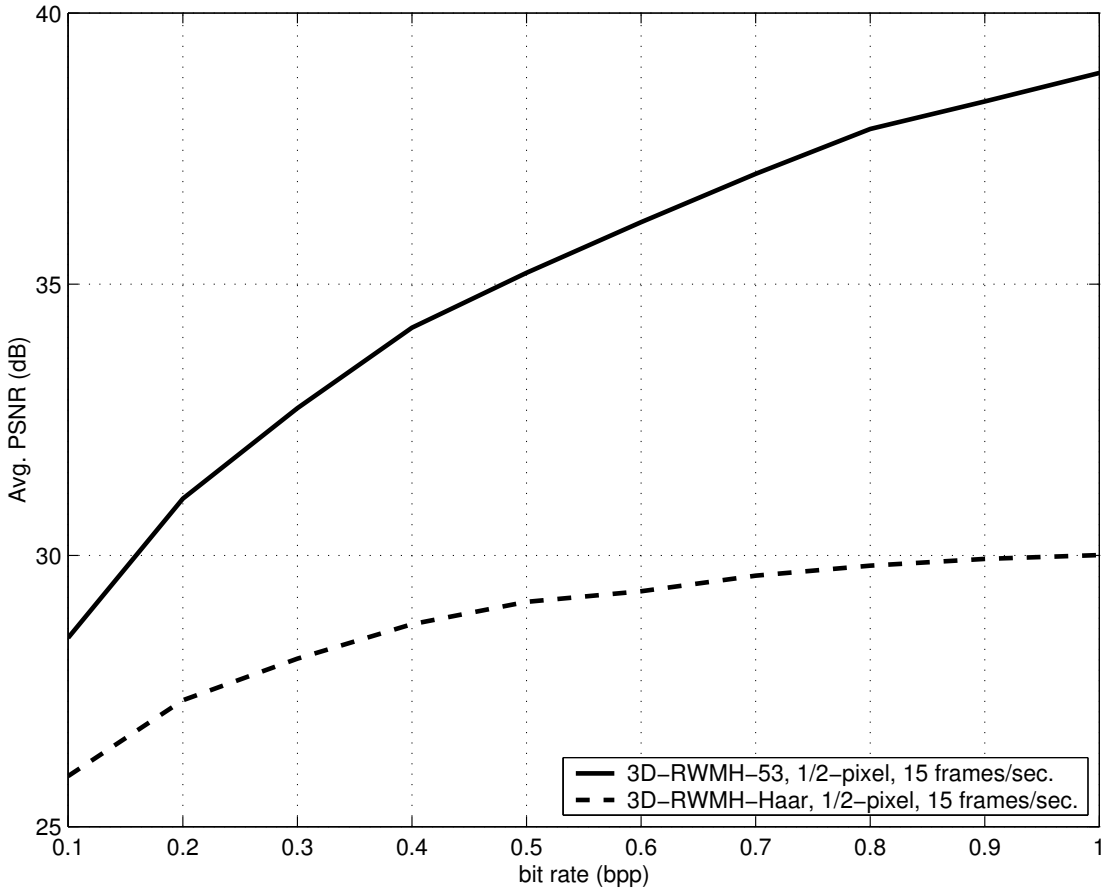


Figure 4.23: Rate-distortion performance of 3D-RWMH-53 for "Coastguard" decoded at 15 frames/sec. The bit rate is encoding rate.

(a)

(b)

(c)

Figure 4.24: Original and reconstructed images for frame 5 of "Football" encoded at 0.5 bpp (1.3 Mbps), decoded at 7.5 frames/sec. (a) Original, (b) 3D-RWMH-Haar, (c) 3D-RWMH-53.

CHAPTER V

BOUNDARY EFFECTS IN 3D-RWMH

In generating the results of the previous chapter, all frames of a given video sequence were processed in one large segment. That is to say, all the input frames were collected as a single group of frames (GOF), MCTF was performed on this single GOF, and the entire GOF was coded by the 3D codec (3D-SPIHT). This approach is reasonable for algorithm simulation but not for real-world applications. Since video sequences may be arbitrarily long, using a single GOF has a potentially long delay and impractically large memory usage, which are major impediments to real-world video applications. Consequently, for a practical system, a limited GOF length could be used, with the video sequence "blocked" into multiple GOFs of some finite size. However, a serious problem, "boundary effects," is associated with finite GOF length and biorthogonal temporal filtering [15, 20].

In temporal filtering, when temporal-transform filters longer than the simple Haar filter are used with a finite GOF length, symmetric extension is commonly used at each temporal boundary of each GOF to accommodate the overlap of the filter's response beyond the extent of the GOF. Consequently, GOF boundary effects arise in such systems due to a significant decrease in PSNR near the GOF boundaries as the temporal filters begin to extend beyond the GOF boundaries. This is a well-known problem which was investigated and resolved in [15, 20] by extending the temporal filtering indefinitely in time. Specifically, the authors of [20] remove the symmetric extension by introducing a sliding window into the temporal transform. That is, a finite frame buffer holds all frames necessary for calculating the next temporal subbands; as frames are no

longer needed, they are replaced in the buffer in a first-in-first-out fashion by succeeding frames. Thus, from the perspective of temporal filtering, the GOFs no longer exist, and symmetric extension will not be necessary, except at the very ends of the video sequence.

In this chapter, we first present a detailed review of the sliding-window approach to temporal filtering as proposed in [20]. Then, since [20] considers only a simple form of temporal filtering in which no motion is involved, we extend the sliding-window concept to MCTF in Sec. 5.2. We note that, although [46] proposes a similar sliding-window MCTF approach, our technique was developed independently, and Sec. 5.2 provides a more detailed description of the approach than that available in [46].

5.1 Sliding Window

The sliding window of [20] is implemented by buffering frames when performing the temporal transform. Specifically, for a one-scale temporal wavelet decomposition, n frames need to be buffered for one output frame, where n depends on the wavelet filters used. For the biorthogonal 9-7 filter, n is 5; for the 5-3 filter, n is 3. Each time a frame is input, the frames in the buffers are updated, and one frame is output as either a highpass frame or a lowpass frame according to the lifting scheme. For an L-scale decomposition, where $L > 1$, a larger frame buffer is needed. With the sliding window, the input video frames are treated essentially as one GOF, and the boundary effects will only exist at the ends of the entire sequence. Thus, in essence, the temporal filtering of this sliding-window method is equivalent to that of the system described in Sec. 4.1.

5.2 Eliminating Boundary Effects in MCTF

As discussed before, boundary effects come from the symmetric extension of temporal wavelet filters at the boundary of a finite-length GOF. By buffering only several frames, sliding window [15, 20] eliminates boundary effects and does not require

buffering the entire video sequence. However, MC was not considered in the original formulation [20]. In our SW-3D-RWMH-53 (sliding-window 3D-RWMH-53) system, we consider MC with triangle-based affine transforms. Below, we will describe the sliding-window technique in RDWT-domain MCTF in detail.

In our SW-3D-RWMH-53 system, the 5-3 biorthogonal filter is used for the temporal wavelet transform. Fig. 5.1 illustrates a single-scale wavelet transform on a sequence of frames using the 5-3 biorthogonal filter via lifting steps with symmetric extension at each end of the signal. Figs. 5.2 and 5.3 illustrate how the transform of Fig. 5.1 is modified to become the sliding-window implementation of the 5-3 temporal transform, which we describe below. First, however, let us note that, in the SW-3D-RWMH-53 system, ME/MC is performed in the RDWT domain; i.e., motion compensation is performed between corresponding RDWT subbands. Thus, each frame buffer in the SW-3D-RWMH-53 system contains N RDWT subbands. In the following, we use the notation, $S_i(j)$, to denote frame buffer j for subband S at temporal transform level i, $S \in \{B, H, V, D\}$. Since the sliding window procedure is identical for all the RDWT subbands in each temporal level, we present the sliding-window operation for only one subband.

Figs. 5.2 and 5.3 illustrate the sliding-window pipeline operations via 5-3 lifting steps in the SW-3D-RWMH-53 system. Specifically, for a 1-level temporal transform, when a subband of a frame is pushed into buffer $S_1(2)$, $S_1(0)$ is output as a highpass or lowpass frame, $S_1(0)$ and $S_1(1)$ are updated, and the input subband is pushed into $S_1(2)$. If the input frame is even numbered, as in Fig. 5.2, $S_1(0)$ and $S_1(1)$ are updated

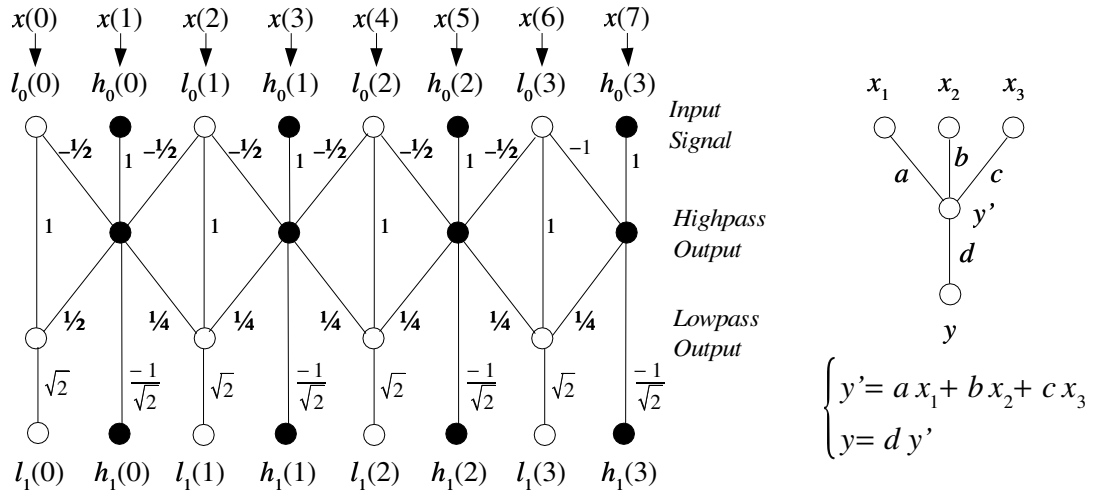


Figure 5.1: Schematic representation of one-scale 5-3 lifting on a sequence of frames. White circles indicate lowpass frames and black circles indicate highpass frames.

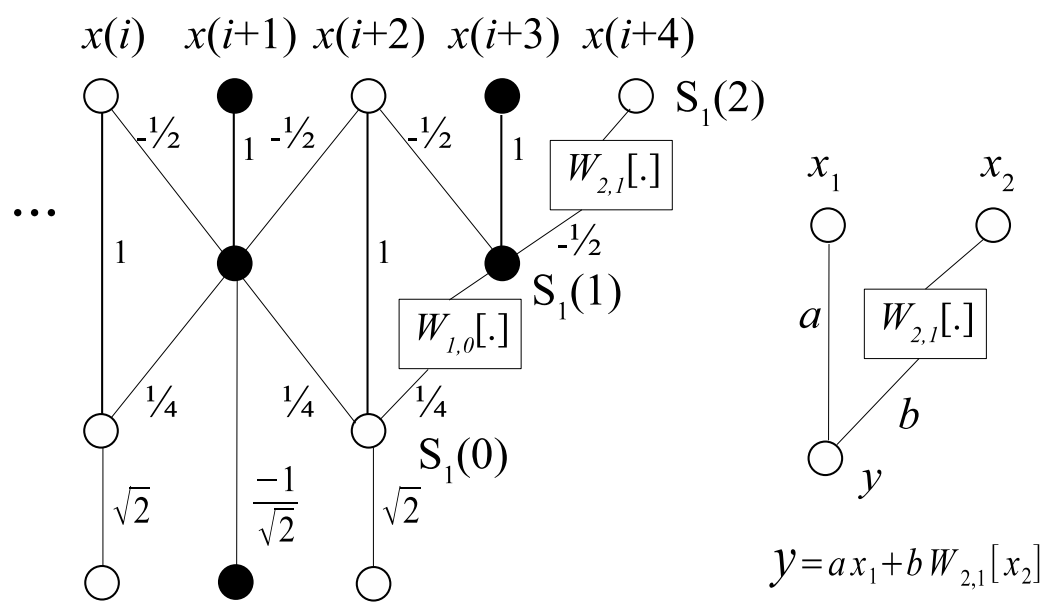


Figure 5.2: The updating operations on the buffer after a subband of an even numbered frame is pushed into $S_1(2)$. $S_i(j)$ is the buffer j for subband S at transform level i. $W_{i,j}$ is the operator that maps frame in buffer i onto the coordinate system of frame in buffer j through the particular MC scheme of choice.

