New Techniques for the Design and Implementation of Efficient Full-Search Algorithms for Block-Matching Motion Estimation

Chun Yang

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montréal, Québec, Canada

September 2007

Your file Votre référence
ISBN: 978-0-494-31144-8
Our file Notre référence
ISBN: 978-0-494-31144-8

ABSTRACT

New Techniques for the Design and Implementation of Efficient Full-Search Algorithms for Block-Matching Motion Estimation

Chun Yang, Ph.D.
Concordia University, 2007

The block-matching motion estimation (BME) is one of the most commonly used techniques for digital video compression in low to moderate bit rate environments. The full search for block-matching motion estimation, as compared to a partial search, provides a higher motion estimation accuracy, yet its computational cost is generally high. Hence, developing new techniques for an efficient implementation of full-search algorithms is of practical significance for the BME.

In this thesis, a new full search algorithm is proposed, wherein the mean squared error (MSE) is used as the matching criterion to provide a higher motion estimation accuracy for the BME than that by any algorithm based on the most commonly-used mean absolute difference. It is shown that the computation of the MSE in the Haar wavelet domain results in a computational complexity that is much lower than or of the same order as that of the best-performing full search algorithms available in the literature.

A new approach has been developed for the multi-reference-frame block-matching motion estimation, wherein a full search is performed in the spatial domain of the multi-

reference-frame memory, and an early termination is imposed in the temporal domain using a novel strategy. It is shown that the computational complexity of the proposed full search method is significantly lower than that of any existing full search technique, and yet has a motion estimation accuracy which is about the same as that of the latter.

A new pseudo-spiral-scan data input scheme has been proposed, which can be used in any existing hardware architecture for the implementation of the successive-elimination-based block-matching motion estimation. This scheme results in significant power savings compared to the conventional raster-scan data input scheme. Several designs to implement the successive elimination algorithm have been given, some of which are shown to provide additional power savings.

To my family

and

To Mr. & Mrs. Yun Poy Lee

ACKNOWLEDGEMENTS

I would like to thank my research supervisors, Dr. M. Omair Ahmad and Dr. M.N.S. Swamy, for their constant support and guidance throughout this valuable learning and research experience.

I would also like to thank Concordia University for providing an excellent academic setting.

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

BME	Block-matching Motion Estimation
$BMAD_C$	Mean absolute difference between C and its matching block
CIF	Common Intermediate Format
DWT	Discrete Wavelet Transform
FGSE	Fine Granularity Successive Elimination
FSSA	Full-Search Systolic Array
g	A group of filters used to construct look-up tables
h_φ, h_ψ	Filters for discrete wavelet transform
L_n	*n-th* level lower bound for the block-matching error
MAD	Mean Absolute Difference
MFBME	Multi-Frame Block-matching Motion Estimation
MSE	Mean Square Error
MSEA	Multi-level Successive Elimination Algorithm
MV	Motion Vector
MV_t	Temporal Motion Vector
NS_E	Number of squaring operations required before elimination
P_i	Percentage reduction in the block-matching error obtained by increasing the depth of the reference frame memory from *i-1* to *i*
P_{sum}	A unit of power consumption
PDE	Partial Distortion Elimination
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format
$R_j(i)$	*j-th* block in the search window of the *i-th* reference frame
SAD	Sum of Absolute Differences
SEA	Successive Elimination Algorithm
SIF	Source Input Format
SS_M	Sum of squares for components in matrix M
T_{ref}	Reference frame number related to a statistical threshold
ρ_{av}	Average reduction in block-matching error due to the use of multiple reference frames

Chapter 1

Introduction

1.1 General

A diverse range of video services and products are becoming increasingly popular and available, and these include video telephony and conferencing, digital broadcasting, multimedia authoring and editing, multimedia content playback etc., which play an important role in the various industries such as telecommunications, entertainment, consumer electronics and information technology [1–4]. The remarkable progress in the video services and products poses an ever-growing demand for storage and bandwidth [5], which calls for more effective digital video compression. For this reason, digital video compression has not only been a major focus of research activity for the past few decades, but also continues to attract a great deal of research efforts to support the on-going evolution of the video services, products and their standardization.

A digital video sequence consists of individual frames appearing at regular time intervals. In general, the spatial and temporal redundancies existing in a digital video sequence can be reduced by the use of intra-frame and inter-frame video compression [6–8]. The intra-frame approach exploits the spatial redundancies by employing

the still-image compression techniques [9–13] on a frame-by-frame basis, whereas the inter-frame approach [14, 15] exploits both the temporal and spatial redundancies by considering more than one frame at a time. The latter approach can offer a higher compression efficiency than the former one can, and is predominantly used in low to moderate bit rate environments [16–24].

A practical inter-frame video compression system adopted by many video compression standards carries out the following operations [16, 17, 19–22].

1. *Frame segmentation*: A target video frame is partitioned into multiple blocks.

2. *Block-matching motion estimation*: For each of the target blocks, a matching block is identified in a reference frame. This block is used as the prediction for the target block. The displacement between the target block and its prediction is referred to as the motion vector of the block in question.

3. *Transformation of the prediction error*: The prediction error, which is the difference between the target block and its prediction, is transformed using a 2-D transformation such as the the 2-D discrete cosine transform.

4. *Encoding*: The motion vectors and the prediction errors are entropy-encoded for transmission or storage.

At the receiver end, the following operations restore the video frame.

1. *Decoding*: The motion vectors and prediction errors are decoded from the data stream.

2. *Restoration of the prediction error*: The prediction error is restored by the 2-D inverse transform.

3. *Motion compensation*: The predicted block is retrieved from the reference frame using the motion vector. The target block is then restored by adding the prediction error to the block prediction.

In the above operations, the block-matching motion estimation [25–28] is the most crucial step in video compression, which is achieved by reducing the temporal redundancy of a video sequence, and has received a lot of research attention. In the following, an overview of the block-matching motion estimation is presented.

1.2 An Overview of the Block-Matching Motion Estimation

The block-matching motion estimation (BME) [29] is a process of searching, within a reference frame, for a block that best matches the given block in the current frame. Every BME technique endeavours to achieve a high motion estimation accuracy along with a low computational cost. However, these two goals cannot be simultaneously reached, since a higher motion estimation accuracy is usually achieved at the expense of an increased computational cost, and a reduced computational cost often results in a lower motion estimation accuracy. The various BME techniques have, therefore, placed different emphases on these two goals, and they differ from one another in their search algorithms, matching criteria and the size of the block employed. The search algorithm is the main factor that determines both the motion estimation accuracy and the computational cost of a BME process.

Numerous search algorithms have been proposed since the introduction of the BME. A majority of them [30–43] substantially reduce the computational cost of the BME by conducting a partial search within a search window of the candidate

blocks. In general, the partial search is conducted within the central part of the search window, based on the assumption that most of the motion vectors are center-biased [33]. In the case of minor motions, it can provide acceptable motion estimation accuracy, but as the motion intensity increases, the accuracy deteriorates, since the searches are more likely to be trapped in some local minima [44]. In such a case, the motion estimation accuracy of the BME is sacrificed by the partial search algorithms in exchange for a lower computational cost.

In another group of search algorithms [45–57], the motion estimation accuracy of the BME takes precedence, and a full search over all the candidate blocks is carried out to ensure the most accurate motion estimation. At the same time, the computational complexity of the full search is reduced by simplifying the criterion used for the evaluation of the block matching. With the recent developments in fast full-search algorithms, especially the successive elimination algorithms [46, 47, 58], the amount of computational complexity of the full search can be reduced to the same order of magnitude as that resulting from a partial search. This has made the full search an attractive alternative to the partial search in software implementations of the BME, and hence, developing new full search algorithms with a reduced computational complexity is of considerable interest.

For a hardware implementation of the BME [59–62], it is the full search, rather than the partial one, that is predominantly used [62], because the pipelining technique [63, 64] can be used to achieve a processing speed that is not likely to be surpassed by any implementation using a partial search. Although the pipelining technique provides a high processing speed, it does not reduce the computational cost in terms of the power consumption, an issue that has become more relevant in recent years due to the increased applications of portable devices. It is, therefore, of a practical value to develop power-saving techniques for full-search hardware implementations [65].

1.3 Scope and Organization of the Thesis

As discussed in the previous section, the full search method for block-matching motion estimation is suitable for both software and hardware implementations. Reducing the computational cost is essential for both types of implementations. In the case of software implementations, the challenge is to reduce the computational complexity of the search algorithms, whereas in the case of hardware implementations, the challenge is to reduce the power consumption associated with the full search. Motivated by these challenges, this thesis focuses on developing new techniques for the design and implementation of efficient full-search algorithms for the block-matching motion estimation. The thesis is organized as follows.

In Chapter 2, some fundamental concepts of the block-matching motion estimation are highlighted, and the full-search-based software and hardware implementations for the BME are reviewed in order to provide the background material necessary for the work undertaken in this thesis.

A new full-search algorithm for block-matching motion estimation is developed in Chapter 3. The algorithm uses the mean squared error (MSE) as the matching criterion, which has the advantage of providing a higher motion estimation accuracy for the block-matching motion estimation than that provided by the commonly-used mean absolute difference (MAD). The proposed algorithm is based on the Haar wavelet transform, and computes the MSE in the wavelet domain to eliminate the impossible motion vector candidates, taking advantage of the coefficient sorting feature provided by the Haar wavelet transform. The computational complexity required by the proposed algorithm is compared with that of some of the bench-mark algorithms.

In Chapter 4, a new approach to the multi-reference-frame block-matching motion estimation (MFBME) is presented, in which the fast full search is conducted in the

spatial domain, and the computational complexity in the temporal domain is reduced by using a full search with an early termination. Statistical experiments are first conducted to show that scanning the multi-frame memory with an early termination is a feasible way to reduce the temporal search complexity of the MFBME. An early termination method that keeps track of the block-matching error, reference frame by reference frame, and terminates the temporal search using a block-matching-error-dependent strategy is then proposed. The results based on extensive simulations show that the proposed method can significantly reduce the computational complexity of the MFBME with no or very little loss in the motion estimation accuracy for the MFBME.

In Chapter 5, a new data input scheme is proposed for a successive-elimination-based hardware implementation of the BME for the purpose of reducing the power consumption. Several designs to implement a successive elimination algorithm are developed based on a newly introduced block segmentation scheme. The reduction in the power consumption is studied using a power simulator. It is shown that using the proposed data input scheme provides significant savings in the power consumption compared to that using the conventional data input scheme.

Chapter 6 concludes the thesis by highlighting the contributions of this research and by suggesting some of the issues arising from this research that can be further investigated.

Chapter 2

Block-Matching Motion Estimation Using Full Search

2.1 Introduction

Block-matching motion estimation (BME) is an effective technique to reduce the temporal redundancy of digital video sequences, and has been adopted by a number of international digital video standards such as MPEG-1 [16], MPEG-2 [17], MPEG-4 [19] and ITU-T recommendations H.261 [20], H.263 [21] and H.264 [22]. The problem of block-matching motion estimation can be posed as follows [29].

Given a current block C of size $N_1 \times N_2$ in a target frame, and a set of reference blocks of the same size, denoted as $R_v, v \in S$, in another frame, where v represents a displacement vector between the reference block and C, and S is a set of vectors, a matching block of C is to be identified among this set of reference blocks such that a specified matching criterion is satisfied. The corresponding displacement vector is referred to as the motion vector of the current block C [44].

The matching of the blocks can be evaluated using various metrics, the mean square error (MSE) and the mean absolute difference (MAD) being the two most commonly used ones [44].

The MSE between C and a reference block R_v is given by

$$MSE(C, R_v) = \frac{1}{N_1 \times N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (C(i,j) - R_v(i,j))^2 \tag{2.1}$$

The motion vector for C is taken to be the value of v which minimizes the MSE, that is,

$$\overline{v} = arg \min_{v} MSE(C, R_v) \tag{2.2}$$

and $R_{\overline{v}}$ is used as a prediction of C.

The MAD between a given block C and a reference block R_v is given by

$$MAD(C, R_v) = \frac{1}{N_1 \times N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} |C(i,j) - R_v(i,j)| \tag{2.3}$$

The motion vector for C is taken to be the value of v which minimizes the MAD, that is,

$$\overline{v} = arg \min_{v} MAD(C, R_v) \tag{2.4}$$

and $R_{\overline{v}}$ is used as a prediction of C.

A variant of the MAD is the sum of the absolute differences (SAD), which is related to the MAD by

$$SAD(C, R_v) = N_1 \times N_2 \times MAD(C, R_v)$$

and is used in situations where the division by $N_1 \times N_2$ is not desirable. The metric MAD/SAD is the one which is used in the case of a VLSI implementation of the

BME, but it is well-known that the performance of this metric deteriorates as the search area becomes larger due to the presence of multiple local minima [44].

The accuracy of the motion estimation can be evaluated by the peak signal-to-noise ratio (PSNR) [44] between the target frame and the motion-predicted frame formed by the various block predictions resulting from the motion estimation process. The PSNR is given by

$$PSNR = 10\log_{10}\frac{255^2 \times N_1 \times N_2 \times N_f}{\sum_C \sum_{i=1}^{N_1}\sum_{j=1}^{N_2}[C(i,j) - R_{\overline{v}}(i,j)]^2} \tag{2.5}$$

for frames containing N_f blocks of $N_1 \times N_2$ pixels represented by an 8-bit luminance intensity. A comparison between (2.5) and (2.1) indicates that the minimization of the MSE also maximizes the PSNR.

2.2 Full Search Algorithms

The computational complexity incurred by the minimization process given in (2.2) or (2.4) is so high that an exact software implementation of (2.2) or (2.4) is prohibitive. However, there are several full search algorithms [45–48, 52–55] that can significantly reduce the computational complexity of the full search by utilizing certain lower bounds of the MAD/MSE to eliminate the impossible candidate motion vectors. These lower bounds require much less computations than the MAD/MSE does, thus resulting in a reduced computational complexity. We now give an overview of these elimination-based full search algorithms.

Successive Elimination Algorithms

The earliest successive elimination algorithm (SEA) [45] is based on a lower bound of the MAD, the computation of which is simpler than that of the MAD itself. Given the current block C and a reference block R_v, both of size $N \times N$, this lower bound is given by [45]

$$\begin{aligned} L_1(C, R_v) &= \frac{1}{N^2} \left| \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} C(n_1, n_2) - \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} R_v(n_1, n_2) \right| \\ &\leq MAD(C, R_v) \end{aligned} \tag{2.6}$$

The value of $\sum_{n_1=1}^{N} \sum_{n_2=1}^{N} C(n_1, n_2)$ is fixed for any two reference blocks, whether they overlap or not, whereas in $\sum_{n_1=1}^{N} \sum_{n_2=1}^{N} R_v(n_1, n_2)$, certain terms are shared if the reference blocks overlap. Therefore, $L_1(C, R_v)$ requires less operations than the MAD does. For each given candidate motion vector, if this lower bound is greater than the current minimum MAD, the given candidate motion vector is eliminated; otherwise, the MAD is computed. Since, in this way, the majority of the candidate motion vectors are eliminated, the computation of their corresponding MAD is avoided, thus reducing the overall computational complexity.

The multi-level successive elimination algorithm (MSEA) [46] extends the SEA to a multi-level framework, with one lower bound for each level. The process of the MSEA is illustrated by a flow chart in Fig. 2.1, where these lower bounds are checked sequentially. If a lower bound is greater than the current minimum MAD, the current candidate motion vector is eliminated; otherwise, the next lower bound is checked.

As a more general definition than the one in (2.6), the lower bound of the MAD at level l may be given as follows [46]. Let C and R_v be divided into sub-blocks of size $2^{l-1} \times 2^{l-1}$, with $l = 1, 2, ..., n$, and $2^n = N$. Also, let the sum of each of the

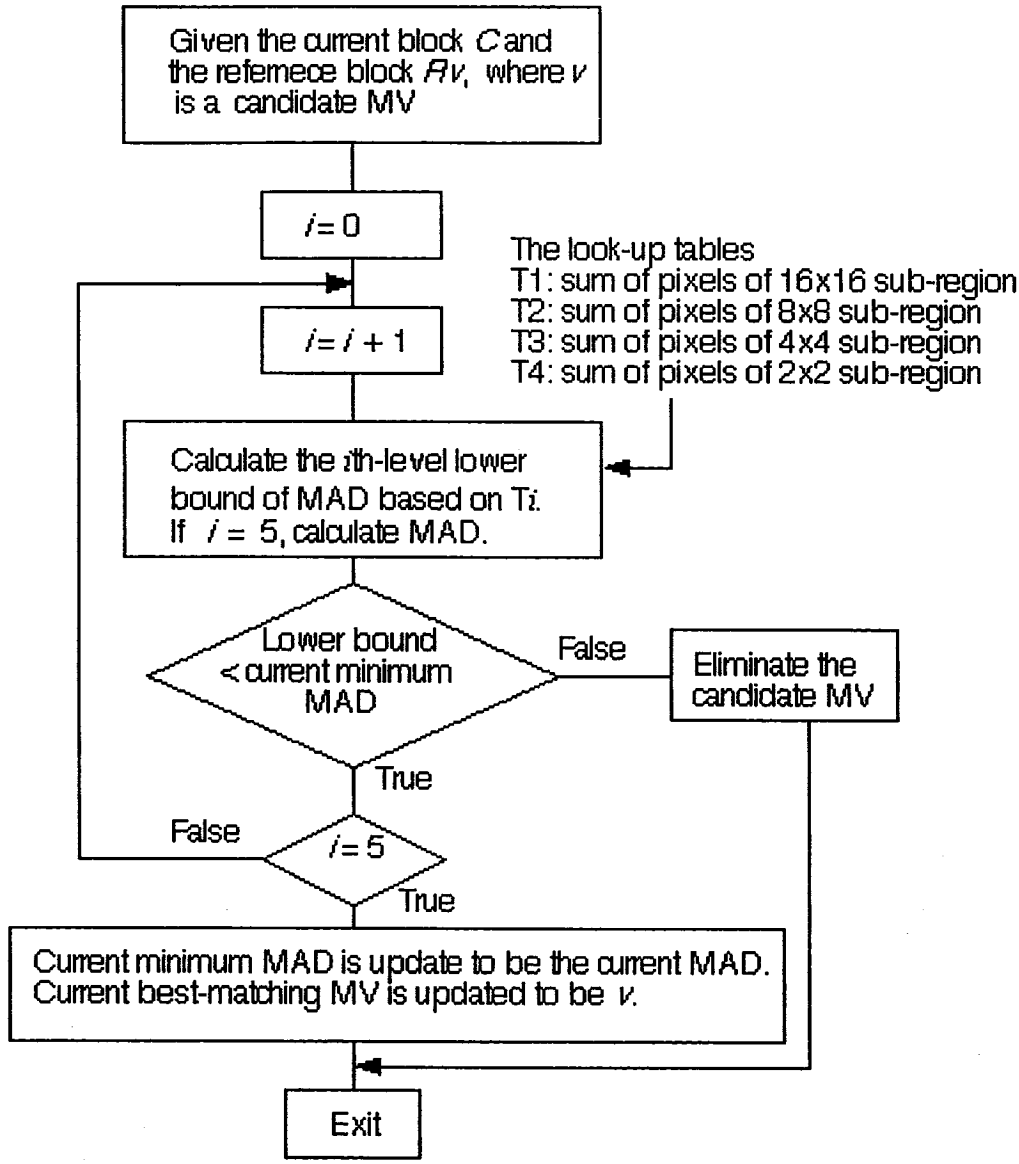


Figure 2.1: Flow chart illustrating the process of the multi-level successive elimination algorithm given in [46].

sub-blocks of C and R_v be stored in matrices SC_l and SR_l, respectively. The *l-th* lower bound is then given by

$$L_l(C, R_v) = \frac{1}{N^2} \sum_{n_1=1}^{N/2^{l-1}} \sum_{n_2=1}^{N/2^{l-1}} |SC_l(n_1, n_2) - SR_l(n_1, n_2)| \tag{2.7}$$

The lower bounds given in (2.7) satisfy

$$L_1(C, R_v) \leq L_2(C, R_v) \leq \ldots \leq L_n(C, R_v) \leq MAD(C, R_v) \tag{2.8}$$

In (2.7), SC_l is fixed for any given R_v, and any two overlapping R_v's may have overlapping elements in SR_l. To avoid duplicate calculations for these overlapping elements, the sum of each of the $2^{l-1} \times 2^{l-1}$ sub-regions of a given reference frame is calculated once for all, and stored in several look-up tables.

The generalization of the inequality from (2.6) of the SEA to (2.8) of the MSEA shows that a set of lower bounds with finer granularity may increase the number of eliminations. This motivated the development of another algorithm called the fine granularity successive elimination (FGSE) algorithm proposed in [47], which extends the inequality of (2.8) to

$$\begin{aligned} L_1(C, R_v) &\leq L_2(C, R_v) \leq L_2^1(C, R_v) \leq \ldots \leq L_2^3(C, R_v) \\ &\leq \ldots \\ &\leq L_n(C, R_v) \leq L_n^1(C, R_v) \leq \ldots \leq L_n^{2^n-1}(C, R_v) \\ &\leq MAD(C, R_v) \end{aligned} \tag{2.9}$$

with a computational complexity lower than that of the MSEA.

Partial Distortion Elimination

The metric used for block matching, whether it be the MAD or the MSE, is the sum of $N \times N$ non-negative terms, where $N \times N$ is the block size. The sum of

any sub-set of these terms is sometimes referred to as the partial distortion in the literature [48–51, 55]. Since the partial distortion not only has a smaller value than the metric of block matching, but also requires less amount of computation, it can be checked prior to the computation of the metric of block matching. If it is greater than the current minimum MAD/MSE, the corresponding candidate motion vector is eliminated and the computation of the metric of block matching is avoided. As illustrated in Fig. 2.2, this process usually starts from the partial distortion that contains only one square term of the difference block and updates it by adding one more term at a time. The partial distortion is continuously compared with the current minimum MAD/MSE until the former becomes greater than the latter, or all the terms are exhausted.

Computational Complexity

Compared to an exact execution of the minimization process defined in (2.2) or (2.4), the speed-up provided by the various partial-distortion-based full search algorithms in the literature is in the range of 4 to 12 [48], and that provided by the successive-elimination-based algorithms is in the range of 2 to 46 [46, 47], as compared to about 40 by the three-step partial search [32] for various test sequences and a search window of ± 15 . This indicates that the full search, especially that based on the successive elimination, has a potential for providing a speed-up comparable to that of the partial search. If the computational complexity of the full search can be further reduced, its superior motion estimation accuracy for the BME along with its low computational complexity should make it an attractive alternative to the partial search.

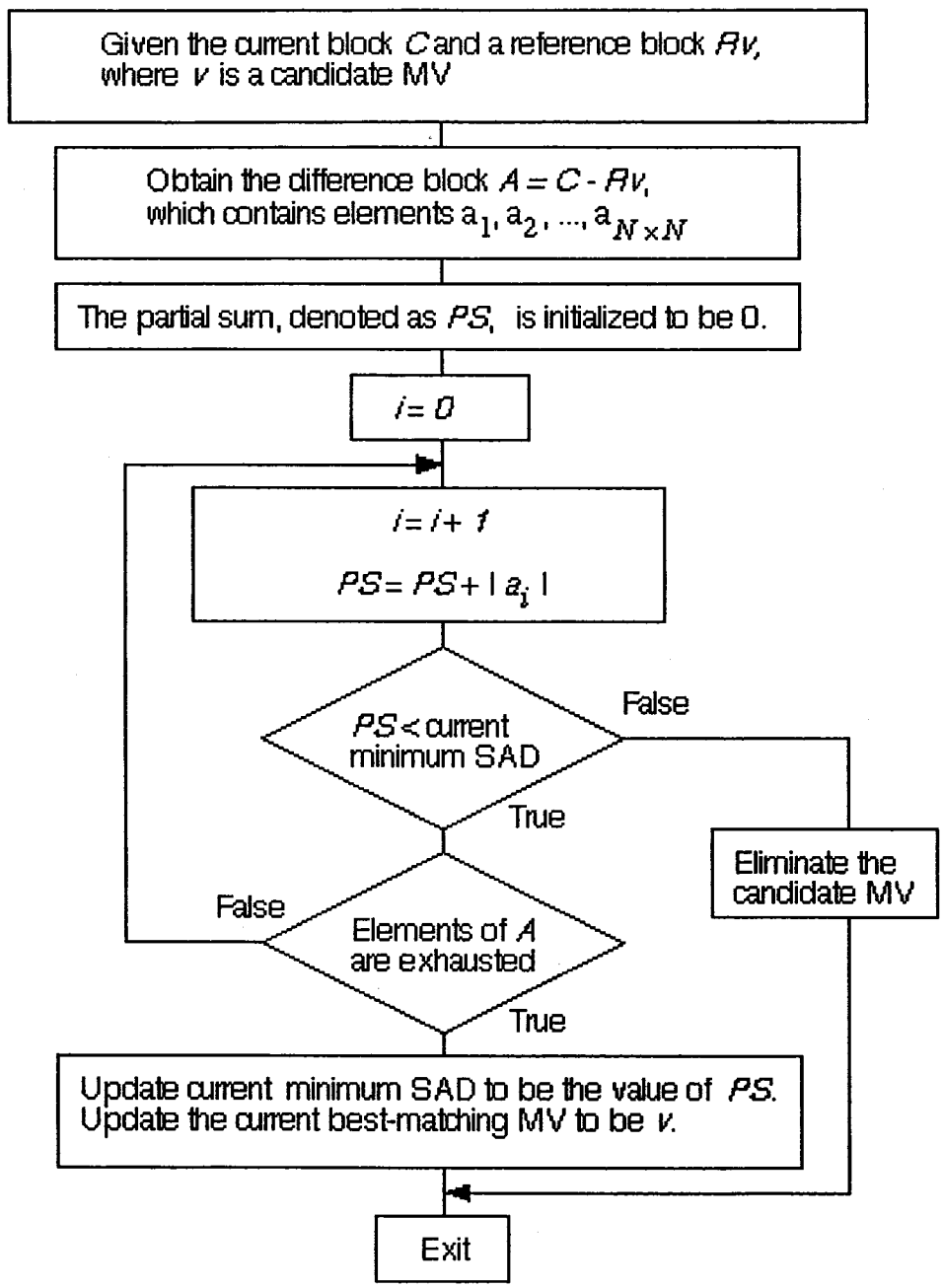


Figure 2.2: Flow chart illustrating the partial distortion elimination algorithm

2.3 Implementation of the Block-Matching Motion Estimation

The full search algorithms, such as those mentioned in the previous section, provide algorithmic-level speed-up in the software implementation of the BME. Further speed-up could be achieved at the implementation level by taking advantage of the parallelism provided by the current computer processors [58, 66–71].

