ANALYSIS AND MOTION ESTIMATION STRATEGIES FOR FRAME AND VIDEO OBJECT CODING

HUI KO CHEUNG

PH. D.

THE HONG KONG POLYTECHNIC UNIVERSITY

2005

Analysis and Motion Estimation Strategies for Frame and Video Object Coding

Student Name:	Hui Ko Cheung
Supervisor Name:	Prof. Siu Wan Chi
Date:	August 2004

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signed)

Hui Ko Cheung (Name of student)

Abstract

Block-based motion estimation is widely used for exploiting temporal redundancies in arbitrarily shaped video objects, which is computationally the most demanding part within the MPEG-4 standard. One of the main differences of MPEG-4 video in comparison to previously standardized video coding schemes is the support of arbitrarily shaped video objects for which the numerous existing fast motion estimation algorithms are not suitable. The conventional fast motion estimation algorithm works well for the opaque macroblocks. This is not the case for boundary macroblocks which contain a large number of local minima on their error surfaces. In view of this, we propose a fast search algorithm which incorporates the binary alpha-plane to accurately predict the motion vectors of boundary macroblocks. Besides, these accurate motion vectors can be used to develop a novel priority search algorithm which is an efficient search strategy for the remaining opaque macroblocks. Experimental results show that our approach requires simple computational complexity, and it gives a significant improvement in accuracy on motion-compensated video object planes as compared with conventional algorithms, such as the diamond search. Numerically, a speed-up of about 27 times as compared with the full search algorithm is obtained in our tested VOs.

Although many fast search algorithms can achieve low computational load and acceptable encoding quality requirement, it is always desirable to look for identical searching results as compared with that of the conventional full search algorithm. For instance, high quality digital video product and object tracking applications need to estimate motion activities accurately. To develop a fast full search algorithm, we have made use of our observation that pixel matching errors with similar magnitudes tend to appear in clusters for natural video sequences on average. Subsequently an adaptive

partial distortion search algorithm has been proposed. The algorithm significantly improves the computation efficiency of the original partial distortion search. In terms of the number of operations, our experimental results show that the computational efficiency of the algorithm outperforms other algorithms. The algorithm can have a speed-up of 3 to 9 as compared with the Full Search Algorithm (FSA). In terms of real-time measurement, our algorithm can speed up the search for about 3.38 times as compared to the FSA on average, which is again better than other tested algorithms including Successive Elimination Algorithm for encoding sequences with high motion activities and arbitrarily shaped video objects.

Discrete Cosine Transform (DCT) is widely used in modern video compression standards including the ISO MPEG-4, to achieve high compression efficiency. The DCT domain scheme works very well for intraframe coding. On the other hand, block-based compensation typically results in a peaky distribution of errors. It leads to a scattering of DCT coefficients and makes the DCT coding inefficient. This disadvantage motivates us to study the spatial distribution of prediction errors resulting from either the full-search motion estimation or other fast search algorithms. As a result, we propose a Mixed Spatial-DCT-based Coding Scheme to code the prediction errors. The scheme divides prediction errors in a block into two components. One component is coded in the spatial domain with the arithmetic coding technique while the other is coded with the traditional DCT method. The coding scheme can improve the rate-distortion performance of the traditional DCT-based coding for high quality video applications. The proposed scheme is especially suitable for arbitrary shaped video objects and, video sequences which contain moderate to high motion activities.

In order to find a possible optimal coding system, a signal-source model has been used, which hopefully can be sufficiently accurate enough to reflect the characteristics of practical signals. The first-order Markov process has been found to be a successful

model for intraframe coding. However, for motion-compensated error signals, the situation is very different. It has been observed that the motion compensation prediction (MCP) errors are space-dependent and the assumption of wide-sense stationary (WSS) is not valid. As a result, it is inaccurate to employ a simple Markov model for the MCP errors. Hence, we have studied a covariance model analytically from the first order Markov model by making use of the space-dependent characteristics. Consequently, we derive an approximated and separable autocorrelation model for the block based motion compensation difference signal. Experimental results show that the proposed model reflects the autocorrelation characteristics of practical prediction errors accurately. Furthermore, this model can provide some very useful insights for an analytical design of the coding system and make possible the investigation of various video signal decomposition algorithms. This is a fruitful direction of further research.

List of Publications

A. Journal Papers

- Accepted or Published papers

1. Ko-Cheung Hui, Wan-Chi Siu and Yui-Lam Chan, "Fast Motion Estimation of Arbitrarily Shaped Video Objects in MPEG-4", *Signal Processing: Image Communication* 18 (2003), p.33-50, Elsevier Science, The Netherlands.
2. Ko-Cheung Hui, Wan-Chi Siu and Yui-Lam Chan, "New Adaptive Partial Distortion Search Using Clustered Pixel Matching Error Charateristic", paper accepted, to be published in *IEEE Transactions on Image Processing*, USA.
3. Yui-Lam Chan, Ko-Cheung Hui and Wan-Chi Siu, "Adaptive Partial Distortion Search Algorithm for Block Motion Estimation," paper accepted, to be published in *Journal of Visual Communication and Image Representation*, USA.

- Submitted paper

4. Ko-Cheung Hui and Wan-Chi Siu, "New Mixed Spatial-DCT-based Coding of the Motion Prediction Error Frame of Video Objects and Frames", *IEE Proceeding Vision, Image and Signal Processing*.

- paper in preparation

5. Ko-Cheung Hui and Wan-Chi Siu, "Extended Analysis on Frame Difference Errors of the Block Based Motion Estimation for Video Coding", paper in preparation, and to be submitted to the *IEEE Transactions on Signal Processing as* a Transaction Brief, USA.

B. International Conference Papers

6. Yui-Lam Chan, Ko-Chenug Hui and Wan-Chi Siu, "Fast Search Algorithm for Edge-Oriented Block Matching Algorithm", Proceedings, *2001 International Symposium on Intelligent Multimedia, Video & Speech Processing (ISIMP)*, pp.225-228, May 2001, Hong Kong.
7. Ko-Cheung Hui, Yui-Lam Chan and Wan-Chi Siu, "Priority Search Technique For Mpeg-4 Motion Estimation of Arbitrarily Shaped Video Object", Proceedings, *2001 International Conference on Image Processing* (ICIP2001), pp.644-647, Vol. III, October 2001, Thessaloniki, Greece.
8. Yui-Lam Chan, Wan-Chi Siu and Ko-Cheung Hui, "Block Motion Estimation Using Adaptive Partial Distortion Search", Proceedings of the *2002 IEEE International Conference on Multimedia and Expo* (ICME 2002), pp. 477-480, Vol. I, August, 2002, Lausanne, Switzerland.
9. Ko-Cheung Hui, Wan-Chi Siu and Yui-Lam Chan, "New Adaptive Partial Distortion Search Using Clustered Pixel Matching Error Charateristic", Proceedings, *2003 International Symposium on Circuits and Systems* (ISCAS '03), pp.97-100, Vol. II, May 2004, Vancouver Canada.
10. Kai-Yin Wong, Wan-Chi Siu and Ko-Cheung Hui, "Fast Motion Estimation for Wavelet-Based Video Coding", Proceedings, *2004 International Symposium on Intelligent Multimedia, Video & Speech Processing (ISIMP)*, pp. 20 – 22, October 2004, Hong Kong.

11. Ko-Cheung Hui and Wan-Chi Siu, “New Pixel-DCT Domain coding Technique For Object Based and Frame Based Prediction Error”, *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, March 2005, Philadelphia, PA, USA.

Acknowledgments

I would like to express sincere gratitude to various peoples from the Hong Kong Polytechnic University, where I have the opportunity to work with. My major appreciation is to my supervisor Prof. W.C. Siu. Not only his professional advices concerning my study, his view on the way of life, hard working style and willingness to devote to the advancement of science and research, are a precious knowledge to me. This study could never reach its present level without his constant and kindly support.

Special thanks are given to Dr. Y.L. Chan for his invaluable advice in the field of motion estimation and those who have worked with me in the first one year. The countless discussions with him have been proved to be fruitful and inspiring. I am greatly indebted to my colleagues, Dr. Bonnie Law, Mr. W.H. Wong, Mr. W.L. Hui, Mr. H.K. Cheung, Mr. K.T. Fung and Mr. K.W. Wong. Their expert knowledge and friendly encouragements have helped me overcome many difficulties in my study. It has been a wonderful time to me in these four years to work with them.

I have also thanked all members of staff of the department and the clerical staff in the general office. They provide a creative environment for me to work in.

Above all, I am deeply grateful to my family and friends for their encouragement and support. Without their understanding and patience, it is impossible for me to complete this research study.

Statements of Originality

The following contributions reported in this thesis are claimed to be original.

1. A new search technique is introduced for the motion estimation of boundary macroblocks (MBs) of an MPEG-4 arbitrarily shaped video object (VO). We incorporate the shape information provided in a binary alpha-plane of the VO to predict accurately the motion vectors of boundary MBs. Furthermore, these motion vectors can be used to assist the estimation of the remaining opaque MBs of the VO. More details can be found in section 3.3.

2. A new priority search algorithm (PSA) is introduced for the motion estimation of a VO. In this algorithm, we recognize that motion activities in opaque MBs are highly correlated with the neighboring boundary MBs. Hence, we perform motion estimation on all boundary MBs first in contrast to the conventional raster-scanning approach. Experimental results show that this strategy works well if the motion vectors in the boundary MBs truly represent the moving video object. After all motion vectors of the boundary MBs are found, we compute a motion vector for each opaque MB by taking the best-matched one among all of its neighboring motion vectors and the zero motion vector (0,0) as the initial centre. A conventional fast block matching algorithm is then employed. On average, the proposed algorithm can speed up the motion estimation process by about 23 times in terms of the total number of operations when compared to the Full Search Algorithm (FSA). More details can be found in section 3.4 and 3.5.

3. We have found that on average, pixel matching errors with similar magnitudes tend to appear in clusters for natural video sequences during the motion estimation. It is different from the past study in the literature, in which most, if not all, researchers made an assumption that pixels with larger gradient magnitudes have larger matching errors on average. More details can be found in section 4.2.

4. We have made use of the observed clustering characteristic to introduce an adaptive Partial Distortion Search (PDS) for the motion estimation of both boundary MBs and opaque MBs in a VO. The clustering characteristic leads us to construct an adaptive index set, and thus a pixel with greater matching error can be firstly computed, and this error is accumulated to the Sum of Absolute Difference (SAD) earlier than other pixels. As a result, the SAD calculation can be terminated sooner. We have analytically found that both mean of pixel values in a target MB and mean of pixel values in a candidate MB are good references to predict the magnitude of each pixel matching error in the target MB. In the study, we have proved that the mean of pixel values in the initial candidate MB at the centre of a search window is the best candidate to explore the clustering characteristic. Hence, it is used to calculate a reference value and to construct the adaptive index set. Our experimental results show that the proposed adaptive PDS can have a speed-up of 3 to 9 as compared with the FSA. More details can be found in section 4.3 and 4.4.

5. Another row-based adaptive PDS is developed in order to remedy the non-uniform memory access problem. We have modified the original adaptive PDS algorithm into a row-based algorithm, in which a row of 4 consecutive pixels with

larger prediction errors is accumulated to a partial SAD sooner than other rows. Experimental results show that the row-based adaptive PDS outperforms all other tested algorithms for encoding sequences with high motion activities and arbitrarily shaped video objects. More details can be found in section 4.3 and 4.4.

6. A New Coding scheme for coding Motion Prediction Error Frames of Video Objects and Frames is proposed. In this algorithm, we make use of the phenomenon that pixel matching errors in some MBs tend to appear together in a cluster form. The algorithm divides a prediction error MB into two components. Each component is characterized by its own spatial correlation. One component is then coded by using the context-based arithmetic encoding (CAE) and variable length coding techniques (VLC), and the second component is coded by using the traditional DCT-based method. Our experimental results show that the algorithm successfully improves the coding efficiency of the traditional DCT-based coding for MBs with clustered prediction errors. More details can be found in section 5.2 and 5.3.

7. An approximated separable autocorrelation model for the block based motion compensation frame difference (MCFD) signal has also been derived. In this study, we have made use of the first order Markov model to derive the approximated autocorrelation model. The major assumption we made in the derivation is that a net deformation of pixels is directional in general rather than a uniform error distribution in a block. Simulation results show that, the derived model can describe the statistical characteristics of the MCFD signals accurately. The model proves that the concern of imperfect block-based motion

compensation is an important step to study the motion-compensated coder; and thus the autocorrelation function of the MCFD signals can be expressed correctly.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

With the development of communication and information processing technologies, the demand for efficient transmission and storage of a video signal for many applications and services has increased rapidly. From simple calculation, we know that a large amount of bits is required for video signal. For example suppose the a video sequence is digitized as discrete arrays with 640 pixels per raster line and 480 lines per frame, which is a resolution very typical in real situation. Assuming each pixel consists of three color components (i.e. the three primaries, red, green, and blue) for each frame. And each color component for a pixel is sampled with 8-bit precision. The storage capacity of each frame requires approximately 340KBytes. If this video sequence is transmitted at 24 frames/second without compression, the raw data rate for the video signal is about 170 Mbit/s. Hence, the need to find an efficient compression and coding technique urge on the people to develop some video compression standards.

1.1 *History and Development Video Coding Standards*

1.1.1 H.261

Digital compression standards for video conferencing were developed in the 1980s by the International Telegraph and Telephone Consultative Committee (CCITT), which is now known as the ITU Telecommunications Standardization Sector (ITU-T). The ITU-T is a permanent organ of the International Telecommunications Union (ITU) and is responsible for studying technical and operating questions and issuing Recommendations with a view to standardizing worldwide telecommunications. The ITU-T was formed from the CCITT as part of the ITU reorganization in 1993. In early

1991, the CCITT finalized a set of coding standards known as H.320 or sometimes P×64 to indicate that it operates at multiple of 64 kbits/s. The video coding part is called H.261 [1]. This standard targeted to code pictures at a Common Intermediate Format (CIF) 352×288 pixels. A lower resolution called Quarter CIF (QCIF) with picture size 176×144 pixels is supported for sending videotelephone or videoconference pictures on integrated services digital network (ISDN) facilities.

It is the first widespread practical successful video standard. It was also the first standard using the basic video coding structure that many current standards still keep using. For instance, H.261 has already utilized 16×16 MB for motion compensation. Moreover, 8×8 block-based DCT, scalar quantization and two-dimensional run-level variable length entropy coding have been employed in H.261.

1.1.2 MPEG-1

In 1988, a working group in charge of the development of standards for coded representation of digital audio and video, was established. The group is named as Moving Picture Experts Group (MPEG). The MPEG working group is formed under the auspices of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). In this year, the working group started the moving picture standardization process with a strong emphasis on real-time decoding of compressed data stored on digital storage devices such as CD-ROMs. Their first standardization product is the ISO/IEC 11172. This international standard is the MPEG-1 and its second part, ISO/IEC 11172-2 [2], is the well-known MPEG-1 video. The ISO/IEC 11172 is officially entitled “Information technology – Coding of moving pictures and associated audio for digital storage media up to 1.5Mbit/s”. As indicated by the title, MPEG-1 is concerned with coding of digital audio and digital video. And because of the primarily target bit rates is set around 1.5 Mbit/s, it is particular suitable

for storage media applications such as CD-ROM retrieval. Consequently, it is a widely successfully video codec capable of approximately VHS videotape quality are better at about 1.5 Mbit/s and covering a bit rate range of about 1-2 Mbit/s. The MPEG-1 video (ISO 11172-2) was finally approved in 1993.

In terms of technical features, it added bi-directionally predicted frames, half-pixel motion compensation and including the H.261 compression techniques. In fact, half-pixel motion had been proposed during the development of H.261, but it was apparently thought to be too complex at the time. When comparing to the H.261, MPEG-1 provides superior quality than H.261 when operated at higher bit rates. On the other hand, H.261 performs better at bitrates below 1 Mbit/s, as MPEG-1 was not designed to be capable of operation in this range.

1.1.3 MPEG-2

In the moment of the technical development of MPEG-1 was nearly complete, the MPEG working group started another new project to target higher bits rates and better quality for applications such as broadcast TV in 1992. That is the MPEG-2 and formally referred to as ISO/IEC 13818. MPEG-2 is aimed at more diverse applications such as television broadcasting, digital high-definition TV (HDTV), and communication. Some of the possible applications listed in the MPEG-2 video standard (ISO/IEC 13818-2 / ITU-T H.262) [3] are: broadcast satellite service (BSS) to the home; cable TV (CATV) distribution on optical and copper wire networks; interactive storage media (ISM) such as optical disks; interpersonal communications (IPC) such as videoconferencing and videophone; etc. MPEG-2 offers little benefit over MPEG-1 for programming material that was initially recorded on film. One difference between MPEG-2 and MPEG-1 is the MPEG-2 provides a more efficient method to code interlaced video signals. MPEG-2 video syntax was frozen in April 1993. Two years later, in 1995, the three primary

documents (systems, video, audio) that comprise the MPEG-2 standard finally reached international standard status.

MPEG-1 video was intended only for progressive video pictures but not interlaced format. MPEG-2 is not only limited to focus on interlaced video coding. It was designed to encompass MPEG-1 and to also provide high quality at much higher bit rates with interlaced video sources, such as the initially recorded programming material on film. Although usually thought of as an ISO/IEC standard, MPEG-2 video was developed as an official joint project of both the ISO/IEC and ITU-T organizations. Its primary new technical features are efficient handling of interlaced-scan pictures. It adds the support of hierarchical bit-usage scalability, such as spatial scalability, temporal scalability, SNR scalability and data partitioning. Its target bit-range was approximately 4-30 Mbit/s.

1.1.4 H.263 and H.263+

H.263 (version 1) [4] is the first codec designed specially to handle very low-bit-rate video. It consists of different video compress techniques that are more advance when comparing to the MPEG-2 video. H.263 is the most dominant standards for practical video telecommunication nowadays. The ITU-T is responsible for the standardization process of H.263 (version 1) and the H.263 was approved in early 1996 (with technical content completed in 1995). The original target bit-range of H.263 was about 10-30 Kbit/s, but this was broadened during development to perhaps at least 10-2048 Kbit/s as it become apparent that it could be superior to H.261 at any bitrate.

The significant coding improvement by H.263 is due to several new technical features. They were 8×8 block-size motion compensation, overlapped block motion compensation, allowing motion vectors pointing outside of a frame (unrestricted motion compensation), three-dimensional run-level-last variable-length coding, median motion

vector prediction for differential coding, and more efficient header information signaling. Comparing to H.261, H.263 also includes arithmetic coding, half-pixel motion compensation, and bi-directional prediction. Note that these techniques were previously employed in different international standard but not firstly find in H.263. At very low bitrate, such as below 30 Kbit/s, H.263 can code with the same quality as H.261 using half or less than half the bitrate. At bit rates above 80 Kbit/s, it can provide a more moderate degree of performance superiority over H.261.

H.263 (the second version) [4] is officially known as H.263+. The H.263+ project added a number of new optional features to H.263. It extends the effective bitrate range of H.263 to essentially any bitrate and support both progressive and interlaced picture formats and any frame rates. H.263+ is targeted to outperform any existing standards over this entire range. In order to support telecommunication in error prone environments, the H.263+ is the first video coding standard that offers a high degree of error resilience for wireless or packet-based transport networks. H.263+ also added a number of improvements in compression efficiency, which is custom and flexible video formats and scalability supporting. The ITU-T approved H.263+ in early of 1998 (with technical content completed in September 1997).

1.1.5 MPEG-4

After the successful development and achievement of the MPEG-1 and MPEG-2, the MPEG working group initiated the standardization phase of MPEG-4 in 1994. MPEG-4, with formal as ISO/IEC designation “ISO/IEC 14496” was finalized in October 1998. In early 1999, MPEG-4 became an international standard. The fully backward compatible extensions under the title of MPEG-4 Version 2 were frozen at the end of 1999, to acquire the formal International Standard Status early in 2000.

Originally, the objective of the MPEG-4 was to develop advanced coding for very low bit-rates (below 64 kbit/s) applications. However, In July 1994 its target was expanded to coding of a scene as a collection of individual Audio-Visual-Object (AVO) and to provide a set of tools for these AVOs, so as to fulfill the requirements of future interactive multimedia applications and services. In order to provide the solutions for these objectives, a set of “tools” and “algorithms” for audio-visual data have been being developed in the MPEG-4 project. Consequently, it will provide significantly superior compression performance and new object-oriented capabilities for artificially generated scene situations. A user can access arbitrarily shaped objects in the scene and manipulate these AVOs in client side terminal. In the future, the MPEG-4 standard should provide the features including universal accessibility and robustness in extremely error prone environments; new interactive functionalities for users when presenting audio-visual material; hybrid coding of natural and synthetic AVOs; increased compression efficiency compared to previously standardized methods; the possibility of “downloading” decoder tools; integration of real-time applications and non-real-time (stored) applications.

MPEG-4 visual which is officially named as ISO/IEC 14496-2 [5] will includes most technical features of the prior video and still-picture coding standards and will also include a number of new coding features such as, shape coding of segmented objects with context-based arithmetic coding, shape-adaptive DCT and padding techniques for arbitrarily shaped texture coding. Hence, it can achieve efficient coding of hybrids of synthetic and natural video-content. In addition, it uses wavelet coding of still pictures, global motion compensation and sprite generation to increase compression efficiency. Reversible variable length coding is applied for robust video coding in error prone environments. It also extends half-pixel motion compensation to quarter-pixel accuracy by using wiener filter for interpolation. MPEG-4 aims to cover essentially all bitrate

ranges, picture formats and frame rates, including both interlaced and progressive-scan video pictures. In generally speaking, it employs similar techniques of H.263 for predictive coding of normal camera-view video content for non-interlaced video and utilizes MPEG-2 approaches for interlaced sources.

1.1.6 H.264

In early 1998, a “long-term” effort proposed by the Video Coding Experts Group (VCEG) to develop a new standard for low bitrate visual communications. The proposal led to the draft “H.26L” standard, which targeted to double the coding efficiency in comparison to any other existing video coding standards for a broad variety of applications. The VCEG working group adopted a first draft design for that new standard in October 1999. In December 2001, VCEG and the Moving Picture Experts Group (MPEG) formed a Joint Video Team (JVT) to finalize the draft of this new video coding standard. In March 2003, it was approved by ITU-T as Recommendation H.264 and by ISO/IEC as International Standard 14496-10 (MPEG-4 part 10) Advanced Video Coding (AVC) [6]. It is often abbreviated as H.264/AVC.

In terms of technical features, it employs, generalizes and improves the prediction techniques in previous standards. It supports flexible selection of block sizes down to 4×4 pixels for motion compensation. Similar to the MPEG-4 visual, it allows motion vector with quarter-pixel accuracy, but reduces the complexity of interpolation filter from 8 taps to 6 taps. Furthermore, it enable efficient coding by allowing selection of reference frames among a larger number of pictures that have been decoded and stored in the decoder for motion compensation purposes. Improved skip-mode and direct-mode in prior standards are also included, which obtain better coding performance for video with global motion contents and bidirectionally predictive frames respectively. Some previous standards allow prediction coding for intra-coded blocks. H.264/AVC,

however, carries out the prediction in spatial domain. It also makes use of loop-deblocking filter to improve both objective and subjective quality for motion compensation. In addition to the above progresses, some significant improvements of the DCT transform are found in this new standard, which includes 4×4 block size transform and adaptive block-size transform. Because of the improved transforms, H.264/AVC is the first standard to realize exact-match of inverse transform in all decoders. Furthermore, two advanced entropy coding methods using arithmetic coding are included in H.264/AVC [7].

1.2 *Literature Review*

The developments of these video compression standards have spanned more than two decades and built upon a large number of experimental works and in depth theoretical studies. The huge successes were results of dedication by hundreds of researchers and engineers from all over the world.

Motion compensation is one key technique to attain the effectiveness of most video coding standards. Different motion representations have been widely investigated in the research. Jain and Jain [8] studied interframe Hybrid Coding involving block-based motion compensation, in which block size is fixed. The major advantages of the block-based motion model are its effective prediction for translational motion and lesser hardware complexity. Fixed block size model suffers the problem of inaccurate matching for complex motions. The natural variation is variable block size techniques [9-11], such that blocks with smaller size will compensate complex motions. Peter Strobach [9], Li, Lin and Wu [10], and Sullivan and Baker [11] used quadtree structure to represent the variable block size model for compression improvement. Instead of using rectangular block-based representation, Mahmoud and Bayoumi [12] suggested to partition a frame into equilateral triangle blocks. The division is represented by a quadtree structure. A scene often contains foreground objects with complex action and a

relative static background. Supporting interactive application in MPEG-4 is due to its exploitation of object-oriented model. Musmann, Hötter and Ostermann [13] proposed a description of objects by means of motion, shape and color of the objects. The Object-oriented approaches for motion compensation can be classified into three main categories. The first category describes the objects in terms of segmented region similar to the method proposed by Yokoyama, Miyamoto and Ohta [14]. The second category applies the parametric model for object motion description. For instance, both [15,16,17] use spatial transform to represent spatial distortion between two successive frames. Nakaya and Harashima [15] studied the use of affine transform, bilinear transform and perspective transform involving triangular grids motion compensation. The corresponding performance is evaluated theoretically and experimentally. Ghanbari, Faria, Goh and Tan, [16] investigated the compensation of block-based distortion by using bilinear transform and bilinear interpolation. Tekalp, Altunbasak, and Bozdagi [17] showed that a triangular mesh model with affine or perspective transform can capture almost all capabilities of 3-dimensional (3-D) object-based model. The major drawback of this category is the problem in mismatching of the model with complex motion. The third category [18,19] makes use of dense motion field to overcome this difficulty. A dense motion field employs at least one motion vector per pixel to provide improved motion compensation such as the works of Stiller [18] and Han and Podilchuk [19].

A proper theoretical treatment of motion-compensated video coding is valuable for the design of state-of-the-art video codecs, even though it requires many assumptions and simplifications for the analysis of a complicated system processing real-world signals. In 1987, Girod [20] presented the first comprehensive rate-distortion analysis of motion-compensated prediction (MCP). This theoretical framework leads motion-compensated video coding away from heuristics and toward an engineering science. Girod [21] presented a theoretical analysis of multi-hypothesis motion-

compensated prediction for hybrid video coders. When more than one prediction values are weighted to generate a pixel value, the approach is generally defined as multi-hypothesis technique. The bidirectionally prediction frame is a kind of the multi-hypothesis prediction. MPEG-4 employs Wiener filter interpolation to extend the half-pixel motion compensation to quarter-pixel accuracy [22]. Quantitative analysis about the dedication of factional-pixel accuracy for coding performance was given by Girod in [23]. He studied the effect of fractional-pixel accuracy on the efficiency of motion-compensation by using various spatial predictions and interpolation filters and found that quarter-pixel accuracy is suitable enough for typical broadcast TV signals. Overlapped block motion compensation (OBMC) is another kind of multi-hypothesis technique employed in the modern video standards. The propose of OBMC proposed by Nogaki and Ohta [24] is originally to remove blocking artifacts of motion prediction error. Orchard and Sullivan [25] presented an approach that combines an optimized overlapping window design technique with optimized motion estimation. They also studied the influence of two shapes of the window support to their motion estimation, which involved a diamond shaped support and a square of 32×32 pixel support. Tao and Orchard [26] statistically modeled the motion field, the field of motion estimation and their relationship for formulating a parametric solution of an optimal OBMC window. In 2002, Zheng et al. [27] investigated thoroughly the theoretical aspects of OBMC by applying a statistical motion distribution model. Moreover, he used the statistical model to interpret the space-dependent characteristics of motion-compensated frame differences.