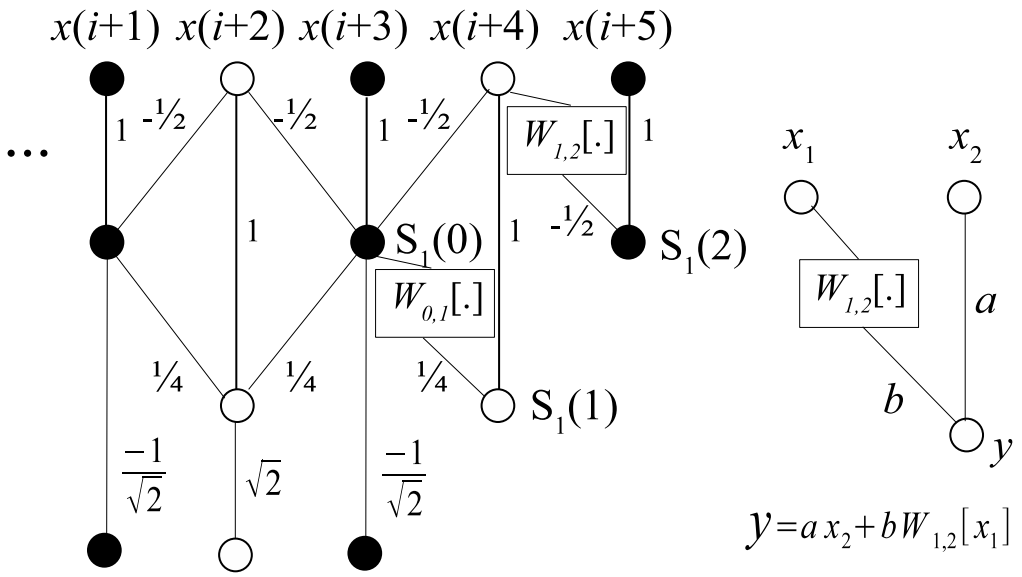


Figure 5.3: The updating operations on the buffer after a subband of an odd numbered frame is pushed into $S_1(2)$. $S_i(j)$ is the buffer j for subband S at transform level i. $W_{i,j}$ is the operator that maps frame in buffer i onto the coordinate system of frame in buffer j through the particular MC scheme of choice.

as

$$S_1(1) \leftarrow S_1(1) - \frac{1}{2}W_{2,1}[S_1(2)], \tag{5.1}$$

$$S_1(0) \leftarrow S_1(0) + \frac{1}{4}W_{1,0}[S_1(1)], \tag{5.2}$$

where $W_{i,j}$ denotes the operator that maps frame i onto the coordinate system of frame j through the particular MC scheme of choice. If the input frame is odd numbered, as in Fig. 5.3, $S_1(2)$ and $S_1(1)$ are updated as

$$S_1(2) \leftarrow S_1(2) - \frac{1}{2}W_{1,2}[S_1(1)], \tag{5.3}$$

$$S_1(1) \leftarrow S_1(1) + \frac{1}{4}W_{0,1}[S_1(0)]. \tag{5.4}$$

Inevitably, at the very ends of the video sequence, symmetric extension is still needed. This procedure can be used for higher levels of temporal transform in which the lowpass frames of one level are passed as the input frames for the next level of the temporal wavelet transform.

Now let us consider the motion fields required for MCTF. For MCTF techniques implemented with multiple finite-length GOFs, the input video sequence is divided into GOFs, and each GOF is coded independently of other GOFs. Assuming that a GOF contains G frames, this ME process results in $G - 1$ motion fields, each mapping the first frame of the GOF into one of the other $G - 1$ frames, as illustrated in Fig. 5.4, which is the same number of motion fields as that of a traditional MCP coder, even if longer wavelet filters are used. With these $G - 1$ motion fields, we can construct affine transforms between any two frames within one GOF by concatenating motion fields from this set. Multiple-scale temporal transforms are also supported since some

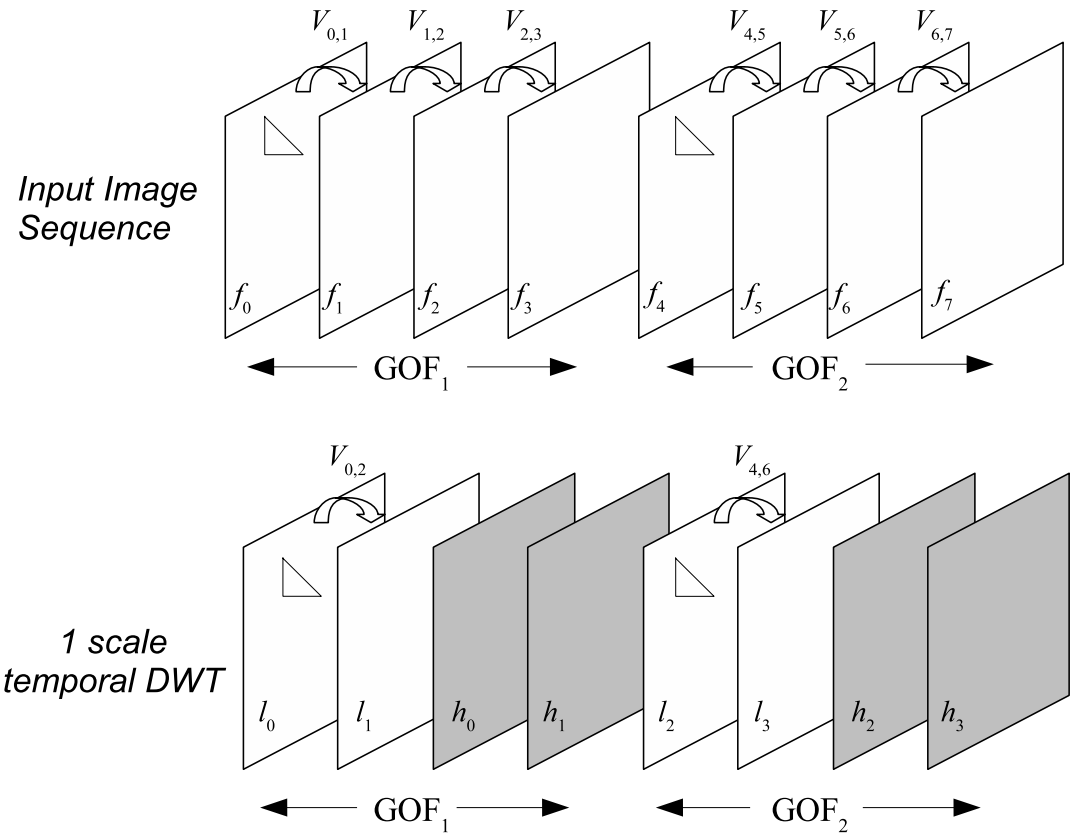


Figure 5.4: Motion fields for MCTF implemented with multiple finite-length GOFs for $G = 4$ and $N' = 4$. A small triangle in a frame indicates where the triangle mesh is reset to the uniform mesh. Field $V_{i,j}$ maps frame i to frame j; concatenated fields are $V_{0,2} = V_{0,1} + V_{1,2}$ and $V_{4,6} = V_{4,5} + V_{5,6}$.

of these motion fields can be directly used for affine transforms in the higher scales, as is illustrated in Fig. 5.4.

On the other hand, if we eliminate boundary effects by using the sliding-window technique for the temporal transform, the temporal filtering is no longer bound to finite-length GOFs. However, we still must use a 3D coder (such as 3D-SPIHT) to produce a bitstream, so we must group temporal subbands into finite-sized groups for purposes of coding. Consequently, an extra motion field is needed to track motion from GOF to GOF, as illustrated in Fig. 5.5. resulting in G motion fields for each GOF (except the very last GOF) of the video sequence. The affine transforms are exactly the same as described in Sec. 2.3.1, and motion-vector information for the G motion fields is transmitted to the decoder. Specifically, for each GOF, all the motion-field information is coded first, followed by an embedded coding of 3D subband coefficients.

5.3 Experimental Results

We set the GOF size to 16 for the following experiments. From Figs. 5.6 and 5.7, we can see that 3D-RWMH-53 with finite-length GOF exhibits significant boundary effects at GOF boundaries, with the PSNR decreasing sharply around frames 16, 32, 48, 64, etc. We see from Figs. 5.6 and 5.7 that using a sliding window for the temporal transform eliminates the boundary effects. Additionally, the sliding window yields a higher average PSNR. The rate-distortion curves for “Susie”, “Football”, “Mother & Daughter”, and “Coastguard” are shown in Figs. 5.8-5.11, and PSNR values averaged over all frames of the sequences are tabulated in Table 5.1 for a fixed bit rate. For all the sequences, SW-3D-RWMH-53 achieves on the order of a 0.4-dB gain over 3D-RWMH-53 with a finite-length GOF. Additionally, SW-3D-RWMH-53 gives almost the same rate-distortion performance as 3D-RWMH-53 using a single GOF. That is, incorporating

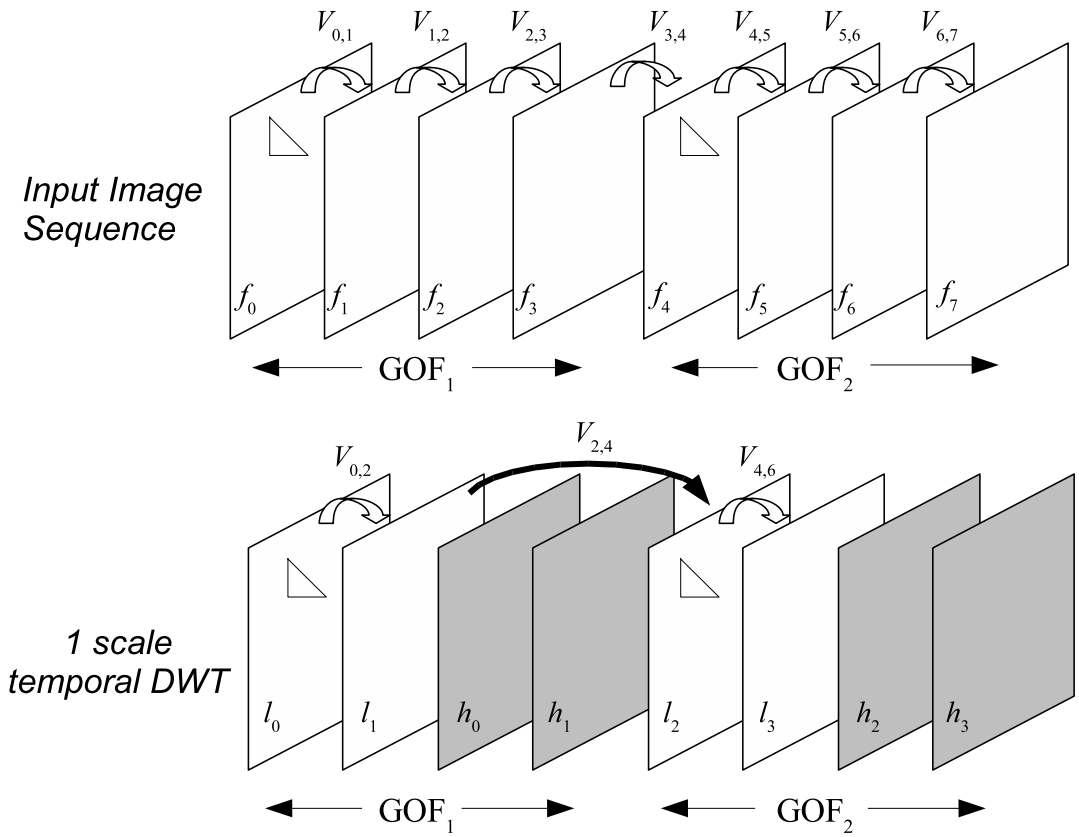


Figure 5.5: Motion fields for MCTF implemented with a sliding window for $G = 4$ and $N' = 4$. A small triangle in a frame indicates where the triangle mesh is reset to the uniform mesh. Field $V_{i,j}$ maps frame i to frame j; concatenated fields are $V_{0,2} = V_{0,1} + V_{1,2}$, $V_{2,4} = V_{2,3} + V_{3,4}$, and $V_{4,6} = V_{4,5} + V_{5,6}$. GOFs are used only for coding of coefficients (e.g. with 3D-SPIHT).

the sliding window as needed for practical use results in negligible performance loss compared to the single-GOF coding.

In the past several chapters, we have focused on MCTF, exploring the use of redundant wavelet expansions in the temporal-filtering process. However, regardless of the mechanisms employed for temporal filtering, ultimately a bitstream must be produced by some 3D coefficient coder. In the previous chapters, results have been carried out using 3D-SPIHT, which is a prominent 3D coder considered by many to be state of the art. In the next chapter, we develop an alternative 3D coder with rate-distortion performance similar to that of 3D-SPIHT but which is more amenable to hardware implementation.

Table 5.1: Distortion averaged over all frames of the sequences.

	PSNR (dB)		
	3D-RWMH-53 single GOF	3D-RWMH-53 GOF size 16	SW-3D-RWMH-53
Football†	30.1	29.7	30.0
Susie	41.2	40.8	41.2
Mother & daughter	45.5	44.9	45.4
Coastguard	32.4	31.9	32.4

All systems use 1/2-pixel ME with 8-tap filter interpolation.
Rate is 0.3 bpp except †, which is 0.5 bpp.

