COMPLEXITY SCALABLE AND ROBUST MOTION ESTIMATION FOR VIDEO COMPRESSION.

by

Hyukjune Chung

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Electrical Engineering)

March, 2007

Dedication

To my beloved family

Acknowledgments

First of all, I am beyond grateful to my advisor, Prof. Antonio Ortega for his guidance, inspiration, patience and advice during my doctoral research at the University of Southern California and I am thankful for the privilege of working with him.

I would like to thank Prof. C.-C. Jay Kuo, Prof. Sandeep K. Gupta, and Prof. Roger Zimmermann for serving on my dissertation committee, and Prof. Alexander A. Sawchuk for his guidance and advice during my first year of doctoral research and for serving on my Ph.D. qualification exam committee. I would like to thank Prof. Melvin A. Breuer for his advice and support during my research on the error tolerant computing. I also would like to thank Prof. Bart Kosko for the valuable teaching assistance experiences and for advice and discussions on various fields.

Thanks to all my friends and colleagues in the Compression Research Group, specially, Sang-Yong Lee, Krisda Lengwehasatit, Phoom Sagetong, Naveen Srinivasamurthy, Zhourong Miao, Kemal Demirciler, Hua Xie, Alexandre Ciancio, David Romacho, Baltasar Beferull-Lozano, Lavanya Vasudevan, Ivy Tseng, Huisheng Wang, Ngai-Man Cheung, Jae Hoon Kim, and In Suk Chong for sharing valuable time and ideas with me. I also would like to thank my friends at USC including

Jingyeong Kim, Hyungsuk Kim, Younggook Kim, Chulmin Lee, Janghoon Yang, Junsung Park, Wonseok Baek, Sunghoon Park, Sunkwang Hong, Nopparit Intharasombat, and Sanghyun Chang.

Special thanks to my grandparents, parents, parents-in-law, and my sister for their their love and continuing supports. And to my wife, Jungshim, for her love, support, and sacrifice. I am so lucky to have her in my life. Last but not least, I would like to express my deepest gratitude to God.

Contents

[1] This chapter is based upon work supported in part by the National Science Foundation under Grant No. 0428940. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

List of Tables

List of Figures

Abstract

Thanks to the fast development of network technology, transmission of high quality multimedia data becomes essential. However, the growth of data transferring capability is not always matched with the growth of bandwidth, specifically for wireless mobile environment. This is why the multimedia compression is so important for information technology development. Due to this importance, multimedia compression standards such as JPEG [6] and JPEG2000 [11] for still image compression and ISO/IEC MPEG-1 [4],MPEG-2 [7],MPEG-4 [9], ITU-T H.261 [5], H.263 [8], H.263+ [10], and JVT H.264/MPEG-4 AVC [29] for video compression have been developed since the early 90's.

Video compression is achieved through computationally complex encoding operations. Among these, the motion estimation / compensation is most complex. The complexity and the memory requirement is further increased in the long-term memory motion compensation (LTMC,[79]) that significantly improves coding efficiency by utilizing multiple reference frames. Therefore, in this dissertation, we propose low complexity motion estimation algorithms which use information of previously encoded and multiresolution frames to speed up the search. The main novelty of our proposed work comes from defining search and complexity reduction techniques that are optimized for LTMC. Also, we propose an efficient memory management control technique to reduce the decoder memory requirement for LTMC. For this we design a novel greedy search algorithm which searches for a subset of reference frames that results in minimal performance degradation rather than checking all the combination of reference frames as the optimal solution does.

Also, we propose a novel system-level error tolerance scheme specifically targeted for multimedia compression algorithms. While current manufacturing process classifies fabricated systems into two classes, namely, perfect and imperfect, our proposed scheme employs categories which are based on acceptable / unacceptable performance degradation. By enabling the use of systems that would otherwise have been discarded we seek to increase the overall yield rate in the system fabrication process. To achieve this, we propose hardware testing algorithms that aim at determining if faults in a given chip produce acceptable performance degradation, and a technique that can cancel the effect of those among the acceptable faults that can be compensated.

Chapter 1

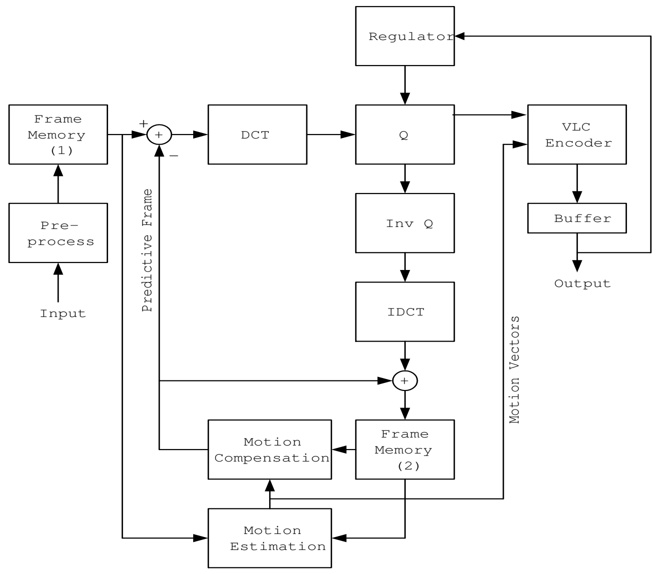
INTRODUCTION

1.1 Overview of Video Compression

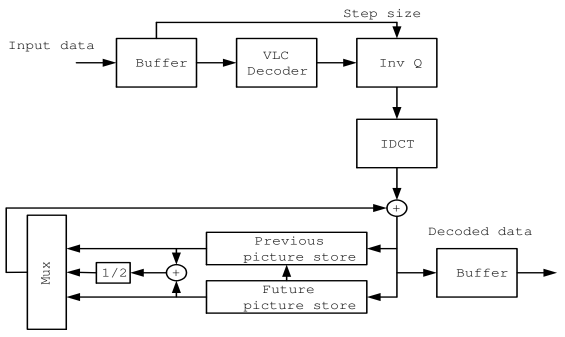
Video compression plays an important role in transmission and storage of video sequences. Lots of video applications have emerged thanks to development of video compression algorithms. These applications include High-Definition TV (HDTV) systems, Digital Versatile Disk (DVD), video conferencing, and streaming video over the Internet. Video compression standards such as ISO/IEC MPEG-1 [4],MPEG-2 [7],MPEG-4 [9], ITU-T H.261 [5], H.263 [8], H.263+ [10], and the most recent video compression standard JVT H.264/MPEG-4 AVC [29] have played a key role in the development of video compression algorithms.

For still image compression, transform based compression standards such as JPEG [6] and JPEG2000 [11] have been widely used. These still image compression algorithms can be used to encode each video frame independently. However, video sequences contain high temporal redundancy which can not be removed by only applying a still image compression algorithm to each frame of video sequence. For video signals, the correlation between two consecutive frames is usually high. Thus, to take advantage of this correlation, motion-compensated DPCM (MC-DPCM) was proposed in [37]. This technique encodes displaced frame difference values for those pixels in the changed area with respect to the previous frame. Due to its high efficiency in reducing temporal correlation, most video compression standards employ motion-compensated predictive coding which uses previous or future information to predict the current frame. The resulting video coding scheme which employs motion-compensated predictive coding and transform coding of frame residual is usually called hybrid video coding [74]. Hybrid video coding schemes achieve its high compression ratio by exploiting spatial and temporal redundancy present in video signals.

Motion-compensated predictive coding is achieved by motion estimation and motion compensation (ME/MC) for most video compression standards. Motion compensation is the process of reducing temporal redundancy and generating residual frame, and motion estimation is the process of searching a best match for a group of pixels in the current frame. Residual frames resulting from ME/MC are spatially transformed usually by discrete cosine transform (DCT, [60]) or a DCT like transform such as the integer transform employed in JVT H.264/MPEG-4 AVC [29]. The transform coefficients are, then, quantized and entropy coded for compression. Most video compression standards employ these schemes, and an example is shown



(a)



(b)

Figure 1.1: Schematic diagram of MPEG (a) encoder and (b) decoder.

in Figure 1.1 where a schematic diagram of an MPEG-2 encoder and decoder is shown [55]. The motion estimation and compensation process is located in the feedback loop of the encoding algorithm. As one can see, a frame that is available at the decoder side is fed through the feedback loop, therefore the motion compensation uses this decoded frame as a reference frame rather than the original frame. The purpose of this feedback loop is to maintain the synchronization between the encoder side and the decoder side so that the encoder and the decoder use same frame for motion compensation and frame reconstruction.

Video compression algorithms have traditionally employed computationally complex encoders and computationally light decoders. Most of the gains in compression performance have been achieved through computationally complex encoder operations. Moreover recent video compression standards which provide better compression performance have more functionalities and require more complex operations. Therefore, for video compression systems, video encoders are much more complicated, and expensive than video decoders. In the past, there were few applications where customers would have to make use of a video encoder. However, recently, many applications which require both video encoding and decoding, such as real time video-conferencing, mobile video telephony, and digital video recording (e.g., TiVo) have emerged. Therefore, it is very important to reduce the computational complexity of a video encoder retaining the compression performance. Because the motion estimation process is one of the most computationally complex operations in an encoder, reducing the computational complexity of motion estimation is very important for achieving a low complexity video encoder.

1.2 Motion Estimation

Motion estimation is an efficient tool for video compression to exploit temporal correlation between adjacent frames in a video sequence. Below, we provide a brief overview of block matching, which is the most widely used motion estimation scheme. Also, we will provide an overview of long-term memory motion compensation which is an extension of the block matching scheme.

1.2.1 Block matching

There have been a lot of research on ME/MC. Motion estimation techniques can be classified into four main groups [28]: i) gradient techniques, ii) pel-recursive techniques, iii) block matching techniques, and iv) frequency-domain techniques. Among these motion estimation techniques, block matching techniques are most widely used for video compression standards due to their simplicity and compression performance.

Block matching is based on the assumption that all pixels in a block move by the same amount [51]. Based on this assumption, block matching divides the scene (each frame) into macroblock regions (16×16 pixel blocks), and then assigns one or several motion vectors to a macroblock or sub-macroblocks. More specifically, to reduce the temporal redundancy between frames, current frame is divided into non-overlapping $N \times N$ (usually $N = 16$) blocks (macro-blocks), then the best matches for these blocks are searched in the previous frame. A schematic diagram of conventional block matching is shown in Figure 1.2.

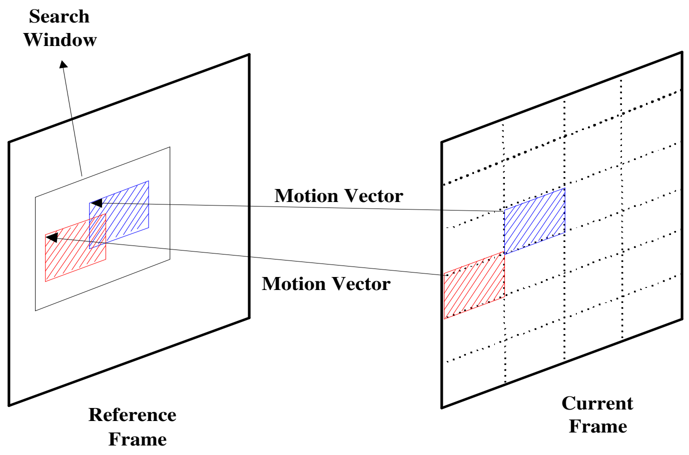


Figure 1.2: Schematic diagram of conventional block matching.

A motion vector indicates the best matching block for the given macro-block. Then this motion vector information and block differences are encoded and transmitted to the decoder. The efficiency of the motion compensation depends on the accuracy (integer pel, half pel), the motion vector, and levels of search (16×16, 8×8) [31]. The motion compensation and the motion estimation process are located in the feedback loop of the encoding algorithm. As one can see in Figure 1.1, a frame that is available at the decoder side is fed through the feedback loop, therefore the motion compensation uses this decoded frame as reference frame rather than the original frame. The purpose of this feedback loop is to maintain the synchronization between the encoder side and the decoder side that is required to reduce the temporal redundancy [67].

A motion estimation algorithm is composed of a searching process and a matching process. The matching process is the process which computes the distortion between the current block and a candidate block. The searching process selects candidate blocks from given possible set of candidates. The simplest but computationally heavy search algorithm is the full search (or exhaustive search) algorithm. For block matching motion estimation using full search, a block of size $N \times N$ (reference macro block X) of the current image is matched with all the blocks (candidate blocks Y) in the search window of size $(2w+1) \times (2w+1)$. The motion estimation can be described as,

$$D(m,n) = \sum_{(i,j)\in A} F(x(i,j) - y(i+m, j+n)) \tag{1.1}$$

$$v = arg \min_{(m,n)\in S} D(m,n), \tag{1.2}$$

where

$$A = [0, N-1] \times [0, N-1],$$

$$S = [-w, w] \times [-w, w].$$

$x(i,j)$ and $y(i,j)$ are the (i,j)th pixel values of blocks X and Y, respectively. The output from a motion estimation is a motion vector v which indicates the translational motion between a current block and the best candidate block from a search. In the matching process, if $F(\ \cdot\ )$ is the absolute value function, then the matching metric is called the sum of absolute differences (SAD), and if $F(\ \cdot\ )$ is the

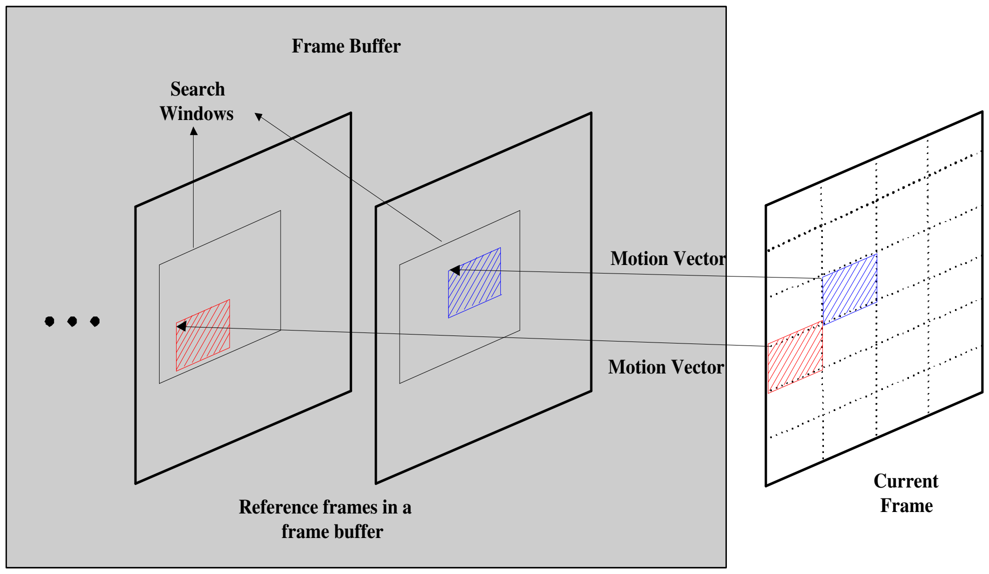


Figure 1.3: Block matching for the long-term memory motion compensation.

square function, then the matching metric is called the sum of squared differences (SSD).

1.2.2 Long-term memory motion compensation

The long-term memory motion compensation is an extended version of conventional block matching which can provide significant compression performance enhancement.

Long term memory motion compensation (LTMC) has been recently introduced in the literature [79] as an approach to extend the motion search range by using multiple decoded frames as reference frames, instead of using just one decoded frame as in conventional motion compensation. A schematic diagram of LTMC is shown in Figure 1.3. By extending the motion search range, motion compensation gain can be significantly increased (1-2dB increase by incorporating with h.263+ [79]). Due to these compression performance gains, LTMC was accepted as the ME/MC scheme for the most recent video compression standard JVT H.264/MPEG-4 AVC [29, 1]. However, the computational complexity of motion estimation for LTMC is significantly increased, as well, as compared to a single-frame approach. Therefore, reducing the computational complexity is even more important in the context of LTMC.

1.3 Low Complexity Motion Estimation Algorithm

In this section, we provide a brief overview of low complexity motion estimation algorithms for conventional block matching and LTMC. The computational complexity of motion estimation process is determined by the total number of candidate vectors searched, the order of search, and the computational cost of metric computation for each candidate vector. Fast motion estimation algorithms can be classified into fast search algorithms or fast matching algorithms. Fast search algorithms search a subset of candidate vectors instead of the whole set of candidate vectors in a search window to speed up the motion estimation process, and fast matching algorithms enhance the speed of motion estimation by reducing the computational cost of matching metric.

1.3.1 Low complexity motion estimation for conventional block matching

For a fast search, the search for the best motion vector takes place within a subset of candidate vectors. A fast search can use a fixed subset of candidate vectors as in [51], or it may use a variable subset that depends on scene characteristics. This variable subset can be determined based on the uni-modal error surface model where the farther we move from the best position, the worse the match becomes [40]. Fast search examples based on the uni-modal error surface model are three step search [42, 45], new three step search [48], two dimensional logarithmic search [40], conjugate direction search [66], parallel hierarchical one dimensional search [21], block-based gradient descent search [52], center-biased diamond search [68], and enhanced predictive zonal search (EPZS) [69, 70]. Other fast search techniques not using the uni-modal error surface model assumption are successive elimination algorithm [49] and candidate selection algorithm based on spatial, temporal, and hierarchical motion vector correlations proposed in [20].

Fast matching algorithms reduce the computational complexity of motion estimation by devising matching criteria which require less computational complexity.

Examples of fast matching are the alternating 4:1 pixel decimation technique proposed in [51], fast matching by three fast measures based on the triangular inequalities [50], fast matching using bit-map transformed pixel data [56], deterministic testing fast matching (DTFM), which has been used in many encoder reference software implementations such as [3, 35, 1], and hypothesis testing fast matching (HTFM) proposed in [46].

There are motion estimation algorithms which combine a fast search technique and a fast matching technique. An example is multiresolution motion estimation (or hierarchical motion estimation) proposed in [33, 19, 15, 14, 80, 72, 81, 47, 57, 13]. The multiresolution motion estimation (MRME) is a fast motion estimation algorithm based on using image pyramids which are composed of several low resolution versions of a frame. The MRME computes a coarse motion vector using the lowest resolution version of a frame, and then refines the coarse motion vector using the higher resolution versions of the frame.

1.3.2 Low complexity motion estimation for LTMC

LTMC is an extension of conventional block matching. Thus, fast search and fast matching algorithms introduced in the previous section can be directly applied to LTMC. However LTMC has its own characteristics, which can be used to design a fast algorithm different from those used in conventional block matching. The main characteristic of LTMC is that a very large number of motion candidates should be searched, and due to the multi reference frame nature of LTMC, many among these motion candidates are similar to each other.

In [79], a fast search algorithm for LTMC is proposed to overcome the high computational complexity. This fast search algorithm employs a hierarchical search strategy that uses a norm ordered search method based on the triangle inequality. In [77, 76], a lossy version of the norm ordered search was proposed to speed up the hierarchical search strategy for LTMC. In [71], the enhanced predictive zonal search (EPZS, [69, 70]) was modified for LTMC. In this algorithm, the authors proposed to use sophisticated predictor and EPZS for search.

In this dissertation we propose two low complexity motion estimation algorithms for LTMC. The first algorithm we propose locates a motion search window at a good position in a frame buffer by using the encoding information of previous frames. We call this proposed algorithm *directed search*. By locating motion search windows at good positions, the performance gains are achieved as compared to the conventional schemes which locate search windows at fixed spatial positions. Also, because an initial search point selected by our algorithm is close to best matches, we can use smaller motion search windows, thus, reduce the computational complexity. To reduce the computational complexity more, we employ HTFM proposed in [46] as a matching metric.

The second algorithm we propose locates the motion search windows at good positions by using multi-resolution search. In our proposed algorithm, we locate a motion search window by using multi-resolution search, and then decide the search range based on the characteristic of the frame gathered from a coarse resolution version of the frame. While multi-resolution motion estimation algorithms are well known, the main novelty of our proposed work comes from defining search and complexity reduction techniques that are optimized for LTMC. In this algorithm, we can also detect cases such as occlusion and scene changes early, which allows to discard early candidates that will not provide a good match. For the proposed algorithm, to reduce the computational complexity more, we employ HTFM as a matching metric.

1.4 Fault Resilient Compression

To achieve real time video encoding, complex encoding operations such as motion estimation, DCT, and entropy coding have been implemented as hard wired circuits using VLSI technologies [59]. Video compression in itself is a "lossy" process, i.e., only approximate versions of the acquired data are stored, transmitted and displayed. For this reason, often circuit imperfections result in perfectly usable compressed data which can be decompressed with a possibly degraded quality. Therefore, by enabling the use of systems that would otherwise have been discarded, we can increase the overall yield rate in the system fabrication process, and decrease the cost of video encoders.

For video compression systems, not all the faults in the implementation of a video encoder are intolerable from the system level point of view. Consider as an example a standard MPEG-2 encoder which contains building blocks, such as motion estimation/compensation, discrete cosine transform (DCT), quantization, entropy coding, and various memory buffers [55]. Faults in each of these components have completely different effects on the operation of the complete system. This is illustrated by Table 1.1 [25], which considers the effects of faults such as memory defects and numerical errors in the DCT computation [54, 36, 75], as well as errors in motion estimation. We consider the resulting errors to be catastrophic if they prevent the creation (or decoding) of a valid bitstream. Special consideration is given to faults that produce an error propagation. It is important to note that many faults affecting different parts of a video compression system are in fact *non catastrophic*, although obviously they result in degradation in coding efficiency and/or visual quality.

In this dissertation, we propose a novel system-level error tolerance scheme specifically targeted for ME/MC process of video compression. Instead of the perfect / imperfect categories which are employed by current manufacturing process, our proposed scheme employs categories which are based on acceptable / unacceptable performance degradation. For this, we analyze the effect of faults in the motion compensation stage of video encoding process, and propose a fault concealment technique which can compensate for the effect of some faults. Also, in this

	Catastrophic Error	Error Propagation	Visual Quality Degradation	Coding Efficiency Degradation
Frame Memory(1)	No	No	Yes	Yes
DCT	No	No	Yes	Yes
Q	No	No	Yes	Yes
Inv Q	No	Yes	Yes	Yes
IDCT	No	Yes	Yes	Yes
Frame Memory(2)	No	Yes	Yes	Yes
ME	No	No	Possible	Yes
MC	No	Yes	Yes	Yes
VLC Encoder	Yes	Yes	Yes	Yes

Table 1.1: Fault effect analysis of a MPEG-2 video encoder depicted in Figure 1.1 (a).

work, we propose a testing algorithm which requires a small number of test vectors, thus leading to low cost testing.

This dissertation is organized as follows. In Chapter 2, we propose the directed search as a fast motion estimation for LTMC. In Chapter 3, we propose a novel multi-resolution motion estimation for LTMC. In Chapter 4, we propose an efficient memory management control algorithm. In Chapter 5, we propose an error tolerance scheme for ME/MC process.

Chapter 2

DIRECTED SEARCH

2.1 Introduction

Fast development of semiconductor technologies has made high capacity, fast access memories available at increasingly lower cost. Long-term memory motion compensation (LTMC) extends the motion search range to multiple decoded frames to significantly increase prediction gain performance [79]. Due to the potential performance gains [78], it is used in the most recent H.264/MPEG-4 AVC video compression standard [29]. However, the computational complexity of motion estimation (ME) for LTMC is also significantly increased as compared to a single-frame approach [38]. A schematic diagram of LTMC is shown in Figure 1.3. As one can see from this figure, search range for LTMC is extended to multiple decoded frames as compared with a conventional block matching in Figure 1.2. By extending the motion search range, motion compensation (MC) gain is significantly increased.

Due to these performance gains, LTMC is used as the ME/MC scheme for JVT H.264/MPEG-4 AVC compression standard.

However, the computational complexity of motion estimation (ME) for LTMC is significantly increased, as well, as compared to a single-frame approach. Therefore, reduction of the required computational complexity is one of the most challenging issues for LTMC. In [79], a fast search algorithm for LTMC is proposed to overcome this high computational complexity. This fast search algorithm employs a hierarchical search strategy that uses a norm ordered search method based on the triangle inequality. In this algorithm, candidate blocks in the search space having a similar norm to the norm of the block being compensated are tested first. In [77, 76], a lossy fast estimation algorithm, which we will call the WLG algorithm, is proposed to speed up the hierarchical search strategy for LTMC. The WLG algorithm employs sub-sampling of both the search range and the block for which distortion is measured, in order to speed up the ME for LTMC. The sub-sampling of the search space was achieved through loosening the lower bound for triangle inequality testing, and the sub-sampling of the block was achieved by measuring the block activity in WLG algorithm.

The WLG algorithm locates the motion search window (MSW) at a fixed position in each of the frames. However, if we locate MSWs at fixed locations in a frame buffer for a given macro-block (MB, a block of 16×16 pixels), it is highly possible that the "oldest" frames in a frame buffer do not contain matching blocks

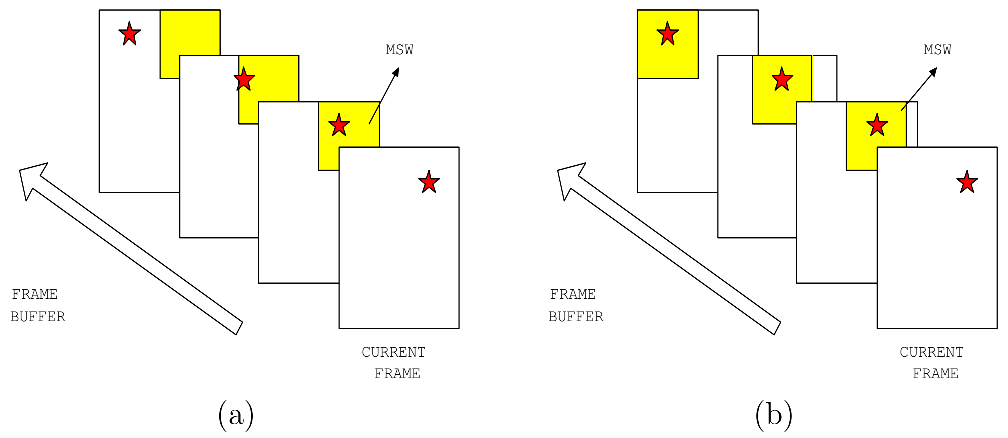


Figure 2.1: Schematic diagram of (a) fixed search window location and (b) directed search.

due to the reduced correlation between the current frame and the reference frames located further in the frame buffer. Therefore, if we can locate MSWs at a good position adaptively in a frame buffer, we can improve the performance of LTMC, specifically for sequences with high motions. This scenario is shown in Figure 2.1. As one can see from this figure, for high motion scenes, a fixed window location can not follow the real matching, however, directed search can follow the real matching.

