ERROR CONCEALMENT FOR 5/3 MOTION COMPENSATED TEMPORAL FILTERING WITH LIFTING

by

Sunghoon Ivan Lee

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF APPLIED SCIENCE

in the School

of

Engineering Science

SIMON FRASER UNIVERSITY

August 2007

APPROVAL

Name: Sunghoon Ivan lee

Degree: Bachelor of Applied Science

Title of thesis: Error concealment for 5/3 motion compensated temporal filt ering with lifting

Examining Committee:

Chair and Academic Supervisor:

Dr. Mehrdad Saif
Director
School of Engineering Science, SFU

Committee Member:

Dr. Ivan V. Bajic
Assistant Professor
School of Engineering Science, SFU

Committee Member:

Dr. Jie Liang
Assistant Professor
School of Engineering Science, SFU

Dr. Atousa HajShirMohammadi
Lecturer
School of Engineering Science, SFU

Date Approved: _____

Abstract

5/3 Motion Compensated Temporal Filtering (MCTF) is a tool for highly scalable video coding which has been recently studied by many researchers. This thesis presents several error concealment algorithms for 5/3 MCTF with lifting, which can be used to improve the quality of compressed video damaged by packet losses. In MCTF video, the low frequency subband frame, abbreviated as *L*-frame, contains most of the signal energy in any given Group-of-Pictures (GOP). We assume that one of these *L*-frames is lost. The proposed error concealment algorithms use the available data to reconstruct the missing *L*-frame. The simplest error concealment method considered in the thesis is Zero Motion Error Concealment. This method simply assumes zero motion through the damaged GOP, and averages the neighboring *L*-frames to reconstruct the missing *L*-frame. Another method called Motion Concatenation finds temporal pathways through the damaged GOP by connecting motion vectors available at the decoder, and copies the corresponding pixel values from the neighboring *L*-frames to the missing *L*-frame. Finally, Motion Re-estimation uses motion estimator at the decoder to find a motion vectors between two neighboring *L*-frames of the missing *L*-frame, and synthesizes the missing *L*-frame halfway between its neighboring *L*-frames. The overall error concealment system combines these three methods to maximize visual performance, as well as the Peak Signal-to-Noise-Ratio (PSNR).

Acknowledgements

I would like to give my appreciation to Dr. Ivan V. Bajic for his great support over the last eight months. This thesis work could not be completed without his guidance, supervision, mentorship and encouragement.

I would like to thank Dr. Jie Liang and Dr. Atousa HajShirMohammadi for their support and interest in my thesis work.

I would like to give a special thanks to Dr. John W. Woods of the Center for Image Processing Research at Rensselaer Polytechnic Institute for his comments on some parts of my thesis project.

I would like to thank my father, Dongheon Lee and my mother Younhee Lee. Their unconditional love and support throughout my life lead me to this point. I would like to thank my brother Ted Lee for helping me to concentrate on my thesis. Finally, I also would like to thank my love, Angelica Shiwon Jang.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In modern multimedia communication systems, multimedia sources including pictures and movie clips are transferred through various channels with limited and time-varying bandwidth. These multimedia sources may be displayed on a mobile phone or on a high-resolution display. The resolution of the sources must be scaled when they are displayed on different pieces of hardware. For example, a movie clip must be in high quality when it is played on a home theater system, but it is not required to have such a high quality for a mobile phone screen. Scalable video coding is the term used for the algorithms which enable compressed video to be decoded in a variety of different spatial or temporal resolutions [1]. Motion-compensated temporal filtering (MCTF) is one of the methods for scalable video coding, which has been recently studied by many researchers due to high compression performance and scalability features [2], [3], [4], [5]. Our goal in this thesis is to develop error concealment methods for MCTF based on 5/3 filtering with lifting. This filtering scheme is used in the most recent version of the highly scalable Motion Compensated Embedded Zero-Block Coder (MC-EZBC) [6], [7], [8], which we use as the test platform. A version of this filter has also been adopted in the scalable extension of H.264/AVC [9]. The work presented in this thesis seems to be the first attempt to develop error concealment algorithms for 5/3 MCTF.

1.1 5/3 Motion Compensated Temporal filtering (MCTF) with Lifting

MCTF is a subband/wavelet-based filtering technique which performs filtering in temporal dimension along motion trajectories. MCTF encoder takes video frames and motion vector field as inputs, and produces filtered frames as outputs. There are two types of filtered output frames: high frequency subband frames (*H*-frames) and low frequency subband frames (*L*-frames) [1]. These filtered frames will be transmitted to the decoder along with motion vectors, and the decoder will reproduce the original frames (losslessly in the absence of quantization).

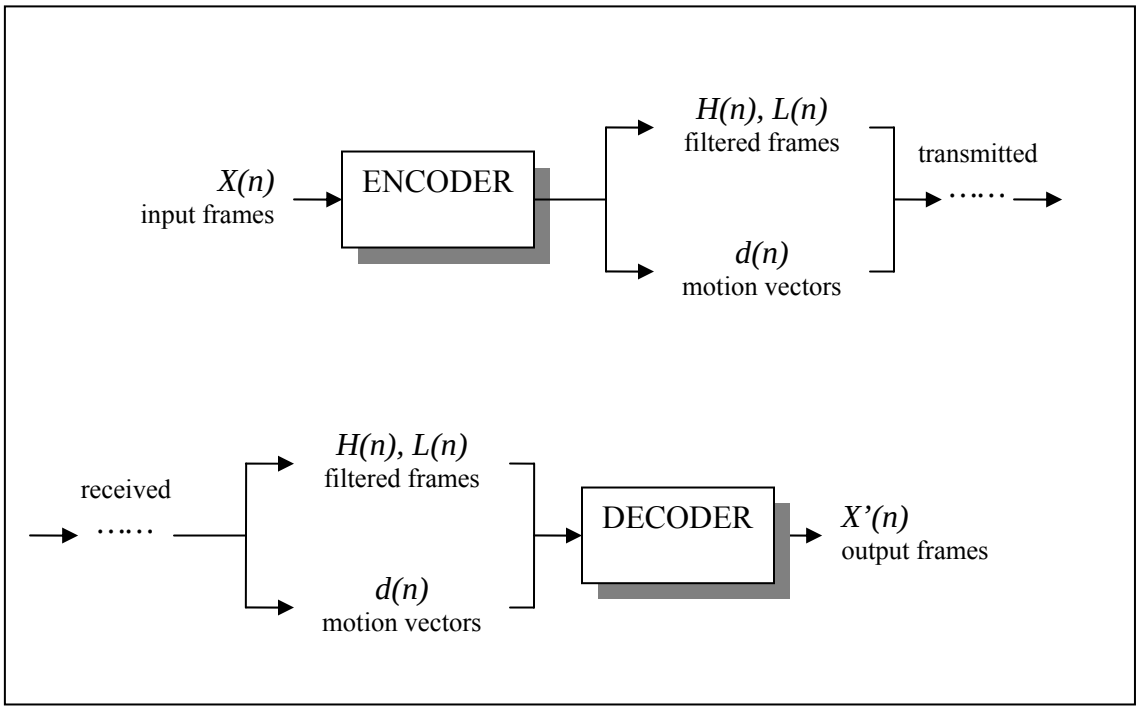


Figure 1.1 Functional block diagram of 5/3 MCTF encoder and decoder

In the 5/3 MCTF case, *H*-frame is the motion-compensated difference between an odd-numbered frame and its two neighboring even-numbered frames. In other words, each pixel value in *H*-frame is the difference between the corresponding pixel in the odd-numbered frame, and the average of the two pixels in the neighboring frames to which it is connected via motion vectors, as specified in equation (1.1) below. Ideally, if every pixel in an odd-numbered frame is connected to one pixel in each of the neighboring frames, and if the corresponding pixel values are all the same, then all pixel values in *H*-frame are equal to zero. On the other hand, the low frequency subband frame is a motion-compensated average between an even-numbered frame and the two neighboring high frequency frames, as specified in equation (1.2). Illustration is provided in Figure 1.2 [1].

$$H_t(\vec{n}) = A_t(\vec{n}) - \frac{1}{2}\left[B_t(\vec{n} + \overrightarrow{df_t}) + B_{t-1}(\vec{n} + \overrightarrow{dr_{t-1}})\right], \tag{1.1}$$

$$L_t(\vec{n}) = B_t(\vec{n}) + \frac{1}{4}\left[H_{t+1}(\vec{n} - \overrightarrow{dr_t}) + H_t(\vec{n} - \overrightarrow{df_t})\right]. \tag{1.2}$$

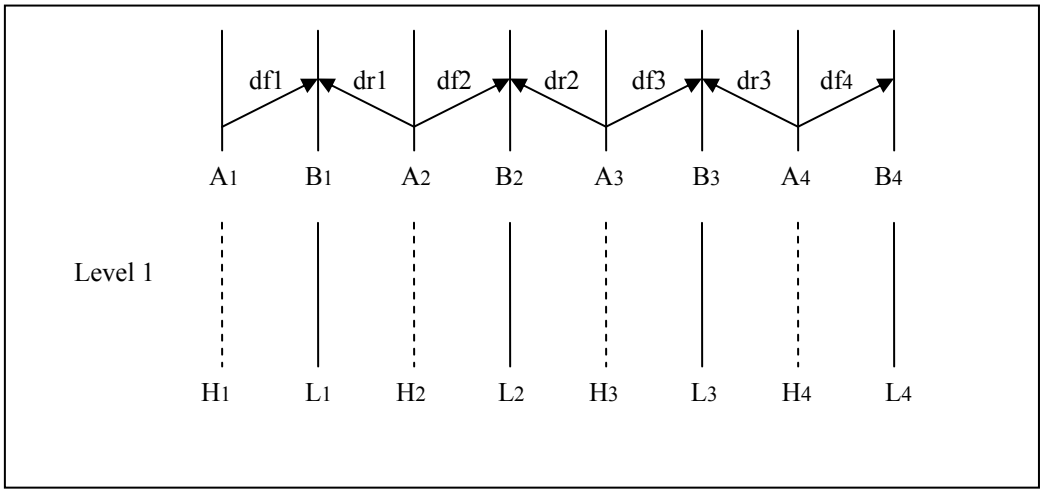


Figure 1.2: First level 5/3 MCTF with lifting

$A_t(\vec{n})$ and $B_t(\vec{n})$ represent original input video frames, while $H_t(\vec{n})$ and $L_t(\vec{n})$ represent high and low frequency frames, respectively. Forward and reverse motion vectors are denoted df_t and dr_t, respectively. Since *H*-frames are close to zero, the low frequency subband frames are very similar to the corresponding even numbered frames. The following figure shows an example of an *H*-frame (left) and an *L*-frame (right).

Figure 1.3: Sample of high frequency frame (left) and low frequency frame (right)

1.2 Error Concealment for 5/3 MCTF with Lifting

Error concealment is a technique that reduces the effects of data loss in visual communications [3], [4]. In practice, data loss may occur in various portions of the data transmitted. We assume that one of the *L*-frames, which contain the most of energy of the transmitted data, is lost during transmission. This assumption is valid for many cases, as explained in [3], because *L*-frames consume most bits in the compressed bitstream, and are most likely to be damaged. The 5/3 motion compensated (MC) error concealment system will reconstruct the missing *L*-frame using the available information at the decoder: its neighboring *H*- and *L*-frames, as well as motion vectors.

The thesis is organized as follows. In Chapter 2, we discuss the theoretical background of 5/3 MC error concealment. Here, we will develop mathematical equations that will be used to reconstruct the missing *L*-frame. In Chapter 3, we discuss the practical implementation of the error concealment system. We present different methods that we used to solve the problem. Experimental results and their analyses are presented in Chapter 4. Finally, in Chapter 5, we close the discussion of the topic with a conclusion and some suggestions future research.

Chapter 2

Theory of 5/3 Motion Compensated Error Concealment

In this chapter, we present theoretical derivation of error concealment equations. As discussed in Chapter 1, the 5/3 MC error concealment algorithm is an additional functional block which will be inserted into the 5/3 MCTF decoder. We assume that one of the *L*-frames is lost during the transmission from the encoder to the decoder. The error concealment system mounted in the decoder will reconstruct the missing *L*-frame. Figure 2.1 helps visualize the problem we are dealing with.

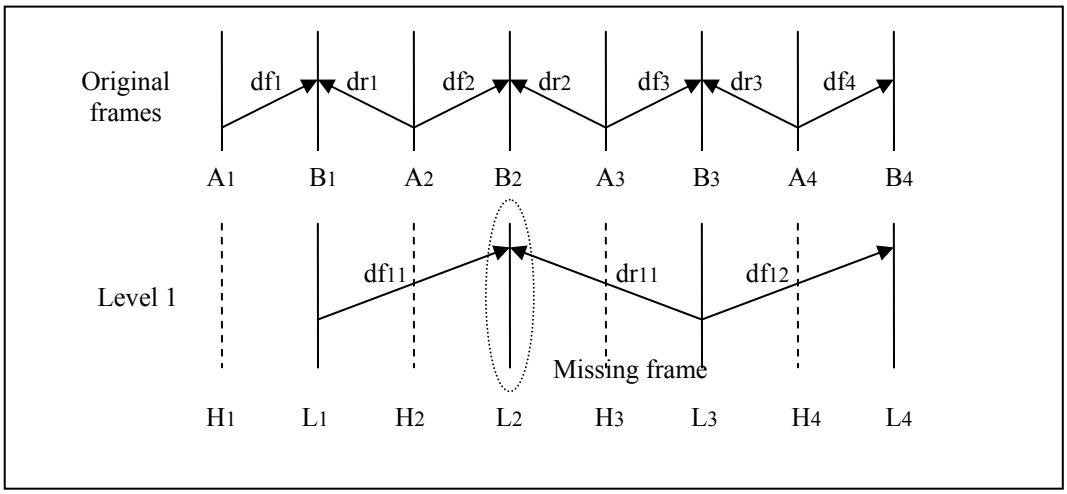


Figure 2.1: Illustration of 5/3 MCTF Error Concealment

In the diagram above, labeling of original frames, filtered frames, and motion vectors on the top level, is identical to that of Figure 1.2. In addition, there are two new motion vector fields at Level 1 in Figure 2.1, labeled df_{1t} and dr_{1t}. These motion vector fields do not exist in the

compressed video bitstream, since they are not needed to produce the L_2 frame at the encoder. However, when L_2 frame is lost, these motion vector fields can help us reconstruct L_2 from L_1 and L_3. One way to obtain these fields is as follows:

$$df_{1t} = df_{2t} - dr_{2t-1},$$

$$dr_{1t} = dr_{2t} - df_{2t+1}.$$

For example, df_{11} can be obtained as $df_{11} = df_2 - dr_1$ for all pixels which are connected through df_2 and dr_1. The operation "$df_2 - dr_1$" means that dr_1 is first reversed, and then the two motion vector fields are concatenated to produce df_{11}. A similar operation can be done to produce dr_{11}. We will use this operation throughout this text, keeping in mind that addition or subtraction of two motion vector fields implies such an operation. While Figure 2.1 uses an example of one MCTF level for simplicity, the same idea can be applied recursively to *L*-frames to produce multiple MCTF levels in a "MCTF pyramid."

Suppose that frame L_2 has been lost during transmission. The resources which may be used to reconstruct L_2 at the decoder include its neighboring low frequency frames (L_t, $t \neq 2$), high frequency frames (H_t), and motion vectors at the upper levels of the MCTF pyramid (df_t and dr_t). To help us derive the error concealment equations, we introduce two approximations. These approximations are important in the sense that they will categorize the error concealment into different conditions according to the usage of various resources at the decoder. They can be stated as follows.

Approximation 1: $H_t(\vec{n}) \approx 0$.

Approximation 2: $A_t(\vec{n}) \approx B_t(\vec{n} + \vec{df}_t)$ and $A_{t+1}(\vec{n}) \approx B_t(\vec{n} + \vec{dr}_t)$.

The first approximation states that pixel values in the *H*-frames are close to zero In our experiments, its impact on performance was negligible. This approximation allows us to eliminate *H*-frames from the error concealment process and it makes the algorithm much simpler. The second approximation tells us that neighboring *A* and *B* frames are almost equal when warped along motion vectors.