MPEG-4 introduces the concept of Video Objects (VOs) to support access of individual semantic objects in visual contents. A temporal instance of a VO is represented by its texture value and shape information [28]. In natural scenes, VOPs are obtained by semi-automatic or automatic segmentation. The MPEG-4 visual has

suggested a framework of segmentation for VOP generation. The framework aims at an appropriate combination of temporal and spatial segmentation strategies. Two kinds of algorithm for temporal segmentation are used in the framework. The first one is temporal segmentation based on change detection, which is proposed by Mech and Wollborn [29]. For this algorithm, Aach, Kaup and Mester [30] smooth the boundaries of changed image areas by a relaxation technique using local adaptive thresholds. The second temporal segmentation algorithm is based on higher order moments and motion tracking. Neri, Colonnese and Russo [31] produce a segmentation map of each frame of a sequence by processing a group of frames, which involves higher order Statistics. The framework uses watershed algorithm for the spatial segmentation. Salembier and Pardàs, [32] had simplified the images for easier processing, before calculating the spatial gradient of an image. Both of the image simplification and spatial gradient calculation make use of morphological operators. The spatial gradients were used by Vincent and Soille [33] as an input of a watershed algorithm to partition an image into homogeneous intensity regions. The European-Algorithmic Group COST211 proposes another platform for video object segmentation. The Group COST211 is a forum and research network on video analysis. During their 5th framework, this forum has focused on video segmentation based on a test model, called COST 211 Analysis Model (AM) [34]. The COST 211 meeting in October 1996 witnessed the definition of the 1st AM, which consists of a full description of tools and algorithms for automatic and semiautomatic image sequence segmentation, object detection, extraction and tracking. In image segmentation problem, Active contour models, or snakes proposed by Kass, Witkin, and Terzopoulos [35], have also been extensively studied and applied for video segmentation in the past decade. Gastaud and Barlaud, [36] proposed a video segmentation using active contours on a group of frames, which was robust to light variations, noise and camera motion, etc. Sun, Haynor and Kim, [37] recommended a

new class of active contour approaches and named it as Vsnakes. The VSnakes algorithm defines a differential contour energy, which reflects the difference between successive contours.

MPEG-4 needs to code shape information of a segmented object for purposes of interactive applications. Within MPEG-4, two different categories of shape coding algorithms have been evaluated. The first one is contour-base shape coding [38-43]. Yamaguchi, Ida and Watanabe, [38] modified the normal MMR for arbitrarily shaped coding. The full name of MMR is “Modified Modified Read”, which is the standard method for the G4 facsimile compression. Lee et al. [39] described the contour of a binary shape by tracing a 1-dimensional baseline and turning points. Ma, Chen and Cheng proposed a selection scheme [40], which could reduce the turning point of the boundary for lossless coding. Hwang, Wang and Wang [41] made use of a differential chain coding technique for shape coding. While Zaletelj and Tasič, [42] approximated an object shape with cubic B-splines technique. The second category is bitmap-based approach [44-48], such as the Context-based Arithmetic Encoding (CAE) that adopted by MPEG-4. Bossen, and Ebrahimi, and Brady, Bossen, and Murphy [44, 45] proposed a shape coding technique based on the JBIG algorithm in both lossless and lossy modes. The Joint Bi-level Image experts Group (JBIG) is a group of experts who work for standardization of bi-level image coding. Instead of using CAE technique, Chen, Hsieh and Wang [47] studied the utilization of quadtree-based decomposition to obtain a shape representation at different resolution levels. Furthermore, Liu, Shieh and Lee, [48] presented their study about the efficiency of hardware architecture for binary shape coding. In the reference [46, 49], Ostermann and, Katsaggelos, Kondi, Meier and Schuster, introduced the above techniques developed for shape coding within MPEG-4 standardization effort and compared their performance in terms of rate-distortion, computational complexity and scalability. Note that, the CAE can also involve in rate-

control algorithm by using lossy coding and efficient entropy coding. Marpe, Blättermann, Heising, and Wiegand [50] developed a context model for efficient prediction of the coding symbols, in which this approach has been integrated into the ITU-T H.26L test model. On the other hand, Lee, Cho, and Eleftheriadis [51] formulated a buffer-constrained adaptive quantization problem for shape coding, and then proposed an algorithm for the optimal solution under buffer constraints.

MPEG-4 extends the conventional block-based techniques for object-based coding. Especially for boundary regions (boundary macroblock) of an arbitrarily shaped object, a wide variety of techniques have been studied to improve these conventional techniques. Chen, Gu and Lee [52] proposed two padding technique, which is the repetitive and morphological padding, for motion compensation of the object's boundary macroblock (MB). Edirisnghe, Jiang and Grecos [53] make use of variation trend of boundary pixels to develop another padding technique for motion compensation. Chen and Liu, [54] suggested motion estimation of the boundary MB in transform domain, such that padding processes will be prevented. In addition, padding of the boundary MB is also necessary for intra-mode coding. Kaup [55, 56] employed a low-pass padding technique to handle the discontinuity problem at an object boundary. Hence, traditional block-based DCT can be utilized directly. Shen, Zeng, and Liou [57] developed a new padding technique that guarantees the number of nonzero transformed coefficient after traditional DCT be equal to the number of opaque pixels in a boundary MB. Other researchers focused on designs of non-block-based transforms. In 1989, Gilge, Engelhardt and Mehlan, [58] used an orthogonalization schemes to obtain a set of basis functions which is orthogonal with respect to the shape of the segmented object, thus shape adaptive transform can be realized. Afterward, a lot of researchers [59-63] studied and improved a shape-adaptive DCT algorithm (SA-DCT). Sikora and Makai [59] was the group who developed the SA-DCT, in which the SA-DCT involves a

predefined orthogonal sets of DCT basis function. Shishikui and Sakaida [64] proposed a Region Support DCT (RS-DCT). The major features of the transform are that it is designed based on conventional 2-DCT but the basis is not orthogonal, so that an iteration processes was suggested. Another special algorithm used to enhance the traditional DCT for boundary MB was investigated by Moon, Kweon and Kim [65]. This algorithm merges two boundary *blocks* together before the block-based DCT, which is called Boundary Block-Merging technique (BBM).

The simplest block matching motion estimation algorithm is the full search algorithm (FSA). The FSA exhaustively compares a matching criterion [66-72] between the target MB and every candidate MOBs in a searching window, W. Hence, it can give the optimum solution. However, heavy computational load is its major disadvantage. Consequently, it attracts a lot of researchers to investigate different fast approaches. These fast search algorithms can be classified into six categories. 1) The fast search algorithms in this category seek for a way to select a subset of the candidate MB in W to reduce the computational time [8, 69, 73-86]. A lot of famous algorithms belong to this group, such as the 2D-Logarithmic Search by Jain and Jain [8], Three-Step Search (TTS) by Koga et al. [73], Genetic Search Algorithm by Chow and Liou [77], Diamond Search by Tham, Ranganath, Ranganath and Kassim [81], Four-Step Search by Po and Ma [82], and Motion Vector Field Adaptive Fast Motion Estimation by Hosur and Ma, [85], etc. 2) The algorithms [73, 89-98] in the second category use a reduced complexity distortion measure to save computation. For instance, Koga et al. [73] and Chan and Siu [90] used pixel decimation for speed up purpose; the partial distortion search adopted in H.263 test model [92] and adaptive PDS by Kim and Choi, [96]; Winner-Update Strategy by Chen, Hung and Fuh [98] are all fall in this category. 3). The third category [99-103] make use of mathematical inequalities to reduce the computational load. The most representative one may be the Successive Elimination Algorithm (SEA) suggested

by Li and Salari [99]. 4) Chan and Siu [104, 105], and Tao and Orchard [106] studied the employing of different image features to reduce computational burden in motion estimation, and the edges are the feature that they have investigated. This feature based techniques are the fourth category. 5) A performance analysis of MPEG-4 codec [107] showed that motion estimation will remain as a computationally intensive step in object-based coding. The fifth category focused on developing object based motion estimation, such as the study by Panusopane and Chen, [108]. 6) The emerging of H.264/AVC also directs the study of motion estimation into another direction, such as Wiegand, Zhang, Girod, Lincoln and Steinbach [109-111] investigated long-term memory motion-Compensated prediction in depth.

Transform coding scheme has proven to be an effective technique for image compression and video coding. In terms of compression efficiency, the objective of a transform is to decorrelate the original signal, and this decorrelation generally results in the signal energy being redistributed among only a small set of transform coefficients. The most efficient transform that can maximize energy packing capability is the Karhunen-Loève Transform (KLT) [112]. Unfortunately, the KLT basis functions are source-dependent. The Discrete Cosine Transform (DCT) is the one widely used in the modern video standards, because it is shown that for the case of the first-order Markov source, the DCT is asymptotically equivalent to KLT as the adjacent element correlation coefficient tends to unity [112, 113]. However, after the motion compensation, Kaneko, Hatori and Koike [114] claimed that the correlation between adjacent prediction errors are far from unity and in fact ranges from 0.3 to 0.5. To study suitability of the original DCT for the prediction errors, Chen and Pang [115] proposed a compound covariance model for motion-compensated frame differences and demonstrated that DCT performs nearly optimally. In 1999, Niehsen and Brünig, [116] found that means and standard deviations may change significantly from block to block and another compound

covariance model that closely fits the empirical covariance sequence was introduced. By using the this model, they again confirmed that DCT is still suitable for video coding when only a single transform is supported in a codec.

1.3 *Organization of the thesis*

The rest of the thesis is organized as follows. In Chapter 2 we make a brief review on the modern video coding techniques, which including block-based motion compensation, arbitrary shaped object coding, block-based motion estimation and transform coding technique theory.

In chapter 3, we investigate a new priority search algorithm (PSA) for motion estimation of arbitrarily shaped object in MPEG-4 by studying the characteristics of different types of macroblocks in the bounding box of a VOP.

In chapter 4, we explain and illustrate the characteristics of the pixel errors that tend to form clusters. We apply these characteristics to develop a new clustered pixel matching error for adaptive partial distortion search algorithm (CPME-PDS). Moreover, we establish an analysis to determine an adaptive index set required for the CPME-PDS.

In chapter 5, we illustrate and discuss the observation of spatial characteristics of the motion compensated prediction errors. Making use of our observations, we develop a new Mixed Spatial-DCT-based Coding Scheme (MSDCS).

Chapter 6 presents the derivation of a mathematical model for autocorrelation of block-based motion prediction error. We use a variety of simulation results to compare our model with others.

We give the conclusion of our work in Chapter 7, where some suggestions for further development can also be found.

Chapter 2. Technical Review

2.1 *Modern video coding standard review*

The modern video coding standards including H.263 [4], MPEG-1 [2], MPEG-2 [3], and MPEG-4 [5], achieve high compression performance for different applications by exploiting the spatial and temporal redundancies remaining in video sequences. These video standards exploited the redundancies by using several common compression techniques. Figure 2-1 depicts a simplified block diagram of an hybrid video encoder and decoder, which shows some common functional blocks of various video standards.

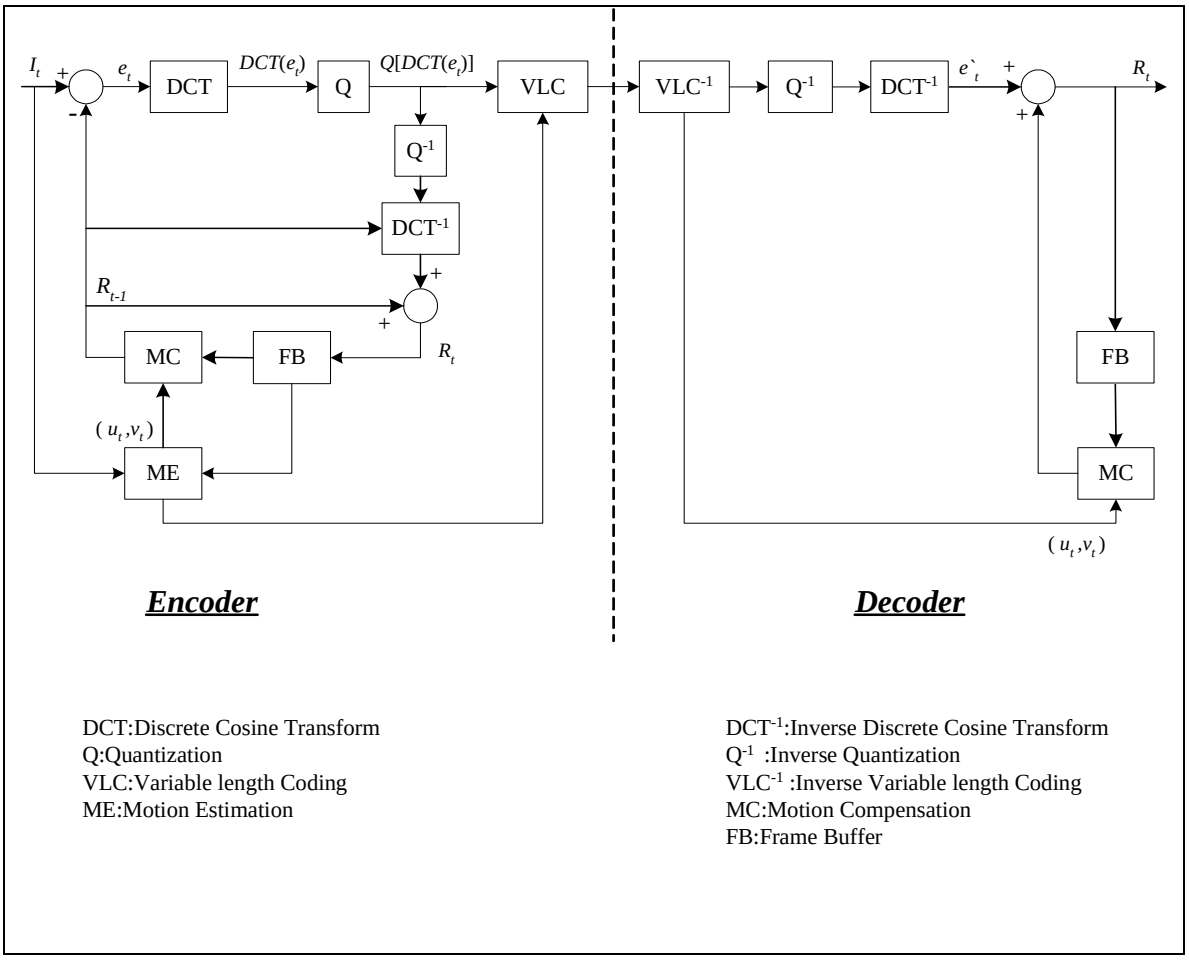


Figure 2-1. Simple block diagram of video encoder and decoder

In the decoder side, the functional block “MC”, for block-based motion compensation, is responsible for the reduction of the temporal redundancies between successive frames. The temporal correlation is utilized by using motion information of a past frame to predict the current frame causally, or predict the current frame from past and future frames by non-causal and interpolative methods. In 70s, some researchers investigated the motion properties inherent in video sequences that can be used to improve the performance of a coding system. After a wide variety of experiments, it has been considered that much of the motion in video sequences is pure translation, i.e. a foreground object moves across a nearly still background in an arbitrary direction, but without rotation, size change or any irregular shape transformation. This property is extremely plausible provided that the motion is not too violent in the short time interval between successive frames. The video standards divided a current frame into a number of non-overlapped blocks with $M \times N$ pixels. A decoder uses one or more than one block/s of pixels in the previous decoded frames to compensate each of these blocks in the current frame. The errors between the current block and the compensated block are then compressed using the discrete cosine transform (DCT) to remove the remaining spatial correlation.

The functional block DCT in Figure 2-1 is responsible for the transform coding of intraframes and interframes. Intraframe compression in these standards makes use of the compression techniques for still images, such as photographs and diagrams to compress individual frames in a video sequence. An intraframe is coded by transform coding without any reference to previously coded frame. On the other hand, an interframe is coded using previously coded frames for motion-compensation prediction. After this prediction process is finished, a video coder compresses the prediction errors using the transform coding. Transform coding is an image conversion process that transforms an image from the spatial domain to the frequency domain. The most

popular transform used in the video standards is the Discrete Cosine Transform (DCT). All traditional DCT-based coding partition a large images into non-overlapping 8×8 square blocks and transform each block separately. The DCT represents any input data as 64 2-dimension (2D) cosine functions with different weighting factors. The resulting weighting factors are represented as a matrix of DCT coefficients. Although the DCT does not in itself result in compression, the transform coefficients tend to be good candidates for compression using run length encoding and predictive coding when it is read inside the system in an appropriate order.

Quantization is the next process after the DCT coding. The DCT coefficients are quantized in an irreversible process that discards the less important information. Although quantization seems to be a simple process, an efficient quantization algorithm involves in-depth studies of distributions of coefficients energy, its relation to that in the data domain, processing of the coefficients in the context of the human visual response and relation to the rate-distortion theory. An appropriate quantization must minimize the distortion of the input data after reconstruction, for a given data rate. Some objective distortion performance measures, such as peak-signal to noise ratio (PSNR) and subjective distortion evaluation, i.e. the perceptibility of the human visual system to the distortion must be considered in the design of a quantization process.

The quantized DCT coefficients and the information for motion compensation are finally entropy coded by a variable-length coding scheme, the Huffman coding. Under the constraints that each source message is mapped to a unique codeword, Huffman coding can provide an optimal statistical coding procedure that approaches the theoretical entropy limit. The variable-length code words used in the standards are derived according to a priori knowledge of the probability of all possible events. Thus, instead of using fixed-length code words for all symbols, relatively short code words are assigned to represent events with the highest probability of occurrence. In the decoder

side, we maintain an identical code book, such that the Huffman coding preserves the coded information.

In the following sub-sections, we will introduce the basic concepts and techniques for the conventional block-based motion compensation. The modern video standards use motion vectors to describe the motion information between successive frames. It is coded differentially in an encoder. The algorithm used to obtain a prediction motion vector is described in Section 2.1.1.1. Section 2.1.1.2 describes the procedure and some studies about the half-pixel interpolation scheme, which is supported in MPEG standards. Both H.263 and MPEG-4 support the Overlapped Block Motion Compensation. The details of its operation are introduced in Section 2.1.1.3. Furthermore, the basic tools of arbitrarily shaped object coding in MPEG-4 will also be presented. These techniques include how to represent a video object planes, content-based arithmetic encoding for shape information coding and padding techniques that extend an arbitrarily shaped object data to rectangular support. Section 2.1.2 describes the details of these techniques.

2.1.1 Block-based motion compensation

We can improve video coding efficiency significantly if the frame-to-frame temporal redundancy is taking into account. A lot of researchers have proposed many different motion representations [8-19] to exploit this temporal information. The block-based motion compensation is the one widely adopted in the modern video standards for compression. The major advantages of the block-based motion compensation are its effective prediction for translational motion, lesser hardware complexity and simple implementation.

The H.263, MPEG-1, MPEG-2, and MPEG-4 support two types of prediction frames, the Predictive Frame (P-Frame) and Bidirectionally Predictive Frame (B-Frame). A P-Frame is coded by a prediction causally from an immediate previous intraframe (I-

Frame) or P-Frame. A B-Frame is non-causally predicted from a previous reference frame or next reference frame or both these frames. A theoretical analysis of multi-hypothesis motion compensation for hybrid video coder [21] explains the role of B-Frame in a video coder. A reference frames must be coded as either an I-Frame or a P-Frame. Note that it is not allowed to code a P-Frame or a B-Frame with reference to any B-Frame.

2.1.1.1 Motion Vectors

To perform the block-based motion compensation, a current frame is divided into a number of non-overlapped $M \times N$ pixels blocks. In the industrial standards, we name the block as a Macroblock (MB) and a *block* with a size of 16×16 pixels, and 8×8 pixels respectively. MPEG-2 and MPEG-4 supports the coding of interlaced frame, and thus an interlaced MB consists of 16×8 pixels.

The industrial standard allows a MB in a P-Frame to be coded with different modes, such as intra-mode (I) and inter-mode (P). The basic idea of block-based compensation is illustrated in Figure 2-2. When a MB in a P-Frame is coded as a P-MB, a forward motion vector is transmitted. The forward motion vector represents a displacement between the MB in the current frame at time *t* and the matched MB in a past reference frame at time *t-m*, *m* is greater than or equal to 1 in general situation. A predictive MB in a B-Frame is named as B-MB. For a B-MB, its forward, or backward, or both of these two motion vectors has to be transmitted. The displacement of the matched block in a reference at time *t+n* is denoted as a backward motion vector. Similar to the forward case, *n* is generally greater than or equal to 1.

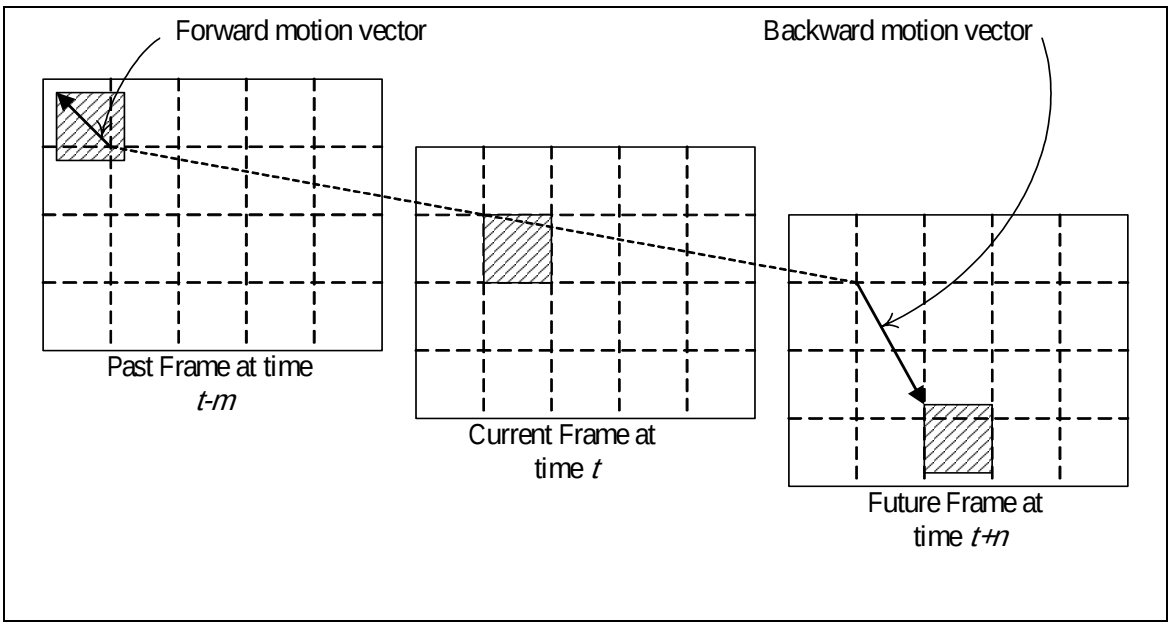


Figure 2-2. Illustration of block-based motion compensation in a P-Frame or a B-Frame

Both H.263 and MPEG-4 do not only support coding of one motion vector per MB, but also include an advanced prediction mode that subdivides a MB into four *blocks* in P-Frame or P-VOP. The codec compensate a MB with four motion vectors for the four *blocks*.

The pixel errors between the current frame and the motion compensated frame are called the prediction errors. For a B-MB, the matched MB from a past frame, or the other matched MB form a future frame, or an average of both can be used as the compensated MB. Consequently, the resulting prediction errors for a P-MB or B-MB involving either a forward or a backward motion vector are expressed as

$$e(x, y; i, j) = f_t(x+i, y+j) - f_{t\pm m,n}(x+i+u, y+j+v) \qquad \textbf{(2-1)}$$

where (x, y) indicates the location of a MB in the current frame, $f_t(\cdot,\cdot)$,

(i, j) is the coordinate of a pixel in the MB, and

(u,v) is a forward or backward motion vector.

When both motion vectors are used, the prediction errors are given by

$$e(x,y;i,j) = f_t(x+i, y+j) - \frac{f_{t-m}(x+i+u_f, y+j+v_f)}{2} - \frac{f_{t+n}(x+i+u_b, y+j+v_b)}{2} \quad \textbf{(2-2)}$$

where (u_f, v_f) is a forward motion vector.

(u_b, v_b) is a backward motion vector.

These motion vectors are differentially coded in the standards [4-5], and thus the motion correlation between adjacent blocks can be utilized for compression. That is using values of previously transmitted motion vectors to obtain a prediction motion vector. The difference between the motion vectors and the corresponding prediction motion vector is encoded and transmitted. We calculate the horizontal and vertical components of a predictor separately. In [4, 5], a median filter is used for three candidate prediction motion vectors to form a predictor. The three candidate predictors, (*MVi*, i = 1, 2, 3), are obtained from the spatial neighborhood macroblocks or blocks that already coded. Figure 2-3 clearly defines the positions of the candidate predictors for each block's motion vector in the advanced prediction mode. In the case of one motion vector per macroblock, the top-left case in Figure 2-3 is applied. The following four decision rules are applied to obtain the value of the three candidate predictors:

1. Let MV_0 be the motion vector found by normal motion estimation. A candidate prediction motion vector is not considered to be valid, if this candidate predictor *MVi* is sited outside of a Frame or a VOP, or in a transparent MB or *block*. Otherwise, such as *MV1*, *MV2* and *MV3* can be used as valid candidate MV for further coding.
2. If one and only one candidate predictor (either *MV1*, *MV2* or *MV3*) is not valid (e.g. it is sited outside the frame, etc), it is set to zero.
3. If two and only two candidate predictors are not valid, they are set equal to the third candidate predictor.
4. If all three candidate predictors are not valid, they are set to zero.