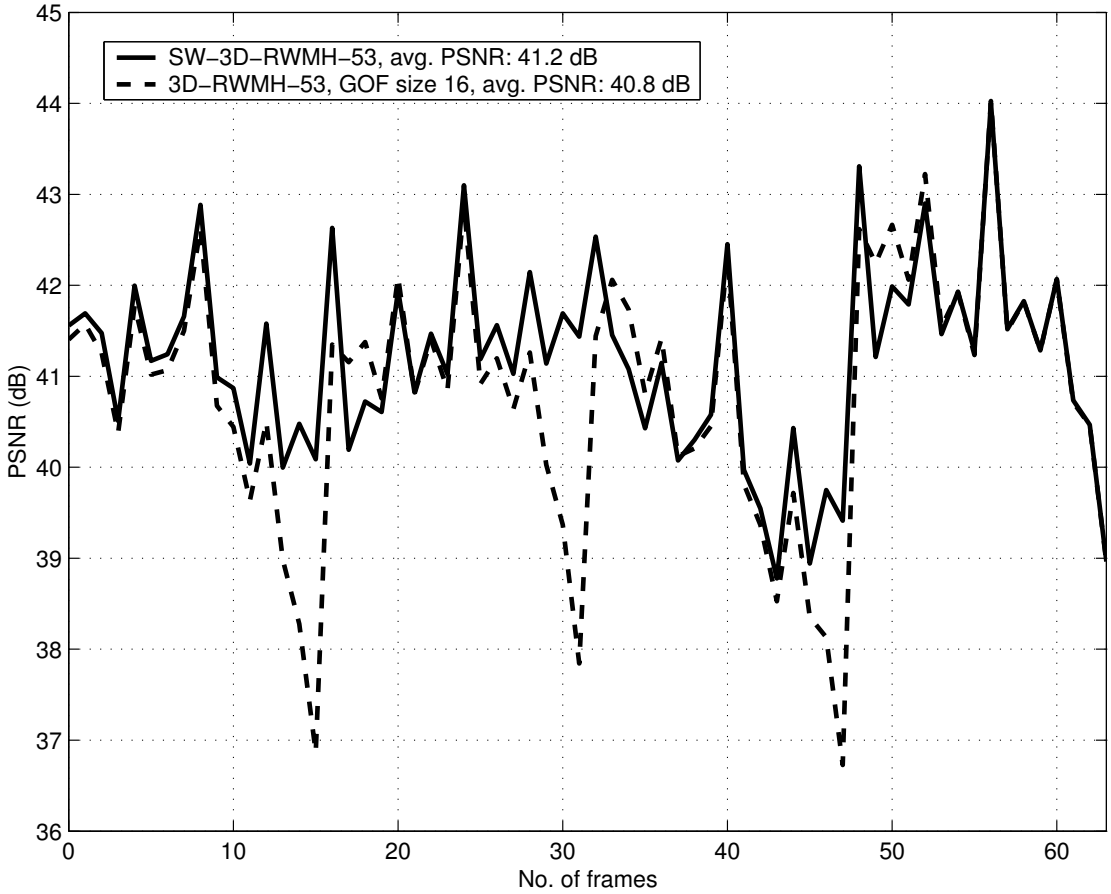


Figure 5.6: Frame-by-Frame PSNR for "Susie" at 0.3 bpp (760 kbps) comparing 3D-RWMH-53 with sliding window and 3D-RWMH-53 with small GOF (16 frames).

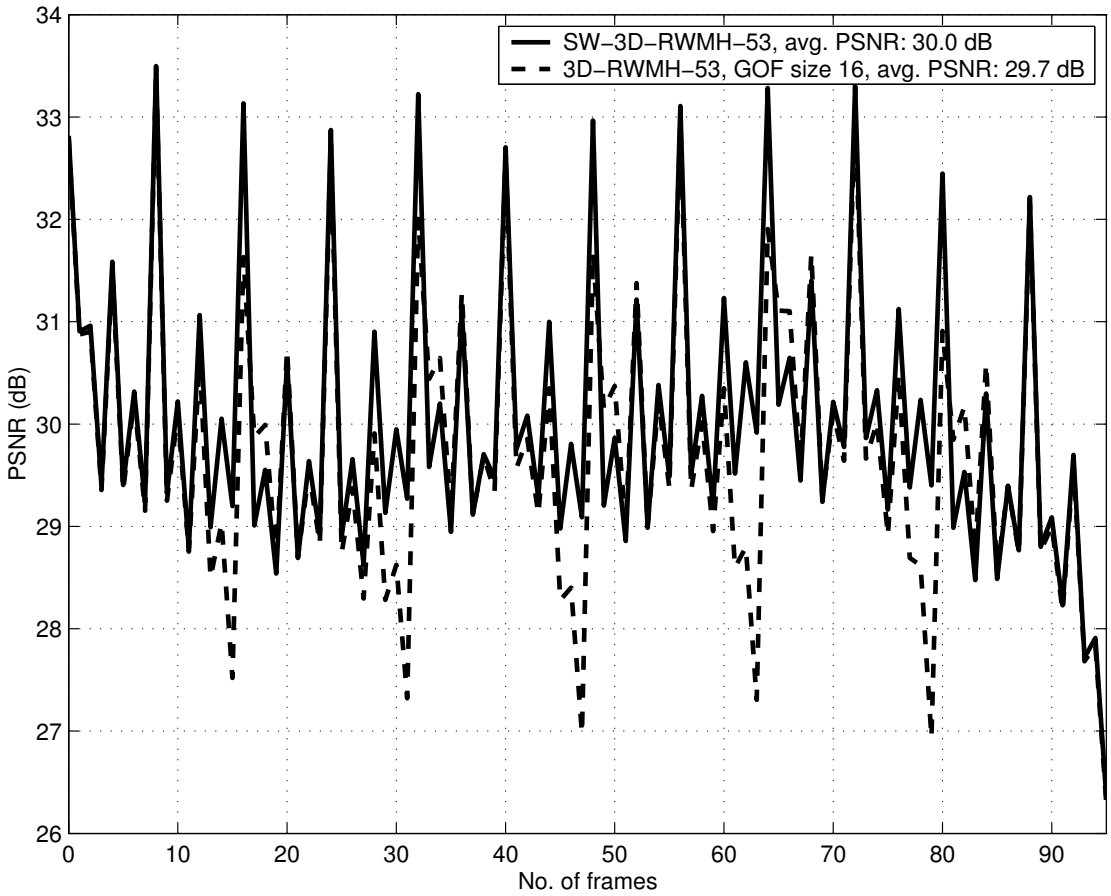


Figure 5.7: Frame-by-Frame PSNR for "Football" at 0.5 bpp (1.3 Mbps) comparing 3D-RWMH-53 with sliding window and 3D-RWMH-53 with small GOF (16 frames).

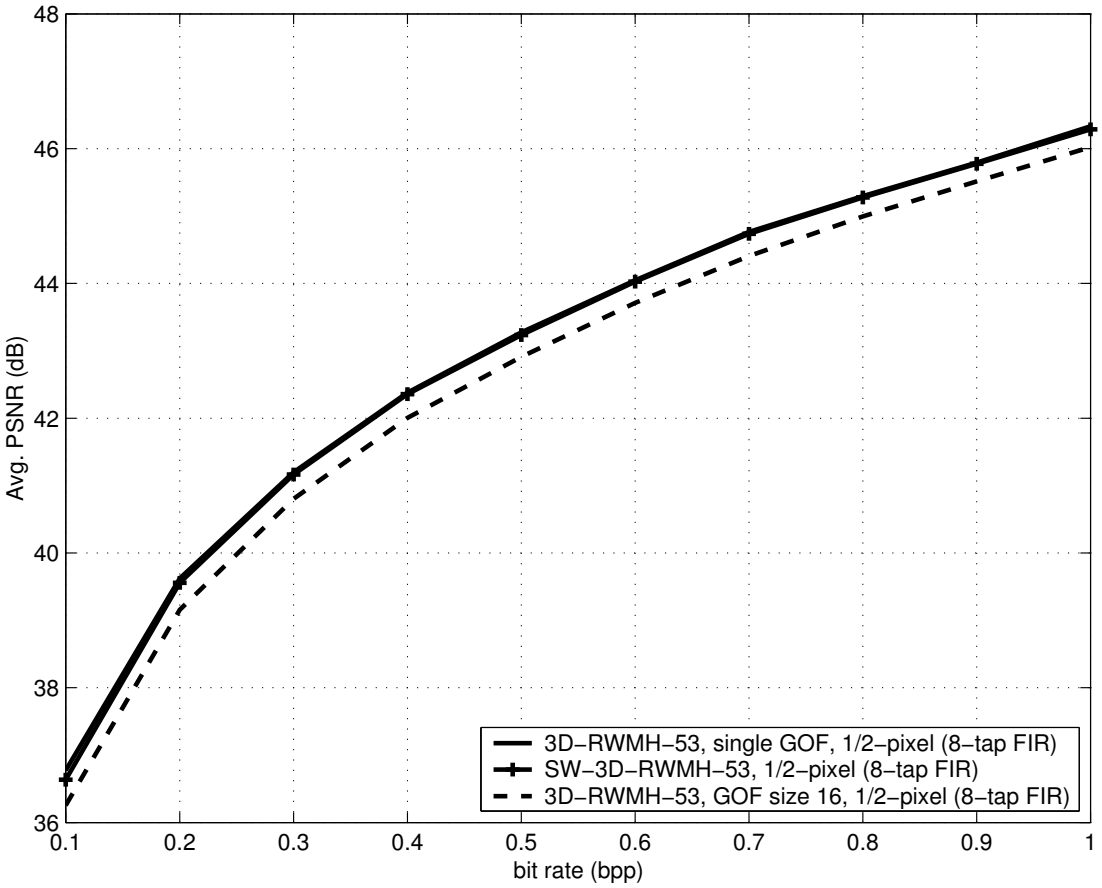


Figure 5.8: Rate-distortion performance of SW-3D-RWMH-53 for "Susie."

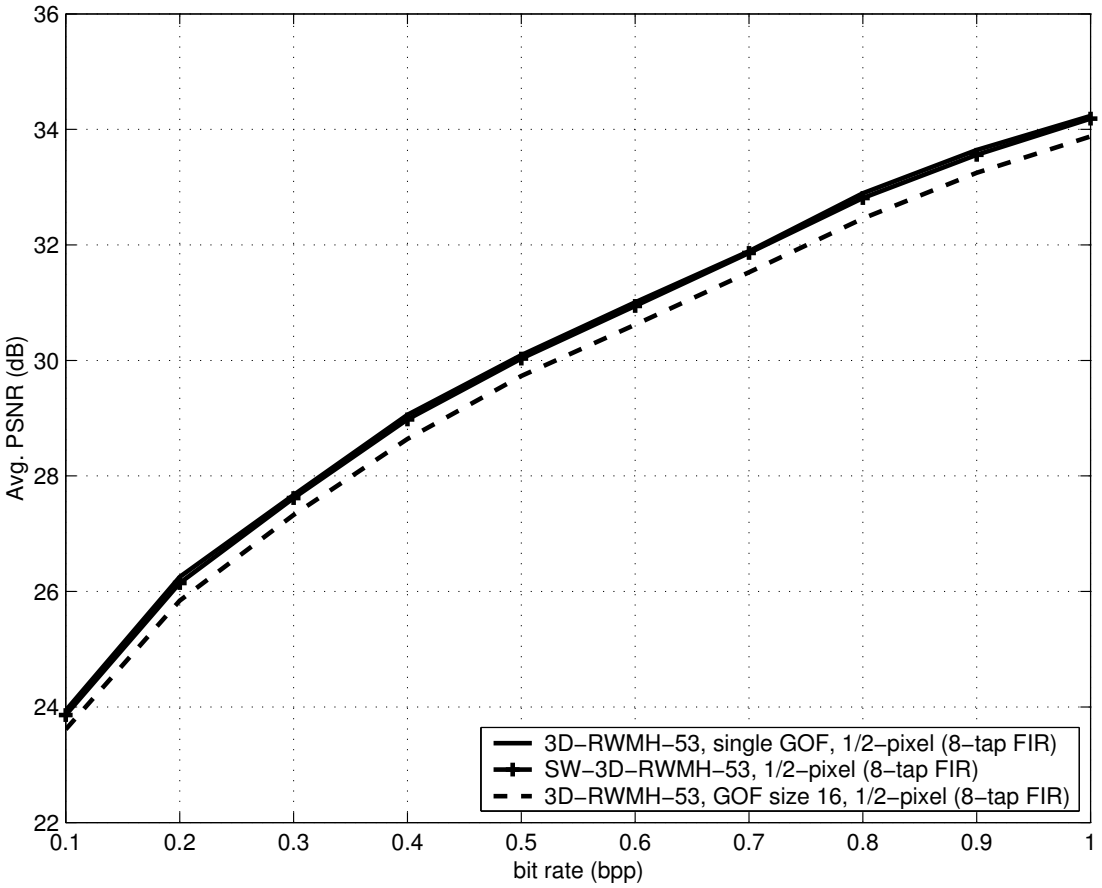


Figure 5.9: Rate-distortion performance of SW-3D-RWMH-53 for "Football."