To test these approximations, we computed the following two parameters: the mean absolute value of the *H*-frames at the first MCTF level,

$$\begin{aligned} \left|H\right|_{avg} &= \frac{1}{N_{connected}} \times \sum_{n \in connected} \left|H(n)\right| \\ &= \frac{1}{N_{connected}} \times \sum_{n \in connected} \left| A_2(\vec{n}) - \frac{1}{2}\left[B_2(\vec{n} + \overrightarrow{df_2}) + B_1(\vec{n} + \overrightarrow{dr_1})\right] \right|, \end{aligned} \tag{2.1}$$

and the mean absolute difference between *A* and *B* frames along motion trajectories,

$$MAD_{A,B} = \frac{1}{N_{connected}} \times \sum_{n \in connected} \left| A_2(\vec{n}) - B_1(\vec{n} + \overrightarrow{df_1}) \right|. \tag{2.2}$$

In the two equations above, "$n \in connected$" indicates those pixels that have temporal connection to their neighboring frame(s), and $N_{connected}$ is the number of such pixels. We

computed these two quantities for the Y-component of the YUV-format sequence *Coastguard*. Their average values (over the entire sequence) were as follows: average $|H|_{avg}$ = 2.8 and average $MAD_{A,B}$ = 4.5. The range of Y-component pixel values is 0 – 255, so the average values of these quantities are within 1% – 2% of the total range. Further information about the results of this test can be found in Appendix A. This empirical result tells us that Approximation 2 is not valid when Approximation 1 is not valid, because Approximation 1 is a better approximation than Approximation 2. Based on this empirical conclusion, there are three cases that need to be considered in error concealment.

Case 1: Approximation 1 is valid and Approximation 2 is valid.

Case 2: Approximation 1 is valid and Approximation 2 is not valid.

Case 3: Approximation 1 is not valid and Approximation 2 is not valid.

These three cases restrict the use of resources to produce the estimate of the missing L_2 frame at the decoder. For example, concealment equations in Case 1 and Case 2 will not contain any *H*-frames because Approximation 1 is valid in these cases. On the other hand, the equations will contain *H*-frames in Case 3, because in this case *H*-frames can no longer be approximated to zero.

We will derive an equation that will approximate the missing L_2 frame for each case. We are using the example that has been illustrated in Figure 2.1 for all cases. In other words, we will produce an equation that will reconstruct the missing L_2 frame. Before we proceed, we

repeat equations (1.1) and (1.2) below for readers' convenience. Please note that these two equations are only valid for those pixels that are bi-directionally connected.

$$H_t(\vec{n}) = A_t(\vec{n}) - \frac{1}{2}\left[B_t(\vec{n} + \overrightarrow{df_t}) + B_{t-1}(\vec{n} + \overrightarrow{dr_{t-1}})\right] \tag{1.1}$$

$$L_t(\vec{n}) = B_t(\vec{n}) + \frac{1}{4}\left[H_{t+1}(\vec{n} - \overrightarrow{dr_t}) + H_t(\vec{n} - \overrightarrow{df_t})\right] \tag{1.2}$$

The following picture is a copy of Figure 2.1.

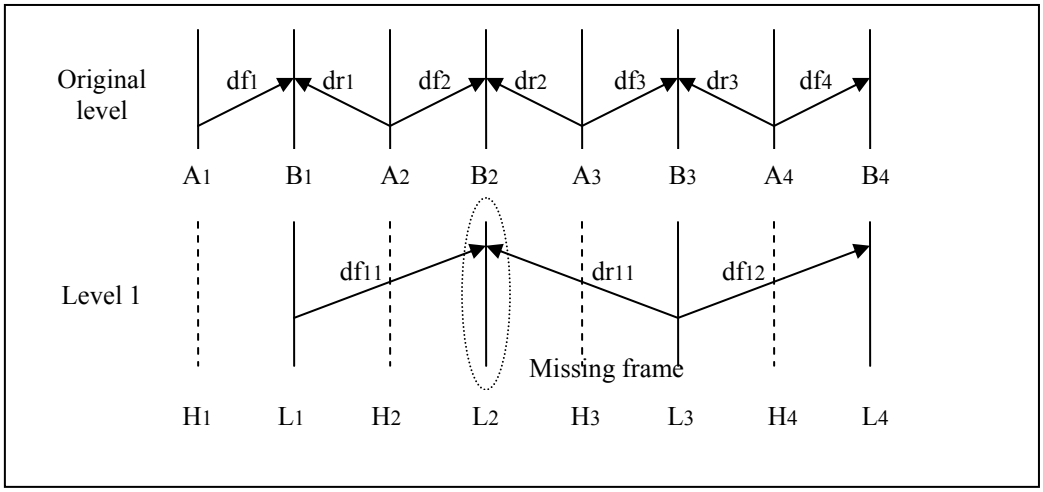


Figure 2.1: Illustration of 5/3 MCTF Error Concealment

In all three cases stated above, the procedure to derive the error concealment equations consists of the following three steps.

1. **We express B_2 frame using equation (1.1).**

   This will allow us to express B_2 frame in terms of *A*- and *H*-frames.

2. **We use Approximation 2 to replace *A*-frames by neighboring *B*-frames.**

   We will then have equations for the B_2 frame composed of *B*- and *H*-frames only.

3. **We then use equation (1.2) to express all *B*-frames in terms of *L*- and *H*-frames.** As a result, the final equation for the missing L_2 frame will contain neighboring *L*- and *H*-frames only. This is exactly what we need since (unlike *A*- and *B*-frames) these frames exist in the compressed bitstream.

2.1 Case 1: Approximation 1 is valid and Approximation 2 is valid

In this section, we will approximate the missing L_2 frame using its temporally neighboring *L*-frames and motion vectors. We assume that both Approximation 1 and Approximation 2 are valid. Starting from the 5/3 MCTF equations,

$$H_t(\vec{n}) = A_t(\vec{n}) - \frac{1}{2}\left[B_t(\vec{n} + \overrightarrow{df_t}) + B_{t-1}(\vec{n} + \overrightarrow{dr_{t-1}})\right], \tag{1.1}$$

$$L_t(\vec{n}) = B_t(\vec{n}) + \frac{1}{4}\left[H_{t+1}(\vec{n} - \overrightarrow{dr_t}) + H_t(\vec{n} - \overrightarrow{df_t})\right], \tag{1.2}$$

the first step of the derivation of error concealment equations is to express H_2 and H_3 frames using equation (1.1):

$$H_2(\vec{n}) = A_2(\vec{n}) - \frac{1}{2}\left[B_2(\vec{n} + \overrightarrow{df_2}) + B_1(\vec{n} + \overrightarrow{dr_1})\right],$$

$$H_3(\vec{n}) = A_3(\vec{n}) - \frac{1}{2}\left[B_3(\vec{n} + \overrightarrow{df_3}) + B_2(\vec{n} + \overrightarrow{dr_2})\right].$$

(2.3)

Since we are assuming that all *H*-frames are zero, the two equations in (2.3) would change as below.

$$A_2(\vec{n}) = \frac{1}{2}\left[B_2(\vec{n} + \overrightarrow{df_2}) + B_1(\vec{n} + \overrightarrow{dr_1})\right],$$

$$A_3(\vec{n}) = \frac{1}{2}\left[B_3(\vec{n} + \overrightarrow{df_3}) + B_2(\vec{n} + \overrightarrow{dr_2})\right].$$

(2.4)

We can reorganize (2.4) so that we have B_2 frame on one side of the equation and other terms on the opposite side.

$$B_2(\vec{n} + \overrightarrow{df_2}) = 2 \cdot A_2(\vec{n}) - B_1(\vec{n} + \overrightarrow{dr_1}),$$

$$B_2(\vec{n} + \overrightarrow{dr_2}) = 2 \cdot A_3(\vec{n}) - B_3(\vec{n} + \overrightarrow{df_3}).$$

(2.5)

We can now apply Approximation 2, which allows us to approximate *A*-frame from its neighboring *B*-frames.

$$A_2(\vec{n}) \approx B_1(\vec{n} + \overrightarrow{dr_1}),$$

$$A_3(\vec{n}) \approx B_3(\vec{n} + \overrightarrow{df_3}).$$

(2.6)

Substituting (2.6) into (2.5) gives

$$B_2(\vec{n}) \approx B_1(\vec{n} + \overrightarrow{df_{11}}),$$

(2.7)

and

$$B_2(\vec{n}) \approx B_3(\vec{n} + \overrightarrow{dr_{11}}),$$

(2.8)

where $\overrightarrow{df_{11}} = \overrightarrow{df}_2 - \overrightarrow{dr_1}$ and $\overrightarrow{dr_{11}} = \overrightarrow{dr_2} - \overrightarrow{df_3}$.

Equations (2.7) and (2.8) are approximations of B_2 frame from its neighboring B_1 and B_3 frames, respectively, using motion vectors that correspond to twice the temporal distance between the neighboring *A*- and *B*-frames. Equation (2.7) would be valid for pixels that are left-connected and equation (2.8) for pixels that are right-connected. For those pixels that are bi-directionally connected (i.e., both left- and right-connected), we can average (2.7) and (2.8) to approximate B_2. The equation becomes

$$B_2(\vec{n}) \approx \frac{1}{2}\left[B_1(\vec{n} + \overrightarrow{df_{11}}) + B_3(\vec{n} + \overrightarrow{dr_{11}})\right].$$

(2.9)

We now approximate L_2 by B_2 using equation (1.2). Remember that we are still assuming that all *H*-frames are zero. Therefore, equation (1.2) becomes

$$L_t(\vec{n}) \approx B_t(\vec{n}).$$

(2.10)

We can now substitute (2.10) into (2.9) in order to get an approximation for $L_2(\vec{n})$,

$$L_2(\vec{n}) \approx \frac{1}{2}\left[L_1(\vec{n} + \overrightarrow{df_{11}}) + L_3(\vec{n} + \overrightarrow{dr_{11}})\right]. \quad (2.11)$$

Equation (2.11) is an approximation of L_2 from its neighboring *L*-frames. It is valid for pixels that are bi-directionally connected. We can create equations for left or right-connected pixels in a similar manner. We substitute (2.10) into (2.7) or (2.8) to obtain such equations. Then, error concealment equation for left-connected pixels would be

$$L_2(\vec{n}) \approx L_1(\vec{n} + \overrightarrow{df_{11}}),$$

and for right-connected pixels,

$$L_2(\vec{n}) \approx L_3(\vec{n} + \overrightarrow{dr_{11}}).$$

Theoretically, there doesn't seem to be any method to recover the unconnected pixels, so we must use a more practical approach to solve this problem, as discussed in Chapter 3.

2.2 Case 2: Approximation 1 is valid and Approximation 2 is not valid

In this section, we derive reconstruction equations for L_2 while assuming that Approximation 1 is valid and Approximation 2 is not valid: (1) $H_t(\vec{n}) \approx 0$, (2) $A_t(\vec{n}) \neq B_t(\vec{n} + \vec{df}_t)$ and $A_{t+1}(\vec{n}) \neq B_t(\vec{n} + \vec{dr}_t)$. The procedure to derive the error concealment equations for Case 2 is same as that of Case 1 up to equation (2.5). Therefore, we will proceed from equation (2.5) in this case:

$$B_2(\vec{n} + \overrightarrow{df_2}) = 2 \cdot A_2(\vec{n}) - B_1(\vec{n} + \overrightarrow{dr_1}), \tag{2.5}$$

$$B_2(\vec{n} + \overrightarrow{dr_2}) = 2 \cdot A_3(\vec{n}) - B_3(\vec{n} + \overrightarrow{df_3}).$$

We now have to express *A*-frames in equation (2.5) in terms of *B*-frames. Unlike Case 1, we cannot use a direct approximation from *A*- to *B*-frame; we must find another way. We use a random field to describe the relationship between *A*- and *B*-frame. As the results in Appendix A show, the difference between connected pixels in the neighboring *A*- and *B*-frames is quite random.

$$A_2(\vec{n}) = B_1(\vec{n} + \overrightarrow{dr_1}) + C_1(\vec{n}),$$

$$A_3(\vec{n}) = B_3(\vec{n} + \overrightarrow{df_3}) + C_2(\vec{n}).$$

(2.12)

In (2.12), $C_i(\vec{n})$ is a zero-mean random field with empirical standard deviation of 6.86. Note that the above equations are identical to (2.6) except that random fields are added to model approximation errors. Hence, (2.12) is no longer an approximation as a result of using the random fields. We substitute equations (2.12), into equations (2.5), which gives

$$B_2(\vec{n}) = B_1(\vec{n} + \overrightarrow{df_{11}}) + 2 \cdot C_1(\vec{n}),$$

$$B_2(\vec{n}) = B_3(\vec{n} + \overrightarrow{dr_{11}}) + 2 \cdot C_2(\vec{n}).$$

(2.13)

where $\overrightarrow{df_{11}} = \overrightarrow{df}_2 - \overrightarrow{dr_1}$ and $\overrightarrow{dr_{11}} = \overrightarrow{dr_2} - \overrightarrow{df_3}$.

Similarly to Case 1, we can average the two equations in (2.13) to approximate the B_2 frame for bi-directionally connected pixels:

$$B_2(\vec{n}) = \frac{1}{2}\left[B_1(\vec{n} + \overrightarrow{df_{11}}) + B_3(\vec{n} + \overrightarrow{dr_{11}})\right] + \left[C_1(\vec{n}) + C_2(\vec{n})\right].$$

(2.14)

The next step would be expressing *B*-frames in terms of *L*-frames using equation (1.2). Since Case 2 still assumes all *H*-frames are zero, we can get the final equation similar to Case 1:

$$L_2(\vec{n}) \approx \frac{1}{2}\left[L_1(\vec{n}+\overrightarrow{df_{11}}) + L_3(\vec{n}+\overrightarrow{dr_{11}})\right] + \left[C_1(\vec{n}) + C_2(\vec{n})\right].$$

(2.15)

Equation (2.15) can only be applied to bi-directionally connected pixels. For left or right-connected pixels, similar procedure can be performed. For left-connected pixels, we can use

$$L_2(\vec{n}) \approx L_1(\vec{n}+\overrightarrow{df_{11}}) + 2 \cdot C_1(\vec{n}),$$

and for right-connected pixels,

$$L_2(\vec{n}) \approx L_3(\vec{n}+\overrightarrow{dr_{11}}) + 2 \cdot C_2(\vec{n}).$$

We will discuss the handling of the unconnected pixels in Chapter 3.

2.3 Case 3: Approximation 1 is not valid and Approximation 2 is not valid

In this section, our goal is to approximate the missing L_2 frame from the neighboring *L*-frames without using the two approximations mentioned above. We do this by first expressing the frame B_2 (which is just above L_2 in the temporal pyramid) from the neighboring *B*-frames, and then relating *B*-frames to the *L*-frames. Starting from the 5/3 MCTF equations,

$$H_t(\vec{n}) = A_t(\vec{n}) - \frac{1}{2}\left[B_t(\vec{n} + \overrightarrow{df_t}) + B_{t-1}(\vec{n} + \overrightarrow{dr_{t-1}})\right], \tag{1.1}$$

$$L_t(\vec{n}) = B_t(\vec{n}) + \frac{1}{4}\left[H_{t+1}(\vec{n} - \overrightarrow{dr_t}) + H_t(\vec{n} - \overrightarrow{df_t})\right], \tag{1.2}$$

we can use equation (1.1) to express H_2 and H_3 frames in terms of *A*- and *B*-frames. The result is identical to equation (2.3).

$$H_2(\vec{n}) = A_2(\vec{n}) - \frac{1}{2}\left[B_2(\vec{n} + \overrightarrow{df_2}) + B_1(\vec{n} + \overrightarrow{dr_1})\right]$$

$$H_3(\vec{n}) = A_3(\vec{n}) - \frac{1}{2}\left[B_3(\vec{n} + \overrightarrow{df_3}) + B_2(\vec{n} + \overrightarrow{dr_2})\right] \tag{2.3}$$

Next, we reorganize (2.3) so that we have B_2 frame on one side of the equation, and other terms on the opposite side:

$$B_2(\vec{n} + \overrightarrow{df_2}) = 2 \cdot \left[A_2(\vec{n}) - H_2(\vec{n})\right] - B_1(\vec{n} + \overrightarrow{dr_1}),$$

$$B_2(\vec{n} + \overrightarrow{dr_2}) = 2 \cdot \left[A_3(\vec{n}) - H_3(\vec{n})\right] - B_3(\vec{n} + \overrightarrow{df_3}).$$

(2.16)

Also, we can represent *A*-frames by warping the neighboring *B*-frames along motion vectors as

$$A_2(\vec{n}) = B_1(\vec{n} + \overrightarrow{dr_1}) + C_1(\vec{n}),$$

$$A_3(\vec{n}) = B_3(\vec{n} + \overrightarrow{df_3}) + C_2(\vec{n}),$$

(2.17)

where $C_1(\vec{n})$ and $C_2(\vec{n})$ are a zero-mean random fields with empirical standard deviation of 6.86. Substituting (2.17) into (2.16) gives

$$B_2(\vec{n}) = B_1(\vec{n} + \overrightarrow{df_{11}}) - 2 \cdot H_2(\vec{n} - \overrightarrow{df_2}) + 2 \cdot C_1(\vec{n}),$$

$$B_2(\vec{n}) = B_3(\vec{n} + \overrightarrow{dr_{11}}) - 2 \cdot H_3(\vec{n} - \overrightarrow{dr_2}) + 2 \cdot C_2(\vec{n}),$$

(2.18)

where $\overrightarrow{df_{11}} = \overrightarrow{df}_2 - \overrightarrow{dr_1}$ and $\overrightarrow{dr_{11}} = \overrightarrow{dr_2} - \overrightarrow{df_3}$.