For hardware implementation of the BME, the regularity of the data flow has given rise to several architectures based on parallel processing and pipelining [64, 72–75]. A 2-D full search systolic array (FSSA) [64], such as the one shown in Fig. 2.3, significantly reduces the processing time of a full search by employing the pipelining technique. As shown in this figure, the architecture contains an array of absolute difference units, the results of which are column-wise accumulated, and then added to compute the SAD. Within a rectangular search window, the reference blocks along the same row are input to the FSSA sequentially, so that the pixels shared by the overlapping reference blocks need not be input repeatedly, resulting in a processing throughput of one candidate motion vector per clock cycle.

Although the use of the pipelining technique provides a high processing speed, it does not reduce the power consumption associated with a full search. In [65], the SEA [45] is applied to the 2-D FSSA for the purpose of reducing the power consumption.

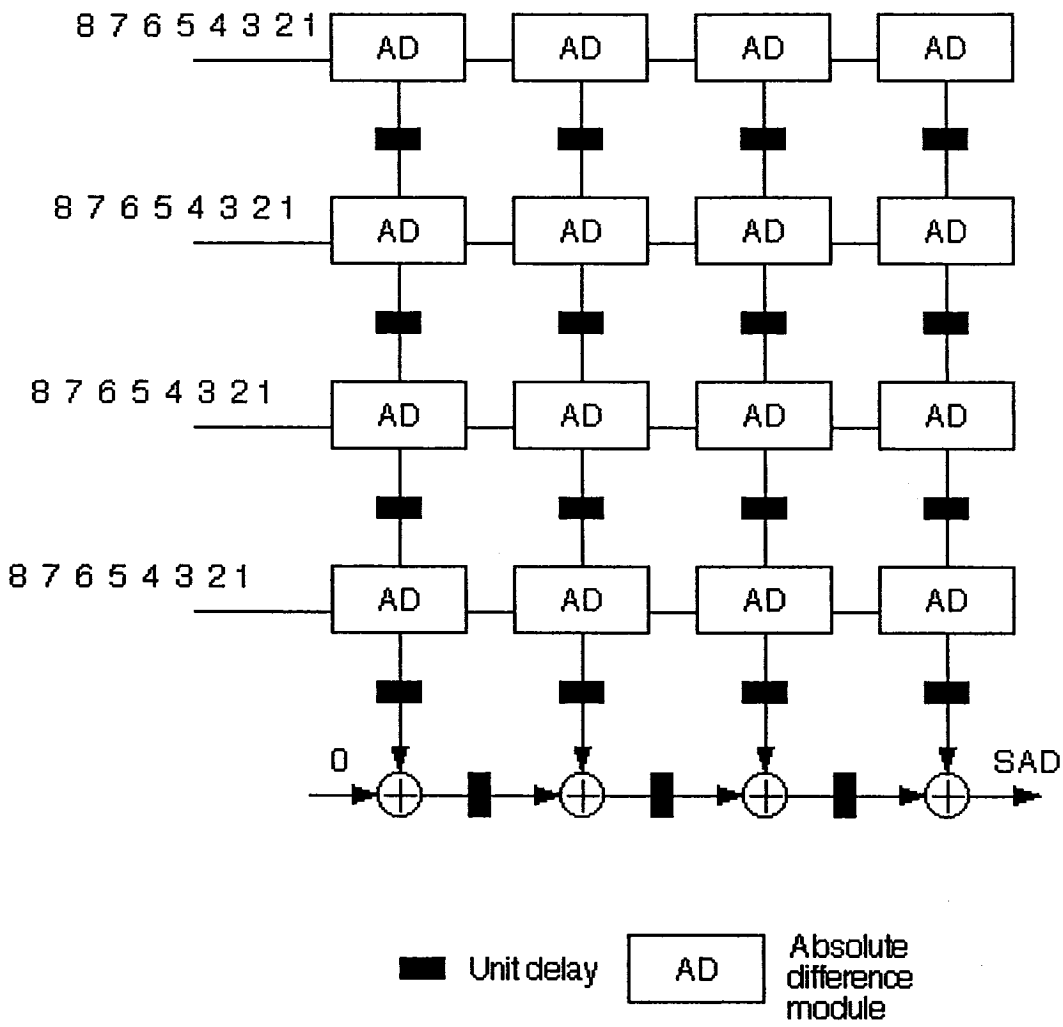


Figure 2.3: A 2-D full search systolic array given in [64]

2.4 Summary

In this chapter, two major categories of full search algorithms, namely, the elimination-based and the partial-distortion-based full search, are reviewed to provide their relevance to the work contained in the later chapters of this thesis. The computational complexity of these algorithms has been analysed, indicating that the full search, especially that based on the successive elimination, has a potential for providing a speed-up comparable to that of the partial search. Both the software and hardware implementations for the BME has also been discussed.

Chapter 3

Fast Full-Search Block-Matching Motion Estimation using 2-D Haar Wavelet Transform

3.1 Introduction

As mentioned in Chapter 2, most of the fast full-search block-matching motion estimation (BME) methods reduce the computational complexity by eliminating the impossible motion vector (MV) candidates. The various elimination methods are based on one of the following two major techniques.

- *The successive elimination algorithm (SEA)* [45] *and multiple-level SEA (MSEA)* [46]: These algorithms are based on generating a non-descending series of lower bounds for the measure of matching. If a lower bound is found to be greater than the current minimum, the corresponding MV candidate is eliminated without the need for an exhaustive calculation of the measure of matching. In [47],

the lower bound series is extended to finer granularity.

- *The partial distortion elimination (PDE)*: The measure of block matching consists of only non-negative terms. These terms as well as the sum of their combinations can be used as the lower bounds for the elimination of a number of MV candidates. The complexity of this type of elimination is directly related as to how many terms are used before the elimination can be made. It is desirable that the terms with the greater contribution to the measure of matching are used first, so that the elimination can occur earlier [48].

The mean squared error (MSE) and mean absolute difference (MAD) are the most commonly used criteria for block matching. Minimizing the MSE leads to the maximization of peak-signal-to-noise ratio (PSNR) [44], which is used to evaluate the goodness of block-matching motion estimation. However, the calculation of the MSE requires a large number of multiplications, resulting in a high computational complexity. On the other hand, the MAD is preferred in most of the BME methods in view of its simplicity, in spite of the fact that the resulting PSNRs are suboptimal.

In this chapter [76], we present an elimination-based full search BME algorithm that uses the MSE as the block matching criterion and carries out the PDE in the Haar wavelet domain (HaarPDE). The wavelet domain provides the coefficient sorting that is desirable for the PDE and leads to a multi-level structure similar to that in the MSEA. The Haar wavelet is used, since it requires only basic operations, such as addition, subtraction and shift. It will be shown that in addition to a small fixed overhead, the number of multiplications required is drastically reduced, making the computational complexity of the HaarPDE significantly lower than that of the PDE algorithms operating in the spatial domain and lower than or comparable with that

of the MSEA.

This chapter is organized as follows. Section 3.2 gives a brief review of the MSE-based full search algorithms. In Section 3.3, we present the proposed algorithm for partial distortion elimination using the Haar wavelet transform. In Section 3.4, a fast scheme for implementing the Haar wavelet transform in the proposed algorithm is discussed and its overhead cost analysis performed. Section 3.5 presents the simulation results with regard to the overall computational complexity of the proposed algorithm and compares it with those of other full-search BME methods. It also contains a study of the rate-distortion optimization of the various algorithms. Finally, certain conclusions are drawn in Section 3.6.

3.2 Full-Search Block-Matching Motion Estimation Using the Mean Squared Error

The block matching criteria using the MSE and the MAD are given in (2.1) and (2.3) in Chapter 2. The use of the MSE is expected to result in higher PSNRs than that using the MAD. For example, a comparison of the PSNR achieved by the full search using the MSE and that using the MAD is given in Table 3.1 for seven sequences. These results are obtained by using a block size of 16×16 and a search range of ± 15 pixels; details of the seven sequences used are given in Table 3.2. As expected, the MSE-based full search provides a higher PSNR for all of the test sequences.

The computational complexity of the MSE can be reduced by simplifying the calculation of the squared error terms in (2.1). The authors of [77] have proposed an algorithm that decomposes the squared error terms into three parts: squares of

Table 3.1: Comparison of the average PSNRs provided by the MSE-based and MAD-based full search algorithms for the seven sequences

Sequence	PSNR in dB using MSE	PSNR in dB using MAD	Avg. diff. per frame	Max. diff. for a single frame
Foreman	32.01	31.82	0.19	1.27
Mother&Daughter	41.05	41.02	0.03	0.62
Car Phone	31.64	31.51	0.13	0.47
Container	43.15	43.09	0.06	1.64
Football	23.06	22.87	0.19	0.36
Flower Garden	23.87	23.79	0.08	0.26
Coast Guard	30.61	30.48	0.13	0.58

Table 3.2: Details of the seven sequences used in the simulations

Sequence	Frames	Type	Resolution
Foreman	2-400	QCIF	176x144
Mother&Daughter	2-400	QCIF	176x144
Car Phone	2-380	QCIF	176x144
Container	2-300	QCIF	176x144
Football	2-125	SIF	352x240
Flower Garden	2-115	SIF	352x240
Coast Guard	2-300	CIF	352x288

the elements of the reference block, squares of the elements of the current block and cross terms between the elements of the two blocks. The reduction in the computational complexity results from using fast 2-D FIR filtering for the cross terms and the fact that the squared terms of the adjacent reference blocks share overlapped elements. The speedup achieved in [77] is not as significant as in the elimination-based algorithms, since the MSE corresponding to each of the MV candidates has to be calculated.

The computational complexity can also be reduced by eliminating the impossible MV candidates. An elimination-based algorithm has been proposed in [53] that uses three different lower bounds for the MSE. These are the so-called sum of squared vertical projections, sum of squared horizontal projections and sum of squared massive projection [53]. If any of these three bounds is found to be greater than the current minimum MSE, the corresponding MV candidate is eliminated. The authors in [52] provide another elimination-based algorithm, wherein the lowest bound is identical to that derived from the massive projection in [53]. The algorithm finds tighter bounds by splitting a given block into sub-regions. The authors in [52] also derive an equation involving the MSE and its lower bounds, so that the PDE can be used in the calculation of the MSE. The concept used in [52] is analogous to that in the MSEA [46] in that it generates tighter lower bounds by dividing the blocks into sub-regions. However, due to complexity considerations, only one level of sub-regions has been used in [52].

In the following, we show that a more general framework than those in [53] and [52] can be established through the Haar wavelet transform. The number of levels can be extended to more than two without a significant increase in the computational

complexity. At the same time, it is shown that the MSE can be expressed in the wavelet domain in terms of sorted coefficients, so that the impossible MVs can be eliminated earlier than is done in the normal PDE process.

3.3 Calculation of the Mean Squared Error in the Wavelet Domain

Let M be a matrix of size $K \times K$, where $K = 2^n$, n being an integer. Let $DWT(M)$ denote the one-scale 2-D discrete wavelet transform [78] of M, and SS_M the sum of the squared value of each of the components in M.

For the current block C and a reference block R, the MSE is equal to

$$MSE(C,R) = \frac{1}{N^2} SS_A$$

where $A = C - R$. If we take the 2-D discrete wavelet transform (DWT) of the difference block using any of the orthonormal wavelets, the orthonormality ensures that SS_A is unchanged in the wavelet domain [79]. Therefore, the MSE can be calculated as

$$MSE(C,R) = \frac{1}{N^2} SS_A = \frac{1}{N^2} SS_{DWT(A)}$$

In the wavelet domain, the coefficients are naturally sorted, since most of the larger-magnitude coefficients are concentrated in the lower-frequency sub-regions. Performing the PDE in the wavelet domain and giving a higher priority to the coefficients in the lower-frequency sub-regions can therefore speed up the elimination of the impossible MV candidates. We use the Haar wavelet to perform the PDE, since it

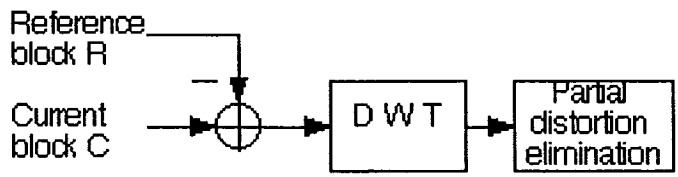


Figure 3.1: Block diagram illustrating the partial distortion elimination using the Haar wavelet

requires only basic operations, such as addition, subtraction and shift. Fig. 3.1 is an illustration of such a PDE scheme.

For a typical 16×16 difference block A, a four-scale Haar wavelet hierarchy is as shown in Fig. 3.2 [79], where

$$\begin{aligned} h_\varphi(m) &= \{1,1\} \\ h_\psi(m) &= \{1,-1\} \end{aligned} \tag{3.1}$$

The output at each decimation block satisfies a relation of the form

$$y_\varphi(m) = x(2m) + x(2m+1) \tag{3.2}$$

or

$$y_\psi(m) = x(2m) - x(2m+1) \tag{3.3}$$

The wavelet representation of the difference block A can be expressed as

$$\begin{aligned} &A1,\ V1,\ H1,\ D1, \text{ each of size } 1\times 1, \\ &V2,\ H2,\ D2, \text{ each of size } 2\times 2, \\ &V3,\ H3,\ D3, \text{ each of size } 4\times 4, \text{ and} \\ &V4,\ H4,\ D4, \text{ each of size } 8\times 8 \end{aligned}$$

It is noted that A_4, A_3, A_2 and A_1 are the lower-frequency approximations of the

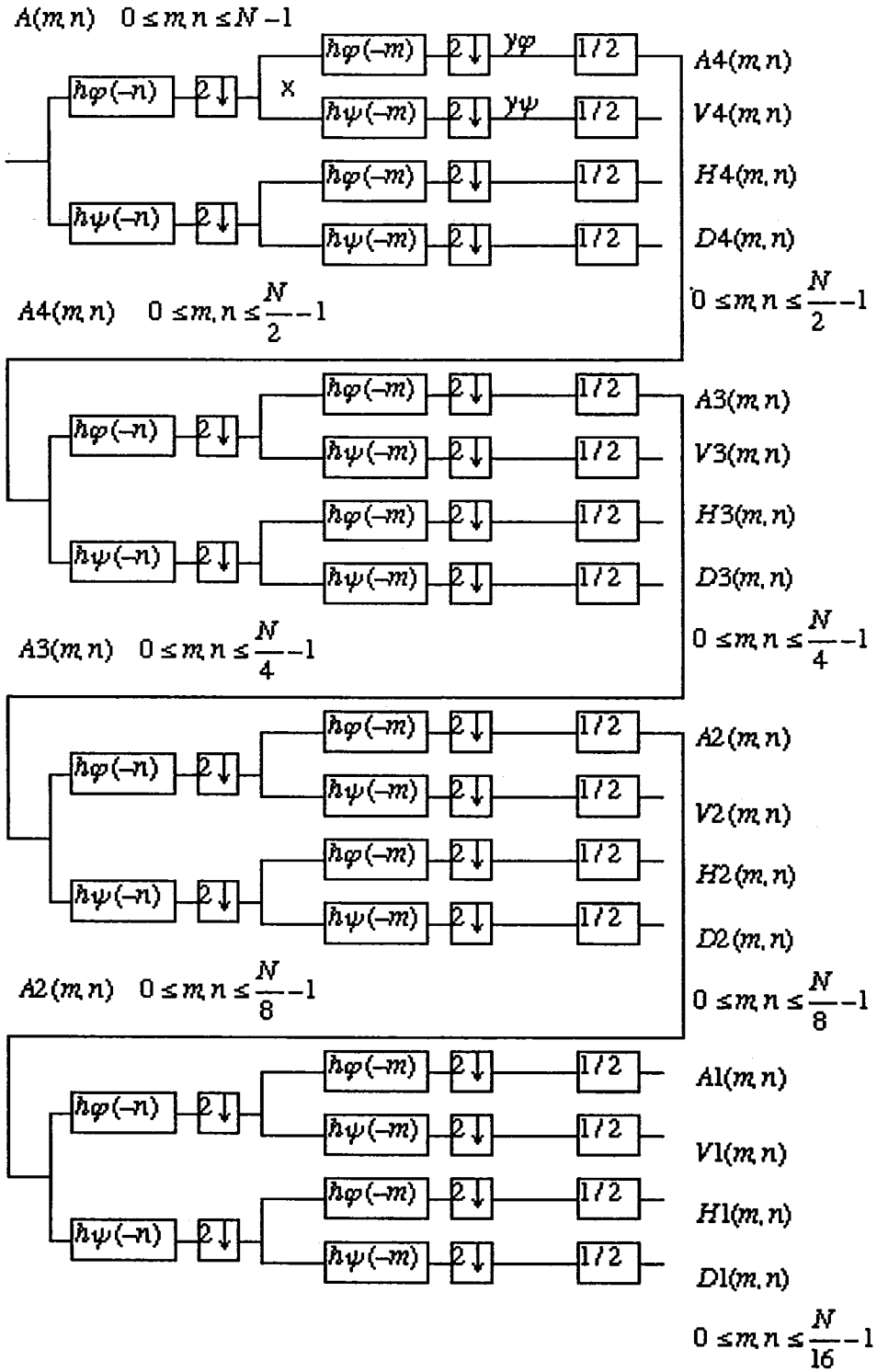


Figure 3.2: Four-scale Haar wavelet transform

input A, A_4, A_3 and A_2, respectively, and are likely to contain a majority of the larger-magnitude coefficients.

Using the following sum-of-square relations

$$MSE(C,R) = \frac{1}{N^2} SS_A \tag{3.4}$$

where

$$\begin{aligned} SS_A &= SS_{A4} + SS_{V4} + SS_{H4} + SS_{D4} \\ SS_{A4} &= SS_{A3} + SS_{V3} + SS_{H3} + SS_{D3} \\ SS_{A3} &= SS_{A2} + SS_{V2} + SS_{H2} + SS_{D2} \\ SS_{A2} &= SS_{A1} + SS_{V1} + SS_{H1} + SS_{D1} \end{aligned}$$

the expression $MSE(C,R)$ can be calculated as

$$\begin{aligned} MSE(C,R) = \frac{1}{N^2}[&(SS_{A1}) + (SS_{V1} + SS_{H1} + SS_{D1}) + (SS_{V2} + SS_{H2} + SS_{D2}) + \\ &(SS_{V3} + SS_{H3} + SS_{D3}) + (SS_{V4} + SS_{H4} + SS_{D4})] \end{aligned} \tag{3.5}$$

In (3.5), the contribution of the coefficients to $MSE(C,R)$ is expected to decrease progressively as we move from the term in the first parenthesis to those in the fifth, since, as mentioned earlier, most of the larger-magnitude coefficients are concentrated in the lower-frequency sub-regions. Therefore, in the proposed HaarPDE, we sort the MSE terms in (3.5) according to the following order: the term in the first parenthesis, namely, SS_{A1}, then the terms in the second parenthesis, the terms in the third and in the fourth, and finally those in the fifth. The partial distortion is initialized to be the first sorted term, SS_{A1}, which is compared with the current minimum MSE. If the partial distortion is greater than the current minimum MSE, the corresponding MV candidate is eliminated; otherwise, one of the terms in the second parenthesis is

added to the current partial distortion to update it. This procedure terminates either upon the elimination of the MV candidate or after all the terms in all the parentheses are exhausted. In the latter case, the resulting partial distortion is equivalent to the MSE associated with the MV candidate being examined. If the value of this MSE is greater than the current minimum MSE, the corresponding MV candidate is eliminated; otherwise, the value of this MSE becomes the current minimum MSE value.

3.3.1 Relation between the Proposed HaarPDE and Some Existing Algorithms

It is interesting to note that in Fig. 3.2, the outputs A_1, A_2, A_3, A_4 have a one-to-one correspondence with the the matrix hierarchy on which the MSEA [46] is based. For example, 16 times A_1 equals the top-level matrix of MSEA, eight times A_2 equals the second-level matrix of MSEA, and so on.

The lowest bound used in [52], or the sum of the squared massive projection in [53], is equivalent to SS_{A1}. The other lower bounds that could be obtained by using 4, 16 or 64 sub-regions in [52] correspond to SS_{A2}, SS_{A3} and SS_{A4}, respectively.

It is to be noted that two of the lower bounds in [53], namely, the sum of squared vertical projections and the sum of squared horizontal projections, are the same as those obtained from the top level of the 1-D version of the proposed HaarPDE, the Haar transform along the rows and columns, respectively.

3.3.2 Efficiency of the Proposed HaarPDE in Terms of Multiplications

The computational complexity of the MSE-based full search is mainly due to the requirement for multiplications. This section considers the efficiency of the proposed HaarPDE in terms of the number of multiplications required. The sequences used in the simulation are those listed in Table 3.2. To investigate the performance of the algorithms with different image sizes, more than two thousand frames are employed with QCIF, CIF and SIF formats. The sequences chosen are with different motion intensities ranging from minor (e.g., *Mother&Daughter*) to large motion (e.g., *Foreman*), and from mainly translational (e.g., *Container*, *Garden*) to complicated motion (e.g., *Football*, *Car Phone*, *Coast Guard*). The block size chosen is 16×16, the search range is ± 15 pixels, and the full search within the search range is conducted along a spiral path [48] starting from the center of the search window.

The choice of the initial minimum MSE is crucial to the performance of the full search. Since there exists a high correlation among the MVs of the adjacent blocks [80], we determine the initial minimum MSE by comparing the MSE of the three most likely MV candidates. A simple procedure to obtain this initial minimum MSE is given below. Let the current block be the p_yth block column-wise and the p_xth block row-wise. The MSE associated with the zero MV, the MV of the block at $(p_y, p_x - 1)$ and the MV of the block at $(p_y - 1, p_x)$ are compared. The smallest MSE is then used as the initial minimum MSE.

After obtaining the initial minimum MSE, the MV candidates in the search window are checked by the proposed HaarPDE algorithm following the spiral path. For each MV candidate, the Haar wavelet transform is performed over the difference

block. As described earlier, the wavelet coefficients are sorted and the partial sums updated on a term-by-term basis, until a decision can be made as to whether or not the MV candidate should be eliminated.

In order to compare these results, two MSE-based full search methods, namely, the full search with no early elimination of any MV (FS) and the spatial-domain PDE [48] are simulated. Table 3.3 gives the average number of squaring operations for each MV candidate. The speedup of the HaarPDE is approximately from 3 to 6 and from 27 to 107 as compared to those of the spatial-domain PDE and the FS, respectively. To further demonstrate the superiority of the HaarPDE over the spatial-domain PDE, we obtain for each given MV candidate, NS_E, the number of operations of squaring needed to compute the MSE before the candidate is eliminated by each of these two methods. A larger NS_E indicates more squaring operations are needed. Fig. 3.3 shows the histogram of the percentage numbers of MV candidates that are eliminated as a function of NS_E for the sequence *Foreman*, wherein the percentage number is calculated with respect to the total number of all the MV candidates to be considered, namely, $W_S \times N_B \times (N_F - 1)$, $W_S = (31 \times 31)$ being the size of the search window, $N_B = (9 \times 11)$ the number of blocks per frame and N_F the number of frames in the sequence. The histograms for the rest of the test sequences are shown in Fig. 3.4- 3.9, respectively. These figures clearly show that the largest elimination of the MV candidates occurs just after the first squaring operation; except for sequence *Mother & Daughter*, the number of such eliminations is much higher in the case of the HaarPDE than that in the case of the spatial-domain PDE.

Table 3.4 gives the breakdown on the percentage of the number of eliminated MV candidates at different levels of the Haar wavelet hierarchy for various sequences.