In this chapter, we propose a novel low complexity ME algorithm for LTMC which employs a directed search. The directed search locates motion search windows in a frame buffer by using ME results of previous frames. This lets the location of the motion search windows change adaptively as the search proceeds to older frames in the frame buffer.

The main benefit of the directed search comes from the fact that it can detect a large motion in a frame buffer. LTMC algorithms like the WLG algorithm use a fixed set of motion vectors, for example, a search region $\Gamma = [-16, 15] \times [-16, 15]$. Therefore, if a better match is located outside of the fixed search region Γ, it will be missed during the search. Instead, our proposed ME algorithm uses a flexible set of motion vectors in a frame buffer, for example, $\Gamma' = [-16 + uc^x, 15 + uc^x] \times [-16 + uc^y, 15 + uc^y]$, where uc^x and uc^y are adaptively determined by the directed search, and therefore, makes it more likely that large motions can be detected and compensated.

In addition, because the directed search keeps track of best matched blocks in a frame buffer, we can reduce the computational complexity significantly by reducing the motion search window area. To further reduce the computational complexity, we use hypothesis testing fast matching (HTFM, [46]) as a fast matching criterion. HTFM can reduce the computational complexity greatly compared to a conventional deterministic testing fast matching (DTFM, [46]) by allowing an early termination of matching metric computation based on the likelihood that a motion candidate will not be the best one.

Simulation results show that by employing both HTFM and the directed search with reduced motion search window, we can reduce the computational complexity approximately 75%-90% and 10%-67%, as compared to full search and the WLG algorithm, respectively. It is also observed that by locating MSWs at better positions, our proposed algorithm can achieve simultaneous complexity reduction and

PSNR gain for high motion video sequences as compared to full search and the WLG algorithm.

2.2 Directed Search

A MSW (or a motion search area) is a set of overlapped blocks in a reference frame that will be searched for a best matching block. Due to the temporal correlation between adjacent video frames, conventional motion estimation algorithms, which employ only one reference frame, center the MSW at the same spatial location of the block being compensated. However it is clear that as the temporal displacement between a current frame and a reference frame increases, as in the case in LTMC, the correlation between these two frames is reduced as well. Therefore, if we locate a MSW at a fixed location for the given macro-block, the correlation between a macro-block in a current frame and blocks in a MSW also decreases as the temporal displacement between a current frame and a reference frame increases. This can be easily seen for example in cases where there is camera panning. Additionally, if the translational motion of an object in a macro-block (MB) of a current frame is very large, then it is highly possible that the object will not be included in the motion search window if the temporal displacement between a current frame and a reference frame is large enough. LTMC uses multiple reference frames for ME, therefore it is very likely that for high motion scenes the "oldest" frames in the frame buffer will not provide good matches if MSWs are restricted to be same in all frames. In this

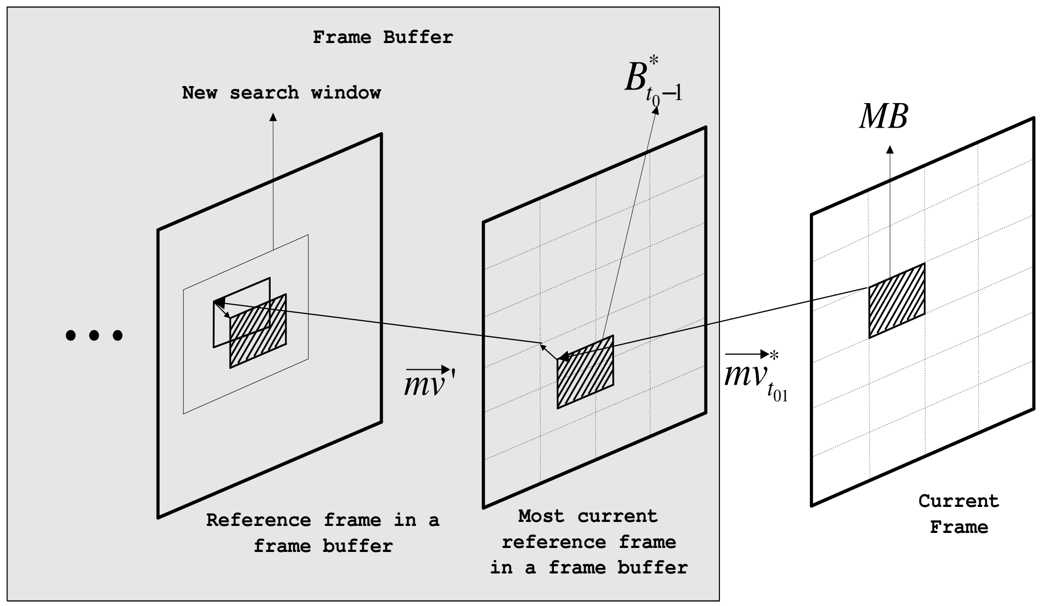

Frame Buffer
New search window
$B^*_{t_0-1}$
MB
$\overrightarrow{mv}'$
$\overrightarrow{mv}^*_{t_{01}}$
Reference frame in a frame buffer
Most current reference frame in a frame buffer
Current Frame

Figure 2.2: Schematic diagram of directed search. The directed search strategy locates motion search windows in a frame buffer for a given macro-block MB. $B^*_{t_0-1}$ is the best matched block in the reference frame f_{t_0-1} for the given macro-block. The directed search locates the motion search window in the reference frame f_{t_0-2} by using the motion vector information of the macro-block that has the largest overlap with $B^*_{t_0-1}$.

section, we propose a directed search as a MSW location algorithm that can solve the problem caused by the fixed location of MSWs in reference frames.

2.2.1 Description of directed search

Before we provide a detailed discussion of directed search strategy, let us introduce the following notation:

- f_{t_0}: frame whose time index is t_0,
- $MB(i, t_0)$: The ith macro-block in f_{t_0},

- $B^*(i, t_0)$: The block that best matches $MB(i, t_0)$ using LTMC,
- ${B^*}_{t_1}(i, t_0)$: The block that best matches $MB(i, t_0)$ in a reference frame f_{t_1},
- $\overrightarrow{mv}^*_{t_1}(i, t_0)$: Spatial displacements between ${B^*}_{t_1}(i, t_0)$ and $MB(i, t_0)$.

In the above definitions, a block is a set of 16×16 pixels which can be located anywhere in a frame, while a MB is a block which is aligned at 16×16 pixels grid. This is an important distinction because motion information is known for MBs but not for all blocks. Using the above definitions, for a given MB, $MB(i, t_0)$, in the current frame, the best motion vector $\overrightarrow{mv}^*_{t_0-1}(i, t_0)$ is found using conventional single-frame motion search algorithm (e.g., full search (FS), three-step search [42], or two-dimensional logarithmic search [40]) in the most recent reference frame located in a frame buffer, and the results are stored. The directed search uses the best match in frame f_{t_0-1} as well as the results of matching MBs in frame f_{t_0-1} to blocks in frame f_{t_0-2}. ${B^*}_{t_0-1}(i, t_0)$ overlaps with up to 4 MBs for which motion was computed before. Each of these MBs points to a best match in frame f_{t_0-2}. Our algorithm then chooses one among these 4 MBs, and makes the corresponding position in frame f_{t_0-2} the center of the search to find a match for $MB(i, t_0)$. In the following, we describe the macro-block selection algorithm used to choose the path of the directed search.

2.2.1.1 Macro-block selection for directed search

The best matching block ${B^*}_{t_0-1}(i, t_0)$ overlaps with up to 4 MBs in frame f_{t_0-1}. The motion vectors for these overlapped MBs for reference frame f_{t_0-2} and SAD values corresponding to these motion vectors (MV) which were stored at the previous motion estimation stage are available as information for our directed search. The other information we can use in deciding which MB to use is the area of the overlap with each macro-blocks overlapping with ${B^*}_{t_0-1}(i, t_0)$. We compare two simple approaches of MB selection.

The first MB selection algorithm (MS I) selects directions based on the ovelapping areas. Since block matching ME algorithms are based on the assumption that all pixels in a block move by the same amount, a good MV candidate can be obtained by using only a fraction of pixels in a block [51]. Therefore, among MBs that overlap with ${B^*}_{t_0-1}(i, t_0)$, MS I chooses the macro-block $MB(i', t_0 - 1)$ that contains the largest fraction of ${B^*}_{t_0-1}(i, t_0)$. If there is a tie for the areas, then we choose the overlapped MB which has a MV with the smallest SAD value calculated at the previous motion estimation stage.

The second approach (MS II) uses previously stored SAD values of the overlapped MBs as well as the areas of overlapping. More specifically, among at most 4 overlapped MBs, MS II calculates the area of the overlapped region for each MB. If the area of an overlapped region is greater than 75%, then MS II chooses this MB $MB(i', t_0 - 1)$, otherwise among these MBs, MS II selects the MBs whose

Table 2.1: Simulation results of directed search using MS I (DS I) and using MS II (DS II). T is the time spent on LTMC per frame and PSNR represents the prediction error variance in dB. For each sequence, frames from 1 to 300 are used with frame skip parameter 2. The length of the frame buffer used is set to 10, and spiral ordered full search is used. For the computational time, the median of 5 experimental results is used. Based on the results we adopt MS I as the macro-block selection algorithm for our directed search.

	Directed Search I		Directed Search II	
	T [sec]	PSNR [dB]	T [sec]	PSNR [dB]
Foreman(QCIF)	6.49	29.81	6.47	29.81
Stefan(QCIF)	9.47	22.07	9.42	22.06
Mother & daughter(QCIF)	6.24	34.70	6.24	34.70

overlapped area is greater than 20%, then compares the SAD values which are computed at the previous ME stage for these selected MBs, and then chooses the MB $MB(i', t_0 - 1)$ whose SAD value is smallest.

If a MB $MB(i', t_0 - 1)$ is chosen among overlapped MBs, then by using the MV $\overrightarrow{mv}^*_{t_0-2}(i', t_0 - 1)$ of $MB(i', t_0 - 1)$, the center of the MSW for $MB(i, t_0)$ in frame f_{t_0-2} is located at $\overrightarrow{mv}^*_{t_0-2}(i', t_0 - 1) + \overrightarrow{mv}^*_{t_0-1}(i, t_0)$. In this way, MSWs can be adaptively located in a frame buffer. The schematic diagram of the directed search is shown in Figure 2.2.

Because for most of the cases, the overlapped MBs correspond to similar MVs, the performances of directed search using MS I (DS I) and directed search using MS II (DS II) are similar, as can be seen in Table 2.1. Since MS I is a much simpler approach, we choose it for our system.

2.2.1.2 Detailed description of the directed search strategy

The directed search using deterministic testing fast matching (DTFM, [46]), MS I, and outward spiral ordered full search method is described as follows:

Algorithm 1: Directed Search

Step 0: Set $k = 1$. For $MB(i, t_0)$, find ${B^*}_{t_0-1}(i, t_0)$, then store the location of ${B^*}_{t_0-1}(i, t_0)$ and the corresponding SAD value. Set $B^*(i, t_0) = {B^*}_{t_0-1}(i, t_0)$.

Step 1: If k is greater than the number of frames in the frame buffer, then return the MV corresponding to $B^*(i, t_0)$, and stop. Otherwise go to Step 2.

Step 2: According MS I, choose the MB providing best overlap with ${B^*}_{t_0-k}(i, t_0)$.

Step 3: Locate the MSW in frame f_{t_0-k-1} for $MB(i, t_0)$ by using the MV information of the chosen macro-block at Step 2.

Step 4: Perform the ME in this MSW. During the motion search in this MSW, find ${B^*}_{t_0-k-1}(i, t_0)$. If this block provides smaller SAD value than the best SAD found so far, then set $B^*(i, t_0) = {B^*}_{t_0-k-1}(i, t_0)$. Increase k by 1, then go to Step 1.

In Step 4 of the above algorithm, if we use a fast matching criterion like DTFM or HTFM, we decide ${B^*}_{t_0-k-1}(i, t_0)$ based on the partial metric computation.

Table 2.2: Motion tracking performance of the directed and non-directed searches. Each entry shows the percentage of blocks for which the best match found with the extreme search (search over all frame) is also found using directed search and non-directed search, both based on the normal small search window. In this table, 1st, 2nd, and 3rd represent the best, the second best, and the third best match found from extreme search. "Other" represents the matching block which is worse than the third best match.

	Directed Search			
	1st	2nd	3rd	other
Foreman(QCIF)	93.78%	1.06%	0.39%	4.77%
Stefan(QCIF)	88.55%	1.83%	0.84%	8.78%
Glasgow(QCIF)	90.20%	1.29%	0.53%	7.98%
Coast guard(QCIF)	99.47%	0.07%	0.03%	0.43%
Stefan(CIF)	65.75%	5.73%	2.46%	26.06%
Foreman(CIF)	83.38%	2.46%	1.37%	12.79%

	Non-Directed Search			
	1st	2nd	3rd	other
Foreman(QCIF)	91.13%	2.37%	0.76%	5.74%
Stefan(QCIF)	74.00%	6.35%	3.98%	15.67%
Glasgow(QCIF)	89.70%	1.11%	0.56%	8.63%
Coast guard(QCIF)	98.76%	0.49%	0.11%	0.64%
Stefan(CIF)	50.34%	5.50%	3.24%	40.92%
Foreman(CIF)	75.61%	4.75%	2.27%	17.37%

2.2.2 Comparison of directed and non-directed search

To demonstrate the benefits of our approach, we simulate the directed and the non-directed search algorithms. All video sequences used for these simulations are made by extracting the last 100 frames from the original sequence. For this simulation the outward spiral ordered full search approach is used. DTFM is used as a matching criterion for LTMC, and all frames are coded as P frames to analyze the efficiency of the search strategies. The number of reference frames in the frame buffer is 10.

For comparison, we find the top three matching blocks for each block *using the whole frame as motion search window.* This allows us to locate the absolute best MVs even when high motion scenes are considered. The results from this simulation are shown at Table 2.2. Each entry in Table 2.2 shows the percentage of blocks for which the best match found with the extreme search (search over the whole frame) is also found using directed search and non-directed search, which both use the normal small search window. From these results, one can see that the directed search can locate MSWs better than a non-directed search. Also one can see that for video sequences such as Stefan, which contains large motion, the performance of directed search strategy is much better than that of non-directed search as we had suggested earlier. As the frame size increases, it is highly probable that the motion becomes larger and larger, and therefore the relative performance of directed search will be better.

However, the directed search has an overhead, because it adaptively locates the motion search window in the frame buffer, and therefore it requires a large range of motion vectors. That is, if the non-directed search uses a fixed set of motion vectors, for example, $\Gamma = [-16, 15] \times [-16, 15]$, the directed search uses a flexible set of motion vectors $\Gamma' = [-16 + uc^x, 15 + uc^x] \times [-16 + uc^y, 15 + uc^y]$, where uc^x and uc^y are adaptively determined. As a result of this loosened restriction on the motion vector range, one possible drawback is the increased bit rate to transmit the motion vector information to a decoder.

Table 2.3: MV overhead for LTMC using the directed search (DS), non-directed search (F10), and the conventional motion estimation which uses just one reference frame (F1), and the total number of bits to represent difference frames. For this simulation, all frames except the first frame are set as P frames. H.263+ ([10]) and the MV prediction algorithm proposed in [79] is used to generate the results.

	Motion Vector Overhead			Total Bits Per Macro-block		
	DS bits/MB	$F10$ bits/MB	$F1$ bits/MB	DS bits/MB	$F10$ bits/MB	$F1$ bits/MB
Foreman(QCIF)	13.68	13.20	7.60	339.20	339.52	374.96
Stefan(QCIF)	12.16	10.88	6.80	874.88	917.52	964.56
Glasgow(QCIF)	10.08	9.92	6.08	493.28	494.64	511.36
Coast guard (QCIF)	8.32	8.32	3.44	526.80	528.24	589.76
Stefan(CIF)	14.8	13.2	8.48	632.00	702.72	729.68
Foreman(CIF)	15.68	13.68	8.56	229.76	231.28	256.40

In [79], a MV prediction algorithm is proposed by modifying the H.263-based median prediction of the spatial displacement vectors [8]. This MV prediction algorithm is based on extending the region of support for prediction, and uses temporal displacements to increase the correlation between the predicted motion vector and a motion vector to be predicted. Even if the directed search loosens the restriction on the size of MVs, the MV prediction algorithm in [79] is found to be very efficient in our simulations. The bit overheads for transmitting MVs are shown in Table 2.3. The histograms of spatial displacements mv_x and mv_y, and the residual components after prediction $DMV_x = mv_x - \hat{mv}_x$ and $DMV_y = mv_y - \hat{mv}_y$ are shown for Stefan QCIF sequence in Figure 2.3 for the directed search, and in Figure 2.4 for the non-directed search. Therefore we employ this prediction method

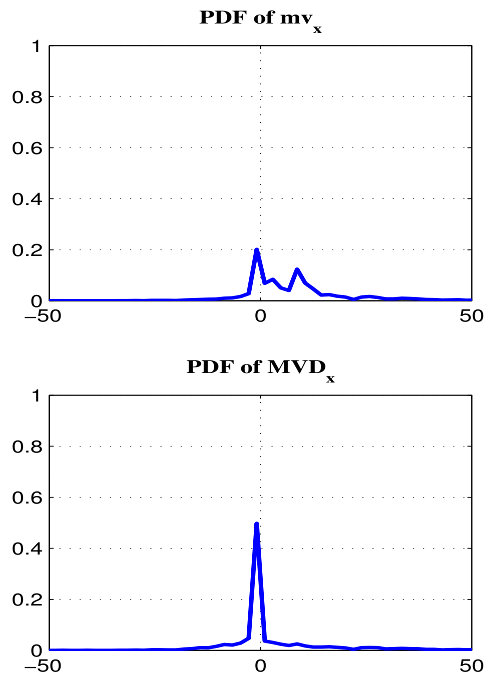


Figure 2.3: Probability density of spatial displacements mv_x, mv_y, $DMV_x = mv_x - \hat{mv}_x$, and $DMV_y = mv_y - \hat{mv}_y$ for *Stefan* QCIF sequence by using the directed search. $\hat{mv}_x$ is the predicted horizontal motion vector component, and $\hat{mv}_y$ is the predicted horizontal motion vector component.

to reduce the bit overhead for transmitting the MVs. To calculate the bit overhead we use the VLC table for motion vector difference (MVD) that is proposed in [10]. As one can see from Table 2.3, the required bit overhead for transmitting MVs is slightly higher for the directed search than for the non-directed search. It can be seen that for video sequences like Stefan, which contains large motion, the required bit overhead is larger than for other sequences which contain small motions. As the frame size increases, it is highly probable that larger motion will be present, so that the bit overhead will also increase. However, because the bit rate saving

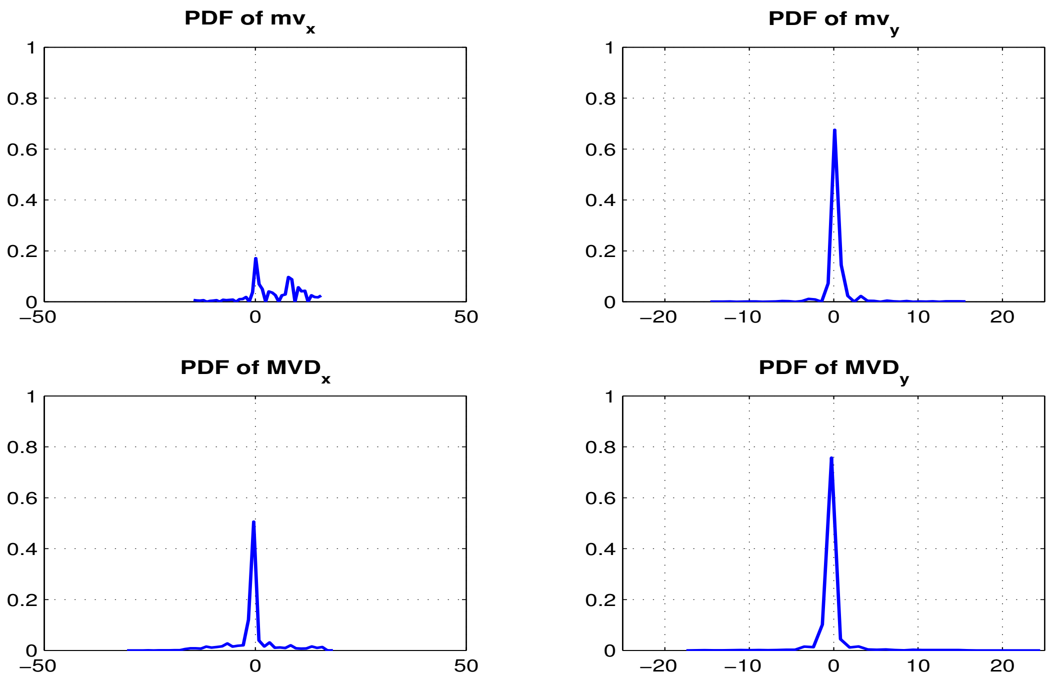
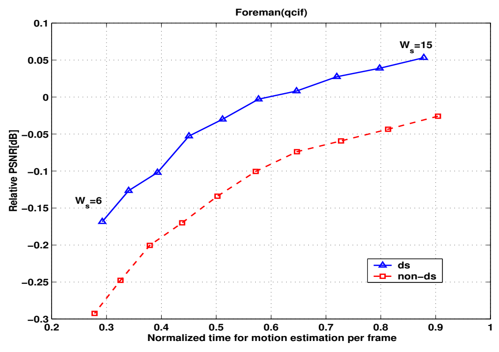
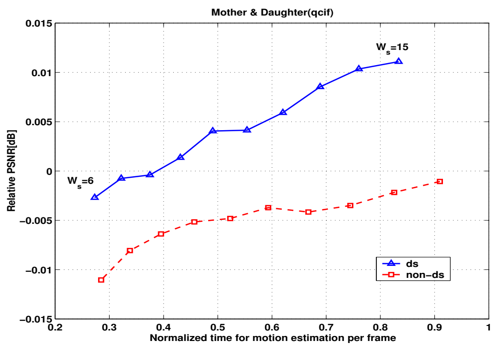


Figure 2.4: Probability density of spatial displacements mv_x, mv_y, $DMV_x = mv_x - \hat{mv}_x$, and $DMV_y = mv_y - \hat{mv}_y$ for *Stefan* QCIF sequence by using the non-directed search strategy. $\hat{mv}_x$ is the predicted horizontal motion vector component, and $\hat{mv}_y$ is the predicted horizontal motion vector component.

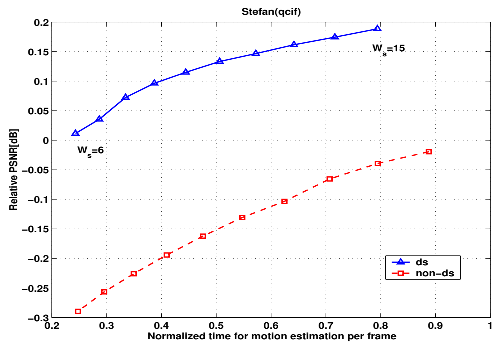
effect is much greater for reducing the residual frame energy, the overhead is well compensated by the reduction in bit rate needed for coding the residual frame energy.



(a)



(b)



(c)

Figure 2.5: The effects of reducing the motion search window in a frame buffer for directed (DS) and non-directed search (NDS) strategy, when the area of motion search window is reduced to $2W_s \times 2W_s$. $W_s = \{6, \ldots, 15\}$ is used for the simulation. (a) QCIF Foreman, (b) QCIF Mother & Daughter, (c) QCIF Stefan.

2.3 Low Complexity Motion Estimation Algorithm for LTMC

To reduce the computational complexity, a reduced motion search range (sub-sampling of search range) can be used, along with fast matching. Sub-sampling of the search range for WLG algorithm is implemented by excluding blocks that are too different from the search space [77]. Fast matching for WLG algorithm is realized by stopping the distortion computation at a certain level in the hierarchy of triangle inequalities [79]. In the proposed algorithm, we employ HTFM ([46]) as our technique to implement fast matching, and reduced size of motion search window with DS to reduce the motion search range.

For the proposed algorithm, we use HTFM as the fast matching criterion of the motion estimation for LTMC. HTFM can reduce the computational complexity greatly as compared to DTFM, by allowing an early termination of SAD calculation based on the likelihood (probability of false alarm, P_f) that current SAD will be greater than the best-SAD-found-so-far, given the partial SAD [46]. For the block sub-sampling method for HTFM, the HTFM in the proposed algorithm employs the UNI sub-sampling partitioning by which the correlation between partial SAD and SAD is maximized [46]. The UNI sub-sampling shows better performance than the ROW sub-sampling which is the row-by-row partitioning. However, the irregular memory access pattern is a disadvantage of UNI sub-sampling. In addition, the histogram parameter estimation is performed at every 10 frames for HTFM. Simulation results show that by employing HTFM only, we can reduce the computational complexity approximately 40%-50% for LTMC at the cost of very minor degradation in PSNR gain (0.005-0.05dB).