Now we focus on the frame B_2, and the connections made from B_2 through df_{11} and dr_{11} to the frames B_1 and B_3. Some of the pixels in B_2 may be bi-connected, some may be left-connected, some right-connected, and some unconnected. We can use the first equation of (2.18) for left-connected pixels and the second equation for right-connected pixels. For bi-directionally connected pixels, we can use the average of the two equations in (2.18), which gives us

$$B_2(\vec{n}) = \frac{1}{2}\left[B_1(\vec{n} + \overrightarrow{df_{11}}) + B_3(\vec{n} + \overrightarrow{dr_{11}})\right] - \left[H_2(\vec{n} - \overrightarrow{df_2}) + H_3(\vec{n} - \overrightarrow{dr_2})\right] + \left[C_1(\vec{n}) + C_2(\vec{n})\right].$$

(2.19)

Finally, unconnected pixels in B_2 (and, therefore, in the missing L_2) will have to be dealt with in other ways, which we will explain in Chapter 3.

Now, we will express *L*-frames using *B*-frames from equation (1.2). We can reorganize (1.2) to produce the equation below.

$$B_t(\vec{n}) = L_t(\vec{n}) - \frac{1}{4}\left[H_{t+1}(\vec{n} - \overrightarrow{dr_t}) + H_t(\vec{n} - \overrightarrow{df_t})\right].$$

(2.20)

We have to relate three *B*-frames in equation (2.19) to the corresponding *L*-frames. These *B*-frames can be expressed as below.

$$B_2(\vec{n}) = L_2(\vec{n}) - \frac{1}{4}\left[H_3(\vec{n} - \overrightarrow{dr_2}) + H_2(\vec{n} - \overrightarrow{df_2})\right]$$

(2.21)

$$B_1(\vec{n}) = L_1(\vec{n}) - \frac{1}{4}\left[H_2(\vec{n} - \overrightarrow{dr_1}) + H_1(\vec{n} - \overrightarrow{df_1})\right]$$

(2.22)

$$B_3(\vec{n}) = L_3(\vec{n}) - \frac{1}{4}\left[H_4(\vec{n} - \overrightarrow{dr_3}) + H_3(\vec{n} - \overrightarrow{df_3})\right]$$

(2.23)

However, equation (2.19) contains $B_1(\vec{n} + \overrightarrow{df_{11}})$ instead of $B_1(\vec{n})$, and $B_3(\vec{n} + \overrightarrow{dr_{11}})$ instead of $B_3(\vec{n})$. Therefore, all frames in (2.22) and (2.23) must be warped along $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$, respectively. Note that by definition of $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$, we have $\overrightarrow{df_{11}} + \overrightarrow{dr_1} = \overrightarrow{df}_2$ and $\overrightarrow{dr_{11}} + \overrightarrow{df_3} = \overrightarrow{dr_2}$, so

$$B_1(\vec{n} - \overrightarrow{df_{11}}) = L_1(\vec{n} - \overrightarrow{df_{11}}) - \frac{1}{4}\left[H_2(\vec{n} - \overrightarrow{df_{11}} - \overrightarrow{dr_1}) + H_1(\vec{n} - \overrightarrow{df_{11}} - \overrightarrow{df_1})\right]$$

$$= L_1(\vec{n} - \overrightarrow{df_{11}}) - \frac{1}{4}\left[H_2(\vec{n} - \overrightarrow{df_2}) + H_1(\vec{n} - \overrightarrow{df_{11}} - \overrightarrow{df_1})\right],$$

(2.24)

$$B_3(\vec{n} - \overrightarrow{dr_{11}}) = L_3(\vec{n} - \overrightarrow{dr_{11}}) - \frac{1}{4}\left[H_4(\vec{n} - \overrightarrow{dr_{11}} - \overrightarrow{dr_3}) + H_3(\vec{n} - \overrightarrow{dr_{11}} - \overrightarrow{df_3})\right]$$

$$= L_3(\vec{n} - \overrightarrow{dr_{11}}) - \frac{1}{4}\left[H_4(\vec{n} - \overrightarrow{dr_{11}} - \overrightarrow{dr_3}) + H_3(\vec{n} - \overrightarrow{dr_2})\right].$$

(2.25)

We can now substitute (2.21), (2.24), and (2.25) into (2.19), which produces

$$L_2(\vec{n}) = \frac{1}{2}\left[L_1(\vec{n} - \overrightarrow{df_{11}}) + L_3(\vec{n} - \overrightarrow{dr_{11}})\right] - \frac{9}{8}\left[H_2(\vec{n} - \overrightarrow{df_2}) + H_3(\vec{n} - \overrightarrow{dr_2})\right]$$

$$+ \frac{1}{8}\left[H_1(\vec{n} - \overrightarrow{df_{11}} - \overrightarrow{df_1}) + H_4(\vec{n} - \overrightarrow{dr_{11}} - \overrightarrow{dr_3})\right] + \left[C_1(\vec{n}) + C_2(\vec{n})\right].$$

(2.26)

Equation (2.26) above can be applied to those pixels that are bi-directionally connected via $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$ to the neighboring frames. Equations for left- or right-connected pixels can be derived in a similar manner from equation (2.18). For left-connected pixels,

$$L_2(\vec{n}) = L_1(\vec{n} - \overrightarrow{df_{11}}) - \frac{9}{4}H_2(\vec{n} - \overrightarrow{df_2}) + \frac{1}{4}\left[H_1(\vec{n} - \overrightarrow{df_{11}} - \overrightarrow{df_1})\right] + 2 \cdot C_1(\vec{n}),$$

and for right-connected pixels

$$L_2(\vec{n}) = L_3(\vec{n} - \overrightarrow{dr_{11}}) - \frac{9}{4}H_3(\vec{n} - \overrightarrow{dr_2}) + \frac{1}{4}H_4(\vec{n} - \overrightarrow{dr_{11}} - \overrightarrow{dr_3}) + 2 \cdot C_2(\vec{n}).$$

In all three equations, *H*-frames can also be modeled as random noise fields with zero mean value and empirical standard deviation of 3.77 (Please refer to Appendix A). We will discuss this issue and the treatment of unconnected pixels in Chapter 3.

Chapter 3

Implementation of 5/3 Motion-Compensated Error Concealment

Equations that we have derived in Chapter 2, such as (2.11), (2.15), and (2.26), can only be applied to those pixels that are connected to neighboring frames via motion vector fields df_{11} and/or dr_{11}. However, not every pixel in the missing *L*-frame can be connected to the neighboring *L*-frames. Such unconnected pixels cannot be reconstructed by the equations we introduced in Chapter 2.

In this chapter, we introduce practical methods to recover the missing *L*-frame. We can recover those connected pixels (pixels that have motion vector connections via df_{11} or dr_{11} to their neighboring frames) by performing the equations introduced in Chapter 2. We also introduce a few methods to guess unconnected pixel values.

It is apparent from equations (2.11), (2.15), and (2.26) that the performance of motion-compensated error concealment critically depends on motion information, especially on the motion vector fields df_{11} and dr_{11}. Accurate motion vectors will enhance the performance of the error concealment system. All three equations that we

derived in Chapter 2 share a common property that they search pixel values of a missing frame in the neighboring *L*- or *H*-frames using motion vectors. Let us use Figure 2.1 and equation (2.11) as an illustration.

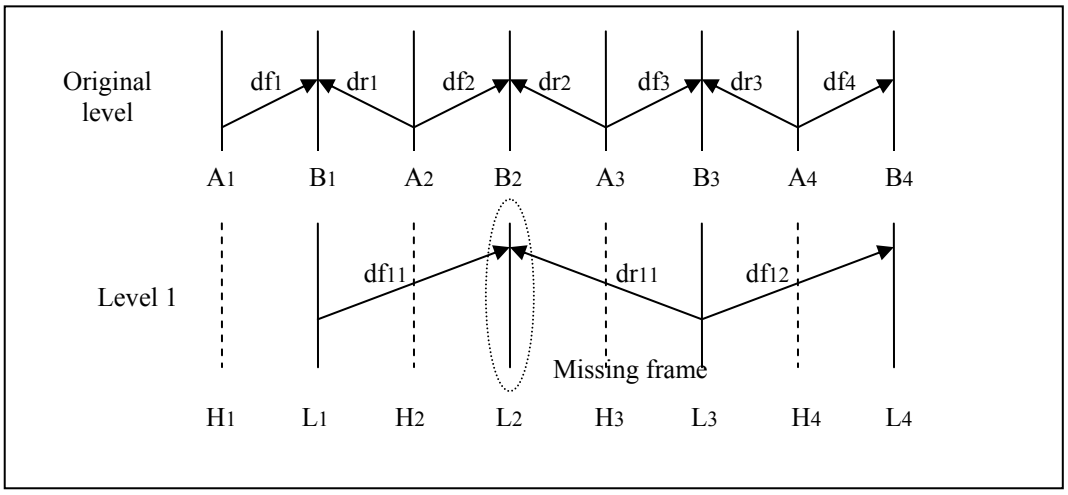


Figure 2.1: Illustration of 5/3 MCTF Error Concealment

$$L_2(\vec{n}) \approx \frac{1}{2}\left[L_1(\vec{n}+\overrightarrow{df_{11}}) + L_3(\vec{n}+\overrightarrow{dr_{11}})\right] \tag{2.11}$$

This equation says that we can recover the pixels of L_2 by averaging pixel values from L_1 and L_3 using motion vectors fields $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$, respectively. The key to successfully implementing (2.11) is to have accurate motion vectors $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$. However, since the *L*-frames are at the last level of the temporal pyramid, these motion vectors at this level are not available in the coded bitstream. Hence, $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$ do not exist at the decoder. We need to find ways to create $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$, and the quality of these motion vectors will determine the performance of the error concealment system. In this chapter, we

introduce three different methods to produce motion vectors in the last temporal level; $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$.

Method 1: Motion Concatenation

Method 2: Motion Re-estimation

Method 3: Zero Motion Error Concealment

We will explain each of these three methods using (2.11) and Figure 2.1.

3.1 Method 1: Motion Concatenation

Motion concatenation will create $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$ by temporally concatenating motion vectors of the previous level, which were generated at the encoder and exist in the coded bitstream; $\overrightarrow{df}_2$, $\overrightarrow{dr}_1$, $\overrightarrow{dr}_2$ and $\overrightarrow{df}_3$. We mentioned in Chapter 1 that one way to obtain $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$ is as follows:

$$\overrightarrow{df_{11}} = \overrightarrow{df}_2 - \overrightarrow{dr_1},$$

$$\overrightarrow{dr_{11}} = \overrightarrow{dr_2} - \overrightarrow{df_3}.$$

Motion concatenation is the implementation of these two equations. Since motion concatenation aggregates motion vectors of the previous MCTF level, which were produced at the encoder, it is likely to synthesize fairly accurate motion vector fields. Figure 3.1 illustrates the idea of motion concatenation.

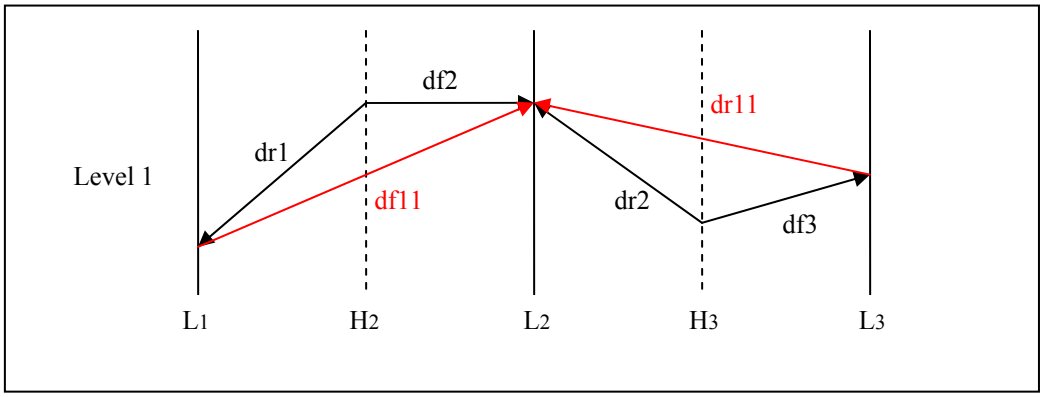


Figure 3.1: Illustration of Motion Concatenation

Figure 3.2: Frame that is reconstructed by Motion Concatenation only (left) and the coded loss-free frame (right)

Figure 3.2 (left) illustrates the result of error concealment using motion concatenation on one frame of the *Stefan* sequence. The corresponding coded loss-free frame is shown in the right part of the figure for comparison. The sequence was encoded using the MC-EZBC coder [6], [7], with one level of MCTF. One *L*-frame was removed from the sequence and then reconstructed using motion concatenation. The figure shows the *B*-frame immediately above

this reconstructed *L*-frame. We can observe there are some green areas and black thin lines in the reconstructed frame. These areas represent the locations of unconnected pixels which could not be found by motion concatenation, because one or both of the corresponding motion vectors do not exist at the upper MCTF level. Observe that most unconnected pixels are located around the tennis player. The reason is that the background is moving relatively slowly compared to the tennis player, so pixels in the background are more likely to be connected.

3.2 Method 2: Motion Re-estimation

The second method, motion re-estimation, will generate $\overrightarrow{df_{11}}$ and $\overrightarrow{dr_{11}}$ by performing motion estimation between L_1 and L_3 at the decoder. This method complements motion concatenation since it does not use motion vectors transmitted from the encoder, but creates new motion vectors at the decoder. The idea of motion re-estimation is illustrated in the following figure.

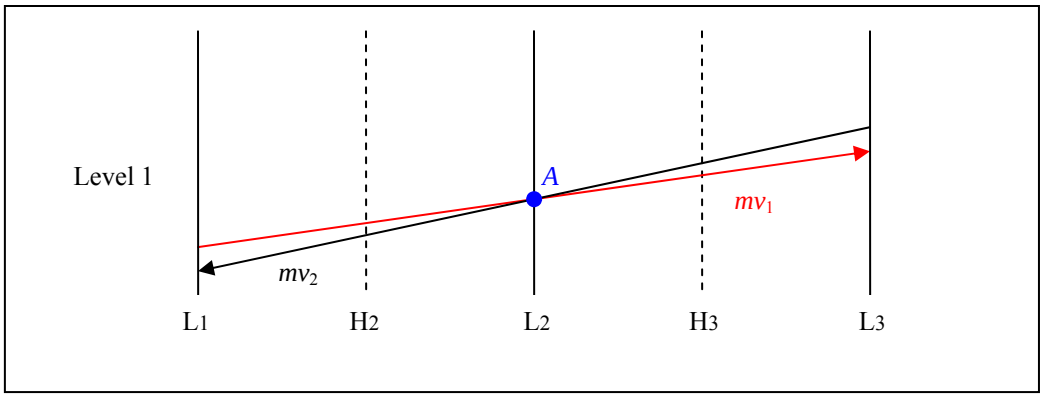


Figure 3.3: Illustration of Motion Re-estimation

Motion estimation between L_1 and L_3 produces two motion vector fields, mv_1 and mv_2, which have opposite directions. One field is created by putting L_1 as the "current frame" and L_3 as the "reference frame" into the motion estimator, the other by putting L_1 as the "reference frame" and L_3 as the "current frame." To reconstruct L_2, we synthesize a frame halfway between L_1 and L_3. To do this, we need to create motion fields df_{11} and dr_{11} from mv_1 and mv_2. Figure 3.3 shows a possible situation where a pixel location *A* in the L_2 frame is covered by two different motion vectors, one from mv_1, the other from mv_2. In this case, we have several options to create df_{11} and dr_{11} at pixel location *A*. The df_{11} motion vector can be created by dividing mv_1 in half, or by reversing the direction of mv_2 and dividing it in half. Mathematically, the first option would be $df_{11} = mv_1 / 2$ and the second would be $df_{11} = -mv_2 / 2$. Then the pixel value at location *A* would depend on which motion vector we adopt. Another option is to average the two candidate pixel values (one based on mv_1, the other based on mv_2) and use the average as the estimate of the missing pixel. A similar operation could be performed for dr_{11} as well. In our system, we use the average if there are multiple candidate pixel values at any one pixel location of a missing frame. If there is only one candidate pixel value, we use that value at the corresponding location.