Table 3.3: Average number of squaring operations required for each MV candidate

	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
FS	256						
Spatial-domain PDE	22.5	20.0	22.9	11.8	47.0	28.9	26.5
Proposed HaarPDE	3.6	6.1	4.6	2.0	9.5	6.6	4.7

It is seen that most of the elimination occurs at the first two levels, and the total number of squaring operations required for these two levels is only four. This is the reason as to why the average numbers of squaring operations are in single digits for the various sequences in the proposed scheme, while the corresponding numbers are approximately 3 to 6 times in the case of the spatial-domain PDE and about 27 to 107 times for the case of the FS.

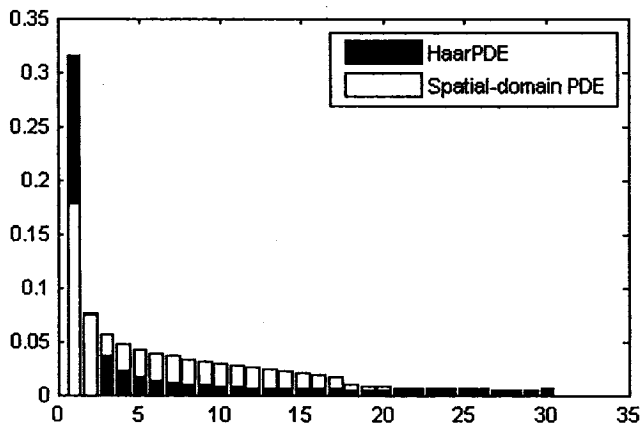


Figure 3.3: Histogram showing the percentage number of MV candidates eliminated Vs. NS_E for the HaarPDE and the Spatial-domain PDE with sequence *Foreman*. Values are shown only for the first 30 of the 256 squaring operations of the MSE.

Intuitively speaking, the number of the terms in the MSE that need to be computed is related to the gradient distribution of the MSE surface of the entire search window. The flatter the surface, the larger the number of terms to be computed in order to eliminate an impossible MV candidate. From this perspective, minor mo-

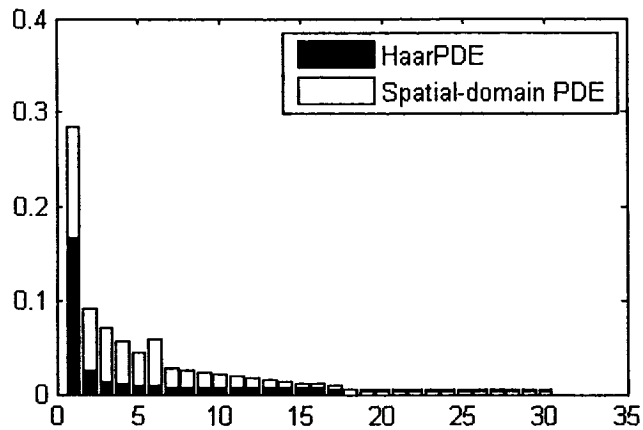


Figure 3.4: Histogram showing the percentage number of MV candidates eliminated Vs. NS_E for the HaarPDE and the Spatial-domain PDE with sequence *Mother & Daughter*. Values are shown only for the first 30 of the 256 squaring operations of the MSE.

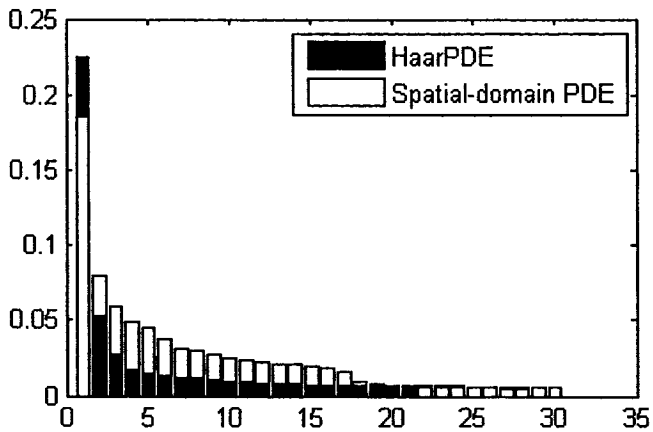


Figure 3.5: Histogram showing the percentage number of MV candidates eliminated Vs. NS_E for the HaarPDE and the Spatial-domain PDE with sequence *Car Phone*. Values are shown only for the first 30 of the 256 squaring operations of the MSE.

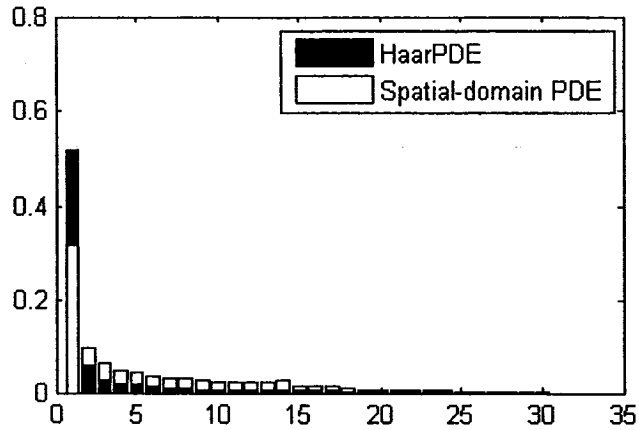


Figure 3.6: Histogram showing the percentage number of MV candidates eliminated Vs. NS_E for the HaarPDE and the Spatial-domain PDE with sequence *Container*. Values are shown only for the first 30 of the 256 squaring operations of the MSE.

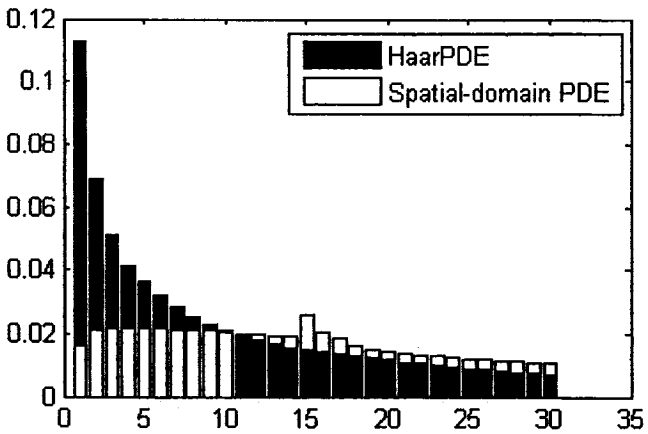


Figure 3.7: Histogram showing the percentage number of MV candidates eliminated Vs. NS_E for the HaarPDE and the Spatial-domain PDE with sequence *Football*. Values are shown only for the first 30 of the 256 squaring operations of the MSE.

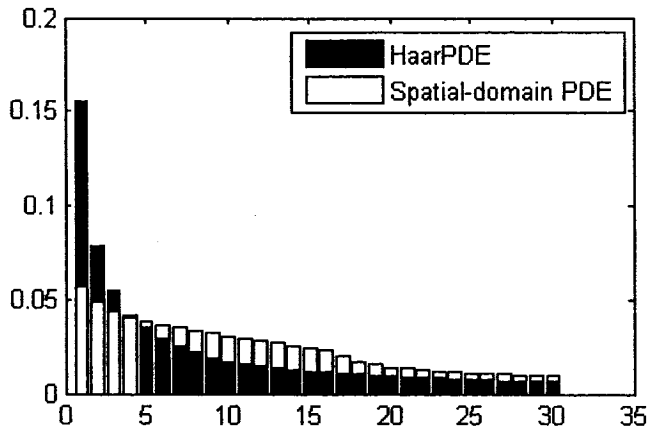


Figure 3.8: Histogram showing the percentage number of MV candidates eliminated Vs. NS_E for the HaarPDE and the Spatial-domain PDE with sequence *Flower Garden*. Values are shown only for the first 30 of the 256 squaring operations of the MSE.

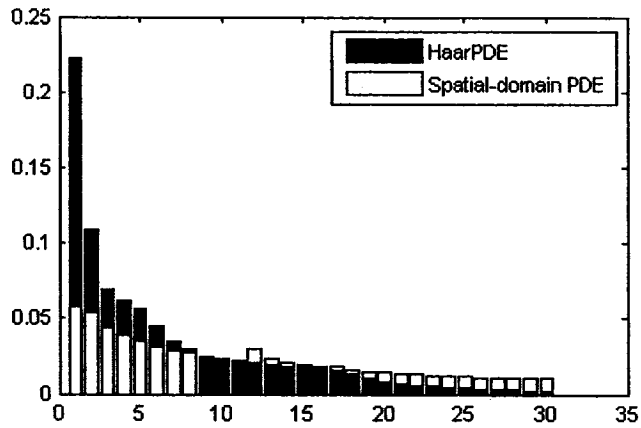


Figure 3.9: Histogram showing the percentage number of MV candidates eliminated Vs. NS_E for the HaarPDE and the Spatial-domain PDE with sequence *Coast Guard*. Values are shown only for the first 30 of the 256 squaring operations of the MSE.

Table 3.4: Percentage of the number of eliminated MV candidates at each level of the Haar wavelet hierarchy

	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
Total eliminations	99.7	99.8	99.7	99.7	99.8	99.8	99.8
Eliminations@level 1	74.4	84.4	76.5	88.7	37.9	49.3	49.9
Eliminations@level 2	20.0	10.1	16.7	7.8	27.8	27.6	23.7
Eliminations@level 3	4.0	1.8	3.6	2.2	20.5	15.7	18.6
Eliminations@level 4	1.2	1.0	1.7	0.9	11.3	5.3	7.4
Eliminations@level 5	0.1	2.5	1.2	0.1	2.3	1.9	0.2

tions, uncovered areas and higher spatial resolutions would not contribute much to the efficiency of the elimination process. The results of the HaarPDE for the sequences *Mother&Daughter*, *Football* and *Garden* confirm this observation, as *Mother&Daughter* is characterized by minor motions, *Football* contains a large amount of uncovered areas, and *Football* and *Garden* have higher resolutions.

3.4 Implementation of the Haar Wavelet Transform in the Proposed Motion Estimation Scheme and Overhead Cost Analysis

For a block of size $N \times N$, $N = 2^n$, the number of different types of operations to be performed for a four-scale Haar DWT illustrated in Fig. 3.2 is given in Table 3.5. It is seen from this table that the overhead cost required to compute each MV candidate is itself of the $O(N^2)$, which is not acceptable. However, there exist algorithms, such as the one in [46], that essentially reduce the computation of the Haar DWT to the use of a few look-up tables that are shared by all the blocks in a given frame.

Given the current block C and the reference block R, the Haar DWT of the difference block A can be implemented as $DWT(A) = DWT(R - C) = DWT(R) - DWT(C)$. For different MV candidates, $DWT(C)$ is fixed, and $DWT(R)$ is obtained through the fast scheme described in the following sub-section. For this purpose, the block diagram of Fig. 3.1 is modified and shown in Fig. 3.10.

Table 3.5: Operations required at each level of the four-scale Haar DWT for each MV candidate

	Addition/Subtraction	Shift	Memory access
4th level	$2N^2$	N^2	$4N^2$
3rd level	$\frac{1}{2}N^2$	$\frac{1}{4}N^2$	N^2
2nd level	$\frac{1}{8}N^2$	$\frac{1}{16}N^2$	$\frac{1}{4}N^2$
1st level	$\frac{1}{32}N^2$	$\frac{1}{64}N^2$	$\frac{1}{16}N^2$
Total	$\frac{85}{32}N^2$	$\frac{85}{64}N^2$	$\frac{85}{16}N^2$

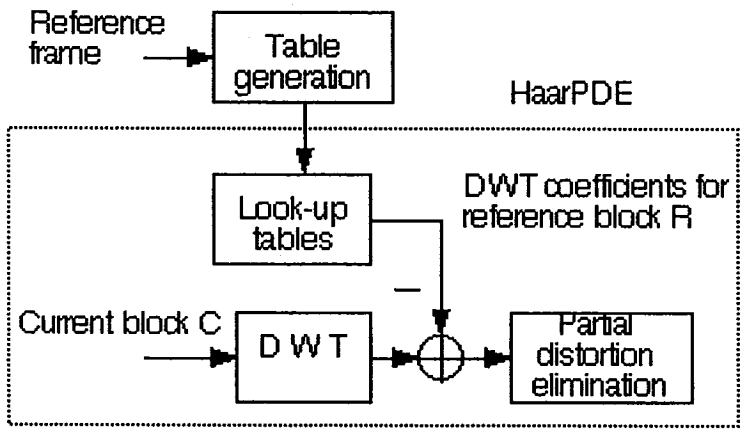


Figure 3.10: Modified block diagram illustrating the partial distortion elimination using the Haar wavelet

3.4.1 Computation of the Haar DWT of the Reference Block

As indicated by (3.2) and (3.3), the one-, two-, three- and four-scale Haar DWT for any given reference block can be calculated by obtaining the weighted sum of the

respective 2×2, 4×4, 8×8 and 16×16 sub-blocks, with the weights determined using (3.2) and (3.3). Some overlapped reference blocks might share some of these weighted sums for their respective Haar DWT. Therefore, we obtain, once for all, the weighted sum of each of the 2×2, 4×4, 8×8 and 16×16 regions within the reference frame and store it in a look-up table. From this set of look-up tables, the Haar DWT coefficients for any given reference block can then be retrieved.

As shown in Fig. 3.11, this process is implemented by a series of filtering operations using the filters for which the impulse-response coefficients are as specified in Table 3.6. In this implementation, the weighted sum of each of the 2×2 regions in the reference frame is first obtained by filtering the reference frame and then stored in T_{A4}, T_{V4}, T_{H4} and T_{D4}. Next, the weighted sums of the various 4×4 regions are calculated based on T_{A4} that contains the 2×2 weighted sums, and stored in T_{A3}, T_{V3}, T_{H3} and T_{D3}. Similarly, the weighted sums of the various 8×8 and 16×16 regions are obtained and stored in T_{A2}, T_{V2}, T_{H2} and T_{D2} and T_{A1}, T_{V1}, T_{H1} and T_{D1}, respectively. The thirteen look-up tables constructed by the above process are T_{A1}, T_{V1}, T_{H1}, T_{D1}, T_{V2}, T_{H2}, T_{D2}, T_{V3}, T_{H3}, T_{D3}, T_{V4}, T_{H4} and T_{D4}.

Table 3.6: Impulse response coefficients of the filters used to construct the look-up tables

$g_{4\varphi}$	{1,1 }	$g_{4\psi}$	{1,-1}
$g_{3\varphi}$	{1,0,1}	$g_{3\psi}$	{1,0,-1}
$g_{2\varphi}$	{1,0,0,0,1}	$g_{2\psi}$	{1,0,0,0,-1}
$g_{1\varphi}$	{1,0,0,0,0,0,0,0,1}	$g_{1\psi}$	{1,0,0,0,0,0,0,0,-1}

The retrieval of the Haar DWT coefficients for a given reference block R from the look-up tables is explained below. Let us first consider retrieving the one-scale Haar DWT coefficients from T_{V4}. As mentioned earlier, T_{V4} contains the weighted

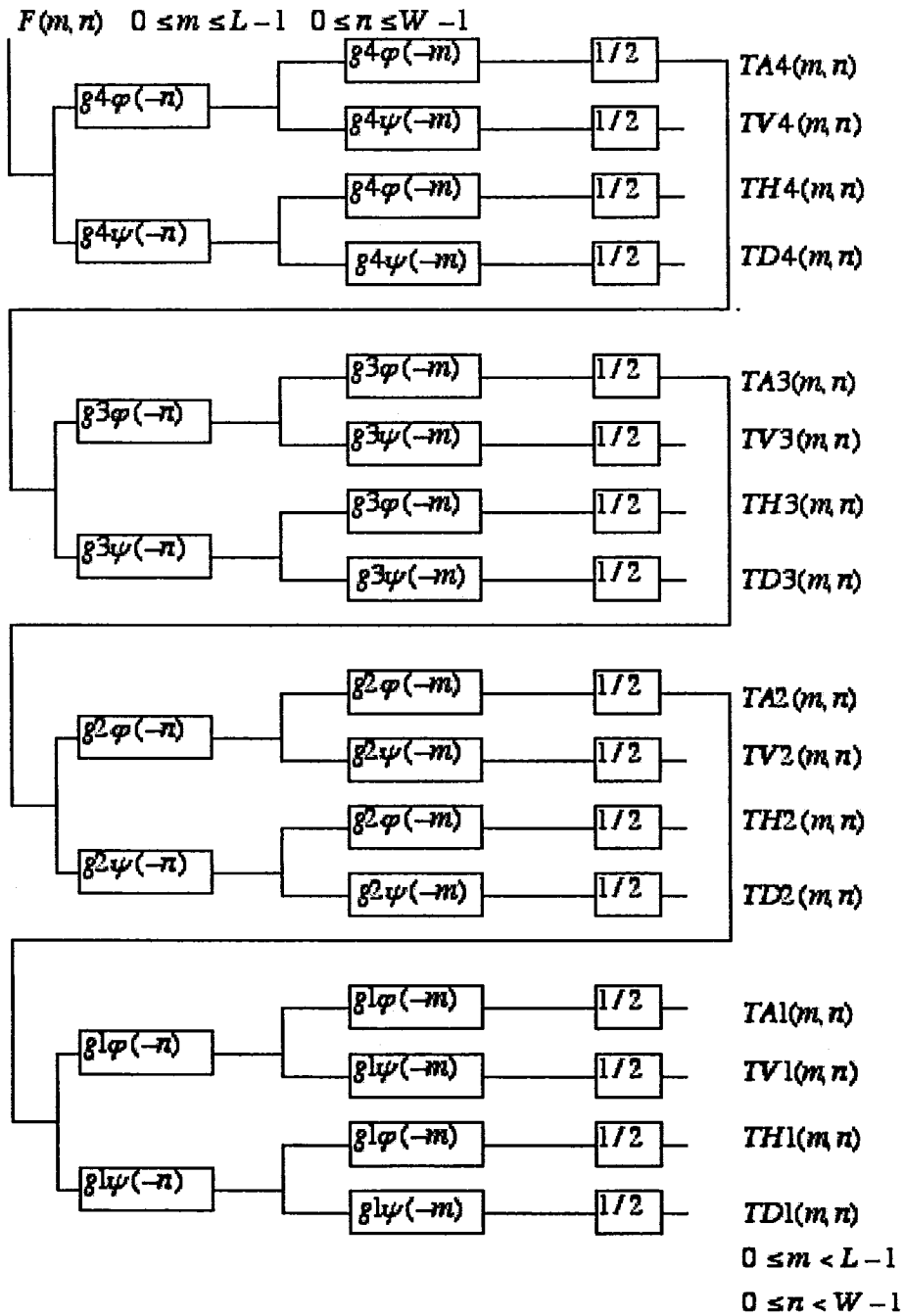


Figure 3.11: Block diagram for constructing the look-up tables for the Haar DWT

sum of each of the 2×2 regions in the reference frame. To be more specific, there exists a one-to-one correspondence between the entry at position *(y, x)* in T_{V4} and the weighted sum of the 2×2 region whose top-left corner is located at *(y, x)* in the reference frame. For the reference block R, the top-left corners of its 2×2 sub-blocks form a rectangular grid with a grid separation of two pixels both along the rows and the columns. Due to this one-to-one correspondence, the entries corresponding to *DWT(R)* in T_{V4} also have the same rectangular grid pattern, as illustrated in Fig. 3.12. Therefore, the Haar DWT coefficients for R can be retrieved by scanning the entries of T_{V4}, starting from (r_y, r_x), the position of the top-level corner of R in the reference frame, and moving rightward and then downward using a step size of two. The two-, three- and four-scale Haar DWT coefficients can be retrieved by using a similar procedure. Table 3.7 gives the look-up tables to be used, the matrix size for the Haar DWT output and the starting position and step size for each scan.

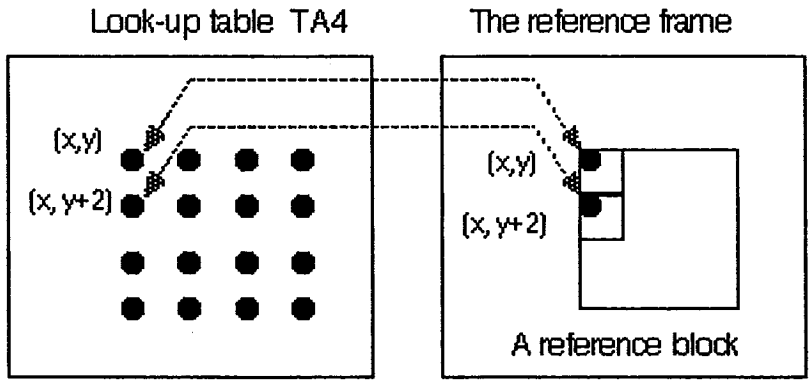


Figure 3.12: One-to-one correspondence between the top-left corners of the 2×2 sub-blocks of a reference block and the entries in the look-up table T_{A4}

Table 3.7: Retrieval of the Haar DWT coefficients from the look-up tables, where (r_y, r_x) is the position for the top-left corner of a given reference block in the reference frame.

Level	Look-up tables used	Size of DWT output	Starting position	Step size
4th level	T_{V4}, T_{H4}, T_{H4}	8×8	(r_y, r_x)	2
3rd level	T_{V3}, T_{H3}, T_{D3}	4×4	(r_y, r_x)	4
2nd level	T_{V2}, T_{H2}, T_{D2}	2×2	(r_y, r_x)	8
1st level	T_{A1}, T_{V1}, T_{H1}, T_{D1}	1×1	(r_y, r_x)	-

3.4.2 Overhead Cost Analysis

The overhead cost for the HaarPDE is due to the need for calculating the Haar DWT for the current block and its candidate reference blocks. The Haar DWT for the current block is carried out only once for each block and the total number of operations required is as shown in Table 3.5. With respect to the construction of the look-up tables, the operations required are addition/subtraction, shift and memory access. The number of additions/subtractions used by any of the filters in Fig. 3.11 is no greater the number of pixels that a frame contains. For a four-scale Haar DWT, there are 6×4 filters, and the total number of additions/subtractions is estimated as

$$\begin{aligned} N_{add/sub} &= 6 \times 4 \times W \times L \\ &= 24 \times W_b \times L_b \times N^2 \end{aligned}$$

where W, L, W_b, L_b and N^2 are the width, the length, the number of blocks along the row, the number of blocks along the column, and the block size of a frame, respectively. This is equivalent to $24N^2$ additions/subtractions per block. It is observed from Fig. 3.11 that the total number of shift operations needed is

$$\begin{aligned} N_{shift} &= 4 \times 4 \times W \times L \\ &= 16 \times W_b \times L_b \times N^2 \end{aligned}$$

which is equivalent to $16N^2$ shifts per block. The memory access includes the memory-read operations at the input and memory-write operations at the output of each filter in Fig. 3.11. Normally, each addition/subtraction requires two memory-read operations, one for each operand. However, a data reuse ratio of about 1:4 should be taken into account, since each pair of the g_φ and g_ψ filters share the same input, and two consecutive additions/subtractions in a filter share one of their operands. At the output of any of the filters, each output value requires one memory-write operation. Therefore, the total number of memory accesses is estimated as

$$\begin{aligned} N_{mem} &= \frac{2}{4} \times 6 \times 4 \times W \times L + 6 \times 4 \times W \times L \\ &= 36 \times W_b \times L_b \times N^2 \end{aligned}$$

which is equivalent to $36N^2$ memory accesses per block.

Thus, for the specific case of N = 16 and a search range of -15 to +15, where 15≈16=N, the per-MV-candidate cost for calculating the Haar DWT for the current block is about $\frac{85}{32}N^2/4N^2 = \frac{85}{128}$ additions/subtractions, $\frac{85}{64}N^2/4N^2 = \frac{85}{256}$ shifts and $\frac{85}{16}N^2/4N^2 = \frac{85}{64}$ memory accesses, while that for constructing the look-up tables is about $24N^2/4N^2 = 6$ additions/subtractions, $16N^2/4N^2 = 4$ shifts and $36N^2/4N^2 = 9$ memory accesses. The number of operations required per MV candidate to carry out the Haar DWT for the current block and to construct the look-up tables is given in Table 3.8.