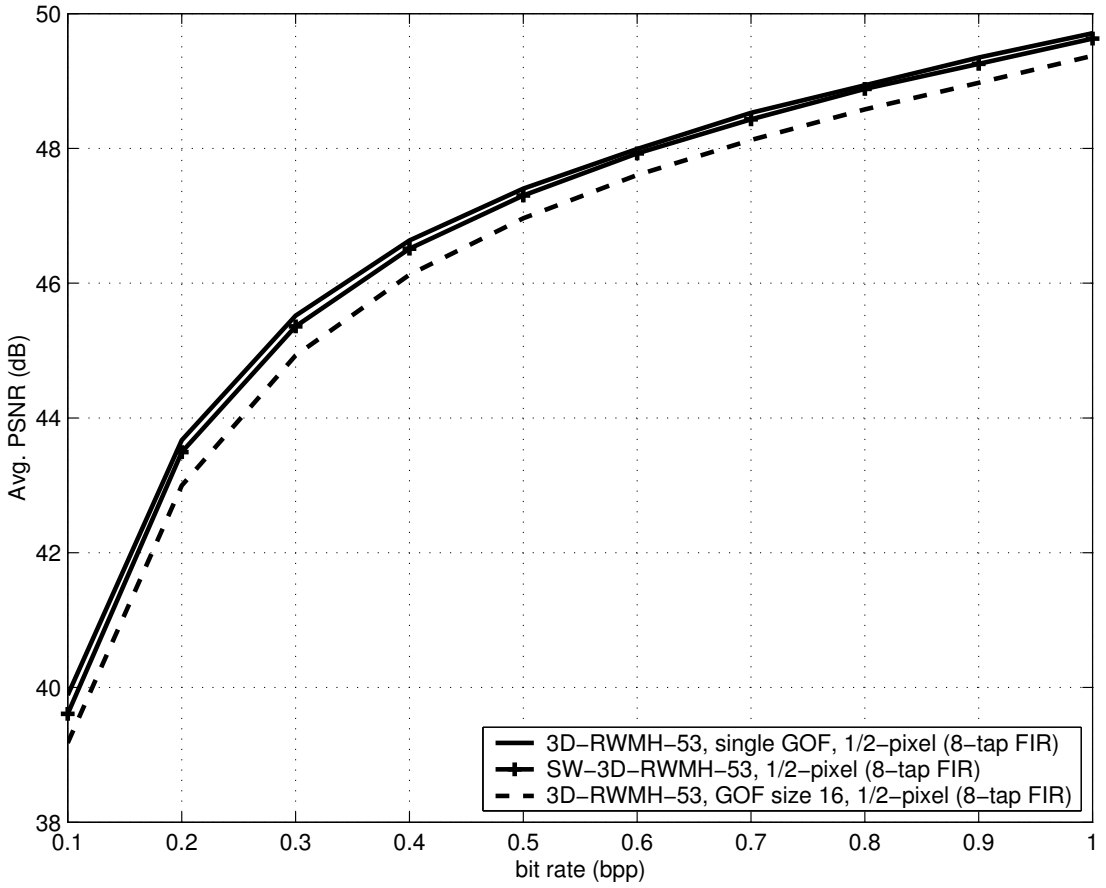


Figure 5.10: Rate-distortion performance of SW-3D-RWMH-53 for "Mother & Daughter."

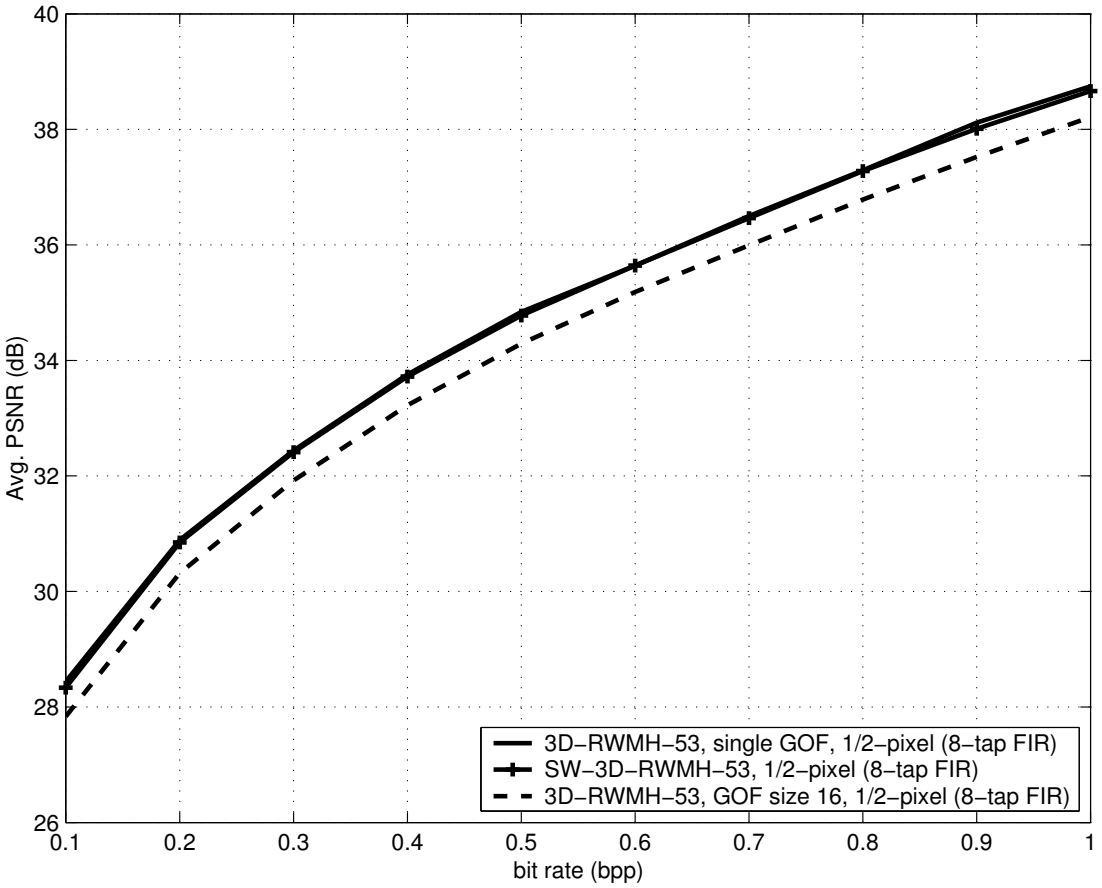


Figure 5.11: Rate-distortion performance of SW-3D-RWMH-53 for "Coastguard."

CHAPTER VI

3D CODING WITH THE TARP ALGORITHM

Currently there are no standards—or even commonly accepted coding algorithms—for 3D video-coding systems. Several embedded wavelet-based 3D codecs have been used by different researchers, such as JPEG-2000 [5, 6], 3D-SPIHT [4], and 3D-EZBC [10, 14]. However, many such embedded wavelet-based coding schemes utilize sophisticated processes such as context conditioning (JPEG-2000), rate-distortion optimization (JPEG-2000), or significance lists (3D-SPIHT and 3D-EZBC) which present significant difficulty for on-board implementations in hardware, particularly when parallel processing is considered.

In the following this chapter, we extend the recently proposed tarp coder [47], a 2D embedded wavelet-based coder with an exceedingly simple implementation, to 3D for the coding of 3D video volumes. The tarp technique employs an explicit estimate of the probability of wavelet-coefficient significance and a simple nonadaptive arithmetic coder, resulting in a still-image coder that is easily scaled to higher-dimensional datasets. While the probability estimate takes the form of Parzen windows, a well known nonparametric probability-estimation technique, the tarp coder implements this Parzen-window probability estimate as a novel sequence of 1D filtering operations coined tarp filtering. Experimental results show that our 3D version of tarp (3D-Tarp) can achieve almost the same rate-distortion performance as the 3D version of SPIHT (3D-SPIHT) [4] used in experimental results in the preceding chapters.

As pertaining to hardware implementation, we show at the end of this chapter that the most time-consuming operation of our tarp coder, the tarp filtering, can

be highly vectorized for implementation on single-instruction-multiple-data (SIMD) architectures. Thus, the proposed tarp coder can exploit the data-parallel capabilities of modern general-purpose processors, or, for greater concurrency, customized hardware with longer vectors could be used. In any event, the tarp coder benefits from the simplicity, elegance, and implicit synchronization of SIMD implementation, whereas other algorithms, such as 3D-SPIHT and JPEG-2000, typically require a more complicated multiprocessor implementation to achieve a smaller amount of parallelism.

Below, we describe the philosophy behind tarp coding, our implementation of the proposed 3D-Tarp coder, and the possibilities for vectorized operation. We note that our initial development [48, 49] of the 3D-Tarp coder was for the purpose of coding 3D hyperspectral imagery; however, like other volumetric coders, 3D-Tarp is equally applicable to the 3D video-coding problem.

6.1 Estimation of Probability of Significance via Parzen Windows

Consider an N-dimensional field of real-valued coefficients, $c[\mathbf{x}] \in \mathbb{R}$, where $\mathbf{x} \in \mathbb{Z}^N$, $\mathbb{R}$ is the set of real numbers, and $\mathbb{Z}$ is the set of integers. Given a threshold $t \in \mathbb{R}$, the coefficient at location $\mathbf{x}$ is defined to be significant with respect to t if $|c[\mathbf{x}]| \geq t$, and is insignificant otherwise. Define the *significance state* with respect to t of $c[\mathbf{x}]$ to be

$$v[\mathbf{x}] = \begin{cases} 1, & |c[\mathbf{x}]| \geq t, \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

Suppose we know that coefficients at locations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ are significant with respect to some given threshold, and we would like to estimate the probability that the coefficient at location $\mathbf{x}$ is also significant. Parzen windows [50] is one approach to performing this probability estimate. Specifically, we estimate the probability that $c[\mathbf{x}]$

is significant as

$$p[\mathbf{x}] = \sum_{i=1}^{m} \phi[\mathbf{x} - \mathbf{x}_i], \tag{6.2}$$

where $\phi[\mathbf{x}]$ is an N-dimensional window sequence. A possible window sequence which is suited to the well known Laplacian distribution nature of wavelet-coefficient magnitudes in images is the Laplacian window,

$$\phi[\mathbf{x}] = \beta\alpha^{||\mathbf{x}||}, \quad \mathbf{x} \in \mathcal{R}, \tag{6.3}$$

where α is a parameter controlling the spread of the window,

$$||\mathbf{x}|| = \sum_{i=1}^{N} |x_i| \tag{6.4}$$

is the l_1 norm of $\mathbf{x} = [x_1, x_2, , \ldots, x_N]$, and β is chosen so that

$$\sum_{\mathbf{x}\in\mathcal{R}} \phi[\mathbf{x}] = 1, \tag{6.5}$$

where $\mathcal{R} \subseteq \mathbb{Z}^N$ is the region of support of the window. As a result, it can be shown [50] that $p[\mathbf{x}]$ is guaranteed to be a valid probability mass function; i.e.,

$$p[\mathbf{x}] \geq 0, \, \forall \mathbf{x} \in \mathbb{Z}^N, \tag{6.6}$$

and

$$\sum_{\mathbf{x}\in\mathbb{Z}^N} p[\mathbf{x}] = 1. \tag{6.7}$$

The density estimation of (6.2) can be considered to be the convolution of an N-dimensional filter of impulse response $\phi[\mathbf{x}]$ with a field of Kronecker impulses situated at $\mathbf{x}_1$, $\mathbf{x}_2$, $\ldots$, $\mathbf{x}_m$. If the region of support $\mathcal{R}$ of window $\phi[\mathbf{x}]$ is causal, then this

convolution can be calculated via a single raster scan through the coefficients. Below, we will define the causal region of support so as to not include $\mathbf{x} = 0$. By not including $\mathbf{x} = 0$ in $\mathcal{R}$, both an encoder and its corresponding decoder in a compression system can make the same estimate of $p[\mathbf{x}]$ by single raster scan since (6.2) depends on only values encountered strictly before the current location in the raster scan.

6.2 Tarp Filtering

In [47], Simard *et al*. propose using the density estimate of (6.2) to code the significance of wavelet coefficients for still-image coding. Specifically, the significance state of a set of coefficients for a given threshold is coded via a raster scan through the coefficients. For coding efficiency, an entropy coder codes $v[\mathbf{x}]$ for each coefficient, using the probability that $v[\mathbf{x}] = 1$ for the current coefficient as determined by the density-estimation procedure. The coder of [47] implements the N-dimensional convolution of (6.2) as a sequence of 1D filtering operations coined *tarp filtering*.[1] This 1D-filtering approach is more efficient than a direct implementation of (6.2) in that only a limited number of probability estimates need be buffered, and that, because probability estimates are propagated from coefficient to coefficient, fewer arithmetic operations are performed.

Once the probability of significance of the coefficients is estimated for a given threshold, the tarp coder of [47] proceeds in the usual bitplane-coding paradigm common to modern embedded coders—significance and refinement passes are applied successively, and the significance threshold decreases after pass. In [47], the significance pass uses the tarp filter to drive a nonadaptive binary arithmetic coder to code $v[\mathbf{x}]$

[1]The name *tarp filtering* comes from the shape of the Laplacian window of (6.3) which resembles a tarp draped over a pole.

in each subband, while coefficient-sign and refinement information is coded using a nonadaptive binary arithmetic coder on a uniform distribution.