The median value of the three candidates for the same component is computed as predictor, denoted by Px and Py:

$$Px = Median(MV1x, MV2x, MV3x)$$
$$Py = Median(MV1y, MV2y, MV3y) \quad \textbf{(2-3)}$$

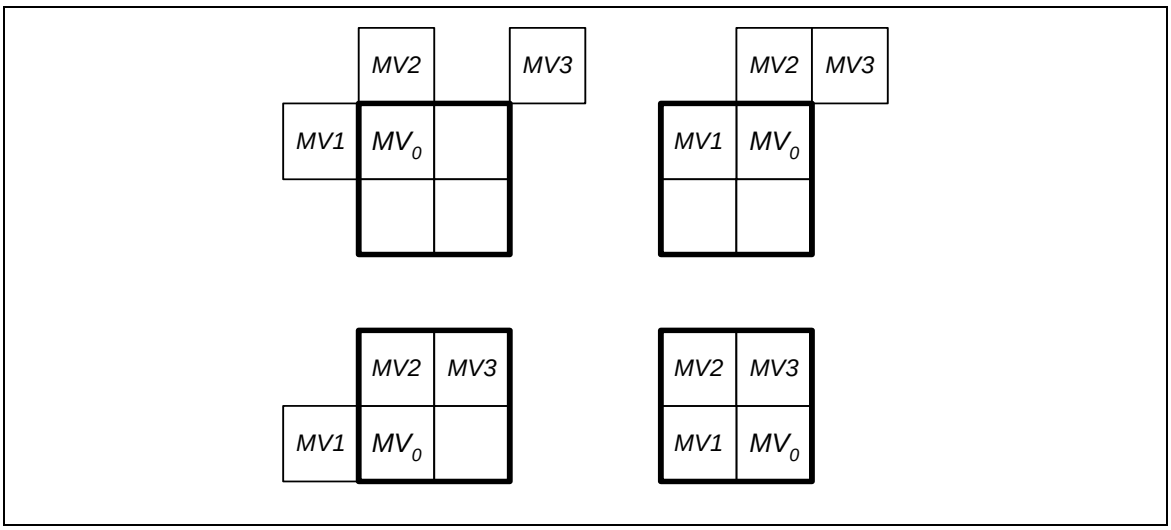
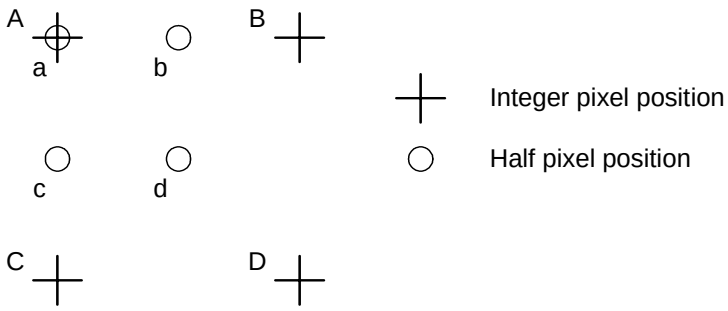


Figure 2-3. Definition of the candidate predictors *MV1, MV2* and *MV3* for each of the luminance blocks in a macroblock.

2.1.1.2 Half-pixel accuracy

In fact, the true physical motion of a moving object between successive frames is not limited to our spatial sampling frequency or the sampling grid. It leads us to expect that we can improve the motion-compensation efficiency with fractional-pixel accuracy.

MPEG standards support motion vector with half-pixel accuracy for 16×16 MB and for 8×8 *block* as well as for 16×8 field block in case of interlaced video. Since decoder prefers a simple interpolation to a complex one, a bilinear interpolation technique is a suitable choice. The interpolation scheme used in the MPEG-4 is shown in Figure 2-4.



a = A,
b = (A + B + 1 - rounding_control) / 2
c = (A + C + 1 - rounding_control) / 2,
d = (A + B + C + D + 2 - rounding_control) / 4
where rounding_control is equal to 1 or 0.

Figure 2-4. Bilinear interpolation scheme.

Motion-compensation with quarter-pixel accuracy is supported in MPEG-4 version 2. The main target of Quarter Pixel Motion Compensation is to enhance the resolution of the motion compensation scheme with only small syntactical and computational overhead, leading to more accurate motion description and less prediction error to be coded [22].

The author of [23] has studied and analysed theoretically and experimentally the contribution gained from motion-compensation with fractional-pixel accuracy. In the theoretical analysis, the power spectral density of the prediction error is related to the probability density function of the displacement error. It predicts that the possibility of further improving prediction by more accurate motion-compensation is small if a critical accuracy is exceed. According to his analysis and experimental results, for a motion-compensation block size of 16×16 and typical broadcast TV signals, quarter-pixel accuracy is sufficient, while for videophone signals, half-pixel accuracy is desirable. Moreover, in views of different interpolation schemes, bilinear interpolation is as good as, or better than sinc-interpolation.

2.1.1.3 Overlapped Block Motion Compensation [5]

We can improve the inefficiency of block-based motion compensation due to variation of motion within a block, quantization of motion vectors, model mismatched such as uncovered background, etc, improved by using overlapped block motion compensation (OBMC). OBMC and its theoretical basis have been studied and proposed in the literature [24-27]. Several video coding standards, such as the H.263 and MPEG-4 also incorporate various forms of OBMC. The following description briefly presents the OBMC scheme in the MPEG-4.

When OBMC is enabled, each pixel in a 8×8 luminance *block* is a weighted sum of three prediction values, divided by 8 with rounding. In order to obtain the three prediction values, three motion vectors, MV_0, *MV*1 and *MV*2 are used:

1. the motion vector of the current luminance *block*, MV_0 ,
2. the motion vector of the block above or below the current luminance *block*, *MV1*, and
3. the motion vector of the block at the left or right side of the current luminance *block*, *MV2*.

For pixels in the current *block*, the motion vectors of *blocks* at the two nearest *block* borders are used. We name these motion vectors as remote *MV*s. For instance, $MV1_{above}$ and $MV2_{left}$ are the remote *MV*s for the pixels in top-left quarter of the current *block*. Figure 2-5 shows an example of a top left block in a MB and depicts the used remote *MV*s for the pixels within each quarter. Moreover, if one of the surrounding *blocks* was not coded, the corresponding remote *MV* is set to zero. If one of the surrounding *blocks* was coded in intra mode, the motion vector for the current *block* replaces the corresponding remote MV. If the current *block* is at the border of a frame or a VOP, the current motion vector replaces the corresponding remote *MV*. In addition, if the current *block* is at the bottom of the MB, the remote motion vector corresponding with a 8×8

luminance *block* in the MB below the current MB is replaced by the motion vector for the current *block.*

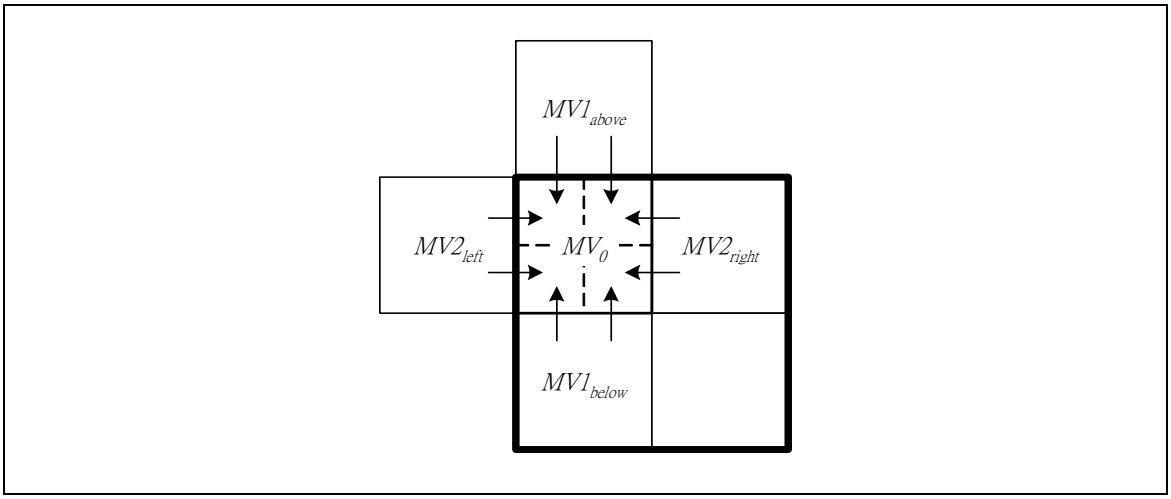


Figure 2-5. Illustration of the used remote motion vectors for the pixels within each quarter of a current *block*. (Solid line – MB; Thin line – 8×8 *block*; Dashed line – Partition of the pixels to four quarters in the current *block*)

The pixel values in the OBMC *block* shown in the example of Figure 2-5 are governed by the following equation

$$\bar{f}(i,j) = \left(f_0(i,j)\times H_0(i,j) + f_1(i,j)\times H_1(i,j) + f_2(i,j)\times H_2(i,j) + 4\right)/8 \qquad \textbf{(2-4)}$$

where $f_0(i,j) = f\left(i + MV_{0x}, j + MV_{0y}\right)$

$f_1(i,j) = f\left(i + MV1_{x,\,above\,or\,below}, j + MV1_{y,\,above\,or\,below}\right)$

$f_1(i,j) = f\left(i + MV2_{x,\,left\,or\,right}, j + MV2_{y,\,left\,or\,right}\right).$

subscript *x*,*y* denotes component of a motion vector.

The weighting matrices or OBMC windows, $H_0(i,j)$, $H_1(i,j)$ and $H_2(i,j)$ are defined in Figure 2-6.

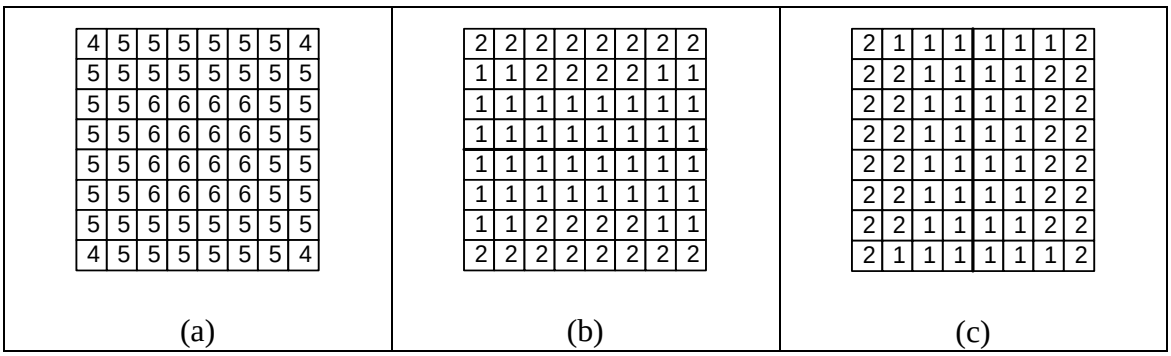


Figure 2-6. The weight matrices $H_0(i,j)$, $H_1(i,j)$ and $H_2(i,j)$ defined in the H.263 and MPEG-4.

2.1.2 Arbitrary shaped object coding

Nowadays, content browsing in the World-Wide Web becomes part of our daily life due to the successful and rapidly growth of Internet activities. Text-based and still image type interactive operations are already familiar to all Internet users. Demands of other interactive functionalities involving various contents, such as audio and video will increasing. In 1993, MPEG (Moving Pictures Experts Group) launched the MPEG-4 work item, which officially called "Coding of audiovisual objects". In addition to increasing compression efficiency for storage and transmission, one major characteristic of the MPEG-4 different from its predecessor is that it offers the capability of video and audio manipulation in multimedia environments. In order to provide the solutions for these objectives, a set of “tools” and “algorithms” for audio-visual data, called audio-visual objects (AV objects or AVOs) are being developed.

The MPEG-4 introduces the concept of Video Objects (VOs) to support access of individual semantic objects in visual contents. A temporal instance of a VO is named as a Video Object Plane (VOP). It is represented by its texture value (i.e. pixels luminance and chrominance values) and shape information [28]. In natural scenes, VOPs are obtained by semi-automatic or automatic segmentation [29-37], and the resulting shape information is represented as a binary alpha plane. On the other hand, for hybrid (of natural and synthetic) scenes generated by blue screen composition, shape information is represented by an 8-bit component, referred to as greyscale alpha plane.

An object-based coder is mainly composed of two parts, namely shape coder, and motion and texture coder. The emerging of the MPEG-4 attracted a number of researchers to study different approaches for shape coding [38-43], including the Context-based arithmetic encoding (CAE) [44-48]. The MPEG-4 adopts the CAE to code the shape of a VOP in the shape coder [49].

For a VOP, it is represented by means of a bounding rectangle, in which the rectangle can be defined as the minimum number of macroblocks that contain the object. There are three kinds of macroblocks within a bounding rectangle: the transparent MB, the boundary MB and the opaque MB. The boundary and opaque MBs include the pixels belonging to the object, and the transparent MB lies completely outside the object area. For different kinds of MBs, different coding techniques are used in MPEG-4. MPEG-4 employs the CAE to code arbitrary shape in a boundary MB. Similar to the frame-based situation, we can use inter-mode and intra-mode for the coding of a boundary MB with the CAE.

The MPEG-4 codes the texture information of a VOP by using similar techniques of the traditional video coding. We code an opaque MB as a normal MB in the fame-based coding. The transparent MB is not necessary to be coded since it has not contained any object pixels. Because of the coding efficiency, it is not suitable for us to apply traditional block-based coding techniques to the boundary MBs directly. We must handle the discontinuity condition at an object boundary carefully, otherwise resulting a degradation of rate-distortion and compression performance. A lot of researchers have been proposing many different approaches to improve the coding efficiency for a boundary MB [52-65] in the past decade.

Padding is one of the studied techniques in the course of the MPEG-4 standardization process. It aims at extending an arbitrarily shaped block to a regular block such that traditional hybrid block-based coding techniques can be applied. Two

padding techniques [52] have been proposed in 1997 for motion estimation and compensation of the boundary MBs. The MPEG-4 adopts the Repetitive padding for object-based video coding. With the repetitive padding, object shape information and prediction error, a boundary MB can be reconstructed as usual. The author in [55,56] proposed an Adaptive low-pass extrapolation padding technique for intra-mode coding, which increases the PSNR of boundary blocks by an average of 2dB. For inter-mode, padding the prediction errors outside the object with zeros gives good efficiency.

In additional to the padding techniques, shape adaptive transformations have been proposed in the literature [56, 58-64]. The Shape-Adaptive DCT (SA-DCT) has been suggested in the MPEG-4 to improve the coding efficiency of boundary MBs.

2.1.2.1 Forming of the bounding rectangle

According to the following procedure, a bounding rectangle with the minimum number non-transparent MBs will be obtained to bound the whole object as shown in Figure 2-7.

1. Generate a tightest rectangle with even numbered top left position.
2. If the top left position of this rectangle is the same as the origin of the image frame, extend the right bottom corner of the rectangle to form a final bounding rectangle that consists of multiples of 16×16 MBs.
3. Otherwise, form a control MB at the top left corner of the tightest rectangle. Note that the top left position of the control MB cannot site outside of the image.
4. Count the number of MBs that completely contain the object, starting at each even numbered point of the control MB. Details are as follows:
    a. Generate a bounding rectangle from the control point to the right bottom side of the object, which consists of multiples of 16×16 blocks.

b. The control point and the corresponding right bottom corner that resulting in the smallest number of the MBs for the given object is the coordinates of the bounding rectangle.

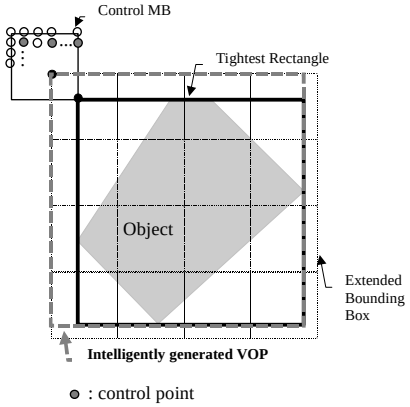


Figure 2-7. Formation of Bounding rectangle according to the procedure in the MPEG-4 VM [22].

2.1.2.2 Context-based arithmetic encoding

There are two major approaches in shape coding. The first one is contour-based shape coding, such as [38-43], which extracts and then codes a description of the closed contour enclosing the shape. Its advantages include scalability and having semantic representation, but complex treatments of intercepted contours and objects with holes are required. The second one is a bitmap-based approach, which encodes the bitmap of the binary alpha plane directly. It can achieve reasonable compression efficiency. However, this kind of approaches lacks direct semantic information about the coded object comparing to the previous approach. The techniques proposed in [44-48] belongs to this category.

MPEG-4 accepts a bitmap approach, which is the Context-based Arithmetic Encoding (CAE), to code an object shape. In CAE, it is assumed that a high degree of local correlation exists in the shape of a boundary MB. We name the shape of the MB as a Binary Alpha Block (BAB) in MPEG-4. Several coding modes are available including intra-mode and inter-mode. In intra-mode, a template of 10-pixels shown in Figure

2-8(a) is used to calculate a context number and defines the causal context for predicting the shape value of the current pixel. The context number is computed as $C = \sum_{k=0}^{9} c_k \times 2^k$, where c_k indicates the corresponding binary pixel value according to the template in Figure 2-8(a). It is used to access a probability table, which contains 1024 different contexts. For encoding the context state, a context-based arithmetic encoder is used.

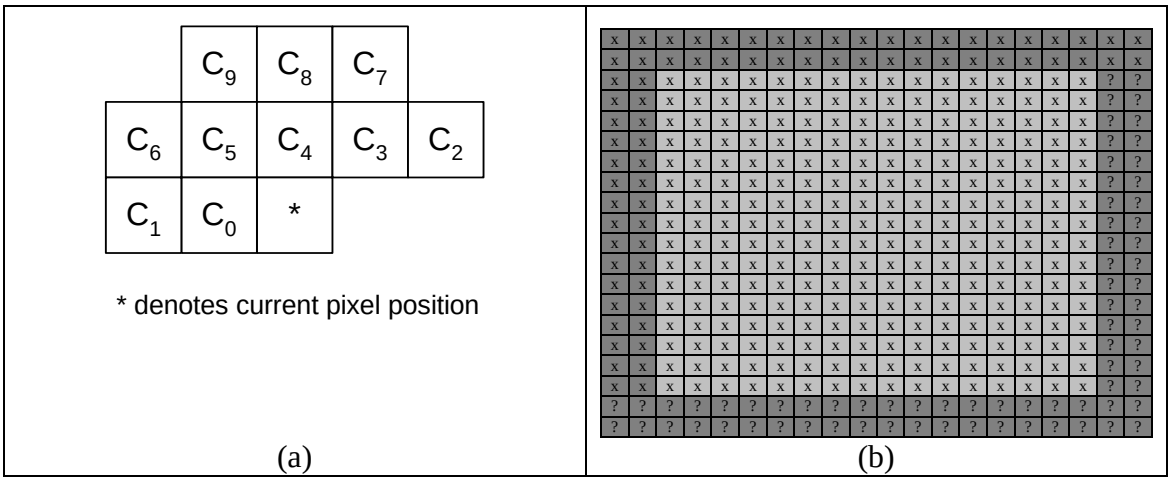


Figure 2-8. (a) The template for intra-mode context construction (b) Current bordered BAB.

When encoding a BAB, a border of width equal to 2 is extended from the current BAB for context number construction. The following rules must be obeyed to construct a current bordered BAB [22].

- Any pixels outside the bounding rectangle of a current VOP to the left and above are assumed to be zero.
- The template may cover pixels from BABs, which are not known at decoding time (value marked as "?" in Figure 2-8(b)). These unknown pixels are therefore estimated by template padding.
- For the intra-mode, the following steps are taken in the sequence
    1. if (c7 is unknown) c7=c8,
    2. if (c3 is unknown) c3=c4,
    3. if (c2 is unknown) c2=c3.
- For the inter-mode, if (c1 is unknown) c1=c2.

For inter-mode, temporal redundancy is exploited by taking advantage of the high correlation between two successive alpha planes. A reference BAB in a previous binary alpha plane is obtained by means of a motion vector. This BAB is regarded as a Motion Compensation BAB (MC BAB) in Figure 2-9(b). Similar to the situation of intra-mode, we calculate a 9-pixel context number, $C = \sum_{k=0}^{8} c_k \times 2^k$ using an inter-mode template in Figure 2-9(a) to access the probability table.

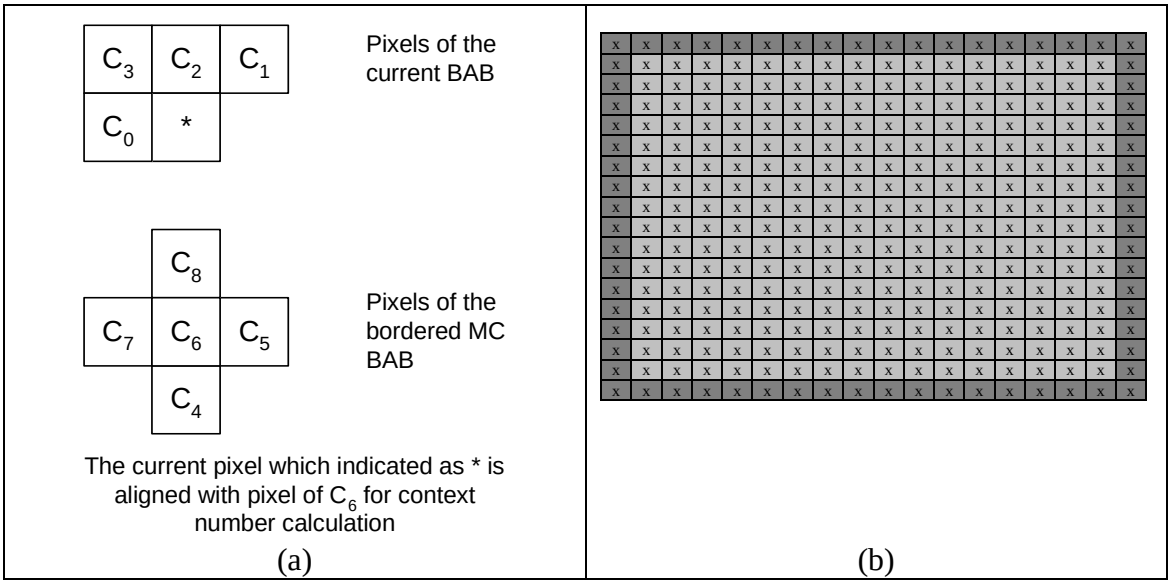


Figure 2-9. (a) The template for inter-mode context construction. (b) Bordered MC BAB.

A lot of shape-coding algorithms, including different bitmap-based and contour-based shaper, were thoroughly investigated in [49] with respect to their coding efficiency, subjective quality for lossy shape coding, hardware and software complexity, and performance in scalable shape coders. With the consideration of the required bandwidth for off-chip memory access and caching, MPEG-4 focused on optimizing the selected CAE for shape coding.

Furthermore, a context-based arithmetic coding is not limited for the purpose of shape coding. It can also serve for symbol coding, such as the study in [50]. MPEG-4 may also involves the CAE in rate control [51] by lossy shape coding.

2.1.2.3 Padding for arbitrarily shaped object coding

Image padding refers to enlarging the area of an image by filling additional pixel values in the enlarged area. For arbitrarily shaped object coding, a padding operation extends the object shape to a rectangle, such that traditional video coding algorithms can be employed. Clearly, a specific padding algorithm should be designed to deal with a particular application.

MPEG-4 treats the boundary MB as a regular MB and encodes the texture of each *block* using an 8×8 DCT. A decoder decodes the texture and discards all pixel values that outside of the object shape. In order to increase the coding efficiency, an encoder must pad the pixels outside of the object adaptively such that the bitrate is minimized. For intra-mode, MPEG-4 makes use of a low-pass extrapolation padding [55,56] to achieve this purpose.

2.1.2.3.1 The Low-pass Extrapolation Padding

For a boundary MB which is coded in intra-mode, it is preferred to pad the given image data to a rectangular area such that conventional block-based DCT coding can be applied. The authors of references [55,56] proposed the low-pass extrapolation (LPE) padding for the intra-mode coding. The LPE aims at seeking for a computationally simple extrapolation method which is operated completely in the spatial domains. Moreover, the padding result must fulfill the following two criteria. 1) The signal extension should be smooth enough, i.e. it should have a low-pass characteristic. 2) Discontinuities at the border between given and extrapolated image data should be avoided. To resolve this problem, it can be regarded as solving the variational problem of Dirichlet. It is identical to the task of solving the differential equation under a given boundary conditions.

$$\Delta f = f_{xx} + f_{yy} = 0 \qquad \textbf{(2-5)}$$

where Δf is gradient of an image function, *f*.

f_{xx} is the second derivate of f with respect to *x*.

f_{yy} is the second derivate of f with respect to *y*.

The above equation can be approximated by finite differences using Laplacian operator for a digitized image. We let *i* and *j* are the variables in x and y direction respectively.

$$\Delta f(i,j) = f(i,j-1) + f(i-1,j) + f(i,j+1) + f(i+1,j) - 4f(i,j). \qquad \textbf{(2-6)}$$

Since a direct solution of this equation is not possible in general, the author adopted a relaxation method for determining the function *f*. This leads to a simple averaging operation, which is applied iteratively to all transparent pixels in a boundary MB.

$$f^{k+1}(i,j) = f^{k}(i,j) + \frac{1}{4}\Delta f^{k}(i,j) \qquad \textbf{(2-7)}$$

where *k* indicates stages of the iteration.

If one or more of the four pixels are outside of an image block during the averaging operation, the corresponding pixels are not considered. Hence, the averaging processes can be equated as equivalent to a convolution process applying iteratively to an image block. Let us represent the image block as elements of a matrix $[F^k]$, i.e. $F^k_{ji} = f^k(i,j)$.

Result of the processes is given by

$$[F^{k+1}] = \left[\sum_{l=0}^{4} c_l\right]^{-1} \begin{bmatrix} 0 & c_1 & 0 \\ c_4 & 0 & c_2 \\ 0 & c_3 & 0 \end{bmatrix} * [F^k] \qquad \textbf{(2-8)}$$

where * is a convolution operator

$$c_l = \begin{cases} 1 & \text{if inside block} \\ 0 & \text{otherwise.} \end{cases}$$

According to the experimental results [56], the author claimed that the first iteration could already provide enough smoothing effect for efficient DCT coding. Hence, it is

sufficient to restrict the LPE padding to a single iteration for the sake of lower computational load.

2.1.2.3.2 The Repetitive Padding

Block-based motion estimation and compensation requires an arbitrarily shaped object to be padded properly so that motion prediction coding can be efficient. MPEG-4 accepts the repetitive padding proposed in [52]. In the repetitive padding, a boundary MB is padded by replicating the boundary samples of the VOP towards the exterior. First, this replicating process is divided into two stages, which are the horizontal repetitive padding and vertical repetitive padding. If a value lies between two pixels in a row or column, an average value is assigned to this pixel. Second, since this repetitive padding puts a significant computational burden on the decoder, we use an extended padding to pad the remaining MBs that are completely outside the object.

In the horizontal repetitive padding, we fill each transparent pixel in a boundary MB by replicating each pixel at the edge of a VOP horizontally to the left and/or right direction. If there are two edge pixels values for filling a transparent pixel outside of a VOP, we average the two edge pixels values. Figure 2-10(a) demonstrates an example of the horizontal padding in a boundary MB. Afterward, the remaining unfilled transparent region in the MB is padded by a similar manner as the horizontal process but in the vertical direction. Pixels already filled in the horizontal repetitive padding are regarded as if they were the video object pixels for the purpose of this vertical stage. Figure 2-10(b) and (c) shows the vertical repetitive padding and the repetitive padded boundary MB respectively.