Table 3.8: Number of operations required per MV candidate to carry out the Haar DWT for the current block and to construct the look-up tables

	Add/Sub	Compare	Multiply Accumulate	Shift	Memory access
Current block	$\frac{85}{128}$	-	-	$\frac{85}{256}$	$\frac{85}{64}$
Reference block	6	-	-	4	9

3.5 Computational Complexity of the Proposed HaarPDE

Block-matching motion estimation is carried out for the seven test sequences listed in Table 3.2 using the proposed HaarPDE, the two MSE-based full search methods, namely, the FS and the spatial-domain PDE, and four MAD-based full search methods, namely, the FS, the spatial-domain PDE , the MSEA and the finer granularity successive elimination (FGSE) algorithm. The spatial-domain PDE, the MSEA and the FGSE are chosen for the comparison, since they are known in the literature to have the best performance amongst all the full-search algorithms. The block size chosen is 16×16, and the search range is ± 15 pixels. All the methods except for the FS use the spiral search pattern, and use the same strategy to determine the initial minimum MSE or MAD as described in Section 3.3.2. The validity of the simulation is verified based on whether each of the algorithms results in an identical PSNR when given an identical target video frame as the input. It has been verified that the above-mentioned simulation is valid based on this validation criterion. Table 3.9 gives the average number of operations required for each MV candidate to perform the PDE in the Haar wavelet domain. That required to perform the spatial-domain methods, namely, the MSEA, the FGSE, the MAD-based PDE and the MSE-based PDE, are given in Table 3.10, Table 3.11, Table 3.12 and Table 3.13, respectively. The average number of operations required by the different methods considered in this chapter are listed in Table 3.14. The computational complexity for each of the methods is also given in Table 3.14, where a weight of two units is assigned to a memory access operation and a weight of one unit is assigned to each of the other

Table 3.9: Average number of operations required for each MV candidate to carry out the partial distortion elimination in the Haar wavelet domain

	Addition/ Subtraction	Compare	Multiply accumulate	Shift	Memory access
Foreman	28.3	3.6	3.6	4.3	17.6
Mother&Daughter	43.3	6.1	6.1	4.3	22.5
Car Phone	34.3	4.6	4.6	4.3	19.6
Container	18.7	2.0	2.0	4.3	14.4
Football	63.7	9.5	9.5	4.3	29.4
Garden	46.3	6.6	6.6	4.3	23.6
Coast Guard	34.9	4.7	4.7	4.3	19.8
Average	38.5	5.3	5.3	4.3	21.0

Table 3.10: Average number of basic operations required for each candidate motion vector to carry out the MSEA

	Addition/ Subtraction	Comparison	Absolute	Memory access
Foreman	45.5	1.3	5.6	16.2
Mother&Daughter	82.4	1.2	10.2	25.4
Car Phone	59.9	1.3	7.4	19.8
Container	28.3	1.1	3.4	11.8
Football	152.9	1.9	19.1	43.2
Flower Garden	98.0	1.6	12.2	29.4
Coast Guard	93.9	1.7	11.7	28.4
Average	80.1	1.4	9.9	24.8

operations. For software implementations, these weights reflect the number of clock cycles normally required for the corresponding basic operations [81]. It is seen from Table 3.14 that the computational complexity of the HaarPDE is 19% lower than that of the FGSE, 32% lower than that of the MSEA, and significantly lower than the rest of the methods.

While the above computational complexity analysis is concerned about the number of eliminations, the CPU time needed for the entire motion estimation phase reflects the actual gain each method can provide. For this purpose, we implement each of

Table 3.11: Average number of basic operations required for each candidate motion vector to carry out the FGSE

	Addition/ Subtraction	Comparison	Absolute	Memory access
Foreman	40.2	1.8	4.2	16.1
Mother&Daughter	64.8	2.4	6.8	22.6
Car Phone	49.7	2.1	5.2	18.8
Container	25.0	1.4	2.6	11.4
Football	115.3	3.8	12.2	40.4
Flower Garden	75.7	2.8	8.0	27.5
Coast Guard	69.1	2.6	7.3	26.5
Average	62.8	2.4	6.6	23.3

Table 3.12: Average number of basic operations required for each candidate motion vector to carry out the MAD-based PDE in the spatial domain

	Addition/ Subtraction	Comparison	Absolute	Memory access
Foreman	91.6	46.3	46.3	92.6
Mother&Daughter	72.4	36.7	36.7	73.4
Car Phone	87.4	44.2	44.2	88.4
Container	53.6	27.3	27.3	54.6
Football	165.4	83.2	83.2	166.4
Flower Garden	119.2	60.1	60.1	120.2
Coast Guard	121.2	61.1	61.1	122.2
Average	101.6	51.3	51.3	102.6

Table 3.13: Average number of basic operations required for each candidate motion vector to carry out the MSE-based PDE in the spatial domain

	Addition/ Subtraction	Comparison	Multiplication-Accumulation	Memory access
Foreman	22.5	22.5	22.5	45.0
Mother&Daughter	20.0	20.0	20.0	40.0
Car Phone	22.9	22.9	22.9	45.8
Container	11.8	11.8	11.8	23.6
Football	47.0	47.0	47.0	94.0
Flower Garden	28.9	28.9	28.9	57.8
Coast Guard	26.5	26.5	26.5	53.0
Average	25.6	25.6	25.6	51.2

Table 3.14: Average number of operations required and the total computational complexity for each MV candidate using various full-search methods

	Add/Sub	Absolute	Compare	Multiply accumulate	Shift	Memory access	Total complexity
Methods using MAD as the matching criterion							
FS	511	256	1	-	-	512	1792
PDE	101.6	51.3	51.3	-	-	102.6	408.6
MSEA	80.1	9.9	1.4	-	-	24.8	141.0
FGSE	62.8	6.6	2.4	-	-	23.3	118.4
Methods using MSE as the matching criterion							
FS	256	-	1	256	-	512	1537
PDE	25.6	-	25.6	25.6	-	51.2	179.2
HaarPDE	38.5	-	5.3	5.3	4.3	21.0	95.4

the methods using C on a general-purpose work station and obtain the CPU time needed to encode each of the sequences. The CPU time required by each of the methods, shown as a percentage with respect to the time needed by the MAD-based FS, is given in Table 3.15. It is seen from this table that after taking into account the computational overhead, the performance of the HaarPDE in terms of the CPU time is slightly better than or comparable to that of the MSEA and FGSE, and significantly better than that of the other methods.

The block-matching motion estimation is also carried out as a rate-distortion optimization (RDO) [82] in terms of $D(mv) + \lambda_{motion} R(mv)$, where $D(mv)$ is the block-matching error using the MSE or the MAD, λ_{motion} is a constant, and $R(mv)$ is the number of bits required to represent the motion vector in question. λ_{motion} is set to 150 when using the MSE, and 15 when using the MAD [82]. The Huffman codebook of H.263 [21] is used to determine $R(mv)$. Table 3.16 gives the average number of operations required for each MV candidate to perform the PDE in the Haar wavelet domain. That required to perform the spatial-domain methods, namely, the MSEA,

Table 3.15: CPU time required to encode each sequence by various full-search methods, shown as a percentage with respect to the time needed by the MAD-based FS.

	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard	Average
Methods using MAD as the matching criterion								
FS	100%	100%	100%	100%	100%	100%	100%	100%
PDE	19.4%	15.8%	18.6%	12.2%	33.4%	24.6%	25.0%	21.3%
MSEA	4.3%	6.3%	5.1%	3.3%	9.7%	7.0%	6.6%	6.0%
FGSE	4.5%	6.2%	5.2%	3.4%	9.5%	6.8%	6.4%	6.0%
Methods using MSE as the matching criterion								
FS	70.0%	69.9%	70.1%	69.9%	69.4%	69.6%	69.8%	69.8%
PDE	7.6%	7.0%	7.7%	4.8%	14.1%	9.3%	8.6%	8.4%
HaarPDE	4.4%	5.0%	4.7%	4.1%	5.9%	5.2%	4.8%	4.9%

the FGSE, the MAD-based PDE and the MSE-based PDE, are given in Table 3.17, Table 3.18, Table 3.19 and Table 3.20, respectively. The computational complexities for the various methods are listed in Table 3.21, and the CPU time required, as a percentage with respect to the time needed by the MAD-based FS, are given in Table 3.22.

As seen from Tables 3.14 and 3.21, the computational complexities of the various fast full-search methods get reduced because of the use of the rate-distortion optimization, and this reduction ranges from 15% to 33%. However, the computational complexity of the HaarPDE still remains the lowest amongst the various full-search methods. Also, it is observed from Tables 3.15 and 3.22 that the CPU run-times of the HaarPDE, the MSEA and the FGSE have slightly increased due to the overhead added by the implementation of the RDO; but, the performance of the HaarPDE in terms of the CPU time is still better than or comparable to that of the MSEA and the FGSE, and significantly better than that of the other methods.

Table 3.16: Average number of operations required for each MV candidate to carry out the partial distortion elimination in the Haar wavelet domain based on the rate-distortion optimization

	Addition/ Subtraction	Compare	Multiply accumulate	Shift	Memory access
Foreman	23.5	2.8	2.8	4.3	16.0
Mother&Daughter	16.9	1.7	1.7	4.3	13.8
Car Phone	24.7	3.0	3.0	4.3	16.4
Container	17.5	1.8	1.8	4.3	14.0
Football	60.7	9.0	9.0	4.3	28.4
Garden	43.9	6.2	6.2	4.3	22.8
Coast Guard	32.5	4.3	4.3	4.3	19.0
Average	31.4	4.1	4.1	4.3	18.6

Table 3.17: Average number of basic operations required for each candidate motion vector to carry out the MSEA based on the rate-distortion optimization

	Addition/ Subtraction	Comparison	Absolute	Memory access
Foreman	29.7	1.2	3.6	12.2
Mother&Daughter	20.1	1.1	2.4	9.8
Car Phone	33.7	1.1	4.1	13.2
Container	17.8	1.0	2.1	9.2
Football	124.2	1.7	15.5	36.0
Flower Garden	75.7	1.5	9.4	23.8
Coast Guard	69.2	1.6	8.6	22.2
Average	52.9	1.3	6.5	18.0

Table 3.18: Average number of basic operations required for each candidate motion vector to carry out the FGSE based on the rate-distortion optimization

	Addition/ Subtraction	Comparison	Absolute	Memory access
Foreman	29.7	1.5	3.1	12.8
Mother&Daughter	20.5	1.3	2.1	10.0
Car Phone	31.7	1.6	3.3	13.5
Container	19.4	1.2	2.0	9.5
Football	96.4	3.3	10.2	34.2
Flower Garden	60.6	2.3	6.4	23.0
Coast Guard	54.8	2.2	5.8	21.4
Average	44.7	1.9	4.7	17.8

Table 3.19: Average number of basic operations required for each candidate motion vector to carry out the MAD-based PDE in the spatial domain based on the rate-distortion optimization

	Addition/ Subtraction	Comparison	Absolute	Memory access
Foreman	66.6	33.3	33.3	66.6
Mother&Daughter	19.8	9.9	9.9	19.8
Car Phone	54.8	27.4	27.4	54.8
Container	8.0	4.0	4.0	8.0
Football	149.2	74.6	74.6	149.2
Flower Garden	98.4	49.2	49.2	98.4
Coast Guard	100.8	50.4	50.4	100.8
Average	71.1	35.5	35.5	71.1

Table 3.20: Average number of basic operations required for each candidate motion vector to carry out the MSE-based PDE in the spatial domain based on the rate-distortion optimization

	Addition/ Subtraction	Comparison	Multiplication-Accumulation	Memory access
Foreman	15.1	15.1	15.1	30.2
Mother&Daughter	3.1	3.1	3.1	6.2
Car Phone	13.5	13.5	13.5	27.0
Container	1.8	1.8	1.8	3.6
Football	43.7	43.7	43.7	87.4
Flower Garden	22.8	22.8	22.8	45.6
Coast Guard	19.8	19.8	19.8	39.6
Average	17.1	17.1	17.1	34.2

Table 3.21: Average number of operations required and the total computational complexity for each MV candidate using various full-search methods for the rate-distortion optimization.

	Add/Sub	Absolute	Compare	Multiply accumulate	Shift	Memory access	Total complexity
Methods using MAD as the matching criterion							
FS	511	256	1	-	-	512	1792
PDE	71.1	35.5	35.5	-	-	71.1	284.3
MSEA	52.9	6.5	1.3	-	-	18.0	96.7
FGSE	44.7	4.7	1.9	-	-	17.8	86.9
Methods using MSE as the matching criterion							
FS	256	-	1	256	-	512	1537
PDE	17.1	-	17.1	17.1	-	34.2	119.7
HaarPDE	31.4	-	4.1	4.1	4.3	18.6	81.1

Table 3.22: CPU time required to encode each sequence by various full-search methods using rate-distortion optimization, shown as a percentage with respect to the time needed by the MAD-based FS.

	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard	Average
Methods using MAD as the matching criterion								
FS	100%	100%	100%	100%	100%	100%	100%	100%
PDE	17.1%	6.8%	14.5%	4.4%	34.8%	23.7%	24.3%	17.9%
MSEA	5.0%	4.4%	5.3%	4.4%	10.2%	7.5%	7.0%	6.3%
FGSE	5.0%	4.6%	5.3%	4.4%	9.8%	7.1%	6.7%	6.1%
Methods using MSE as the matching criterion								
FS	70.0%	69.9%	70.1%	69.9%	69.4%	69.6%	69.8%	69.8%
PDE	7.5%	3.5%	6.9%	3.2%	16.5%	9.8%	8.8%	8.0%
HaarPDE	5.3%	5.0%	5.5%	5.2%	6.9%	6.0%	5.7%	5.7%

3.6 Summary

In this chapter, we have presented a fast full-search block-matching motion estimation algorithm that performs partial distortion elimination in the Haar wavelet domain. The proposed algorithm uses the MSE to measure the block-matching error and can provide a higher motion estimation accuracy for the BME than any of the full search algorithms that use the MAD as the matching criterion. Since the value of the MSE is unchanged in the Haar wavelet domain, wherein the larger-magnitude coefficients are expected to be concentrated in the lower-frequency sub-regions, the partial distortion elimination conducted in the Haar wavelet domain starting with the coefficients in the lower-frequency sub-regions has resulted in an earlier elimination of the impossible motion vector candidates than is the case with any existing MSE-based spatial-domain method.

Extensive simulations, including those of the rate-distortion optimization, have been conducted to compare the overall computational complexity as well as the CPU time needed for the motion estimation of the proposed algorithm with that of the various fast full search methods. The results show that the proposed algorithm results in a complexity that is significantly lower than that of the MSE-based and MAD-based spatial-domain partial distortion elimination algorithms and lower than or comparable to that of the multi-level successive elimination algorithm (MSEA) and the finer granularity successive elimination (FGSE) algorithm. The CPU time needed in the case of the proposed algorithm is lower than that needed for the MSEA and FGSE, and significantly lower than that needed for the other methods.

Chapter 4

Temporal Search Complexity Reduction in Multi-frame Block-Matching Motion Estimation

4.1 Introduction

Multi-frame block-matching motion estimation (MFBME) is capable of yielding a higher motion prediction accuracy than single-frame block-matching motion estimation (SFBME) [83]. Multi-frame block-matching motion estimation using up to five frames has become a feature of the H.264 to improve the coding performance [22]. However, the increased accuracy is achieved at the expense of a tremendous increase in the computational complexity. A great deal of effort has, therefore, been devoted to the development of faster and simpler algorithms for MFBME [84–87].

Essentially, MFBME is a 3-D search within a multi-frame memory for the best block matching. In addition to an exhaustive spatial and temporal search, partial

search in the spatial or temporal domain may be used to reduce the computational complexity. In the approach of Duanmu et al [86], full search is conducted in the temporal domain while only a partial search is done in the spatial domain. In [85] and [87], a partial search is conducted in both the spatial and temporal domains. Since in both of these approaches, only a partial search is conducted in the spatial domain, and as we will show later, most of the block matching is found in the first reference frame of the multi-frame memory, there exists the possibility that, for certain frames, the search accuracy of these approaches may be worse than that of the single-frame full search of SFBME, and hence, better accuracy is not always guaranteed. In view of this, as well as the fact that there exist in the spatial domain fast full search algorithms that can significantly reduce the computational complexity of the full search, such as the multi-level successive elimination algorithm (MSEA) [46] and the finer granularity successive elimination (FGSE) algorithm [47] mentioned in Chapter 2 and the HaarPDE algorithm proposed in the previous chapter, we now focus our attention to an approach of the MFBME, where full search is conducted in the spatial domain, and the computational complexity in the temporal domain is reduced by using a full search with early termination.

The chapter [88] is organized as follows. Section 4.2 contains an investigation on the statistical characteristics of the temporal motion vectors, and a study as to the frame number beyond which the inclusion of additional reference frames contributes little to the reduction in the block-matching error. Based on this study, an early-termination method is proposed in Section 4.3 for the purpose of reducing the temporal complexity of MFBME. In this method, the block-matching motion estimation is carried out using the mean-absolute-difference-based optimization as well

as the rate-distortion-based optimization, and the resulting performance is evaluated and compared to that of the 3-D full search in Section 4.4. Finally, certain conclusions are drawn in Section 4.5.

4.2 Statistical Investigation on Temporal Motion Vectors and Block-Matching Errors

It is a widely-used assumption in the case of the MFBME that the best block match is most likely to be found in the first few frames of the multi-frame memory. That it is indeed so is demonstrated by the following two statistical experiments. The first experiment examines the distribution of the temporal motion vectors (MV_t) along the temporal dimension of a multi-frame memory. The second one examines the reduction of the block-matching errors along the temporal dimension of the multi-frame memory. For these experiments, the size of the search window is chosen to be -15 to +15, the maximum memory depth as 100 frames, and the mean absolute differences (MAD) as the measure for the block-matching error. Seven test video sequences are used, details of which are given in Table 4.1. These sequences cover different frame resolutions and motion possibilities.

4.2.1 Distribution of the Temporal Motion Vectors

To observe the statistical distribution of the MV_t, a full search is carried out within the 100-frame memory for all the sequences. Table 4.2 gives the number in percentage of the temporal motion vectors occurring in a reference frame as a function of the reference frame number for all the sequences. It is seen from this table that the largest

number of MV_t occur in the first reference frame itself, and the majority of the MV_t occurrence is in the first few reference frames, beyond which it drops drastically.

Table 4.1: Video sequences used in the simulations

Sequence	Frames	Format	Resolution	Motion intensity/Content
Foreman	201-300	QCIF	176x144	mixed/foreground human, camera panning
Mother&Daughter	201-300	QCIF	176x144	minor/foreground human, stationary background
Car Phone	201-300	QCIF	176x144	mixed/foreground human, background partially moving
Container	201-300	QCIF	176x144	small/object translations
Football	101-125	SIF	352x240	medium/human in sports
Flower Garden	101-115	SIF	352x240	medium/camera panning
Coast Guard	101-125	CIF	352x288	medium/object translations

Table 4.2: Normalized MV_t occurrence as a function of the reference frame number

Sequence	Reference frame number								
	1	2	3	4	5	6-10	11-20	21-50	51-100
Foreman	54.2%	9.2%	4.1%	2.7%	2.2%	4.6%	3.9%	9.6%	9.5%
Mother&Daughter	56.8%	5.8%	2.2%	2.3%	1.9%	6.1%	6.0%	8.8%	10.1%
Car Phone	40.0%	14.8%	8.9%	5.6%	4.7%	7.8%	7.6%	5.8%	4.8%
Container	84.6%	3.0%	1.8%	1.1%	0.9%	6.3%	1.3%	1.0%	0.0%
Football	63.0%	8.0%	6.1%	4.2%	3.2%	6.6%	3.5%	4.2%	1.2%
Flower Garden	56.3%	6.1%	2.4%	5.7%	11.9%	12.0%	3.3%	0.7%	1.6%
Coast Guard	62.3%	9.2%	4.4%	4.2%	4.3%	10.0%	3.4%	1.6%	0.6%

4.2.2 Reduction in the Block-Matching Error

Given the current block C in any frame of the sequence and the j-th block $R_j(i)$ in the search window of the i-th reference frame, let $MAD(C, R_j(i))$ be the mean absolute difference between C and $R_j(i)$. Then,

$$MAD_C(i) = \min_{\forall j} MAD(C, R_j(i)) \tag{4.1}$$

is the MAD between C and its matching block in the search window of the *i-th* reference frame. Also,

$$BMAD_C(i) = \min_{\forall k \in [1, ..., i]} MAD_C(k) \tag{4.2}$$

is the MAD between C and its matching block found within the search windows of all the reference frames 1,2, ..., i.

In order to study the contribution of each reference frame to the reduction in the block-matching error, we increase the memory depth by one frame at a time, and calculate the reduction in the MAD due to the additional reference frame. The maximum reduction that could be obtained is when the memory comprises all the 100 reference frames, and in such a case, the average reduction in the block-matching error is

$$\rho_{av} = \frac{1}{N} \sum_C (BMAD_C(100) - BMAD(1)) \tag{4.3}$$

where $\sum_C$ indicates that the summation is over all the blocks in the sequence, the total number of such blocks being N. Then, the percentage reduction obtained by increasing the memory depth from *(i-1)* frames to i frames is

$$P_i = \frac{100 \sum_C (BMAD_C(i-1) - BMAD_C(i))}{N \cdot \rho_{av}} \tag{4.4}$$

for i = 2,3, ..., 100. Table 4.3 gives the values of P_i resulting from successive increments in the memory depth for the various sequences. It is seen from this table that the bulk of the reduction is due to the first few reference frames, beyond which the reduction drops drastically.

Table 4.3: Normalized reduction in the MAD resulting from successive increments in the memory depth

Sequence	P_2	P_3	P_4	P_5	$\sum_6^{10} P_i$	$\sum_{11}^{20} P_i$	$\sum_{21}^{50} P_i$	$\sum_{51}^{100} P_i$
Foreman	30.2%	13.6%	8.9%	6.2%	11.1%	10.0%	11.9%	8.1%
Mother&Daughter	37.7%	11.2%	7.9%	6.0%	16.2%	7.0%	9.8%	4.2%
Car Phone	39.5%	16.3%	8.4%	6.2%	11.0%	7.2%	5.1%	6.3%
Container	29.0%	17.3%	5.9%	1.6%	45.0%	0.7%	0.5%	0.0%
Football	38.3%	19.4%	11.6%	4.9%	11.6%	5.6%	6.5%	2.1%
Flower Garden	12.5%	5.3%	23.5%	32.2%	20.0%	3.4%	1.6%	1.5%
Coast Guard	41.8%	12.0%	13.7%	9.6%	17.0%	3.2%	2.7%	0.0%

4.2.3 Observations from the Two Experiments

We characterize the rapid drop in the occurrence of the MV_t, as seen in Section 4.2.1, by making the following observation. Using a threshold occurrence of 2%, the reference frame number at which the occurrence drops below the threshold for the first time (T_{ref}) is given in Table 4.4 for each of the seven test sequences. Table 4.4 also gives the total number of the MV_t occurrences in the first T_{ref} reference frames, the total MAD reduction when the memory has T_{ref} reference frames, the average MAD for the 1st reference frame and the maximum reduction in the average MAD, ρ_{av}. This table shows that for the seven test sequences considered, (1) the occurrences of the MV_t drops below 2% within the first 3-9 reference frames, (2) except for the sequence *Container*, these first few reference frames account for the majority of the occurrences of the MV_t and the total MAD reduction compared to that using the previous 100 reference frames, and (3) for sequences such as *Mother & Daughter* and *Container* for which the average MAD for the first reference and ρ_{av} are small, the amount of reduction in the MAD obtained in the first T_{ref} reference frames is irrelevant, since the block-matching within the first reference frame is already accurate enough. These observations support our earlier remark regarding the widely-used

Table 4.4: Cumulative occurrence of the MV_t and cumulative MAD reduction at the reference frame T_{ref}

Sequences	T_{ref}	MV_t occurrences in [1, T_{ref}]	Total % reduction in MAD $\sum_2^{T_{ref}} P_i$	Avg. MAD for 1st reference frame $\frac{1}{N}\sum_C MAD_C(1)$	Max. avg. reduction in MAD ρ_{av}
Foreman	6th frame	74%	63%	3.76	0.58
Mother&Daughter	5th frame	69%	63%	1.25	0.077
Container	3rd frame	89%	46%	0.89	0.038
Car Phone	6th frame	76%	73%	4.34	0.84
Football	6th frame	85%	76%	9.45	0.78
Flower Garden	8th frame	92%	90%	10.65	1.20
Coast Guard	9th frame	94%	93%	5.36	0.69

assumption that, in the case of the MFBME, the best block match is likely to be found in the first few frames of the multi-frame memory.

The occurrence of the majority of the MV_t and the large MAD reduction in the first T_{ref} reference frames is a reflection of the high correlation between the current frame and the reference frames in its close neighborhood. This correlation decreases beyond T_{ref}, and might be due to motion occlusions, uncovered areas, camera panning, content changes and so on. Although block matching may still be possible for these cases beyond T_{ref}, it is of little practical significance. Therefore, searching beyond T_{ref} in the temporal dimension is not necessary in the statistical sense, and a frame-by-frame scan that starts from the first reference frame with an early termination could be a method to capture the majority of the MV_t and to obtain a large MAD reduction. An early termination method is discussed in the next section.

4.3 Early Termination in the Temporal Dimension of the Multi-Frame Memory

In order to implement an early termination for the search in the temporal domain, we turn to the block matching error rather than T_{ref}, since the former can be tracked on a reference frame by reference frame basis. Table 4.5 shows the percentage number of blocks of C for which $MAD_C(1)$, as defined in (4.1), lies in various intervals for each of the test sequences under consideration. It is seen that the first 12 intervals shown in the table account for more than 60% of the blocks for each of the sequences, i.e., $MAD_C(1) < 12$ for these blocks. For the sequences that contain only minor motions or stationary backgrounds, such as *Mother & Daughter* and *Container*, the value of $MAD_C(1)$ is less than two for the majority of the blocks.