Below, we describe the tarp-filtering procedure in greater detail, first concentrating on the $N = 2$ case, which was the only dimensionality considered in the original development [47]. However, since our focus is the coding of 3D data, we extend the tarp algorithm to the $N = 3$ case in Sec. 6.2.2. We consider use of the 3D tarp-filtering operation in the coding of 3D video wavelet coefficients subsequently in Sec. 6.3.

6.2.1 2D Tarp Filtering

For $N = 2$, $\mathbf{x} = [x_1, x_2]$, where x_1 and x_2 are the row and column indices, respectively. The Laplacian window (6.3) in this case is

$$\phi[\mathbf{x}] = \beta\alpha^{|x_1|+|x_2|}, \quad \mathbf{x} = [x_1, x_2] \in \mathcal{R}, \tag{6.8}$$

where the causal region of support is $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$,

$$\begin{aligned} \mathcal{R}_1 &= \{\mathbf{x} = [x_1, x_2] : x_1 = 0, x_2 > 0\}, \\ \mathcal{R}_2 &= \{\mathbf{x} = [x_1, x_2] : x_1 > 0, x_2 \in \mathcal{Z}\}. \end{aligned} \tag{6.9}$$

In order for (6.5) to hold for this $\phi[\mathbf{x}]$ and $\mathcal{R}$, it can be derived that

$$\beta = \frac{(1-\alpha)^2}{2\alpha}. \tag{6.10}$$

In essence, the tarp coder of [47] uses three 1D filters to implement the density estimate of (6.2)—one filter processes each row from left to right, another filter processes each row from right to left, and a third filter processes each column from top to bottom. Pseudocode for this filtering operation is given in Fig. 6.1. In Fig. 6.1,

p_1 forms the left-to-right row filter, the updating of p_3 corresponds to the right-to-left row filter, and the updating of p_2 implements the top-to-bottom filter carried out on each column. We note that the memory overhead of these filtering operations is one row of p_2 values. For more detail on how tarp filtering is combined with bitplane coding to produce an embedded image coder, see [47, 51] and the tarp-coder implementation in QccPack [52].

6.2.2 3D Tarp Filtering

In this section, we extend to 3D volumes the 2D tarp filter described above. For $N = 3$, $\mathbf{x} = [x_1, x_2, x_3]$, where x_1, x_2, and x_3 are the spatial-row, spatial-column, and temporal-frame indices, respectively. The Laplacian window (6.3) in this case is

$$\phi[\mathbf{x}] = \beta\alpha^{|x_1|+|x_2|+|x_3|}, \quad \mathbf{x} = [x_1, x_2, x_3] \in \mathcal{R}, \tag{6.11}$$

where the causal region of support is $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$,

$$\begin{aligned} \mathcal{R}_1 &= \{\mathbf{x} = [x_1, x_2, x_3] : x_1, x_3 = 0, x_2 > 0, \}, \\ \mathcal{R}_2 &= \{\mathbf{x} = [x_1, x_2, x_3] : x_1 > 0, x_2 \in \mathcal{Z}, x_3 = 0\}, \\ \mathcal{R}_3 &= \{\mathbf{x} = [x_1, x_2, x_3] : x_1, x_2 \in \mathcal{Z}, x_3 > 0\}. \end{aligned} \tag{6.12}$$

In order for (6.5) to hold for this $\phi[\mathbf{x}]$ and $\mathcal{R}$, it can be derived that

$$\beta = \frac{(1-\alpha)^3}{3\alpha + \alpha^3}. \tag{6.13}$$

Fig. 6.2 shows a typical Laplacian window in 3D.

To estimate the probability of significance in 3D, we propagate information from three neighboring values, one at the left, one above, and one in the same spatial position

```
for x_1 = 0, ..., N_1 - 1
   p_1 = 0
   for x_2 = 0, ..., N_2 - 1
      p[x_1, x_2] = αp_1 + αp_2[x_2]
      p_1 = αp_1 + βv[x_1, x_2]
      p_2[x_2] = p_1 + αp_2[x_2]
   endfor
   p_3 = 0
   for x_2 = N_2 - 1, ..., 0
      p_2[x_2] = p_2[x_2] + αp_3
      p_3 = αp_3 + βv[x_1, x_2]
   endfor
endfor
```

Figure 6.1: Pseudocode for the 2D tarp filter of [47]. The image is of size $N_1 \times N_2$.

		$x_3 = 0$		
0	0	0	0	0
0	0	0	0	0
0	0	$\boxed{0}$	0.0385	0.0192
0.0096	0.0192	0.0385	0.0192	0.0096
0.0048	0.0096	0.0192	0.0096	0.0048
		$x_3 = 1$		
0.0024	0.0048	0.0096	0.0048	0.0024
0.0048	0.0096	0.0192	0.0096	0.0048
0.0096	0.0192	0.0385	0.0192	0.0096
0.0048	0.0096	0.0192	0.0096	0.0048
0.0024	0.0048	0.0096	0.0048	0.0024
		$x_3 = 2$		
0.0012	0.0024	0.0048	0.0024	0.0012
0.0024	0.0048	0.0096	0.0048	0.0024
0.0048	0.0096	0.0192	0.0096	0.0048
0.0024	0.0048	0.0096	0.0048	0.0024
0.0012	0.0024	0.0048	0.0024	0.0012

Figure 6.2: The 3D Laplacian window for $\alpha = 0.5$. The boxed value indicates the window origin $(x_1, x_2, x_3 = 0)$.

in the previous temporal frame. Raster scanning proceeds in the order column, row, and then frame, and we use 1D filters to propagate probability estimates. Specifically, five 1D filtering steps are used. Three filters (p_1, p_2, and p_4) essentially operate at the current frame in a fashion similar to the 2D tarp filter. That is, in the current frame, one filter processes each row from left to right, another filter processes each row from right to left, and a third filter processes each column from top to bottom. Next we propagate information in the temporal direction. To do so, we use buffers that hold probabilities for the entire previous frame, and after each frame is coded, the probabilities in the frame buffer are updated. Consequently, after a full frame is coded with the first three filters, another two 1D filtering steps (p_3 and p_5) update the probabilities for the current frame. Pseudocode for the 3D tarp filter is shown in Fig. 6.3.

The 3D tarp filtering operation requires somewhat greater buffer storage than its 2D counterpart. Specifically, single rows are stored for p_2 and p_5, while entire frames are stored for p_1, p_3, and p_4. Below, we describe how 3D tarp filter is combined with bitplane coding to create an embedded coder for 3D volumes.

6.3 The 3D-Tarp Coder

The tarp coder is built upon the embedded bitplane-coding architecture common to modern wavelet-based image coders. Specifically, a significance pass describes the significance state of coefficients, while a refinement pass produces a successive approximation to the values of the coefficients.

In the significance pass, the significance state $v[\mathbf{x}]$ of each coefficient is encoded, and, when a coefficient transitions from insignificant to significant, the sign of the coefficient is also encoded. In the refinement pass, all the coefficients known to be significant (except those that became significant in the immediately preceding significance pass) are refined by coding the value of the bit in the current bitplane.

```
for x_1 = 0, ..., N_1 - 1
   for x_2 = 0, ..., N_2 - 1
      p_3[x_1, x_2] = 0
   endfor
endfor
for x_3 = 0, ..., N_3 - 1
   for x_2 = 0, ..., N_2 - 1
      p_2[x_2] = 0
   endfor
   for x_1 = 0, ..., N_1 - 1
      for x_2 = 0, ..., N_2 - 1
         p[x_1, x_2, x_3] = αp_1[x_1, x_2 - 1] + αp_2[x_2] + αp_3[x_1, x_2]
         p_1[x_1, x_2] = αp_1[x_1, x_2 - 1] + βv[x_1, x_2, x_3]
         p_2[x_2] = p_1[x_1, x_2] + αp_2[x_2]
      endfor
      for x_2 = N_2 - 1, ..., 0
         p_2[x_2] = p_2[x_2] + αp_4[x_1, x_2 + 1]
         p_3[x_1, x_2] = p_2[x_2] + αp_3[x_1, x_2]
         p_4[x_1, x_2] = αp_4[x_1, x_2 + 1] + βv[x_1, x_2, x_3]
      endfor
   endfor
   for x_2 = 0, ..., N_2 - 1
      p_5[x_2] = 0
   endfor
   for x_1 = N_1 - 1, ..., 0
      for x_2 = 0, ..., N_2 - 1
         p_3[x_1, x_2] = p_3[x_1, x_2] + αp_5[x_2]
         p_5[x_2] = p_1[x_1, x_2] + αp_5[x_2] + αp_4[x_1, x_2 + 1]
      endfor
   endfor
endfor
```

Figure 6.3: Pseudocode for the 3D tarp filter. The volume is of size $N_1 \times N_2 \times N_3$.

Contrary to most wavelet-based embedded coders, which use multiple-context adaptive arithmetic coding which is responsible for a significant portion of their rate-distortion performance, the tarp coder uses a relatively simple nonadaptive binary arithmetic coder. The tarp-filtering operation produces the estimate $p[\mathbf{x}]$ of the probability of significance of the current coefficient, and this probability estimate drives the arithmetic coder when coding the significance state $v[\mathbf{x}]$ in the significance pass. For the coding of sign bits in the significance pass, and for the coding of refinement bits in the refinement pass, we use a constant probability of 0.5 in the nonadaptive arithmetic coder. Although it is possible to use more sophisticated codings of these sign and refinement bits [53], in practice, a nonuniform probability distribution would result in minimal rate-distortion improvement, while the use of the uniform distribution greatly simplifies the implementation and reduces computational complexity.

6.4 Experimental Results

In our experiments, we compare the performance of 3D-Tarp coder to that of 3D-SPIHT coder. Specifically, in the 3D-RWMH-53 system presented in Chap. IV, a 3D-Tarp coder replaces the 3D-SPIHT coder. For our proposed 3D-Tarp coder, α is fixed at 0.3, and the filters $p_1, \ldots, p_5$ are initialized to 0 beyond the boundaries of the 3D subbands.

Rate-distortion performance for 3D-RWMH-53 with 3D-Tarp, 3D-RWMH-53 with 3D-SPIHT, and MC-EZBC are shown in Figs. 6.4 through 6.7. In these results, we see that 3D-RWMH-53 with 3D-Tarp and MC-EZBC provide largely similar rate-distortion performance for the sequences considered, with 3D-RWMH-53 with 3D-SPIHT usually slightly outperforming the other two. Especially at low bit rates (less than 0.5 bpp), 3D-RWMH-53 with 3D-Tarp gives better performance than MC-EZBC does.

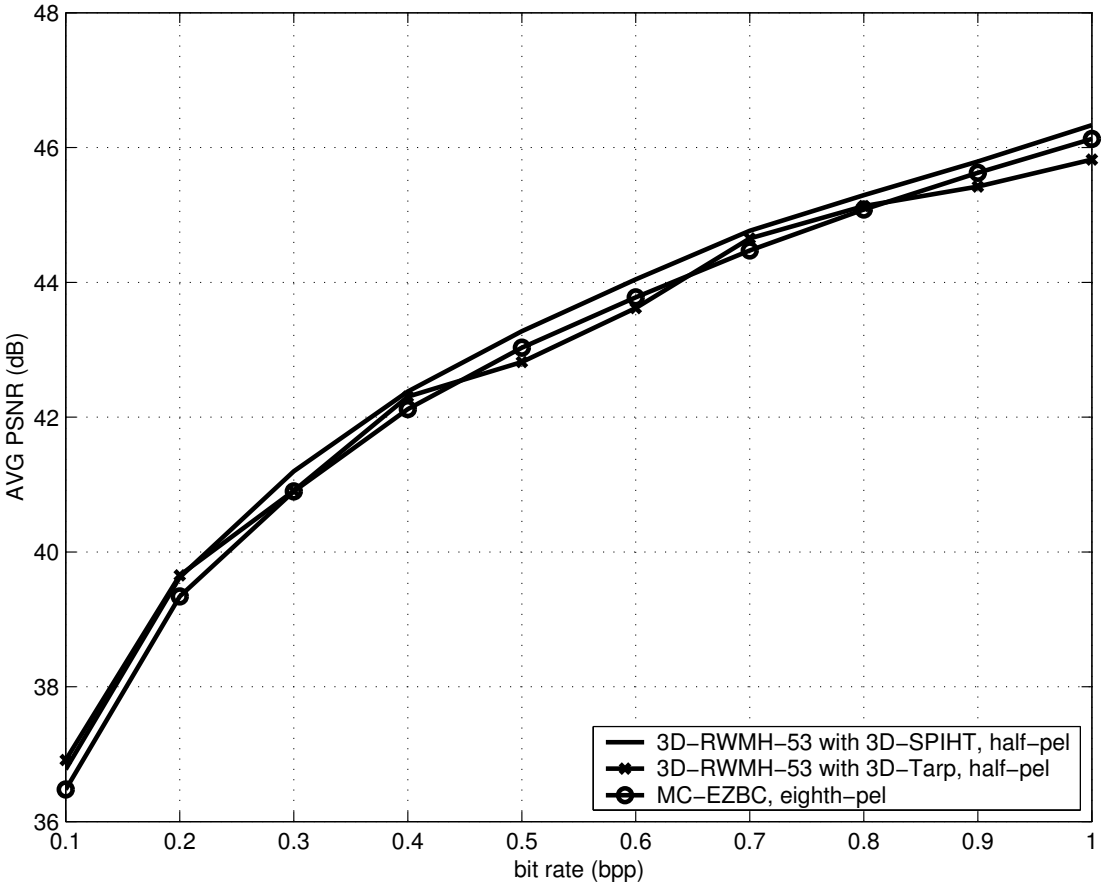


Figure 6.4: Rate-distortion performance for “Susie.”