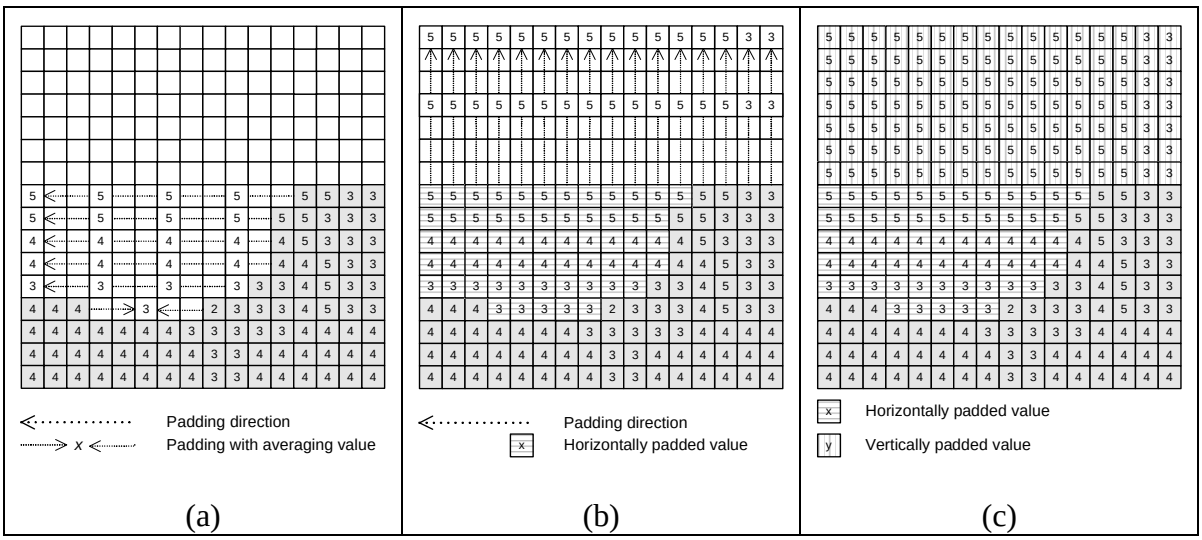


Figure 2-10. (a) Horizontal repetitive padding by replicating pixels at the edge of a VOP horizontally to the left. In the thirteen row, averaging of two edge pixels values is used to fill the transparent pixel values lied between. (b) Processes of the vertical repetitive padding. (c). The resulting boundary MB after the repetitive padding.

The macroblocks immediately next to boundary MBs are named as exterior MB in the padding processes. They are filled by horizontally or vertically replicating the pixel values at the border of their surrounding boundary MBs in the extended padding. If an exterior MB is surrounded by more than one boundary MBs, we shall pick one of these MBs according to the priority stated in Figure 2-11 for this padding. The boundary MBs with the largest priority number is used for the extended padding of an exterior MB. The remaining unfilled MBs in the bounding rectangle are filled with mean of all possible pixel value. We assume that probability of occurrence of each pixel value is uniform. For 8-bit pixel representation, they are filled with 128.

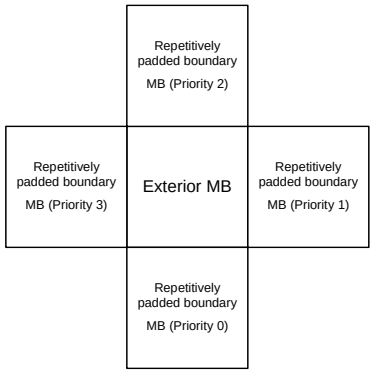


Figure 2-11. Priority of boundary MBs surrounding an exterior MB for the extended padding.

The author [52] claimed that efficient motion estimation and compensation for the object boundary is most critical to the performance of object-based video coding. Moreover, the discontinuity at object boundary could degrade the coding performance seriously if not handled properly. Comparing to the non-padded case, the repetitive padding improves the rate-distortion performance of motion prediction for about 20%.

2.1.2.4 Motion compensation for arbitrarily shaped object

After a reference VOP has been repetitively padded, motion compensation can be performed for a VOP reconstruction. A special treatment should be done for a boundary MB in MPEG-4. Using the coded motion vector and shape information, we can reconstruct a boundary MB according to the following equation.

$$f_c(i,j) = \left[f_{ref}(x+i+u, y+j+v) + e_{coded}(i,j) \right] \times Alpha_{coded}(i,j) \quad \textbf{(2-9)}$$

where $f_c(\cdot,\cdot)$ and $f_{ref}(\cdot,\cdot)$ is a reconstructed MB and reference VOP respectively

$e_{coded}(\cdot,\cdot)$ is the coded prediction error.

$Alpha_{coded}(\cdot,\cdot)$ is the coded alpha plane.

(x,y) denotes location of a current MB.

(i,j) is coordinate of each pixel in a MB.

(u,v) is the motion vector.

2.2 *Block-based motion estimation review*

In the Block-based compensation scheme, we use motion vectors to represent motion activities of objects between the current frame and reference frames. Although industrial standards do not specify a particular motion estimation technique, block-based motion estimation is a natural selection for video encoding.

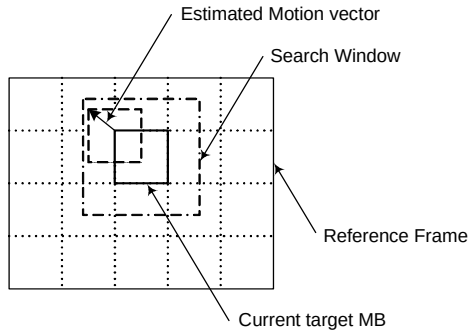


Figure 2-12. Block-based motion estimation.

A motion estimation process in an encoder obtains a motion vector by the using block-based matching technique. The motion estimation detects the interframe motion with the use of a cost function between a target MB in the current frame and a candidate MB within a search window in a reference frame. The displacement between the most suitable candidate MB and the target MB is defined as the resulting motion vector, $(\hat{u},\hat{v})$. A variety of cost functions have been investigated in the literature. One such function is the cross-correlation function, (CCF) which is mentioned in [69] and defined as

$$CCF(u,v)=\frac{\sum_{i=0}^{16}\sum_{j=0}^{16} f_t(i,j)\,f_{ref}(i+u,j+v)}{\left[\sum_{i=0}^{16}\sum_{j=0}^{16} f_t^{\,2}(i,j)\right]^{1/2}\left[\sum_{i=0}^{16}\sum_{j=0}^{16} f_{ref}^{\,2}(i+u,j+v)\right]^{1/2}}. \quad \textbf{(2-10)}$$

Where $f_t(\cdot,\cdot)$ is a target MB in the current frame at time *t*.

$f_{ref}(\cdot,\cdot)$ is a candidate MB in a reference frame with a motion vector (*u*,*v*).

However, the authors in [8] claimed that correlation method is not suitable for small block size condition, and a mean distortion function was subsequently proposed. In fact, this distortion function is a mean of squared l_2-norm distance. An l_2-norm of a vector $A=[a_1,a_2,\ldots,a_n]$ is defined by $\sqrt{\sum_{i=1}^{n}|a_n|^2}$. The squared l_2-norm distance for the prediction errors of a MB between a target MB at position (x,y) in the current frame, f_t, and a candidate MB at position $(x+u, y+v)$, in a reference frame, f_{ref}, is defined as below,

$$SSD_{M\times N}(x,y;u,v)=\sum_{i=0}^{N}\sum_{j=0}^{M}\left|f_t(x+i,y+j)-f_{ref}(x+i+u,y+j+v)\right|^n \quad \textbf{(2-11)}$$

where $n = 2$ and $N = M = 16$.

It is also called the Sum of Squared Difference (*SSD*). Block matching with l_2-norm minimizes the energy of the prediction errors, and for Gaussian signals, it minimizes the bitrate required to encode the errors. In practice, block matching with l_1-norm distance is often used due to its lower complexity for hardware implementation. The l_1-norm is defined by (2-11) with $n = 1$. It is normally named as Sum of Absolute Difference (*SAD*). The *SAD* can be viewed as an approximation to block matching with l_2-norm distance. Furthermore, some researchers have studied other cost functions, including the Pixel Difference Classification (PDC) [66], the Minimized Maximum Error (MME) [67] and the Geometric Mean of the DCT coefficient variances (GMDCT) [68]. The former two methods were explored when the consideration of hardware realization is taken. For the GMDCT, the authors developed the criterion by considering a spatial domain coder, when the coder is based on DCT and dynamic bit allocation. When the problem of optimally allocating a limited bitrate to the displacement vector field and the motion compensation prediction error is addressed, a rate-constrained motion estimation theoretical framework was introduced in [70]. The authors [71,72] introduce a Lagrangian cost function by making use of the Lagrange multiplier for the rate-constrained motion estimation.

Instead of being as a matching criterion, we often use *SSD or* its mean value to evaluate the searching ability between different motion estimation algorithms. Another frequently used metric for performance evaluation is the peak signal-to-noise ratio (PSNR), which is defined as follows,

$$PSNR = -10\log_{10}\left(\frac{\frac{1}{M\times N}\sum_{x}^{N}\sum_{y}^{M}\left|f_t(x,y)-f_{rec}(x,y)\right|^2}{255^2}\right) \quad \textbf{(2-12)}$$

where N and M are the width and height of a frame respectively, in frame-based situation,

$f_t(\cdot,\cdot)$ and $f_{rec}(\cdot,\cdot)$ are the current and compensated frame respectively.

When the SAD is used for block matching criteria, the motion vector of the best matched block, $(\hat{u},\hat{v})$ is given by,

$$(\hat{u},\hat{v}) \equiv \arg\min_{(u,v)\in W} SAD(x,y;u,v) \quad \textbf{(2-13)}$$

where $W=\{(u,v)\,|\,-D\le u,v\le D\}$ is a set of all possible locations in a search window and D is the maximum possible displacement of the motion vector (u,v).

The simplest block matching motion estimation algorithm is the full search algorithm (FSA). This algorithm can give an optimum solution by exhaustively examining all possible locations within the search window. In addition to its optimum result, the advantages of block-based FSA in video-coding application are its regularity and simplicity. It is suitable for a simple hardware realization. However, its heavy computational load for a large search range can be a significant problem in real-time applications. For example, a search window for a maximum possible displacement, *D*,

will require $(2D+1)^2$ number of calculation of *SAD*. Equation (2-11) shows that a *SAD* involves $M \times N$ absolute operations and $M \times N - 1$ additions for a block. In order to resolve this difficulty, many fast search algorithms have been developed in the past.

Basically, the factors that determine the performance of a block matching motion estimation algorithms are matching criteria, search schemes and the search area. A lot of researchers make use of these factors to investigate a large variety of fast algorithms in the past [8, 69, 73-108]. These fast search algorithms can be classified into the following categories. 1) In the first category, the fast search algorithms seek for a way to select a subset of the candidate MB in *W* to reduce the computational time [8, 69, 73-86]. The most challenging part of these algorithms is to determine the subset of *W* for searching. Because these algorithms can easily be trapped into local minima, degradation in predicted images is an inevitable result on average. Many researchers select an initial searching point by studying the motion field to reduce the probability of being trapped in local minima. Another approach to find a good initial point is hierarchical or multiresolution techniques [87, 88]. 2) The algorithms [73, 89-98] in this category use a reduced complexity distortion measure to save computation, such as pixel decimation [73, 89-91] and partial distortion (PDS) techniques [92-96]. The pixel decimation techniques subsample the pixels in a target MB and the candidate MBs within the computation of the SAD. Hence, the computation for each SAD can be reduced. The PDS reduces the computation complexity by terminating the SAD calculation early when it finds that a partial SAD is already greater than the minimum SAD encountered so far in the search. In general, PDS is regarded as a fast full search algorithm because it has identical prediction quality as that of the FSA. 3). The algorithms [99-103] in the third category make use of mathematical inequalities to reduce the computational load; this includes the Successive Elimination Algorithm (SEA) [99-101]. By making use of the Minkowski's inequality, the SEA eliminates an

impossible candidate MB without calculating the SAD. 4) Edges are the most prominent feature in image processing. They are also frequently used to predict pixel matching errors in motion estimation. The fourth category [104, 105] uses information of edges in a block to reduce computational burden in motion estimation. 5) Moreover, a performance analysis of MPEG-4 decoder and encoder [107] shows that motion estimation will remain as a computationally intensive step in MPEG-4 arbitrarily shaped VOs encoding. Object based motion estimation is a new direction in this field [108]. Many other algorithms combine the above techniques together in order to further improve the coding efficiency.

In addition to the above classification, another frequently used classification is to compare the quality of a motion compensated frame by a motion estimation to that of the FSA. If a searching technique can produce identical quality as the FSA, The motion estimation is regarded to be a lossless algorithm. Otherwise, it is a lossy one. The algorithms proposed in [96-101, 103] are categorized as lossless algorithms, which search all candidate positions in W and save the computational load by making use of the Minkowski's inequality or partial distortion in a *SAD*. Other algorithms introduced are lossy because they do not search all of possible locations in W and involve subsampling of pixels or approximation of mathematical inequalities in the calculation of matching criteria.

One of the differences between the H.264/AVC [6] and previous video standards is that it increases the coding efficiency by allowing an encoder to select reference pictures for motion compensation among a larger number of pictures. These reference pictures have been decoded and stored in the decoder. The studies in [109-111] have investigated the subject of this new motion compensation and estimation problems.

Among these wide variety of motion estimation techniques, four fast algorithms are going to be discussed in detail. Section 2.2.1 introduces the major ideas of lossy

motion estimation. The discussions mainly focus on the Partial Distortion Search (PDS) algorithm and its modification, Normalized PDS. In Section 2.2.2, a simple idea that can improve the efficiency of the traditional PDS is illustrated. Two adaptive PDS algorithms, which utilize this technique, are presented. Their basic idea is explained in details. Section 2.2.3 gives the procedure of the Diamond Search (DS). We shall present the explanation about the success of the DS. Another algorithm making use of the mechanism is also introduced and compared. Section 2.2.4 concentrates on the introduction of the Motion Vector Field Adaptive Search Technique (MVFAST) and its generalization, the Predictive MVFAST (PVMVFAST). It gives the step by step implementation of the MVFAST. Note that MPEG-4 has already accepted these two algorithms in its optimization model.

2.2.1 Partial Distortion Search

Using Partial Distortion or the Minkowski's inequality are two major approaches, which are mostly utilized to develop different lossless motion estimation algorithms or fast Full Search Algorithm.

The l_1-norm version of the Minkowski's inequality gives a relation between two arbitrarily sets of non-negative real numbers. For two given sets $A = \{a_0, a_1, ..., a_{N-1}\}$ and $B = \{b_0, b_1, ..., b_{N-1}\}$ which have same number of elements, they obey the following inequality,

$$\|A - B\| \geq \|\|A\| - \|B\|\|. \tag{2-14}$$

When (2-14) is applied involving *SAD*, it is expressed as

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|I_t(i,j) - I_{ref}(i+u, j+v)\right| \geq \left|\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|I_t(i,j)\right| - \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|I_{ref}(i+u, j+v)\right|\right| \tag{2-15}$$

It means that an absolute different between l_1-norm of a reference block and that of a target MB is always smaller or equal to their *SAD*. The Successive Elimination Algorithm (*SEA)* compares the norm differences to the encountered minimum *SAD*, SAD_{temp_min}. The intensive computations of *SAD* are not necessary if the norm differences are greater than the SAD_{temp_min}.

The basic idea of the PDS algorithm is described as follows. For a given reference MB and a target MB, a partial distortion is defined as a part of the total distortion. The *p*-th accumulated partial distortion in the traditional PDS algorithm is given by (2-16)

$$SAD_p(x,y;u,v) = \sum_{j=0}^{p}\sum_{i=0}^{15}\left|I_t(x+i,y+j) - I_{ref}(x+i+u,y+j+v)\right|. \quad \textbf{(2-16)}$$

The value *p* indicates the number of rows accumulated into the *p*-th partial distortion as shown in Figure 2-13(a). If the *p*-th accumulated partial distortion, SAD_p, is greater than the SAD_{temp_min}, the coder can simply rejects this candidate without calculating the remaining partial distortion. This algorithm can greatly reduce computation of the distortion calculation if the saving of computation in the remaining partial distortion can compensate the additional comparison operations.

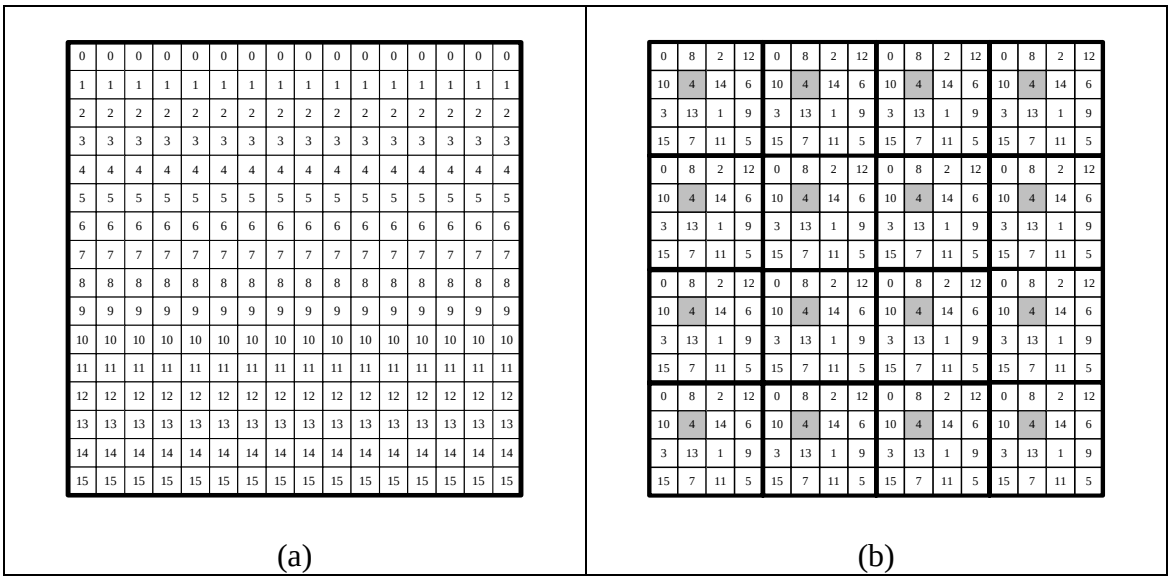


Figure 2-13. Order of calculation of the partial distortions.

Instead of define the partial distortion in row-by-row manner, the author who proposed the Normalized Partial Distortion Search (NPDS) [95] using a uniform pattern to define the order of partial distortion calculation. The recommended order of calculation is depicted in Figure 2-13(b). Afterward, they normalize the accumulated partial distortion and the SAD_{temp_min} before comparison. The probability of early rejection of non-possible candidate MB is thus increased.

Every location in the search window is searched one by one with normal raster scanning order from left to right and top to bottom in the traditional PDS. However, it is well know that most real-world sequences have a centrally biased motion vector distribution. The motion vectors tend to concentrate at the central region of a search window. We can use spiral scanning path to exploit this motion-vector distribution characteristics. The spiral scanning begins the searching at the center of a search window and then moves outwards with a spiral manner as shown in Figure 2-14. This order of scanning will increase the probability of meeting the global minimum and thus rejects the impossible candidate MBs earlier.

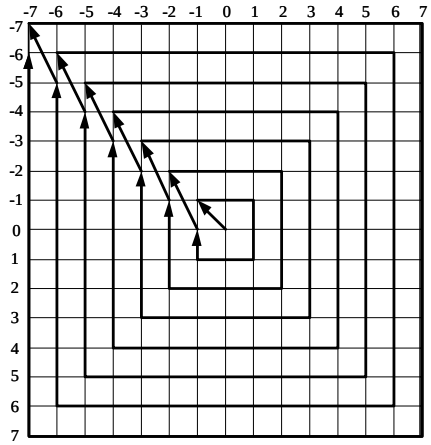


Figure 2-14. Spiral scanning path in a search window with *D*=7.

The Minkowski's inequality and partial distortion can be used to reduce computation required for the calculation of SAD. If we examine all possible candidates in a search window, identical result as the FSA can be obtained. The overhead

introduced by the SEA is the calculation and comparison of the sum norm. The authors [99] reduce the computation required for the sum norm by using a recursive technique. Hence, all sum norms are pre-calculated and stored in a memory buffer. The computational saving of the SEA and spiral-PDS have been evaluated in [96]. The results shows that performance of the SEA and spiral-PDS are comparable but the PDS does not require additional preprocessing. Note that the normalized PDS is not a lossless algorithm although it scans all possible candidate MBs. Moreover, the partial distortion of pixels is accumulated in a uniform pattern; such that rejection of impossible candidates by normalize partial distortion is more accurate. However, for a hardware implementation, uniform pattern may not be desirable as it results in more irregular memory access.

2.2.2 Adaptive Partial Distortion Search

The capability of eliminating an impossible candidate MB by conventional PDS depends on the comparison between SAD_p and SAD_{temp_min}. If one can eliminate the candidates at lower p-th accumulated partial distortion, the saving of computation can be increased. We can develop different Adaptive Partial Distortion Searches based on this idea.

The algorithm [96], "Fast Full-Search Motion-Estimation Algorithm Using Representative Pixels and Adaptive Matching Scan", (AMS-PDS) is one of these adaptive approaches. The authors approximated the gradient magnitude of an image, $\left|G\left[I(x,y)\right]\right|$, by the following representation,

$$\left|G\left[I(x,y)\right]\right| = \sqrt{\left(\frac{\partial I(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I(x,y)}{\partial y}\right)^2} \approx \left|\frac{\partial I(x,y)}{\partial x}\right| + \left|\frac{\partial I(x,y)}{\partial y}\right| \quad \textbf{(2-17)}$$

They used Taylor series expansion to formulate a predicting function of an image at some position, α_{t+1}, in terms of the image function and its spatial derivative at a nearby position α_t as given below,

$$I_t(\alpha') = I_t(\alpha'') + \frac{\partial I_t(\alpha'')}{\partial \alpha''}(\alpha' - \alpha'') + \frac{1}{2!}\frac{\partial^2 I_t(\alpha'')}{\partial \alpha''^2}(\alpha' - \alpha'')^2 + \cdots + \frac{1}{n!}\frac{\partial^n I_t(\alpha'')}{\partial \alpha''^n}(\alpha' - \alpha'')^n + \cdots \quad \textbf{(2-18)}$$

where α_t indicates the position (x,y) of the t-th frame.

After that, they approximately described the relation between a current frame at time $t+1$ and a reference frame at time t using (2-18). Let $e_{t+}1(\alpha)$ be the prediction error and given by

$$e_{t+1}(\alpha) = \left| I_{t+1}(\alpha) - I_t(\alpha + cmv) \right| \quad \textbf{(2-19)}$$

where cmv is a candidate motion vector, ($cmvx$, $cmvy$),

assuming that $I_{t+1}(\alpha) = I_t(\alpha+mv)$, where mv=(mvx, mvy) is the true motion vector of a pixel at position α. By substituting $\alpha' = \alpha + mv$ and $\alpha'' = \alpha + cmv$ into (2-18), we have

$$e_{t+1}(\alpha) \approx \left| \frac{\partial I_t(\alpha + cmv)(cmv - mv)}{\alpha} \right| \quad \textbf{(2-20)}$$

According to the approximation of the gradient magnitude defined in (2-17), the authors claimed that the matching distortion at position α is proportional to the gradient magnitude of reference block in the current frame. Another researcher [106] also derived this relation by different approach.

The development of AMS-PDS is based on the above relation. Authors of the AMS-PDS used adaptive block-matching-scan instead of the conventional raster matching scan to calculate the SAD_p. They use the image's gradient magnitude to determine the scanning direction and order. Two approaches have been developed in [96]. We only describe the details of the one with the best performance in the following paragraph.

The AMS-PDS, first calculates magnitudes of pixel gradient in a MB by using the finite difference from (2-17). Second it sums up gradient magnitudes of four *blocks* as labeled in Figure 2-15(a). Let us define the sum of gradient magnitudes of *block* (i) to be |*block*(i)|. Then, results of four cases, |*block*(i)|+|*block*(ii)|, |*block*(iii)|+|*block*(iv)|, |*block(i)*|+|*block*(iii)| and |*block*(ii)|+|*block*(iv)| are evaluated to find the maximum value. These results are used to decide horizontal-matching or vertical-matching scan. Conceptually, the AMPDS assumes that the sum of gradient magnitudes of two aligned *blocks* is maximum, the larger errors tends to line up along the same direction. It makes the decision according to the following rules: 1) Horizontal-matching scan if |*block*(i)|+|*block*(ii)| or |*block*(iii)|+|*block*(iv)| has maximum value. 2) Vertical-matching scan if |*block*(i)|+|*block*(iii)| or |*block*(ii)|+|*block*(iv)| has maximum value. With the selected scanning direction, it sorts the gradient magnitudes of rows or columns in the MB as illustrated in Figure 2-15(b) and (c) respectively. Finally, the SAD_p is accumulated with the sorted order. For instance, the numbers indicated in Figure 2-15(b) and (c) determines the order of a row or column that sum up to the SAD_p.

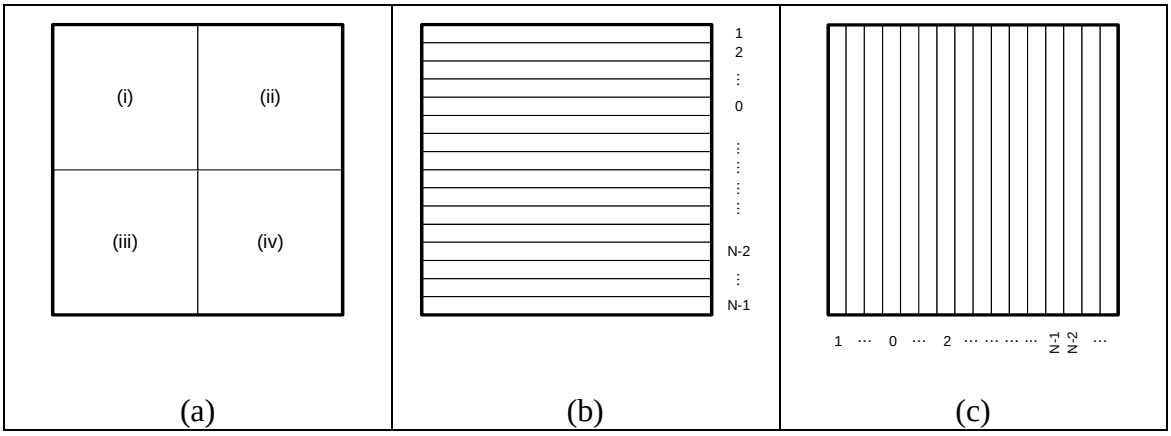


Figure 2-15. The AMS-PDS scheme. (a) Block division for determination of rows or columns scanning. (b) Horizontal matching scan by sorted row gradient magnitudes. (c) Vertical matching scan by sorted column gradient magnitudes.