Table 4.5: Percentage number of blocks of C for which $MAD_C(1)$ lies in various intervals

$MAD_C(1)$	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
[0,1)	13	64	13	85	0	5	0
[1,2)	23	21	25	8	5	9	5
[2,3)	18	8	17	5	11	3	18
[3,4)	13	4	12	2	10	3	18
[4,5)	9	2	8	0	9	4	15
[5,6)	6	1	6	0	7	5	12
[6,7)	5	0	4	0	6	6	9
[7,8)	3	0	3	0	6	6	7
[8,9)	2	0	2	0	5	5	5
[9,10)	2	0	2	0	5	6	3
[10,11)	2	0	2	0	5	5	3
[11,12)	2	0	1	0	4	6	1
[12, 255]	2	0	6	0	27	37	4

Let us now look at $MAD_C(i)$ as a function of the reference frame number i,

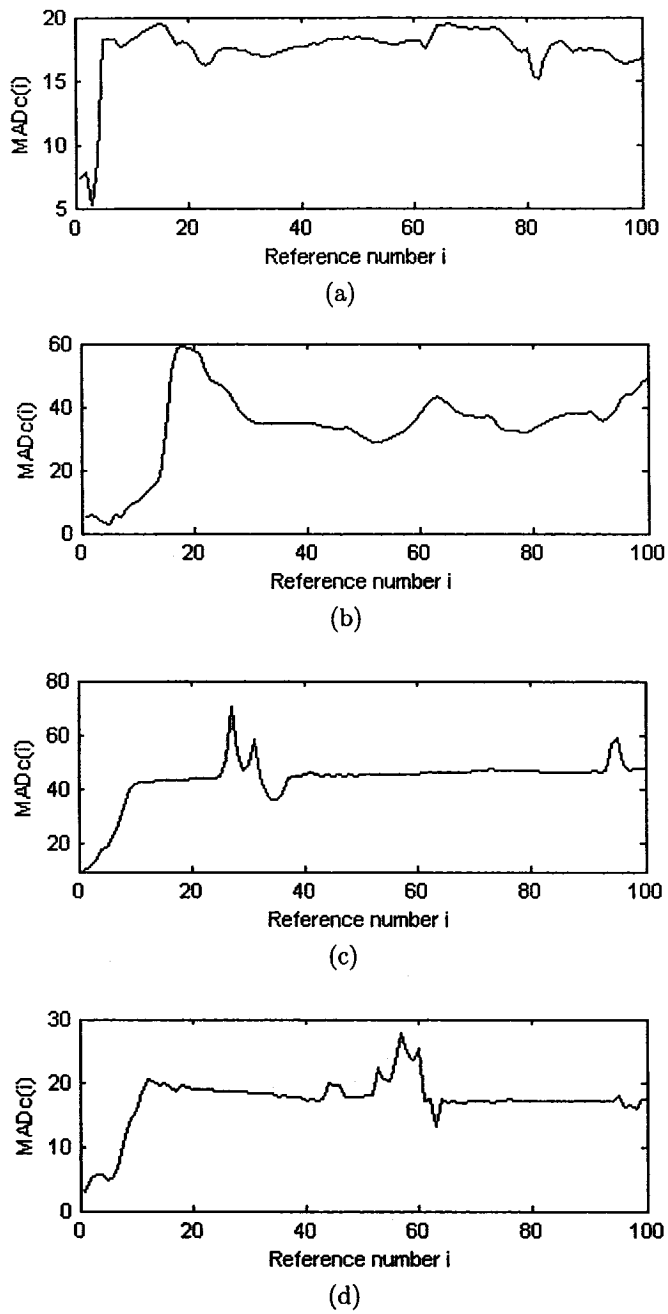


Figure 4.1: Variation of $MAD_C(i)$ as a function of the reference frame number i, when (a) C is block (7,9) of the 257th frame of *Foreman*, containing occlusion, (b) C is block (3,3) of the 270th frame of *Foreman*, containing uncovered area, (c) C is block (9,11) of the 285th frame of *Foreman*, containing camera panning, and (d) C is block (2,8) of the 115th frame of *Football*, containing uncovered area.

$1 \leq i \leq 100$, for a given C. For this purpose, the value of $MAD_C(i)$ for each of the blocks in a given test sequence is obtained. For the purpose of illustration, plots showing the variation of $MAD_C(i)$ as a function of i for four blocks in the sequences *Foreman* and *Football* are shown in Fig. 4.1. These blocks are chosen so as to include motion discontinuities such as occlusion, uncovered area and camera panning. It is seen from these figures that, in the presence of motion discontinuities, there is a sudden change in the value of $MAD_C(i)$, indicating a drastic reduction in the temporal correlation between C and its matching block. In view of the motion discontinuities, any search beyond such an abrupt change in $MAD_C(i)$ would only lead to a numerically better matching instead of one based on content. Therefore, the sharp increase in $MAD_C(i)$ can be used as an indication to terminate the temporal search, and techniques such as thresholding can be employed for this purpose.

To avoid the need for determining the thresholds, which might be sequence-dependent, a simpler method that does not need thresholds is proposed and is described by the flow chart of Fig. 4.2(a). The method keeps track of $MAD_C(i)$ reference frame by reference frame and looks for the local minimum. The process continues if a smaller $MAD_C(i)$ is found, otherwise the termination control is triggered.

Since early termination may incur some inaccuracy in the resulting motion vectors compared to those obtained by a full search, the termination control is applied only to two groups of blocks, leaving the rest still to be processed through a full search. These two groups are those that fall into the interval [0,α) or [β, 255] with $\alpha \ll 255$ and $\beta \gg 0$. An extremely small $MAD_C(i)$ indicates that relative to the reference block, the current block contains minor or stationary motions, and a search beyond the current reference frame will not significantly reduce the block matching error,

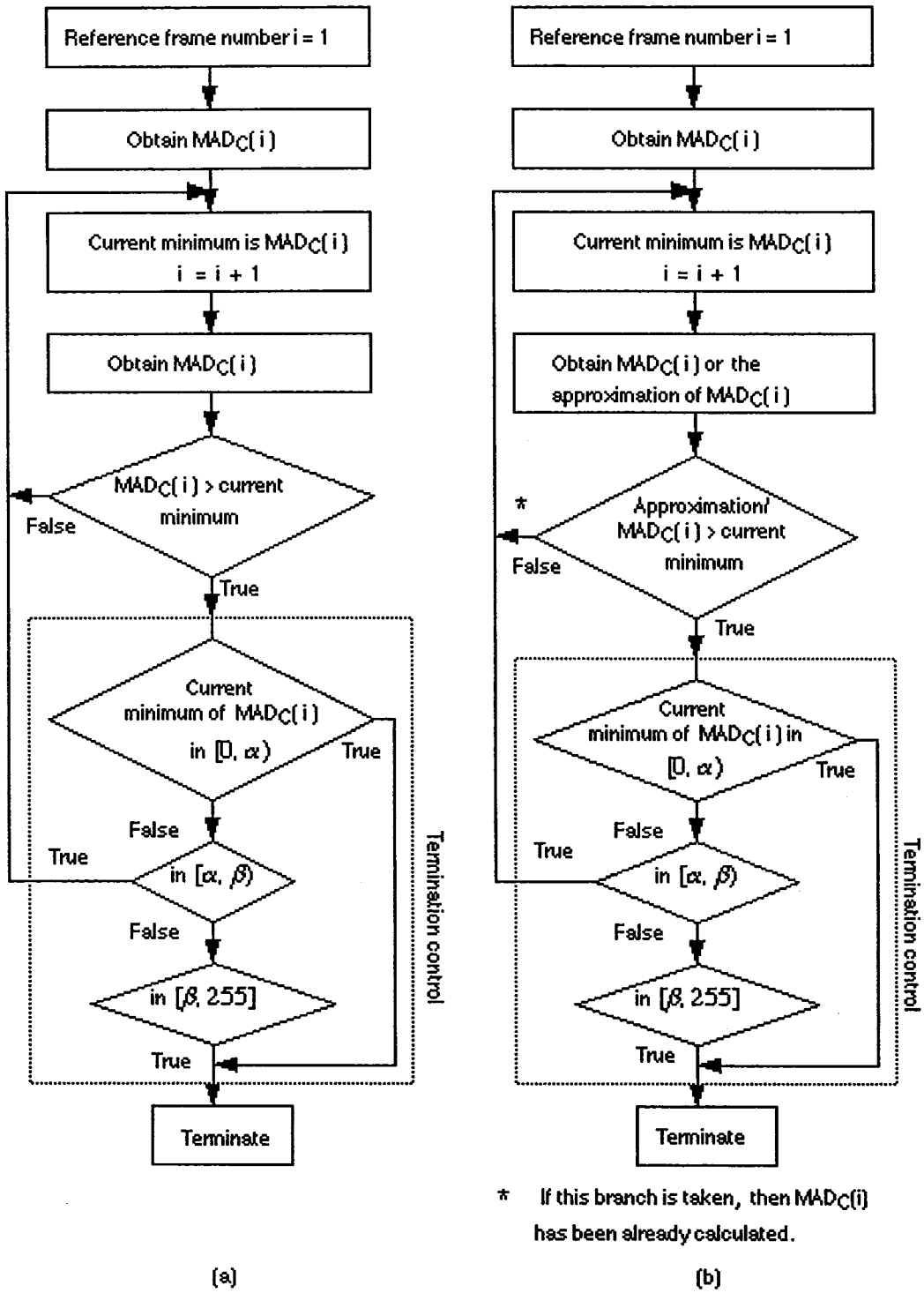


Figure 4.2: Flow chart illustrating the proposed early termination method for a given current block C when (a) the full search in the spatial domain is not expedited by any fast method, and (b) an SE-based algorithm is used as the fast full search method in the spatial domain.

since the current one is already small enough. On the other hand, a very large $MAD_C(i)$ indicates that a content discontinuity such as scene change, occlusion or uncovered area might exist, and a further search could only result in a numerically better matching rather than one based on content. Therefore, for these two groups of blocks, the search could be terminated, once a local minimum of $MAD_C(i)$ is identified. Since these two groups of blocks with α=1 and β=12 account for up to 85% of the total blocks (see Table 4.5), the termination is expected to result in significant savings in the search complexity. The actual performance of the proposed early termination method will be presented in the next section.

When an SE-based algorithm is used as the fast full search method in the spatial domain, some extra considerations should be given, since it is possible that $MAD_C(i)$ is not available for certain blocks C and certain reference frames i.

Given the current block C and the *j-th* block $R_j(i)$ in the search window of the *i-th* reference frame, an SE-based algorithm gives a series of lower bounds of $MAD(C, R_j(i))$, denoted as $L_1(C, R_j(i))$, $L_2(C, R_j(i))$, ..., $L_n(C, R_j(i))$, respectively, with

$$L_1(C, R_j(i)) \leq L_2(C, R_j(i)) \leq ... \leq L_n(C, R_j(i)) \leq MAD(C, R_j(i))$$

$L_1(C, R_j(i))$ is obtained first. If it is greater than the current minimum MAD between C and the reference blocks that have been checked so far in reference frame i, $R_j(i)$ is eliminated as a match for C; otherwise, $L_2(C, R_j(i))$ is obtained and the comparison is repeated. This process continues until either $R_j(i)$ is eliminated or $MAD(C, R_j(i))$ is computed.

If each of $R_j(i)$ is eliminated by the SE-based algorithm, then $MAD(C, R_j(i))$ is not computed, and hence, $MAD_C(i)$ will not be available. In such a case, let

$R_j(i)$ be eliminated by $L_k(C, R_j(i))$, then $L_k(C, R_j(i))$ is the largest lower bound of $MAD(C, R_j(i))$ obtained by the SE-based algorithm, and we have

$$\begin{aligned} L_k(C, R_j(i)) &\leq MAD(C, R_j(i)) \quad and \\ \min_{\forall j} L_k(C, R_j(i)) &\leq \min_{\forall j} MAD(C, R_j(i)) = MAD_C(i) \end{aligned}$$

Therefore, $\min_{\forall j} L_k(C, R_j(i))$ is a lower bound of $MAD_C(i)$. Using it as an approximation of $MAD_C(i)$, the flow chart in Fig. 4.2(a) is modified and shown in Fig. 4.2(b).

Since it is the lower bound of $MAD_C(i)$ that is used in the termination control, it is possible that the termination does not happen immediately on finding the first local minimum of $MAD_C(i)$. In other words, the termination can be delayed until after more local minima of $MAD_C(i)$ are found, leading to search results that are more accurate or at least the same as those obtained based on the flow chart of Fig. 4.2(a).

4.4 Simulation Results

We implement the early termination method proposed in Section 4.3 employing the four-level MSEA and FGSE, respectively, as the search method in the spatial domain. The block-matching motion estimation is carried out using a MAD-based optimization as well as a rate-distortion optimization (RDO) [82]. In the case of the latter, the optimization is in terms of $D(mv)+\lambda_{motion}R(mv)$, where $D(mv)$ is the block-matching error using the MAD, λ_{motion} is a constant, and $R(mv)$ is the number of bits required to represent the motion vector in question. λ_{motion} is set to 15 [82], and the Huffman codebook of H.263 [21] is used to determine $R(mv)$. The spatial full search follows a spiral path starting from the center of the search window. The test sequences used are the ones listed in Table 4.1, the size of the search window being from -15 to +15

pixels and the memory depth being 1, 5, 10 or 15 frames.

The validity of the simulation is verified based on two criteria: (1) for each early termination scheme, whether the use of the MSEA and the FGSE result in an identical PSNR when given an identical target video frame as the input; (2) for each early termination scheme, when the functionality of early termination is disabled, whether an identical PSNR with that resulting from the FS is obtained.

Two schemes for the early termination are simulated. The first one, denoted by ET-I, corresponds to the case of early termination of the search when $MAD_C(i) \in [0, 1)$. The second one, denoted by ET-II, corresponds to the case when $MAD_C(i) \in [0, 1) \cup [12, 255]$. Based on the two above-mentioned validation criteria, it has been verified that the simulation is valid. The performances of the proposed early termination method using ET-I and ET-II are compared with that of the 3-D full search (FS), which employs the same search method (the MSEA or the FGSE) in the spatial domain as that used in the proposed early termination method. For this purpose, we obtain the following three sets of results for the proposed method and the FS.

- *Peak signal-to-noise ratio* [44]: The difference between the PSNR for each of the two schemes of the proposed method and that of the FS is computed and presented in Table 4.6 and 4.9 for the case of the MAD-based optimization and in Table 4.12 and 4.15 for the case of the RDO.

- *Computational complexity in terms of the number of equivalent MAD calculations*: The computation of the MAD involves N^2 subtractions, $N^2 - 1$ additions and N^2 absolute operations for a block of size $N \times N$ [44]. The number of equivalent MAD calculations required by the proposed method and the FS are as shown in Table 4.7 and 4.10 for the case of the MAD-based optimization and

in Table 4.13 and 4.16 for the case of the RDO.

- *Temporal motion vectors*: The percentage of temporal motion vectors correctly estimated by the proposed early termination method are as shown in Table 4.8 and 4.11 for the case of the MAD-based optimization and in Table 4.14 and 4.17 for the case of the RDO.

It is seen from Tables 4.6, 4.9, 4.12 and 4.15 that ET-I hardly incurs any loss in the PSNR for any of the test sequences. In terms of the search complexity shown in Tables 4.7, 4.10, 4.13 and 4.16, the amount of savings achieved by ET-I is motion-dependent, ranging from 0% for the *Football* and *Coast Guard* sequences, where none of the blocks has a value of $MAD_C(i)$ in [0, 1) (see Table 4.5), to more than 33% for the *Mother & Daughter* and *Container* sequences, where the majority of the blocks have a value of $MAD_C(i)$ in [0, 1).

As shown in Tables 4.7, 4.10, 4.13 and 4.16, the search complexity for the sequence *Mother & Daughter* is significantly reduced by more than 33% in the case of ET-I. Also, more than 38% of the savings in the search complexity can be achieved for the sequence *Container*, which contains only minor motions, by using the scheme ET-I. Therefore, it is seen from the simulation results that compared to the 3-D full search, ET-I can effectively reduce the search complexity for areas that contain minor motions or stationary backgrounds, while incurring only negligible losses in the motion estimation accuracy for the MFBME.

Compared to ET-I, ET-II achieves greater savings in the search complexity at the expense of a loss in the PSNR, both the savings and the loss being related to the number of blocks whose $MAD_C(i)$ values fall in the interval [12, 255]. A larger number of blocks falling in this interval may result in a greater saving in the search

Table 4.6: Average difference in the PSNR between the proposed method and the FS, both of which using the MSEA in the spatial domain

Mem. depth		Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	PSNR of FS in dB	30.86	41.64	28.80	41.86	22.68	22.93	29.56
	PSNR diff. between ET-I and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	PSNR diff. between ET-II and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	PSNR of FS in dB	31.72	41.94	29.74	42.12	23.07	23.46	30.42
	PSNR diff. between ET-I and FS	-0.02	-0.02	-0.02	-0.00	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.11	-0.02	-0.11	-0.00	-0.10	-0.34	-0.16
10	PSNR of FS in dB	31.88	42.03	29.96	42.32	23.13	23.57	30.62
	PSNR diff. between ET-I and FS	-0.01	-0.02	-0.03	-0.01	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.14	-0.02	-0.24	-0.01	-0.14	-0.40	-0.22
15	PSNR of FS in dB	31.98	42.05	30.08	42.32	23.16	23.59	30.64
	PSNR diff. between ET-I and FS	-0.02	-0.03	-0.02	-0.01	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.17	-0.03	-0.29	-0.01	-0.16	-0.41	-0.22

Table 4.7: Computational complexity in terms of the number of equivalent MAD calculations for the proposed method and the FS, both of which using the MSEA in the spatial domain.

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	FS	19	31	24	12	42	49	38
	ET-I	19 (0%)	31 (0%)	24 (0%)	12 (0%)	42 (0%)	49 (0%)	38 (0%)
	ET-II	19 (0%)	31 (0%)	24 (0%)	12 (0%)	42 (0%)	49 (0%)	38 (0%)
5	FS	57	140	75	48	153	209	161
	ET-I	51 (-9%)	82 (-42%)	70 (-7%)	30 (-38%)	153 (-0%)	198 (-5%)	161 (-0%)
	ET-II	49 (-14%)	82 (-42%)	62 (-17%)	29 (-39%)	119 (-22%)	142 (-32%)	158 (-2%)
10	FS	90	269	119	86	262	358	285
	ET-I	77 (-15%)	93 (-65%)	107 (-10%)	40 (-53%)	262 (-0%)	327 (-9%)	285 (-0%)
	ET-II	71 (-22%)	93 (-65%)	88 (-27%)	37 (-57%)	173 (-33%)	194 (-45%)	279 (-2%)
15	FS	121	395	156	120	356	485	390
	ET-I	100 (-17%)	104 (-73%)	138 (-11%)	49 (-60%)	356 (-0%)	439 (-9%)	390 (-0%)
	ET-II	90 (-25%)	104 (-73%)	110 (-29%)	45 (-62%)	220 (-38%)	237 (-51%)	380 (-2%)

Note: The quantities in the parenthesis indicate the difference in percentage of the computational complexities of the FS and ET-I (or FS and ET-II).

Table 4.8: Percentage of the temporal motion vectors correctly estimated by the proposed early termination method using the MSEA in the spatial domain, as compared to those of the FS

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	100%	100%	100%	100%	100%	100%	100%
5	ET-I	98%	85%	97%	95%	100%	98%	100%
	ET-II	97%	85%	95%	95%	97%	84%	99%
10	ET-I	97%	76%	96%	93%	100%	97%	100%
	ET-II	96%	76%	94%	93%	96%	81%	98%
15	ET-I	96%	74%	95%	92%	100%	97%	100%
	ET-II	95%	74%	93%	92%	95%	81%	98%

complexity and a bigger loss in the PSNR at the same time. For example, *Garden* is the sequence with the highest percentage of blocks in [12,255] (see Table 4.5). As shown in the case of a five-frame memory in Table 4.7, ET-II achieves an additional saving of 27% in the search complexity for this sequence compared to ET-I, but suffers from a loss of 0.34 dB in the PSNR. It can be expected that if the lower bound of this interval increases, both the savings in the search complexity and the loss in the accuracy of the motion estimation will decrease.

Tables 4.8, 4.11, 4.14 and 4.17 show, for each of the sequences, the accuracy of the temporal motion vectors estimated using the two schemes. It is seen from these tables that an accuracy in excess of 90% is achieved by either of the schemes for all the sequences, except for the sequences *Flower Garden* (in the case of ET-II) and *Mother & Daughter* (in the case of MAD-based optimization). A relatively lower accuracy for the sequence *Flower Garden* in the case of ET-II is due to the fact that for a large number of the blocks, the $MAD_C(i)$ values are within the range [12, 255], and these blocks are targeted by the scheme ET-II for early termination. In such a

Table 4.9: Average difference in the PSNR between the proposed method and the FS, both of which using the FGSE in the spatial domain.

Mem. depth		Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	PSNR of FS in dB	30.86	41.64	28.80	41.86	22.68	22.93	29.56
	PSNR diff. between ET-I and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	PSNR diff. between ET-II and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	PSNR of FS in dB	31.72	41.94	29.74	42.12	23.07	23.46	30.42
	PSNR diff. between ET-I and FS	-0.02	-0.02	-0.02	-0.00	-0.00	-0.01	-0.00
	PSNR diff. between ET-II and FS	-0.11	-0.02	-0.11	-0.00	-0.11	-0.34	-0.16
10	PSNR of FS in dB	31.88	42.03	29.96	42.32	23.13	23.57	30.62
	PSNR diff. between ET-I and FS	-0.02	-0.02	-0.03	-0.01	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.14	-0.02	-0.24	-0.01	-0.14	-0.40	-0.22
15	PSNR of FS in dB	31.98	42.05	30.08	42.32	23.16	23.59	30.64
	PSNR diff. between ET-I and FS	-0.02	-0.03	-0.02	-0.01	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.17	-0.03	-0.29	-0.01	-0.16	-0.41	-0.22

Table 4.10: Computational complexity in terms of the number of equivalent MAD calculations for the proposed method and the FS, both of which using the FGSE in the spatial domain.

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	FS	19	27	22	13	36	39	31
	ET-I	19 (0%)	27 (0%)	22 (0%)	13 (0%)	36 (0%)	39 (0%)	31 (0%)
	ET-II	19 (0%)	27 (0%)	22 (0%)	13 (0%)	36 (0%)	39 (0%)	31 (0%)
5	FS	55	114	69	47	128	165	126
	ET-I	51 (-7%)	70 (-39%)	65 (-6%)	28 (-40%)	128 (-0%)	156 (-5%)	126 (-0%)
	ET-II	49 (-11%)	70 (-39%)	59 (-14%)	28 (-40%)	101 (-21%)	114 (-31%)	124 (-2%)
10	FS	90	215	108	83	221	276	224
	ET-I	79 (-12%)	82 (-62%)	100 (-7%)	36 (-57%)	221 (-0%)	256 (-7%)	224 (-0%)
	ET-II	74 (-18%)	82 (-62%)	86 (-20%)	36 (-57%)	150 (-32%)	160 (-42%)	220 (-2%)
15	FS	123	313	148	110	303	376	310
	ET-I	104 (-15%)	95 (-70%)	132 (-11%)	45 (-59%)	303 (-0%)	357 (-5%)	310 (-0%)
	ET-II	96 (-22%)	95 (-70%)	111 (-25%)	43 (-61%)	209 (-31%)	197 (-48%)	304 (-2%)

Note: The quantities in the parenthesis indicate the difference in percentage of the computational complexities of the FS and ET-I (or FS and ET-II).

Table 4.11: Percentage of the temporal motion vectors correctly estimated by the proposed early termination method using the FGSE in the spatial domain, as compared to those of the FS.

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	100%	100%	100%	100%	100%	100%	100%
5	ET-I	98%	85%	97%	95%	100%	98%	100%
	ET-II	97%	85%	95%	95%	97%	84%	99%
10	ET-I	97%	76%	96%	93%	100%	97%	100%
	ET-II	96%	76%	94%	93%	96%	81%	98%
15	ET-I	96%	74%	95%	92%	100%	97%	100%
	ET-II	95%	74%	93%	92%	95%	81%	98%

case, the computational complexity is reduced at the expense of a loss in the search accuracy. The sequence *Mother & Daughter*, as mentioned earlier in Section 4.2.3, contains only minor motions and a stationary background, and hence, the block-matching error obtained using only the first reference frame is already small enough, and the lower accuracy of the estimated temporal motion vectors shown in Table 4.8 and Table 4.11 has no effect on the accuracy of the motion estimation itself.

4.5 Summary

In this chapter, we have investigated, in the case of the multi-frame block-matching motion estimation, the statistical characteristics of the temporal motion vector as well as the reduction in the block-matching error due to an increased memory depth. Our findings confirm the widely-used assumption that the best block matching most likely occurs in the first few reference frames of the memory.

In view of this, we have developed a method for an early termination of the temporal search for block matching, and it is based on keeping track of the block-

Table 4.12: Average difference in the PSNR between the proposed method and the FS using the MSEA in the spatial domain and the rate-distortion optimization

Mem. depth		Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	PSNR of FS in dB	30.81	41.64	28.72	41.86	22.67	22.91	29.56
	PSNR diff. between ET-I and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	PSNR diff. between ET-II and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	PSNR of FS in dB	31.61	41.86	29.64	42.08	23.04	23.45	30.39
	PSNR diff. between ET-I and FS	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.08	-0.00	-0.10	-0.00	-0.09	-0.35	-0.15
10	PSNR of FS in dB	31.74	41.91	29.83	42.20	23.09	23.57	30.57
	PSNR diff. between ET-I and FS	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.11	-0.00	-0.19	-0.00	-0.13	-0.42	-0.21
15	PSNR of FS in dB	31.82	41.92	29.93	42.20	23.11	23.59	30.59
	PSNR diff. between ET-I and FS	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.13	-0.00	-0.24	-0.00	-0.14	-0.43	-0.21

Table 4.13: Computational complexity in terms of the number of equivalent MAD calculations of the proposed method and that of the FS using using the MSEA in the spatial domain and the rate-distortion optimization.