Both motion vector fields produced by motion re-estimation (mv_1 and mv_2) may lead to unconnected pixels in the missing L_2 frame. However, the locations of unconnected pixels in mv_1 need not be the same as the locations of unconnected pixels in mv_2. Hence, some pixel

locations that are unconnected in mv_1 might be connected in mv_2 and vice versa. Therefore, creating both of these fields maximizes the total number of connected pixels.

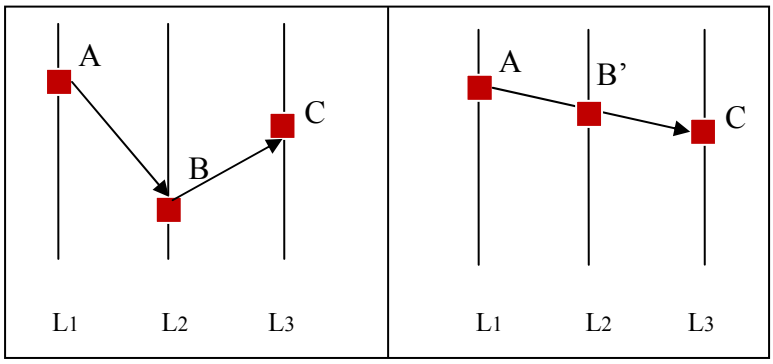
Figure 3.4 Frame that is reconstructed by Motion Re-estimation (left) and the coded loss-free frame (right)

Figure 3.4 above shows a frame that was reconstructed by motion re-estimation. The setup is the same as in the example with motion concatenation as we shown in Figure 3.3. The green areas, again, represent pixel locations which could not be reconstructed by motion re-estimation.

3.2.1 Comparison between Motion Concatenation and Motion Re-estimation

In the previous two sections, we have introduced two methods to estimate the motion between the missing *L*-frame and its neighboring *L*-frames. The natures of these two methods are different. Motion re-estimation assumes that pixels of the missing *L*-frame are located halfway along the motion vectors between its neighboring *L*-frames. On the other hand,

objects' position predicted by motion concatenation should be more accurate, since this method uses motion vectors generated at the encoder.



(a) **(b)**

Figure 3.5 A comparison between (a) Motion Concatenation, and (b) Motion Re-estimation

The figure above shows a situation where motion concatenation will produce more accurate motion estimate compared to motion re-estimation. Suppose that block *A* moves through *B* to *C* as in part (a); then motion concatenation is more likely to track this position *B*, since it uses motion vectors from the upper MCTF level which, hopefully, contain a relatively accurate trajectory of the block. Motion re-estimation will incorrectly move the block *A* to position *B'*, midway between *A* and *C*. We expect to observe this problem more often with motion re-estimation as the number of MCTF levels increases.

In addition to the accuracy of estimated motion, the two methods also differ in their ability to find connections for the pixels from fast moving objects. Comparing the reconstructed frames in Figure 3.4 and Figure 3.2, we observe that motion concatenation can reconstruct

more pixels in the fast moving tennis player and his racket. We have observed this phenomenon in several other sequences.

On the other hand, motion re-estimation has an advantage over motion concatenation on slow, uniformly moving objects, such as the background in our example. In Figure 3.2 (left) we see that there are some thin lines of unconnected pixels. These lines are usually one or two pixels wide and they are only observed in frames that are reconstructed using motion concatenation. These lines negatively affect visual quality because they need to be filled up using other methods (for example, by spatial interpolation), and may lead to additional visual distortion. We call this *line distortion*. Meanwhile, motion re-estimation tends to produce better quality video without line distortion on slow moving objects.

3.3 Method 3: Zero Motion Compensated Error Concealment

Zero motion compensated (MC) error concealment is the simplest but the worst performing mechanism. It assumes that there is no motion between L_1 and L_3, and simply averages L_1 and L_3 to reconstruct L_2. While the assumption of zero motion between L_1 and L_3 is clearly not correct in most cases, zero-motion concealment does have one other good feature in addition to its simplicity. This is the fact that there is no problem of unconnected pixels in this method, since each pixel in the missing L_2 frame is reconstructed as the average of the two pixels at the same spatial location in the neighboring *L*-frames.

Due to the absence of the unconnected pixel problem, we use zero-motion concealment as a benchmark method to evaluate the performance of our motion-compensated error concealment systems.

Figure 3.6: Frame reconstructed by zero-motion error concealment (left) and the coded loss-free frame (right)

The figure above shows the frame from same example as Figures 3.2 and 3.4, this time reconstructed by zero-motion error concealment. We can see that zero-motion concealment performs worse compared to the previous two methods on those pixels that the other two methods are able to recover, but it does not suffer from the unconnected pixel problem.

3.4 Combination of motion concatenation, motion re-estimation and zero-motion error concealment

In order to produce a complete frame without unconnected pixels (green “holes” as in Figure 3.4 and Figure 3.2), we combine all three methods previously discussed. The procedure that

we use in this text is as follows. We first fill out the L_2 frame as much as we can with motion concatenation. The reason for using motion concatenation ahead of the other two methods is its higher accuracy. However, motion concatenation produces line distortion, as discussed in Section 3.2.1. We can minimize the effect of line distortion by interpolating across those lines from the neighboring pixels. After motion concatenation, some "holes" are left in the reconstructed frame. We fill as many "holes" as possible using motion re-estimation. If there are any remaining "holes" left, we fill these by averaging the co-located pixels in the neighboring *L*-frames (i.e., using zero-motion error concealment). There may be other methods we can use to fill those "holes" remaining after motion concatenation and motion re-estimation. We leave this task for our future research; some possible solutions to this problem will be briefly discussed in Chapter 5. The following figure shows the reconstructed frame produced by the complete error concealment system using the same example as before.]

Figure 3.7: Frame that is reconstructed by motion concatenation followed by motion re-estimation and zero-motion error concealment (left) and the coded loss-free frame (right)

Chapter 4

Experimental Results

We tested the proposed 5/3 motion compensated error concealment system on several YUV-format sequences. The frame size of tested sequences is CIF (352 × 288 pixels) and the bit rate was set to 1000 Kbps for every sequence. The tested sequences are *Coastguard*, *Foreman*, *Mobile Calendar*, and *Stefan*. *Coastguard* has characteristics of relatively slow and constant movement of objects. *Forman* contains relatively higher and more complex motion of both the object (foreman's head) and the camera, which makes this sequence challenging for error concealment. *Mobile Calendar* can be characterized as relatively slow and constant movement and lots of texture. *Stefan* contains both slow and fast movement. In this sequence, the background audience moves relatively slowly, while the tennis player and his racket move relatively fast.

Our goal in this chapter is to test the three error concealment equations derived in Chapter 2. We will also test different orders of applying the three methods for motion estimation between the missing *L*-frame and the neighboring *L*-frames (motion concatenation, motion re-estimation and zero-motion) presented in Chapter 3. The test results are reported in terms of the Peak Signal-to-Noise Ratio (PSNR) of the Y-component, as well as subjective visual quality of each sequence.

4.1 Test Result of Case 1 and Case 2 of Chapter 2

Case 1 and Case 2, which were discussed in Sections 2.1 and 2.2 respectively, share a common assumption: $H_t(\vec{n}) \approx 0$. The difference between the two cases is that Case 1 assumes $A_t(\vec{n}) \approx B_t(\vec{n} + \vec{df}_t)$, while Case 2 does not. As a result, we derived the final equation for Case 1 is

$$L_2(\vec{n}) \approx \frac{1}{2}\left[L_1(\vec{n} + \overrightarrow{df_{11}}) + L_3(\vec{n} + \overrightarrow{dr_{11}})\right], \tag{2.11}$$

and for Case 2 it is

$$L_2(\vec{n}) \approx \frac{1}{2}\left[L_1(\vec{n} + \overrightarrow{df_{11}}) + L_3(\vec{n} + \overrightarrow{dr_{11}})\right] + \left[C_1(\vec{n}) + C_2(\vec{n})\right]. \tag{2.15}$$

In (2.15), $C_i(\vec{n})$ represent zero-mean random fields with empirical standard deviation value of 6.86. In our current implementation of the error concealment system, we do not estimate the values of $C_i(\vec{n})$, but rather set them to zero. This makes equations (2.11) and (2.15) the same for implementation purposes. Estimating the values of the random fields for improved concealment performance is the topic for future research.

The order of operations in the error concealment system is the following – motion concatenation, followed by motion re-estimation and then zero-motion error concealment. Y-component PSNR and visual quality results of this system will be compared against the benchmark system which is constructed using only zero-motion error concealment.

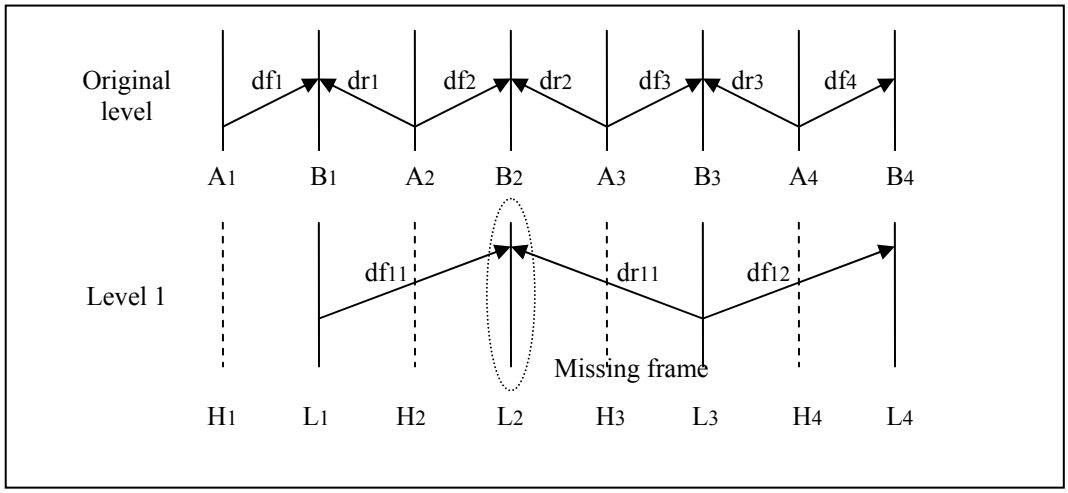


Figure 2.1: Illustration of 5/3 MCTF Error Concealment

Figure 2.1 is repeated above for readers' convenience. Suppose that L_2 frame is damaged during transmission. To simulate this situation, we remove L_2 from the compressed bitstream at the decoder and reconstruct it using its neighboring *L*-frames, L_1 and L_3. The decoder will use the reconstructed L_2 frame and its neighboring *L*- and *H*-frames to recreate the original frames, A_i and B_i. As a result of error concealment, PSNR value and visual quality of the decoded *A*- and *B*-frames will be degraded compared to its coded loss-free version. Due to the nature of the 5/3 MCTF system, the *B*-frame which is directly above the missing *L*-frame will suffer most degradation in both PSNR and visual quality. In each sequence, we remove one of the *L*-frames, reconstruct it using the neighboring *L*-frames, decode the sequence, and record the PSNR value for the *B*-frame directly above the concealed *L*-frame. We then remove the next *L*-frame, and repeat this procedure until all the *L*-frames have been removed. The PSNR graphs in the remainder of this chapter show the

PSNR values of the *B*-frames directly above the concealed *L*-frames. This procedure allows us to examine error concealment performance throughout the test sequence.

4.1.1 Y-component PSNR Results

Horizontal axes in the graphs below show the index of the *L*-frame where error concealment is performed and the vertical axes represent Y-component PSNR in dB of the *B*-frame directly above the concealed *L*-frame. Figure 4.1 shows the results for one-level MCTF, Figure 4.2 shows the results for two-level MCTF and Figure 4.3 shows the results for three-level MCTF. All three figures are based on the *Coastguard* sequence. Advantage with respect to zero-motion error concealment can be up to 6 dB.

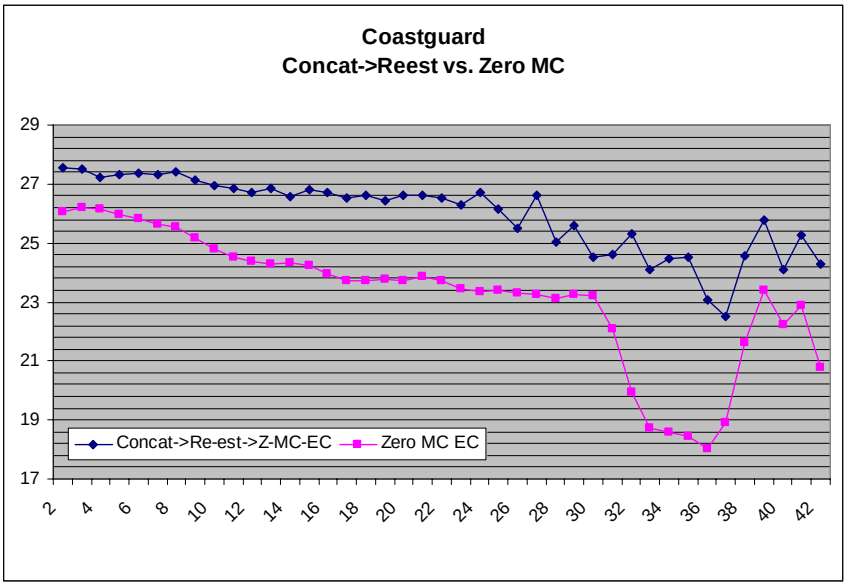


Figure 4.1: Y-component PSNR graph of error concealment system versus zero-motion error concealment for one-level MCTF

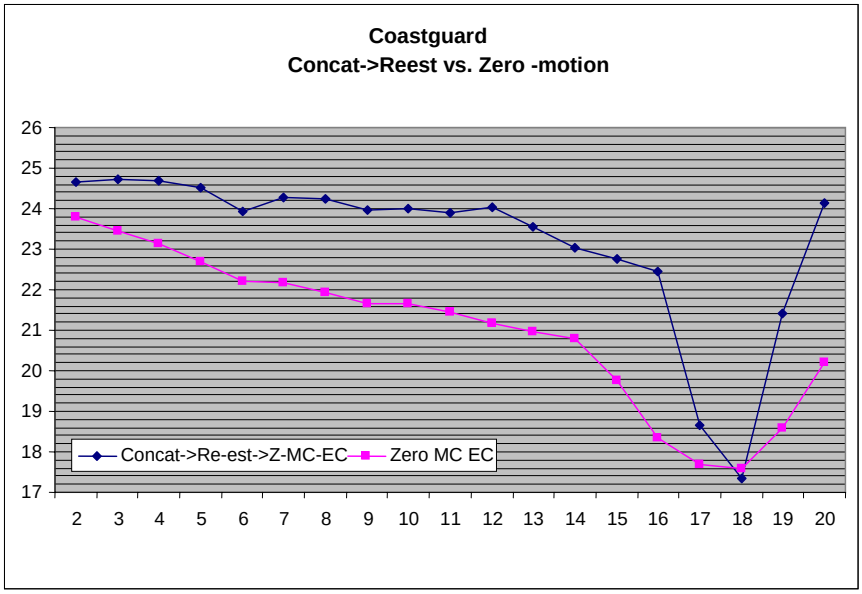


Figure 4.2: Y-component PSNR graph of error concealment system versus zero-motion error concealment for two-level MCTF

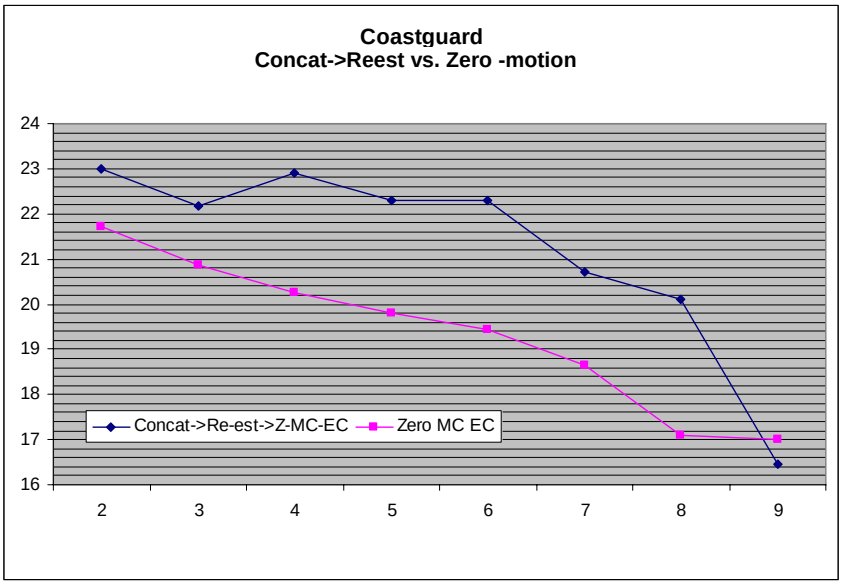


Figure 4.3: Y-component PSNR graph of error concealment system versus zero-motion error concealment for three-level MCTF

4.1.2 Visual Quality Results

Figure 4.4 - Figure **4.6** below show visual quality results. In each figure, the frame in the left part is the *B*-frame directly above the concealed *L*-frame reconstructed by the complete error concealment system, while the frame in the right part is the corresponding frame recovered by zero-motion error concealment. Figure 4.4 corresponds to one-level MCTF, Figure 4.5 to the two-level MCTF and Figure 4.6 to the three-level MCTF.