Another researchers [97] also proposed a different adaptive PDS using the same relation (2-19). They determined the accumulation order of pixel matching errors to SAD_p in a more detailed way. Gradient magnitudes of a set of nearby pixel pairs are

used to determine the accumulation order. Besides, they suggest using the Hilbert scan [123] to further reduce the computational complexity of the PDS. The Hilbert scan illustrated in Figure 2-16 preserve some of the 2-dimension spatial coherence of the scanned data-space on a single dimensional sequence formed by the scan. They calculated the gradient along the Hilbert scanned sequence and sorted the results.

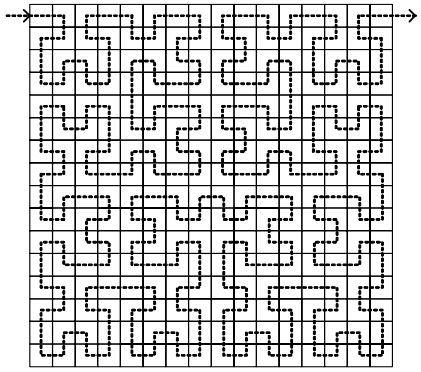


Figure 2-16. The Hilbert scan in a MB.

In this algorithm, a total of 255 gradient magnitudes need to be sorted and a conventional Count Sort algorithm is used. An outline of the Count Sort is expressed here. For a given positive integer data set, $A = \{a_0,...,a_n,...,a_{N-1}\}$, the algorithm counts the number of elements in A not exceeding a_n for every n. By using the element values to indexing an array of counter which is declared as $C[m]$, $m=0,\ldots,max\{A\}$, the counting results are kept in the array. At the end of the counting, the array, $C[m]$, carries the required complete information to form the sorted A. (need a simple illustration?)

Accumulating the matching errors adaptively to SAD_p is a simple and efficient method for improving the performance of the partial distortion search. All of these two algorithms provide significant progress. It must be point out that irregular memory access is its major disadvantage for a hardware implementation. In addition, the relation formulated by (2-19) is only valid in a region close enough to the best matching position due to the limitation of the Taylor expansion series. In a practical search window, a lot of candidate motion vectors are far from the best one. As a result, gradient-based

adaptive technique limits the improvement of the PDS. Moreover, these two algorithms do not suitable for coding of boundary MBs in arbitrarily shaped objects, unless modification is applied.

2.2.3 Diamond Search Algorithm

Lossy motion estimation development attracts much attention in the field of video coding, although different fast lossless algorithms have been proposed in the literature. The computational requirement of these lossless techniques is still not suitable for real time applications such as video conferencing and visual telephony sequences. Fortunately, it is observed that the locally changed areas usually small and restricted especially for sequences with low motion activities. Moreover, empirical experiences show that error surfaces encountered during motion estimation are often decreasing monotonically. These characteristics motivate the design of any fast search algorithms that searching only a subset of the candidate MB in a search window.

The MPEG-4 verification model, VM14 [120] has adopted the Diamond Search algorithm [80] for the motion estimation. In the proposition of the Diamond Search (DS), the authors stated that the shape and size of search patterns jointly determine not only the error performance of fast block matching algorithms but also their search speed. The authors using several commonly used test image sequences to investigate the motion vector distribution probabilities based on the FSA with the mean-square difference (MSD) matching criterion. Their results indicated that about 53% (in large motion case) to 99% (in small motion case) of the motion vectors are enclosed in a circular support with radium of 2 pixels and obey the centrally biased motion vector distribution. Furthermore, the displacement of real-world video sequences could be in any direction but mainly in horizontal and vertical directions due to camera panning. Based on these two observations, the author advised using the search points within a

circle of radius equal to 2 pixels units to compose the search pattern. These search points are indicated by "×" in Figure 2-17(a) within a dotted-line circle (radius = 2).

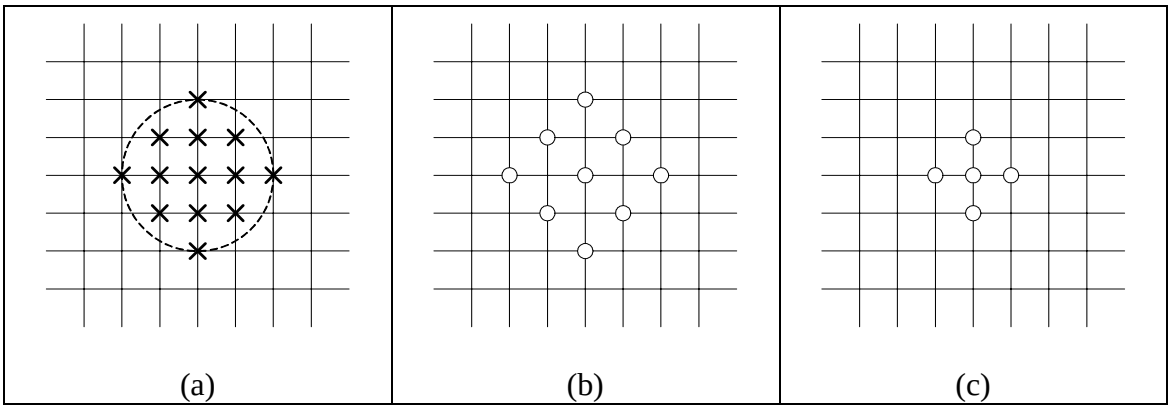


Figure 2-17. (a) The appropriate search pattern support proposed in Diamond Search. (b) Large diamond search pattern (LDSP), and (c) Small diamond search pattern (SDSP)

The DS algorithm employs two search patterns as illustrated in Figure 2-17(b) and (c) which are derived from the crosses "×" marked in Figure 2-17(a). The first pattern, called large diamond search pattern (LDSP), comprises of 9 checking points from which eight points surround the center one to compose a diamond shape. The second pattern consisting of 5 checking points forms a smaller diamond shape, called small diamond search pattern (SDSP). During the search, LDSP is repeatedly used until the step in which the minimum block distortion occurs at the center point. The search pattern is then switched from LDSP to SDSP as reaching to the final search stage. Among the five checking points in SDSP, the position yielding the MBD provides the motion vector of the best matching block. Note that a maximum overlapping region is chosen such that number of search points at each next step will be minimized. The *SAD* values of the search points obtained in the previous stage were stored in the memory, and thus re-computation is not necessary. In addition, DS algorithm does not restrict the number of search range. However, since all the *SAD* values found along the search path are always in a decreasing order, the search path is impossible to form a closed search loop. Therefore, the convergence of DS algorithm is guaranteed.

In 2002, the authors [84] investigated the principle of the diamond shape pattern that its speed improvement outperforms other square-shaped search pattern. Hence, they proposed a Hexagon-Based Search Algorithm (HEXBS) that can achieve substantial speedup comparing to the DS with similar MAD performance.

The authors found that the LDSP can arrive a far minimum position with fewer search points and also have lesser probability to be trapped in local minima due to its relatively large step size in both horizontal and vertical directions. Nevertheless, the authors pointed out that the advancing speed of the DS is 2 pixels/step horizontally and vertically but $\sqrt{2}$ diagonally as illustrated in Figure 2-18(a). It means that speedup performance of the DS is sensitive to motion vectors in different directions. In order to remedy this disadvantage, it prefers to have a search pattern approximate enough to a circle. Hence, each search point can be utilized with maximum efficiency. The authors proposed a large and small hexagonal search patterns, which depicted in Figure 2-18(b) and (c) respectively.

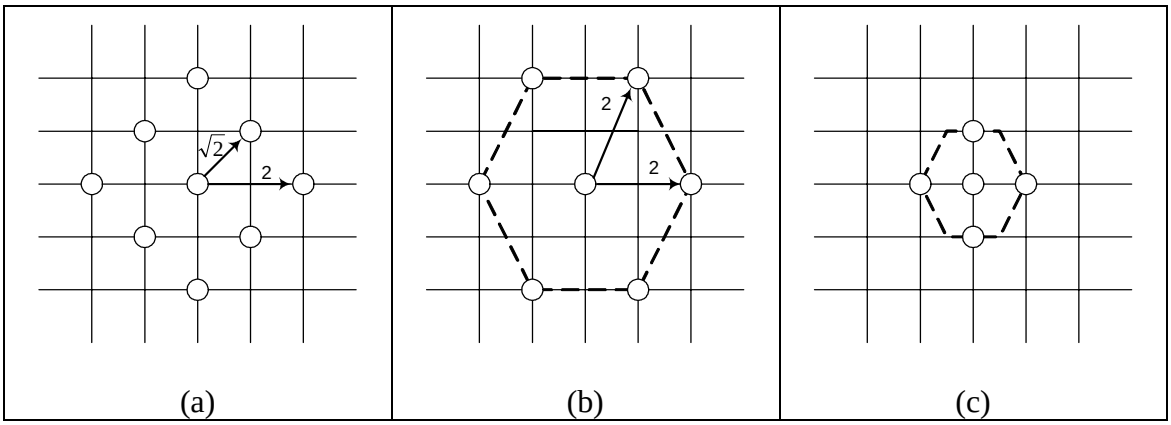


Figure 2-18. (a) Search Step size of Large diamond search pattern, (b) proposed large HEXBS pattern and (c) small HEXBS pattern.

The DS and HEXBS have much better performance when comparing to other square-based searching algorithms, such as the [73, 75, 78 and 82]. Comparison between the DS and HEXBS shows that the speed improvement rates of HEXBS over DS are as high as about 40% with similar distortion performance. Because these algorithms assume monotonically decreasing error surface, the major disadvantage is

trapping in local minima. This problem often occurs in sequences with high motion activities and boundary MBs of video objects, in which complex error surface happens frequently.

2.2.4 Motion Vector Field Adaptive Search Technique

The Motion Vector Field Adaptive Search Technique (MVFAST) [85] is regarded as an enhancement of the DS. It can get higher search speed with better PSNR performance. The significant improvement of search efficiency is due to the application of the large and small diamond patterns adaptively. The main idea is to select an appropriate initial search point, while the point has high probability to close to the global minimum. Then it performs the searching starting from the initial point with different search pattern according to the motion activity. Details of the MVFAST are described in the following.

Determination of local motion activity is the first step of the algorithm. A local motion vector field at a macroblock position is defined as the set of motion vectors in a region of support (ROS) of that MB. The ROS of a MB defined in MVFAST includes 3 neighborhood MBs as shown in Figure 2-19.

Figure 2-19. Region of support (ROS) for the current MB consists of MB1, MB2 and MB3.

Let $V = \{MV_{zero}, MV1, MV2, MV3\}$ **(2-21)**

where $MV_{zero} = (0,0)$, and MVi ($i=1,\ldots,3$) is the motion vector of MBi in the ROS.

We define sum of each component's length of a motion vector as a MV-length. A MV-length of a motion vector $MV_i = (u_i, v_i)$ is thus given by $l_{v_i} = |u_i| + |v_i|$. By using MV-lengths of the *MVi*, the motion activity at the current MB position is defined as follows.

Let $L = \max\{l_{v_i}\}$ for all MV_i.

$$\begin{aligned} \text{Motion Activity} &= \text{Low}, && \text{if } L \le L_1; \\ &= \text{Medium}, && \text{if } L_1 < L \le L_2; \\ &= \text{High}, && \text{if } L > L_2; \end{aligned}$$

where L_1=1 and L_2=2 are the greatest distance from the centre point of the DS patterns to any point on the small and large search pattern (Figure 2-17(b) and (c)) respectively.

The second step is the selection of the search center. A search center is the initial search point in motion estimation. The choice of the search center depends on the local motion activity at the current MB position. If the motion activity is low or medium, the search center is the origin. Otherwise, the *MV* that yields the minimum *SAD* is chosen as the search center.

With the determined motion activity, different searching is performed around the search center. The MVFAST uses the original diamond search if the motion activity is medium. On the other hand, it uses small diamond search pattern for the local search and the resulting motion vector is obtained if center point yields the minimum *SAD*.

The MVFAST also includes an optional mode, early elimination of search. It terminates a local search immediately when the *SAD*(*0,0*) is less than a threshold *T=512*. The resulting motion vector is assigned as (*0,0*).

Another new algorithm, named Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) [86], which generalize the predictor (search center) selection and use adaptive thresholding techniques. PMVFAST is a median motion vector biased algorithm. The author [86] found that median vector have a much higher correlation

with the optimal one than the MV_0 for different test sequences. A brief description of PMVFAST is given below.

In additional to the set *V* defined in (2-21), two more motion vector predictors are involved in the PMVFAST. They are the median of the ROS motion vectors (MV_m) and motion vector of the collocated MB in the reference frame (MV_{col}). The PMVFAST computes the SAD of the MV_m, and the termination depends on the following criteria.

$$\text{Stop if } \begin{cases} MV_m = MV_{col}, \textit{and } SAD(MV_m) < SAD \text{ of collocated MB} \\ \textit{or} \\ SAD(MV_m) < 256 \end{cases}$$

After that, the PMVFAST computes the SAD of MV_m, MV_{col}, and the *MVi* in Set *V* and uses early elimination technique again by these criteria,

$$\text{Stop if } \begin{cases} MV \text{ of minimum } SAD\ (SAD_{\min}) = MV_{col}, \text{and } SAD_{\min} < SAD \text{ of collocated MB} \\ \textit{or} \\ SAD_{\min} < \text{an adaptive threshold, } ThresA \end{cases} .$$

Then the *MV* associated with SAD_{min} is used for searching with the search pattern of MVFAST. If $MV = MV_0$ or an adaptive threshold, *ThresB* greater than 1535, the large diamond search pattern is applied. On the hand, if the MV_i and MV_{col} are identical to each other, small diamond search is applied only once to find the best *MV*. Otherwise, the small diamond search of MVFAST is applied. Please refer to [121] for the definitions of the adaptive thresholds, *ThresA* and *ThresB*.

Part 7 of the MPEG-4 [121, 122] have already accepted the MVFAST and PMVFAST after extensive experiments. These two algorithms attain great speedup performance and maintain the best PSNR when comparing to different lossy algorithms. The PMVFAST is faster than MVFAST at the expense of higher hardware complexity. Both the MVFAST and PMVFAST apply the ‘stop when good enough’ spirit, and the spatial and temporal correlation between neighboring motion vectors to achieve such significant improvement.

2.3 *Transform coding techniques for intraframe and interframe*

Transform coding scheme has proven to be an effective technique for image compression and video coding. A transform operation can be considered as using a set of basis functions to synthesize an image given its corresponding transform coefficients. Hence, a forward transform decomposes the original data into transform coefficients for a given set of basis function. In terms of compression efficiency, the objective of a transform is to decorrelate the original signal, and this decorrelation generally results in the signal energy being redistributed among only a small set of transform coefficients. As a result, many coefficients can be discarded after a certain quantization scheme and for further encoding.

The most efficient transform that can maximize energy packing capability is the Karhunen-Loève Transform (KLT). Unfortunately, the KLT basis functions are source- or image-dependent and require an estimate of the image covariance function for basis functions generation. These drawbacks make the KLT less than ideal for image and video coding.

In the fields of image processing and video coding, the first-order Markov model is often used due to it simplicity for theoretical analysis. For the KLT, some researchers made use of the first-order Markov model to analyze the KLT. At this simplified situation, determination of the basis vectors of the KLT is not still data-dependent. Hence, this model can be used for transform comparison.

The Discrete Cosine Transform (DCT) is the one widely used in the modern video standards due to its fine decorrelation and energy compaction properties. In fact, one can show that, for the case of the first-order Markov source, the DCT is asymptotically equivalent to KLT as the adjacent element correlation coefficient tends to unity [112,113].

The analytical results show that the DCT is near optimal for intraframe coding. However, modern video codecs apply the DCT to frames in both intra- and inter-mode coding. It is well known that, after motion-compensation, the correlation between adjacent prediction errors is less than those in the original image [114]. This observation motivates different researchers to consider the role of the DCT in inter-mode coding. For this reason, using the simple first-order Markov model to describe the statistical properties of prediction errors is inaccurate and insufficient. Ref. [115] proposes a compound covariance model for an analysis of the motion-compensated frame difference. In [116], another compound covariance model was introduced to fit the empirical covariance sequence.

The above issues shall be discussed with the following organization. First-order Markov model plays an important role for still image and video sequence analysis. This model is included in Section 2.3.1. Section 2.3.2 review the basic concept of the optimum transform for coding, KLT. The relation between the KLT and the DCT will also be discussed. Section 2.3.3 expresses the transform coding scheme of modern video coding standards. This coding scheme involves the DCT for signal transform. It can be shown that this scheme is designed extremely suitable for intraframe coding. Utilizing the DCT for both intra- and inter-mode coding in the video codec is supported empirically. Lack of theoretical basis motivates the research in this problem. The analytical works for this purpose are illustrated in Section 2.3.4. Moreover, Section 2.3.4 also gives the conclusions drawn by these analytical models.

2.3.1 First order Markov model for Image Processing

The interdependence of data-source lies at the heart of any statistical data compression scheme, and thus the correlation properties of images are of considerable importance in the development of transform processing techniques.

By regarding a one-dimensional frame as a set of random variable denoted as, f_n, where n is index or position of a frame element, we may express its autocovariance matrix as,

$$COV(I)=\begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \sigma_{13}^2 & \cdots & \sigma_{1N}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \sigma_{23}^2 & \cdots & \sigma_{2N}^2 \\ \sigma_{31}^2 & \sigma_{32}^2 & \sigma_{33}^2 & \cdots & \sigma_{3N}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{N1}^2 & \sigma_{N2}^2 & \sigma_{N3}^2 & \cdots & \sigma_{NN}^2 \end{bmatrix} \quad \textbf{(2-22)}$$

where

$$\sigma_{mn}^2 = E\left[\left(f_m-\overline{f_m}\right)\left(f_n-\overline{f_n}\right)\right] \quad \textbf{(2-23)}$$

$\overline{f_m}$ is the mean of random variable, f_m

If we assume the data is wide-sense stationary (WSS), the mean and variance can be treated as constant and the covariance is a function of relative displacement only. Hence, we do not need to consider the direction of displacement, d, between two elements in (2-23), i.e. $\sigma_{mn}^2=\sigma_{|d|}^2$ where $d=m-n$. Another important approximation for an image field is that we frequently assume the value of $\sigma_{|d|}^2$ is related to the correlation coefficient at distance one by $\sigma_{|d|}^2 \propto \rho_1^{|d|}$ and $\rho_1=\rho$. After normalization, we may rewrite (2-22) to

$$COV(I)=\begin{bmatrix} 1 & \rho & \rho^{|2|} & \cdots & \rho^{|N-1|} \\ \rho & 1 & \rho & \cdots & \rho^{|N-2|} \\ \rho^{|2|} & \rho & 1 & \cdots & \rho^{|N-3|} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{|N-1|} & \rho^{|N-2|} & \rho^{|N-3|} & \cdots & 1 \end{bmatrix} \quad \textbf{(2-24)}$$

where $\rho=\dfrac{\sigma_{|1|}^2}{\sigma_{|0|}^2}$, usually named as correlation coefficient.

It is the correlation (normalized covariance) matrix of the stationary first-order Markov process.

2.3.2 The Karhunen-Loève Transform

The Karhunen-Loève Transform is a decorrelating transform for wide sense stationary processes when the second order statistics (covariances) are known. For the random variables F_n, its autocovariance matrix defined in (2-22) is symmetric and nonnegative definite. All of its eigenvalues are greater or equal to zero. Let T be a $N \times N$ transform matrix which is unitary. The transformed F is given by $Y=TF$. The autocovariance of Y is

$$\begin{aligned} COV(Y) &= E[YY'] = E[TFF'T'] = TE[FF']T \\ &= TCOV(F)T \end{aligned} \quad \textbf{(2-25)}$$

where "' " is stand for a transpose.

For efficient compression, we would like to obtain uncorrelated transform coefficients, such that their covariance has to be zero. That means the $COV(Y)$ is diagonal after an optimum transform. Let v_n be the eigenvector of $COV(F)$ corresponding to the eigenvalue λ_n. With the following ordering of the λ_n,

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N .$$

We obtain a complete set of orthonormal eigenvectors. These eigenvectors form the rows of the transform matrix required for the KLT.

$$T = [v_1, v_2, \cdots, v_N]' \quad \textbf{(2-26)}$$

To show that the KLT has the optimum energy packing capability (i.e. minimum mean square error performance can be obtained by retain only most important M out of the N transform coefficients), we may represent F by using only M preceding basis vectors instead of totally N. By minimizing the resulting error energy, one will find that the KLT achieves the optimal performance. Detailed derivation please refers to [112, 113].

The above description apparently shows that the KLT is source-dependent. This problem makes it not ideal for compression usage. For theoretical analysis, researchers therefore take the alternative approach and employ the first-order Markov model. In this case, the solutions of these basis vectors are expressed involving the eigenvalues, $\lambda_n(\rho,w_n)$, where w_m are real positive roots of a transcendental equation. Consequently, it can be illustrated that the DCT is asymptotically equivalent to KLT for the Markov model as ρ tends to unity [112, 113].

2.3.3 Discrete Cosine Transform for Video Coding

The separable 2-dimensional Discrete Cosine Transform used in the industrial video standard is defined as,

$$Y(u,v)=\frac{2}{N}C(u)C(v)\sum_{i}^{N-1}\sum_{j}^{N-1}f(i,j)\cos\frac{(2i+1)u\pi}{2N}\cos\frac{(2j+1)v\pi}{2N} \quad \textbf{(2-27)}$$

where i, j are spatial coordinates in a *block*.

u, v are coordinates in the transform domain.

$$C(u),C(v)=\begin{cases}\frac{1}{\sqrt{2}} & \text{for } u,v=0\\ 1 & \text{otherwise}\end{cases}$$

The most useful property of DCT coded blocks is that the coefficients can be coarsely quantized without seriously affecting the quality of the image that results from an inverse DCT of the quantized coefficients. In MPEG standards, the lowest spatial frequency coefficients, which generally possess the greatest energy, are quantized most finely, and the highest spatial frequency coefficients are quantized coarsely. The quantized coefficients are stored in a register according to a predefined scanning order. MPEG-4 defines a zigzag scan and other two alternative scans which are depicted in Figure 2-20.

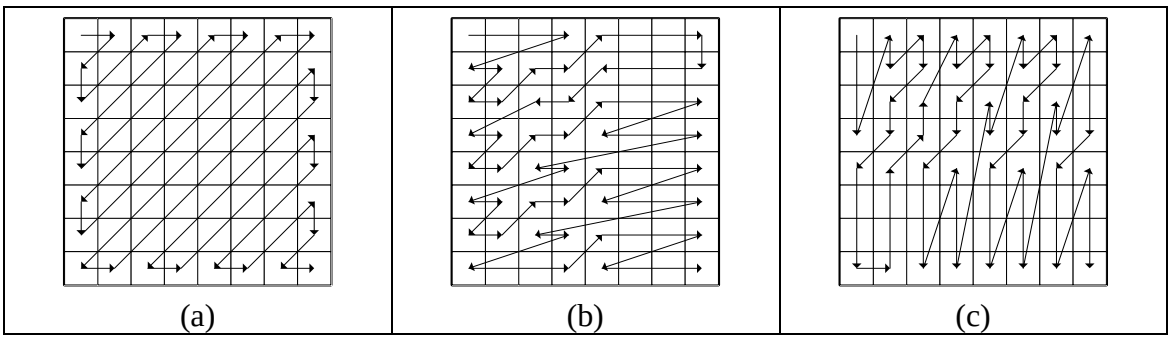


Figure 2-20. (a) Zigzag scan, (b) Alternate-Horizontal scan, and (c) Alternate-Vertical scan.

The scanned coefficients are then entropy coded. It is well know that natural images are dominated by low frequency components. These scanning orders favor the coding of natural images. Because the quantized high frequency coefficients often equal zero, and thus this long tailing zeros do not need to be coded.

The above coding scheme is well designed for intraframe coding. Some researchers claimed that most of the performance gain of a coding scheme is obtained by carefully designing quantizers that are tailored to the transform structure [124]. Nevertheless, this benefit is not obvious for coding of inter-mode. First, the lower correlation between adjacent prediction errors predicted that among all famous orthogonal transform, such as the Discrete Sine Transform (DST), Discrete Fourier Transform (DFT) and DCT etc, the DCT may not again be the one closest to the KLT. Second, it is observed that block-based compensation typically results in a peaky distribution of the errors, with high residual concentration at block edges and image edges [117, 118]. It leads to a scattering of the DCT coefficients and makes the scanning order inappropriate. However, experimental results support using the DCT for both intra- and inter-mode coding. It is because the difference between other transform and the DCT is so marginal for the prediction error coding [115, 116]. It would be simpler to employ a single transform in a practical coder.

2.3.4 Theoretical analysis about the using of DCT for inter-mode coding

Analytical investigation of transform coding algorithms requires a simple but sufficiently accurate signal-source description for analytical treatment and to reflect the practical signal characteristics. The first-order Markov-process satisfies these criteria, and thus is widely used for still image analysis. However, statistical behaviors of the motion-compensated errors is very different from that of natural image. The results predicted from the simple Markov model did not persuade other researchers [115] of its accuracy. Beside, they are unsatisfying the lack of theoretical basis about the suitability of utilizing the DCT for encoding the prediction error. They proposed a compound covariance model for the motion-compensated frame difference and demonstrated that the DCT performs nearly optimally as the intraframe coding. A difference covariance model, which takes overlapped block motion compensation into account, was derived in [116] empirically.

To derive the compound covariance model [115], the authors started from the definition of autocorrelation function. For the motion-compensated prediction error, its autocorrelation function is given by

$$R_e(I,J) = E\left[\left\{f_{t+1}(i+u,j+v)-\hat{f}_{t+1}(i+u,j+v)\right\}\times \left\{f_{t+1}(i+u+I,j+v+J)-\hat{f}_{t+1}(i+u+I,j+v+J)\right\}\right] \quad \textbf{(2-28)}$$

where $f_{t+1}(\cdot,\cdot)$ is an incoming frame.

$\hat{f}_{t+1}(\cdot,\cdot)$ is a predicted frame with motion vector (*u*,*v*).

Due to the inaccuracy of motion estimation, by defining the true motion vector as $(\Delta i,\Delta j)$, error of the quantization of the motion vector to an integer pixel unit is known as

$$(\delta i,\delta j)=(\Delta i+u,\Delta j-v)$$

The author assumed that δi and δj is a pair of independent identical random variables with a uniform distribution in the interval [$-a$, a]. The parameter a represents inaccuracy of the motion estimation, with a typical value of 0.5.