Since the directed search tends to locate the MSWs around potentially good candidates, it should be possible to reduce MSW size. Therefore, if we use the directed search for ME for LTMC, we can reduce the area of MSWs in a frame buffer, and can save the computational complexity greatly at the cost of slight increase in the residual frame energy. In Figure 2.5, we show the effects of reducing the area of motion search window in a frame buffer for the directed and the non-directed search strategies. In this figure, each computational time is normalized by T_{F10} which is the computation time of a full search with non-directed search. Relative PSNR is $PSNR - PSNR_{F10}$, where PSNR is average error variance in dB scale. For each sequence, frames from 1 to 300 are used with frame skip parameter 2. The length of the frame buffer used is set to 10, and spiral ordered full search is used with DTFM. The median of 5 experimental results is chosen to estimate computation complexity. For this simulation, we use $[-16, \ldots, 15]$ as a motion search window for adjacent frames, and we use motion search window whose area is $2W_s \times 2W_s$ in a frame buffer for LTMC. From Figure 2.5, if we use $W_s = 12$, then we can achieve computational complexity savings of around 35%, 37%, and 42% for Foreman (QCIF), Mother & Daughter (QCIF), and Stefan (QCIF), respectively. The cost of the reduced computational complexity is 0.051dB, 0.005dB, and 0.052dB PSNR decrease for each sequences. One can see that in spite of the PSNR decreases due to the reduced computational complexity, the directed search still can achieve smaller residual frame energy than the non-directed search. The PSNR gain (residual frame energy reduction) of directed search is greater for Stefan than for Foreman or Mother & Daughter. This is because directed search has advantages for video sequences which have large motions, and the gains tend to be greater for larger sequences, as shown

Table 2.4: Computational complexity and prediction error variance in dB scale for F1, F10. F1: Outward spiral ordered conventional full search which uses just one reference frame, and employs DTFM. F10: Outward spiral ordered full search which employs non-directed search and DTFM, and the frame buffer length is 10. T means the time required for LTMC per frame and PSNR represents the prediction error variance in dB scale. For each sequence, frames from 1 to 300 are used with frame skip parameter 2. The length of the frame buffer used is set to 10, and spiral ordered full search is used. For the computational time, median is selected among 5 experimental results for fair comparisons.

	Foreman(QCIF)		Mother & Daughter(QCIF)		Stefan(QCIF)	
	T [sec]	PSNR [dB]	T [sec]	PSNR [dB]	T [sec]	PSNR [dB]
F1	0.88	27.88	0.70	33.54	1.13	21.13
F10	6.73	29.75	6.84	34.68	10.77	21.87

before. Because setting $W_s = 12$ is a reasonable choice for QCIF video sequences from the simulation, we use $W_s = 12$ for the proposed fast motion estimation algorithm for LTMC for QCIF video sequences.

2.4 Experimental Results

In this section, simulation results of the proposed ME algorithm for LTMC are presented. The simulation is done using an Intel PentiumIII 750 MHz PC with 384MB RAM. For the directed search of the proposed algorithm, the motion search range is set to $[-16, \ldots, 15] \times [-16, \ldots, 15]$ in most current reference frame, and $[-W_s + uc^x, \ldots, W_s - 1 + uc^x] \times [-W_s + uc^y, \ldots, W_s - 1 + uc^y]$ for motion estimation in other reference frames, where $W_s = 12$. For WLG algorithm, the motion search range is set to $[-16, \ldots, 15] \times [-16, \ldots, 15]$ for all reference frames. The simulation

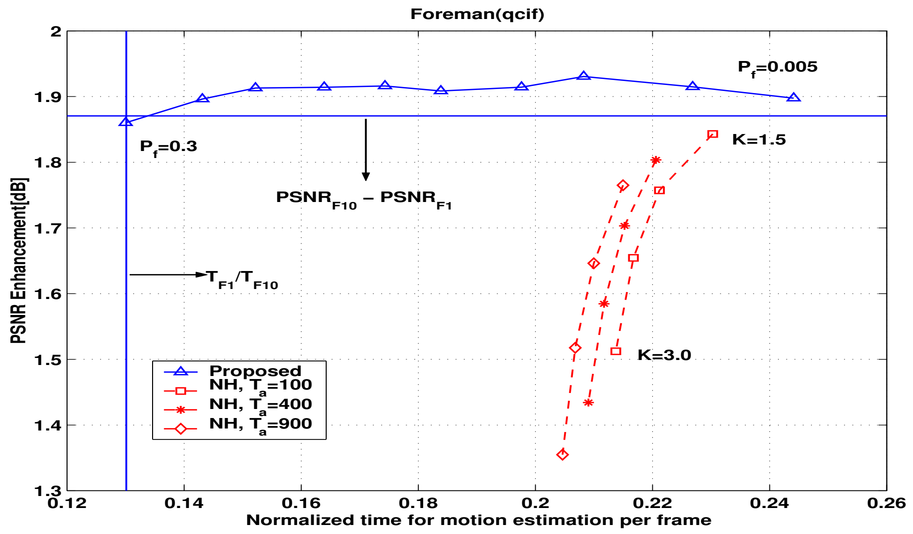


Figure 2.6: Comparison of the proposed algorithm with the hierarchical norm ordered search (NH) algorithm for Foreman QCIF sequence.

results are normalized by and compared to the results of $F1$ and $F10$ algorithm. $F1$ is the algorithm which uses the outward spiral ordered conventional full search which uses just one reference frame, and employs DTFM. $F10$ is the algorithm which uses the outward spiral ordered full search which employs non-directed search and DTFM, with frame buffer length 10.

In Figures 2.6 - 2.8, we compare the performance of the proposed algorithm with the performance of the WLG algorithm. For the proposed algorithm the set of P_f is $\{0.005, 0.01, 0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3\}$. For the WLG algorithm, the set of values for the activity threshold T_a is $\{100, 400, 900\}$, and the set of values for K, which is the control parameter to terminate the norm-ordered search early, is $\{1.5, 2.0, 2.5, 3.0\}$. The matching criterion used for the WLG algorithm is SAD

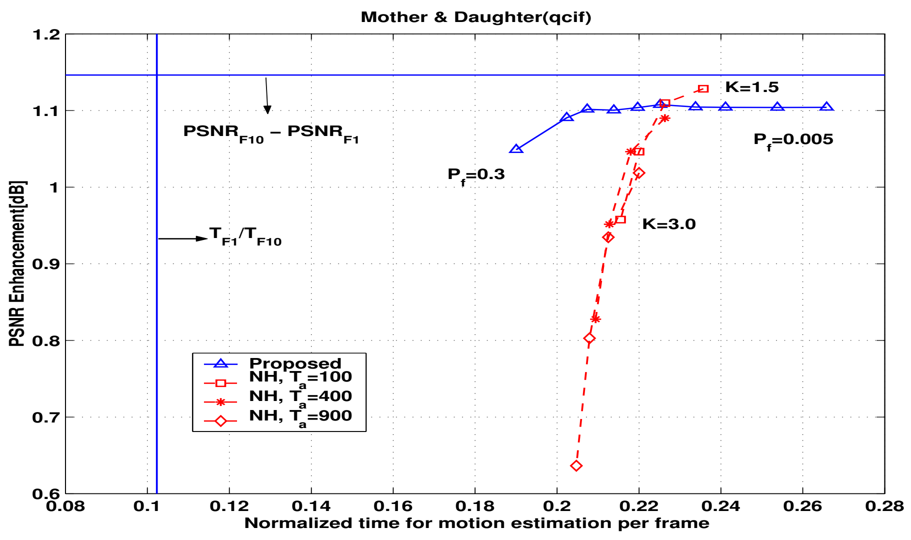


Figure 2.7: Comparison of the proposed algorithm with the hierarchical norm ordered search (NH) algorithm for Mother & Daughter QCIF sequence.

with DTFM, because SAD is most popular matching criterion for most of real applications due to the simple implementation structure. In these figures, NH algorithm represents a WLG algorithm which employs SAD as a matching metric, DTFM as a fast matching criterion, 4 hierarchical levels of triangle inequalities, and two fast lossy search methods [77]. Each computational time is normalized by T_{F10} which is the computational complexity of $F10$ algorithm in Table 2.4. PSNR enhancement is $PSNR - PSNR_{F1}$, where PSNR is average energy of residual frame in dB scale, and $PSNR_{F1}$ is the average energy of residual frame in dB scale for $F1$ algorithm in Table 2.4. For comparison, T_{F1}/T_{F10} and $PSNR_{F10} - PSNR_{F1}$ are shown in these figures. For both algorithms only integer pel search is used.

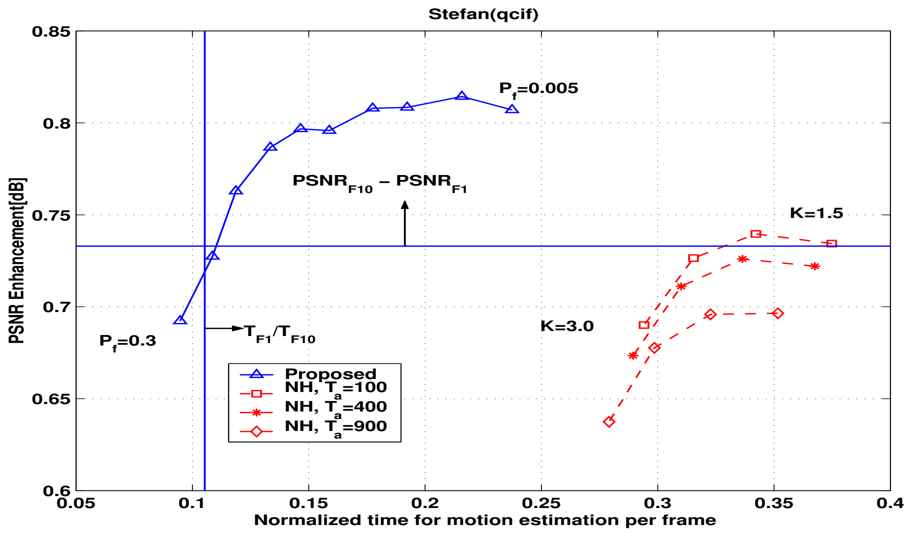


Figure 2.8: Comparison of the proposed algorithm with the hierarchical norm ordered search (NH) algorithm for Stefan QCIF sequence.

As one can see from Figures 2.6 and 2.8, for sequences like Stefan and Foreman which contain large motions, the proposed algorithm shows better performance than WLG algorithm. However, as shown in Figure 2.7, because the Mother & Daughter sequence contains small motions, the proposed algorithm shows similar performance to WLG algorithm in terms of complexity or even slightly worse for the high PSNR gain region. Therefore as expected, the proposed algorithm enhances significantly ME performance when there are large motions. Interesting results for Stefan sequence and Foreman sequence are that we can achieve with the PSNR gain with less computational complexity than $F1$ algorithm. For WLG algorithm, pre-computation of norm values is required, but for the proposed algorithm no pre-processing is required. Also, the proposed algorithm requires less memory as compared to WLG algorithm, because for WLG algorithm the hierarchical structure of norm calculation requires significant memory. The disadvantage of the proposed algorithm is the irregular pattern of memory access due to UNI sub-sampling for HTFM.

2.5 Conclusion

In this chapter, we propose a low complexity ME algorithm with directed search for LTMC. By employing the directed search strategy, the proposed algorithm can enhance ME performance significantly for video sequences which contain large motions. The proposed algorithm reduces the computational complexity greatly by employing HTFM as a fast matching criterion, and by using reduced search area with directed search. Simulation results show that the proposed algorithm can reduce the computational complexity approximately 75%-90% as compared to a full search with fixed MSW location algorithm. The proposed algorithm shows significant performance enhancements for high motion sequences. The main advantage of the proposed algorithm is great reduction of the computational complexity with still good PSNR gain performance. A disadvantage of the proposed algorithm is the irregular memory access pattern due to UNI sub-sampling for HTFM.

Chapter 3

MULTIRESOLUTIONAL MOTION ESTIMATION

3.1 Introduction

In chapter 2, we proposed a directed search strategy with reduced size MSW. The directed search locates MSWs adaptively by using previous SAD results and motion vectors that are computed from previous ME stages, leading to improvements in ME performance. Therefore, the directed search strategy highly depends on the ME results of the previous ME stages. This dependency can degrade the ME performance for LTMC, for example, when an object occludes the area used for matching.

In this chapter, we propose a novel low complexity ME algorithm for LTMC based on a multiresolution search. The main idea of our approach is to use a lower resolution version of a given video sequence to obtain information about the ME

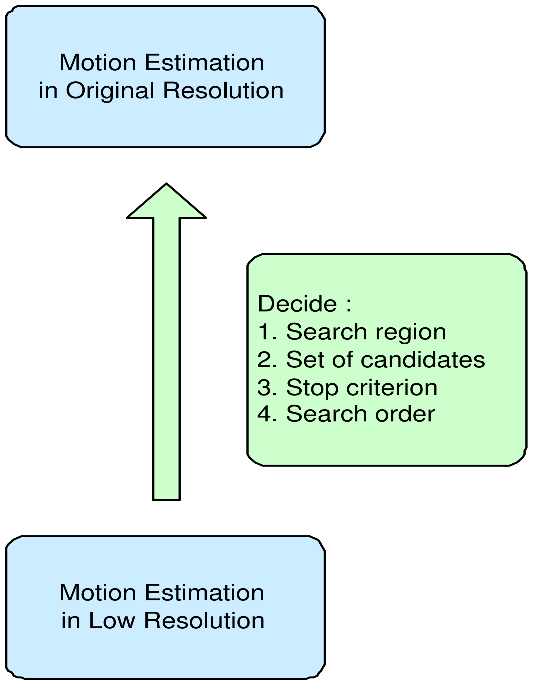


Figure 3.1: Proposed motion estimation using multiresolution search.

on the original resolution sequence. Multiresolution motion estimation (MRME) is well known and has been used for standard single-frame motion compensation [33, 19, 15, 14, 80, 72, 81, 47, 57, 13]. These MRME algorithms employ hierarchies of lower resolution versions of a frame to locate search positions only. However, our proposed MRME algorithm is used to estimate not only search positions but also to select the search range, order of search, and stop criterion, as shown in Figure 3.1. Here we extend our previous work [24, 27] and [61], and introduce several novel techniques to exploit MRME efficiently in the context of LTMC.

Our proposed algorithm also uses adaptive MSW location in a frame buffer as proposed in the previous chapter, and obtains low complexity through a reduced motion search range and HTFM [46]. However, instead of directed search, in this chapter, we use a MRME to locate MSWs in a frame buffer. In addition, instead of using fixed size small MSWs as in directed search, the proposed algorithm adaptively chooses the MSW size or sets of motion candidates based on the scene characteristics determined from low resolution frames. These characteristics include error surface smoothness computed from lower resolution search as well as a ME equivalent function that we will define later. This adaptive search range selection leads to significant reduction in complexity without significant loss in quality. Our approach also incorporates temporal reduction of the motion search range to reduce the complexity further. With this temporal search range reduction, we can detect cases such as object occlusions and scene changes and we can discard "bad" candidates early to reduce computational complexity. With respect to the full-search method, our proposed algorithm has two advantages. First, due to the spatial and temporal search range reduction, our algorithm is faster. Second, due to the multiresolution search, our proposed algorithm can use larger motion search range with limited additional complexity. The schematic diagram of the proposed algorithm is shown in Figure 3.2.

Simulation results show that the proposed algorithm can reduce the computational complexity significantly with slight increase in the frame residual energy compared to a full search and the WLG algorithm [77, 76]. Also, to evaluate the performance of our proposed multiresolution search, the algorithm is implemented

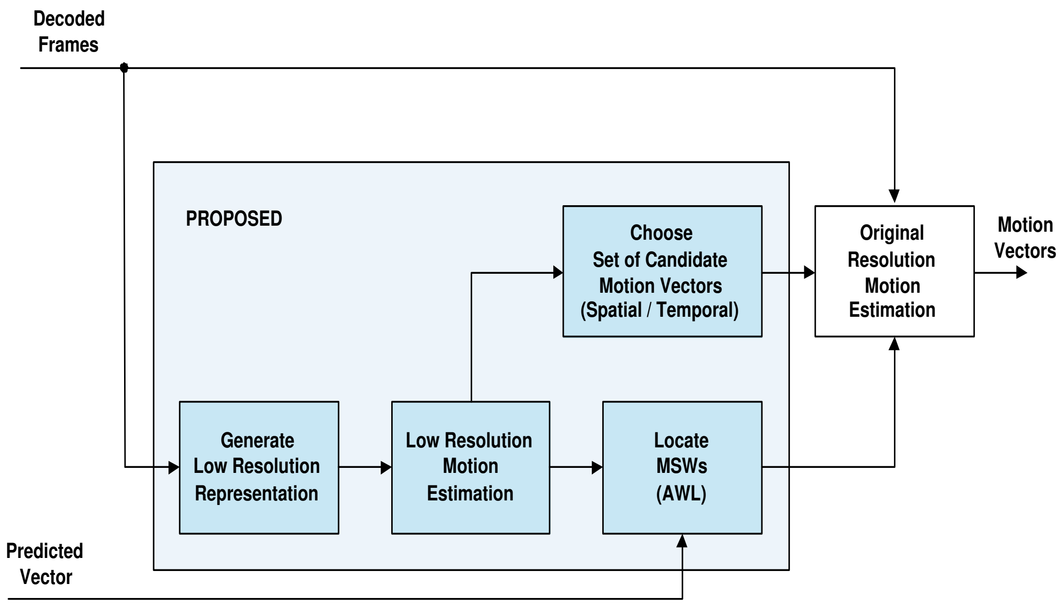

Decoded Frames
PROPOSED
Choose Set of Candidate Motion Vectors (Spatial / Temporal)
Original Resolution Motion Estimation
Motion Vectors
Generate Low Resolution Representation
Low Resolution Motion Estimation
Locate MSWs (AWL)
Predicted Vector

Figure 3.2: Schematic diagram of proposed multiresolution search.

in a H.264/MPEG-4 AVC encoder reference software (JM 5.0c, [1]). Experiments show that the proposed algorithm can speed-up around 70-110 times, with slightly lower RD performance, as compared to a full-search.

3.2 Proposed Low Complexity Algorithm

The computational complexity of full search (FS) for LTMC is very high primarily because the number of candidate motion vectors is very large. As with standard ME, computational complexity reduction can be achieved through the reduction of the number of motion candidates and by using a lower complexity metric. In this section, we propose techniques to reduce the complexity of ME for LTMC. For the proposed low complexity algorithm, the number of motion candidates is reduced by using information from a pre-processing based on a low resolution version of frames. As before the computational complexity for the metric computation is reduced by employing HTFM [46].

3.2.1 Multiresolution based MSW location

The main idea of the adaptive MSW location algorithm using multiresolution search (MRWL) is to use the lower resolution version of a given sequence to acquire coarse information on the ME for LTMC. MRME algorithms achieve fast motion search by using image pyramids which are composed of several low resolution versions of a frame. While MRME algorithms are well known, the main novelty of this work comes from defining search and complexity reduction techniques that are optimized for LTMC.

If we assume that the motion field is sufficiently smooth, the lower resolution frame still contains a coarse version of motion information. Because of this, the MRWL locates MSWs at the positions corresponding to the coarse MVs computed from a lower resolution frame. More specifically, the original frame (level 0 frame) is low-pass-filtered and downsampled to generate a low resolution frame (level 1 frame) and this low resolution frame is low-pass-filtered and downsampled again to produce the lower resolution frame (level 2 frame). Then, MVs are searched in the level 2 low resolution video sequence to get coarse MVs, and in the level 0 frame,

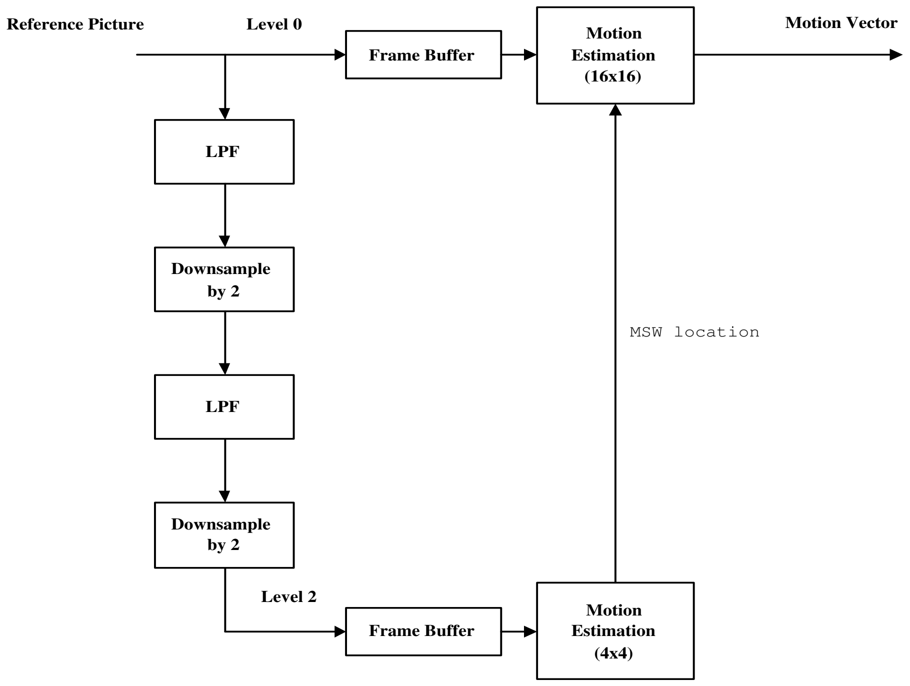


Figure 3.3: Schematic diagram of motion search window location by multiresolution search.

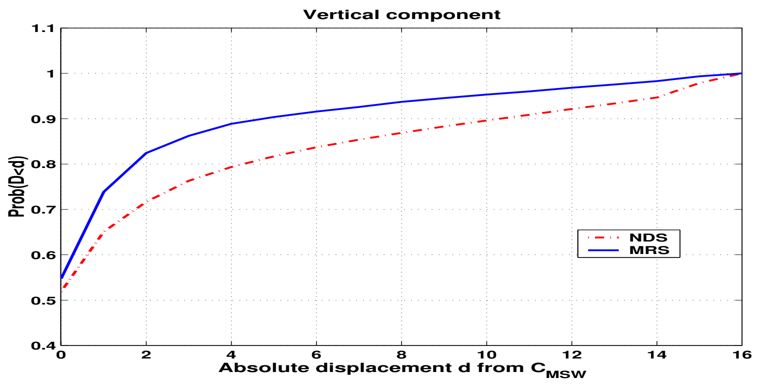
MSWs are located at the positions corresponding to these coarse MVs. A schematic diagram for this is shown in Figure 3.3 In this figure, the horizontal and vertical sizes of the level 2 frame are 1/4 of those of the level 0 frame. ME in the level 2 frame is performed on blocks of 4×4 pixels instead of blocks of 16×16 pixels as in the level 0 frame. The computational complexity of low-pass-filtering and downsampling to produce low resolution frames and the ME on the level 2 frame is small compared with the computational complexity of the ME for LTMC. Both filtering and subsampling are separable, and the lowpass filter used in the proposed algorithm is an average filter, $h[n] = (\delta[n] + \delta[n-1])/2$. The main advantage of the MRWL is that we can improve ME performance by locating MSWs adaptively in a frame buffer for LTMC.

3.2.2 Spatial reduction of motion search range

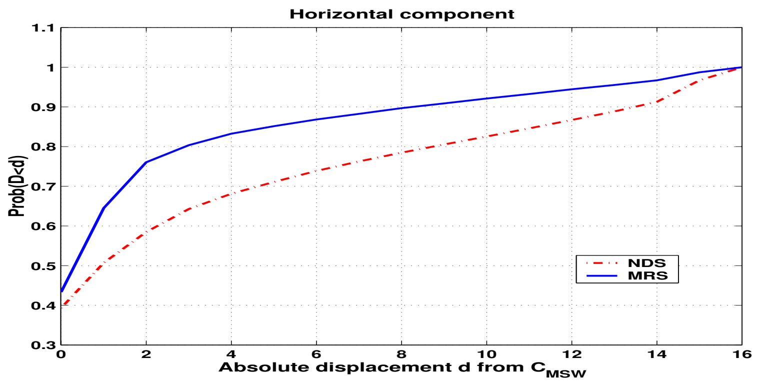
In this section we propose two algorithm to lower ME complexity by reducing spatial search range. The first algorithm makes use of estimates of the error surface smoothness, while the second employs a probability model derived from observed characteristics of a ME.

3.2.2.1 Search space selection based on error surface smoothness

In Figure 3.4, it is shown that the best motion blocks are located near the center of MSWs obtained using the MRS-AWL. In this figure, $Prob\{D \leq d\}$ represents the cumulative distribution function of the absolute distance D (vertical or horizontal) between the best match and the center of the MSW in a reference frame. From this figure one can see that if MRS-AWL is used, almost 95% of the best matching blocks have absolute displacements less than 12, which correspond to MSWs whose size is 24×24. Due to this observation, we can reduce the spatial motion search range for the proposed algorithm as in [26]. This reduction leads to significant computational complexity savings, at the cost of a slight increase in residual frame energy.



(a)



(b)

Figure 3.4: $Prob\{D \leq d\}$ which represents the cumulative distribution function of the absolute distance D between the best match and the center of the MSW in a reference frame. (a) vertical component and (b) horizontal component of the absolute displacement D.Seven QCIF test sequences are used, all the test sequences contain 100 frames.

From the lower resolution sequence, we can obtain additional information to reduce the computational complexity further. For example, consider the error surface computed from level 2 ME. We observed that on this error surface, if the best level 2 motion vector corresponds to a much smaller error value than neighboring level 2 motion vectors, then it is highly likely that the best matching block in this reference frame at level 0 will be located closer to the MSW center at level 0. That

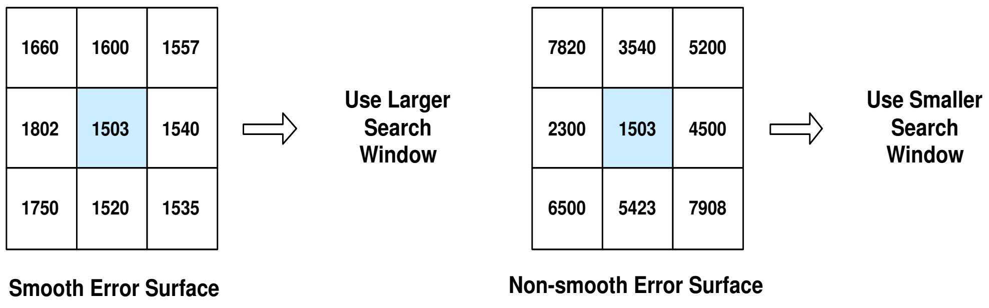


Figure 3.5: Examples of smooth error surface and non-smooth error surface.

is, because the best vector in level 2 is much better than its neighbors, we can assume that the same will be true at level 0 thus enabling us to use a smaller MSW to reduce complexity. An example of this is shown in Figure 3.5.