Figure 4.4: Frame that is reconstructed by complete error concealment (left) and by zero-motion error concealment (right) in one-level MCTF

Figure 4.5: Frame that is reconstructed by complete error concealment (left) and by zero-motion error concealment (right) in two-level MCTF

Figure 4.6: Frame that is reconstructed by complete error concealment (left) and by zero-motion error concealment (right) in three-level MCTF

The figures above illustrate that the proposed error concealment system outperforms zero-motion error concealment in terms of visual quality of the reconstructed frames. The frames

produces by the proposed system look clean, crisp, and fairly natural. On the other hand, the frames produced by zero-motion concealment suffer from the "ghost" effects due to temporal averaging. Similar results were observed for other sequences such as *Foreman, Mobile,* and *Stefan*.

4.2 Test Results of Case 3 of Chapter 2

Case 3 does not make assumptions $H_t(\vec{n}) \approx 0$ and $A_t(\vec{n}) \approx B_t(\vec{n} + \vec{df}_t)$, as discussed in Section 2.3. Since no assumptions are made, Case 3 should produce the most accurate error concealment equations. However, this improvement in accuracy makes the equations very complicated. The final equation for Case 3, which was derived in Section 2.3 for bi-directionally connected pixels, is repeated below.

$$L_2(\vec{n}) = \frac{1}{2}\left[L_1(\vec{n} - \overrightarrow{df_{11}}) + L_3(\vec{n} - \overrightarrow{dr_{11}})\right] - \frac{9}{8}\left[H_2(\vec{n} - \overrightarrow{df_2}) + H_3(\vec{n} - \overrightarrow{dr_2})\right]$$

$$+ \frac{1}{8}\left[H_1(\vec{n} - \overrightarrow{df_{11}} - \overrightarrow{df_1}) + H_4(\vec{n} - \overrightarrow{dr_{11}} - \overrightarrow{dr_3})\right] + \left[C_1(\vec{n}) + C_2(\vec{n})\right] \qquad (2.26)$$

Part of the complexity of equation (2.26) is due to the extended temporal connection from $\overrightarrow{df_2}$ to $\overrightarrow{dr_1}$ to $\overrightarrow{df_1}$. Please refer to the following picture for detailed information.

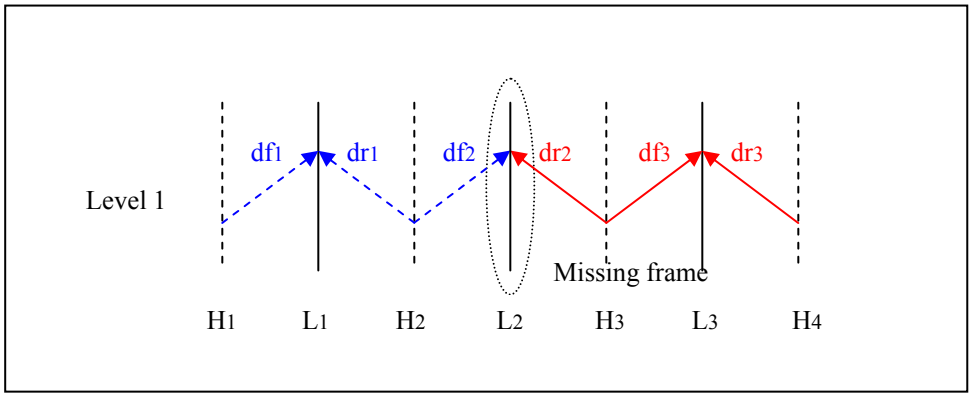


Figure 4.7: Illustration of complexity of Case 3

According to equation (2.26), in order to reconstruct the L_2 frame, we need at least one or both of the following two temporal connections; (1) $\overrightarrow{df_2}$ → $\overrightarrow{dr_1}$ → $\overrightarrow{df_1}$ and (2) $\overrightarrow{dr_2}$ → $\overrightarrow{df_3}$ → $\overrightarrow{dr_3}$. Let us take the first temporal connection as an example. This three temporal distance-long connection implies that we need to find a pixel in L_2 frame which we can find in all of the following frames: H_2, L_1, and H_1. Due to the unconnected pixel problem, the chance of finding such a pixel is much less compare to Case 1 or Case 2, where it only needs a connection to the L_1 frame. We can solve this problem of complexity of Case 3 by simplifying the equation (2.26). On the right-hand side of equation (2.26), the weighting for L_1 and L_3 is $\frac{1}{2}$, the weighting for H_2 and H_3 is $-\frac{9}{8}$, and for H_1 and H_4 it is $\frac{1}{8}$. Interestingly, when we sum up the weighting for all *H*-frames, we get $2\times(-\frac{9}{8}+\frac{1}{8})=-2$ and the total weighting for all *L*-frames is $2\times\frac{1}{2}=1$. In the example in Chapter 2, we showed that

for the *Coastguard* sequence, the average mean absolute value of the *H*-frames is approximately to 2.8, compared to the maximum pixel value for Y-component of 255. If we divide H_1 and H_4 frames by 8, as suggested by (2.26), their mean absolute value will be approximately 0.35 (at least for the *Coastguard* sequence), which is small enough to be approximated to zero. We can conserve the total weighting for all *H*-frames, which is –2, by changing the weighting for H_2 and H_3 from $-\frac{9}{8}$ to -1. As a result, equation (2.26) can be simplified as below.

$$L_2(\vec{n}) \approx \frac{1}{2}\left[L_1(\vec{n}-\overrightarrow{df_{11}}) + L_3(\vec{n}-\overrightarrow{dr_{11}})\right] - \left[H_2(\vec{n}-\overrightarrow{df_2}) + H_3(\vec{n}-\overrightarrow{dr_2})\right] + \left[C_1(\vec{n}) + C_2(\vec{n})\right] \tag{4.1}$$

In equation (4.1), to reconstruct L_2, we only need to access frames that are up to two temporal distances away from L_2, which reduces the number of unconnected pixels compared to (2.26). In other words, using equation (4.1) will allow us to cover more pixels with motion concatenation and fewer pixels with zero-motion error concealment. We have tested the error concealment system using equation (4.1). These simulation results are compared with the results of Case 1 and Case 2, which were presented in Section 4.1. The order of operations is the same as the one used before: motion concatenation, followed by motion re-estimation, followed by zero-motion error concealment. In our current implementation, we do not estimate the values of $C_i(\vec{n})$, but rather set them to zero, which effectively makes them disappear from (4.1).

4.2.1 Y-components PSNR Results

Horizontal axes in the graphs below show the index of the *L*-frame where error concealment is performed and the vertical axes represent Y-component PSNR in dB of the *B*-frame directly above the concealed *L*-frame.

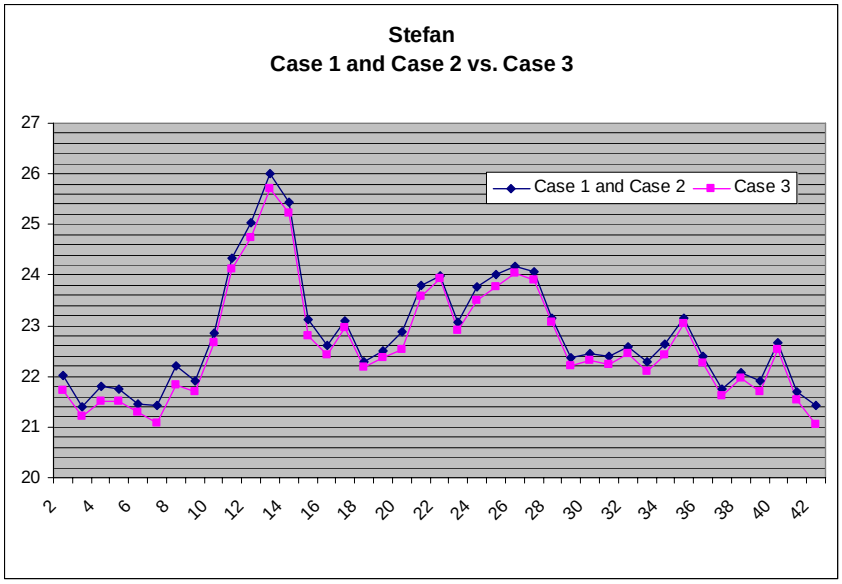


Figure 4.8: Y-component PSNR graph of Case 1 and 2 error concealment system versus Case 3 error concealment for one-level MCTF

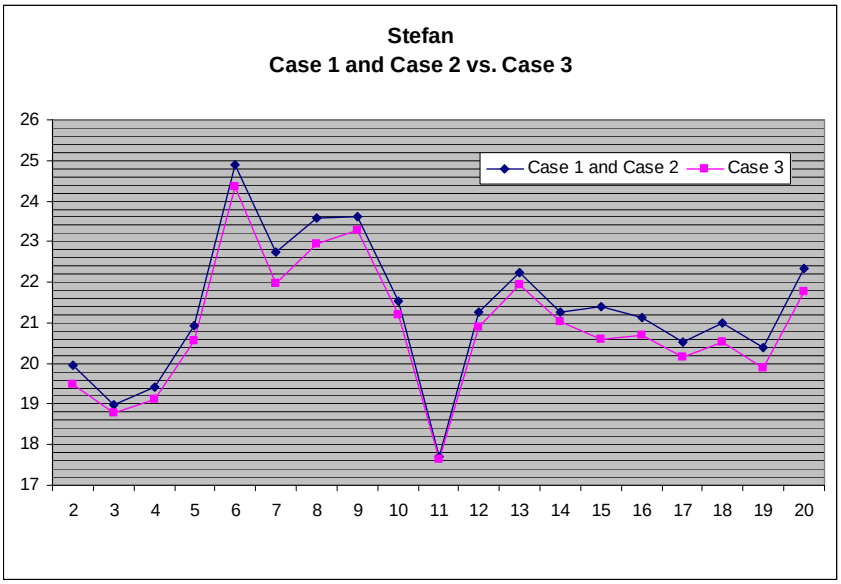


Figure 4.9: Y-component PSNR graph of Case 1 and 2 error concealment system versus Case 3 error concealment for two-level MCTF

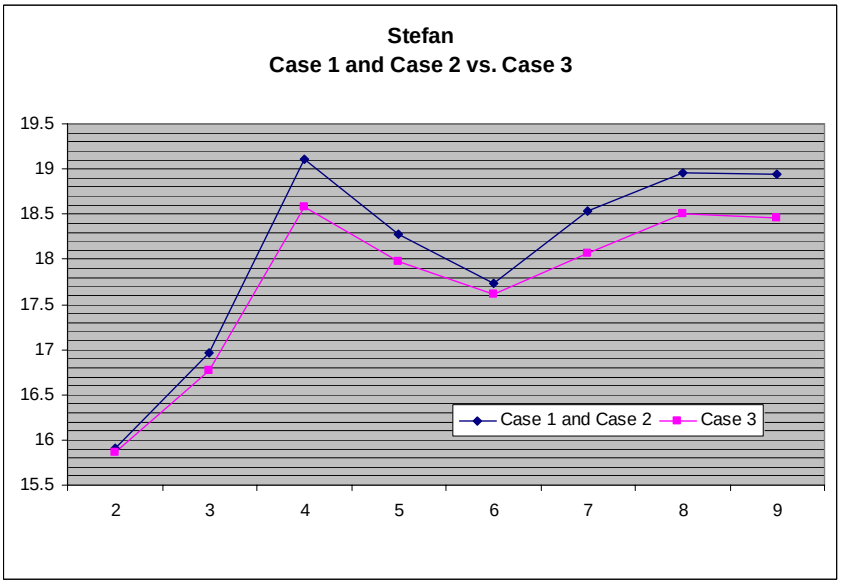


Figure 4.10: Y-component PSNR graph of Case 1 and 2 error concealment system versus Case 3 error concealment for three-level MCTF

4.2.2 Visual Quality Results

The visual comparison between Case 1 and Case 2 versus Case 3 is shown below.