Hence, it can be shown [115] that the variance-normalized autocorrelation function, $C_e(I,J)$, of the motion-compensated frame difference (MCFD) can be expressed as

$$C_e(I,J) = \frac{\rho^I \rho^J - \left(\frac{1}{2a\ln\rho}\right)^2 \left(\rho^{I+a} - \rho^{I-a}\right)\left(\rho^{J+a} - \rho^{J-a}\right)}{1 - \left(\frac{\rho^a - 1}{\ln \rho^a}\right)^2} \qquad \textbf{(2-29)}$$

when $0 < a \le I,J$

where ρ is correlation coefficient

To simplify the problem, the author further assumed that the autocorrelation function for the MCFD is separable. For instance, along the x-axis, the autocorrelation function is derived as

$$\begin{aligned} C_e(I) &= \frac{\rho^I - \left(\frac{1}{2a\ln\rho}\right)^2 \left(\rho^{I+a} - \rho^{I-a}\right)\left(2\rho^a - 2\right)}{1 - \left(\frac{\rho^a - 1}{\ln \rho^a}\right)^2} \\ &= \rho^I A(a,\rho) \end{aligned} \qquad \textbf{(2-30)}$$

$$\text{where} \quad A(a,\rho) = \frac{1 - \left(\frac{1}{2a\ln\rho}\right)^2 \left(\rho^a - \rho^{-a}\right)\left(2\rho^a - 2\right)}{1 - \left(\frac{\rho^a - 1}{\ln \rho^a}\right)^2} \le 1 \qquad \textbf{(2-31)}$$

when $0 < a \le I$

The value of function $A(a, \rho)$ is very close to 0.5 over a wide range of values by a and ρ. The 2-dimensional separable autocorrelation function for the MCFD is thus given by

$$C_e(I,J) = C_e(I)C_e(J) \tag{2-32}$$

The aim of transform coding is to decorrelate the signals and thereby compress the bitrate further. The heart of the KLT is the autocorrelation function of a data-source. The authors used the proposed model to investigate transform gains of different orthogonal transform coding, including the KLT, DCT, Discrete Fourier Transform (DFT) and Discrete Sine Transform (DST) for the MCFD signals. Considering the MCFD signals in a single direction with $a \leq 1$, its normalized autocorrelation matrix in Toeplitz form is shown as below.

$$COV(I) = \begin{bmatrix} 1 & C_e(1) & C_e(2) & \cdots & C_e(N-1) \\ C_e(1) & 1 & C_e(1) & \cdots & C_e(N-2) \\ C_e(2) & C_e(1) & 1 & \cdots & C_e(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_e(N-1) & C_e(N-2) & C_e(N-3) & \cdots & 1 \end{bmatrix} \tag{2-33}$$

Replacing $C_e(I)$ by $\rho^I A(a, \rho)$ as derived in (2-30), we have

$$COV = \begin{bmatrix} 1 & \rho A(a,\rho) & \rho^2 A(a,\rho) & \cdots & \rho^{N-1} A(a,\rho) \\ \rho A(a,\rho) & 1 & \rho A(a,\rho) & \cdots & \rho^{N-2} A(a,\rho) \\ \rho^2 A(a,\rho) & \rho A(a,\rho) & 1 & \cdots & \rho^{N-3} A(a,\rho) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{N-1} A(a,\rho) & \rho^{N-2} A(a,\rho) & \rho^{N-3} A(a,\rho) & \cdots & 1 \end{bmatrix} \tag{2-34}$$

(2-34) was used to evaluate the transform gains of the four transform coding. The results proved that the DCT appears to be closest to the KLT. The difference between the DCT and other transforms are only marginal. In fact, the $COV(I)$ in (2-34) can be separated into two parts,

$$\begin{aligned} COV &= R_{e1} + R_{e2} \\ &= A(a,\rho) \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix} + [1 - A(a,\rho)][I] \end{aligned} \tag{2-35}$$

where $[I]$ is an identity matrix

The first matrix R_{e1} in (2-35) represents the autocorrelation matrix of a first-order Markov process e1 with the correlation coefficient ρ. The second one is a diagonal matrix. It reflects that the second component e2 is white noise with a flat power spectrum. When we employ the KLT to (2-34), we have

$$\begin{aligned} T \cdot COV \cdot T' &= T(R_{e1} + R_{e2})T' \\ &= T \cdot R_{e1} \cdot T' + R_{e2} \end{aligned}$$

Consequently, the author concluded that the KLT for the first-order Markov model is also the KLT for COV in (2-34) and the DCT remains a near optimal transform for MCFD signal.

However, other researchers [116] studied the covariance of inter-coded blocks of four MPEG test sequences. According to their experimental results, they claimed that the covariance model (2-34) do not appropriately fit the empirical covariance results. Therefore, they proposed two empirical models, which are autoregressive covariance models and a compound covariance model, to fit their requirements.

Autoregressive (AR) process may be used as a parametric description of a random signal. An *p*-th order autoregressive process, AR(*p*), is defined by the following stochastic difference equation

$$s_I + \sum_{m=1}^{p} a_m s_{I-m} = w_I \qquad \textbf{(2-36)}$$

where w_I denotes a write noise process with zero mean and variance σ_w^2.

An AR(1) process represents the first-order Markov process by the difference equation

$$s_I + a_1 X_{I-1} = \delta(I) \qquad \textbf{(2-37)}$$

where $a_1 = -\rho$.

$\delta(I)$ is a delta function.

The covariance C_s (I) of a zero-mean WSS process X_I satisfies the Yule-Walker equations

$$C_s(I)+\sum_{m=1}^{p} a_m C_s(I-m)=\sigma_w^2\delta(I) \qquad ; \qquad I\geq 0 \tag{2-38}$$

The authors proposed an autoregressive covariance models with $p \leq 4$ using (2-38). The corresponding parameter a_m for (2-38) are listed in the Table 2-1.

Table 2-1. AR(*p*) models for parametric description of the empirical covariance of MCFD [116].

p	a_1	a_2	a_3	a_4
1	-0.4683	-	-	-
2	-0.5080	0.0848	-	-
3	-0.5205	0.1595	-0.1471	-
4	-0.5217	0.1608	-0.1514	0.0081

They also pointed out that the compound covariance model (2-34) is equivalent to the equation,

$$C_e(I)=A\rho^{|I|}+(1-A)\delta(I) \qquad ; \qquad I\geq 0 \tag{2-39}$$

where $A= A(a, \rho) \approx 0.5$, with a and ρ is equal to 0.5 and 0.95 respectively.

However, this model deviates significantly from their experimental results. Hence, they proposed another compound covariance model

$$C_e(I)=c\rho_0^{|I|}+(1-c)\rho_1^{|I|^2} \tag{2-40}$$

where c , ρ_0 and ρ_1 are model parameters.

The model parameters, c=0.17, ρ_0=0.91 and ρ_1=0.38, were chosen to fit the empirical covariance in the l_1-norm sense.

This compound model was then used for transform gain comparison. For a block size equal to 8, the coding gain of the DST and KLT is negligible comparing to that of the DCT. This negligible loss and the superiority of the DCT for intraframe transform coding explain the suitability of the DCT for both intra- and inter-mode transform coding. With respect to the energy compaction performance for nonzero-mean signals, a further advantage of the DCT compared to the DST is that the DC component of the

signal is compactly represented by the first DCT coefficient whereas the DC component is distributed over the even indexed DST coefficients.

The close fit to empirical results and simplicity make the compound model (2-40) very suitable for the analysis of block-based MCFD signal. On the other hand, this model is pure empirical and the physical mean behind the model is still mystery to us. The using of this model for other analytical purposes is so limited.

Chapter 3. Fast motion estimation of arbitrarily shaped video objects in MPEG-4

3.1 *Introduction*

MPEG-4 is an international standard which provides a coding scheme for arbitrarily shaped video objects (VOs) [5, 120]. Each VO is composed of its temporal instances, video object planes (VOPs), which is the central concept of MPEG-4 video. Block-based motion estimation is also used for exploiting temporal redundancies in arbitrarily shaped video objects, which is computationally the most demanding part within the MPEG-4 standard. The support of arbitrarily shaped video objects makes most of the existing fast motion estimation algorithms unsuitable.

In the past, much work such as the Motion Vector Field Adaptive Search Technique (MVFAST) in the Part 7 of MPEG-4 [122], the Diamond Search (DS) [80] and many other searching techniques [8, 69, 73-108] were reported for reducing the complexity of motion estimation. It is known that motion estimation remains as a very computationally intensive step in MPEG-4 arbitrarily shaped VOs encoding [104, 127]. Hence, It is highly desirable to reduce the computational requirement of motion estimation. In MPEG-4, motion estimation is performed on two kinds of macroblocks: the boundary macroblock (MB) and the opaque MB. Since the motion activities in opaque MBs are highly correlated with the neighboring boundary MBs, a new priority search algorithm (PSA) for motion estimation is proposed in this chapter, which performs motion estimation on all boundary macroblocks first in contrast to the conventional raster-scanning approach. This search strategy works well if the motion vectors in the boundary macroblocks truly represent the moving video object. Consequently, the full search algorithm (FSA) is applied to the boundary MB in order to

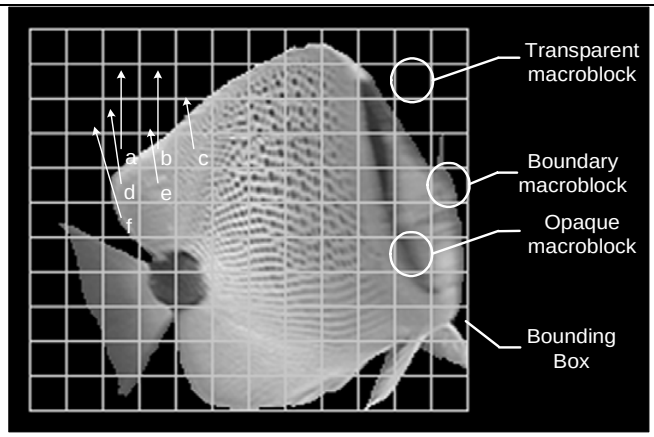
ensure its accuracy. However, the computational burden of motion estimation of the boundary MBs must be reduced. Fast search algorithms proposed in the past tend to reduce the amount of computation by limiting the number of locations to be searched. Nearly all of these algorithms assume that the distortion function increases monotonically as the search location moves away from the global minimum. Unfortunately, this is usually not true in boundary MBs. We can reasonably assume that it is monotonic in a small neighborhood around the global minimum. Consequently, one simple strategy, but perhaps the most efficient and reliable one, is to place the checking point as close as possible to the global minimum. In this charter, we also propose a fast search algorithm, which incorporates the binary alpha-plane to predict accurately the motion vectors of boundary MBs such that these motion vectors can be used in the PSA. Experimental results show that, when compared to conventional methods, our approach requires low computational complexity and provides significant improvement in terms of accuracy in motion-compensated video object planes. The proposed algorithm incorporates the binary alpha-plane to accurately predict the motion vectors of boundary MBs such that the motion-compensated VOPs are tied more closely to the video object. Besides, these accurate motion vectors can be used to develop an efficient motion estimation algorithm for the remaining opaque MB.

In Section 3.2, the characteristics of different types of macroblocks in the bounding box of a VOP are studied. According to these characteristics, a new priority search algorithm (PSA) is introduced. This section also shows that the accuracy of the motion vectors of boundary macroblocks is critical in the performance of the proposed PSA. In section 3.3, we present an in-depth study of the correlation between the SAD error surface and shape information of alpha-plane, and then formulate a reliable search algorithm for boundary macroblock. Section 3.4 describes the details of the proposed

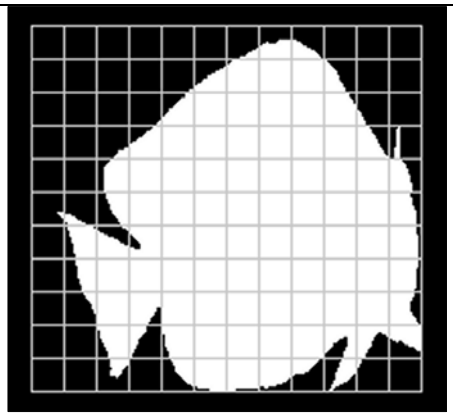
PSA with the new search algorithm for boundary macroblock. Experimental results are then presented in Section 3.5. Finally a conclusion is drawn in Section 3.6.

3.2 *Priority Search Algorithm (PSA) on Arbitrarily Shaped Video Objects*

A VOP can be fully described by texture variations and shape representation, as shown in Figure 3-1. The shape information is represented as a binary alpha-plane. The alpha-plane contains the information to identify pixels which are inside an object (value of alpha-plane = 1), and pixels which are outside the object (value of alpha-plane = 0), as depicted in Figure 3-1(b). For efficient block motion estimation, it is important to know the characteristics of different types of macroblocks. MPEG-4 defines an arbitrarily shaped VOP by means of a bounding box, in which details of the definition of a bounding box have been introduced in Charter 2, Section 2.1.2.1. In Figure 3-1, three types of macroblocks exist in the bounding box of the VOP. Their corresponding motion search strategies are summarized as follows.



(a)



(b)

Figure 3-1. Representation of the VOP. (a) Image of original "Bream" VOP. (b) Binary alpha-plane of "Bream" VOP.

- Transparent macroblocks: They are not coded and recovered at the decoder side from the shape information. Thus no motion search is required.
- Boundary macroblocks: This type of MB partially includes object pixels, and polygon matching is employed to adopt arbitrarily shaped moving video objects. The human visual system is very sensitive to poor motion-compensated prediction along the moving contours of video objects, which are located on boundary macroblocks. A correct motion estimation of boundary macroblocks is critical to the development of an efficient motion estimation algorithm for arbitrarily shaped moving video objects.
- Opaque macroblocks: The opaque MB is coded using the conventional block matching motion estimation algorithm. Usually the motion activities within the video object are consistent; hence the motion activities in these MBs are highly correlated with the neighboring boundary MBs provided that the motion vectors in the boundary MBs truly represent the moving video object. For example, the motion vector of the opaque macroblock, MB_e, in Figure 3-1 is correlated highly with those of the boundary macroblocks, MB_a, MB_b, MB_c, MB_d and MB_f. Thus, the motion vectors of the boundary MBs can play a significant role in motion estimation for the opaque MB.

The characteristics of different types of MBs described above inspired to develop a new priority search algorithm (PSA) which performs motion estimation on all boundary MBs first within the bounding box of a VOP in contrast to the conventional raster-scanning approach (scanning MBs in the order of top-to-bottom and left-to-right). The idea behind the new search strategy is that the opaque MBs which are inside of moving video objects are correlated highly with the moving boundary MBs. For each opaque MB, if all motion vectors of its neighboring boundary MBs have already been computed, the current opaque MB can take the best-matched one among all its neighboring motion vectors as the initial centre and employ a conventional fast block matching algorithm such as the Diamond Search (DS) [80] to compute its motion vector for a reduction of the computational complexity. It is likely that the global minimum can be found by a local search such as using the DS if the initial centre is close enough to the global minimum. Hence, the computations for finding the motion vectors of opaque MBs will be postponed until all motion vectors of the boundary MBs are available. The advantage of this new search strategy is that it avoids unnecessary computations of the opaque MB so that the motion search can be conducted more efficiently.

In order to ensure the accuracy of the motion vectors of the boundary MBs, a full search algorithm (FSA) which evaluates the SAD at all possible locations of the search window is employed. By using the accurate motion vectors of the boundary MBs, the motion vectors of the opaque MBs can be found by the DS. This PSA produces smaller motion compensation errors, and has a lower computational complexity as compared with the traditional raster-scanning motion estimation. Figure 3-2 depicts the performance of the PSA in encoding the “Bream” video object. The figure plots the mean square errors (MSE) between the original VOP and the motion-compensated VOP of the PSA with the FSA performing on boundary macroblocks and the DS on opaque

MBs using the best-matched motion vector among all its neighboring motion vectors as the initial centre (PSA(FSA+DS)) and compares the results with those of the full search algorithm (FSA). The results show that the MSE performance of the PSA(FSA+DS) is very close to the FSA. Details on the simulation environment are described in Section 3.5.

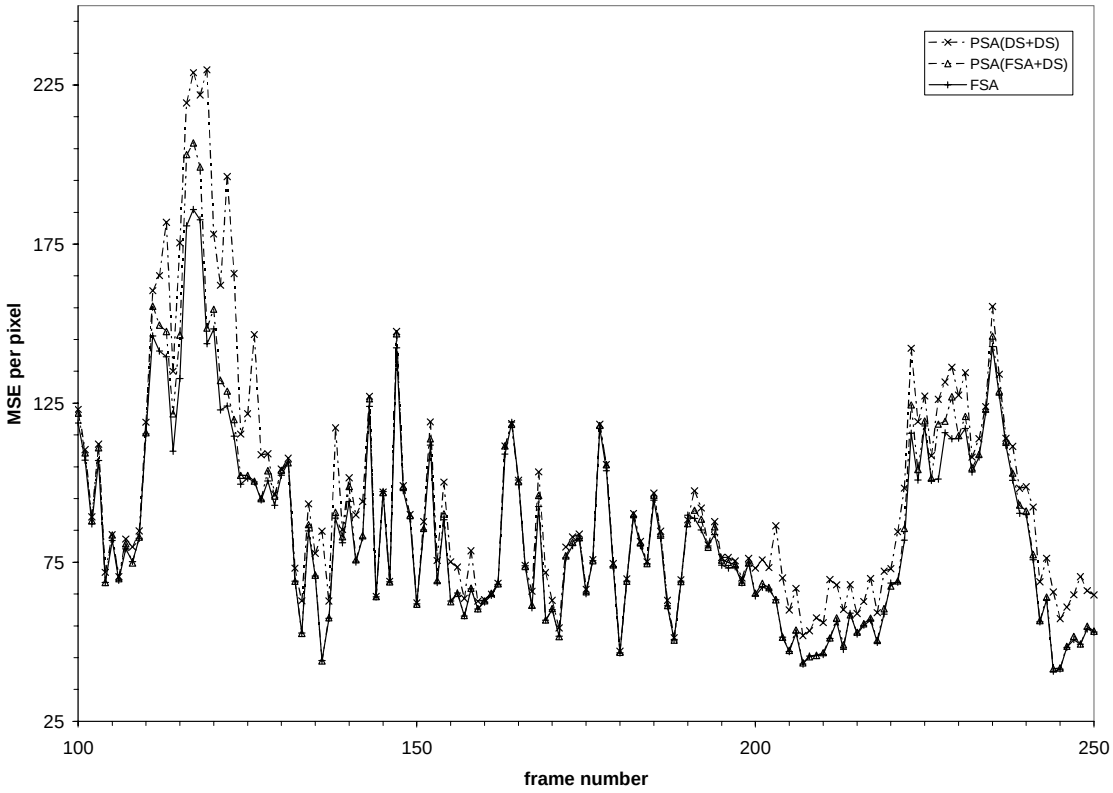


Figure 3-2. MSE performance of PSA for "Bream" video object.

As mentioned above, the accuracy of the motion information of boundary macroblocks is critical to PSA(FSA+DS). Consequently, the FSA is used to ensure its accuracy. Overall speaking, over 90% of the total search points required that the whole motion estimation process are performed for the boundary macroblocks. In order to increase the flexibility and practicability of the PSA(FSA+DS), the computational burden of the motion estimation of the boundary macroblocks must be reduced. In Figure 3-2, we also analyze the MSE performance of the PSA by using the DS for both the boundary macroblocks and the opaque macroblocks (PSA(DS+DS)), in which the

best-matched motion vector among all of its neighboring motion vectors obtained by the DS act as an initial centre for performing the DS on opaque MBs. Figure 3-2 shows that there is a big prediction error in PSA(DS+DS) as compared with that of the FSA. This is because the probability of encountering the local minimum problem is more often in the boundary MB. This phenomenon could achieve our desire to develop a fast and efficient search algorithm for the boundary MB, which is described in the following section.

3.3 *Binary Alpha-plane Assisted Search algorithm (BAAS) of the Boundary Macroblock*

In traditional block based motion estimations, the error measure criterion such as the sum of absolute differences (SAD) for block matching motion estimation is calculated using all pixels. This conventional block matching approach is applied to the opaque MB. But, on the boundary MB, the binary alpha-plane for the VOP is used to exclude the pixels of the MB that are outside of the VOP. This forms a polygon matching instead of block matching for the motion estimation of the boundary MB. The *SAD* matching criterion used in the boundary MBs is a measure of the error between a MB of the present frame and a MB with displacement *(u,v)* of the previous frame with a block size of $N \times N$ pixels, and is calculated only on the pixels inside the object, and can be written as

$$SAD_{N\times N}(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} | f_t(i,j) - f_{t-1}(i+u, j+v) | \cdot (\underset{n}{alpha} \neq 0) \qquad \textbf{(3-1)}$$

where $f_t(i,j)$ is the intensity of the pixel at location (i,j) within the macroblock in the t^{th} frame.

From (3-1), the maximum motion in the vertical and horizontal directions is $\pm D$. There are thus $(2D+1)^2$ candidates in total to be checked if the full search algorithm is

used, each corresponding to a checking point in the search window. The SAD values that result from these checking points form an error surface.

The statistical behavior of an error surface has a significant impact on the performance of a fast search algorithm. For the surface of the boundary MB as shown in Figure 3-3(a), it contains a large number of local minima due to the repetitive padding [52] of an arbitrarily shaped video object. Almost all conventional fast algorithms have assumed explicitly or implicitly [8] that the error surface is unimodal over the search window. As a consequence, it is unlikely that conventional fast search algorithms would converge to the global minimum when performing motion estimations on the boundary MB. In other words, the search would easily be trapped into a local minimum.

Without loss of generality, we employ the Diamond Search algorithm (DS) as an example to illustrate the problem of conventional fast search algorithms on the boundary macroblock. Let us recall that, the original diamond pattern as shown in Figure 2-17(b) is a basic searching pattern of the DS. It consists of nine candidate search points. For each of the nine candidate search points, the *SAD* is computed. If the minimum *SAD* is found at the centre of the diamond pattern, a shrunk diamond pattern as shown in Figure 2-17(c) is used with the same centre and the candidate point that gives the lowest *SAD* is chosen as the estimated motion vector. Otherwise, the minimum *SAD* point in the previous search step could be regarded as a new centre of the original diamond pattern for a next search step. Of course, the DS relies on the assumption that the *SAD* measure decreases monotonically as the search point moves closer to the optimal point. It can easily be trapped into the local minimum in cases where the error surface looks like the one in Figure 3-3(a).

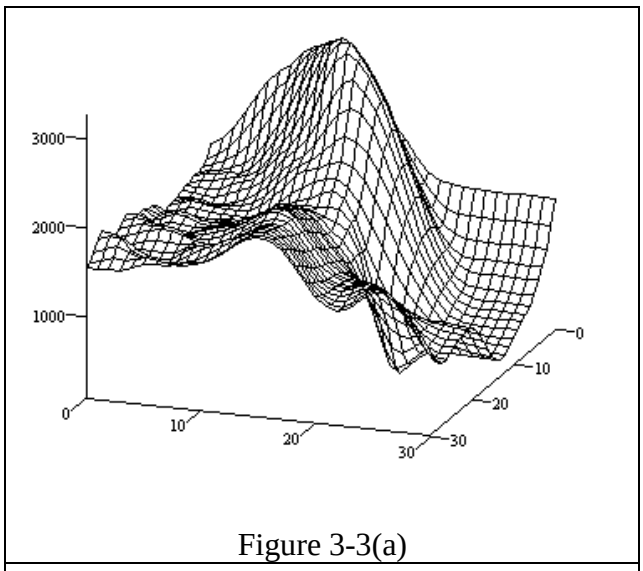


Figure 3-3(a)

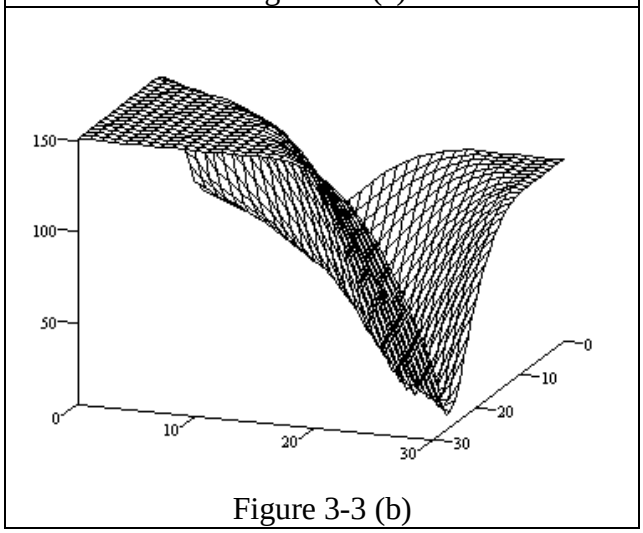


Figure 3-3 (b)

Figure 3-3. The relationship between (a) the error surface and (b) the proposed BAMS surface of the boundary macroblock.

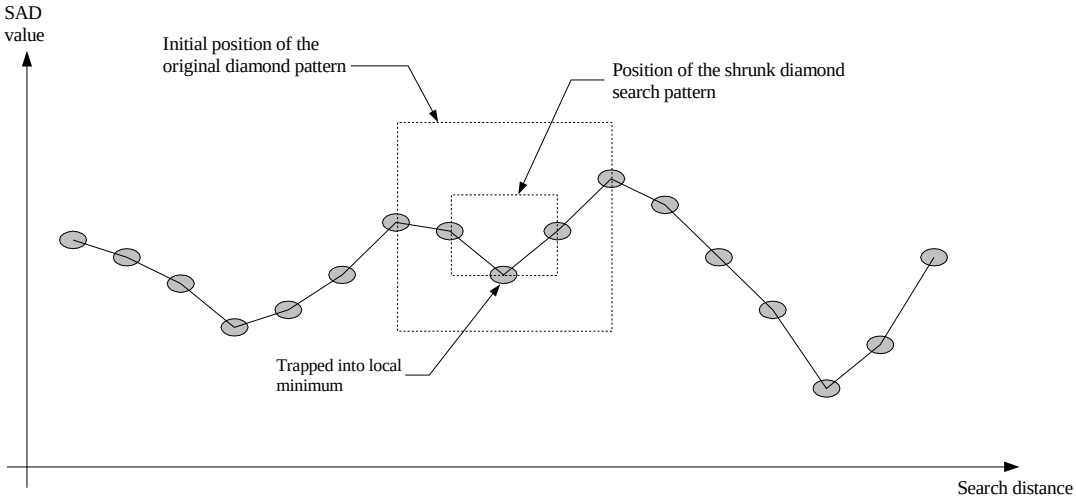


Figure 3-4 (a)

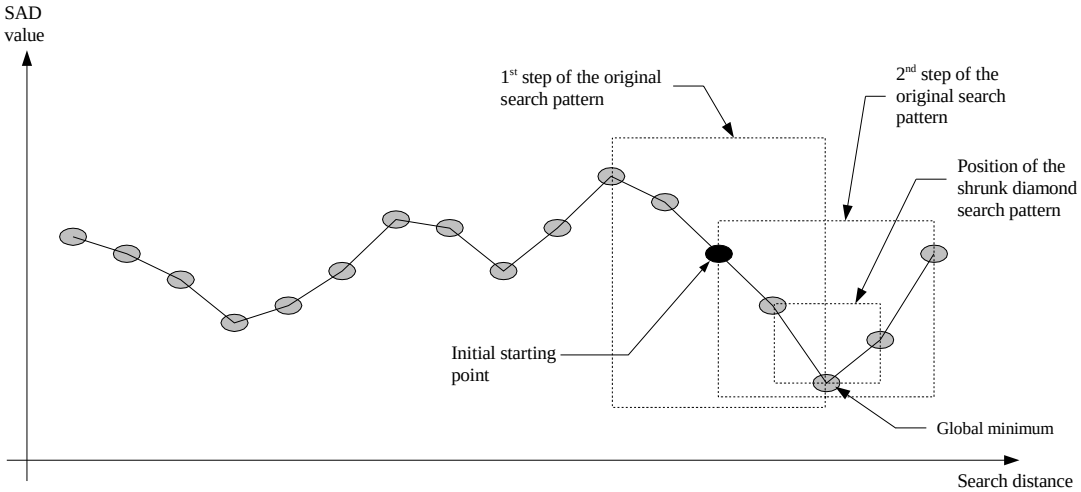


Figure 3-4 (b)

Figure 3-4. A nonumimodal error surface tested by a checking block. (a) The checking block starts at the origin and a false checking results, hence a local minimum is found. (b) If the initial checking block is close enough to the global minimum, the global minimum can be successfully found.