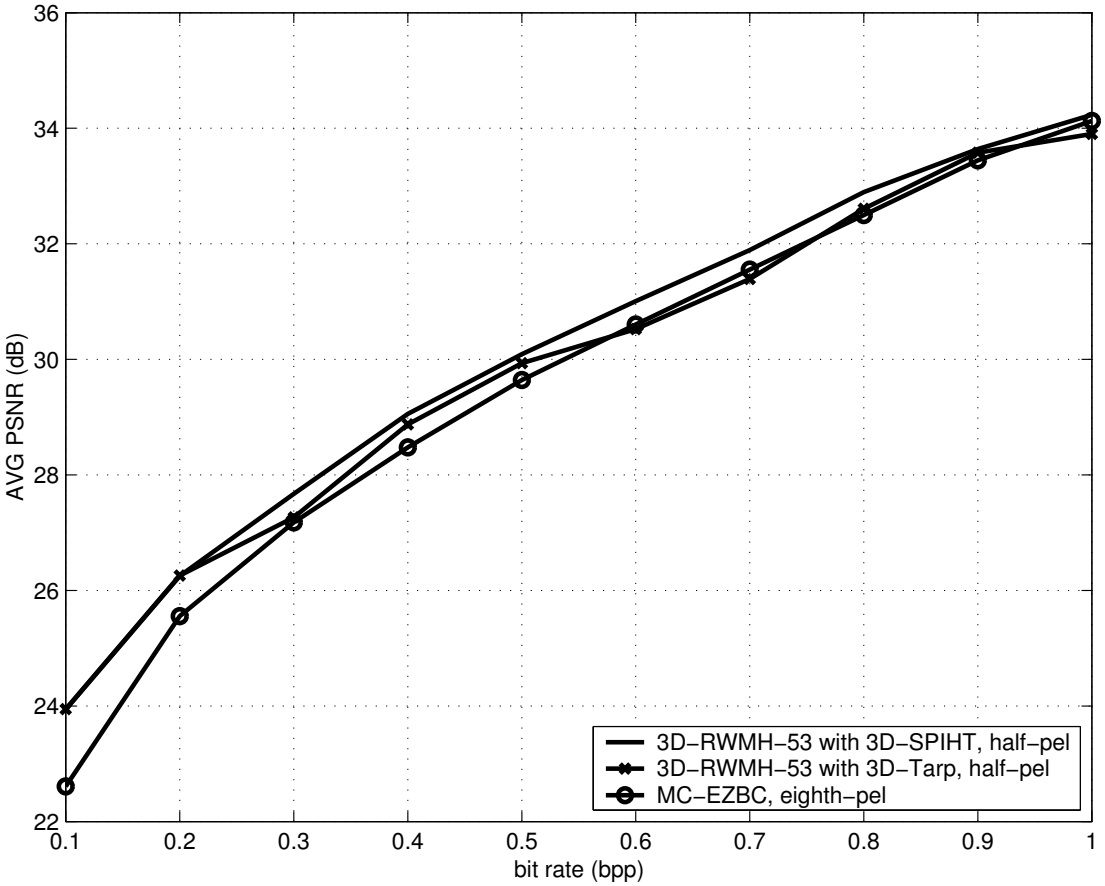


Figure 6.5: Rate-distortion performance for “Football.”

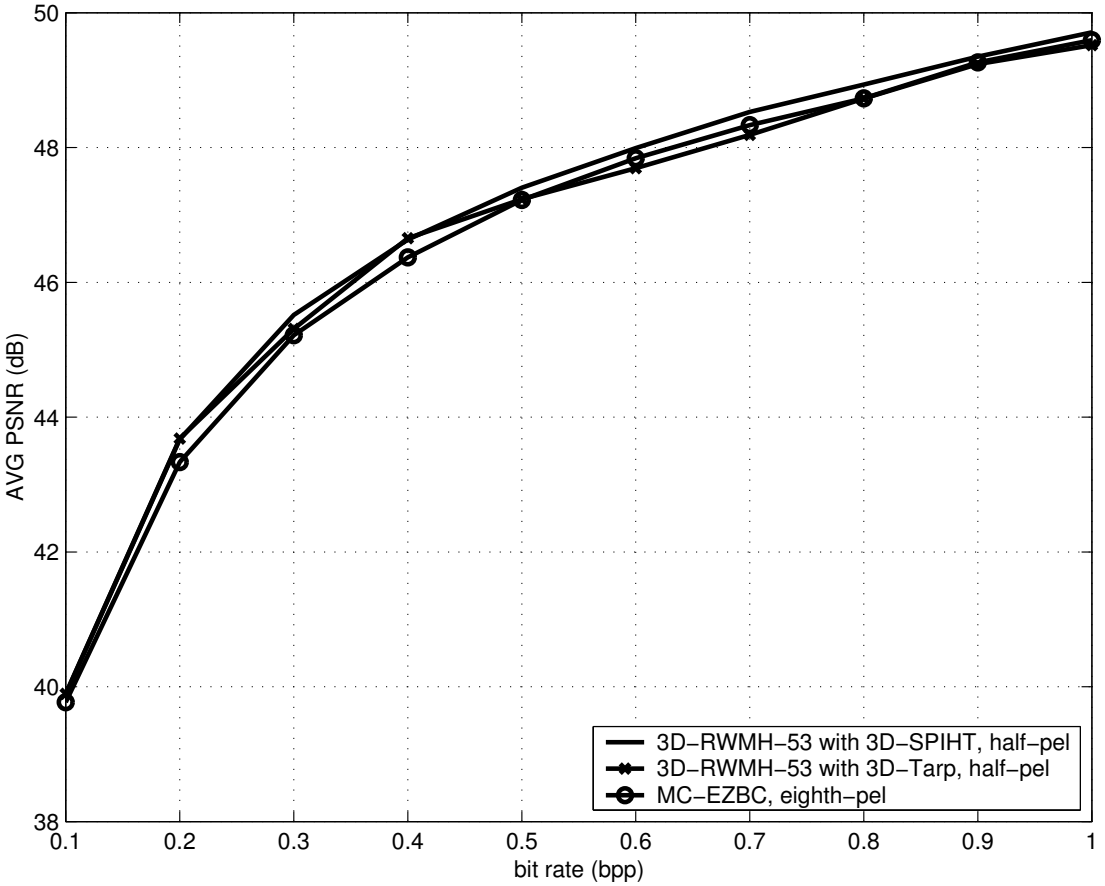


Figure 6.6: Rate-distortion performance for “Mother & Daughter.”

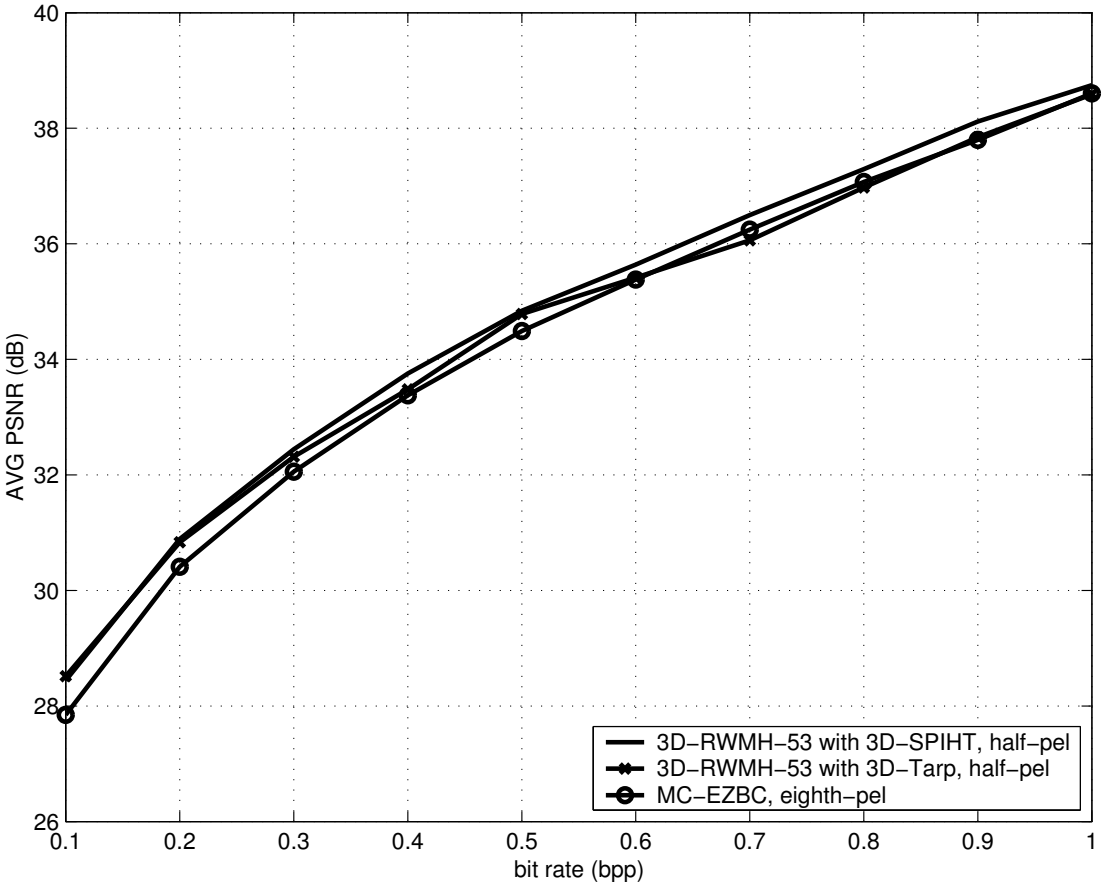


Figure 6.7: Rate-distortion performance for “Coastguard.”

6.5 Vectorized Implementation of 3D-Tarp

The tarp-filtering operation is responsible for an overwhelmingly large portion of the execution time of the software tarp coder used in the experimental results of the previous section. However, the tarp filter permits a significant amount of vectorization resulting in potentially substantial acceleration of the tarp coder when implemented in SIMD hardware. In the tarp-filtering operation, a large number of the filters are confined within one temporal frame, thereby allowing vectorization in the temporal direction; i.e., the filtering of multiple frames in parallel. Specifically, the p_1, p_2, p_4, and p_5 filters support vectorization in the temporal direction, although the ordering of the computations must be rearranged somewhat from that originally presented in Fig. 6.3. Additionally, the temporal-direction filter, p_3, can be vectorized in the column direction. Finally, the calculation of the final probability p can be vectorized in either the row, column, or temporal direction. Fig. 6.8 gives the resulting parallelized version of the 3D-Tarp encoder filter. We note that the cost of the reordering of the algorithm from that of Fig. 6.3 is increased memory usage since one must maintain entire buffer volumes for $p_1, \ldots, p_5$ rather than the single frames needed for p_1, p_3, and p_4 originally. However, recall that the tarp coder employs tarp filtering on a subband-by-subband basis; consequently, buffer volumes need be only as big as the largest subband to be processed, specifically, $N_1 N_2 N_3/8$. Additionally, we note that, since the decoder needs p for the current coefficient in order to decode v, the reordering of the tarp filter shown in Fig. 6.8 is suitable for only the encoder of a tarp-coder system. As shown in Fig. 6.9 for the decoder, p_1 and p_4 have to be calculated one by one, while p_2, p_3, and p_5 can be vectorized.