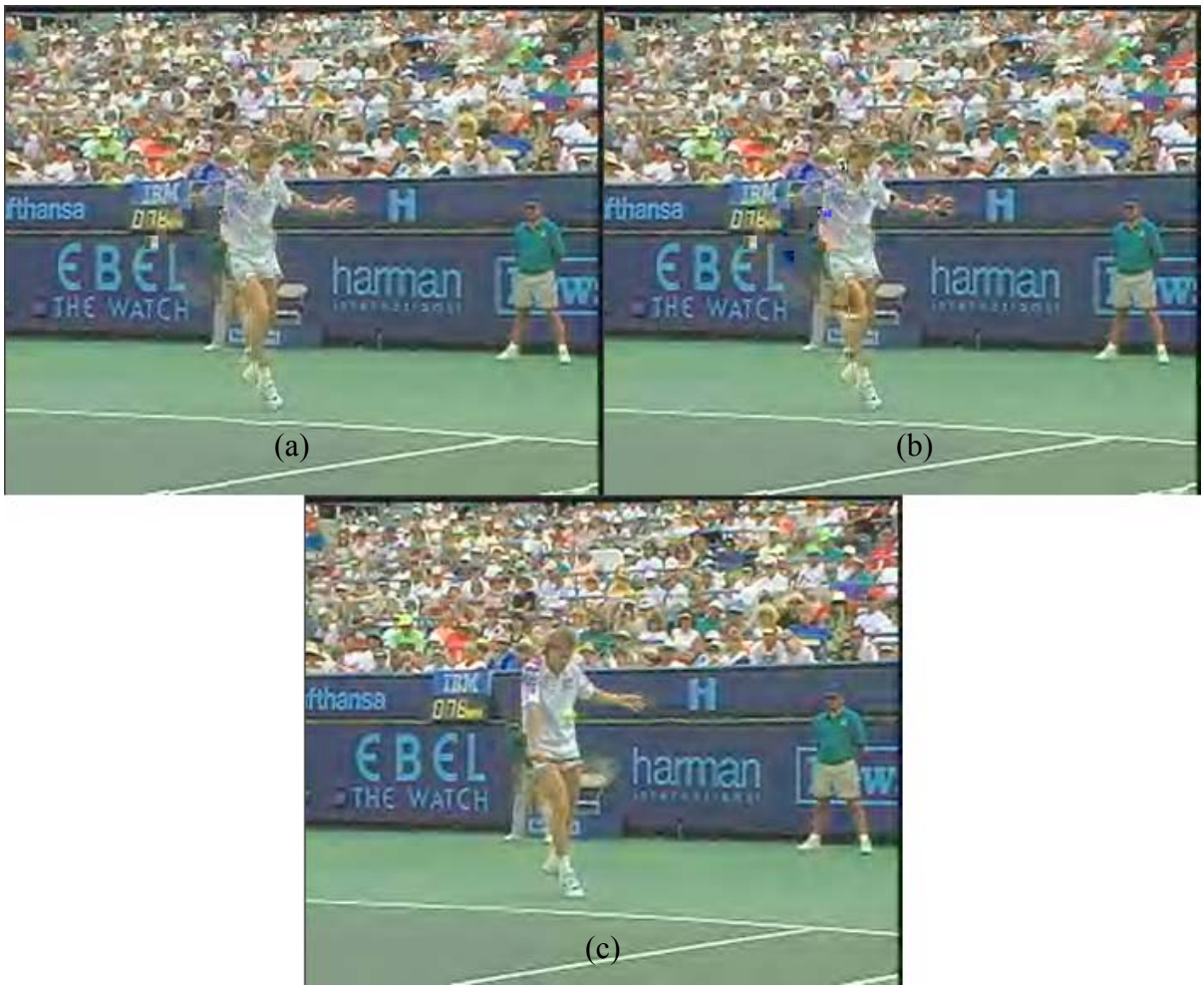


Figure 4.11: (a) Frame reconstructed using equation of Case 1 and Case 2

(b) Frame reconstructed using equation of Case 3

(c) Coded loss-free frame (*Stefan,* one-level MCTF)

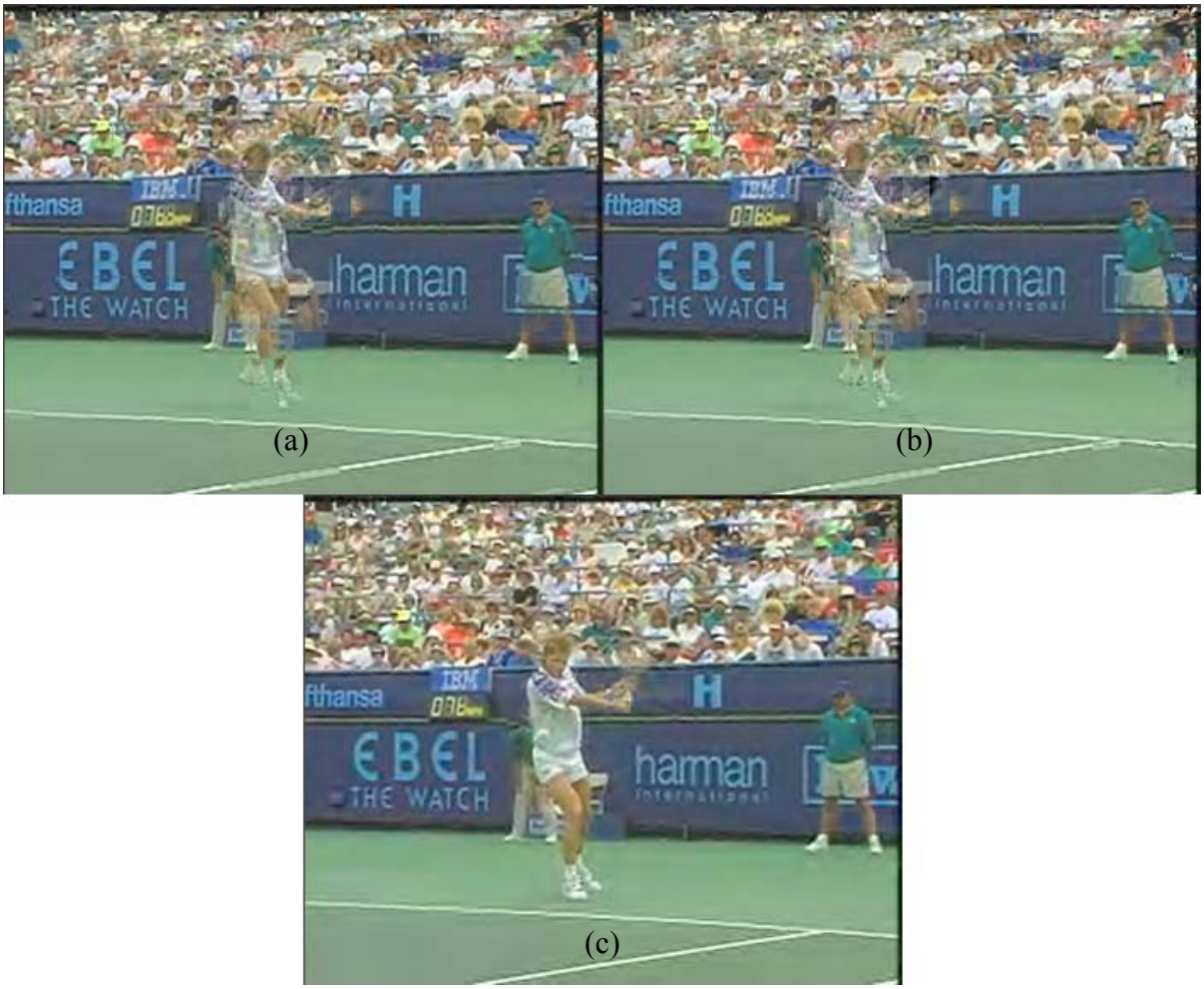


Figure 4.12: (a) Frame reconstructed using equation of Case 1 and Case 2

(b) Frame reconstructed using equation of Case 3

(c) Coded loss-free frame (*Stefan*, two-level MCTF)

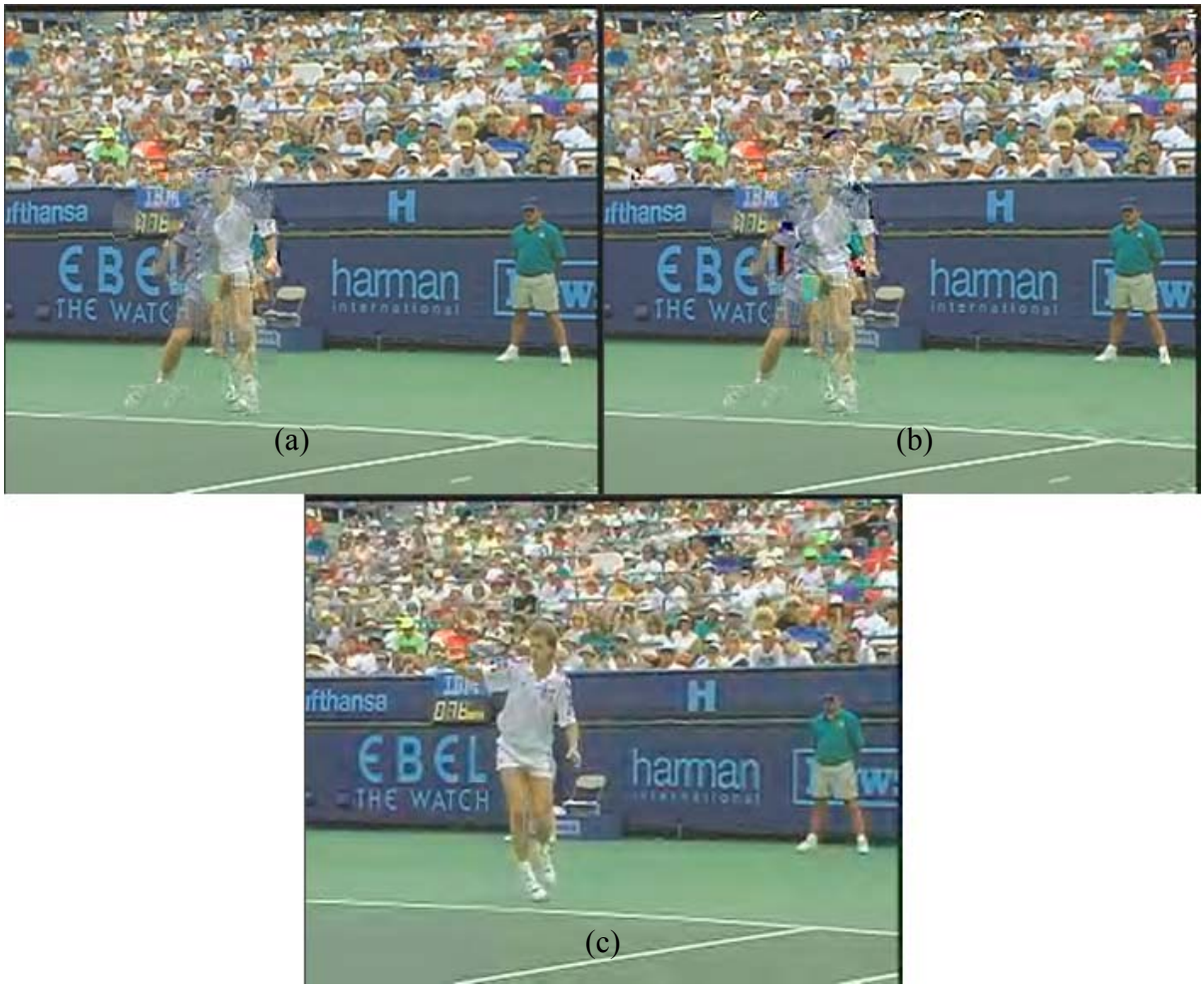


Figure 4.13: (a) Frame reconstructed using equation of Case 1 and Case 2

(b) Frame reconstructed using equation of Case 3

(c) Coded loss-free frame (*Stefan*, three-level MCTF)

The results in Y-component PSNR and captured frames tell us that equation (2.11) or/and (2.15) performs better than (4.1). Figure 4.11, Figure 4.12 and Figure 4.13 show the visual results of the *B*-frames just above the concealed *L*-frames for each case. It appears that the inclusion of *H*-frames has added some noise to the reconstructed frame. This

result was surprising because equation (4.1) was believed to be more accurate than (2.11) or (2.15). Possible explanations for this phenomenon might be the changing of *H*-frame weights in (4.1) compared to (2.26), and the fact that the random fields are ignored in the implementation.

4.3 The Order of Performing Motion Concatenation and Motion Re-estimation

In Chapter 3, we discussed the practical methods to estimate the motion between the missing *L*-frame and its neighboring *L*-frames. The two methods that rely on motion compensation are motion concatenation and motion re-estimation. We can combine these two methods to create a complete error concealment system. In Section 3.4, we explained the importance of the order of applying these two mechanisms. We expected that applying motion concatenation followed by motion re-estimation would provide better accuracy compared to the alternative order of motion re-estimation followed by motion concatenation. In this section, we test which order of operations performs better by analyzing the PSNR results as well as visual quality. To this end, we test the performance of two complete error concealment systems, each employing one of these two orders of operations.

Order 1: Motion Concatenation → Motion Re-estimation → Zero-motion Error Concealment,

Order 2: Motion Re-estimation → Motion Concatenation → Zero-motion Error Concealment.

The tests were performed in the same manner as before, using concealment equations for Case 1 and Case 2.

4.3.1 Y-component PSNR Results

Figure 4.14, Figure 4.15 and Figure 4.16 show Y-component PSNR curves for Order 1 and Order 2 for the *Coastguard* sequence. As before, the horizontal and vertical axes represent the index of the *L*-frame where error concealment is performed, and Y-component PSNR in dB of the *B*-frame directly above the concealed *L*-frame, respectively.

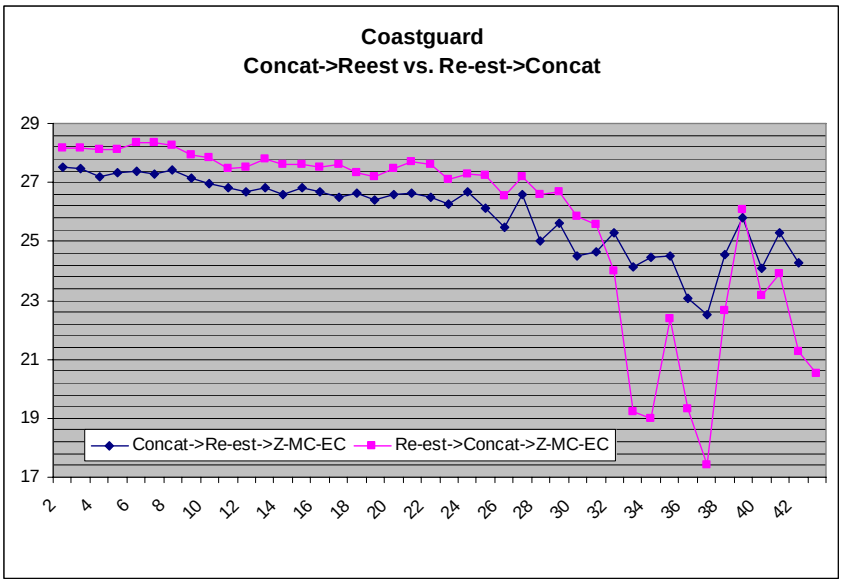


Figure 4.14: Y-component PSNR graph of Order 1 versus Order 2 for one-level MCTF (*Coastguard* sequence)

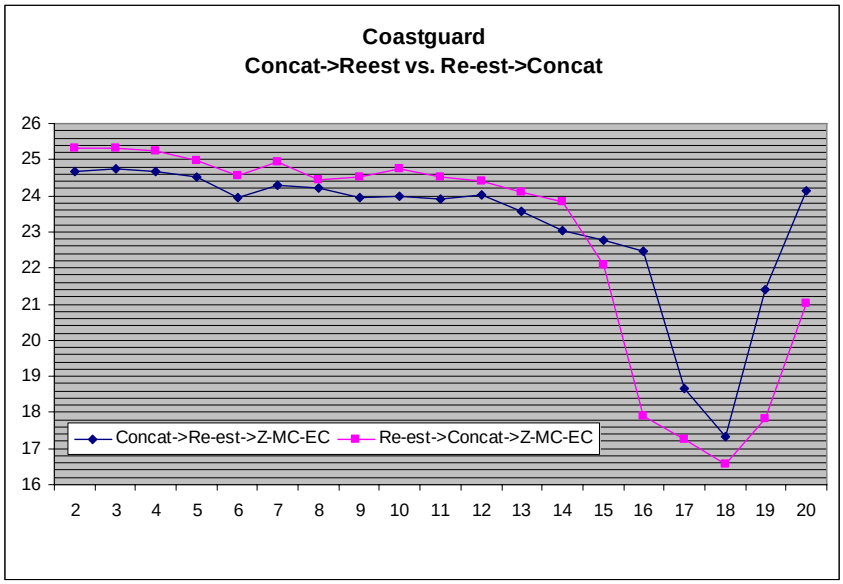


Figure 4.15: Y-component PSNR graph of Order 1 versus Order 2 for two-level MCTF (*Coastguard* sequence)

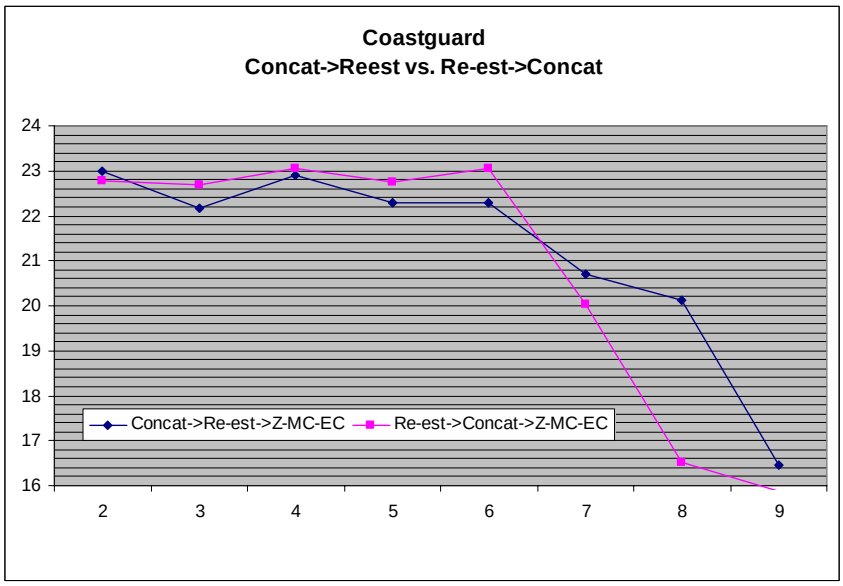


Figure 4.16: Y-component PSNR graph of Order 1 versus Order 2 for three-level MCTF (*Coastguard* sequence)

PSNR curves for the *Coastguard* sequence tell us that Order 2 performs better than Order 1 for scenes with relatively slow and constant movement, while Order 1 performs better for scenes with faster and more complex motion. Similar results were obtained for the *Mobile Calendar* sequence. This is because the scenes with relatively constant motion have less chance of having a problem that was illustrated in Figure 3.5. Another important aspect of PSNR curves for the *Coastguard* sequence is that Order 1 starts to perform better than Order 2 from GOP = 33. Interestingly, objects' movement in *Coastguard* sequence becomes relatively fast from that point. Following three graphs show PSNR results for Order 1 and Order 2 for the *Foreman* sequence, which is characterized by more intense and complex motion.