Let us assume that the best motion vector in a level 2 reference frame for a given macro-block is (mv_r, mv_c), and the corresponding minimum error at level 2 is ${SAD^*}_{L2,local}$. Also let us assume that $\overline{SAD}_{L2,local}$ is the average of the error values corresponding to the neighboring motion vectors, $(mv_r + k_r, mv_c + k_c)$ where $k_r, k_c \in \{-2, -1, 1, 2\}$. Then, if ${SAD^*}_{L2,local}/\overline{SAD}_{L2,local}$ is smaller than a threshold T_S, it is likely that the coarse motion vector will be a good candidate motion vector at level 0, and thus we can reduce the MSW size for the given macro-block at level 0.For the proposed algorithm, we experimentally choose $T_S = 0.5$. It is observed that performance is not sensitive to around 10% variation of T_S. We use 16×16 MSW if ${SAD^*}_{L2,local}/\overline{SAD}_{L2,local} < T_S$, otherwise use 24×24 MSW. The effect of this further reduction of the spatial motion search range is significant computational reduction at the cost of a slight increase in residual frame energy.

3.2.2.2 Search space selection based on a probability model

Some parameters in the approach introduced in Section 3.2.2.1 are chosen experimentally (e.g., T_s) and if it is unclear how this selection would be made adaptive to scene characteristics. While the algorithm provides gains, performance can be improved by devising a search space reduction algorithm that can be adaptive to varying scene characteristics. In this section, we propose an adaptive search space reduction algorithm, based on a probability model.

When the low resolution version of the sequence does not capture enough scene characteristics, the low resolution search may center MSW at local minimum points, leading to degraded ME performance. This can happen when blocks contain large portion of high frequency energy, and when the error surface in the original resolution is not smooth. Therefore, we would like to introduce a decision rule which takes into account how likely it is for the low resolution search to produce a good candidate. This decision rule is based on the characteristics of the current block, as well as the error surface information found from the low resolution search.

To design our decision rule we consider the equivalent transfer function $T_{equi}(\Lambda)$ proposed in [32] to analyze the efficiency of ME. Denote $S_{ee}(\Lambda)$ and $S_{ss}(\Lambda)$ the power spectral density of the residual signal and the input signal respectively. Then [32] proposes to following relationship:

$$S_{ee}(\Lambda) = T_{equi}(\Lambda) \cdot S_{ss}(\Lambda) + \Theta, \tag{3.1}$$

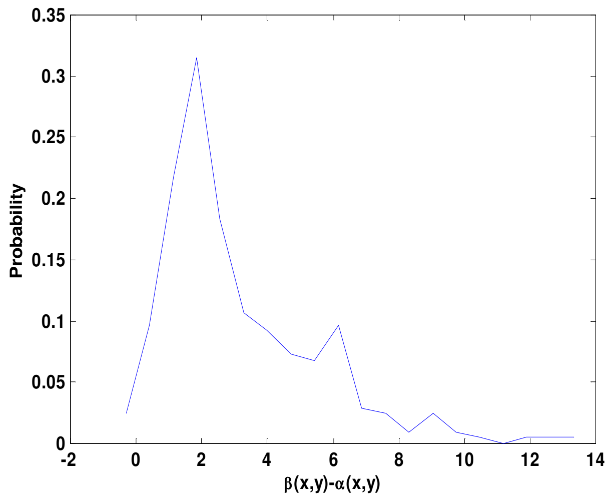


Figure 3.6: $\beta(x,y) - \alpha(x,y)$ for good candidates. The experiment is done for several typical QCIF sequences.

where Λ represents the spatial frequency and Θ is a noise term. The main observation we use is that $T_{equi}(\Lambda)$ shows high-pass characteristics. While this relation is derived under ideal conditions such as a Gaussian image model, mean square error criterion, and operation at rate-distortion bound, we have observed the high pass characteristic in real situations as well. The following notation is used for the algorithm design:

- x: current block.
- $y(i)$: candidate block whose index is i
- $D_0(x,y)$ and $D_2(x,y)$: SAD in the original resolution and the low resolution (level 2), respectively.

- $S_{SS}(x)$: energy of a given block x.
- ${S^L}_{SS}(x)$: energy contained in the low frequency coefficients (LL in Haar decomposition) of a given block x.
- ${S^H}_{SS}(x) = S_{SS}(x) - {S^L}_{SS}(x)$.
- $\alpha(x,y) \triangleq \frac{D_2(x,y)}{\sqrt{{S^L}_{SS}(x)}}$
- $\beta(x,y) \triangleq \frac{(D_0(x,y)-D_2(x,y))}{\sqrt{{S^H}_{SS}(x)}}$

From equation (3.1), we approximate

$$\begin{aligned} T_{equi}(\Lambda) = & \ [T_{equi}(\Lambda_{low}); T_{equi}(\Lambda_{high})] \\ \approx & = \frac{S_{ee}(\Lambda)}{S_{ss}(\Lambda)}. \end{aligned} \tag{3.2}$$

By approximating $S_{ee}(\Lambda_{low})$ and $S_{ee}(\Lambda_{high})$ using $D_2(x,y)^2$ and $(D_0(x,y)-D_2(x,y))^2$ respectively, $\alpha(x,y)^2$ and $\beta(x,y)^2$ can be seen as approximations of the equivalent transfer characteristic for low and high frequencies, respectively. We observed that this approximation is simple but good in capturing the high-pass characteristic of $T_{equi}(\Lambda)$. Experimentally, we observed that $\beta(x,y)-\epsilon\cdot\alpha(x,y), 1 \leq \epsilon \leq 2$ tends to be non-negative, which can be interpreted as a high-pass characteristic of $T_{equi}(\Lambda)$ for a ME process. As an example, in Figure 3.6, we show a distribution of $\beta(x,y)-\alpha(x,y)$ for relatively good candidates y. For each x, we pick candidates y which satisfy $D_2(x,y) \leq 2 \times D_0(x,y^*)$, where y^* is the best candidate for x, which guarantees that relatively good candidates are chosen. This is needed because our goal is to devise a decision rule for relatively good candidates. The experiment is performed using several typical QCIF sequences (100 frames taken from each sequence). As one can see from this figure, for most of the cases, $\beta(x,y) - \alpha(x,y) \geq 0$. We design a search algorithm based on this high-pass characteristic.

Now, let us derive a decision rule based on this observation. From the low resolution search, we know $D_2(x, y_G(i))$ for each $y_G(i)$, where $y_G(i)$ (G represents 4×4 grid) is the down sampled search points located at 4 pel $\times$ 4 pel grid points. Then, in the refinement search, we need to decide for each $y_G(i)$ if we will continue or stop the refinement search. Using the definition of $\alpha(x,y)$ and $\beta(x,y)$, $D_0(x, {y^*}_G(i))$ can be expressed as

$$D_0(x, {y^*}_G(i)) = \alpha(x, {y^*}_G(i)) \cdot \sqrt{{S^L}_{SS}(x)} + \beta(x, {y^*}_G(i)) \cdot \sqrt{{S^H}_{SS}(x)}. \qquad (3.3)$$

Let us define ${y^*}_G(i)$ as the best candidate (in the original resolution) found in the refinement search for each $y_G(i)$. If ${y^*}_G(i)$ is a better candidate than the best found so far, then it should satisfy $D_0(x, {y^*}_G(i)) \leq D_{0,bsf}$, where $D_{0,bsf}$ is the best distortion value found so far in the original resolution search. Using equation (3.3), this condition can be expressed as follows.

$$\alpha(x, {y^*}_G(i)) \cdot \sqrt{{S^L}_{SS}(x)} + \beta(x, {y^*}_G(i)) \cdot \sqrt{{S^H}_{SS}(x)} \leq D_{0,bsf}. \qquad (3.4)$$

By arranging this, the the following should be satisfied for ${y^*}_G(i)$ to be the best candidate.

$$\beta(x, {y^*}_G(i)) \leq \frac{D_{0,bsf} - \alpha(x, {y^*}_G(i)) \cdot \sqrt{{S^L}_{SS}(x)}}{\sqrt{{S^H}_{SS}(x)}}. \tag{3.5}$$

Because $\beta(x, {y^*}_G(i))$ is lower bounded by $\epsilon \cdot \alpha(x, {y^*}_G(i))$, we can write the range of $\beta(x, y)$ as follows.

$$\epsilon \cdot \alpha(x, {y^*}_G(i)) \leq \beta(x, {y^*}_G(i)) \leq \frac{D_{0,bsf} - \alpha(x, {y^*}_G(i)) \cdot \sqrt{{S^L}_{SS}(x)}}{\sqrt{{S^H}_{SS}(x)}}. \tag{3.6}$$

From this, if

$$\epsilon \cdot \alpha(x, {y^*}_G(i)) \geq \frac{D_{0,bsf} - \alpha(x, {y^*}_G(i)) \cdot \sqrt{{S^L}_{SS}(x)}}{\sqrt{{S^H}_{SS}(x)}}, \tag{3.7}$$

then, it is unlikely that a better candidate be found. By the definition of α, the condition (3.7) can be rewritten as follows.

$$D_2(x, {y^*}_G(i)) \geq \frac{\sqrt{{S^L}_{SS}(x)}}{\sqrt{{S^L}_{SS}(x)} + \epsilon \cdot \sqrt{{S^H}_{SS}(x)}} \cdot D_{0,bsf}. \tag{3.8}$$

Therefore, if (3.8) is likely to be satisfied, then we can safely remove the corresponding search points from the search space.

Because we only know $D_2(x, y_G(i))$ at the down sampled search points, we need to estimate $D_2(x, {y^*}_G(i))$ for a given $D_2(x, y_G(i))$. Due to the wide variations in the relationship between $D_2(x, y_G(i))$ and $D_2(x, {y^*}_G(i))$ for each block, we assume a uniform distribution of $D_2(x, {y^*}_G(i))$ for the given $D_2(x, y_G(i))$ as following.

$$\begin{aligned} P(D_2(x, {y^*}_G(i)|D_2(x, y_G(i))) = & \ \frac{1}{D_2(x, y_G(i))} \quad (3.9) \\ D_2(x, {y^*}_G(i)) \in & \ [0, D_2(x, y_G(i))]. \end{aligned}$$

Using this assumption and the condition of equation(3.8), the probability of false alarm $P_f(x, y_G(i))$, which is the probability of not finding better candidate in the refinement search for $y_G(i)$ can be derived and modeled as follows:

$$\begin{aligned} & P_f(x, y_G(i)) \\ = & \ P(D_2(x, {y^*}_G(i)) \geq \frac{\sqrt{{S^L}_{SS}(x)}}{\sqrt{{S^L}_{SS}(x)} + \epsilon \cdot \sqrt{{S^H}_{SS}(x)}} \cdot D_{0,bsf} | \ D_2(x, y_G(i))) \\ = & \ max(0, \ \frac{1}{D_2(x, y_G(i))} \cdot (D_2(x, y_G(i)) - \frac{\sqrt{{S^L}_{SS}(x)}}{\sqrt{{S^L}_{SS}(x)} + \epsilon \cdot \sqrt{{S^H}_{SS}(x)}} \cdot D_{0,bsf})) \\ = & \ max(0, \ 1 - \frac{\sqrt{{S^L}_{SS}(x)}}{\sqrt{{S^L}_{SS}(x)} + \epsilon \cdot \sqrt{{S^H}_{SS}(x)}} \cdot \frac{D_{0,bsf}}{D_2(x, y_G(i))}). \quad (3.10) \end{aligned}$$

Our proposed decision mechanism will only use the search points which satisfy $P_f(x, y_G(i)) \leq Th_F$, where Th_F is a probability threshold for false alarm. As Th_F decreases, more candidates are discarded from the search space, and performance degradation may occur. The trade-off between performance and speed can be controlled by adjusting Th_F.

A key parameter in the decision rule is $\sqrt{\frac{S^H{}_{SS(x)}}{S^L{}_{SS(x)}}}$. When this parameter is very small, then usually the error surface tends to be quite smooth, which means we can in general trust the decision made based on low resolution search. However, when this parameter is large, it is likely that the low resolution search does not capture enough block characteristics. Therefore, we need to search more candidates for this case. However, as one can see from equation (3.10), the proposed decision mechanism tends to use more candidates as the key parameter becomes smaller, and vice versa. Therefore, to address this, we modify the decision rule as follows.

$$\textit{Only refine best candidates from level 2 search if } \sqrt{\frac{S^H{}_{SS}(x)}{S^L{}_{SS}(x)}} \leq c_1$$

$$\textit{Otherwise,}$$

$$\textit{Search if} \quad P_f(x, y_G(i)) \leq \gamma \cdot Th_F, with$$

$$\gamma = \begin{cases} \gamma_1\ (>\ 1), & \text{if} \sqrt{\frac{S^H{}_{SS(x)}}{S^L{}_{SS(x)}}} \geq c_2 \\ \gamma_2\ (<\ 1), & \text{if} \sqrt{\frac{S^H{}_{SS(x)}}{S^L{}_{SS(x)}}} \leq c_3 \\ 1, & \textit{otherwise.} \end{cases} \tag{3.11}$$

More specifically, we force the decision algorithm to use less number of candidates when the key parameter is small enough (smooth error surfaces), and we allow more candidates to be searched when the key parameter is large enough, otherwise we use the decision algorithm proposed in equation (3.10).

Using this decision algorithm, we show the performance of the proposed algorithm in Table 3.1. For these results, we used $\gamma_1 = 1.2, \gamma_2 = 0.1$ and $c_1 = 0.05, c_2 = 0.08, c_3 = 0.01$, which were selected experimentally. It is observed that performance does not change much by varying the parameter values by around 10%. As one can see from these results, the probability model based search space reduction proposed in this Section can reduce the computational complexity significantly with minor degradation in performance, as compared to our earlier proposed algorithm in Section 3.2.2.1.

3.2.3 Temporal reduction of motion search range

Computational complexity can be further reduced by temporal reduction of the motion search range. From the level 2 ME for a given macro-block, we know the best motion vector corresponding to ${SAD^k}_{L2,best}$ for the reference frame whose index is k. The main idea of the temporal reduction of the motion search range is that if ${SAD^{k'}}_{L2,best}$ is much larger than the other ${SAD^k}_{L2,best}$ values obtained with other reference frames for a given macro-block, we can safely exclude the frame k' from the set of reference frames for a given macro-block. Let us define $\overline{SAD}_{L2}$ and σ_{L2} as the average and the standard deviation of ${SAD^k}_{L2,best}$, where k is in the set of reference frame indices for a given macro-block. Let us normalize ${SAD^k}_{L2,best}$ for each k in the set of the frame indices for a given macro-block, $Z = ({SAD^k}_{L2,best} - \overline{SAD}_{L2})/\sigma_{L2}$.

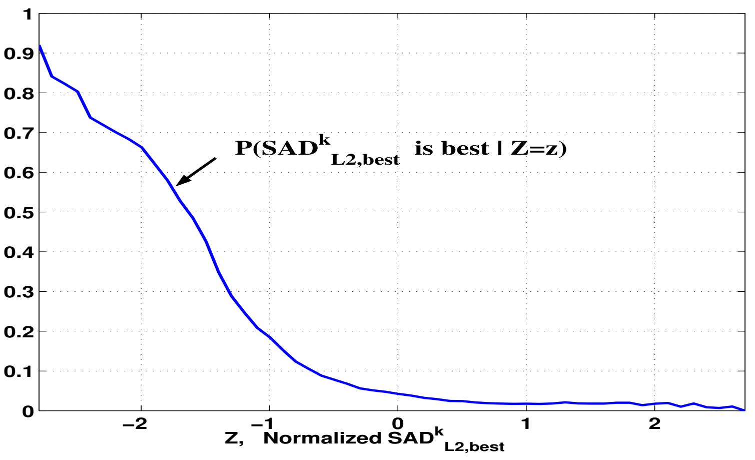


Figure 3.7: $Prob\{SAD^k{}_{L2,best}$ is best$|Z = z\}$. For this simulation, seven QCIF test sequences are used, all the test sequences contain 100 frames.

An experimental conditional probability $Prob\{SAD^k{}_{L2,best}$ is best $\mid Z = z\}$ is shown in Figure 3.7. From this figure, one can see that if Z, is greater than 1, then the probability for $SAD^k{}_{L2,best}$ to be the best matching block for a given macro-block is very small. Therefore we can safely exclude the reference frame where Z is greater than 1 from the set of reference frames for a given macro-block.

3.2.4 Complexity reduction by employing HTFM as a metric

The computational complexity also can be reduced by employing a metric with low computational complexity. For the proposed algorithm, we use HTFM as the fast matching criterion of the ME for LTMC. HTFM can reduce the computational complexity greatly, by allowing an early termination of SAD calculation based on

Table 3.1: Performance of the proposed algorithm. For FWL, 32 × 32 MSW is used. For MRS algorithms, level 2-MSW is 16 × 16. All ME algorithms employ DTFM except $MRS_{S2,H}$ and $MRS_{S2,T,H}$ and spiral ordered FS in the given MSW. The length of the frame buffer is 10. For each sequence, the first 300 frames are used with frame skip parameter 2. PSNR represents the residual frame energy in dB scale.

	Stefan		*Foreman*		*Mother*	
	T [s]	PSNR [dB]	T [s]	PSNR [dB]	T [s]	PSNR [dB]
FWL	9.56	21.87	5.95	29.75	6.15	34.68
DS	9.47	22.07	6.49	29.81	6.24	34.70
MRS	9.26	22.46	6.52	30.10	6.38	34.69
MRS_{S1}	5.75	22.39	4.17	30.06	4.14	34.68
MRS_{S2}	4.36	22.37	2.97	30.02	2.63	34.66
MRS_T	8.09	22.43	5.69	30.03	5.52	34.63
$MRS_{S1,T}$	5.03	22.36	3.65	30.00	3.58	34.62
$MRS_{S2,T}$	3.77	22.34	2.61	29.96	2.31	34.60
$MRS_{S2,H}$	1.42	22.30	1.12	30.04	1.19	34.61
$MRS_{S2,T,H}$	1.27	22.28	1.02	29.99	1.07	34.55
MRS_P	0.91	22.46	0.21	30.04	0.35	34.69

the likelihood (probability of false alarm, P_f) [46]. As the block sub-sampling method for HTFM, UNI sub-sampling is used for the proposed algorithm, and the histogram parameter estimation is performed at every 10 frames in the proposed algorithm. Simulation results show that by employing HTFM, we can significantly reduce the computational complexity at the cost of slight degradation in PSNR gain.

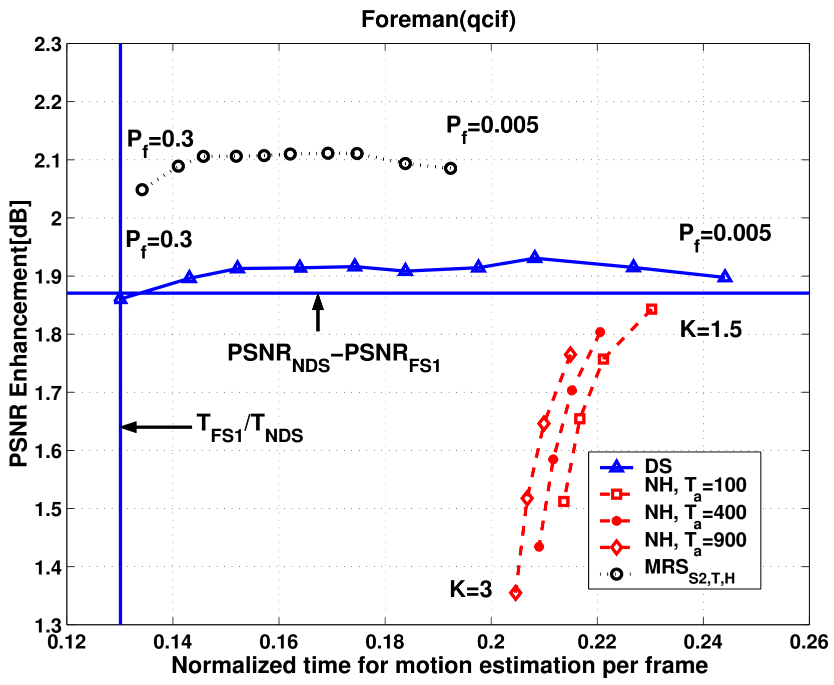


Figure 3.8: Comparison of the performance for various ME algorithms for LTMC using *Foreman* sequence. Each computational time is normalized by T_{NDS}, where T_{NDS} is the computational complexity of the algorithm which employs FWL and DTFM. PSNR is average energy of residual frame in dB scale.

3.2.5 Experiments

In this section, performance of the various ME algorithms proposed are shown in Figures 3.8 - 3.10 and Table 3.1. The simulations are performed using an Intel PentiumIII 750 MHz PC. In Table 3.1, each row represents one of the following algorithms:

- **FWL**: fixed MSW location with full search
- **DS**: the directed search in Chapter 2
- **MRS**: MRS-AWL with full search

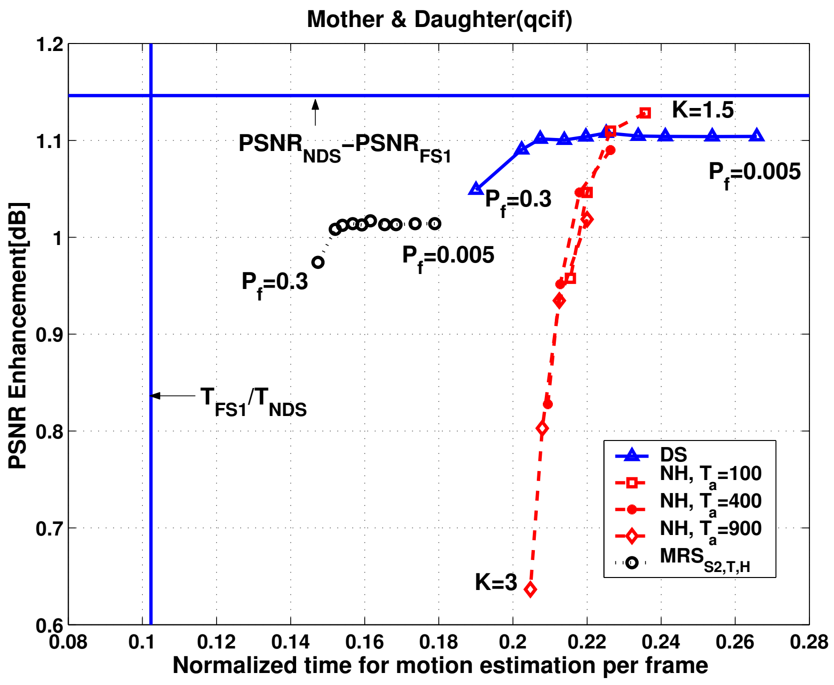


Figure 3.9: Comparison of the performance for various ME algorithms for LTMC using *Mother & Daughter* sequence. Each computational time is normalized by T_{NDS}, where T_{NDS} is the computational complexity of the algorithm which employs FWL and DTFM. PSNR is average energy of residual frame in dB scale.

- $\mathbf{MRS_{S1}}$: MRS-AWL with 24×24 level 0-MSW with full search
- $\mathbf{MRS_{S2}}$: MRS_{S1} with additional complexity reduction using smoothness information
- $\mathbf{MRS_{T}}$: MRS-AWL with the temporal search range reduction
- $\mathbf{MRS_{S1,T}}$: MRS_{S1} with the temporal search range reduction
- $\mathbf{MRS_{S2,T}}$: MRS_{S2} with the temporal search range reduction
- $\mathbf{MRS_{S2,H}}$: MRS_{S2} with HTFM ($P_f = 0.1$)

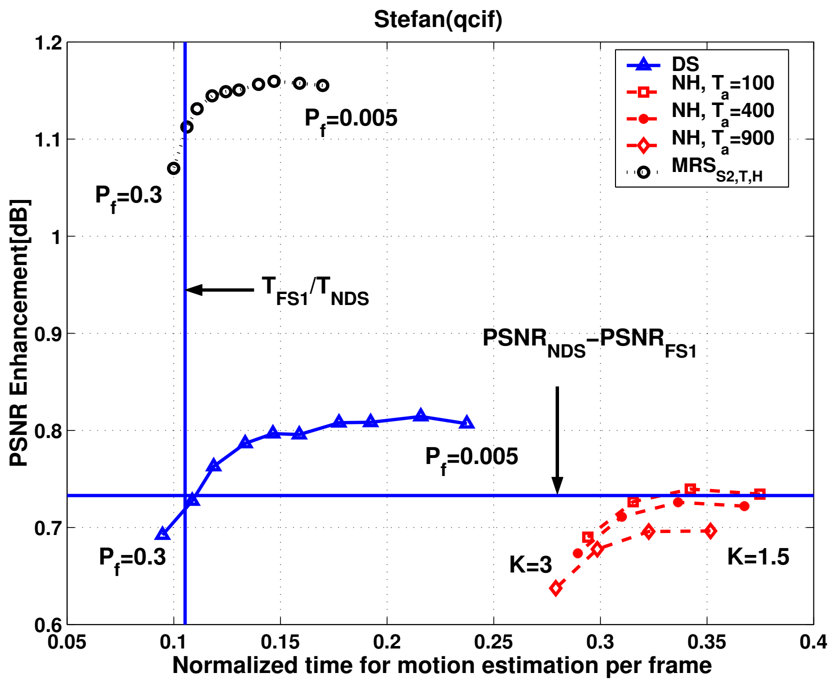


Figure 3.10: Comparison of the performance for various ME algorithms for LTMC using *Stefan* sequence. Each computational time is normalized by T_{NDS}, where T_{NDS} is the computational complexity of the algorithm which employs FWL and DTFM. PSNR is average energy of residual frame in dB scale.

- $\mathbf{MRS_{S2,T,H}}$: $MRS_{S2,T}$ with HTFM ($P_f = 0.1$)
- $\mathbf{MRS_P}$: Probability based search space reduction proposed in Section 3.2.2.2 with DTFM.

The gain-complexity performance is shown in Figure 3.8 - Figure 3.10. In these figures, each algorithm is as follows: FS1 represents the outward spiral ordered conventional full search which uses just one reference frame, and employs DTFM. NH algorithm is a norm ordered hierarchical search algorithm, which employs SAD as a matching metric, DTFM as a fast matching criterion, 4 hierarchical levels of triangle inequalities, and two fast lossy search methods proposed in [77]. For the NH algorithm, the set of activity thresholds T_a is $\{100, 400, 900\}$, and the set of values for K, which is control parameter to terminate the norm-ordered search early, is $\{1.5, 2.0, 2.5, 3.0\}$. DS represents the directed search algorithm proposed in Chapter 2. $MRS_{S2,T,H}$ uses $P_f = \{0.005, 0.01, 0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3\}$.