Let us use Figure 3-4 to give a clearer account of this phenomenon. Figure 3-4(a) shows a nonunimodal surface of a boundary macroblock. The initial step of the DS starts at the origin of the error surface, the centre point in the checking block wins. It stops the search process and a local minimum will be found. However, it is seen that the global minimum is located at the far side of the winning point and the SAD value of the winning point is significantly larger than that of the global minimum. This will affect the accuracy of the motion vector of the boundary macroblock.

Despite the error surface exhibiting uncertainties in a large spatial scale, we can reasonably assume that it is monotonic in a small neighborhood around the global minimum. In the existence of local minima, one simple strategy, but perhaps the most efficient and reliable, is to place the starting diamond pattern as close as possible to the global minimum, as depicted in Figure 3-4(b). If the initial diamond pattern is close enough to the global minimum, it will likely be able to find it through a local search. One possible solution to prevent the problem of trapping to a local minimum is to test more starting points which spread across the search window.

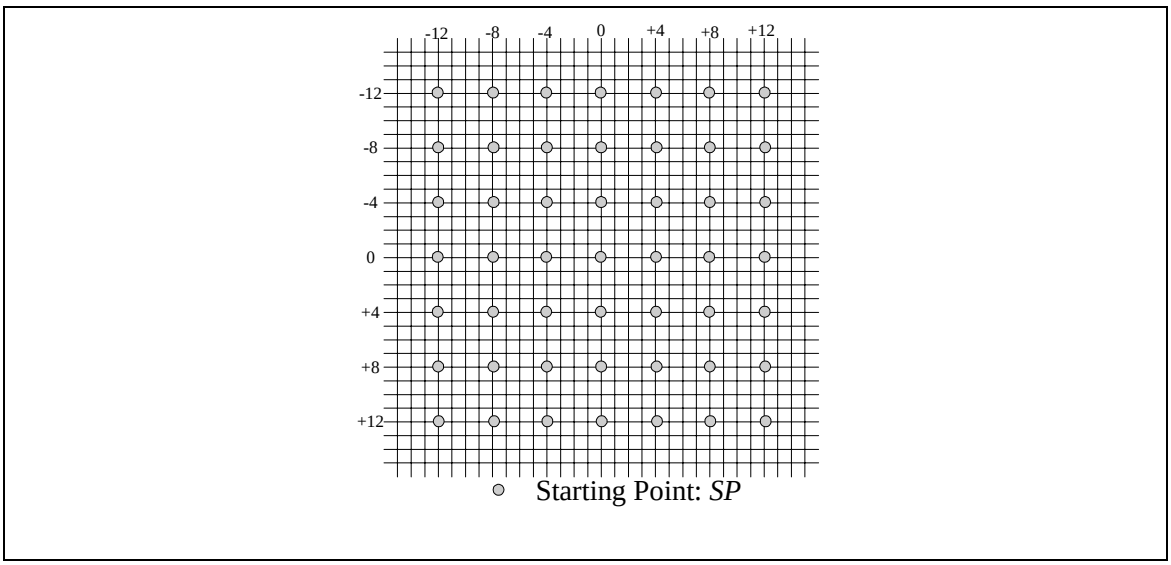


Figure 3-5. Regular SPP

Figure 3-5 shows one of the starting point patterns (SPP) in which the starting points (SP) are distributed evenly over the search window. However, it is inefficient to use so many of the starting points in this regular SPP. It is obvious that if the number of starting points is reduced as much as possible and the starting point is as close as possible to the true motion vector, the search algorithm becomes efficient. Hence, we have to adjust the regular SPP so that the limited SPs have a higher chance of catching the global minimum. In this study, we try to employ the binary alpha-plane of an arbitrarily shaped video object for the adjustment of the regular SPP. It generally includes a matching process for tracking a polygon shape in the separated VOPs and we will refer it to as a Binary Alpha-plane Assisted Search algorithm (BAAS). The proposed algorithm first estimates an initial probability of being the global minimum for each possible matching pair between the current boundary macroblock and the macroblock at the regular SPP which is updated based on the shape information. In the following, we highlight the main steps of our BAAS.

- Step 1: Adjustment of the regular SPP

In order to evaluate the similarity of arbitrary shapes between two macroblocks, we define a cost function which should have a small value or be zero only if the two

macroblocks have identical shapes. A binary alpha-plane matching score (BAMS) is introduced to measure the shape similarity between a boundary macroblock of the present VOP and a macroblock with displacement (*u*,*v*) of the previous VOP with a block size of *N*×*N* pixels.

$$BAMS(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} BA_t(i,j) \otimes BA_{t-1}(i+u, j+v) \quad \textbf{(3-2)}$$

where $BA_n(\cdot,\cdot)$ and $BA_{n-1}(\cdot,\cdot)$ are the values of the binary alpha-plane of the current boundary macroblock at the t^{th} VOP and the reference macroblock at the $(t\text{-}1)^{th}$ VOP that are to be compared respectively, and ⊗ denotes the exclusive-or operation.

The adjustment of the regular SPP is based on the measure of how high the probability of it being the global minimum of each possible matching pair between the current target MB and the MB at the regular SPP. The *BAMS* is used to determine if the polygon shapes in the two boundary macroblocks are similar. Hence, the macroblock in the regular SPP has a high probability of being closest to the global minimum. Figure 3-3 plots the error surfaces and the *BAMS* surfaces of a boundary MB. We have found that the correlation between these two surfaces is very high and it further ensures that the motion search algorithm can be guided by the *BAMS*. Thus a macroblock in the regular SPP whose BAMS is less than a pre-defined threshold, T_{BAMS}, is good enough to be an interested SP. In other words, this SP is reserved in the updated SPP. In order to normalize thresholding, the T_{BAMS} must be proportional to the number of opaque pixels of the current macroblock. That is,

$$T_{BAMS} = \alpha \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} BA_n(i,j) \quad \textbf{(3-3)}$$

where α is a proportional constant.

- Step 2: The formation of the final SPP

In order to reduce further the computational complexity, the updated SPP can be refined by using the image intensity. A simple way is to employ the *SAD* polygon matching criterion. Selection of the best-matched SP as compared to other SPs in the updated SPP is based upon the SAD values, and it is defined as

$$G_k = SAD_k - SAD_{\text{min\_in\_updated\_SPP}} \quad \textbf{(3-4)}$$

where k means to cover all selected SPs of the updated SPP, except the SP with the smallest value of SAD in the updated SPP, $SAD_{\text{min\_in\_updated\_SPP}}$ is the smallest value in the updated SPP and SAD_k is the value of the SAD from the SP in the updated SPP. Two major criteria are used to form the final SPP. First, the SP with the smallest value must be reserved as the final SPP. Second, the value of G_k is used to establish the final SPP. If the value of G_k is small enough (smaller than $\beta \times SAD_{\text{min\_in\_updated\_SPP}}$, where β is another constant of proportionality), it implies that the probability of this SP being the global minimum is high. In other words, this SP must be included in the final SPP; otherwise, this SP is eliminated from the updated SPP. After examining all the SPs in the updated SPP, the final SPP is formed.

- Step 3: Motion vector estimation using the final SSP as the starting point

After the establishment of the final SPP, each SP in the final SPP serves as the starting point for one of the conventional fast searching algorithms, such as the DS. Finally, a search is conducted to find the minimum value of the SAD.

3.4 *The proposed PSA with the BAAS performing on the Boundary Macroblock, PSA(BAAS+DS)*

In this section, a fast motion estimation algorithm for arbitrarily shaped video objects is described and this new algorithm employs both PSA and BAAS as mentioned above. The proposed algorithm is mainly divided into two stages. The first stage involves a fast motion estimation of boundary MBs with the help of the alpha-plane

(BAAS) while the second stage estimates the motion of opaque macroblocks through motion vectors of its neighboring boundary MBs (PSA). In other words, the boundary MBs employ the BAAS to ensure accuracy of their motion vectors. While the motion vectors of opaque MBs can be found by the DS through the help of the accurate motion vectors of boundary macroblocks. Some details of the proposed algorithm are given below.

1. To estimate the motion vectors of boundary macroblocks: By using the binary alpha-plane, the macroblocks of a VOP can be easily classified into three types: transparent macroblocks, boundary macroblocks, and opaque macroblocks. Only motion estimation of the boundary macroblocks is handled in this stage by using the BAAS as mentioned in Section 3.3, and motion estimation of opaque macroblocks is postponed to the second stage.
2. To estimate the motion vectors of opaque macroblocks: The opaque MBs inside a video object correlate highly with the neighboring MBs. Thus, if there are neighboring MBs for which the motion vectors have already been computed, the current MB will select one of these motion vectors as the initial centre of the DS. Besides, the zero motion vector, (0,0) is also considered as an initial candidate centre. It can exploit the center-biased motion-vector distribution characteristics of real-world video sequences and avoid incorrect predictions if all neighboring MB motion vectors fail. In the following, the process of motion estimation of the remaining opaque macroblocks is described as shown below.
    a) Determination of the initial search centre: Since the correlation between the opaque macroblock and its corresponding macroblocks is high in general as shown in Figure 3-6, the initial search centre of the opaque macroblock is the best-matched motion vector among all of its neighbouring macroblocks and it is given by

$$\overrightarrow{SC}_{init} = \arg_{(u,v)} \min SAD(u,v) \qquad \text{(3-5)}$$

where $(u,v) \in$ {estimated motion vectors of neighbouring macroblocks, (0,0)}

b) $\overrightarrow{SC}$ is then regarded as the initial search centre to perform the DS.

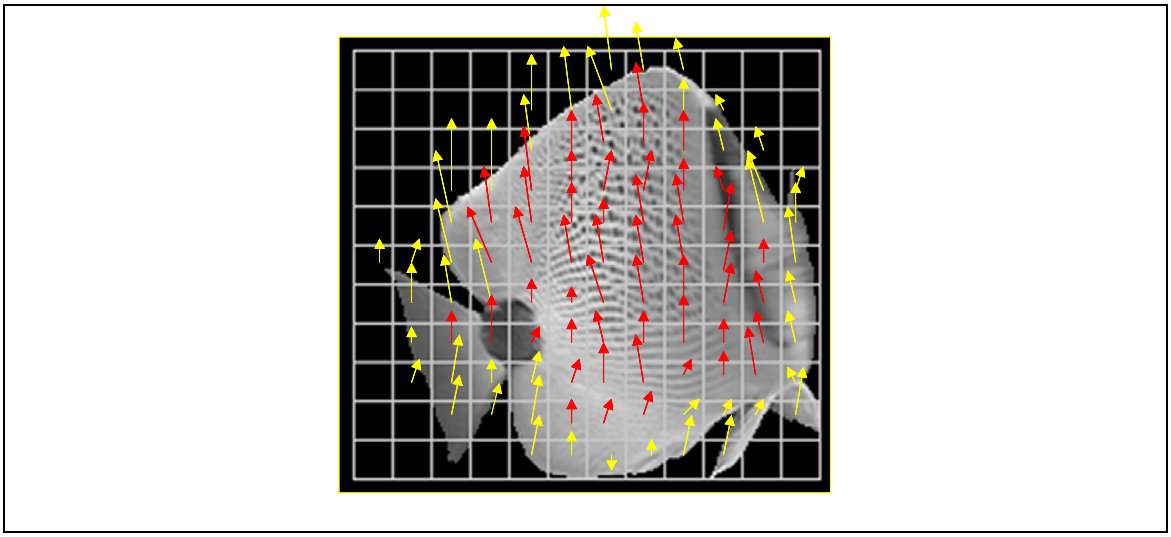


Figure 3-6. Example of high correlation between the motion vectors of opaque macroblock and those of the boundary macroblocks.

3.5 *Simulation Results*

3.5.1 Quality comparison

A series of computer simulations have been conducted to evaluate the performance of the proposed algorithm. It is obvious that quality of the provided binary alpha-plane of a video object seriously affects our proposed BAAS algorithm. Hence, we generally look for sequences with available binary alpha planes for video objects, such as the "Goldfish", "Children", "Bream", etc for our experimental work. However, we have also included a segmented video object in the "Stefan" sequence in our simulation to test this kind of situation. Note that the sequence "Stefan" contains some fast moving objects and fast camera motion. A moving object of the "Stefan" sequence is depicted in Figure 3-7 and the alpha plane of which was obtained from private communications. In our simulation, the maximum allowable displacement in both

horizontal and vertical directions is 15 with a block size of 16×16. The mean square error (MSE) is used to compare the performance of the proposed algorithm with some related techniques reported in the literature. For our proposed PSA(BAAS+DS) algorithm, α and β were set to 0.3 and 0.25, respectively. Table 3-1 lists the experimental results to determine the values of α and β. The case with α = 0.3 and β = 0.5 is the optimal setting that obtains the minimum MSE result for our tested video objects. In order to minimize the required computational load, we have decreased α and β to 0.25 separately. It is because decreasing α and β can reduce the number of interested starting points (SP) in the regular starting point patterns (SPP). Moreover, we can use a bitwise shift operation to replace the floating-point multiplication of 0.25. We have found that if α = 0.3 and β = 0.25, it becomes the best setting to trade off the required number of operators and MSE performance.

(a) VOP at Frame 87

(b) VOP at Frame 88

Figure 3-7. Examples of the video object plane in segmented "Stefan".

Table 3-1. Experimental results to determine α and β.

	Number of Addition	Number of Absolute operation	MSE
$\alpha = 0.3$; $\beta = 0.5$			
Bream	3174676	1126172	173.1
Children	3733316	1531701	101.6
Goldfish	8874788	3555965	116.5
$\alpha = 0.25$; $\beta = 0.5$			
Bream	2991453	1025084	173.7
Children	3547168	1431670	101.8
Goldfish	8424077	3312306	117.5
$\alpha = 0.3$; $\beta = 0.25$			
Bream	2620971	839357	173.1
Children	2982690	1148788	101.9
Goldfish	7159290	2678477	116.8

Figure 3-8 compares the results of the MSE of the motion-compensated video object of the proposed PSA(BAAS+DS) algorithm together with the results of other approaches, including the FSA, the DS, the PSA(FSA+DS), and the PSA(DS+DS). It shows that there is a big increment in prediction error for the PSA(DS+DS) and the conventional DS as compared to that of the FSA. This is because the probability of having complex error surfaces as shown in Figure 3-3(a) is large for boundary macroblocks with fast moving objects, especially after the transparent pixels in a bounding box are padded using the repetitive padding process defined in MPEG-4. This situation causes an unreliable stop in the search for using the conventional DS, and it implies that these kinds of algorithms can be trapped in a local minimum easily. However, our BAAS can resolve this problem by placing one of the starting points closest to the global minimum which is obtained by making use of the binary alpha-plane. For the video sequence, “Stefan”, the object is not perfectly represented by its segment mask (Figure 3-7). However, shape correlation between masks of successive VOPs can provide useful shape information for our BAAS to assist the boundary macroblock motion estimation in a VOP. The evidence is shown in Figure 3-9 and Table 3-2. Figure 3-9 compares the performance of different search algorithms

including the BAAS, DS and FSA when they are applied to process boundary macroblocks only. Table 3-2 summarizes the average peak signal to noise ratios (PSNR) and Mean Square Errors (MSE) of different kinds of macroblocks with different search algorithms. The proposed BAAS is significantly better than that of the DS, which provides about 0.5dB PSNR improvement in the boundary macroblock. Moreover, Figure 3-10 compares the PSNR performance of the algorithms for boundary macroblocks of Video objects with high motion activity, such as "Bream" and "Segmented Stefan". We can find that up to about 4dB PSNR improvement can be achieved by the BAAS, while the human visual system is very sensitive to the defeat at object edge regions.

Table 3-2. Comparison of average PSNR and MSE for various algorithms per VOP and in diffrenent types of macroblock.

Algorithms	PSNR (db) of VOP	MSE of VOP	PSNR (dB) of boundary macroblock	MSE of boundary maccroblock	PSNR (dB) of opaque macroblock	MSE of opaque macroblock
Children (Average Num. of boundary MB : Average Num. of Opaque MB = 74 : 20)						
FSA	*26.0*	*169.9*	*27.0*	*133.1*	*23.5*	*306.9*
DS	*25.6*	*189.9*	*26.6*	*151.8*	*23.3*	*330.7*
PSA(BAAS+DS)	*25.9*	*173.1*	*27.0*	*136.1*	*23.5*	*310.0*
Bream (Average Num. of boundary MB :Average Num. of Opaque MB = 54 :78)						
FSA	*28.7*	*98.5*	*32.3*	*41.8*	*27.4*	*130.0*
DS	*27.6*	*128.4*	*31.5*	*53.0*	*26.3*	*170.7*
PSA(BAAS+DS)	*28.6*	*101.9*	*32.0*	*44.7*	*27.3*	*133.8*
Goldfish (Average Num. of boundary MB : Average Num. of Opaque MB = 141 : 106)						
FSA	*28.1*	*106.2*	*27.3*	*125.2*	*29.5*	*78.9*
DS	*27.1*	*132.3*	*26.3*	*159.9*	*28.8*	*93.6*
PSA(BAAS+DS)	*27.7*	*116.8*	*26.9*	*137.8*	*29.1*	*85.4*
Segmented Stefan(Average Num. of boundary MB : Average Num. of Opaque MB = 28 :6)						
FSA	*22.6*	*381.6*	*22.9*	*358.7*	*21.8*	*505.7*
DS	*21.8*	*462.1*	*22.0*	*442.7*	*21.4*	*562.8*
PSA(BAAS+DS)	*22.4*	*401.9*	*22.6*	*378.8*	*21.7*	*525.7*

The PSNR of opaque MBs by using DS to process video objects in "Children", "Bream", "Goldfish" and "Stefan" are 23.3dB, 26.3dB, 28.8dB and 21.4dB respectively. Results of using PSA(BAAS+DS) are 23.5dB, 27.3dB, 29.1dB and 21.7dB for the

above video objects respectively. This shows that the PSA coupled with the BAAS can provide about 0.45dB PSNR improvement for coding the opaque macroblocks in our tested sequences. The above results show that an accurate estimation of motion vectors of boundary macroblocks making use of the BAAS and the PSA can successfully provide motion estimation for the opaque macroblock of VOs.

The average PSNR performance per VOP of the PSA(BAAS+DS) is about 22.4dB. It is very close to the average PSNR of the FSA which is about 22.6dB for the tested sequences.

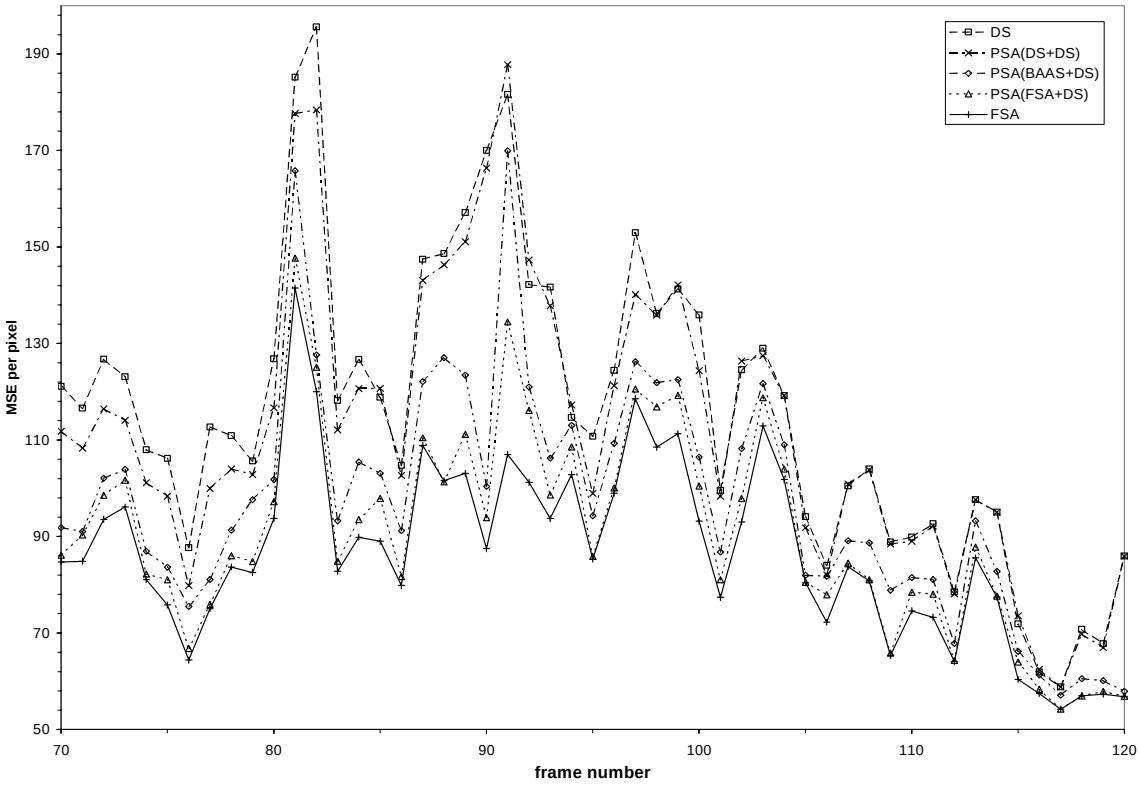


Figure 3-8(a) “Goldenfish”

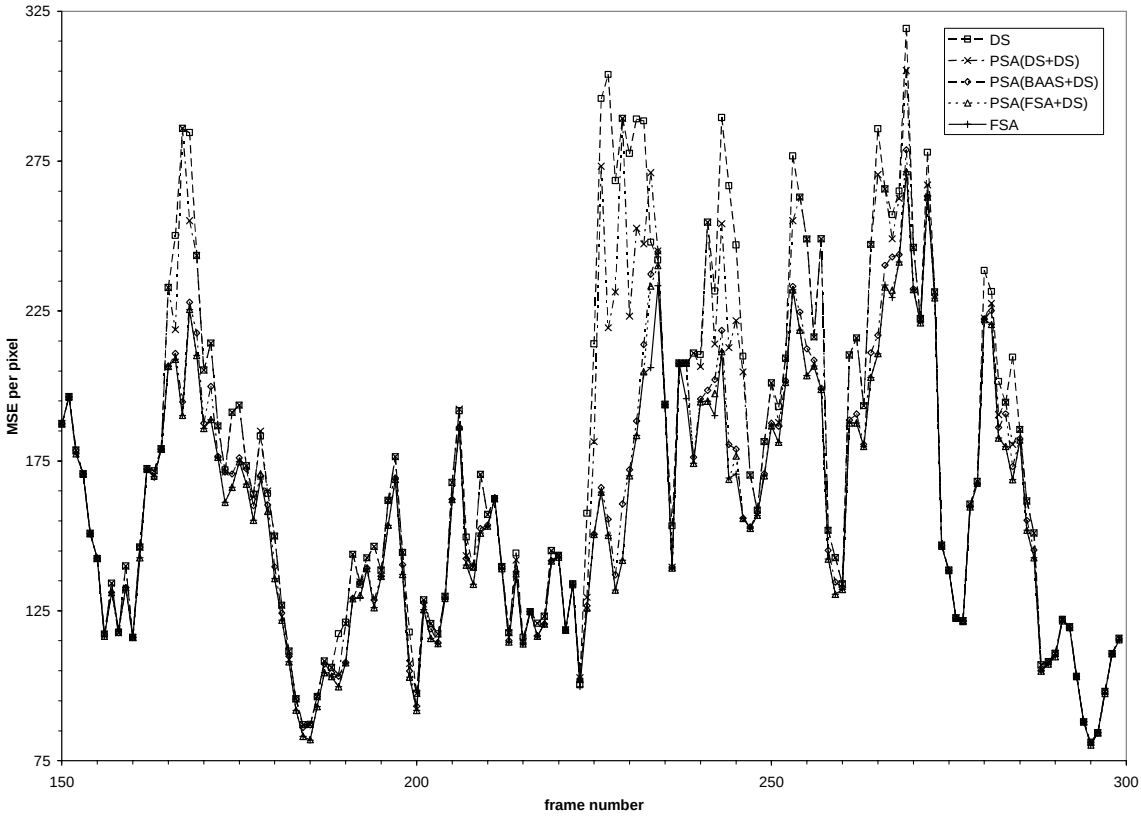


Figure 3-8(b) “Children”

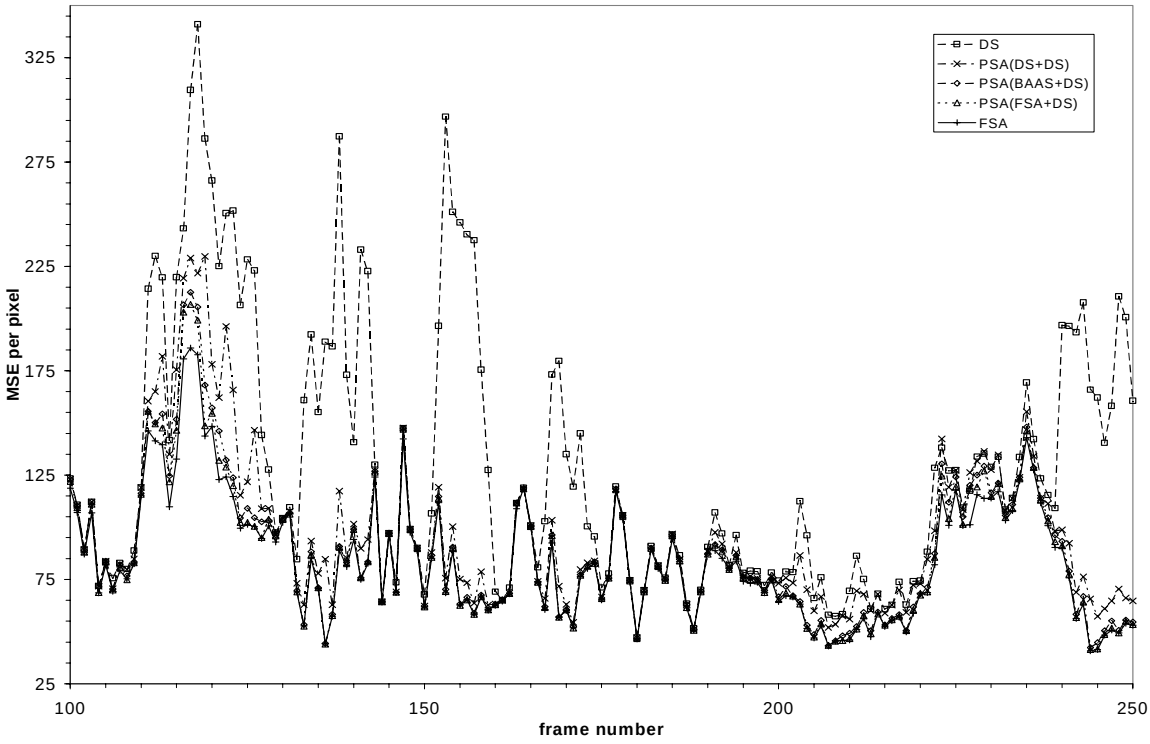


Figure 3-8(c) “Bream”