The degree of acceleration achieved by the vectorized tarp coder will depend on the amount of data-parallelism supported by the underlying SIMD architecture. To increase parallelization and reduce computational complexity, the tarp-filtering operations can

```
for x_1 = 0, ..., N_1 - 1
   for x_2 = 0, ..., N_2 - 1
      p_1[x_1, x_2, :] = αp_1[x_1, x_2 - 1, :] + βv[x_1, x_2, :]
      p_2[x_1, x_2, :] = p_1[x_1, x_2, :] + αp_2[x_1 - 1, x_2, :]
   endfor
   for x_2 = N_2 - 1, ..., 0
      p_2[x_1, x_2, :] = p_2[x_1, x_2, :] + αp_4[x_1, x_2 + 1, :]
      p_4[x_1, x_2, :] = αp_4[x_1, x_2 + 1, :] + βv[x_1, x_2, :]
   endfor
endfor
for x_1 = N_1 - 1, ..., 0
   for x_2 = 0, ..., N_2 - 1
      p_5[x_1, x_2, :] = p_1[x_1, x_2, :] + αp_5[x_1 + 1, x_2, :] + αp_4[x_1, x_2 + 1, :]
   endfor
endfor
for x_3 = 0, ..., N_3 - 1
   for x_1 = 0, ..., N_1 - 1
      p_3[x_1, :, x_3] = p_2[x_1, :, x_3] + αp_3[x_1, :, x_3 - 1] + αp_5[x_1 + 1, :, x_3]
   endfor
endfor
for x_3 = 0, ..., N_3 - 1
   for x_1 = 0, ..., N_1 - 1
      p[x_1, :, x_3] = αp'_1[x_1, :, x_3] + αp_2[x_1 - 1, :, x_3] + αp_3[x_1, :, x_3 - 1]
   endfor
endfor
```

Figure 6.8: Pseudocode for the vectorized 3D-Tarp encoder filter for SIMD architectures. All buffer volumes initialized to zero at algorithm start. The ":" indicates vectorization along the corresponding dimension. p'_1 is p_1 offset by a one-column shift to the right; i.e., $p'_1[x_1, x_2, x_3] = p_1[x_1, x_2 - 1, x_3]$. This shift is accomplished during loading of the vector.

```
for x_3 = 0, ..., N_3 - 1
  for x_1 = 0, ..., N_1 - 1
    for x_2 = 0, ..., N_2 - 1
      p[x_1, x_2, x_3] = αp_1[x_1, x_2 - 1, x_3] + αp_2[x_1 - 1, x_2, x_3] + αp_3[x_1, x_2, x_3 - 1]
      decode v[x_1, x_2, x_3]
      p_1[x_1, x_2, x_3] = αp_1[x_1, x_2 - 1, x_3] + βv[x_1, x_2, x_3]
    endfor
    for x_2 = N_2 - 1, ..., 0
      p_4[x_1, x_2, x_3] = αp_4[x_1, x_2 + 1, x_3] + βv[x_1, x_2, x_3]
    endfor
    p_2[x_1, :, x_3] = p_1[x_1, :, x_3] + αp_2[x_1 - 1, :, x_3] + αp'_4[x_1, :, x_3]
  endfor
  for x_1 = N_1 - 1, ..., 0
    p_5[x_1, :, x_3] = p_1[x_1, :, x_3] + αp_5[x_1 + 1, :, x_3] + αp'_4[x_1, :, x_3]
  endfor
  for x_1 = 0, ..., N_1 - 1
    p_3[x_1, :, x_3] = p_2[x_1, :, x_3] + αp_3[x_1, :, x_3 - 1] + αp_5[x_1 + 1, :, x_3]
  endfor
endfor
```

Figure 6.9: Pseudocode for the vectorized 3D-Tarp decoder filter for SIMD architectures. All buffer volumes initialized to zero at algorithm start. The ":" indicates vectorization along the corresponding dimension. p'_4 is p_4 offset by a one-column shift to the left; i.e., $p'_4[x_1, x_2, x_3] = p_4[x_1, x_2+1, x_3]$. This shift is accomplished during loading of the vector.

be easily performed with fixed-point, rather than floating-point, arithmetic. Modern general-purpose processors typically support some integer-based SIMD processing. For example, assuming that 16-bit fixed-point representations are used, Motorola's AltiVec [54] SIMD implementation would support eight parallel operations, while Intel's MMX [55] would support four. Custom hardware implementation could conceivably employ longer vectors such that the acceleration obtainable would be limited by primarily the subband size. Let us assume N arithmetic operations can be performed simultaneously in one CPU cycle, where N is the vector length of SIMD CPU, and an "arithmetic operation" is an add, subtract, multiply, or divide. And let us assume the size of a subband is G frames, R rows, and C columns. From Fig. 6.3, for original 3D-Tarp implementation, we derive that the number of the total CPU cycles need to perform the tarp filter on this subband for either encoder or decoder as

$$\mathrm{CYC}_{\mathrm{orig}} = 23 \times R \times C \times G. \tag{6.14}$$

Note that this figure is for the tarp-filtering process only and does not include cycles needed for the wavelet transform or arithmetic coding. We see that the number of the total cycles is not related to the vector length N. From Fig. 6.8, we get the number of the total CPU cycles need for the encoder of SIMD 3D-Tarp implementation as

$$\mathrm{CYC}_{\mathrm{simd,encoder}} = 14 \times R \times C \times \lceil G/N \rceil + 9 \times C \times G \times \lceil R/N \rceil, \tag{6.15}$$

and from Fig. 6.9, the number of the total CPU cycles for the decoder is

$$\mathrm{CYC}_{\mathrm{simd,decoder}} = 11 \times R \times C \times G + 12 \times C \times G \times \lceil R/N \rceil. \tag{6.16}$$

Using (6.14)-(6.16), we plot two graphs, Figs. 6.10 and 6.11, to illustrate that the degree of acceleration increases with increasing vector length N. From Figs. 6.10 and 6.11,

we notice that 3D-Tarp encoder can be accelerated much more than the decoder can because the vectorization of the encoder is more complete. When $N = 16$, the number of the total CPU cycles of the SIMD version encoder is only $1/16$ of that of the original encoder, while the number of the CPU cycles of the SIMD implementation decoder is about $1/2$ of that of the original decoder.

Finally, we note that both SPIHT and JPEG-2000 support parallelization to a certain extent; for example, see [56, 57] and Chap. 17 of [58]. However, these algorithms are highly sequential by nature and are difficult to make parallel. Additionally, the amount of parallelization is limited and typically relies on pipelining and multiprocessor, i.e., multiple-instruction-multiple-data (MIMD), architectures. Consequently, such implementations lack the simple and implicitly synchronized architecture of SIMD-based tarp filtering.

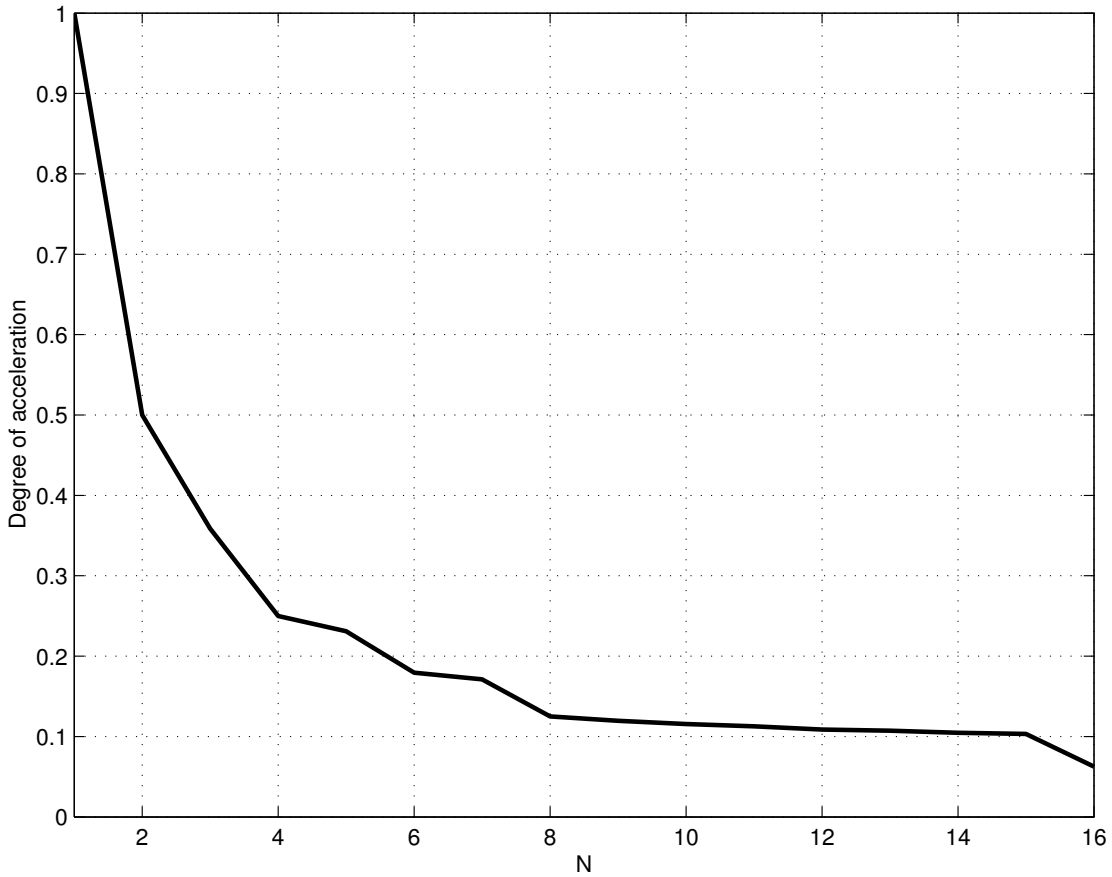


Figure 6.10: The degree of acceleration for encoder, $\frac{\mathrm{CYC}_{\mathrm{simd,encoder}}}{\mathrm{CYC}_{\mathrm{orig}}}$, for tarp filtering on a subband with 16 frames, 288 rows, and 352 columns. N is the vector length of SIMD CPU.

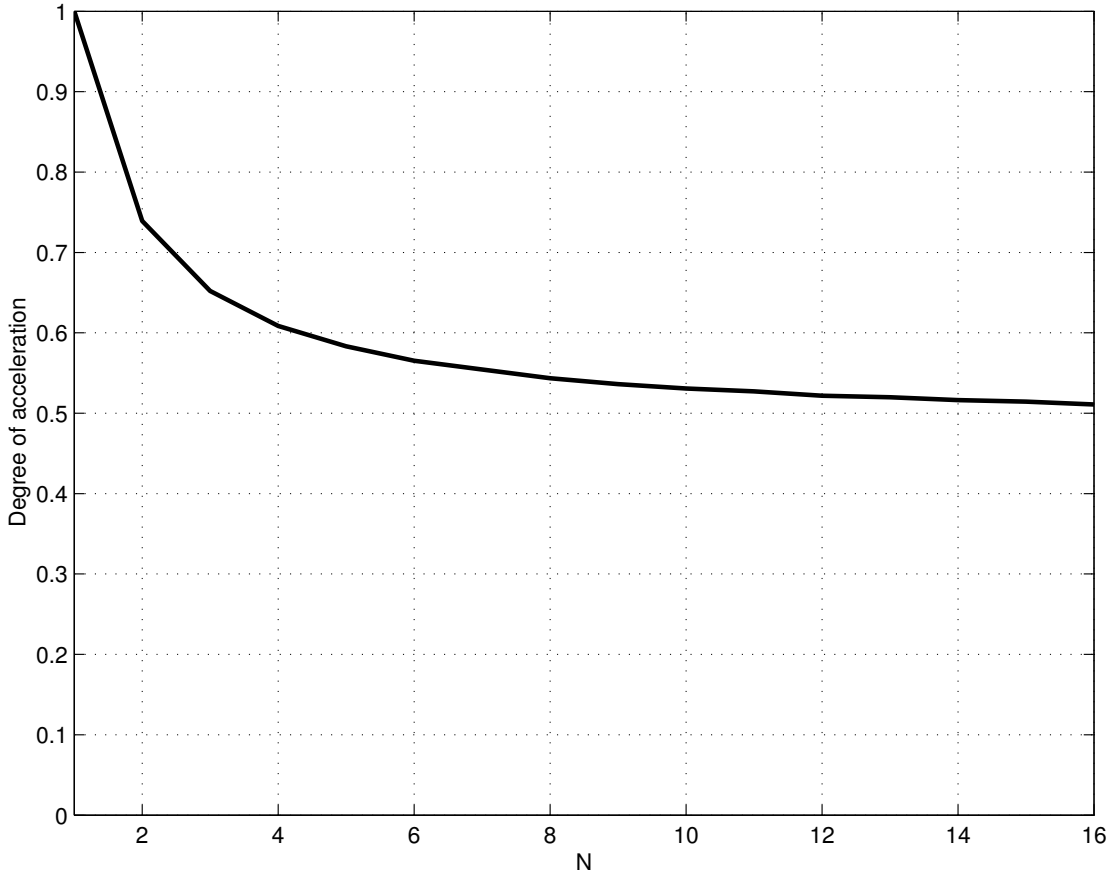


Figure 6.11: The degree of acceleration for decoder, $\frac{\mathrm{CYC}_{\mathrm{simd,decoder}}}{\mathrm{CYC}_{\mathrm{orig}}}$, for a subband with 16 frames, 288 rows, and 352 columns. N is the vector length of SIMD CPU.

CHAPTER VII

CONCLUSION

The overall goal of this dissertation was the design of a fully scalable 3D video-coding system. Such a system contains two major parts: a procedure for temporal filtering with MC and a 3D coefficient coder. The work presented here is focused on these two components.