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	FS	19	13	25	12	41	44	36
	ET-I	19 (0%)	13 (0%)	25 (0%)	12 (0%)	41 (0%)	44 (0%)	36 (0%)
	ET-II	19 (0%)	13 (0%)	25 (0%)	12 (0%)	41 (0%)	44 (0%)	36 (0%)
5	FS	68	55	87	55	152	181	144
	ET-I	64 (-6%)	37 (-33%)	83 (-5%)	30 (-45%)	152 (-0%)	179 (-1%)	144 (-0%)
	ET-II	61 (-10%)	37 (-33%)	74 (-14%)	30 (-45%)	116 (-24%)	123 (-33%)	141 (-2%)
10	FS	121	103	151	103	266	313	253
	ET-I	111 (-8%)	56 (-45%)	139 (-8%)	41 (-60%)	266 (-0%)	310 (-1%)	252 (-0%)
	ET-II	104 (-14%)	56 (-45%)	118 (-22%)	39 (-62%)	175 (-34%)	171 (-45%)	245 (-3%)
15	FS	173	150	210	152	369	435	348
	ET-I	157 (-9%)	75 (-50%)	190 (-10%)	50 (-67%)	368 (-0%)	429 (-1%)	347 (-0%)
	ET-II	144 (-17%)	75 (-50%)	158 (-25%)	48 (-68%)	230 (-38%)	217 (-50%)	336 (-4%)

Note: The quantities in the parenthesis indicate the difference in percentage of the computational complexities of the FS and ET-I (or FS and ET-II).

Table 4.14: Percentage of the temporal motion vectors correctly estimated by the proposed early termination method using using the MSEA in the spatial domain and the rate-distortion optimization, as compared to those of the FS using the rate-distortion optimization.

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	100%	100%	100%	100%	100%	100%	100%
5	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	99%	100%	98%	100%	97%	87%	99%
10	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	99%	100%	97%	100%	97%	85%	98%
15	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	99%	100%	97%	100%	96%	85%	98%

matching error between any given block and its matching block on a reference frame by reference frame basis, and terminating the temporal search using a strategy that classifies a current block based on the mean absolute difference between the current block and its matching block in each reference frame. Using the strategy, two schemes, ET-I and ET-II, have been developed for the early termination, based on whether the mean absolute difference lies in the interval [0, α) with $\alpha \ll 255$ or in the interval $[0, \alpha) \cup [\beta, 255]$ with $\alpha \ll 255$ and $\beta \gg 0$. For each of these two schemes, the block-matching motion estimation has been carried out using the mean-absolute-difference-based optimization as well as the rate-distortion optimization, which shows that ET-I with α=1 incurrs no degradation in the motion estimation accuracy for the MFBME, and therefore, can be considered a lossless scheme. Also, the savings in the search complexity is motion-dependent, and significant savings can be obtained for areas that contain minor or stationary motions. ET-II with α=1 and β=12 provides additional savings in the search complexity compared to that obtained by ET-I, at the expense of some small loss in the accuracy of the motion estimation. The lower bound of the

Table 4.15: Average difference in the PSNR between the proposed method and the FS using the FGSE in the spatial domain and the rate-distortion optimization

Mem. depth		Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	PSNR of FS in dB	30.81	41.64	28.72	41.86	22.67	22.91	29.56
	PSNR diff. between ET-I and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	PSNR diff. between ET-II and FS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	PSNR of FS in dB	31.61	41.86	29.64	42.08	23.04	23.45	30.39
	PSNR diff. between ET-I and FS	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.08	-0.00	-0.10	-0.00	-0.09	-0.35	-0.15
10	PSNR of FS in dB	31.74	41.91	29.83	42.20	23.09	23.57	30.57
	PSNR diff. between ET-I and FS	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.11	-0.00	-0.19	-0.00	-0.13	-0.42	-0.21
15	PSNR of FS in dB	31.82	41.92	29.93	42.20	23.11	23.59	30.59
	PSNR diff. between ET-I and FS	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
	PSNR diff. between ET-II and FS	-0.13	-0.00	-0.24	-0.00	-0.14	-0.43	-0.21

Table 4.16: Computational complexity in terms of the number of equivalent MAD calculations of the proposed method and that of the FS using using the FGSE in the spatial domain and the rate-distortion optimization.

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	FS	17	13	20	13	31	31	26
	ET-I	17 (0%)	13 (0%)	20 (0%)	13 (0%)	31 (0%)	31 (0%)	26 (0%)
	ET-II	17 (0%)	13 (0%)	20 (0%)	13 (0%)	31 (0%)	31 (0%)	26 (0%)
5	FS	62	52	73	52	114	128	106
	ET-I	59 (-5%)	35 (-33%)	70 (-5%)	28 (-45%)	114 (-0%)	127 (-0%)	106 (-0%)
	ET-II	56 (-10%)	35 (-33%)	63 (-13%)	27 (-47%)	89 (-22%)	89 (-31%)	103 (-2%)
10	FS	113	99	130	99	204	229	190
	ET-I	104 (-8%)	53 (-46%)	119 (-8%)	37 (-62%)	204 (-0%)	226 (-1%)	190 (-0%)
	ET-II	98 (-13%)	53 (-46%)	105 (-19%)	36 (-63%)	140 (-31%)	132 (-42%)	185 (-3%)
15	FS	164	146	185	146	288	322	268
	ET-I	148 (-10%)	72 (-51%)	165 (-11%)	45 (-69%)	288 (-0%)	318 (-1%)	267 (-0%)
	ET-II	139 (-15%)	72 (-51%)	143 (-22%)	44 (-70%)	189 (-35%)	170 (-47%)	258 (-4%)

Note: The quantities in the parenthesis indicate the difference in percentage of the computational complexities of the FS and ET-I (or FS and ET-II).

Table 4.17: Percentage of the temporal motion vectors correctly estimated by the proposed early termination method using using the FGSE in the spatial domain and the rate-distortion optimization, as compared to those of the FS using the rate-distortion optimization.

Memory depth	Scheme	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
1	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	100%	100%	100%	100%	100%	100%	100%
5	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	99%	100%	98%	100%	97%	87%	99%
10	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	99%	100%	97%	100%	97%	85%	98%
15	ET-I	100%	100%	100%	100%	100%	100%	100%
	ET-II	99%	100%	97%	100%	96%	85%	98%

interval [β, 255] affects the performance of scheme ET-II; the larger the value of β, the lower the savings in the search complexity and the smaller the loss in the accuracy of the motion estimation.

With the spatial full search complexity significantly reduced by the fast full search algorithm such as the MSEA and the FGSE, and the temporal search complexity reduced by the early termination method proposed in this chapter, the approach of combining the two would lead to a low-complexity high-accuracy implementation for multi-frame block-matching motion estimation.

Chapter 5

Low-Power Successive-Elimination-Assisted Full-Search Architecture for Block-Matching Motion Estimation

5.1 Introduction

As mentioned in Chapter 2, the regularity of the data flow of the block-matching motion estimation (BME) has led to a two-dimensional full-search systolic array (FSSA) architecture for VLSI implementation of the BME [64]. In spite of the excellent real-time performance of the 2-D FSSA, it is a full-search implementation of the BME without making use of any existing algorithm-level simplification [45], [46], [48]. In a scheme proposed in [65], the computational complexity and the power consumption of the 2-D FSSA are reduced through the application of the successive elimination

algorithm (SEA) [45]. In such an SEA-assisted 2-D FSSA system, the SEA module calculates a lower bound of the block-matching error for each motion vector candidate, and eliminates the candidate if the lower bound is greater than the current minimum block-matching error. In view of this, the corresponding operations in the 2-D FSSA module are disabled, leading to a significant reduction in the switching activities, and hence, a reduction in the power consumption inside the 2-D FSSA.

The performance of the SEA in terms of the number of eliminated motion vector candidates is sensitive to the scanning pattern adopted within the search window [48]. Since in most cases, the optimum motion vector is found in the central part of the search window, an outward spiral scan starting from the center of the search window, as shown in Fig. 5.1, eliminates a larger number of motion vector candidates than a raster scan does. In this chapter, we propose a pseudo-spiral scan to replace the conventional raster scan for the purpose of increasing the number of eliminated motion vector candidates.

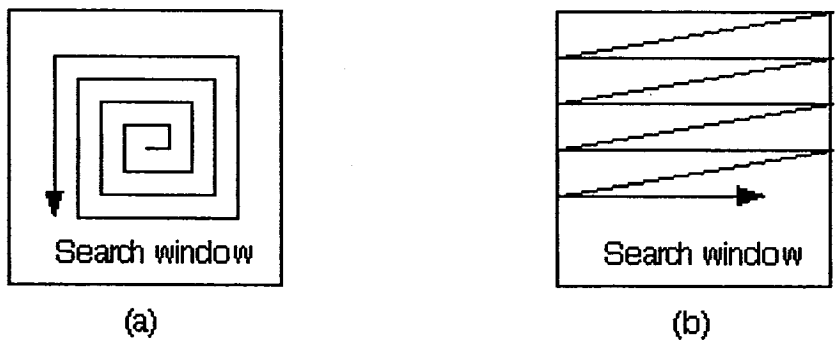


Figure 5.1: Two search patterns, (a) the spiral scan and (b) the raster scan, within the search window

In an effort to further reduce the power consumption of the system, several designs different from that proposed in [65] for the implementation of the SEA module are also considered, and the consequent reductions in the overall power consumption of

the SEA-assisted 2-D FSSA system are investigated using a power simulator.

This chapter [89] is organized as follows. In Section 5.2, a pseudo-spiral scanning scheme to be used in the search window is proposed. In Section 5.3, several designs to implement the successive elimination algorithm are developed based on a segmentation scheme of horizontally-stacked sub-blocks. In Section 5.4, the power consumption of the SEA-assisted 2-D FSSA system, resulting from the proposed pseudo-spiral scanning scheme and the various implementations of the SEA module, are evaluated using a power simulator that monitors the switching activities inside the system. Finally, certain conclusions are drawn in Section 5.5.

5.2 The Pseudo-Spiral Scan

Given the current block of size $N \times N$ pixels and a search range of $\pm p$ pixels in both dimensions, we define the following two terms in order to facilitate the development of the proposed pseudo-spiral scan.

- *Search window:* A set of reference blocks each of size $(N + 2p) \times (N + 2p)$ from which a matching block is to be identified for the current block.
- *block-row:* The set of reference blocks along a given row in a search window.

The pixels in a search window are input into the 2-D FSSA of [64] or [65] on a block-row by block-row basis, starting from the first block-row, as shown in Fig. 5.2(a). Scanning the search window in such a raster pattern is the key to the input data reuse that contributes to the real-time performance of the 2-D FSSA. On the other hand, for elimination-based full search algorithms, it is known that a spiral scan starting from the center of the search window can eliminate more impossible motion vector

candidates than the raster scan does [48]. Based on these two considerations, we propose the pseudo-spiral scan that contains two raster sub-scans, one starting from the *(p+1)-th* block-row and moving towards the bottom of the search window, and the other starting from the *p-th* block-row and moving towards the top of the search window, as shown in Fig. 5.2(b). Since both the sub-scans start from a block-row in the center of the search window, the pseudo-spiral scan is expected to hit the matching block sooner than the conventional raster scan does, resulting in more motion vector candidates being eliminated, thus resulting in larger savings in power.

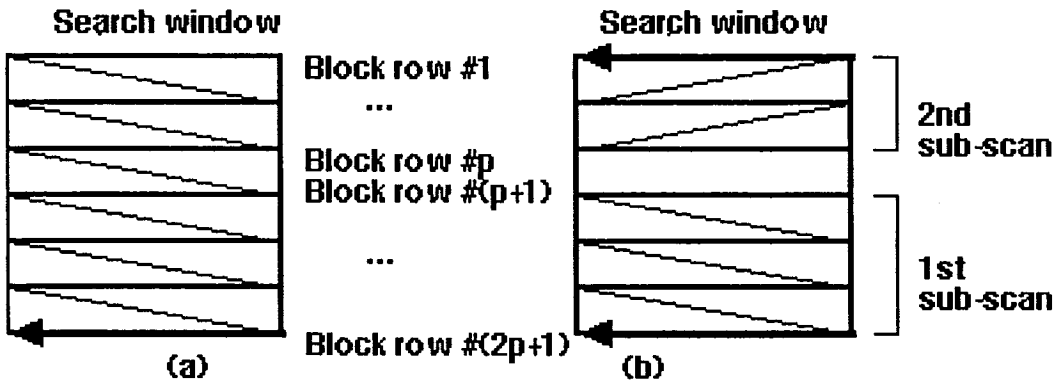


Figure 5.2: Two scanning patterns inside the search window: (a) the raster scan and (b) the pseudo-spiral scan

To evaluate the effectiveness of the pseudo-spiral scan in terms of the number of eliminated motion vector candidates, we conduct a software simulation of the SEA using the pseudo-spiral scan as well as the conventional raster scan. The number of consecutively-eliminated motion vector candidates resulting from each of the scans is obtained, as it is this kind of eliminations that results in a reduction of the switching activities, and hence, in a reduction of the power consumption. Seven test sequences are used in the simulation, and the details are given in Table 5.1. The number of frames used is more than one thousand, with QCIF, CIF and SIF formats to

represent the different frame resolutions. The sequences chosen cover different motion possibilities ranging from minor (e.g., *Mother&Daughter*) to more intensive motions (e.g., *Foreman*), and from mainly translational (e.g., *Container, Flower Garden*) to more complicated motions (e.g., *Football, Car Phone, Coast Guard*).

Table 5.1: Details of the seven sequences used in the simulation

Sequence	Frames	Format	Resolution
Foreman	1-400	QCIF	176x144
Mother&Daughter	1-400	QCIF	176x144
Car Phone	1-382	QCIF	176x144
Container	1-300	QCIF	176x144
Football	1-125	SIF	352x240
Flower Garden	1-115	SIF	352x240
Coast Guard	1-150	CIF	352x288

For each of the sequences, the percentage of motion vector candidates that are consecutively eliminated is shown in Table 5.2. It is seen from this table that the pseudo-spiral scan increases the consecutive eliminations by a significant percentage for each of the sequences considered, the lowest percentage being 14%. This indicates that, with the implementation of the pseudo-spiral scan, a further reduction in the power consumption of an SEA-assisted 2-D FSSA architecture is possible.

Table 5.2: Percentage of motion vector candidates that are consecutively eliminated by the SEA using the pseudo-spiral scan or the raster scan

	Foreman	Mother& Daughter	Car Phone	Container	Football	Flower Garden	Coast Guard
SEA using pseudo-spiral scan	74.2%	81.4%	76.5%	88.1%	44.1%	55.3%	53.6%
SEA using raster scan	54.2%	58.0%	56.0%	51.3%	29.6%	34.4%	30.8%
Difference	20.0%	23.4%	20.5%	36.8%	14.5%	20.9%	22.8%

For the pseudo-spiral scan, the starting block-row for the first and second raster sub-scans, namely, the *(p+1)-th* and *p-th* block-rows, have an overlap of *N-1* lines of pixels that have to be re-buffered at the beginning of the second raster sub-scan. Compared to the case of a conventional raster scan, re-buffering these *N-1* lines of pixels requires not only additional processing time for the system, but also an extra power consumption to shift these lines into the data input buffer. To avoid these, a twin data input buffer is added to the system, as shown in Fig. 5.3, where one of them is the working buffer, and the other a stand-by. During the input of the *(p+1)-th* block-row in the first raster sub-scan, when the working buffer is filled up, the data in the buffer are copied into the stand-by buffer. When the first raster sub-scan is finished, the stand-by buffer becomes the working buffer, and the *N-1* lines of the duplicated data in this buffer are input to the system as the *N-1* lines of the *p-th* block-row. As such, this buffering scheme does not require extra processing time or shift-registering power for the duplication of the *N-1* lines of the *p-th* block-row.

5.3 Implementation of the Successive Elimination

Given the current block C and one of its reference blocks R, both of size $N \times N$, the block matching is most commonly performed using as a measure the sum of absolute difference (SAD) given by

$$SAD_{C,R} = \sum_{i=1}^{N}\sum_{j=1}^{N}|C(i,j) - R(i,j)| \tag{5.1}$$

The SEA algorithms obtain one or more lower bounds of the SAD with calculations which are simpler than that for the SAD itself. If the lower bound is greater than the minimum SAD obtained so far, calculation of the current $SAD_{C,R}$ is not necessary.

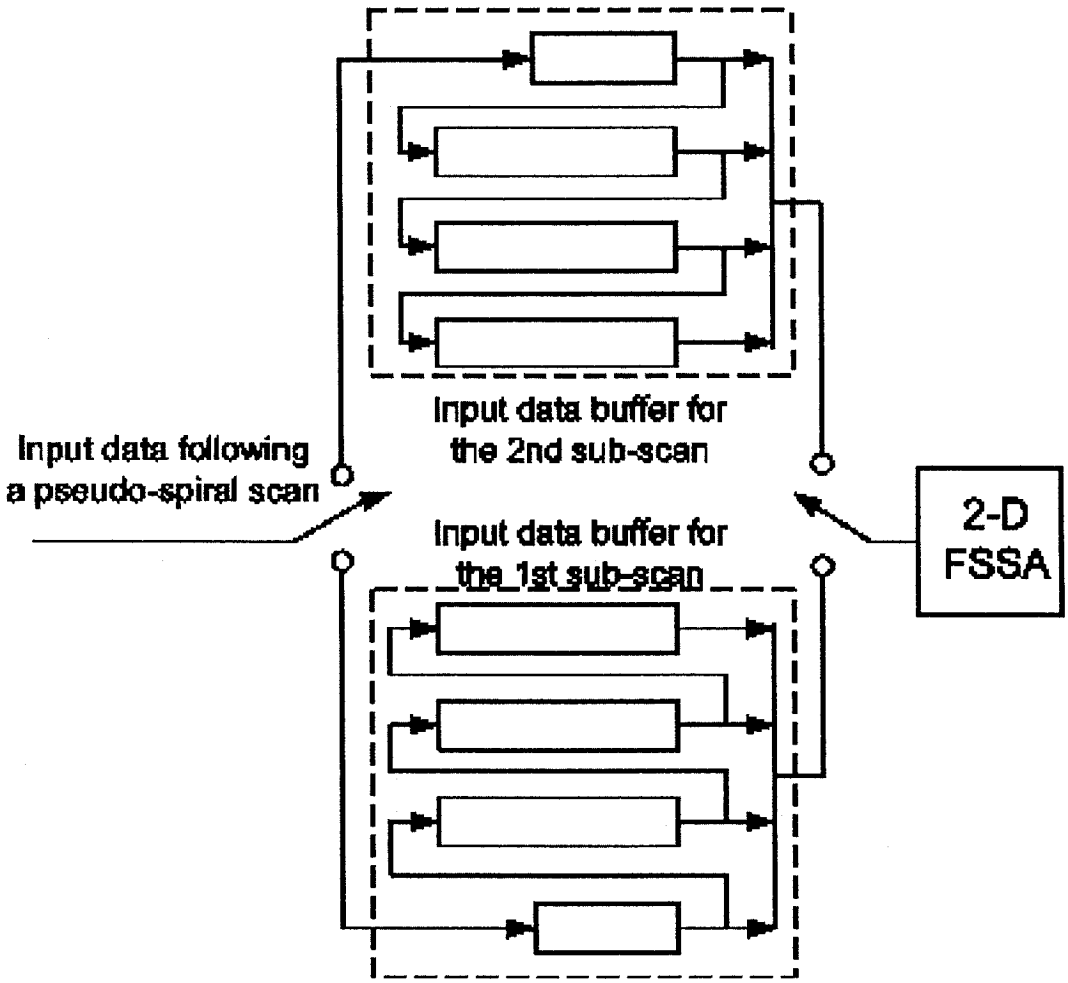


Figure 5.3: A twin data input buffer is used by the second raster sub-scan of the pseudo-spiral scan.

The essence of the SEA is to divide both the current block and the reference block into M sub-blocks, $\{C_1, C_2, ..., C_M\}$ and $\{R_1, R_2, ..., R_M\}$, respectively, so that the triangle inequality could be utilized to obtain a lower bound for the SAD in the form

$$LB = \sum_{k=1}^{M} |\sum_{i}\sum_{j}(R_k(i,j) - C_k(i,j))| \leq SAD_{C,R} \tag{5.2}$$

It is noted that the sum of any combination of the terms in (5.2) is also a lower bound of the SAD.

The SEA algorithm given in [45] is a special case of the above in which a single sub-block of size $N \times N$ is used. The resulting lower bound of the SAD is given as $|\sum_{i=1}^{N}\sum_{j=1}^{N} R(i,j) - \sum_{i=1}^{N}\sum_{j=1}^{N} C(i,j)|$, where the computational complexity of $\sum_{i=1}^{N}\sum_{j=1}^{N} R(i,j)$ can be reduced by reusing the terms common to two overlapping reference blocks. The scheme proposed in [65] is another example of the SEA, which utilizes *N-1* sub-blocks of size 1×N, as shown in Fig. 5.4(a); the corresponding lower bound is given by

$$LB_v = \sum_{k=1}^{N-1} |\sum_{1}^{1}\sum_{1}^{N} R_k(i,j) - \sum_{1}^{1}\sum_{1}^{N} C_k(i,j))| \tag{5.3}$$

The applications of the SEA can be recursive in that the sub-blocks can be further divided to generate tighter lower bounds of the SAD, the multi-level SEA (MSEA) [46] being one such example.

5.3.1 Implementing the Successive Elimination Algorithm

Along with the block-row by block-row scanning pattern used in the 2-D FSSA, a segmentation scheme that contains vertically-stacked *N-1* identical rectangular sub-blocks of size $1 \times N$, as shown in Fig.5.4(a), has been used for the implementation of the SEA in [65]. The corresponding lower bound for $SAD_{C,R}$ is given by (5.3),

as mentioned earlier. An alternate segmentation scheme that we could have used for the implementation of the SEA is shown in Fig.5.4(b), which contains horizontally-stacked M identical rectangular sub-blocks of size $(N-1) \times L$, where $M \times L = N$. It can be shown that in this case the lower bound of $SAD_{C,R}$ is given by

$$LB_h = \sum_{k=1}^{M} | \sum_{1}^{N-1} \sum_{1}^{L} R_k(i,j) - \sum_{1}^{N-1} \sum_{1}^{L} C_k(i,j))| \tag{5.4}$$

Any segmentation scheme, other than the horizontally- or vertically-stacked sub-blocks, used in the implementation of the SEA would severely hamper the ability to reuse the input data of the SEA module, resulting in a significant increase in the power consumption.

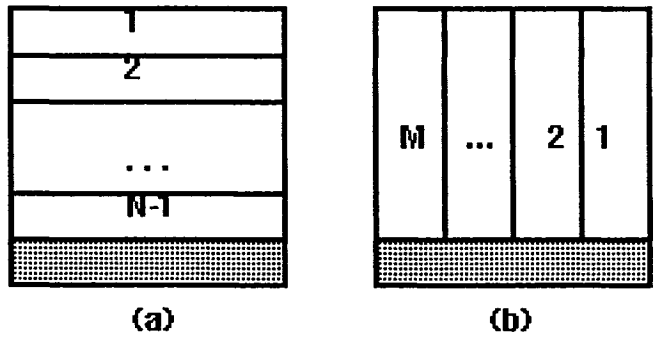


Figure 5.4: A block containing (a) vertically-stacked sub-blocks and (b) horizontally-stacked sub-blocks

Implementation based on vertically-stacked sub-blocks

The SEA module in [65] makes use of the first *N-1* rows of the reference block to obtain a lower bound of the SAD given by (5.3), so that the decision as to whether a motion vector candidate can be eliminated is made prior to the computation of the SAD, which begins with the inputting of the last row of the reference block to the system. This design requires *N-1* row accumulators and an absolute difference summation unit, as shown in Fig. 5.5.