As one can see from these results, the MRS-AWL can provide performance gains with greatly reduced computational complexity. The gain achieved for the proposed algorithm outperforms the gain achieved with the directed search strategy proposed in Chapter 2. This is because the multiresolution search can track the real motion better than the directed search. Also, one can see that various low complexity techniques proposed significantly reduce the computational complexity. For most of the cases, the computational complexity of the proposed multiresolution search is lowest compared to the other fast algorithms such as the directed search, and the norm ordered hierarchical search. For high motion sequences like *Stefan* and *Foreman* the proposed algorithm achieves significant gain as compared to the fixed window location schemes. This is because our proposed algorithm can track motion across frames efficiently even if sequences exhibit high motion. For low motion sequences the proposed algorithm shows slight decrease in performance, as compared to the fixed window location schemes. This is because AWL does not provide much gain in performance for the low motion sequences, and our proposed algorithm uses less motion candidates for speed up.

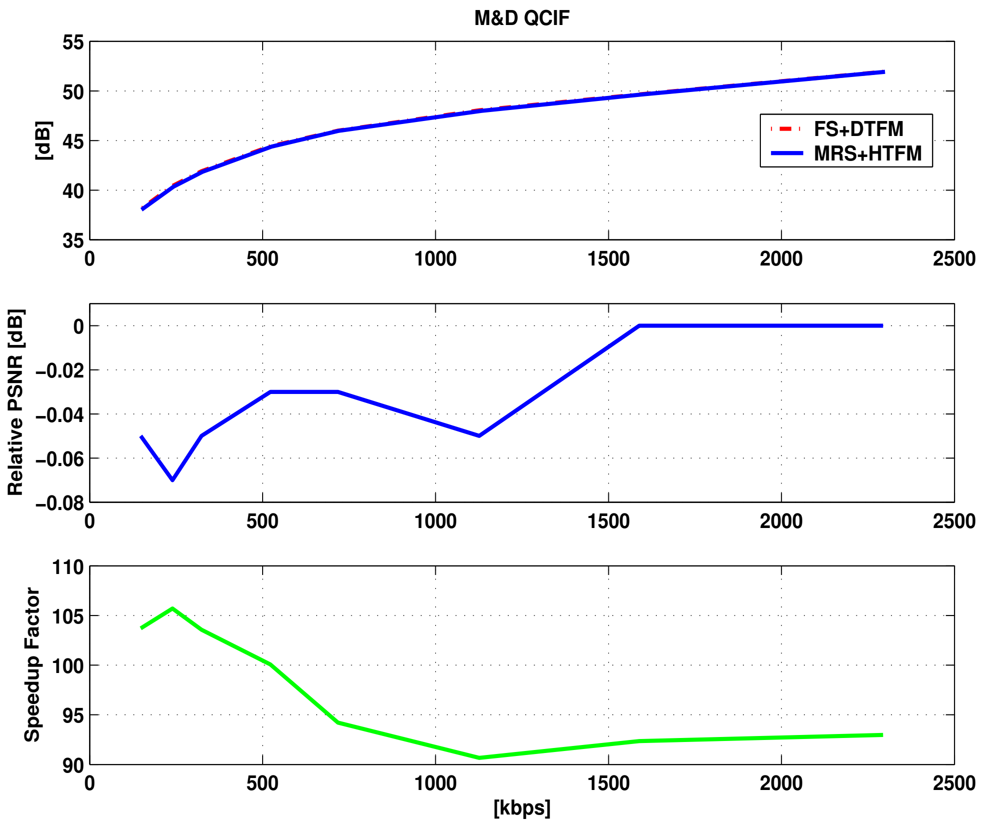


Figure 3.11: RD and complexity performance for it Mother & Daughter QCIF sequence. Each graph from the top shows RD performance, relative PSNR ($PSNR_{FS+DTFM} - PSNR_{MRS+HTFM}$), and the speed-up factor of our proposed algorithm (MRS with HTFM) compared to FS with DTFM, respectively.

3.2.6 Multiresolution Search for H.264 / MPEG-4 AVC

As one can see from Section 3.2.5, search space selection based on our proposed probability model provides better performance than the error surface smoothness technique. Therefore, we employ the probability model along with temporal search space reduction and HTFM in our H.264 / MPEG-4 AVC simulations. The resulting RD-performance and complexity is compared to those for the ME which employs

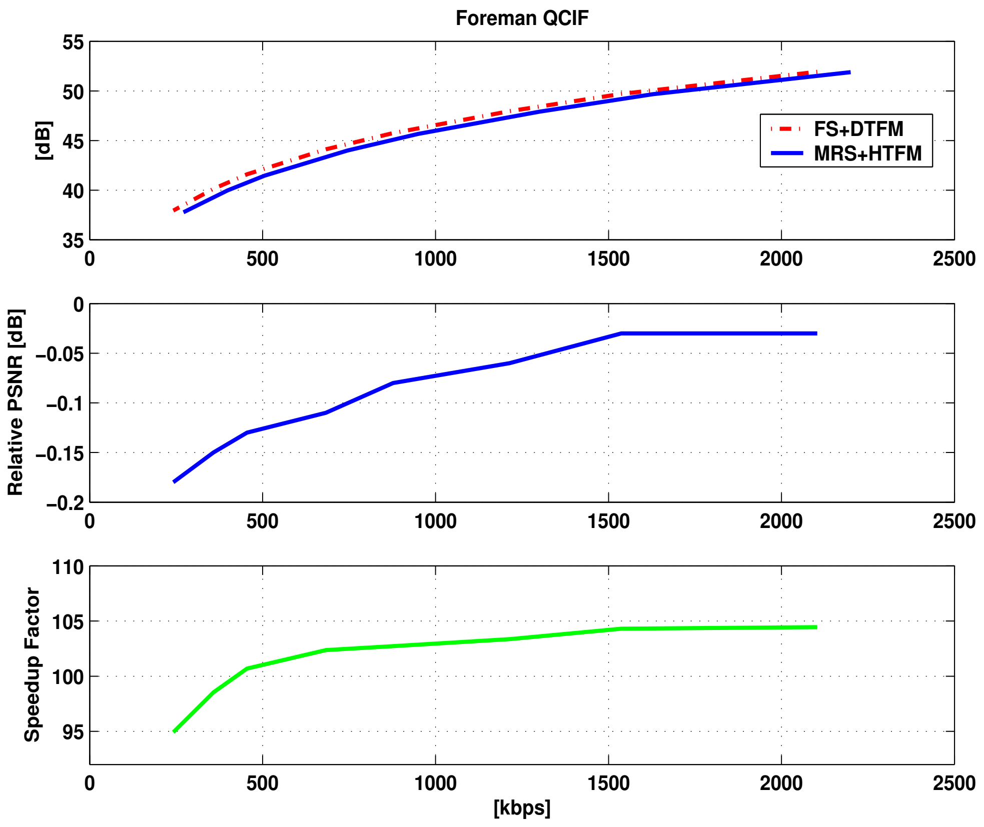


Figure 3.12: RD and complexity performance for it Foreman QCIF sequence. Each graph from the top shows RD performance, relative PSNR ($PSNR_{FS+DTFM} - PSNR_{MRS+HTFM}$), and the speed-up factor of our proposed algorithm (MRS with HTFM) compared to FS with DTFM, respectively.

FS and DTFM. For our proposed algorithm, the parameters are selected to optimize H.264 / MPEG-4 AVC ME for LTMC.

To demonstrate the performance of the proposed fast search algorithm, we implemented our fast search algorithm in H.264 encoder reference software version JM 5.0c [1]. The experiment settings are as follow: a) 300 frames are used for each sequence, b) all the sequences are in QCIF format, c)16 reference frames are used (note that this is the maximum number of reference frames which can

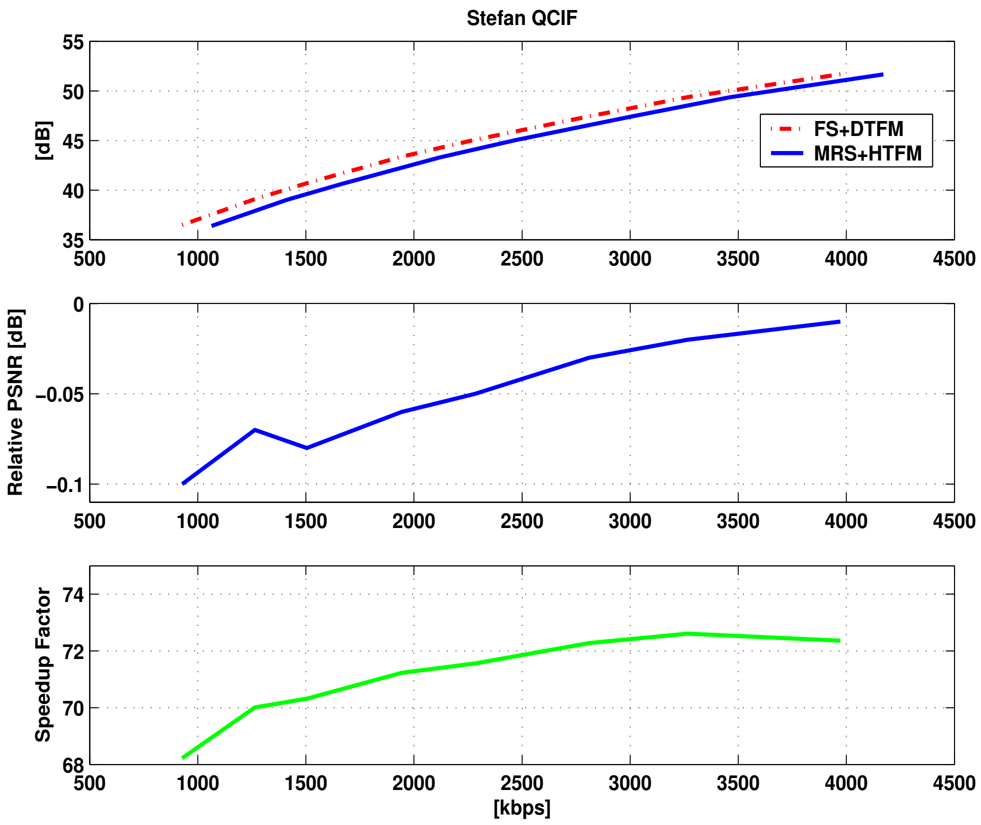


Figure 3.13: RD and complexity performance for it Stefan QCIF sequence. Each graph from the top shows RD performance, relative PSNR ($PSNR_{FS+DTFM} - PSNR_{MRS+HTFM}$), and the speed-up factor of our proposed algorithm (MRS with HTFM) compared to FS with DTFM, respectively.

be used), d) only first frame is coded as Intra frame, e) 16x16 motion partitions are used for ME, f) SearchRange is set as 16. For the proposed algorithm, a) for the search space selection by a probability model, $\{\epsilon, Th_f, c_1, c_2, c_3, \gamma_1, \gamma_2\} = \{2, 0.1, 0.02, 0.05, 0.1, 1.2, 0.1\}$, b) for temporal search space reduction, $Z = 0.25$, and c) for HTFM, $P_f = 0.4$. The parameters are optimized to reduce complexity with slight RD performance degradation. In Figures 3.11-3.13, we compare our proposed algorithm with FS with DTFM, and show RD performance and complexity. In these figures, upper graph shows RD performance, and lower graph shows speed-up factor that is the number of operations for FS with DTFM divided by that of our proposed algorithm. The number of operations includes, addition, absolute difference, division/multiplication and comparison.

As one can see from these results, the proposed algorithm can speed up H.264/AVC LTMC ME process significantly with minor RD performance degradation. The speed-up factors are approximately in the range of 70-110, and the PSNR degradation is very marginal for *Mother & Daughter*, approximately ~ 0.1dB for *Foreman*, and approximately ~ 0.2dB for *Stefan*. The speed-up factor can be further increased if we allow slightly higher RD performance degradation. Our proposed algorithm is scalable in terms of RDC (rate-distortion-complexity) performance. This scalability can be achieved by adjusting thresholds (Th_f and P_f). If we decrease Th_f and increase P_f, then the complexity will be further reduced with a slight increase in RD performance degradation. Conversely, if we increase Th_f and decrease P_f, then we can achieve less RD performance degradation with decreased complexity saving performance.

3.3 Conclusion

In this chapter, we proposed an adaptive fast motion estimation algorithm which estimates scene characteristics based on lower resolution motion estimation. To reduce spatial search range, we propose two fast algorithms. The first algorithm reduces search space based on error surface smoothness determined from lower resolution search. The second algorithm employs a probability model which selects a search space to adapt to varying scene characteristics. To reduce temporal search range, we employ a probability model which determines reference frames which are not likely to provide best matching. To reduce matching complexity, we propose to employ HTFM which allows an early termination of SAD calculation based on the likelihood not to find a better match. As a final proposed algorithm, we propose to employ the probability based spatial/temporal search reduction and HTFM. The proposed algorithm is implemented in the most recent video standard codec H.264/AVC, and the RD and complexity performance is compared with that of ME algorithm which employs FS and DTFM. The simulation shows significant speed up $(70 - 110)$ with slight performance degradation.

Chapter 4

EFFICIENT MEMORY MANAGEMENT CONTROL FOR H.264

4.1 Introduction

LTMC extends the motion search range by using multiple decoded frames as reference frames to improve video encoding efficiency as shown in Chapter 2 and 3. However, due to this multi-frame usage for motion compensation, the memory requirement to store reference frames at the decoder side increases significantly. High capacity, fast access memories are now available at increasingly lower cost. Reference frames, however, have to be fetched in expensive processor cache memory due to large number of data accesses to the reference frames. For mobile telephony application, it is difficult to have enough cache memory for LTMC. Therefore, reducing the memory requirement for the motion compensation is very important specifically for mobile applications.

Memory management control, which aims to reduce the memory requirement for motion compensation, can be based on block-based or frame-based techniques. Block-based memory management schemes form new references by storing only relevant blocks for motion compensation. Frame-based memory management schemes discard reference frames in order to form a smaller reference space.

In [44], a block-based efficient memory management control for LTMC was proposed. In this paper, the author proposed a scheme that selects relevant blocks for motion compensation, and forms a memory space by storing these relevant blocks. Block relevance is decided based on a similarity measure, computed using the same search algorithm at the encoder and the decoder. Therefore, this technique does not require signaling to the decoder side to identify the blocks selected for motion compensation. However, due to the search of relevant blocks at the decoder, this scheme requires a special decoder to be used.

A block based memory management scheme with signaling is possible with no search at the decoder side. However, this scheme needs to signal the utilization of every block for each frame, and therefore, the signaling requires substantial overhead. Instead, signaling in a frame-based memory management scheme requires much lower overhead. For example, for the H.264 video compression standard, utilization of each frame can be signaled efficiently using the memory management control operation (MMCO) [29].

In this chapter, we propose a novel frame-based memory management scheme which results in minimal performance degradation and requires no search at the decoder side. A selection of reference frames are signaled using MMCO as defined in H.264. Therefore, our technique can be used with any standard H.264 decoder. Our goal is to discard the least important reference frames from the frame buffer, i.e., those whose removal will result in the smallest reduction in prediction gain. An optimal solution would require checking all the possible combinations of discarded reference frames to exactly determine the different losses in coding efficiency. Thus this would lead to significant search complexity and long encoding delays. The simplest memory management control scheme is the sliding-window technique which drops the oldest reference frames. Usually this technique works well because temporal correlation of video signals tends to decrease for older reference frames. However, there are a significant number of cases in which the oldest reference frames are important, e.g., sequences with repetitive motions and/or uncovered backgrounds by object motions. For these cases, dropping the oldest reference frames may degrade the prediction performance significantly. For our proposed algorithm, we employ a greedy search which requires low additional complexity and short encoding delay.

4.2 Efficient Memory Management Control For LTMC

The optimal solution of frame-based management control discards the least important reference frames from the frame buffer to form a smaller set of reference frames with the smallest prediction gain degradation. However, to find the optimal solution we would need to check all the combinations of discarded reference frames. This is because for each combination the encoding result for a given frame may be different, and to find the optimal solution we need to encode a frame multiple times; one for each combination of reference frames. Therefore, finding the optimal solution requires high computational complexity. Each combination of reference frames may result in different quality for the current frame being encoded, and this may result in different quality for the future frames until an intra refresh frame is coded. Thus, for the optimal solution we would need to determine coding performance for the future frames till a refresh frame as well. Therefore, finding the optimal solution requires very long encoding delay. Due to the high computational complexity and the long encoding delay of the optimal solution, we propose to use a greedy search as a suboptimal solution. To describe our proposed algorithm, let us define some notation first:

- X_n : reconstructed frame with index n.
- Y_m : original frame with index m.

- ${B_v}^m(M) = \{X_{m-M}, \ldots, X_{m-1}\}$: virtual frame buffer of length M for the motion-estimated prediction of Y_m. The virtual frame buffer contains all possible reference frames for a given frame.

- ${B_r}^m(L)$: real frame buffer of length L for the motion estimated prediction of Y_m. The real frame buffer contains the reduced number of reference frames which is decided by a memory management control.

- $D_i(X_n, Y_m)$: distortion between the i_{th} block of Y_m and the best match in X_n.

The proposed algorithm is composed of three stages, namely, data generation, frame-selection, and encoding. Let us assume that M is the maximum number of reference frames that can be used (size of virtual frame buffer), and L is the number of reference frames to be used by our proposed scheme. For a current frame Y_m, our proposed algorithm first forms a set of available reference frames $S_r(m) = \{X_{m-1}\} \cup ({B_r}^{m-1}(L) \cap {B_v}^m(M))$ which is a set of the previously encoded frame X_{m-1} and the subset of ${B_r}^{m-1}(L)$ from which Y_m can be predicted. Note that $|S_r(m)| \leq L+1$, where $| \cdot |$ is the number of elements (in this case, the number of reference frames). Therefore, to form ${B_r}^m(L)$ out of $S_r(m)$, our proposed algorithm discards discards a reference frame from $S_r(m)$, whenever $|S_r(m)| = L+1$ and does not refer to the discarded reference frame for encoding future frames.

Discarding a reference frame from $S_r(m)$ may affect encoding of future frames including Y_m because this decision would provide different reference frames and different motion candidates. The optimal solution should consider all these effects to form a ${B_r}^m(L)$, which would require a long delay of encoding. As a sub-optimal solution, we propose to consider future frames which are directly affected by discarding a reference frame. For this, we define $S_a(m) = \{Y_m, Y_{m+1}, \ldots, Y_{M+m-1}\}$ as a selection of all the frames which can use at least one frame in $S_r(m)$ as a reference. In our proposed algorithm, rather than basing our frame selection on all future frames, we propose to discard the frame from $S_r(m)$ that is least important in terms of achieving efficient motion compensation of the frames in $S_a(m)$. In this way, we can achieve smaller encoding delay and complexity by reducing the amount of look-ahead encoding required.

4.2.1 Data generation stage

The data generation stage collects motion and distortion data by look-ahead encoding of frames $Y_k \in S_a(m)$. By using these results, in the following frame-selection stage, we select a set of reference frames. If $|S_r(m)| \leq L$, at the data generation stage, we just perform normal encoding of Y_m without look-ahead encoding and frame-selection. Otherwise, we perform look-ahead encoding of frames in $S_a(m)$. The optimal solution searches for the reference frame that provides the least distortion increase in encoding frames $S_a(m)$ by discarding it. However, the look-ahead encoding of frame $Y_k \in S_a(m)$ needs to use X_j, the encoded and reconstructed version of the Y_j, for $m \leq j < k$ that are not available because the corresponding

${B_r}^j(L)$ of Y_j is not decided yet. Therefore, for the optimal solution, we need to encode $Y_k \in S_a(m)$ multiple times for each combination of reference frames, which requires very high computational complexity and long encoding delay. In our proposed algorithm, instead, we use $(S_r(m) \cap {B_v}^k(M)) \cup \{X'_m, \ldots, X'_{k-1}\}$ as references instead of using ${B_r}^k(L)$ for look-ahead encoding of Y_k, where X'_j is defined as the encoded and reconstructed frame by using all the available references in the virtual frame buffer of frame Y_k. Using X'_j instead of X_j may results in suboptimal solution because we do not consider all the combinations of frame-selections. However, in this way, we can avoid long encoding delay and high computational complexity of the optimal solution.

Note that for some $Y_k \in S_a(m)$, the motion estimation has already been performed when making a decision on the previous frames, and the corresponding motion vectors and distortion values have been stored. However, after a decision on ${B_r}^m(L)$ is made, X_m becomes available, and this may be different from X'_m used for the look-ahead encoding of $\{Y_{m+1}, \ldots, \}$, and this difference may results in different motion results from those that are stored. This would require performing motion estimation again for $\{Y_{m+1}, \ldots, \}$ using X_m rather than X'_m. Because the motion estimation process is the most complex processing for video encoding, we propose to use the motion vectors and distortion values stored from look-ahead encoding, and perform the motion estimation only for Y_k for which the motion estimation has not been performed. This may result in some coding inefficiency due to the difference between X_m and X'_m. However, in this way, we can perform the motion estimation just once for each frame Y_k.

After motion estimation, for each block, i, of frame $Y_k \in S_a(m)$, we store the motion vectors $\mathbf{mv}_i(X_j, Y_k)$ and the corresponding distortion values $D_i(X_j, Y_k)$ for each reference frame used, $X_j,\ j \in \{k - M, \ldots, k - 1\}$. As a distortion measure D_i, in this work, we use $SATD + \lambda_{MOTION} \cdot COST$, where $SATD$ is the sum of absolute transformed differences, $COST$ is the bit cost to encode a motion vector and a reference frame index, and λ_{MOTION} is the Lagrangian parameter used to incorporate rate constraint. We choose this metric as the distortion measure because it is used to find the best reference frame in the H.264 reference software [2].

4.2.2 Frame-selection stage

In the second stage, using D_i obtained from the data generation stage, we compute $C(X_j, Y_k)$ which is defined as the total distortion increase for each frame $Y_k \in S_a(m)$ when $X_j \in S_r(m)$ is discarded.

$$C(X_j, Y_k) = \sum_{i \in A} (D_i(X_{j_i}, Y_k) - D_i(X_j, Y_k)), \tag{4.1}$$

where A is the set of block indices for which $D_i(X_j, Y_k)$ is the minimum for the given j. In (4.1), X_{j_i} is the second best reference frame for i_{th} block of Y_k. By using

$C(X_j, Y_k)$, we compute the total distortion increase, $\Gamma(X_j, m)$, of all $Y_k \in S_a(m)$ by discarding X_j from $S_r(m)$. $\Gamma(X_j, m)$ is computed as follows.

$$\Gamma(X_j, m) = \sum_{Y_k \in S_a(m)} C(X_j, Y_k) \tag{4.2}$$

Then, we remove X_{min} from $S_r(m)$ that minimizes $\Gamma(\cdot)$, and we form ${B_r}^m(L)$ as follows.

$$X_{min} = arg \min_{X_j \in S_r(m)} \Gamma(X_j, m) \tag{4.3}$$

After discarding X_{min} from $S_r(m)$, we delete the stored motion vectors and the distortions corresponding to X_{min}.

4.2.3 Encoding stage

By using ${B_r}^m(L)$ decided from the frame-selection stage, we encode the current frame Y_m. We use the motion vectors collected at the data-gathering stage for the encoding. Therefore, we do not need to perform the motion estimation again for this stage. For the encoding, for each i_{th} block of Y_m, we choose the best motion vector $\mathbf{mv}_i^*$ as follows.

$$X_j^* = arg \min_{X_j \in {B_r}^m(L)} D_i(X_j, Y_m) \tag{4.4}$$

$$\mathbf{mv}_i^* = \mathbf{mv}_i(X_j^*, Y_m)$$

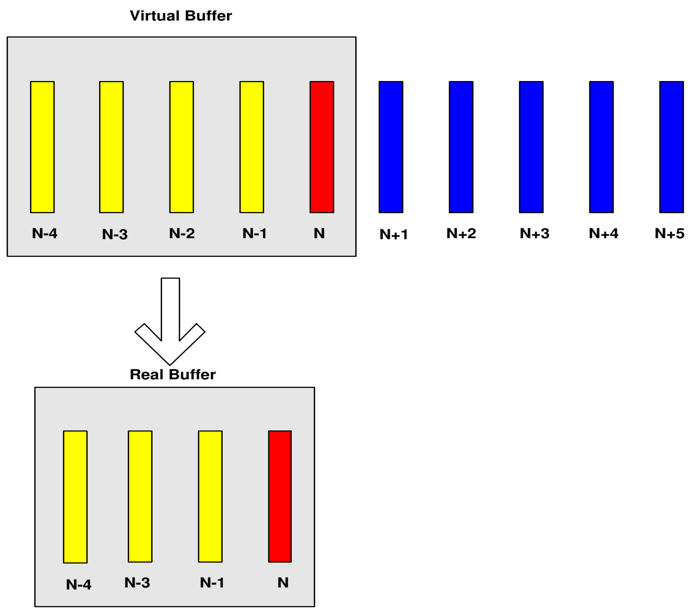


Figure 4.1: Discarding a reference frame from the virtual frame buffer to form a set of references for the real frame buffer.

Using these chosen motion vectors, we encode Y_m, then add the resulting X_m to the frame buffer to be used for encoding the future frames.

4.2.4 Example

Let us describe our proposed algorithm by a simple example in Figure 4.1. Assume that the length of the real frame buffer is $L = 4$, while the length of the virtual frame buffer is $M = 5$. We will encode frame Y_{N+1} using the proposed technique. Now we want to discard a reference frame from $S_r(N+1) = \{X_{N-4}, X_{N-3}, X_{N-2}, X_{N-1}, X_N\}$ as shown in Figure 4.1. First, at the data gathering stage, we perform the motion estimation for $S_a(N + 1) = \{Y_{N+1}, \ldots, Y_{N+5}\}$, and store all the motion vectors

$\mathbf{mv}_j(X_{N-i}, Y_m)$ and the distortions $D_j(X_{N-i}, Y_m)$. Then, at the frame-selection stage, we compute $\Gamma(X_{N-i}, N+1)$ as follows.

$$\Gamma(X_{N-i}, N+1) = \sum_{m=N+1}^{N-i+5} C(X_{N-i}, Y_m), \tag{4.5}$$
$$i \in \{0, 1, 2, 3, 4\}.$$

By comparing all $\Gamma(X_{N-i}, N+1)$, $\forall i \in \{0, 1, 2, 3, 4\}$, we discard the reference frame which provides the smallest $\Gamma(\cdot)$. Let us assume that $\Gamma(X_{N-2}, N+1)$ is smallest, then we discard X_{N-2} from $S_r(N+1)$. The real frame buffer is ${B_r}^{N+1}(4) = \{X_N, X_{N-1}, X_{N-3}, X_{N-4}\}$. Then we delete $D_j(X_{N-2}, Y_{N+1})$ and $\mathbf{mv}_j(X_{N-2}, Y_{N+1})$ from stored motion vectors and distortions.