The initial contribution of this dissertation work is the 3D-RWMH approach which introduces multihypothesis into motion-compensated temporal filtering (MCTF) to achieve a significant improvement in rate-distortion performance in a 3D video coder. Specifically, we deploy MCTF in the domain of a redundant wavelet transform, exploiting the transform redundancy to provide multiple hypothesis temporal filterings that are diverse in transform phase. Additionally, we depart from the block-based motion models commonly employed in MCTF by implementing temporal filtering with mesh-based lifting. In essence, the proposed 3D-RWMH system combines the flexibility and scalability of the mesh-based lifting MCTF of [5, 7] with the demonstrated performance gains associated with the RWMH introduced in [18, 19]. Additionally, performance is enhanced using half-pixel accuracy for both describing the motion of the mesh as well as within the affine transforms implementing the MCTF. Our 3D-RWMH system demonstrates superior performance by combining three types of multihypothesis—spatial diversity (subpixel accuracy), temporal diversity (5-3 biorthogonal temporal filtering), and phase diversity (RWMH). We note that our proposed system is apparently the first to combine all these concepts in the context of a 3D MCTF video coder.

As a second contribution of this dissertation, a sliding-window approach to MCTF was investigated in order to eliminate performance degradation characteristic to temporal filtering deployed in finite-length groups of frames (GOFs). Experimental results show that our sliding-window system (SW-3D-RWMH-53) yields nearly the same rate-distortion performance as the 3D-RWMH-53 applied to a video sequence with a single GOF. However, the sliding-window approach has a small frame delay and can be implemented in practical settings with a finite frame buffer even for arbitrarily long video sequences.

Finally, as the third contribution of this dissertation, an efficient 3D coder, 3D-Tarp, was designed based on the tarp-filtering algorithm originally proposed in 2D in [47]. Our experimental observations indicate that 3D-RWMH-53 with 3D-Tarp provides rate-distortion performance largely similar to that of 3D-EZBC [10, 14], a recent MCTF-based coder that is currently considered state of the art for 3D video coding. The 3D-RWMH-53 coder with 3D tarp also performed close to the 3D-RWMH-53 system using the sophisticated 3D-SPIHT coefficient coder. However, given its simplicity of implementation and its ability to exploit a high degree of vectorization, 3D tarp is perhaps the coder of the three that is best suited to on-board implementation, particularly when customized data-parallel (i.e., SIMD) hardware with long vector lengths is possible.

Recent activity within the video-standards community (e.g. [31, 59]) indicate there is increasing interest in wavelet-based video. The fact that much of this work is focused on providing increased scalability suggests that next-generation video-coding standards may adopt the MCTF approach to 3D coding so as to eliminate the scalability issues with traditional predictive-loop coders. In this case, the topics considered in this dissertation will play a central role in the development of future video coders. In particular, the concept of deploying MCTF in the domain of a redundant transform as well as that of

combining multiple forms of multihypothesis with spatial, temporal, and phase diversity will be central issues. Consequently, we hope that the work presented in this dissertation will contribute in no small way to the ongoing development of video technology.

REFERENCES

[1] ITU-T, *Video Coding for Low Bitrate Communication*, November 1995, ITU-T Recommendation H.263, Version 1.

[2] ITU-T, *Video Coding for Low Bitrate Communication*, January 1998, ITU-T Recommendation H.263, Version 2.

[3] ISO/IEC 13818-2, *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video*, 1995, MPEG-2 Video Coding Standard.

[4] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 8, pp. 1374–1387, December 2000.

[5] A. Secker and D. Taubman, "Highly scalable video compression using a lifting-based 3D wavelet transform with deformable mesh motion compensation," in *Proceedings of the International Conference on Image Processing*, Rochester, NY, September 2002, vol. 3, pp. 749–752.

[6] A. Secker and D. Taubman, "Motion-compensated highly scalable video compression using an adaptive 3D wavelet transform based on lifting," in *Proceedings of the International Conference on Image Processing*, Thessaloniki, Greece, October 2001, vol. 2, pp. 1029–1032.

[7] A. Secker and D. Taubman, "Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression," *IEEE Transactions on Image Processing*, December 2003, to appear.

[8] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 559–571, September 1994.

[9] S.-J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 155–167, February 1999.

[10] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with lifting implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, 2003, to appear.

[11] B. Pesquet-Popescu and V. Bottreau, "Three-dimensional lifting schemes for motion compensated video compression," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, UT, May 2001, vol. 3, pp. 1793–1796.

[12] D. S. Turaga and M. van der Schaar, "Wavelet coding for video streaming using new unconstrained motion compensated temporal filtering," in *Proceedings of the 2002 Tyrrhenian International Workshop on Digital Communications (IWDC 2002): Advanced Methods for Multimedia Signal Processing*, Capri, Italy, September 2002.

[13] M. van der Schaar and D. S. Turaga, "Unconstrained motion compensated temporal filtering (UMCTF) framework for wavelet video coding," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, April 2003, vol. 3, pp. 81–84.

[14] J. C. Ye and M. van der Schaar, "Fully scalable 3-D overcomplete wavelet video coding using adaptive motion compensated temporal filtering," in *Visual Communications and Image Processing*, Lugano, Switzerland, July 2003, to appear.

[15] C. Parisot, M. Antonini, and M. Barlaud, "3D scan based wavelet transform for video coding," in *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, Cannes, France, October 2001, pp. 403–408.

[16] J.-R. Ohm, "Advanced packet-video coding based on layered VQ and SBC techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 3, pp. 208–221, June 1993.

[17] V. Bottreau, M. Bénetière, B. Felts, and B. Pesquet-Popescu, "A fully scalable 3D subband video codec," in *Proceedings of the International Conference on Image Processing*, Thessaloniki, Greece, October 2001, vol. 2, pp. 1017–1020.

[18] S. Cui, Y. Wang, and J. E. Fowler, "Multihypothesis motion compensation in the redundant wavelet domain," in *Proceedings of the International Conference on Image Processing*, Barcelona, Spain, 2003, vol. 2, pp. 53–56.

[19] S. Cui, Y. Wang, and J. E. Fowler, "Motion compensation via redundant-wavelet multihypothesis," *IEEE Transactions on Circuits and Systems for Video Technology*, February 2003, submitted.

[20] J. Xu, Z. Xiong, S. Li, and Y.-Q. Zhang, "Memory constrained 3-D wavelet transform for video coding without boundary effects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 9, pp. 812–818, September 2002.

[21] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, pp. 339–366, June 1994.

[22] Y. Altunbasak, A. M. Tekalp, and G. Bozdagi, "Two-dimensional object-based coding using a content-based mesh and affine motion parameterization," in *Proceedings of the International Conference on Image Processing*, Washington, DC, October 1995, vol. 2, pp. 394–397.

[23] M. Eckert, D. Ruiz, J. I. Ronda, and N. Garcia, "Evaluation of DWT and DCT for irregular mesh-based motion compensation in predictive video coding," in *Visual Communications and Image Processing*, K. N. Ngan, T. Sikora, and M.-T. Sun, Eds. Proc. SPIE 4067, June 2000, pp. 447–456.

[24] S. Cui, Y. Wang, and J. E. Fowler, "Mesh-based motion estimation and compensation in the wavelet domain using a redundant transform," in *Proceedings of the International Conference on Image Processing*, Rochester, NY, September 2002, vol. 1, pp. 693–696.

[25] S. Cui, Y. Wang, and J. E. Fowler, "Motion estimation and compensation in the redundant-wavelet domain using triangle meshes," *IEEE Transactions on Circuits and Systems for Video Technology*, October 2002, submitted.

[26] M. Wollborn, I. Moccagatta, and U. Benzler, "Natural video coding," in *The MPEG-4 Book*, F. Pereira and T. Ebrahimi, Eds., chapter 8, pp. 293–382. Prentice-Hall, Upper Saddle River, NJ, 2002.

[27] M. J. Shensa, "The discrete wavelet transform: Wedding the à trous and Mallat algorithms," *IEEE Transactions on Signal Processing*, vol. 40, no. 10, pp. 2464–2482, October 1992.

[28] H.-W. Park and H.-S. Kim, "Motion estimation using low-band-shift method for wavelet-based moving-picture coding," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 577–587, April 2000.

[29] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, CA, 1998.

[30] H. S. Kim and H. W. Park, "Wavelet-based moving-picture coding using shift-invariant motion estimation in wavelet domain," *Signal Processing: Image Communication*, vol. 16, no. 7, pp. 669–679, April 2001.

[31] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, J. Barbarien, P. Schelkens, and J. Cornelius, "Wavelet-based fine granularity scalable video coding with in-band prediction," ISO/IEC JTC1/SC29/WG11, MPEG2002/M7906, Jeju Island, South Korea, March 2002.

[32] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens, and J. Cornelius, "Wavelet-based fully-scalable video coding with in-band prediction," in *Proceedings of the* 3rd *IEEE Benelux Signal Processing Symposium*, Leuven, Belgium, March 2002, pp. 217–220.

[33] X. Li, L. Kerofsky, and S. Lei, "All-phase motion compensated prediction in the wavelet domain for high performance video coding," in *Proceedings of the International Conference on Image Processing*, Thessaloniki, Greece, October 2001, vol. 2, pp. 538–541.

[34] X. Li and L. Kerofsky, "High-performance resolution-scalable video coding via all-phase motion-compensated prediction of wavelet coefficients," in *Visual Communications and Image Processing*, C.-C. J. Kuo, Ed. Proc. SPIE 4671, January 2002, pp. 1080–1090.

[35] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, P. Schelkens, and J. Cornelius, "Scalable wavelet video-coding with in-band prediction—Implementation and experimental results," in *Proceedings of the International Conference on Image Processing*, Rochester, NY, 2002, vol. 3, pp. 729–732.

[36] S. Cui, *Motion Estimation and Compensation in the Redundant Wavelet Domain*, Ph.D. dissertation, Mississippi State University, August 2003.

[37] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.

[38] G. J. Sullivan, "Multi-hypothesis motion compensation for low bit-rate video coding," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, MN, April 1993, vol. 5, pp. 437–440.

[39] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Transactions on Communications*, vol. 41, no. 4, pp. 604–612, April 1993.

[40] S. Nogaki and M. Ohta, "An overlapped block motion compensation for high quality motion picture coding," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, San Diego, CA, May 1992, vol. 1, pp. 184–187.

[41] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 693–699, September 1994.

[42] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 70–84, February 1999.

[43] M. Flierl, T. Wiegand, and B. Girod, "Rate-constrained multihypothesis prediction for motion compensated video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 957–969, November 2002.

[44] Y. Wang, S. Cui, and J. E. Fowler, "3D video coding using redundant-wavelet multihypothesis and motion-compensated temporal filtering," in *Proceedings of the International Conference on Image Processing*, Barcelona, Spain, 2003, vol. 2, pp. 755–758.

[45] Y. Wang, S. Cui, and J. E. Fowler, "3D video coding with redundant-wavelet multihypothesis," *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003, submitted.

[46] C. Parisot, M. Antonini, and M. Barlaud, "3D scan-based wavelet transform and quality control for video coding," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 1, pp. 56–65, January 2003.

[47] P. Simard, D. Steinkraus, and H. Malvar, "On-line adaptation in image coding with a 2-D tarp filter," in *Proceedings of the IEEE Data Compression Conference*, J. A. Storer and M. Cohn, Eds., Snowbird, UT, April 2002, pp. 23–32.

[48] Y. Wang, J. T. Rucker, and J. E. Fowler, "Embedded wavelet-based compression of hyperspectral imagery using tarp coding," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, Toulouse, France, July 2003, vol. 3, pp. 2027–2029.

[49] Y. Wang, J. T. Rucker, and J. E. Fowler, "3D tarp coding for the compression of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, July 2003, submitted.

[50] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, Inc., New York, 2nd edition, 2001.

[51] J. E. Fowler, "Shape-adaptive tarp coding," in *Proceedings of the International Conference on Image Processing*, Barcelona, Spain, September 2003, vol. 1, pp. 621–624.

[52] J. E. Fowler, "QccPack: An open-source software library for quantization, compression, and coding," in *Applications of Digital Image Processing XXIII*, A. G. Tescher, Ed., San Diego, CA, August 2000, Proc. SPIE 4115, pp. 294–301.

[53] A. T. Deever and S. S. Hemami, "Efficient sign coding and estimation of zero-quantized coefficients in embedded wavelet image codecs," *IEEE Transactions on Image Processing*, vol. 12, no. 4, pp. 420–430, April 2003.

[54] J. Tyler, J. Lent, A. Mather, and H. Nguyen, "AltiVec™: Bringing vector technology to the PowerPC™ processor," in *Proceedings of the IEEE International Conference on Performance Computing and Communications*, Scottsdale, AZ, February 1999, pp. 437–444.

[55] A. Peleg and U. Weiser, "MMX technology extension to the Intel architecture," *IEEE Micro*, vol. 16, no. 4, pp. 42–50, August 1996.

[56] F. W. Wheeler and W. A. Pearlman, "Low-memory packetized SPIHT image compression," in *Proceedings of the* 33rd *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 1999, vol. 2, pp. 1193–1197.

[57] J.-S. Chiang, Y.-S. Lin, and C.-Y. Hsieh, "Efficient pass-parallel architecture for EBCOT in JPEG-2000," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Scottsdale, AZ, May 2002, vol. 1, pp. 773–776.

[58] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Boston, MA, 2002.

[59] P. Chen and J. W. Woods, "Comparison of MC-EZBC and H.26L TML 8 on digital cinema test sequences," ISO/IEC JTC1/SC29/WG11, MPEG2002/M8130, Jeju Island, South Korea, March 2002.