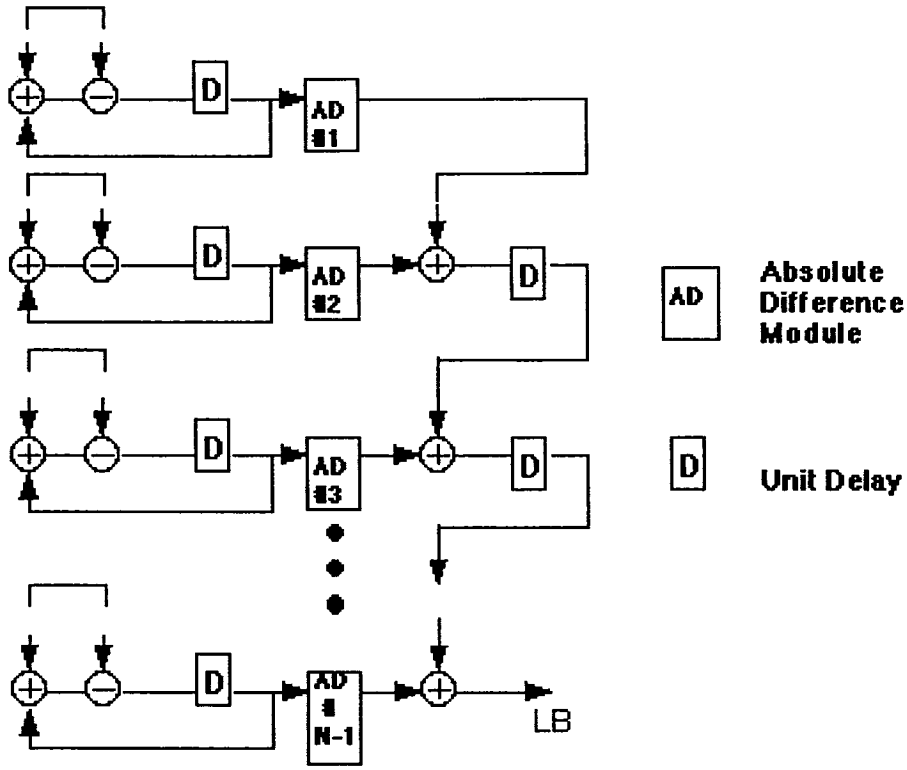


Figure 5.5: SEA design based on N-1 sub-blocks of sub-size 1×N. Each pair of inputs corresponds to a couple of pixel values, N registers apart, taken from one of the lines of the input data buffer.

Implementation based on horizontally-stacked sub-blocks

The SEA based on horizontally-stacked sub-blocks requires a different implementation. Our design uses a y-accumulator to calculate the column-wise sums of the contents of a block-row. An illustration of the y-accumulator is shown in box A of Fig. 5.6 for the case of the SEA using one sub-block of size $(N-1) \times N$, and in box D for the case of the SEA using four sub-blocks, each of size $(N-1) \times \frac{N}{4}$. The column-wise sums are stored in the column-sum buffer that consists of $N+2p$ shift registers corresponding to the block-row length. When processing the next block-row, the column-wise sums are updated by adding the corresponding contents of the last line of the new block-row and subtracting the first line of the previous block-row. As

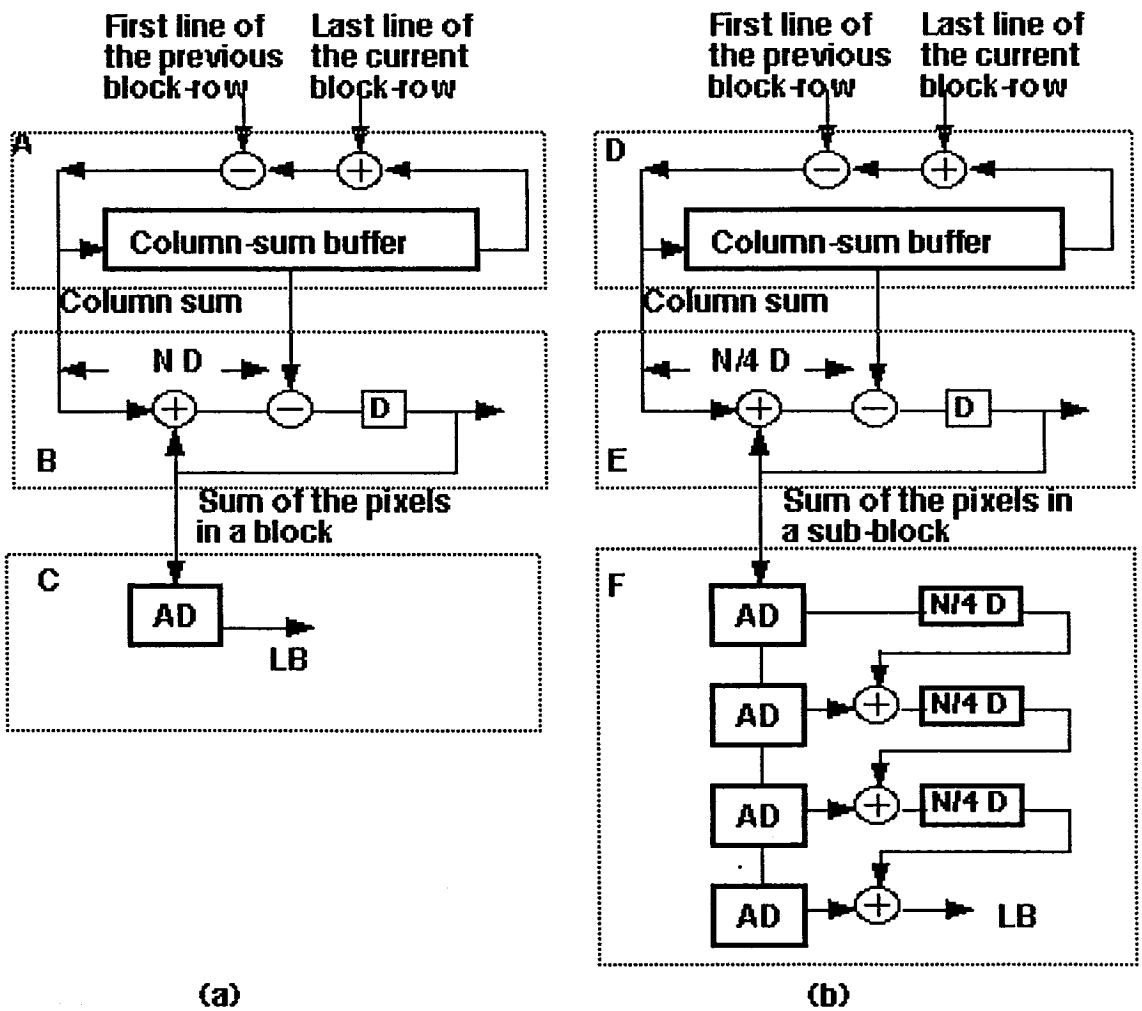


Figure 5.6: SEA based on (a) one single sub-block of size $(N-1) \times N$ and (b) four horizontally-stacked sub-blocks, each of size $(N-1) \times \frac{N}{4}$.

in the case of the vertically-stacked sub-blocks, the y-accumulator only accumulates the first N-1 lines of a block-row so that the elimination signal can be generated prior to the SAD calculation in the 2-D FSSA for each of the motion vector candidates.

The column-wise sums are then added up by the x-accumulator to obtain the sum of the contents of a sub-block, as shown in box B of Fig. 5.6 for the case of the SEA using one sub-block of size $(N-1) \times N$, or in box E of Fig. 5.6 for the case of the SEA using four sub-blocks, each of size $(N-1) \times \frac{N}{4}$. There is a delay of $\frac{N}{M}$ units (M being the number of sub-blocks) between the two inputs to the x-accumulator, and this delay corresponds to the width of a sub-block. Finally, as shown in boxes C and F of Fig. 5.6, a one-dimensional absolute difference array generates the lower bound of the SAD that is to be compared with the current minimum SAD.

In view of the fact that a twin input data buffer is used to store the contents of *N-1* lines of the *(p+1)-th* block-row for the case of the pseudo-spiral scan, as described in Section 5.2, a twin column-sum buffer, as shown in Fig. 5.7, must be used to store the column-wise sums resulting from these *N-1* lines of data. These sums will be reused as the column-wise sums for the *p-th* block-row at the beginning of the second raster sub-scan.

5.3.2 Two Levels of Successive Elimination

As mentioned earlier, dividing a block into sub-blocks is expected to result in more eliminated motion vector candidates, and hence, in increased power savings in the 2-D FSSA module. On the other hand, the implementation of the SEA with an increased number of sub-blocks increases the complexity of the circuitry and hence, results in a higher power consumption in the SEA module. This may neutralize the savings

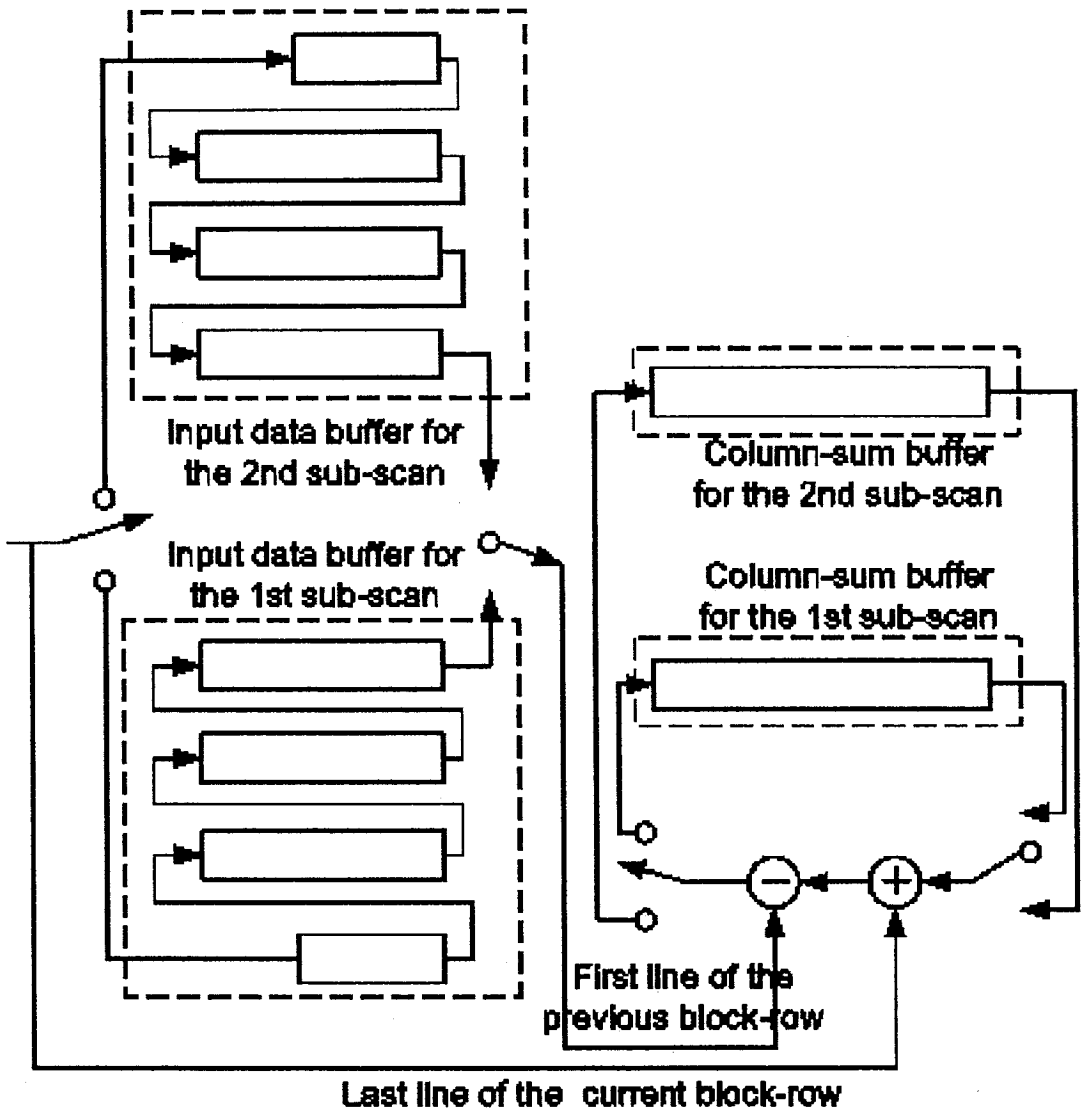


Figure 5.7: Twin buffer used to store the column-wise sum of the *(p+1)-th* block-row

achieved in the 2-D FSSA module. Thus, it is desirable to use an SEA with as many sub-blocks as possible, while at the same time keeping the power consumption low. For this purpose, we consider a two-level SEA as shown in Fig. 5.8. The lower level SEA has a number of sub-blocks, whereas the upper-level one uses the whole block of size $(N-1) \times N$. The disable signal *dis0* generated by the upper level is not only used to disable the operations within the 2-D FSSA, but also those within the lower-level SEA. As such, the work load of the lower-level SEA can be reduced. An example of a two-level SEA based on horizontally-stacked sub-blocks is shown in Fig. 5.9, where the lower-level SEA is based on four sub-blocks of size $(N-1) \times \frac{N}{4}$. Each level of the SEA has its own x-accumulator as well as its own absolute difference unit, but they share the same y-accumulator and column-sum buffer.

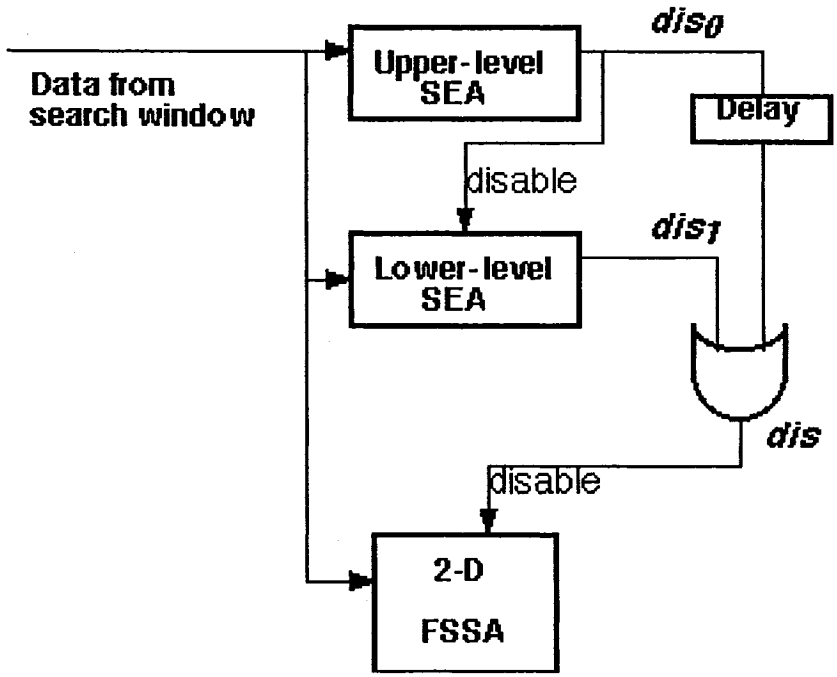


Figure 5.8: 2-D FSSA assisted by a two-level SEA

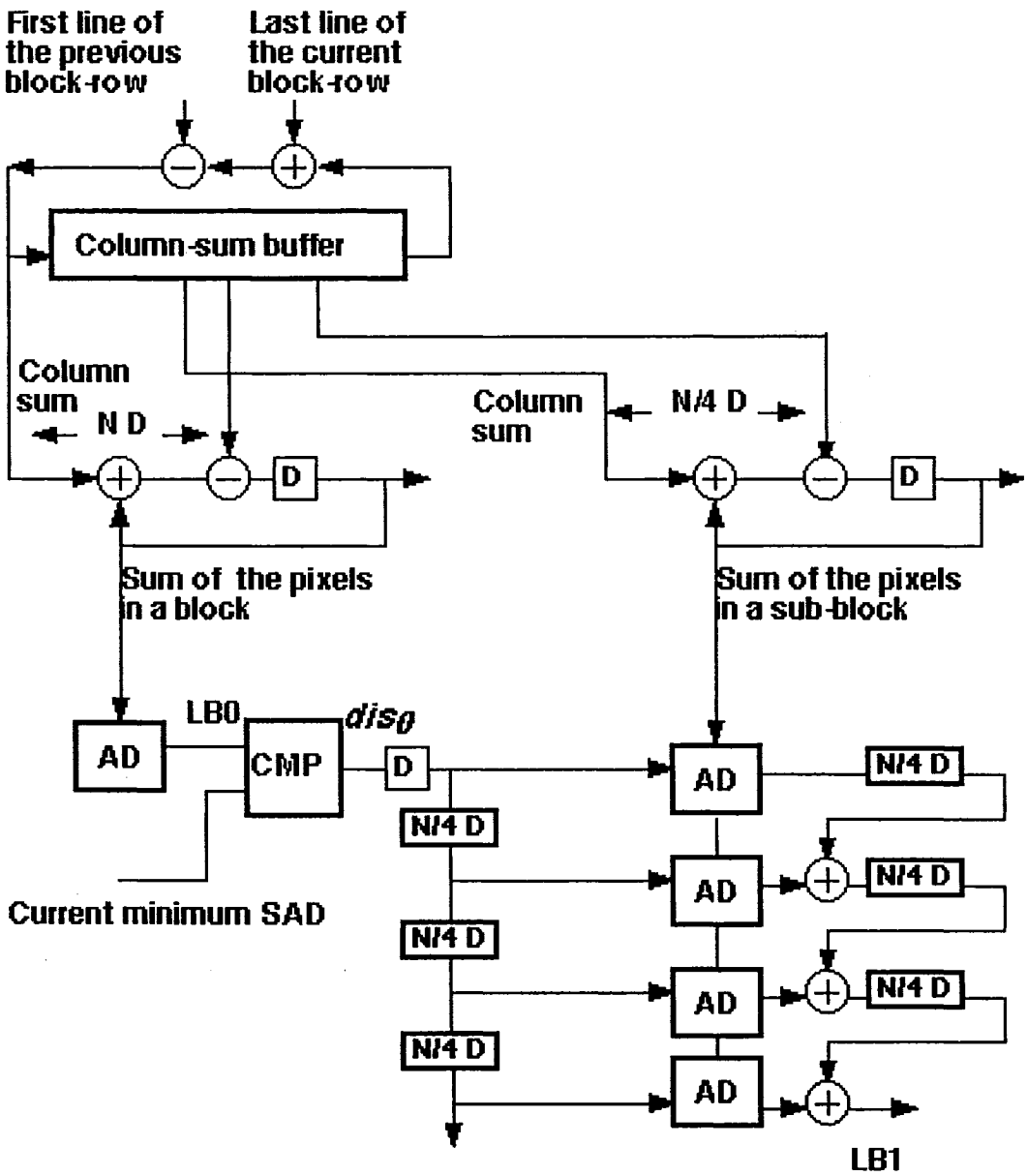


Figure 5.9: An example of a two-level SEA. The first level is an SEA with one sub-block of size $(N-1) \times N$, and the second level is an SEA with four sub-blocks of size $(N-1) \times \frac{N}{4}$.

5.4 Simulation of Power Consumption

For the evaluation of the power consumption of the SEA-assisted 2-D FSSA systems, we focus on the power consumption resulting from the switching activities in the system [90]. A simulator using Matlab is designed to simulate the functionalities of the 2-D FSSA and SEA, and monitor the bit-level switching activities in both modules. These two modules comprise the following operations: absolute difference, addition/subtraction, shift and comparison. As to the circuit-level components corresponding to these operations, we note that the operation of comparison can be implemented using a subtracter, and the absolute difference can be implemented as shown in Fig. 5.10, where the critical path contains a subtracter, a bit-wise inverter and an adder. The required circuit-level components are summarized in Table 5.3, where both the shift register and the inverter have one output, while the adder and subtracter have two outputs, one being the sum, and the other being the carry. The simulator counts the number of switching at the output of each of these components. The validity of the simulator is verified based on whether the resulting block-matching error is identical with that obtained from a direct matrix calculation when given an identical pair of the current block and reference block. It has been verified that the simulator is valid based on this criterion.

In order to evaluate the power consumption of the system, we define a unit of power consumption, denoted by P_{sum}, to be that required for switching one bit of the sum output in an adder/subtracter. Based on the complexity of the circuitry of the various components [90], the power consumption for switching one bit of the output in each of the other components is estimated as P_{sum} multiplied by a weighting factor. A weight of 0.25 units is assigned to the output of a one-bit inverter, a weight of one

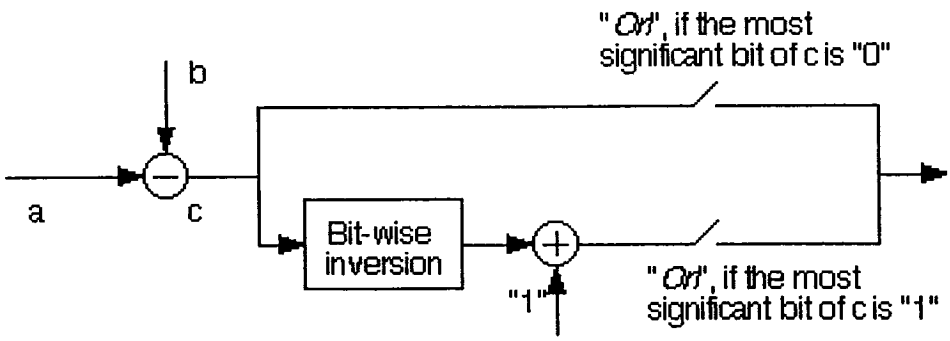


Figure 5.10: An absolute difference unit

unit to each bit of the carry output of the adder/subtracter, and a weight of one unit to the output of a one-bit shift register, as shown in Table 5.3. The total power consumption of the system for processing one frame is then estimated as

$$P = S_{add-sum} \times P_{add-sum} + S_{add-carry} \times P_{add-carry} + S_{shift} \times P_{shift} + S_{inv} \times P_{inv} \quad (5.5)$$

where

$S_{add-sum}$ and $S_{add-carry}$ denote, respectively, the numbers of switchings observed by the simulator at the two outputs of the adders during the processing of one frame,

S_{shift} and S_{inv} denote, respectively, the numbers of switchings observed by the simulator at the outputs of the shift registers and inverters during the processing of one frame,

$P_{add-sum}$ and $P_{add-carry}$ denote, respectively, the power consumptions for switching one bit at the two outputs of the adder, and

P_{shift} and P_{inv} denote, respectively, the power consumptions for switching one bit at the outputs of the shift registers and inverters.

The test frames used by the simulator are taken from the video sequences listed in Table 5.1 at a step size of 20 or 30 frames. The block size used is 16×16, and the search range is ± 15. Ten schemes of the SEA-assisted 2-D FSSA are considered. The details of the schemes in terms of the number of SEA levels, type of data scan in the search window, and the number and type of sub-blocks used are given in Table 5.4, where the scheme RAS_V16 corresponds to the one proposed in [65].

Table 5.3: Power consumption for switching one bit of the output of each component is estimated as P_{sum} multiplied by a weighting factor, where P_{sum} is the power consumption for switching one bit of the sum output in a full adder.

Adder/Subtracter		Shift Register	Inverter
P_{sum}	P_{carry}	P_{shift}	P_{inv}
P_{sum}	$1 \times P_{sum}$	$1 \times P_{sum}$	$0.25 \times P_{sum}$

The power consumption for processing each test frame is estimated using (5.6) for each of the schemes. Table 5.5 presents the frame-by-frame results for the sequence *Foreman*, where the power consumption for the conventional 2-D FSSA is recorded as a multiple of $10^6 P_{sum}$, and the power consumptions of the other schemes are shown as percentage savings compared to that of the 2-D FSSA. The average power consumption per frame for each of the sequences is shown in Table 5.6.

It is seen from these two tables that, in terms of the percentage power savings, the eight one-level SEA schemes form two distinct groups: one using raster scan and the other using pseudo-spiral scan. From Table 5.5, it is seen that for each test frame, the smallest saving of the latter group is higher than the largest saving of the former group by an amount ranging from 5.1% - 18.7%. Similarly from Table 5.6, it is seen that in terms of the average savings per frame, the smallest saving of the latter group is higher than the largest saving of the former group by an amount ranging from 5.4%

Table 5.4: Configuration of the various simulated SEA schemes

	2-D FSSA	RAS _V16	RAS _H1	RAS _H4	RAS _H16	SP _V16	SP _H1	SP _H4	SP _H16	SP2 _H4	SP2 _H16
Scan type: raster (RAS) or pseudo-spiral(SP)	RAS	RAS	RAS	RAS	RAS	SP	SP	SP	SP	SP	SP
No. of SEA levels	-	1	1	1	1	1	1	1	1	2	2
Sub-block type: vertically- (V) or horizontally- (H) stacked	-	V	-	H	H	V	-	H	H	H	H
No. of sub-blocks	-	16	1	4	16	16	1	4	16	4	16

Note: For SP2 schemes, the number of sub-blocks refers to the number at the lower level of the SEA.

- 30.4%. This clearly shows that the use of the pseudo-spiral scan significantly reduces the power consumption of the SEA-assisted 2-D FSSA compared to that using the raster scan.

Within the group using the pseudo-spiral scan, SP2_H16, the two-level SEA scheme using the maximum number of horizontally-stacked sub-blocks provides the largest savings; on the other hand, SP_H1, the one-level SEA scheme with a single sub-block, has the smallest savings except for the case of the sequence *Container*. The difference in the savings among the rest of the schemes is not significant.

Within the group using the raster scan, RAS_H1 , the one-level SEA scheme with a single sub-block is, in general, outperformed by the rest of the three schemes, the difference in the savings among the latter three schemes being not significant.

Based on the simulation results, a comparison of the power consumption of all the schemes is shown in Fig. 5.11(a) for the sequences in QCIF format and Fig. 5.11(b) for those in CIF or SIF format.

Table 5.5: Power consumption per frame, expressed as a multiple of $10^6 P_{sum}$, required to process certain frames of the *Foreman* sequence using the conventional 2-D FSSA, and the percentage savings obtained by using various schemes compared to that of the 2-D FSSA.