4.3 Experiments

In this section, we show the experimental results of our proposed memory management control technique. For the experiments, we implement the proposed algorithm into a H.264 reference software, JM8.1a [2] with the baseline profile. For the experiments, we use CIF sequences of length 100. Among 100 frames for each sequence, the first frame is intra-coded, and remaining frames are inter-coded. For motion estimation, we use the fast motion estimation implemented in the reference software and quarter-pel refinement, set the search parameter to 32, and use 16×16 macro-block partitioning. For the experiments, we use a virtual frame buffer of size

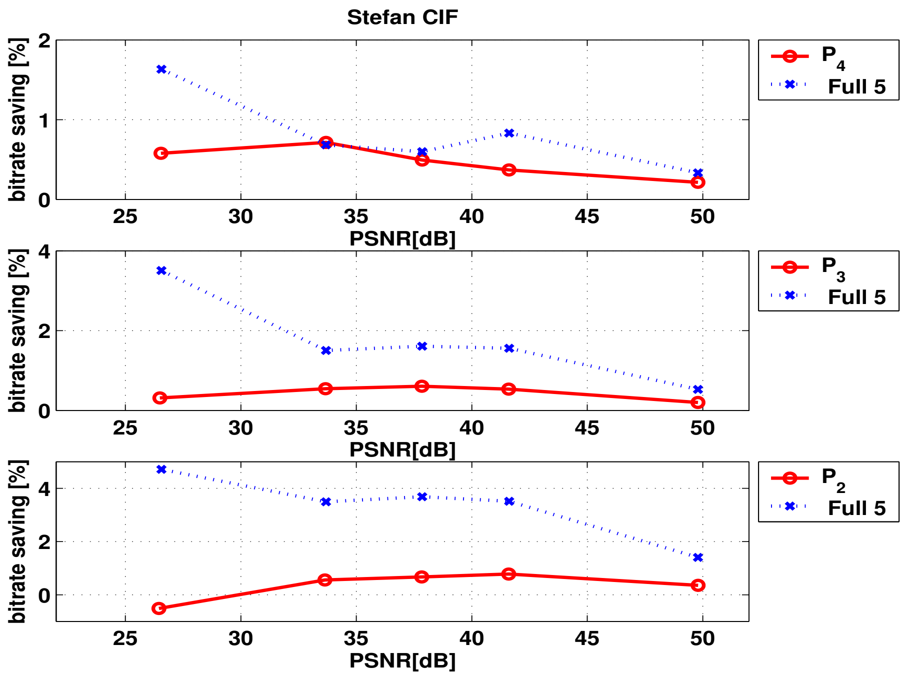


Figure 4.2: Relative rate-distortion performance of the proposed technique with respect to the sliding-window technique, Top: SW4, middle: SW3, and bottom: SW2. **Full 5** uses virtual and real buffers of size 5 frames. The result is for 100 frames of CIF *Stefan* sequence.

5, and vary the size of the real frame buffer. The proposed memory management scheme requires signaling the utilization of the frame buffer at the decoder side. For this, we use the memory management control operation (MMCO) of H.264[29].

The proposed algorithm is compared with the sliding-window memory management scheme which discards the oldest frames. In Figure 4.2 and 4.3, we show the relative rate-distortion (RD) performance of the proposed algorithm with respect to that of the sliding-window technique. In these figures, **SW4**, **SW3**, and **SW2** represents the sliding-window technique which discards the oldest 1, 2, and 3

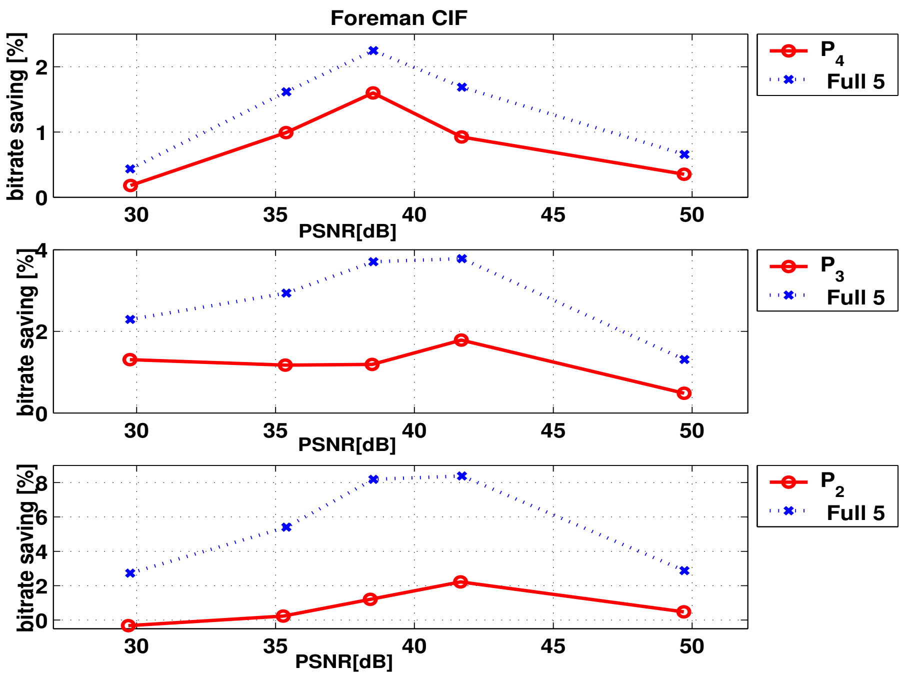


Figure 4.3: Relative rate-distortion performance of the proposed technique with respect to the sliding-window technique, Top: SW4, middle: SW3, and bottom: SW2. **Full 5** uses virtual and real buffers of size 5 frames. The result is for 100 frames of CIF *Foreman* sequence.

reference frames among 5 reference frames, respectively. Also, P_4, P_3, and P_2 represents the proposed technique which utilizes 4, 3, and 2 frames, respectively. As one can see from these results, our proposed algorithm usually achieves better RD performance compared to the sliding-window technique. However, the relative gain of our proposed technique decreases in the lower bit-rate range. This is because as bit-rate decreases, more matches tend to be found from more recent reference frames due to the cost of signaling temporal distance between a current frame and a reference. Therefore, the sliding-window memory management scheme performs reasonably well in this low bit-rate region.

4.4 Conclusion

In this chapter, we propose an efficient memory management control scheme which can reduce the memory requirements for LTMC at a decoder. The proposed technique employs a greedy search to discard the least important reference frames, and shows better prediction gain performance than the sliding-window memory management that discards the oldest reference frames. The proposed technique is implemented with relatively low additional complexity at the encoder, and requires no search at the decoder. The proposed technique requires look-ahead encoding which introduces relatively small amount of encoding delay, and requires a signaling to let the decoder know when to discard reference frames. For a video sequence where many matches come from the older reference frames, our proposed technique shows substantial improvement in coding gains.

Chapter 5

FAULT RESILIENT COMPRESSION [1]

5.1 Introduction

Widespread deployment of multimedia applications is continuing to create a need for highly integrated chips which support various multimedia functionalities. Also, as technologies advance, the dimension of chips tends to decrease, which leads to increases in the effects of manufacturing defects and reductions in yield rate. The manufacturing yield rate (or yield rate) is the percentage of chips which pass manufacturing tests among tested chips. Low yield rates can increase the cost of chips and delay the start of their mass production stage [30].

[1]This chapter is based upon work supported in part by the National Science Foundation under Grant No. 0428940. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

To address these problems, fault tolerance (FT) and defect tolerance (DT) techniques have been proposed. FT techniques provide reliable operations in the presence of faults or errors by utilizing additional error detection and correction circuits. DT techniques enhance the yield rate by using redundancy (spares) and/or defect avoidance techniques in layout [17, 53], routing [73], and circuit design [63, 43], etc. Examples of FT schemes are [39, 22, 65, 62]. Some approaches [39, 22] are based on algorithmic level computations using redundant data. In other cases [65, 62], the algorithm itself is modified to enable fault location and correction. In these systems the goal is to compensate the effect of the faults, so that a system that contains a faulty subsystem behaves *exactly* as a fault-free one.

In [18] a system-level error tolerance (ET) scheme was proposed to increase effective yield by enabling use of faulty circuits which provide acceptable performance degradation. ET is a new design and test paradigm, which takes into consideration whether erroneous outputs of defective circuits produce *acceptable* results. ET classifies a chip as being *acceptable/unacceptable* by estimating the performance degradation due to faults, rather than relying solely on the conventional perfect/imperfect classification. This enables the use of systems that would otherwise have been discarded and increases the effective yield. ET analyzes the system-level effects of faults, and accepts chips if the performance degradation they lead to is within some, application-specific, ranges of acceptability.

In this chapter, we propose fault resilient video compression, an ET scheme specifically designed for digital video compression systems. A common characteristic of all compression standards for digital video (and indeed for images, speech, or audio) is that they rely on lossy compression, that is, the decoded video is not an exact copy of the original. If we view the effect of faults as potential additional distortion suffered by the decoded video, this added distortion will in some cases still lead to an acceptable output. Therefore, multimedia compression systems are good applications for ET techniques. Our ultimate goal is to separate acceptable from unacceptable faults by estimating the fault's impact on the decoded video quality. Therefore, analysis of system behavior under faults is a key to ET. While our focus in this chapter is system-level analysis of ME techniques for acceptable degradation, testing can also be applied to lower level components of a system [41] and to other subsystems within a video encoder [23].

Our proposed schemes are aimed at separating the systems being tested into four classes, namely, (i) fault-free systems, (ii) faulty systems such that a simple programmable algorithmic level compensation can make them error-free (i.e., the output will then be the same as that of a fault-free system), (iii) faulty systems producing acceptable quality degradation, and (iv) faulty systems producing unacceptable quality degradation. Only systems in the last class will be discarded. Traditional FT schemes identify systems in the second class. Thus the main novelty of our approach is to identify systems in the third class, which do not produce an output identical to that of a fault-free system, but can still be used. Also, we provide a novel FT scheme for the second class. To achieve the proposed goal, we first analyze and model system-level effect of faults within the ME process. We then propose test vector generation and testing algorithms for the proposed ET scheme.

5.2 System Level Error Tolerance for Video Compression Systems

Fault effects on various computing components of digital video compression systems vary greatly depending on which components are affected. For example, some faults may may result in the codec generating non-compliant bitstreams which can not be decoded properly. However, it is also possible that some faults may only lead to small degradation in output quality. In this section, we provide a high level analysis of fault effects for each essential video compression system component.

Let us first categorize the various types of system-level faults that arise in typical multimedia compression systems [25]. We consider faults to be *catastrophic* if they prevent the creation (or decoding) of a valid bitstream. Non-catastrophic faults are such that the syntax of the resulting bitstream is valid even though the bitstreams (and therefore the decoded signals) will be different from those produced by a fault-free system given the same input. Acceptable faults are those non-catastrophic

	Catastrophic Error	Error Propagation	Visual Quality Degradation	Coding Efficiency Degradation
Frame Memory(1)	No	No	Yes	Yes
DCT	No	No	Yes	Yes
Q	No	No	Yes	Yes
Inv Q	No	Yes	Yes	Yes
IDCT	No	Yes	Yes	Yes
Frame Memory(2)	No	Yes	Yes	Yes
ME	No	No	Possible	Yes
MC	No	Yes	Yes	Yes
VLC Encoder	Yes	Yes	Yes	Yes

Table 5.1: Fault effect analysis of a MPEG-2 video encoder depicted in Figure 1.1 (a).

faults leading to acceptable degradation. Discrimination between acceptable and unacceptable faults will be achieved by estimating the impact on decoded quality and may depend on the target application. Thus, acceptability criteria are likely to be more strict (i.e., only minimal degradation will be acceptable) for high-end systems where quality is most important. In addition to these categories, it is important to determine which faults can be compensated so as to have no effect on the system output.

Consider as an example an MPEG-2 encoder, which includes among its building blocks motion estimation (ME) / compensation (MC), the discrete cosine transform (DCT), quantization, entropy coding, and various memory buffers [55]. Faults in each of these components have completely different effects on the operation of the complete system [25]. Fault effects for these various components are summarized in Table 1.1, and showed again in Table 5.1 for convenience. It is important to note that many faults affecting different parts of one such system are in fact *non-catastrophic*, although obviously they result in a reduction in coding efficiency and/or visual quality. In fact, only faults that affect the entropy coding block are likely to be catastrophic.

Note that in some cases a relatively well localized fault can have effects that propagate over time and may have a significant impact on decoded quality. For example, a motion compensated video encoder has to include a version of the video decoder, because encoder and decoder need to use the same decoded video frames as predictors. Then, if a fault affects the encoder decoding loop (e.g., there is a fault in the IDCT computation at the encoder) this will lead to errors that can accumulate over time, as the previous frame memory is not the same at encoder and decoder.

5.3 System Level Error Tolerance for Motion Estimation: Analysis

As one can see from Table 5.1, the ME process exhibits significant robustness to faults. Faults at a ME circuit may cause worse rate-distortion (RD) performance, but do not produce catastrophic errors nor error propagation. ME represents a significant percentage of computational complexity and memory bandwidth requirements in video encoding as one can see from an example in Figure 5.1. Therefore, the ME process is a potentially good candidate to which error tolerance scheme can be applied. In this section, we analyze and model effects of faults on ME circuits in order to facilitate the design of error tolerant ME.

5.3.1 Motion estimation

For each non-overlapped block of size $N \times N$ pixels in the current frame (the *reference* macroblock, MB, X), ME seeks to find the best match among all the blocks (*candidate* blocks Y_i) in a search window located in the previous and/or future frames. ME algorithms include a search strategy and matching metric computation. The search strategy (e.g., full search, three-step search [42], and two-dimensional logarithmic search [40]) decides the set of candidate blocks to be tested and the order in which their metrics should be computed. The matching metric (e.g., the sum of absolute differences, SAD, or the sum of square differences, SSD, between a candidate block and the reference block) is computed for successive candidate blocks, until a sufficiently good candidate block has been found, i.e., one whose SAD or SSD with respect to the reference block is sufficiently low. The encoder then subtracts this chosen candidate from the reference block and transmits the resulting difference data (i.e., the prediction residual) to the decoder (as long as the

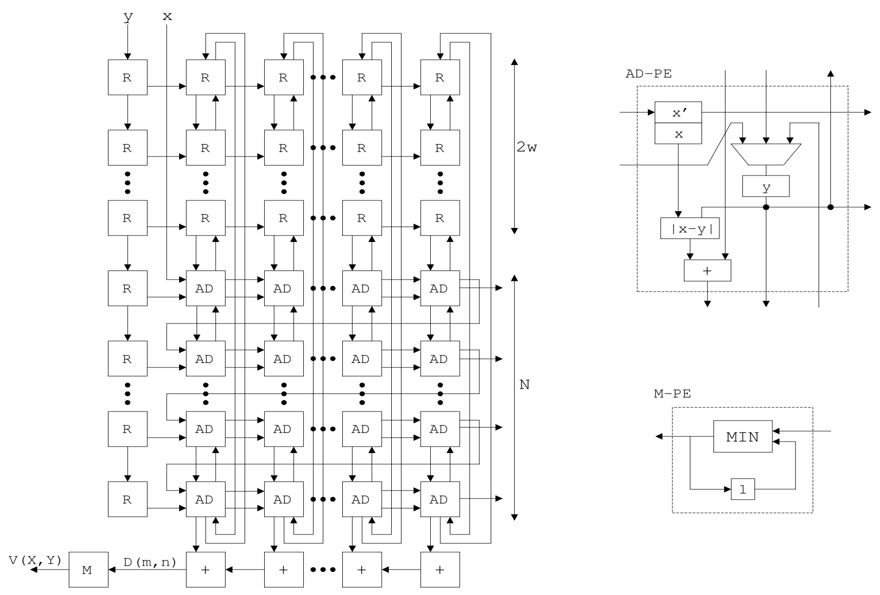


Figure 5.1: Example of ME implementation architecture with a parallel matching process architecture. **AD** block represents an absolute-difference and an add processing element (AD-PE), **M** block represents minimum processing element (M-PE), and **R** block represents memory register [59].

energy in the difference is sufficiently low this leads to better compression efficiency than sending the reference block directly.)

In Figure 5.1, we show an example of a ME implementation architecture [59]. This ME example finds the best match for $N \times N$ block X in $2w \times 2w$ motion search window Y (x and y represents a pixel value of X and Y respectively). In this figure, $V(X, Y)$ represents the resulting motion vector found and $D(m, n)$ represents a distortion of a candidate which corresponds to (m, n) position. Also in this figure, **AD** represents an absolute-difference (the matching metric used is SAD) and an add processing element (AD-PE), **M** represents minimum processing element (M-PE), and **R** represents memory register. AD-PE computes the absolute pixel difference for the current pixels and adds it together with one or more partial SAD values. M-PE compares the SAD value of a current candidate with the best SAD value found so far, and picks the candidate that provides minimum distortion. Memory register **R** stores and feeds pixels of candidate blocks to the matching process. Therefore a search strategy is performed by a logic which selects a candidate and by **R** blocks, while the matching metric computation is mainly performed by AD-PEs.

It is important to note that even if the encoder does not choose the "best" candidate block (e.g., the one having the lowest possible SAD among all the candidates for a given reference block), encoding and decoding are still possible. There is a penalty in compression performance (i.e., more bits are needed to represent the residual signal when a sub-optimal candidate block is chosen) but otherwise the encoder and decoder operate normally. It is also important to note that the exact computed SAD metric is itself of little concern; what is important is the relative ranking of the candidate blocks because the ME process seeks to find the best matches among candidate blocks. Thus, *certain faults in the metric computation will result in no errors or negligible errors in the resulting video encoding.* If the faulty systems only rarely affect the block ranking during ME, then the coding penalty is likely to be minimal. This is a key of observation in our analysis of motion estimation hardware: we will seek to estimate the impact of faults on the accuracy of the metric computation (therefore we will focus on the matching process)and hence on the quality of the candidates chosen by the encoder. Acceptable hardware faults will be such that the optimal candidate may not always be chosen, but such that, in general, sufficiently good candidates are chosen instead.

While we focus our analysis on the architectures shown in Figure 5.2, and assume full search block matching with SAD metric, the results can be applied to other implementation structures as well.

5.3.2 Matching process architecture and model

There are numerous implementation architectures for the matching metric computation (MMC) process, with different levels of parallelism as shown in Figure 5.2 [59]. Because AD-PE is the major building block for matching process, the architectures in the figure only show AD-PE (without pixel inputs for simplicity) and data flows. In this figure, Type-1 is a fully serial architecture, Type-2 is a columnwise parallel architecture, Type-3 is a fully parallel architecture, and Type-4 uses the same circuit for metric computation of each column. As one can see from this figure, data dependency and data flow of a matching process implementation are different depending on the implementation architecture. To analyze fault effects on various matching process architectures, and to design generic testing algorithms, we model a matching process architecture as a tree graph model. For this we make the following assumptions.

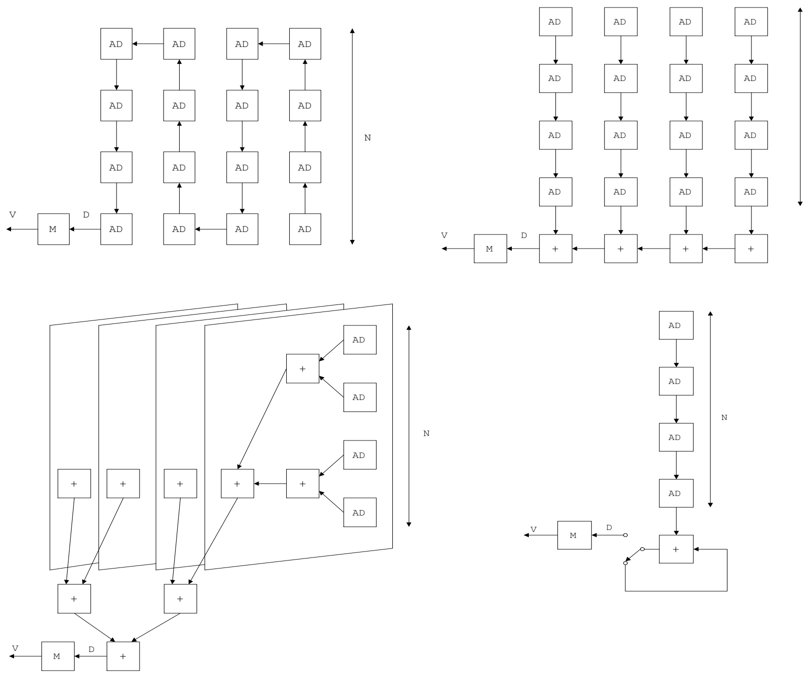


Figure 5.2: Examples of motion estimation matching process architecture. In this figure, only processing elements are shown without showing inputs for simplicity. Details of these processing elements are shown in Figure 5.1. Type-1: upper left, Type-2: upper right, Type-3: lower left, and Type-4: lower right [59].

5.3.2.1 Assumptions

We assume in our analysis that full search is used and that the SAD is used as a metric. We also make the following assumptions. First, all the outputs of AD-PEs and adders are 16 bit wide. Usually ME is performed on 16×16 luminance MBs, where each luminance pixel is represented with 8 bits. Thus the absolute difference between two luminance values can also be represented by 8 bits. Therefore, the maximum SAD value when combining the ADs of 16×16 pixels can be represented using 16 bits[2]. Second, our work is focused on the interconnect faults that affect the data transfer between PEs. Therefore, we assume that the absolute difference operation and the carry generation process in an adder are error-free. These error-free processes can be achieved by well-known self checking design techniques [64, 34, 58]. Third, we assume that the faults in the interconnect between processing elements are stuck-at-0 (SF0) or stuck-at-1 (SF1) faults, which cause the given data line to produce a constant value (0/1) independent of other signal values in the circuit.

5.3.2.2 Tree graph model of metric computation architectures

In a matching metric computation (MMC) architecture, the main component for each PE is an adder that computes the absolute pixel difference for the current pixel and adds it together with one or more partial SAD values. MMC architectures can be viewed as arrays of cascaded adders and represented as tree graphs (see Figure 5.3), where each node represents an adder, and edges connecting two nodes represent a data bus (i.e., a set of data lines). Tree graphs similar to that in Figure 5.3 can be constructed for all non-recursive architectures in Figure 5.2, i.e., all except Type-4. In this model, for simplicity, we will omit the arrows if there is no ambiguity in data flow. Under our assumption that numerical operations are correct, this tree model will be used to determine the errors caused at the output by faults in the data transfer between PEs.

[2]This is obtained as $\lceil \log_2(N^2 \times (2^8 - 1)) \rceil$, $N = 16$, where $\lceil \cdot \rceil$ is the ceiling operator.

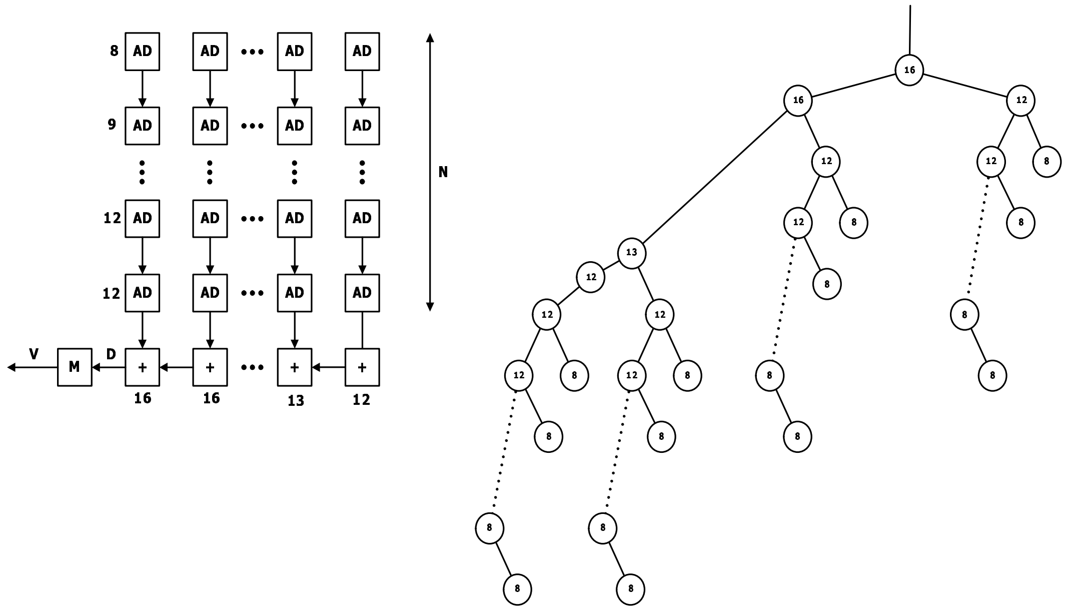


Figure 5.3: Dependence graph (left) of the Type-2 architecture and a corresponding tree graph model (right). Next to each PE we show the number of bits needed to represent partial SADs at the output of each node.

In the tree graph model, we define the root node as the closest node to the final output. Each node is cascaded in the tree graph model. Thus, faults in an input data line can be interpreted as the faults in the output data line of a child-node. Therefore, in our analysis, we only allow faults in the output data line for each node. To construct a tree model, we follow a convention which puts the root node at the top node.