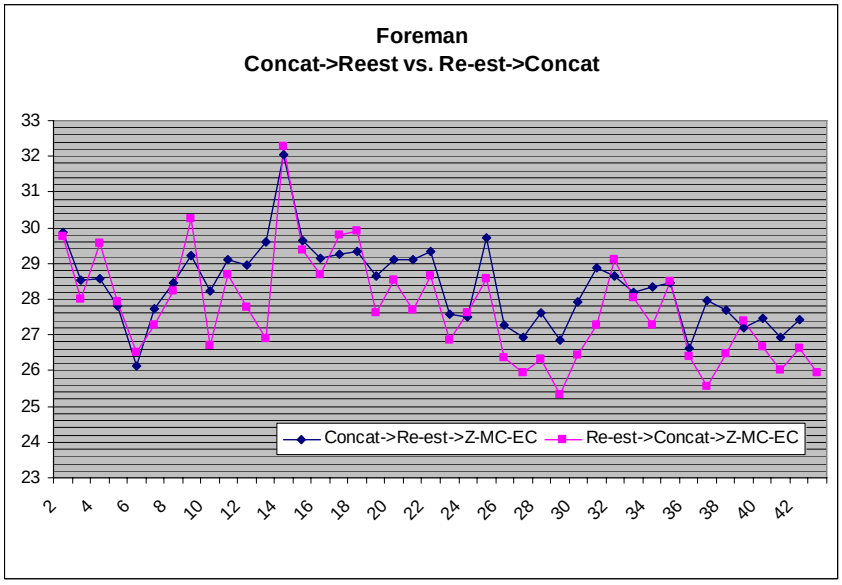


Figure 4.17: Y-component PSNR graph of Order 1 versus Order 2 for one-level MCTF (*Foreman* sequence)

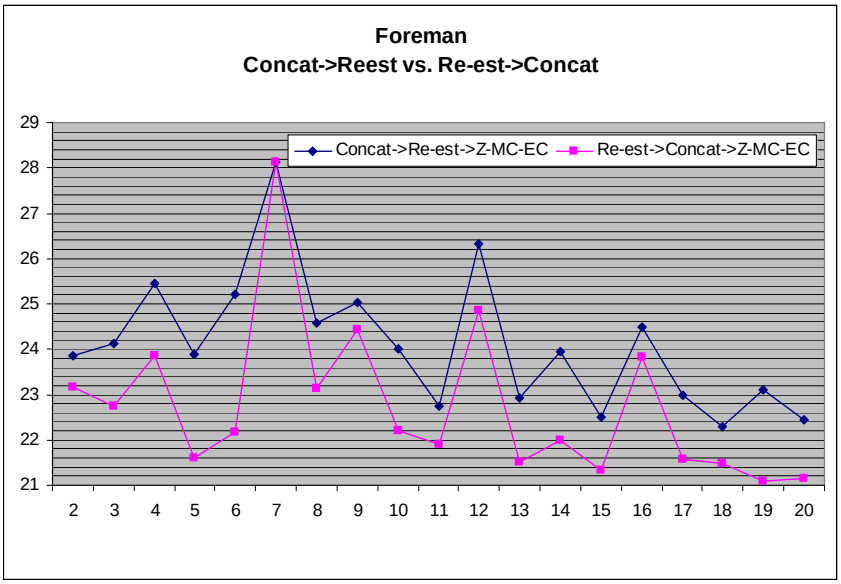


Figure 4.18: Y-component PSNR graph of Order 1 versus Order 2 for one-level MCTF (*Foreman* sequence)

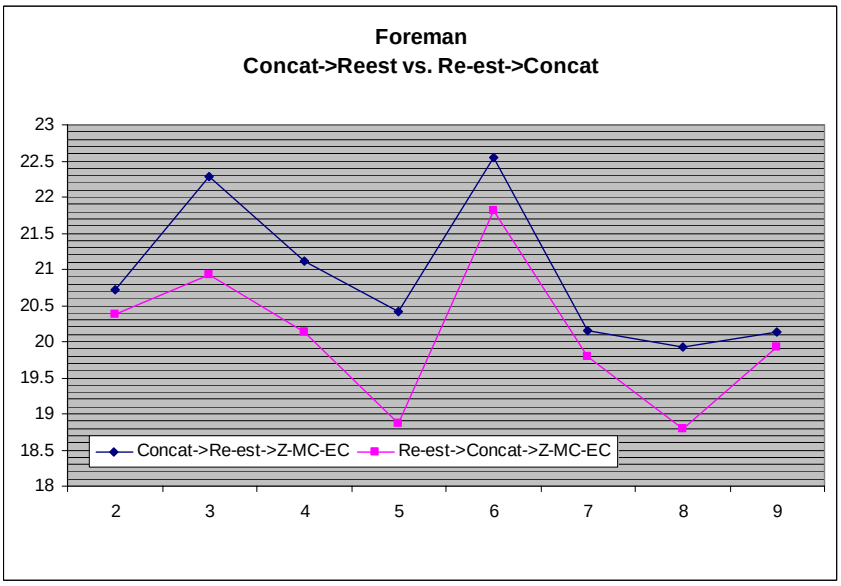


Figure 4.19: Y-component PSNR graph of Order 1 versus Order 2 for one-level MCTF (*Foreman* sequence)

Here, we can observe that Order 1 produces higher PSNR values than Order 2 in the above three graphs, unlike the result for the *Coastguard* sequence. Due to more complex motion in the *Foreman* sequence, the problem of Figure 3.5 starts to occur more often. This makes it difficult for motion re-estimation to guess the correct position of pixels due to dynamic movement of objects. We also find that the PSNR difference between Order 1 and Order 2 increases as the number of levels in MCTF increases. For higher levels, temporal distance between the neighboring *L*-frames is greater and it is more difficult for motion re-estimation to successfully guess the position of pixel value in the missing *L*-frame. We found that *Stefan* sequence gives similar results to the *Foreman* sequence.

4.3.2 Visual Quality Results

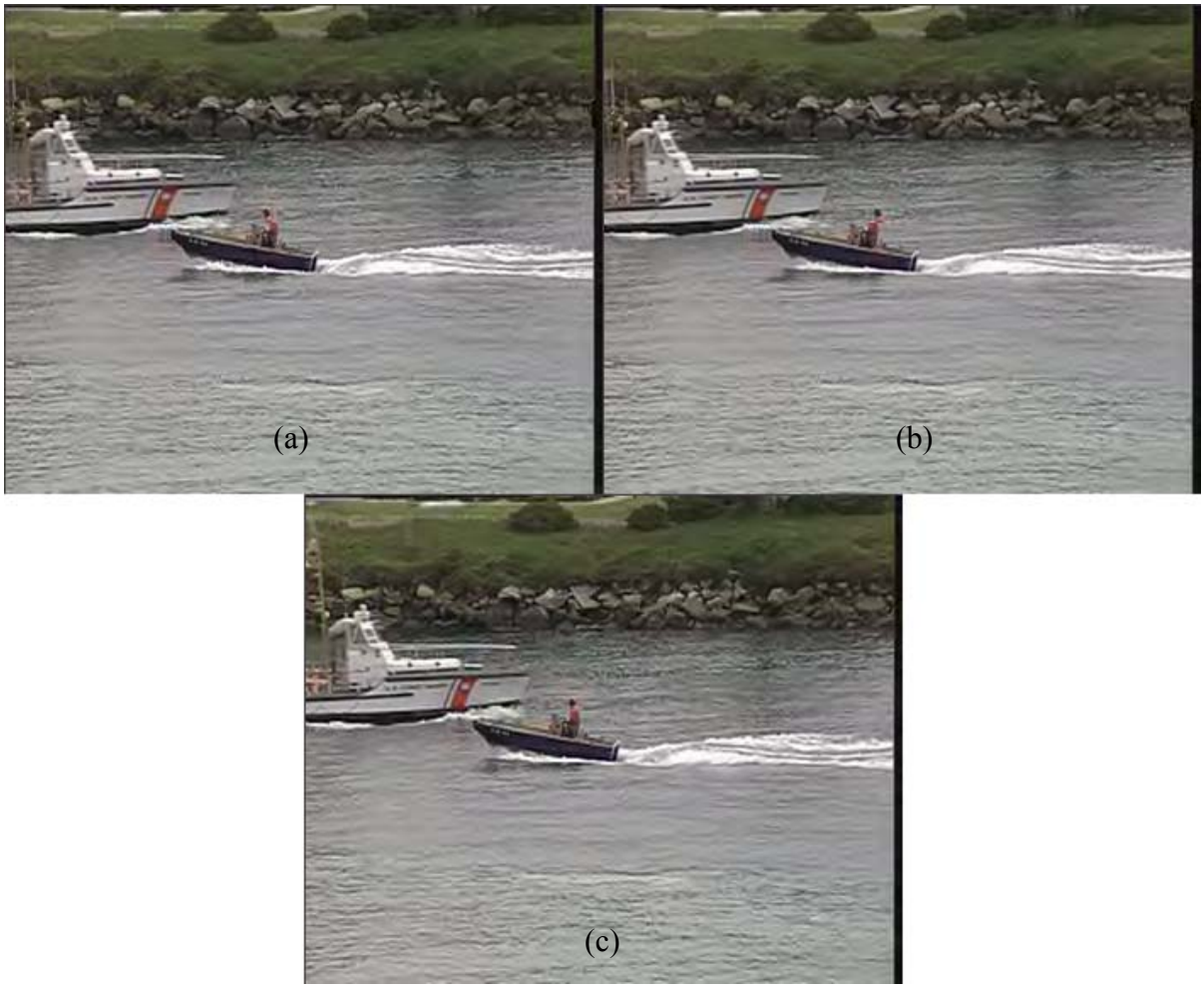


Figure 4.20: (a) Frame reconstructed by Order 1

(b) Frame reconstructed by Order 2

(c) coded loss-free frame (one-level MCTF, *Coastguard*)

Figure 4.21: (a) Frame reconstructed by Order 1

(b) Frame reconstructed by Order 2

(c) coded loss-free frame (two-level MCTF, *Coastguard*)

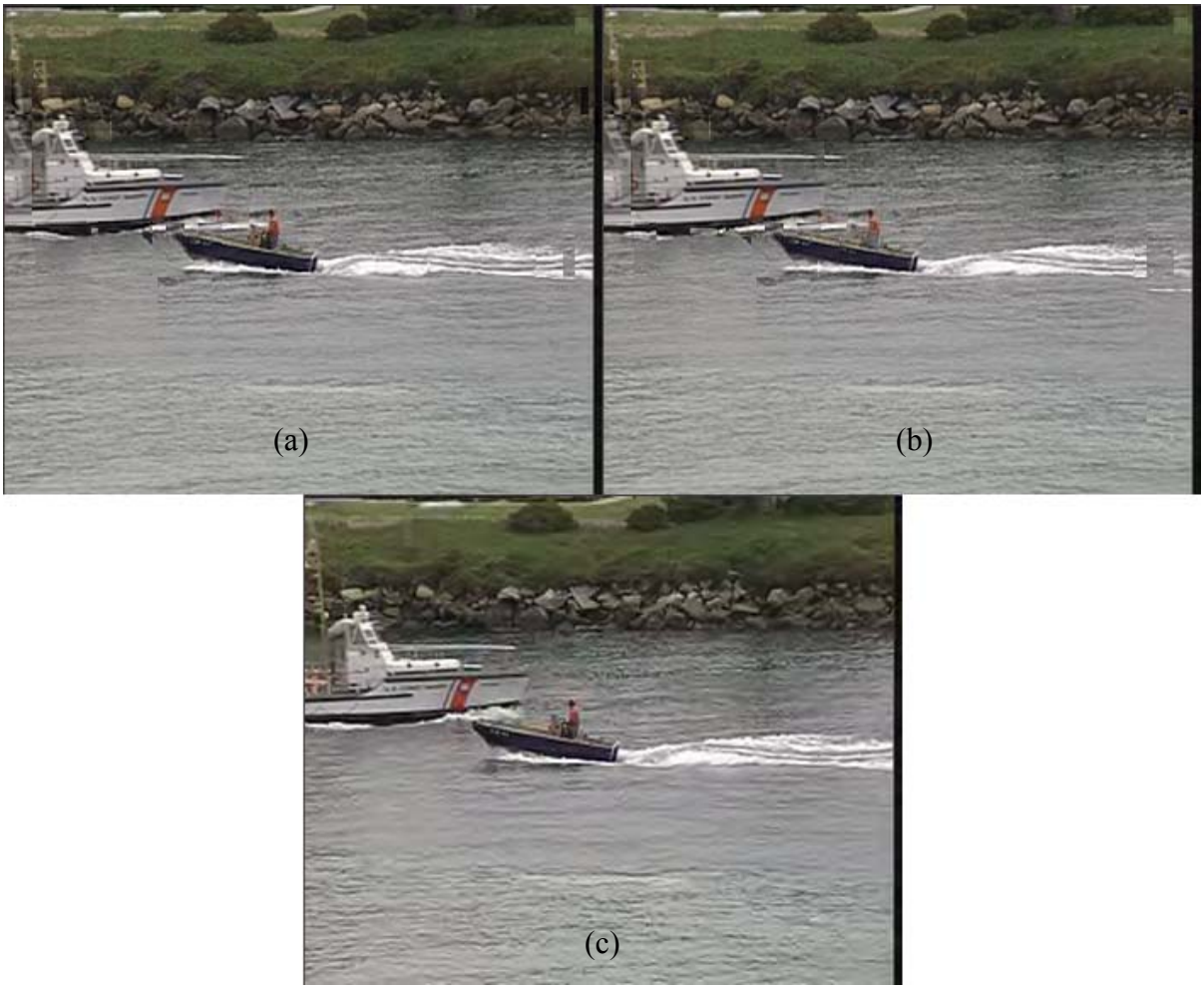


Figure 4.22: (a) Frame reconstructed by Order 1

(b) Frame reconstructed by Order 2

(c) coded loss-free frame (three-level MCTF, *Coastguard*)

Figure 4.20,

Figure 4.21 and Figure 4.22 show the *B*-frames directly above the concealed *L*-frames of the *Coastguard* sequence for one-, two-, and three-level MCTF, respectively. In all three examples above, we could not find considerable differences between the frames produced by

Order 1 and Order 2. This is probably due to the relatively constant movement of objects in the *Coastguard* sequence. However, Order 1 may produce more accurate details in the reconstructed frames, such as the position of the sailor's head in the small boat. Next, we display visual results for the *Foreman* sequence with more complex motion.

Figure 4.23: (a) Frame reconstructed by Order 1

(b) Frame reconstructed by Order 2

(c) coded loss-free frame (one-level MCTF, *Foreman*)

Figure 4.24: (a) Frame reconstructed by Order 1

(b) Frame reconstructed by Order 2

(c) coded loss-free frame (two-level MCTF, *Foreman*)

Figure 4.25: (a) Frame reconstructed by Order 1

(b) Frame reconstructed by Order 2

(c) coded loss-free frame (three-level MCTF, *Foreman*)

Here, we could observe distinct differences between the frames produced by Order 1 and Order 2. The frames produced by Order 2 seem to be more noisy and pixels are off the position. The difference in quality becomes larger as the number of MCTF levels increases.

For example, in Figure 4.25, the face and the background in the frame produced by Order 2 shows more distortion compare to the frame produced by Order 1.