Fr.	Power of 2-D FSSA	Savings in power consumption									
		RAS _V16	RAS _H1	RAS _H4	RAS _H16	SP _V16	SP _H1	SP _H4	SP _H16	SP2 _H4	SP2 _H16
20	367	43.3%	39.3%	47.9%	45.3%	63.1%	57.8%	67.7%	65.1%	69.4%	70.4%
40	370	44.4%	41.4%	49.1%	46.3%	66.4%	63.1%	71.3%	68.3%	73.1%	74.2%
60	367	46.1%	45.2%	51.0%	47.9%	68.0%	68.7%	73.6%	69.9%	75.5%	76.3%
80	368	45.8%	42.9%	50.9%	48.0%	60.6%	56.0%	65.2%	62.8%	66.7%	67.9%
100	361	46.9%	45.3%	52.2%	49.4%	66.6%	64.7%	72.3%	69.1%	74.1%	75.0%
120	367	45.2%	43.4%	50.5%	47.4%	65.6%	62.6%	71.0%	67.9%	72.7%	73.6%
140	361	44.4%	41.0%	49.2%	46.6%	63.4%	59.2%	68.3%	65.6%	70.0%	71.1%
160	372	45.9%	42.8%	50.6%	47.9%	65.6%	62.3%	70.4%	67.6%	72.2%	73.4%
180	358	43.5%	39.9%	48.4%	45.7%	63.0%	57.6%	67.8%	65.2%	69.3%	70.5%
200	359	45.5%	42.9%	50.1%	47.3%	65.3%	61.2%	69.9%	67.1%	72.0%	73.2%
220	356	46.1%	44.1%	51.2%	48.2%	67.2%	65.1%	72.7%	69.3%	74.6%	75.5%
240	372	44.3%	41.1%	49.1%	46.4%	62.2%	57.1%	66.8%	64.3%	68.2%	69.4%
260	357	41.8%	39.9%	47.7%	44.8%	62.2%	58.5%	68.1%	65.2%	68.9%	69.8%
280	360	38.1%	31.7%	40.6%	38.7%	58.6%	49.4%	60.8%	59.3%	63.1%	64.9%
300	326	34.5%	30.6%	35.6%	34.8%	56.9%	49.8%	58.3%	57.3%	62.1%	64.4%
320	395	43.4%	33.3%	43.7%	43.6%	64.1%	52.2%	64.7%	64.3%	66.3%	69.3%
340	403	41.9%	29.1%	41.7%	42.9%	65.7%	48.7%	65.2%	66.7%	65.5%	70.2%
360	408	46.0%	37.1%	48.5%	47.2%	71.3%	63.2%	75.3%	72.5%	75.8%	77.0%
380	406	45.6%	38.5%	47.4%	45.7%	71.5%	66.1%	74.9%	71.5%	76.7%	77.5%
399	406	45.3%	36.7%	46.4%	45.5%	71.2%	63.6%	74.0%	71.3%	75.7%	77.0%
Avg.	372	43.9%	39.3%	47.6%	45.5%	64.9%	59.3%	68.9%	66.5%	70.6%	72.0%

5.5 Summary

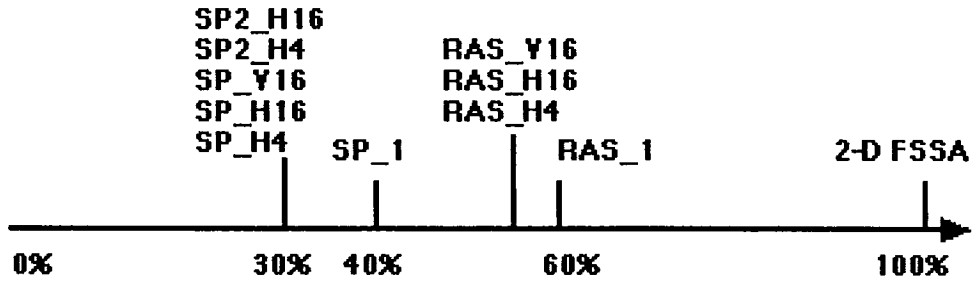
In this chapter, we have introduced a pseudo-spiral scan and used it in an existing architecture that combines the successive elimination algorithm with the conventional 2-D full search systolic array for the purpose of increasing the number of motion vector candidates that are eliminated. We have developed several schemes for the successive elimination algorithm that are based on horizontally-stacked sub-blocks introduced in this chapter. A simulator monitoring the circuit-level switching activities has been designed to estimate the power consumption of the various schemes using either the raster scan or the pseudo-spiral scan. The simulation results have shown that using

Table 5.6: Average power consumption per frame, expressed as a multiple of $10^6 P_{sum}$, required to process certain frames of the test sequences using the conventional 2-D FSSA, and the percentage savings obtained by using various schemes compared to that of the 2-D FSSA.

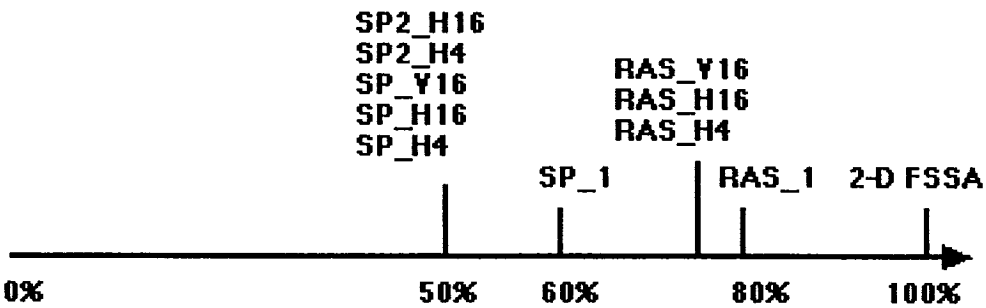
	Power of 2-D FSSA	Savings in power consumption									
		RAS _V16	RAS _H1	RAS _H4	RAS _H16	SP _V16	SP _H1	SP _H4	SP _H16	SP2 _H4	SP2 _H16
For	372	43.9%	39.3%	47.6%	45.5%	64.9%	59.3%	68.9%	66.5%	70.6%	72.0%
Mot	353	44.5%	43.1%	48.8%	45.9%	66.7%	66.3%	71.5%	68.1%	73.4%	74.2%
Car	346	44.9%	41.5%	49.0%	46.5%	64.8%	62.0%	69.7%	66.3%	71.6%	72.2%
Con	329	33.3%	36.8%	38.1%	34.9%	68.5%	71.9%	73.8%	70.1%	75.8%	76.6%
Foo	1516	27.4%	17.6%	26.1%	26.7%	48.6%	32.8%	45.2%	47.9%	45.7%	50.7%
Gar	1567	27.1%	18.5%	26.3%	26.4%	58.3%	39.2%	54.3%	57.7%	54.9%	61.2%
Coa	1669	15.1%	14.4%	18.4%	15.6%	42.0%	38.5%	44.1%	42.5%	44.8%	45.7%

Note: The sequence *Foreman* is sampled every 20 frames and the rest every 30 frames.

a pseudo-spiral scan can provide significant power savings compared to that using a raster scan. Among the various schemes considered for the implementation of the successive elimination algorithm, the one using two levels and the maximum number of horizontally-stacked sub-blocks has been shown to provide the maximum amount of power savings.



(a)



(b)

Figure 5.11: Comparison of the power consumptions of the ten SEA schemes: (a) for the QCIF sequences; (b) for the SIF and CIF sequences.

Chapter 6

Conclusion

6.1 Concluding Remarks

The block-matching motion estimation is an effective and most commonly used technique to reduce the temporal redundancy in digital video sequences. The elimination-based full search algorithms, as one category of block-matching motion estimation algorithms, have the advantage of providing a higher motion estimation accuracy than that by any partial search algorithm; the computational complexity of the former is generally of the same order as that of the latter. This has made the elimination-based full search a good competitor of the partial search in the BME. Motivated by this, several new techniques using the elimination-based full search have been developed in this thesis for software and hardware implementations of the BME.

A new full search algorithm for the BME that employs the mean squared error (MSE) as the matching criterion, and provides a motion estimation accuracy higher than that provided by any search algorithm which uses the mean absolute difference (MAD), has been developed. The MSE is calculated in the Haar wavelet domain

so that the larger terms of the MSE are given a higher priority for the purpose of elimination of the impossible motion vector candidates. It has been shown that this elimination strategy has led to a computational complexity, which is significantly lower than that of the existing partial distortion elimination algorithms, and lower than or comparable to that of the successive elimination algorithms.

For the multi-reference-frame block-matching motion estimation, we have investigated the statistical characteristics of the temporal motion vectors and the reduction of block-matching error resulting from an increase of the memory depth. Our findings confirmed the validity of the commonly-made assumption that the best block matching most likely occurs in the first few frames of the memory. Based on these findings, a new approach to the multi-reference-frame block-matching motion estimation has been investigated, wherein a full search is performed over the first few reference frames of the multi-reference-frame memory to provide a motion estimation accuracy that is higher than what can be achieved by any approach that uses a partial search. The computational complexity of the multi-reference-frame block-matching motion estimation has been significantly reduced by an early termination method proposed in the thesis, which keeps track of the block-matching error on a reference-frame-by-reference-frame basis and terminates the temporal search when further search is deemed unnecessary. It has been shown that this early termination method incurs no or a little loss in the motion estimation accuracy.

A new pseudo-spiral data input scheme, which can be used in any existing hardware architecture for the implementation of the successive-elimination-based block-matching motion estimation, has been proposed. Compared to the conventional raster data input scheme used in such an architecture, the proposed data input scheme has

been shown to provide significant power savings. Several designs, which are based on a new block segmentation scheme, have been given to implement the successive elimination algorithm. Some of these designs have been shown to provide additional savings in the power consumption of the system.

6.2 Scope for Future Investigation

The MSE-based fast full search algorithm proposed in Chapter 3 uses a four-level Haar wavelet transform, in which the lower-frequency region is decomposed into four sub-regions A, V, H and D at each level. Since it is in the sub-region A that the larger coefficients are expected to be concentrated, any other scheme that gives rise to the sub-region A and preserves the value of the MSE can also be considered for the purpose of developing alternate full search algorithms. An optimal scheme would be the one that requires the least number of look-up tables and has the lowest computational complexity.

For the multi-reference-frame block-matching motion estimation, it has been observed in this study that there is a high probability that the motion vector of a given block are identical to that of one or more of its neighboring blocks. Making use of this correlation might further reduce the computational complexity of the multi-reference-frame block-matching motion estimation. The investigation could be carried out in a broader framework, wherein blocks of variable size are used, as in the case of H.264.

In the study for reducing the power consumption of the 2-D full search systolic array, further investigation could be carried out concerning the development of various implementations of the successive elimination algorithms and new algorithms that can be applied to the 2-D full search systolic array instead of the successive elimination

algorithms. For example, the partial distortion elimination may be applied to the 2-D full search systolic array by adding additional comparison units at the output of certain absolute-difference-accumulation units. It is obviously not economical to add a comparison unit to each of the absolute-difference-accumulation units; hence, an interesting investigation would be to determine the optimal number and location of these comparison units.

Apart from the efforts to optimize the full search at the algorithmic level, an interesting study would be the processor-level optimization of the various fast full search algorithms, given the many features that the current computer processors can offer, such as the pipelined data processing units and the instruction-level pipelining.

References

[1] H. Mimura. Overview of video application for DVD media. In *Optical Data Storage Topical Meeting*, pages 162–164, 2006.

[2] M. Etoh and T. Yoshimura. Wireless video applications in 3G and beyond. *IEEE Wireless Communications*, 12(4):66–72, August 2005.

[3] A. Chatterjee. Overview of digital video in broadband networks. In *Digest of Technical Papers, ICCE International Conference on Consumer Electronics*, pages 378–380, June 1998.

[4] An overview of interactive video applications in the UK. In *IEE Colloquium on Interactive Video Applications*, pages 1–6, November 1990.

[5] Borivoje Furht. *Motion Estimation Algorithms for Video Compression.* Kluwer Academic, Boston, 1997.

[6] Y. Wang, J. Ostermann, and Y.Q. Zhang. *Video Processing and Communications.* Prentice Hall, 2002.

[7] Wolfgang Effelsberg. *Video Compression Techniques.* Dpunkt-Verlag, Heidelberg, Germany, 1998.

[8] J.B. Waltrich. Digital video compression – an overview. *Journal of Lightwave Technology*, 11(1):70–75, January 1993.

[9] R.J. Clarke. *Transform Coding of Images.* London; Orlando: Academic Press, 1985.

[10] G.K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[11] M.L. Hilton, B.D. Jawerth, and A. Sengupta. Compressing still and moving images with wavelets. *Multimedia Systems*, 2(5):218–227, 1994.

[12] M. Vetterli and J. Kovacevic. *Wavelet and Subband Coding.* Prentice Hall, 1995.

[13] K.R. Rao and P. Yip. *Discrete Cosine Transform – Algorithms, Advantages Applications.* Academic Press Inc., London, 1990.

[14] S. Kappagantula and K.R. Rao. Motion compensated interframe image prediction. *IEEE Transactions on Communications*, 33(9):1011–1015, September 1985.

[15] M. Ghanbari. *Standard Codecs: Image Compression to Advanced Video Coding.* Institution of Electrical Engineers, London, 2003.

[16] ISO/IEC 11172. *MPEG-1 Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to About 1.5 Mb/s*, 1993.

[17] ISO/IEC 13818. *MPEG-2 Coding of Moving Pictures and Associated Audio Information*, 1996.

[18] B.G. Haskell and A. Puri. *Digital Video: An Introduction to MPEG-2.* Kluwer Academic Publishers, 1996.

[19] ISO/IEC 14496-2. *MPEG-4–Information Technology–Coding of Audio-Visual Objects–Part 2: Visual*, 2000.

[20] International Telecommunications Union, ITU-T Recommendation H.261. *Video Codec for Audiovisual Services at P×64 kbits*, 1993.

[21] International Telecommunications Union, ITU-T Recommendation H.263. *Video Coding for Low Bitrate Communication*, 1998.

[22] Joint Video Team(JVT) of ISO/IEC MPEG and ITU-T VCEG, 7th Meeting: Pattaya, Thailand. *Joint Video Specification(ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC) - Joint Committee Draft*, 7-14 March 2003.

[23] M.L. Liou. Overview of the K×64 kbps video coding standard. *Communications of the ACM*, 34:47–58, April 1991.

[24] B.G. Haskell, P.G. Howard, Y.A. LeCun, A. Puri, J. Ostermann, M.R. Civanlar, L. Rabiner, L. Bottou, and P. Haffner. Image and video coding: emerging standards and beyond. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):814–817, November 1998.

[25] B. Girod. Motion compensation: visual aspects, accuracy and fundamental limits. In *Motion Analysis and Image Sequence Processing, Kluwer Academic Publishers*, 1993.

[26] H.-M. Hang, Y.-M. Chou, and S.-C Cheng. Motion estimation for video coding standards. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 17(2-3):113–136, November 1997.

[27] H.G. Musmann, P. Pirsch, and H.-J. Grallert. Advances in picture coding. In *Proceedings of the IEEE*, pages 523–548, 1985.

[28] C. Stiller and J. Konrad. Estimating motion in image sequences. *IEEE Signal Processing Magazine*, 16(4):70–91, July 1999.

[29] J.R. Jain and A.K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, 29(12):1799–1808, December 1981.

[30] M. Manikandan. Motion estimation methods for video compression – an overview. In *IFIP International Conference on Wireless and Optical Communications Networks*, pages 11–13, April 2006.

[31] M. Ghanbari. The cross-search algorithm for motion estimation. *IEEE Transactions on Communications*, 38(7):950–953, July 1990.

[32] T.Koga, K.Iinuma, A.Hirano, Y.Iijima, and T.Ishiguro. Motion compensated interframe coding for video conferencing. In *Proc. Nat. Telecommun. Conf. New Orleans*, pages G.5.3.1–5.3.5, 1981.

[33] R. Li, B. Zeng, and L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4):438–442, August 1994.

[34] R. Srinivasan and K.R. Rao. Predictive coding based on efficient estimation. *IEEE Transactions on Communications*, 33(8):888–895, August 1985.

[35] Chun-Ho Cheung and Lai-Man Po. Novel cross-diamond-hexagonal search algorithms for fast block motion estimation. *IEEE Transactions on Multimedia*, 7(1):16–22, February 2005.

[36] Ce Zhu, Xiao Lin, L. Chau, and Lai-Man Po. Enhanced hexagonal search for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(10):1210 – 1214, October 2004.

[37] Y. Keller and A. Averbuch. Fast motion estimation using bidirectional gradient methods. *IEEE Transactions on Image Processing*, 13(8):1042 – 1054, August 2004.

[38] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 9(2):287 – 290, February 2000.

[39] Jiajun Zhang. *New Motion Estimation and Compensation Techniques for Vdieo Compression.* PhD thesis, Concordia University, 1998.

[40] Jinwen Zan. *Median Filtering-based Hierarchical Motion Estimation Techniques for Video Compression.* PhD thesis, Concordia University, 2001.

[41] Jinwen Zan, M.O. Ahmad, and M.N.S. Swamy. New techniques for multiresolution motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(9):793–802, September 2002.

[42] Jinwen Zan, M.O. Ahmad, and M.N.S. Swamy. Multiplicationless burt and adelson's pyramids for motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):136–141, January 2004.

[43] Jinwen Zan, M.O. Ahmad, and M.N.S. Swamy. Wavelet-based multiresolution motion estimation through median filtering. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP '02)*, volume 4, pages 3273–3276, 2002.

[44] A. Murat Tekalp. *Digital Video Processing.* Prentice Hall, 1995.

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

[46] X.Q. Gao, C.J. Duanmu, and C.R. Zou. A multilevel successive elimination algorithm for block matching motion estimation. *IEEE Trans. Image Processing*, 9(3):501–504, March 2000.

[47] C. Zhu, W.S. Qi, and W. Ser. Predictive fine granularity successive elimination for fast optimal block-matching motion estimation. *IEEE Trans. Image Processing*, 14(2):213–221, February 2005.

[48] B. Montrucchio and D. Quaglia. New sorting-based lossless motion estimation algorithms and a partial distortion elimination performance analysis. *IEEE Trans. Circuits and Systems for Video Technology*, 15(2):210–220, February 2005.

[49] W.G. Hong and T.M. Oh. Sorting-based partial distortion search algorithm for motion estimation. *Electronics Letters*, 40(2):113–115, January 2004.

[50] Chou-Chen Wang, Chia-Jung Lo, and Cheng-Wei Yu. Efficient motion estimation using sorting-based partial distortion search. In *IEEE International Conference on Multimedia and Expo*, pages 153–156, 2006.

[51] Chok-Kwan Cheung and Lai-Man Po. Normalized partial distortion search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):417–422, April 2000.

[52] M. Brünig and W. Niehsen. Fast full-search block matching. *IEEE Trans. Circuits and Systems for Video Technology*, 11(2):241–247, February 2001.

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

[54] Jong-Nam Kim, Dae-Kap Kang, Sung-Cheal Byun, Il-Lo Lee, and Byung-Ha Ahn. A fast full search motion estimation algorithm using sequential rejection of candidates from hierarchical decision structure. *IEEE Transactions on Broadcasting*, 48(1):43–46, March 2002.

[55] Chun-Ho Cheung and Lai-Man Po. Adjustable partial distortion search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(1):100–110, January 2003.

[56] Jiajun Zhang, M.O. Ahmad, and M.N.S. Swamy. Dc matching for motion estimation. *Electronic Letters*, 33:1447–1448, 1997.

[57] Jiajun Zhang, M.O. Ahmad, and M.N.S. Swamy. A feature-bit-plane based technique for the estimation of motion vectors. *Electronic Letters*, 34:1090–1091, 1998.

[58] C.J. Duanmu, M.O. Ahmad, and M.N.S. Swamy. Fast block motion estimation with eight-bit partial sums using SIMD architectures. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(8):1041 – 1054, August 2007.

[59] Shuo Yao, Hai-Jun Guo, Lu Yu, and Ke Zhang. A hardware implementation for full-search motion estimation of avs with search center prediction. *IEEE Transactions on Consumer Electronics*, 52(4):1356 – 1361, November 2006.

[60] Shih-Chang Hsia. VLSI implementation for low-complexity full-search motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(7):613 – 619, July 2002.

[61] Jun-Fu Shen, Tu-Chih Wang, and Liang-Gee Chen. A novel low-power full-search block-matching motion-estimation design for h.263+. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(7):890 – 897, July 2001.

[62] H. Yee and Yu Hen Hu. A novel modular systolic array architecture for full-search block matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(5):407 – 416, October 1995.

[63] K. Parhi. *VLSI Digital Signal Processing Systems.* John Wiley & Sons, Inc, 1999.

[64] T. Komarek and P. Pirsch. Array architectures for block matching algorithms. *IEEE Trans. Circuits and Systems*, 36(10):1301–1308, October 1989.

[65] V.L. Do and K.Y. Yun. A low-power VLSI architecture for full-search block-matching motion estimation. *IEEE Trans. Circuits and Systems for Video Technology*, 8(4):393–398, August 1998.

[66] C.J. Duanmu, M.O. Ahmad, M.N.S. Swamy, and A. Shatnawi. A vector based fast block motion estimation algorithm for implementation on SIMD architectures. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages IV 337–340, 2002.

[67] C.J. Duanmu, M.O. Ahmad, and M.N.S. Swamy. 8-bit partial sums of 16 luminance values for fast block motion estimation. In *Proceedings of the IEEE International Conference on Multimedia and Expo(ICME)*, pages I 689–692, 2003.

[68] Chunjiang Duanmu. *New Motion Estimation Techniques and Their SIMD implementations for Video Coding.* PhD thesis, Concordia University, 2005.

[69] *VIS Instruction Set.* Sun Microsystems Inc. http://www.sun.com/processors/vis.

[70] Daniel Tabak. *RISC Systems and Applications.* Research Studies Press, 1996.

[71] *Understanding SIMD.* Apple Computer Inc http://developer.apple.com/hardwaredrivers/ve/simd.html, 2005.

[72] P. Pirsch, N. Demassieux, and W. Gehrke. VLSI architecture for video compression – a survey. *Proceedings of the IEEE*, 83(2):220–246, February 1995.

[73] L. de Vos and M. Stegherr. Parametrizable VLSI architectures for full-search block-matching algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, 36(10):1309–1316, October 1989.

[74] C. Hsieh and T. Lin. VLSI architecture for block-matching motion estimation algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 2(2):169–175, June 1992.

[75] K. Yang, M. Sun, and L. Wu. A family of VLSI designs for the motion compensation block-matching algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 36(10):1317–1325, October 1989.

[76] C. Yang, M.O. Ahmad, and M.N.S. Swamy. Fast full-search block-matching motion estimation using 2-d haar wavelet transform. *Submitted for publication.*

[77] Y. Naito, T. Miyazaki, and I. Kuroda. A fast full-search motion estimation method for programmable processors with a multiply-accumulator. In *Proc. ICASSP*, pages 3221–3224, 1996.

[78] S.G. Mallat. A theory for multiresolution signal decomposition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[79] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*, chapter 7. Prentice hall, 2002.

[80] J. Chalidabhongse and C.-C. J. Kuo. Motion vector estimation using multiresolution-spatio-temporal correlation. *IEEE Trans. Circuits and Systems for Video Technology*, 7(3):477–488, June 1997.

[81] Motorola Semiconductor Product Sector DSP Division. *DSP56300 Family Manual.* Motorola, 1995.

[82] M.C. Chen and A.N. Wilson. Rate-distortion optimal motion estimation algorithms for motion-compensated transform video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(2):147–158, April 1998.

[83] Thomas Wiegand, Xiaozheng Zhang, and Bernd Girod. Long-term memory motion-compensated prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):70–84, February 1999.

[84] X. Li, E.Q. Li, and Y.-K. Chen. Fast multi-frame motion estimation algorithm with adaptive search strategies in h.264. In *Proc. IEEE ICASSP '04*, volume 3, pages 369–372, May 2004.

[85] Hyukjune Chung and Antonio Ortega. Low complexity motion estimation algorithm by multiresolution search for long-term memory motion compensation. In *Proceedings of IEEE International Conference on Image Processing*, volume 2, pages 261–264, 2002.

[86] C.J. Duanmu, M.O. Ahmad, and M.N.S. Swamy. A continuous tracking algorithm for long-term memory motion estimation. In *Proceedings of the 2003 International Syposium on Circuits and Systems*, volume 2, pages 25–28, May 2003.

[87] M.E. Al-Mualla, C.N. Canagarajah, and D.R. Bull. Simplex minimization for single- and multiple-reference motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(12):1209–1220, December 2001.

[88] C. Yang, M.O. Ahmad, and M.N.S. Swamy. Temporal search complexity reduction in multi-frame block-matching motion estimation. *Under review.*

[89] C. Yang, M.O. Ahmad, and M.N.S. Swamy. Low-power successive-elimination-assisted full-search architecture for block-matching motion estimation. *Under review.*

[90] Sung-Mo Kang and Yusuf Leblebici. *CMOS Digital Integrated Circuits Analysis and Design.* WCB/McGraw-Hill, 1999.