Depending on the architecture used, the set of possible outputs of each PE is different. Here we define the *dynamic range* as the set of all possible signal values in a given data bus. Clearly, since the system computes the total SAD by adding partial SAD values, the dynamic range of nodes closer to the final output node

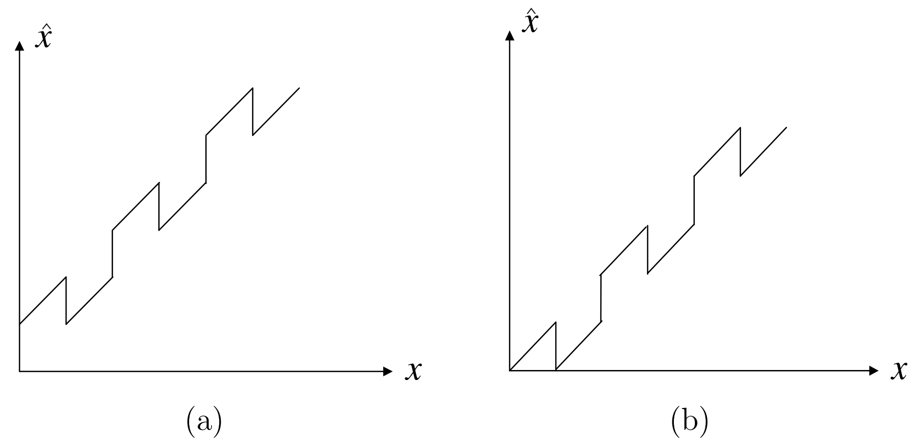


Figure 5.4: Input x vs. output $\hat{x}$ of a data bus (a) when there is SF1 fault in a data line $n = p$, and (b) when there is SF0 fault in a data line $n = p$.

(i.e., the root node) will tend to be larger. This can be seen in Figure 5.3 where we include the bit-width needed to represent the output dynamic range of each node in the fault-free case. The dependence graphs and the tree graph models can be constructed in a similar way for other (non-recursive) architectures.

5.3.3 Fault effect modeling for MMC process

5.3.3.1 Uniform offset interval

Let us first analyze a single stuck-at fault in an m-bit data bus, where data line 0 corresponds to the LSB. Let x and $\hat{x}$ denote the input and output of the data bus, respectively, and let the error in the data bus be $e = \hat{x} - x$. If there is a single SF1 fault in the p-th data line then we have that:

$$e = \begin{cases} 2^p, & 2k \cdot 2^p \leq x \leq (2k+1) \cdot 2^p - 1 \\ 0, & (2k+1) \cdot 2^p \leq x \leq (2k+2) \cdot 2^p - 1 \end{cases} \tag{5.1}$$
$$\forall\ k = 0, 1, \ldots, 2^{m-p-1} - 1.$$

A similar relationship can be derived for the case of SF0 fault. The relationship between $\hat{x}$ and x is illustrated in Figure 5.4. As can be seen from (5.1), a single stuck-at fault causes some inputs to be shifted by a constant amount, while some other inputs remain unchanged. Thus all inputs belonging to an interval

$$I(k,p) = [k \cdot 2^p, (k+1) \cdot 2^p - 1], \tag{5.2}$$

are shifted by the same amount (2^p if k is even, 0 if odd). We will call $I(k,p)$ the *uniform offset intervals* for a fault at data line p. All the elements in a uniform offset interval are shifted by the same amount. If there are multiple faults in a data bus, then the uniform offset interval is determined by the fault which is closest to LSB (that is, the fault leading to the smallest error).

Note that if all the inputs to a faulty data bus are shifted by the same amount (i.e., the dynamic rage of the input falls within one of these $I(k,p)$), then the ranking of values at the output remains unchanged in spite of the fault. This is

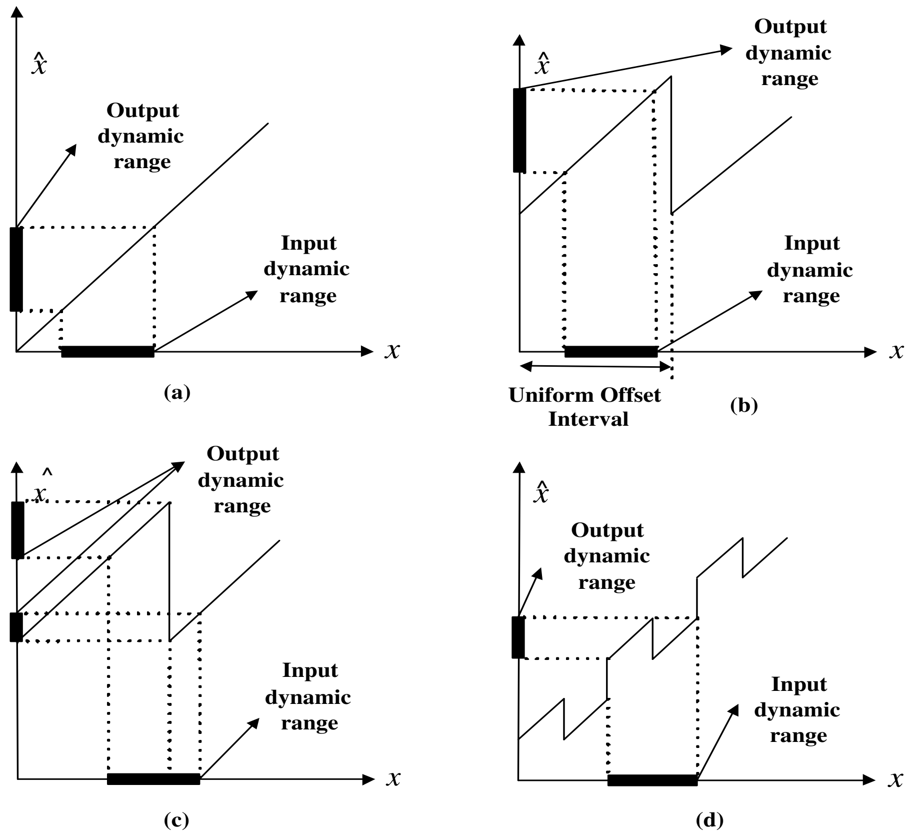


Figure 5.5: Dynamic range transform (a) no fault, (b) uniform shift, (c) and (d) non-uniform shift. The faults in this example are single SF1.

illustrated by Figure 5.5, where Figure 5.5(a) represents the fault-free case and Figure 5.5(b) shows the case when all input values are equally shifted. Conversely, when the input dynamic range is not completely enclosed in one of the $I(k,p)$, a *non-uniform offset* of the inputs is introduced (see Figures 5.5(c) and (d)), which can lead to changes in the ranking at the output. Note that in the fault-free case the lower bound of all dynamic ranges is zero, but this is no longer the case if faults occur.

Table 5.2: Effect of in-range SF1 faults in the cumulative adders in terms of PSNR degradation (dB). Negative entry represents PSNR degradation, and positive entry represents PSNR enhancement. These results are achieved by using MPEG-2 TM5 and the matching process architecture in Figure 5.1 with test sequences.

Bit/Column	0	4	8	12	15
0	0	0.007	0.002	0.011	0.006
2	0.037	0.014	0.001	0.049	-0.006
4	0.034	0.004	-0.010	0.005	-0.001
6	-0.005	-0.045	-0.021	0.009	-0.019
8	-0.007	-0.179	-0.126	-0.124	-0.234
10	0	-0.111	-1.019	-1.127	-0.879
12	0	0.014	-0.364	-2.466	-3.395
14	0	0	0	-0.001	-1.876

More specifically, for the case as in Figure 5.5 (c), the ranking of the input values are changed, but the one-to-one correspondence between input and output values of the data line is maintained. However, to recover the real input value, inverse mapping information is needed, and a compensation circuit is needed for each node. In this work we will only allow the compensation circuit to be located, if needed, at the very end of the metric computation. Therefore for this case, it is difficult to retrieve the real value. For cases such as that of Figure 5.5 (d), the input dynamic range is shifted and shrunk due to a fault. Therefore the ranking of the input values and the one-to-one correspondence between input and output are not maintained.

5.3.3.2 MMC Performance degradation by faults

We define *in-range* bits as the bit positions needed to represent the maximum dynamic range of a data bus, and *out-range* bits as the unused bit positions in a data bus. For example there may be 16 data lines for an operation which is known to have only 12 bit outputs when a MMC system is designed using a single standard module for all AD operations. Also, we define *in-range* and *out-range* faults as the faults which affect in-range bits and out-range bits respectively.

If all the faults in a MMC architecture are out-range faults, then all the dynamic ranges of intermediate metric values will be contained in uniform offset intervals at each node, which do not affect the ME outcome by preserving relative rankings (as an example see some of the in-range faults in high bit positions in Table 5.2). However, an in-range fault may introduce performance degradation due to altered relative rankings of matching metric. It is important to note that in-range faults may result in an uniform shift for specific inputs. For example, for a specific input, if the actual dynamic range of a specific input source is below an in-range fault, then it will result in uniform shift. However, because video input sources vary significantly, we analyze and design based on maximum dynamic range (worst case scenario). Therefore, the actual performance degradation depends on the distribution of residuals (frame differences) for each block, and fault combinations / locations. This is shown in Tables 5.2 and 5.3. In these tables, we show average peak signal to noise ratio (PSNR) degradations due to a SF1 fault at specified bit positions and PEs. ‘Column’ and ‘Row’ entry represents the horizontal and vertical position of each PE in the MMC architecture example shown in Figure 5.1 respectively: 0 represents the rightmost and the topmost, and 15 represents the leftmost and the bottommost for horizontal and vertical positions, respectively. In these tables, negative PSNR values represent degradation, and positive PSNR values represent quality improvements. Note that there are cases where average PSNR is higher in spite of faults. This is because a best candidate found by optimizing SAD metric may not provide optimal PSNR (which is computed based on mean squared error) results.

The performance degradation is relatively smaller for AD-PEs than for cumulative adders because AD-PEs affect a smaller number of partial SADs than cumulative adders. Also, the degradation becomes more severe when in-range faults occur at higher bit positions and near the output. Another interesting observation is that for cumulative adders, degradations become more severe for those closer to the output because average pixel differences for the good candidates are quite small, thus, a significant portion of partial sums are not affected by high bit positioned in-range faults for the cumulative adders far from the output. However, for those closer to the output, they are likely to be affected by these faults, leading to significant increases in distortion.

Table 5.3: Effect of single fault in AD-PE in terms of PSNR degradation (dB). Negative entry represents PSNR degradation, and positive entry represents PSNR enhancement. These results are achieved by using MPEG-2 TM5 and the matching process architecture in Figure 5.1 with test sequences.

Row/Column	0	4	8	12	15
0	0.0024	0.0102	0.0086	-0.0082	0.0066
4	-0.0009	-0.0039	0.0046	0.0019	-0.0015
8	-0.0019	-0.0037	-0.0058	-0.0019	-0.0015
12	0.0005	0.0022	0.0003	0.0062	-0.0015
15	0.0006	-0.0042	-0.0003	0.0016	-0.0038

5.3.3.3 Multiple faults

For the multiple fault case, the problem becomes more complex. At each individual node, the dynamic ranges are transformed by the mechanism shown in Figure 5.5, but the effect of multiple faults located in different data buses is not necessarily the sum of the effects of each individual fault. We illustrate examples of these cases below.

Case1-Multiple faults in a single data bus: For this case, the total error $e(x)$ is the sum of each error $e_i(x)$ due to each fault in a data line i.

$$e(x) = \sum_i e_i(x)$$

Case2-Multiple faults in cascaded data buses: Let us assume that two data buses are cascaded, Also let us assume that for the bus closer to the input x, there exist a SF1 fault at p data line, and for the data bus closer to the output z, there

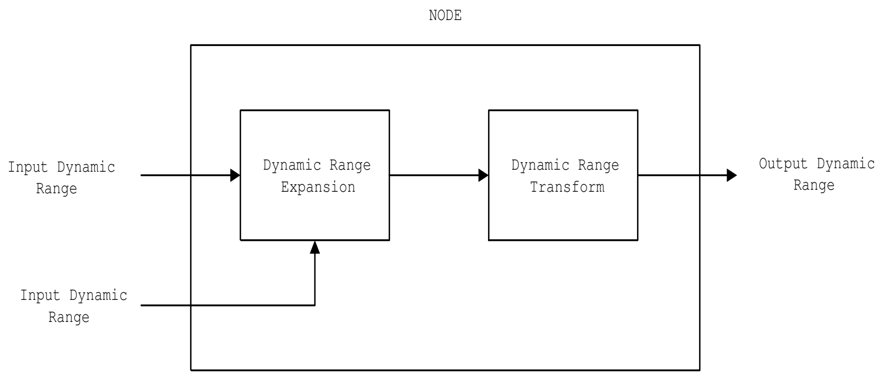


Figure 5.6: Interpretation of each node as the cascaded operations on input dynamic ranges.

exist a SF1 fault at q data line. Then, the relationship between the input x and the output z is

$$z = x + e_p(x) + e_q(x + e_p(x)).$$

The error term is not necessarily same as $e_p(x) + e_q(x)$. Thus, for this case the effect of multiple faults is not the sum of the individual effect.

5.3.3.4 Dynamic range transform

To analyze effect of faults in a MMC architecture, we use the concept of dynamic range and tree graph model we defined above. In a MMC architecture, as metric computation progresses from a leaf node (defined as an absolute pixel difference) to the root node (defined as the closest node to the output), dynamic ranges of intermediate metric values at each node are expanded (due to addition in a node), and are uniformly or non-uniformly shifted by faults at each node as shown in

Figure 5.5. We define the change of dynamic range due to faults in a data bus as the *dynamic range transform* at a node. Therefore in each node, the input dynamic ranges form an expanded input dynamic range (due to addition in a node), and the expanded input dynamic range is transformed by the faults in the node output as shown in Figure 5.6.

5.4 Test Vector Generation & Testing Algorithm

Testing of a system is an experiment in which the system is exercised by certain input sequences, and the responses are analyzed to determine the existence of faults in the system. A testing algorithm determines how the stimuli (tests which are input vectors) are applied and what the response to the stimuli is expected to be. Based on a chosen testing algorithm, a test generation algorithm generates pairs of tests and corresponding responses (output vectors) to be used for testing [12].

In this section, we design a testing algorithm and a test generation algorithm for our proposed error tolerant schemes for MMC architectures.

5.4.1 Testing metric

To design a test-generation/testing algorithm, we need to design a metric which will be used to analyze the system for (test, response) pairs. Let $D_{max}(\eta)$ and $D_{min}(\eta)$ be the observed distortion values at the root node of a MMC architecture when we apply to node η the maximum and minimum inputs, respectively. We propose to use $D_{max}(\eta) - D_{min}(\eta)$ as the metric for the proposed testing algorithm. If there is no in-range fault, which will ensure that the uniform offset interval is larger than the dynamic range at a node, and $D_{max}(\eta) - D_{min}(\eta)$ is same as the expanded dynamic range of a node, then the expanded dynamic range is embedded in a single uniform offset interval of a node, and this will ensure faults to cause uniform shift for the node. Note that, in contrast, most existing testing algorithms compare, for each node, the observed output values $D(\eta)$ at the root node with the expected output values, and if those are different, then mark the chip as a faulty circuit.

5.4.2 Testing algorithm design

We use the following observations from our error tolerance analysis to design a test vector generation and a testing algorithm for MMC architectures:

- From the leaf nodes to the root node of a tree graph model, the input dynamic ranges of the leaf nodes are expanded and transformed, and the dynamic ranges tend to increase.

- At each node, if an expanded input dynamic range (which is determined by faults in the subtree of the node) can be contained in a single uniform offset interval (which is determined by the faults in the node) at each node (we call this *lossless error tolerance*), then the faults cause uniform offset which preserves relative rankings; otherwise, the faults may cause non-uniform offset which results in distorted rankings.

- If the distorted rankings introduced by non-uniform shifts at each node are small enough, then the performance degradation will be minimal (we call this *lossy error tolerance*).

- $D_{max}(\eta)$ and $D_{min}(\eta)$ (more generally any output of node η, $D(\eta)$) is only observed at the root node, therefore, we need to test the data path from the parent node of η to the root node, *before* we test for a node η.

Thus our testing algorithm operates in a *top-down* manner, where the top is the root node, and the bottom contains the leaf-nodes. More specifically, the least significant k bit data lines of a given node are tested after all the data lines connecting these k bit lines to the root node need have been tested.

Based on these observations, our proposed testing algorithm is structured as follows (refer to Figure 5.7 for a flow chart of the proposed testing algorithm). First, based on the tree graph, we compute the minimum size of the uniform offset interval for each node that guarantees lossless operation. This step can be done independently for each MMC architecture. Therefore, we do not show this step in Figure 5.7.

Second, we check if the dynamic range distortion due to in-range faults passes admissibility testing. For each node, we test for potential in-range faults which are defined as faults occurring in data lines within the dynamic range of data being

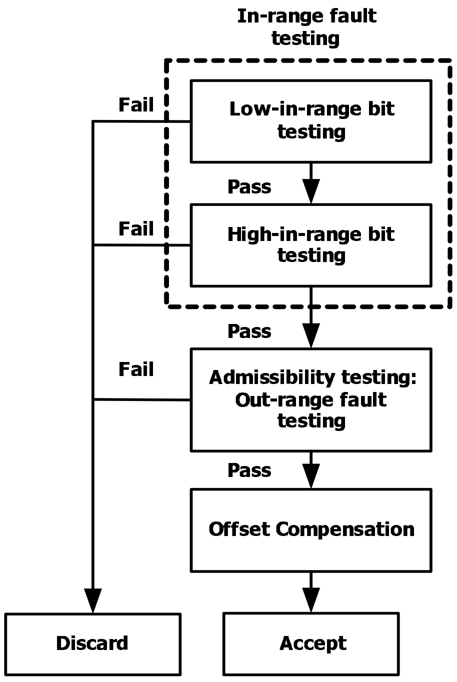


Figure 5.7: Schematic flow chart for the proposed testing algorithm

transferred. “Low-in-range” bits are the 8 least significant bits, which are enough to represent the dynamic range of leaf nodes. “High- in-range” bits are those bit-lines above the 8th most significant one that are within the dynamic range of the node. We first test low-in-range bits (low 8 bit) because low-in-range bits from multiple nodes are needed to excite high-in-range bits. This test is performed in top-down manner to ensure that all the low 8 bit data path from the leaf nodes to the root node are fault-free. During the tests, if faults are detected, then the dynamic range distortion introduced by the faults is compared with a quality threshold for admissibility testing. Admissibility testing is defined in the following section.

Table 5.4: $\alpha(p)$: number of maximum inputs required to excite $n = p$ bit position.

p	8	9	10	11	12	13	14	15
$\alpha(p)$	2	3	5	9	17	33	65	129

Testing for high-in-range bits for each node is performed from lower significance to higher significance bits and also in a top-down manner. Admissibility testing is also performed for high-in-range bits based on the dynamic range distortion they introduce. To excite $n = p$ bit ($n = 0$ is LSB) for a node, $\alpha(p)$ (shown in Table 5.4) consecutive inputs in one subbranch of the node are excited to the maximum value 255, where $\alpha(p) = \lceil \frac{2^p}{2^8-1} \rceil$, and $\lceil\cdot\rceil$ is the ceiling operation.

Third, we check if the dynamic range distortion due to out-range faults passes admissibility testing. For this, we check if the difference between $D_{max}(\eta) - D_{min}(\eta)$ and the size of dynamic range is within a quality threshold. For lossless error tolerance, when there is no in-range fault, the size of the uniform offset interval is larger than that of the expanded input dynamic range. Therefore, input dynamic ranges overlap with at most two uniform offset intervals. Thus, if $D_{max}(\eta) - D_{min}(\eta)$ is same as the size of dynamic range at a node, then this will ensure that the expanded dynamic range is contained in a uniform offset interval. Otherwise, non-uniform offsets will occur, potentially leading to errors at the output. For lossy error tolerance, $D_{max}(\eta) - D_{min}(\eta)$ may be different from the size of dynamic range of a node. In this case, we check if the dynamic range distortion is within a quality threshold defined by admissibility testing. Also, in this step, we need to check

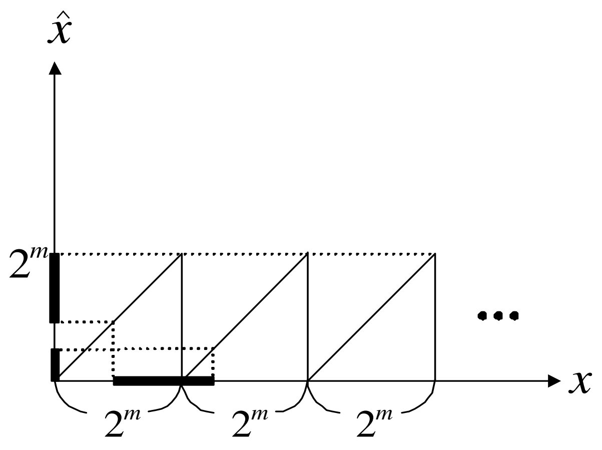


Figure 5.8: Overflow effect at a node when data bus width is m.

overflow effect due to dynamic range distortion at each node. An overflow should not occur except for the root node, because it should be compensated in the node where it occurs, and in the proposed scheme, compensation circuit is used only after the root node. To allow overflows at each node, we need to implement offset compensation at each node. The effect of overflow is shown in Figure 5.8 when data bus width is m.

5.4.3 Admissibility testing

For lossless error tolerance, there should be no in-range fault, and for out-range fault testing, $D_{max}(\eta) - D_{min}(\eta)$ should be same as the size of dynamic range at a node. Therefore the admissibility testing for each node η is as follows.

$$D_{max}(\eta) - D_{min}(\eta) == M_\eta \times \ 255, \tag{5.3}$$

where M_η is the number of leaf nodes (inputs) contained in the subtree of η.

For lossy error tolerance, we will accept low-in-range faults which result in acceptable performance degradation. If maximum distortion Δ_D of the matching metric of a MMC architecture for a given fault is less than a threshold Th_D, this fault will be acceptable. For single fault cases, Δ_D can be estimated using the proposed testing algorithm. However, because the testing algorithm is based on the observed output at the root node, multiple faults in the same bit positions at different nodes in the subtree may not be discriminated because our low-in-range testing uses only maximum and minimum inputs for each node rather than exhaustive input combinations. Denote $\delta_D(\eta)$ the maximum distortion of the matching metric for a node η. During the low-in-range bit test, if we detect a single fault, then the upper-bound $\widehat{\delta}_D(\eta)$ for $\delta_D(\eta)$ is estimated as:

$$\widehat{\delta}_D(\eta, p) = N_s(\eta) \times 2^p, \tag{5.4}$$

where $N_s(\eta)$ is the number of nodes in the subtree of η and p is the bit position of the fault. This means that the effect of the single fault is weighted by the number of sub-tree inputs. Then, the upper-bound $\widehat{\Delta}_D$ of Δ_D in (5.1) can be estimated as:

$$\widehat{\Delta}_D = \sum_{\eta \in \Theta_f} \sum_{p \in B_f(\eta)} \widehat{\delta}_D(\eta, p), \tag{5.5}$$

where Θ_f is the set of faulty nodes which are first detected through top-down testing, that is, for the same bit fault, Θ_f only includes the nodes closest to the root node. $B_f(\eta)$ is the set of faulty data lines of node η. Therefore, if $\widehat{\Delta}_D \leq Th_D$, then we can guarantee that the performance degradation is less than Th_D. Thus, for lossy error tolerance, to limit the distortion in the matching metric of a MMC architecture, we propose to use the following admissibility testing. For in-range fault testing, we use

$$\widehat{\Delta}_D \ \leq \ Th_D, \tag{5.6}$$

and for out-range fault testing, we use

$$|\ |D_{max}(\eta) - D_{min}(\eta)| - M_\eta \times \ 255| \ \leq \ Th_D, \tag{5.7}$$

where M_η is the number of leaf nodes (inputs) contained in the subtree of η.

Lossy fault tolerance is based on the idea that the effect of low-in-range fault is relatively small as compared to that of high-in-range. This could suggest not using some LSB data lines at all from the design stage. However, this could lead to quite significant distortion in metric computations because the results would be same as those when all the low-in-range data lines have faults. More specifically, let us assume that we do not have data lines whose indices are less than $n = p$ ($n = 0$ is LSB), then for each node, $\widehat{\delta}_D(\eta) = 2^p - 1$, and $\widehat{\Delta}_D = N_T \times (2^p - 1)$, where N_T

is the total number of nodes. Therefore the performance degradation can be quite significant.

5.4.4 Pseudo code for high-in-range bit testing

In the proposed testing algorithm, we first test low-in-range bits (low 8 bit) in top-down manner to ensure that all the low 8 bit data path from the leaf nodes to the root node satisfy an admissibility testing. Then, we perform high-in-range bits testing by traversing tree graph model from top-down manner and from low bits to high bits manner. In this section, we describe an example of pseudo code for a high-in-range bit testing of a generic MMC architecture. For this we define the following.

- η_0: root node, i.e. the closest node to the output of a MMC architecture.
- η_{leaf}: a leaf node.
- $left(\eta)$: left child-node of a node η.
- $right(\eta)$: right child-node of a node η.
- $\kappa(\eta)$: minimum number of bits to represent the dynamic range of a node η.

For the pseudo code in this section, we construct a tree graph model for a MMC architecture as follows: For a node η, the $left(\eta)$ and $right(\eta)$ should satisfy $\kappa(left(\eta)) \geq \kappa(right(\eta))$. An example of resulting tree graph model is shown in Figure 5.3. Below we show the pseudo code.

High-in-range bit testing:

- Step 0: $p \leftarrow \kappa(\eta_{leaf})$.
- Step 1: If $TreeCheck(\eta_0, p)$ is TRUE, then go to Step 2, otherwise go to Step 4.
- Step 2: If $p < \kappa(\eta_0)$, then $p \leftarrow p + 1$, otherwise go to Step 3.
- Step 3: Pass the test.
- Step 4: Fail the test.

```
DynamicRangeCheck($\eta$,$p$){

IF neither $left(\eta)$ or $right(\eta)$ is a leaf-node,
    RETURN TRUE
ELSE
    Excite $\alpha(p)$ right-most leaf nodes in the subtree of $\eta$,
    then perform an admissibility testing.
    IF Pass the admissibility testing
        RETURN TRUE
    ELSE
        RETURN FALSE

}
TreeCheck($\eta$,$p$){
```

```
IF $\kappa(\eta) \leq p$,
    RETURN TRUE
ELSE
    IF NOT DynamicRangeCheck($\eta$,$p$)
        RETURN FALSE
    ELSE
        IF $\kappa(right(\eta)) > p$
            IF TreeCheck($right(\eta)$,$p$)
                RETURN TreeCheck($left(\eta)$,$p$)
            ELSE
                RETURN FALSE
        ELSE
            RETURN TreeCheck($left(\eta)$,$p$)
}
```

TreeCheck is the function that traverses the tree graph model in top-down and low-bit to high-bit order. *DynamicRangeCheck* function checks if the dynamic range is distorted.