4.4 Removal of Line Distortion

In Figure 3.2, we briefly introduced *line distortion* that can be observed in the frames produced by motion concatenation. This distortion manifests itself as one or two pixels-wide "lines" which could not be covered by motion concatenation. These lines must be filled up using other methods such as motion re-estimation and zero-motion error concealment, and slight difference in pixel values might create visual distortion. We briefly introduced a solution which minimizes the effect of *line distortion*. We blur the pixel boundary of two different mechanisms used by averaging pixel values around the boundaries. The idea of blurring effect could be understood by the following diagram.

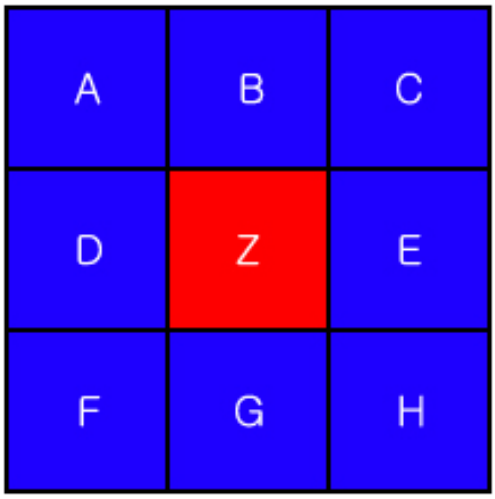


Figure 4.26: Blurring effect to minimize the line distortion

Suppose we would like to blur pixel Z in Figure 4.26. Then its blurred pixel value can be constructed using the following equation,

$$Z = \frac{A + B + C + D + E + F + G + H}{8}.$$

We perform the above equation for all pixels near the boundary of two the blocks of pixels produced by two different mechanisms (e.g., motion concatenation and zero-motion concealment). If some of the neighboring pixels of the Z pixel are not available (e.g., if Z is on the edge of the frame), then we only take those available pixels into the equation.

As a result of the blurring effect, Y-component PSNR values did not show a significant change. However, the blurring effect improved the visual quality significantly. The following figures show the frames after the blurring effect. They are produced by Order 1 with Case 1/Case 2 equations.

Figure 4.27: (a) Frame reconstructed after blurring effect

(b) Frame reconstructed before blurring effect

(c) coded loss-free frame (one-level MCTF, *Foreman*)

4.5 Time Complexity and Practical Issues

We have so far discussed about performance of the error concealment system in Y-component PSNR and visual quality. Another standard for evaluating the performance would

be time complexity. In this section, we compare time complexity of the two methods; motion concatenation and motion re-estimation. Method we used to compare time complexity of the two methods is as follows. First, we create two independent error concealment systems with different sequences of mechanisms.

Test Bench 1: Motion Concatenation → Zero MC Error Concealment

Test Bench 2: Motion Re-estimation → Zero MC Error Concealment

Then, we calculate the amount of time required to complete the concealment for each test bench systems. In theory, motion re-estimation would take a lot more time to complete its computation compare to motion concatenation because motion re-estimation involves motion vector estimation. The test was performed on four different sequences for four different temporal levels (*Coast Guard, Foreman, Mobile Calendar,* and *Stefan.*) We measure time duration of the error concealment algorithm for every GOP. Table 4.1 below shows the averaged results of all tested sequences.

	Coast Guard	*Foreman*	*Mobile Calendar*	*Stefan*	Overall
Motion Concatenation	0.44025 sec	0.43575 sec	0.454263 sec	0.438538 sec	0.434123 sec
Motion Re-estimation	50.52849 sec	77.49326 sec	40.30155 sec	80.85935 sec	62.79541 sec

Table 4.1: Test result of time complexity of different sequences

According to Table 4.1, averaged time duration for motion concatenation is 0.43 seconds and for motion re-estimation is 62.7 seconds. As it was expected, the motion re-estimation performs much slower than motion concatenation. Averaged time duration of decoding process without error concealment for one GOP is approximately equal to averaged time

duration of motion concatenation. Motion re-estimation would not be appropriate for practical error concealment system since it take unrealistic time duration to complete its task. This implies that we must construct an error concealment system with motion concatenation and zero MC error concealment for practical usage. However, Y-component PSNR and visual quality results would be degraded by eliminating motion re-estimation from the error concealment process. If the degradation in PSNR and visual performance is not significant, we can compensated it for reduction in time duration. We compare PSNR result of *Test Bench 1* with "Motion Concatenation → Motion Re-estimation → zero MC Error Concealment" sequence.

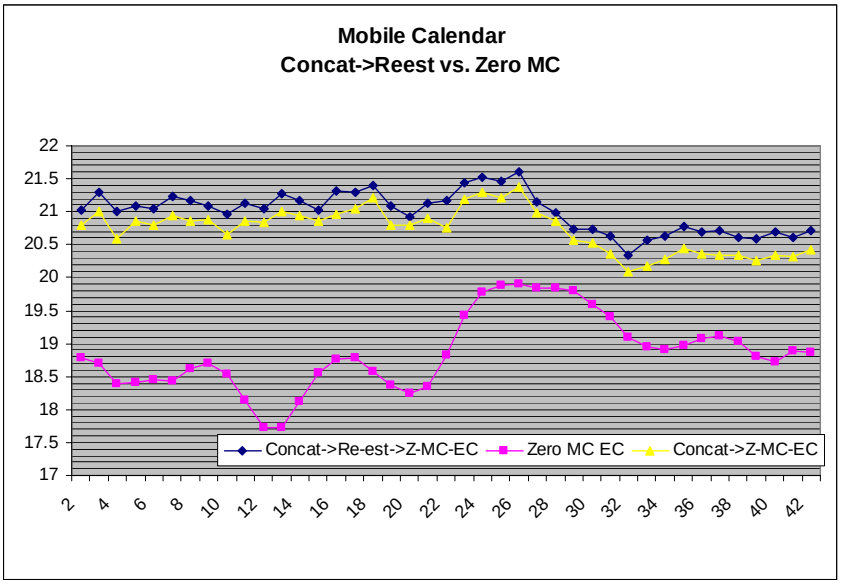


Figure 4.28: Y-component PSNR graph of *Test Bench 1*

We compared Y-component PSNR of three different methods; 1) "Motion Concatenation → Motion Re-estimation → Zero MC Error Concealment" 2) *Test Bench 1* and 3) Benchmark zero MC error concealment. According to Figure 4.28, we can conclude that eliminating

motion re-estimation from the process would result in degradation in Y-component PSNR but it is insignificant. Let us now compare results of visual quality of the three different error concealment systems showed in Figure 4.28.

Figure 4.29: (a) Frame reconstructed by *Test Bench 1*

(b) Frame reconstructed by "Motion Concatenation → Motion Re-estimation → Zero MC Error Concealment"

(c) Frame reconstructed by benchmark zero MC error concealment

(d) coded lossless frame

Visual quality experiments showed similar results. *Test Bench 1* performs poorer compare to "Motion Concatenation → Motion Re-estimation → Zero MC Error Concealment," but the quality is reasonably acceptable compare to benchmark zero MC system.

We presented performance comparison of time complexity, Y-component PSNR and visual quality for two different error concealment systems; one with motion re-estimation and the other without motion re-estimation. Time required to complete the error concealment task was reduced dramatically by eliminating motion re-estimation from the process. On the other hand, PSNR and visual quality resulted in acceptable level. Therefore we can conclude that motion concatenation is more suitable for practical error concealment system.

Chapter 5

Conclusions and Future Research

In this thesis, we introduced error concealment algorithms for 5/3 motion compensated temporal filter with lifting. We introduced three different sets of concealment equations, Case 1, Case 2 and Case 3, each valid for a certain set of assumptions. We tested the three sets of equations and concluded that Case 1 and Case 2, which share the same set of equations in our current implementation, perform better than the equations for Case 3. We also introduced three practical mechanisms estimate the motion between the missing *L*-frame and its neighboring *L*-frames; motion concatenation, motion re-estimation and zero-motion error concealment. We discussed the importance of the sequence of operation of these three mechanisms and demonstrated that the sequence of motion concatenation and motion re-estimation followed by zero-motion error concealment performs the best. Therefore, the best error concealment system should:

1. Not use *H*-frames to reconstruct the missing *L*-frame.
2. Use the following sequence of operations: motion concatenation → motion re-estimation → zero-motion error concealment.
3. Use the blurring effect on the block boundaries.

The proposed motion-compensated error concealment system built on the above three principles showed an average advantage of 2.75 dB in PSNR (and occasionally up to 7 dB) compared to the benchmark zero-motion concealment system.

For our future research, we plan to investigate other methods to combat the problem of unconnected pixels and “holes” produced by motion concatenation and motion re-estimation. We believe that motion re-estimation using boundary matching [5] would be able to improve our error concealment system.

Appendix A

Test Results for $|H|_{avg}$ and $MAD_{A,B}$

The following table shows the of mean absolute values of *H*-frames ($|H|_{avg}$) and mean absolute difference between *A*- and *B*-frames ($MAD_{A,B}$) on the *Coastguard* sequence for the one-level MCTF using Y-component values. The first column of represents GOP (Group Of Pictures) index where the corresponding *H*-, *A*- and *B*-frames belong. The second column represents averaged $MAD_{A,B}$ of that GOP. We averaged all temporally connected pixels on a given pair of frames. The third column contains the maximum $MAD_{A,B}$ value. The fourth column, *N*, represents number of temporally connected pixels. Fifth column contains results of the equation below.

$$s^2 \times (N-1) = \sum (X - M)^2$$

Where *X* is $MAD_{A,B}$ value for each pixel and *M* is the global mean value, which is equal to 4.5. These values will be used to calculate standard deviation for $MAD_{A,B}$ and $|H|$. The sixth and seventh columns represent averaged $|H|$ and maximum $|H|$ values of each GOP, respectively. The eighth column, *N*, contains number of temporally connected pixels of *H*-frames of each GOP. The ninth column represents $s^2 \times (N-1)$ value for *H*-frames. We used the global mean value of 2.3 in this case. We use values in fifth and ninth colonms to calculate standard deviation value for $MAD_{A,B}$ and $|H|$, which are 6.86 and 3.77 respectively.

GOP	$MAD_{A,B}$	$maxMAD_{A,B}$	N	$s^2 \times (N-1)$	$\|H\|_{avg}$	$\|H\|_{max}$	N	$s^2 \times (N-1)$

2	3.83567	124.017	101376	885695.9	1.94921	59.8933	101376	3371806.75
3	4.53429	117.602	95744	1187821	2.26227	96.4505	101376	4293540
4	4.18216	113.929	98240	976514.6	2.09297	58.0915	101376	3718805.25
5	3.7301	122.097	101312	969466.9	2.03705	92.5798	101376	3350085.75
6	3.76336	109	101376	889957.9	1.96385	76.4169	101376	3419700.75
7	4.06783	104.732	100288	948005	1.99514	61.1758	101376	3771965
8	3.87686	113.866	101376	938301.3	1.96523	57.3349	101376	3612510
9	3.82825	101.676	101312	966304.3	1.99407	62.2141	101376	3581641.5
10	3.91853	109.79	101376	970047.1	2.01451	62.0405	101376	3593405.75
11	3.81497	99.0974	101376	1053667	2.1366	69.5189	101376	3478845
12	4.01934	96	99904	996140.4	2.0374	57.92	101312	3826887.75
13	3.99957	113	100352	1012561	2.04201	61.231	101376	3894877.75
14	4.0007	119	99520	1034684	2.06483	65.2799	101376	3776482.5
15	3.99652	147.309	101312	1134855	2.12683	59.2661	101376	3893938
16	4.31148	122.42	100288	1284141	2.22532	67.8437	101376	4667716
17	4.17675	108.019	100160	1240897	2.25589	55.9542	101312	4139423
18	4.19973	119.37	100256	1169625	2.17062	61.0359	101376	4237860.5
19	4.11241	103.527	101312	1184811	2.19017	57.6498	101376	4222726
20	4.17788	165.598	101056	1155492	2.14647	75.9256	101376	4276461
21	4.44038	125.173	101376	1156176	2.15702	73.315	101376	4614105.5
22	4.16857	165.661	101312	1402269	2.35939	72.7082	101376	4484496
23	4.20659	146.428	101312	1190731	2.14541	67.4749	101376	4371840
24	4.74629	158.81	85836	1284228	2.27421	78.8446	101312	5165467.5
25	4.12299	132.297	101376	1466945	2.49397	86.6377	101376	4207175
26	4.13516	136.301	101360	1184425	2.15413	84.3552	101376	4260744.5
27	4.60772	119.886	101120	1334956	2.25171	101.735	101376	5035996
28	4.6227	160.962	101312	1560265	2.50466	126.321	101376	5333560.5
29	4.87451	137.748	94912	1446953	2.41505	127.169	101376	5012012
30	5.50213	106.491	101312	1570514	2.56831	58.5623	101376	6929827.5
31	4.47194	140.262	101312	1669952	2.58044	118.912	101376	5003563
32	6.49669	123.313	101312	2110109	3.09639	73.4156	101376	9210992
33	6.90617	127.738	96832	3656328	4.18886	98.9203	100736	9970264
34	4.9222	137.64	96112	2672672	3.38063	100.609	101056	5873696.5
35	5.26535	121.994	91296	2140945	3.03638	75.0849	101056	5706917
36	5.98325	133.597	97824	1355328	2.19343	113.263	99840	4007946.5
37	4.09823	131.563	80384	1659263	2.42997	111.538	101376	3666527.75
38	5.9653	160.626	91488	2175327	2.81845	136.177	99968	7878801.5
39	4.85728	156.808	86784	1516252	2.42655	79.0198	101376	4969671
40	5.68911	140.072	70144	2411268	3.20783	63.7486	101376	5363701.5
41	3.70971	186	101216	1526970	2.41073	85.2788	101376	3892697.5
42	3.69966	143.092	96936	1722767	2.62529	72.3141	101376	3598706.75
43	5.06147	162	95424	1350375	2.29231	76.2205	101376	6050984
44	5.66312	133.202	94880	1456926	2.40999	62.6138	101376	7339351.5
45	4.25286	91.7173	95552	1809451	2.75675	83.9313	101376	4517225.5

Table 5.1: Test results for $H|_{avg}$ and $MAD_{A,B}$ of *Coastguard* sequence

References

[1] J. W. Woods, *Multidimensional signal, image, and video processing and coding*, Elsevier - Academic Press, 2006.

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

[3] G. Pau and B. Pesquet-Popescu, "Uniform motion-compensated 5/3 filterbank for subband video coding," *Proc. IEEE ICIP'04*, vol. 5, pp. 3117 – 3120, Oct. 2004.

[4] J. R. Ohm, "Advances in scalable video coding," *Proc. IEEE*, vol. 93, no. 1, pp. 42-56, Jan. 2005.

[5] C.-Y. Chen, C.-T. Huang, Y.-H. Chen, S.-Y. Chien, and L.-G. Chen, "System analysis of VLSI architecture for 5/3 and 1/3 motion-compensated temporal filtering," *IEEE Trans. Signal Processing*, vol. 54, no. 10, pp. 4004 – 4014, Oct. 2006.

[6] P. Chen and J. W. Woods, "Bi-directional MC-EZBC with lifting implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1183–1194, Oct. 2004.

[7] Y. Wu, "Fully scalable subband/wavelet video coding system," Ph.D. dissertation, ECSE Dept., Rensselaer Polytechnic Institute, , Troy, NY, Aug. 2005.

[8] T. Rusert, K. Hanke, and M. Wien, "Optimization for locally adaptive MCTF based on 5/3 filtering," presented at the *Picture Coding Symposium*, San Francisco, CA, Dec. 2004.

[9] H.-C. Huang, W.-H. Peng, T. Chiang, and H.-M. Hang, "Advances in the scalable amendment of H.264/AVC," *IEEE Communications Magazine*, vol. 45, no. 1, pp. 68-76, Jan. 2007.

[10] I. V. Bajic and J. W. Woods, "Error concealment for scalable motion-compensated subband/wavelet video coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, pp. 508-514, Apr. 2007.

[11] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*, Upper Saddle River, NJ: Prentice-Hall, 2002.

[12] Y.-M. Chen and I. V. Bajic, "Predictive decoding for delay reduction in video communications," accepted for presentation at *IEEE Globecom'07*, Washington, DC, Nov. 2007.