5.4.5 Test vector generation

A test is a pair constituting of an input and an expected output. In our proposed testing algorithm, an input vector is composed of minimum difference inputs (0) and maximum difference inputs (255) because the proposed testing metric is $D_{max}(\eta) -$

Table 5.5: The number of tests required for the testing of Type-2 structure in Figure 5.2.

Testing Stage	# of tests
All $0x00$ input	1
Single $0xFF$ input	256
High-in-range bit test	833
All $0xFF$ input	1
Total # of tests	1091

Table 5.6: The number of tests required for each high-in-range bit testing for Table 5.5.

Bit position $n = p$	# of tests
$n = 8$	240
$n = 9$	224
$n = 10$	192
$n = 11$	128
$n = 12$	15
$n = 13$	14
$n = 14$	12
$n = 15$	8

$D_{min}(\eta)$. For low-in-range bit testing, from the top node to leaf-nodes, a single node is excited to the maximum input in top-down manner. Because a single input is excited, the expected output is 255. For high-in-range bit testing, the test input sequence depends on how we traverse nodes and data buses for the tree graph model during testing. Therefore, the test inputs can be generated in a similar way as *TreeCheck* function. The expected response is determined by the expected size of the dynamic range of each node, which is determined by the number of excited leaf-nodes $\alpha(p)$ for each step.

Table 5.7: The number of tests for the proposed and exhaustive testing for Type-1, Type-2, and Type-3 architectures in Figure 5.2.

	Type-1	Type-2	Type-3
Proposed	2051	1091	512
Exhaustive	7682	5738	4606

In Table 5.5, we show the number of tests required in each testing stage for the proposed testing algorithm. For this table, Type-2 architecture on Figure 5.2 is used. As one can see from this table, the most test inputs are needed for high-in-range bit testing, because high-in-range faults are tested for each bit. The number of tests required for each high-in-range bit testing is shown in Table 5.6.

In Table 5.7, we show the number of tests for the proposed testing and exhaustive testing for Type-1, Type-2, and Type-3 architectures in Figure 5.2. In this table the exhaustive testing method is not based on an analysis of the ME system and therefore it views the given system as a black box and checks for the correctness of each bit, so that the number of tests is proportional to the number of data lines that can be excited separately. Clearly, our proposed technique can be implemented with limited test complexity, as compared to exhaustive testing. Moreover, for the proposed testing algorithm, all inputs applied during testing are either the minimum or maximum value, so that we can represent each node test input with a single bit, and then generate the corresponding test by using a multiplication by maximum difference (255) in the testing hardware. Therefore, our proposed testing algorithm requires small storage space for the test vectors.

Table 5.8: Yield-rate Y_r and the probability of k fault case $P_X(k)$

Y_r	$P_X(1)$	$P_X(2)$	$P_X(3)$
0.2	0.3219	0.259	0.139
0.3	0.3612	0.2174	0.0873
0.4	0.3665	0.1679	0.0513
0.5	0.3466	0.1201	0.0278
0.6	0.3065	0.0783	0.0133
0.7	0.2497	0.0445	0.0053
0.8	0.1785	0.0199	0.0015
0.9	0.0948	0.005	0.0002

Table 5.9: The percentage of acceptable faults by the proposed error tolerance scheme for Type-1, Type-2, and Type-3 architecture in Figure 5.2. SSF: single stuck-at fault, DSF: double stuck-at fault.

	Type-1	Type-2	Type-3
$SSF_{Lossless}$	6.23%	34.58%	43.86%
$DSF_{Lossless}$	0.36%	8.54%	19.13%
SSF_{Lossy}	9.16%	56.06%	75.27%
DSF_{Lossy}	0.42%	10.84%	36.77%

5.5 Performance of Proposed System Level Error Tolerance Scheme

In this section, we evaluate the yield-rate increase achievable with our proposed error tolerance techniques. In our evaluation we assume uniform spatial distribution of faults, a Poisson distribution model, and stuck-at faults affecting only the data buses.

5.5.1 Yield-rate and probability of fault

First, we describe a probability model which we will use for yield rate analysis. Based on the assumptions we made and a simple probability model, we can estimate the composition of a fault space. We employ the poisson statistics to analyze the composition of a fault space. Let us assume that faults are distributed uniformly on a wafer which is used to fabricate a chip. Also let us assume that SF0 and SF1 are equi-probable, that is, $P_{SF0} = P_{SF1}$where P_{SF0} and P_{SF1} are the probability of SF0 and SF1 respectively. In this case, the probability of having k faults within the wafer area is given by the Poisson probability density function [16].

$$P_X(k) \triangleq Prob\{X = k\} = \frac{e^{-\lambda}\lambda^k}{k\ !} \tag{5.8}$$

In this model, X is a random variable which specifies the number of faults in a chip. For $k = 0$, we get the yield-rate Y_r as $Y_r = e^{-\lambda}$. From this we can compute λ, and compute $P_X(k), k > 0$. Table 5.8 shows an example of fault space composition when Y_r is given. From this table, one can see that the most dominant fault space is single fault case.

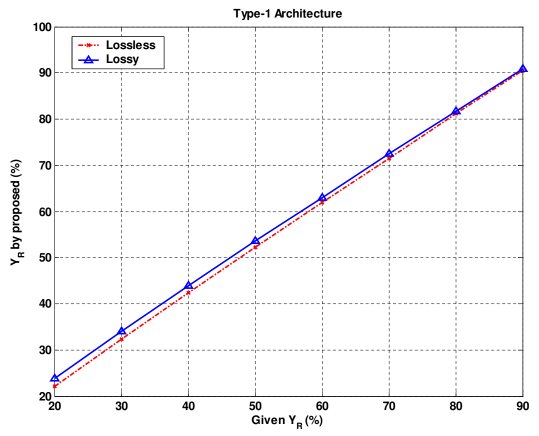


Figure 5.9: Given yield-rate (Y_R) and the improved yield rate by the proposed fault tolerance scheme for Type-1 architecture in Figure 5.2.

5.5.2 Yield-rate increase achievable with error tolerance

In this section, we show the yield-rate increase for various MMC architectures by employing our proposed error tolerance. For this, we assume uniform spatial distribution of faults to use Poisson distribution model in (5.8).

In Table 5.9, we show the acceptable faults percentage by employing proposed *lossless and lossy* error tolerance scheme. For the lossy scheme, we set the threshold TH_D to 64, which is observed to result in less than 0.1 dB degradation as can be seen from Table 5.2. Note that the proposed testing scheme accepts a significant portion of chips which would have been discarded otherwise. Also, one can notice that the percentage of acceptance is much higher for Type-2 & Type-3 architectures than for Type-1 architecture. This can be explained as follows: First, due to the lack of parallelism of Type-1 architecture, there are fewer redundant data buses,

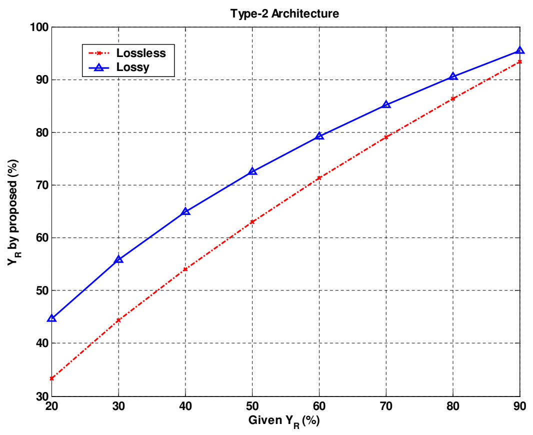


Figure 5.10: Given yield-rate (Y_R) and the improved yield rate by the proposed fault tolerance scheme for Type-2 architecture in Figure 5.2.

therefore, the percentage of acceptable faults is lower for Type-1 architecture. By employing variable width data line, we can completely remove the redundant out-range data buses. In this case lossless error tolerance cannot be achieved. Second, for lossy error tolerance, from (5.5), the upper-bound of performance degradation is proportional to the number of nodes $N_s(\eta)$ in the subtree. For Type-1 architecture due to the serial connection of all nodes, $N_s(\eta)$ is larger, and therefore, fewer in-range faults are accepted.

In Figure 5.12, an example of typical progressions in yield learning is shown [30]. The dotted lines in this Figure are the upper and lower bound in yield learning progressions. Around $20-30\%$ of yield rate is considered as production readiness, and 50% of yield rate is considered as volume production readiness [30].

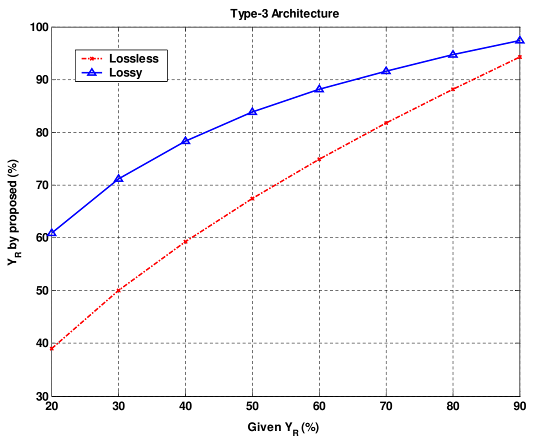


Figure 5.11: Given yield-rate (Y_R) and the improved yield rate by the proposed fault tolerance scheme for Type-3 architecture in Figure 5.2.

By using these results and the composition of a fault space in Table 5.8, in Figure 5.9 - 5.11, we show the improved yield-rate by the proposed error tolerance schemes. If the manufacturing is in the volume production stage where $Y_r > 90\%$, then the yield-rate increase is not that significant. However, for the chip development stage where the given yield-rate is quite small, the yield-rate increase achievable with our proposed fault tolerance scheme is quite significant. Therefore, as one can see from Figure 5.12, the beginning timing of Stage 2 and Stage 3 in manufacturing can be advanced.

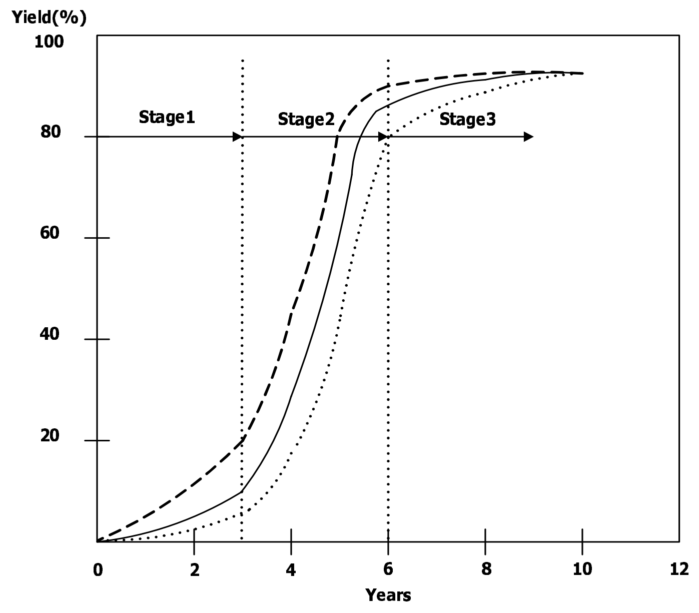


Figure 5.12: Typical progressions in yield learning [30].

5.6 Conclusion

In this chapter, a novel system-level error tolerance scheme for VLSI system is proposed. The proposed design and testing algorithm is applied to the matching process computation (MMC) of motion estimation for video compression, and the corresponding error tolerance characteristics is analyzed in detail. Based on this analysis, in this work, the proposed testing algorithm uses a tree structured modeling of a MMC architecture and the dynamic range concept proposed.

The proposed scheme can be divided into *lossless* and *lossy* fault tolerance. Lossless approach investigates the effect of faults in redundant data lines. If the effect by these faults can be easily compensated by our scheme, then faulty chips are accepted. Lossy scheme provides an upper-bound of performance degradation by which we can trade-off between the yield rate improvement and slight degradation of matching performance.

Also, we propose a testing metric which relies on the difference between maximum and minimum response of processing elements, which enables us to reduce the number of test vectors required. Due to this, each input for a node is minimum or maximum value, therefore, we can represent each input for a node by a single bit, and can generate the corresponding test by using a multiplication in the testing hardware. This enables us to use less storage space for the test vectors.

By the proposed scheme, the yield rate for a MMC architecture can be significantly improved if the production is in the early stage. This can advance the beginning timing of volume production of the given architecture.

Bibliography

[1] *JVT Reference Software, version JM50c.* http:// bs.hhi.de/ suehring/ tml/ download/.

[2] *JVT Reference Software, version JM8.1a.* http:// bs.hhi.de/ suehring/ tml/ download/.

[3] *TMN H.263+ encoder version 3.0.* University of British Columbia, Canada.

[4] *ISO/IEC JTC1 Coding of Moving Pictures and Associated Audio for Digital Storage Media at Upto About 1.5Mbit/s-Part2: Video.* International Standard, Mar. 1993.

[5] *ITU-T Recommendation H.261, Video Codec for Audiovisual Services at px64kbits/s.* Mar. 1993.

[6] *Information Technology-JPEG-Digital compression and coding of continuous-tone still image-Part 1: Requirement and Guidelines.* ISO/IEC 10 918-1 and ITU-T Recommendation T.81, 1994.

[7] *ISO/IEC JTC1 13818-2; ITU-T Recommendation H.262, Generic Coding of Moving Pictures and Associated Audio Information-Part2:Video.* International Standard, Nov. 1994.

[8] *ITU-T Recommendation H.263 Version 1, Video coding for low bitrate communication.* Nov. 1995.

[9] *ISO/IEC JTC1/SC29/WG11 N2202 Committee Draft.* Mar. 1998.

[10] *ITU-T Recommendation H.263 Version 2 (H.263+), Video coding for low bitrate communication.* Jan. 1998.

[11] *Information Technology-JPEG 2000-Image Coding System-Part 1: Core Coding System.* ISO/IEC 15 444-1, 2000.

[12] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design.* The Institute of Electrical and Electronics Engineering, Inc., New York, 1990.

[13] V. Bhaskaran and K. Konstantibides. *Image and Video Compression Standards: Algorithm and Architectures.* Kluwer Academic Publishers, Boston, MA, 1995.

[14] M. Bierling. Displacement estimation by hierarchical blockmatching. In *Proc. SPIE conf. Visual Commun. Image Processing*, pages 942–951, Boston, Nov. 1988.

[15] M. Bierling and R. Thoma. Motion compensating field interpolation using a hierarchically structured displacement estimator. *Signal Processing*, 11:387–404, Dec. 1986.

[16] P. J. Bonk, M. R. Gupta, R. A. Hamilton, and A. V. S. Satya. Manufacturing yield. In *Microelectronics Manufacturing Diagnostics Handbook, A. H. Ladzberg Ed., Van Nostrand Reinhold*, pages 9–35, New York, 1993.

[17] D. G. Boyer. Symbolic layout compaction review. In *Proc. 25th ACM/IEEE Design Automation Conf.*, pages 383–389, Sep. 1988.

[18] M.A. Breuer, S.K. Gupta, and T.M. Mak. Defect and error tolerance in the presence of massive numbers of defects. *IEEE Design & Test of Computers*, 21:216–227, May–June 2004.

[19] P. J. Burt. Multiresolution techniques for image reperesentation, analysis, and 'smart' transmission. In *Proc. SPIE conf. Visual Commun. Image Processing*, pages 2–15, Philadelphia, PA, nov. 1989.

[20] J. Chalidabhongse and C.-C. J. Kuo. Fast motion vector estimation using multi-resolution-spatio-temporal correlations. *IEEE Trans. Circuits Syst. Video Technol.*, 7(3):477–488, Jun. 1997.

[21] E. Chan, A. Rodriguez, R. Gandhi, and S. Panchanathan. Experiments on block matching techniques for video coding. *Multimedia Systems*, 2:228–241, 1994.

[22] Y.-H. Choi and M. Malek. A fault-tolerant fft processor. *IEEE Trans. Computers*, 37(5):617–621, May 1988.

[23] I. Chong and A. Ortega. Harware testing for error tolerance in multimedia compression based on linear transforms. In *Proc. IEEE Intl. Symp. on Defect and Fault Tolerance in VLSI Systems, DFT'05*, Monterey, CA, Oct. 2005.

[24] H. Chung and A. Ortega. Low complexity motion estimation algorithm by multiresolution search for long-term memory motion compensation. In *Proc. IEEE Int. Conf. Image Processing*, Rochester, NY, Sep. 2002.

[25] H. Chung and A. Ortega. *System Level Fault Tolerance for Motion Estimation : Technical Report USC-SIPI 354.* Signal and Image Processing Institute, University of Southern California, Los Angeles, CA, 2002.

[26] H. Chung, A. Ortega, and A. A. Sawchuk. Low complexity motion estimation for long-term memory motion compensation. In *Proc. SPIE conf. Visual Commun. Image Processing*, Jan. 2002.

[27] H. Chung, D. Romacho, and A. Ortega. Fast long-term motion estimation for h.264 using multiresolution search. In *Proc. IEEE Int. Conf. Image Processing*, volume 3, pages 905–908, Barcelona, Spain, Sep. 2003.

[28] F. Dufaux and F. Moscheni. Motion estimation techniques for digital tv: A review and a new contribution. *Proceedings of the IEEE*, 83(6):858–876, Jun. 1995.

[29] T. Wiegand (ed.). Joint working draft, version 2 (wd-2). *Joint Video Team(JVT) of ISO/IECMPEG and ITU-T VCEG, JVT-B118r2*, Mar. 2002.

[30] B. El-Kareh, A. Ghatalia, and A. V. S. Satya. Yield management in microelectronic manufacturing. In *Proc. Electronic Components and Technology Conference*, pages 21–24, May 1995.

[31] B. Furht, J. Greenberg, and R. Westwater. *Motion Estimation Algorithms for Video Compression.* Kluwer Academic Publishers, Boston, MA, 1997.

[32] B. Girod. The efficiency of motion-compensationg prediction for hybrid coding of video sequences. *IEEE J. Selected Areas in Comm.*, SAC-5(7):1140–1154, Aug. 1987.

[33] F. Glazer, G. Reynolds, and P. Anandan. Scene matching by hierarchical correlation. In *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, pages 432–441, Washington, DC, Jun. 1983.

[34] M. Gossel and E. S. Sogomonyan. New totally self-checking ripple and carry look-ahead adders. In *Proc. 3rd Int. On-line Testing Workshop*, pages 36–40, 1997.

[35] MPEG Software Simulation Group. *MPEG2 Video Codec Version 1.2.*

[36] K. S. Harish and K. M. M. Prabhu. Fixed-point error analysis of two dct algorithms. In *IEE Proc. Vis. Image Signal Process.*, volume 147, 2000.

[37] B. G. Haskell. Frame replenishment coding of television. In *Image Transmission Techniques.* W. K. Pratt Ed., Academic Press, 1978.

[38] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/avc baseline profile decoder complexity analysis. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):704–716, Jul. 2003.

[39] K.-H. Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Computers*, 33(6):518–528, Jun. 1984.

[40] J. Jain and A. Jain. Displacement measurement and its application in interframe image coding. *IEEE Trans. on Comm.*, 29(12):1799–1808, Dec. 1981.

[41] Z. Jiang and S. Gupta. An atpg for threshold testing: Obtaining acceptable yield in future processes. In *International Test Conference*, 2002.

[42] J. Koga, K. Iiunuma, A. Hirani, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. In *Proceedings of the National Telecommunications Conference*, pages G5.3.1–5.3.5, 1981.

[43] I. Koren and Z. Koren. Analysis of a hybrid defect-tolerance scheme for high-density memory ics. In *Proc. 1997 IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Oct. 1997.

[44] R. Kutka. Content-adaptive long-term prediction with reduced memory. In *Proc. IEEE Int. Conf. Image Processing*, volume 3, Barcelona, Spain, Sep. 2003.

[45] W. Lee, Y. Kim, R. J. Grove, and C. J. Read. Media station 5000: Integrating video and audio. *IEEE Multimedia*, 1(2):50–61, 1994.

[46] K. Lengwehasatit and A. Ortega. Probabilistic partial-distance fast matching algorithms for motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, 11(2):139–152, Feb. 2001.

[47] J. Li, X. Lin, and Y. Wu. Multiresolution tree architecture with its application in video sequence coding: A new result. In *Proc. SPIE conf. Visual Commun. Image Processing*, pages 730–741, 1993.

[48] R. Li, B. Zeng, and M. L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, 4:438–442, Aug. 1994.

[49] W. Li and E. Salari. Successive elimination algorithm for motion estimation. *IEEE Trans. Image Processing*, 4(1):105–107, Jan. 1995.

[50] Y.-C. Lin and S.-C. Tai. Fast full-search block-matching algorithm for motion-compensated video compression. *IEEE Trans. on Comm.*, 45(5):527–531, May 1997.

[51] B. Liu and A. Zaccarin. New fast algorithms for the estimation of block motion vectors. *IEEE Trans. Circuits Syst. Video Technol.*, 3(2):148–157, Apr. 1993.

[52] L.-K. Liu and E. Feig. A block-based gradient descent search algorithm for block motion estimation in video coding. *IEEE Trans. Circuits Syst. Video Technol.*, 4:419–422, Aug. 1994.

[53] M. Lorenzetti. The effect of channel router algorithms on chip yield. In *MCNC International Workshop on Layout Synthesis*, May 1990.

[54] Y. MA. An accurate error analysis of fft algorithm. *IEEE Trans. Signal Processing*, 45(6):1641–1645, Oct. 1997.

[55] J. Mitchell, W. Pennebaker, C. E. Fogg, and D. J. LeGall. *MPEG Video Compression Standard*. Chapman and Hall, New York, 1997.

[56] B. Natarajan, V. Bhaskaran, and K. Konstantinides. Low-complexity block-based motion estimation via one-bit transforms. *IEEE Trans. Circuits Syst. Video Technol.*, 7(4):702–706, Aug. 1997.

[57] T. Naveen and J. W. Woods. Motion compensated multiresolution transmission of high definition video. *IEEE Trans. Circuits Syst. Video Technol.*, 4(1):29–41, Feb. 1994.

[58] M. Nicolaidis. Efficient implementations of self-checking adders and alus. In *Proc. FTCS 23*, pages 586–595, 1993.

[59] P. Pirsch, N. Demassieux, and W. Gehrke. Vlsi architectures for video compression-a survey. *Proc. IEEE*, 83(2):220–246, Feb. 1995.

[60] K. R. Rao and P. Yip. *Discrete Cosine Transform*. Academic Press, Inc., San Diego, CA, 1990.

[61] D. Romacho. *Fast Search Algorithms for Long-Term Memory Motion Compensation : Diploma Thesis*. Polytechnic University of Catalonia, 2007.

[62] A. Roy-Chowdhury and P. Banerjee. Algorithm-based fault location and recovery for matrix computations on multiprocessor systems. *IEEE Trans. Computers*, 45(11):1239–1247, Nov. 1996.

[63] S. E. Schuster. Multiple word/bit redundancy for semiconductor memories. *IEEE J. Solid-State Circuits*, SSC-13:698–703, Oct. 1978.

[64] F.W. Shih. High performance self-checking adder for vlsi processor. In *Custom Integrated Circuits Conference, Proc. IEEE*, pages 15.7/1–15.7/3, May 1991.

[65] R. Sitaraman and N. K. Jha. Optimal design of checks for error detection and location in fault-tolerant multiprocessor systems. *IEEE Trans. Computers*, 42(7):780–793, Jul. 1993.

[66] R. Srinivasan and K. Rao. Predictive coding based on efficient motion estimation. *IEEE Trans. on Comm.*, 33(8):888–896, Aug. 1985.

[67] A. M. Tekalp. *Digital Video Processing.* Prentice Hall PTR, NJ, 1995.

[68] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim. A novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, 8(4):369–377, Aug. 1998.

[69] A. M. Tourapis. Enhanced predictive zonal search for single and multiple frame motion estimation. In *Proc. SPIE conf. Visual Commun. Image Processing*, pages 1069–1079, Jan. 2002.

[70] A. M. Tourapis, O. C. Au, and M. L. Liou. Highly efficient predictive zonal algorithms for fast block-matching motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, 12(10):934–947, Oct. 2002.

[71] H.-Y. C. Tourapis. Fast motion estimation within the jvt codec: Jvt-e023. *Joint Video Team(JVT) of ISO/IECMPEG and ITU-T VCEG 5th Meeting*, Oct. 2002.

[72] K. M. Uz, M. Vetterli, and D. J. LeGall. Interpolative multiresolution coding of advanced television with compatible subchannels. *IEEE Trans. Circuits Syst. Video Technol.*, 1(1):86–99, Mar. 1991.

[73] A. Venkataraman, H. Chen, and I. Koren. Yield enhanced routing for high-performance vlsi designs. In *Proc. Microelectronics Manufacturing Yield, Reliability and Failure Analysis, SPIE97*, Oct. 1997.

[74] Y. Wang, J. Ostermann, and Y.-Q. Zhang. *Video Processing and Communications.* Prentice-Hall, Inc., NJ, 2002.

[75] C. J. Weinstein. *Quantization Effects in Digital Filters*, volume 468. MIT Lincoln Laboratory Technical Report, 1969.

[76] T. Wiegand and B. Girod. *Multi-frame Motion-compensated Prediction for Video Transmission.* Kluwer Academic Publishers, 2001.

[77] T. Wiegand, B. Lincoln, and B. Girod. Fast search for long-term memory motion-compensated prediction. In *Proc. IEEE Int. Conf. Image Processing*, volume 3, pages 619–622, Chicago, IL, 1998.

[78] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):560–576, Jul. 2003.

[79] T. Wiegand, X. Zhang, and B. Girod. Long term memory motion compensated prediction. *IEEE Trans. Circuits Syst. Video Technol.*, 9(1):70–84, Feb. 1999.

[80] J. W. Woods and T. Naveen. Subband encoding of video sequences. In *Proc. SPIE conf. Visual Commun. Image Processing*, pages 724–732, Nov. 1989.

[81] Y.-Q. Zhang and S. Zafar. Motion-compensated wavelet transform coding for color video compression. *IEEE Trans. Circuits Syst. Video Technol.*, 2(3):285–296, Sep. 1992.