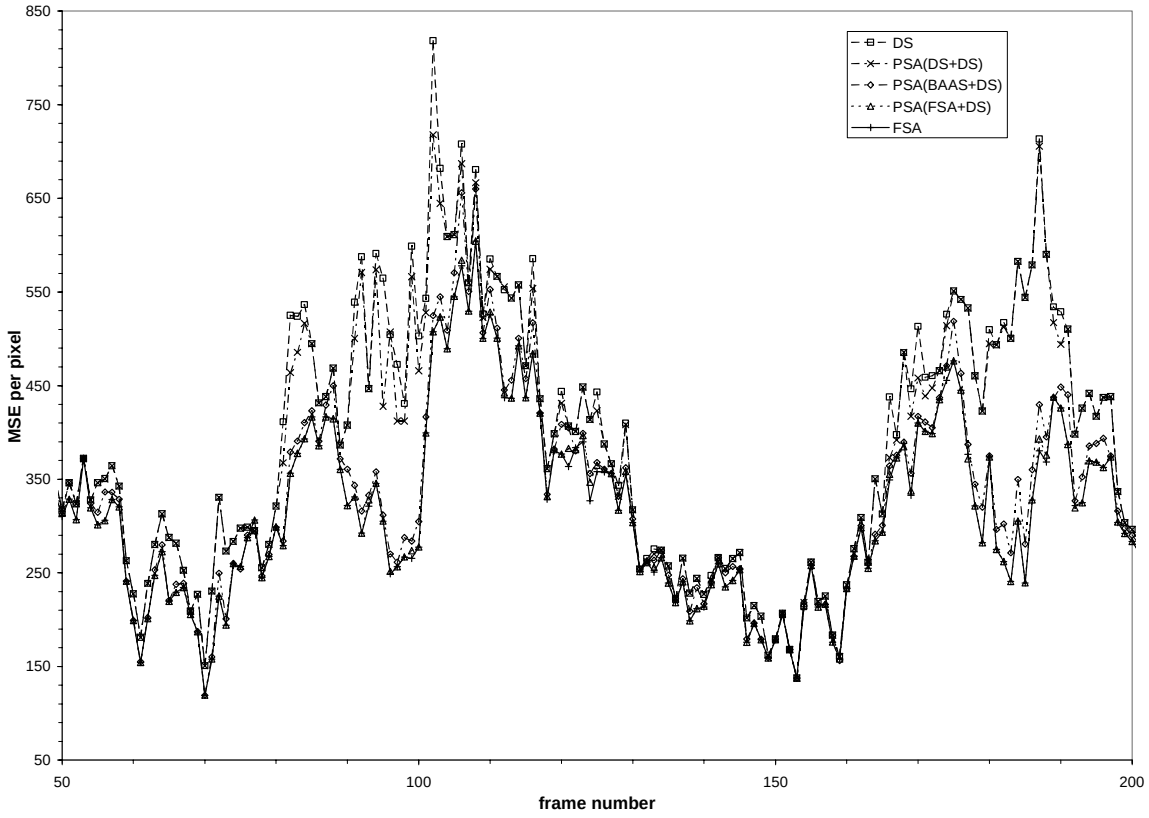


Figure 3-8(d) Segmented “Stefan”

Figure 3-8. MSE performance comparison of MPEG-4 video objects. (a) “Goldenfish”, (b) “Children”, (c) “Bream” and (d) Segmented “Stefan”

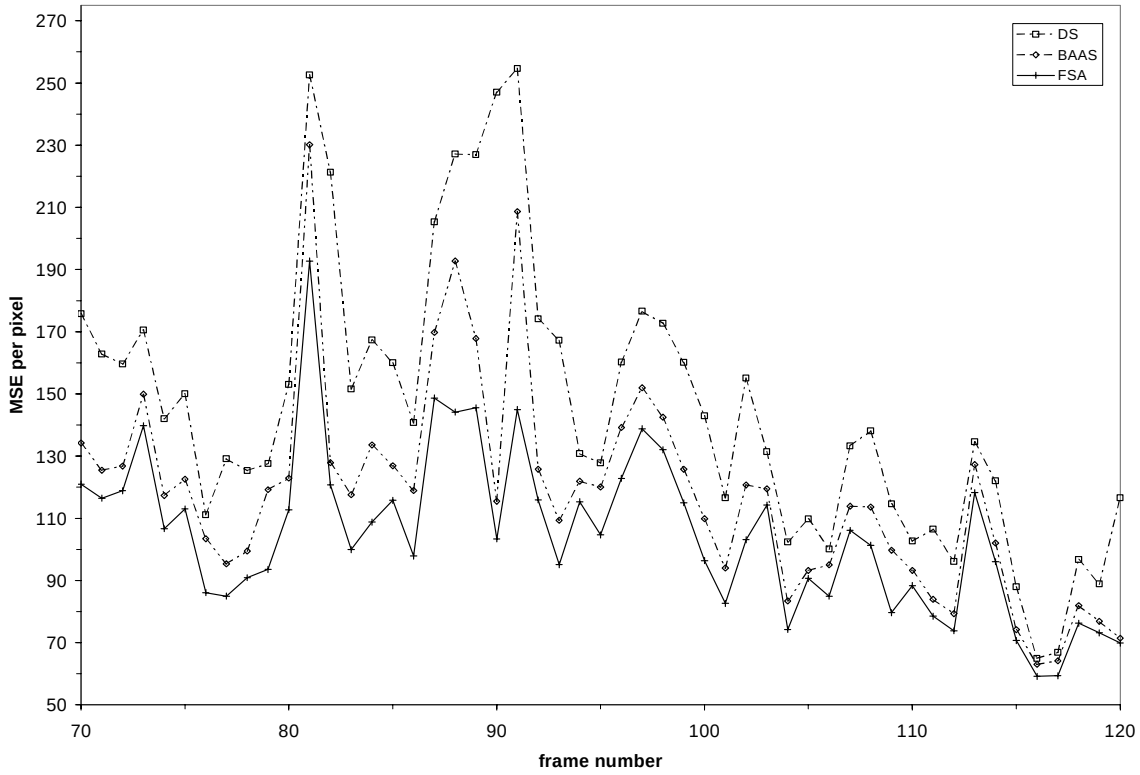


Figure 3-9(a) “Goldfish”

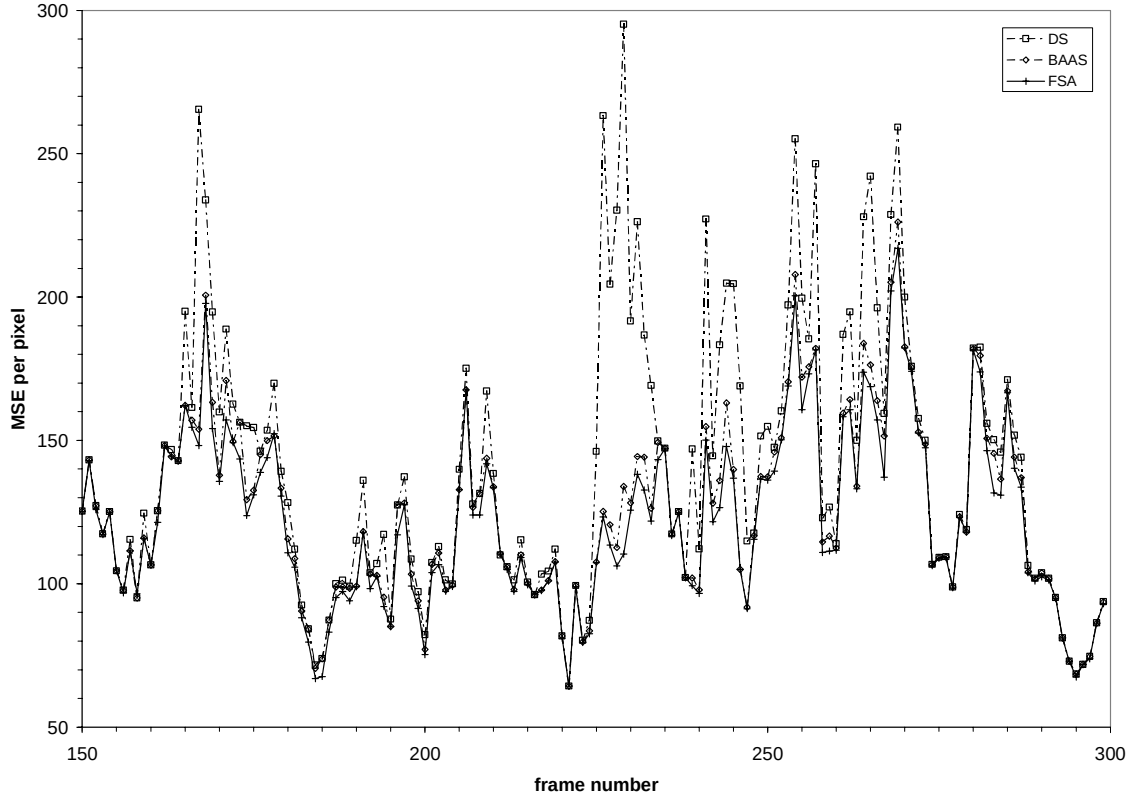


Figure 3-9(b) “Children”

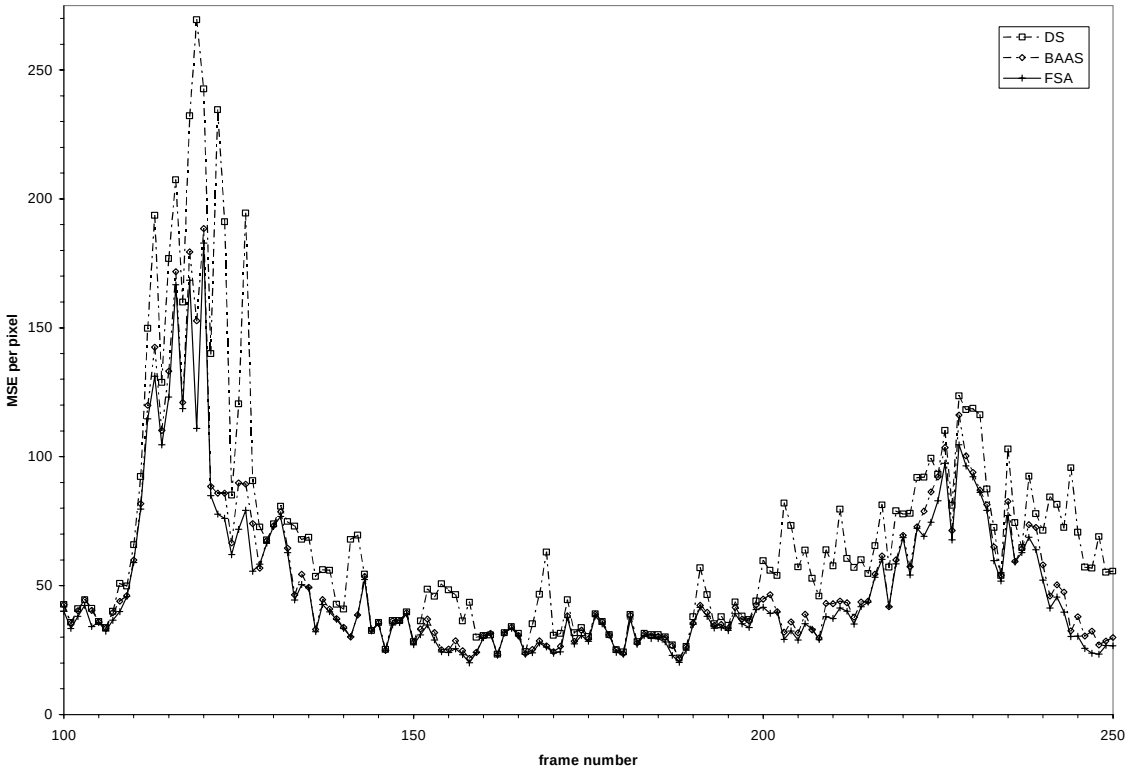


Figure 3-9(c) “Bream”

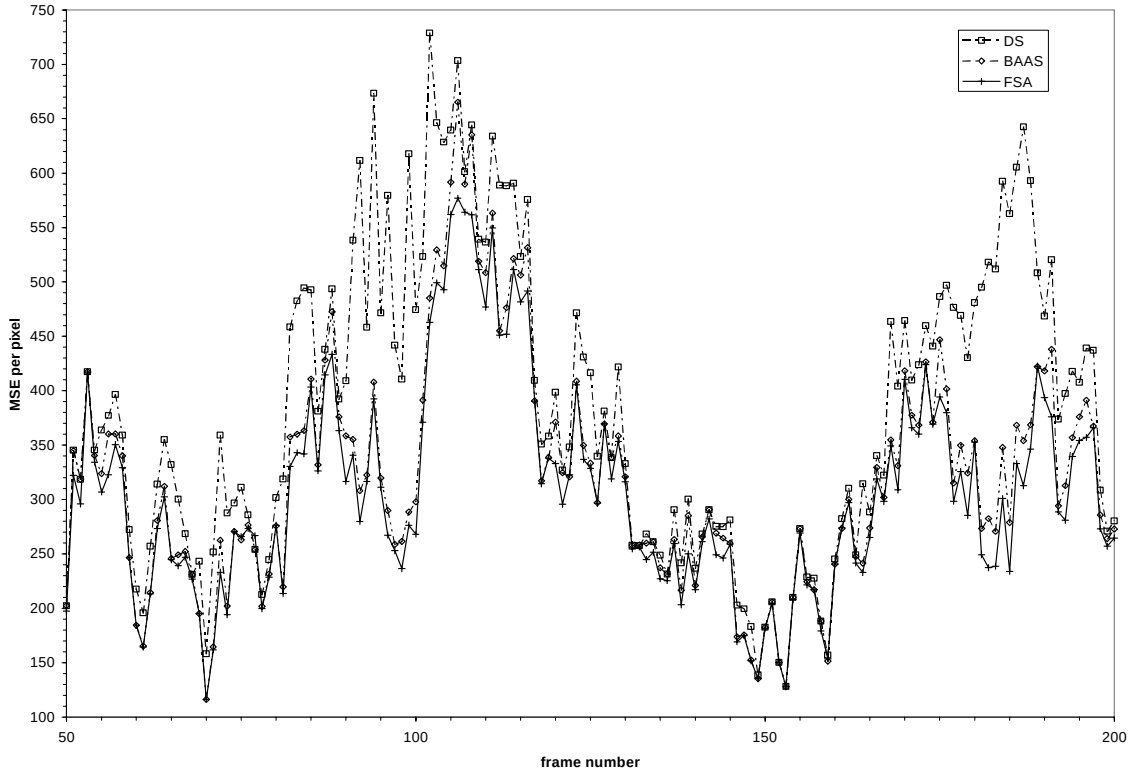


Figure 3-9(d) Segmented “Stefan”

Figure 3-9. MSE performance comparison of boundary macroblocks. (a) “Goldfish”, (b) “Children”, (c) “Bream” and (d) Segmented “Stefan”.

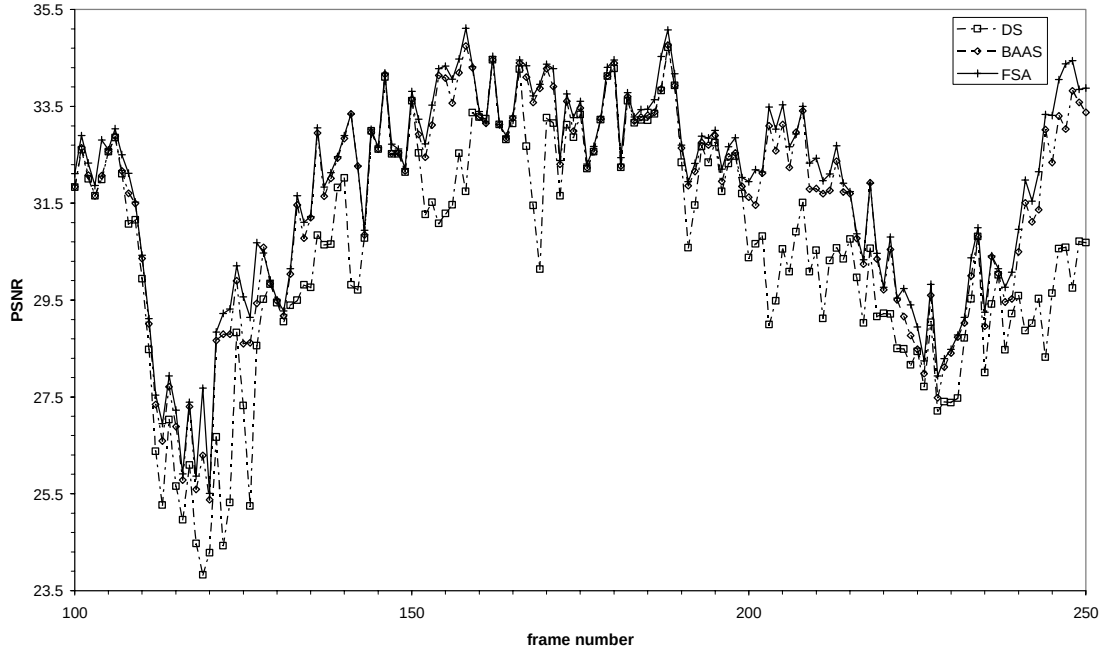


Figure 3-10(a) Bream

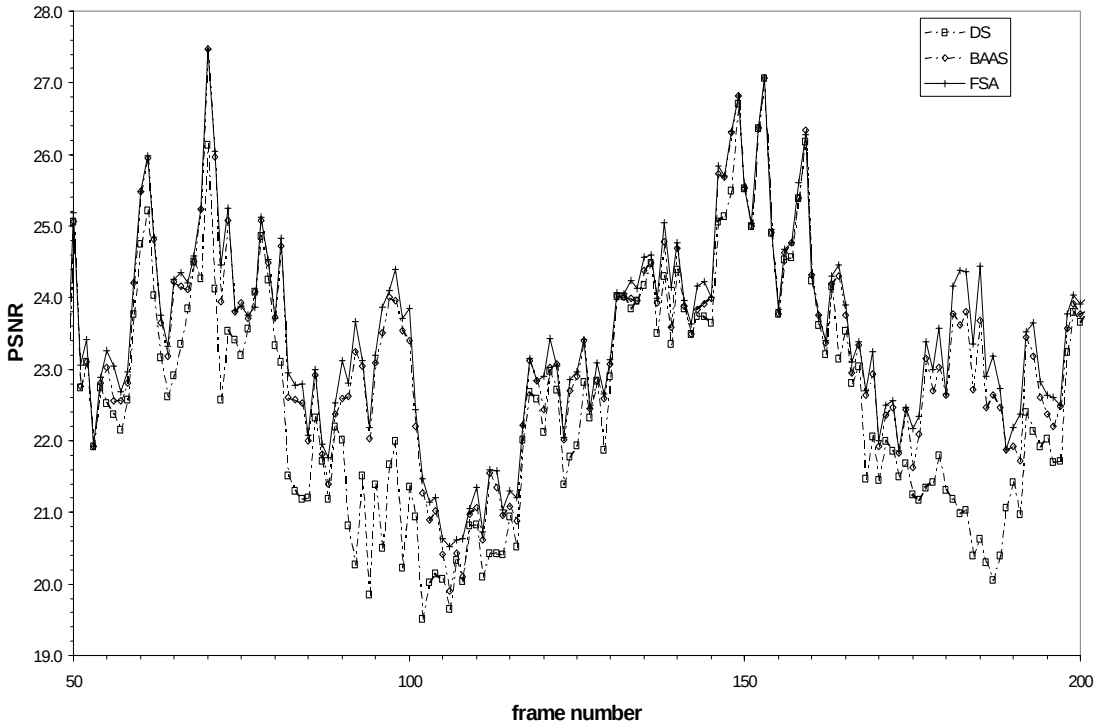


Figure 3-10(b) Segmented “Stefan”

Figure 3-10. PSNR performance comparison of boundary macroblocks. (a) “Bream” and (b) Segmented “Stefan”.

3.5.2 Complexity analysis

In this section, the computational complexity of our proposed algorithm is compared with that of the conventional algorithms including the FSA, the DS and that of the PSA(FSA+DS) and the PSA(DS+DS). In general, several factors are needed to be taken into account in comparing the cost associated with various algorithms. These factors include speed, chip area and power, and a trade off among these factors can be made depending upon the architecture to be used, hence a comparison of the costs associated with various algorithms is not an easy task. However, it is possible to choose a simple way of defining complexity. In comparing 8-bit fixed point implementations, it is assumed that the cost of an 8-bit addition is 16 times more than that of bitwise manipulation [128].

The fixed-point implementation of the proposed algorithm is now compared with that of the FSA, that of the DS, that of the PSA(FSA+DS) and that of the PSA(DS+DS). The matching criterion as shown in (3-1) requires two-dimensional operations; i.e. N_{opaque}-1 additions, N_{opaque} subtractions, and N_{opaque} absolute conversions per search point are needed; where N_{opaque} is the number of opaque pixels in the MB. Therefore, for its 961 search points in the search range of the FSA, each MB requires 961×(N_{opaque}-1) additions, 961×(N_{opaque}) subtractions, and 961×(N_{opaque}) absolute conversions. For the DS and the search algorithms with the help of our proposed PSA and BAAS, only the average number of search points per MB for the entire sequence is reported. For the conventional DS and the proposed BAAS, the number of search points required depends on whether the stop criterion is fulfilled.

Our proposed BAAS employs multiple initial search points to enhance the accuracy of the motion vectors of boundary MBs. Apart from calculating the matching criterion, the formation of the final SPP for providing multiple initial search points is the major overhead of our proposed BAAS. Its computational complexity is now examined. Step (2) in the BAAS is not considered as overhead since its major computation has been taken into account in the required search points as mentioned above and it just requires one additional multiplication for the calculation of b ´ $SAD_{\text{min\_in\_updated\_SPP}}$. To adjust the regular SPP, a computational effort which is equivalent to 49 times of that required for BAMS calculation and selection process, as shown in (3-3) and (3-4), is required in the boundary macroblock. From (3-3), it is obvious that the *BAMS* can be easily implemented by a simple circuitry containing an 'XOR' logic gate and a counter. The major advantage in using this binary alpha-plane is its simplicity, which requires 256 bitwise operations per search point and it is equivalent to 16 additions per search point. The selection process involves the calculation of T_{BAMS},

as shown in (3-4). Again, the calculation of T_{BAMS} is simply implemented by a counter and one multiplication.

In the PSA algorithm, the initial search centre for an opaque MB is determined by finding the best-matched motion vector in a set of candidate initial centres which include all estimated neighboring motion vectors and the zero motion vector, (0,0). The determination is just considered as the additional search points. Combining all these, Table 3-3 shows a comparison of the operational complexity of the proposed PSA with different combinations of using the new BAAS, the DS and the FS, the conventional DS and the conventional FSA. The table shows that the PSA(BAAS+DS) requires more computational effort as compared with the DS and the PSA(DS+DS). This is because BAAS can avoid the serious local minimum problem of the DS by involving a reasonable number of starting points in the boundary MBs, which have a high degree of similarity of arbitrary shapes between the current MB and the MB in the search window. However, the proposed PSA(BAAS+DS) is fast compared with the PSA(FSA+DS), and it is more significant in the case of the FSA. On average, the PSA(BAAS-DS) can speed up the motion estimation about 15 and 23 times in terms of the total number of operations when compared to the PSA(FS+DS) and the FSA respectively.

Table 3-3. Comparison of computational complexity and average MSE for various algorithms.

Algorithms	Average Num. of additions per VOP	Average Num. of absolute conversions per VOP	Average Num. of bitwise XOR per VOP (equivalent to 8-bit additions)	Average Num. of multiplication per VOP	Average MSE	Average PSNR
Children						
FSA	*45,835,483*	*22,962,590*	–	–	*169.9*	*26.0*
DS	*804,671*	*403,123*	–	–	*189.9*	*25.6*
PSA(FSA+DS)	*36,405,768*	*18,238,441*	–	–	*170.7*	*26.0*
PSA(DS+DS)	*829,356*	*415,425*	–	–	*185.5*	*25.6*
PSA(BAAS+DS)	*1,680,712*	*839,357*	*57,817 (925,068/16)*	*147*	*173.1*	*25.9*
Bream						
FSA	*64,836,153*	*32,481,517*	–	–	*98.5*	*28.7*
DS	*1,278,221*	*640,361*	–	–	*128.4*	*27.6*
PSA(FSA+DS)	*27,066,110*	*13,559,355*	–	–	*100.7*	*28.6*
PSA(DS+DS)	*1,213,546*	*607,777*	–	–	*105.2*	*28.4*
PSA(BAAS+DS)	*2,297,516*	*1,148,788*	*42,132 (674,104/16)*	*108*	*101.9*	*28.6*
Goldfish						
FSA	*121,509,689*	*60,873,738*	–	–	*106.2*	*28.1*
DS	*2,931,646*	*1,468,692*	–	–	*132.3*	*27.1*
PSA(FSA+DS)	*70,485,018*	*35,311,215*	–	–	*109.6*	*27.9*
PSA(DS+DS)	*2,709,257*	*1,357,018*	–	–	*129.2*	*27.2*
PSA(BAAS+DS)	*5,356,807*	*2,678,477*	*110,835 (1,773,361/16)*	*283*	*116.8*	*27.7*
Stefan						
FSA	*16,443,488*	*8,237,833*	-	-	*381.6*	*22.6*
DS	*434,657*	*120,396*	-	-	*462.1*	*21.8*
PSA(FSA+DS)	*14,164,687*	*6,855,377*	-	-	*385.2*	*22.6*
PSA(DS+DS)	*426,708*	*120,092*	-	-	*456.0*	*21.9*
PSA(BAAS+DS)	*1,226,416*	*355,775*	*21,748 (347,960/16)*	*56*	*401.9*	*22.4*

3.6 *Conclusions*

In this charter, a fast search algorithm for block motion estimation of arbitrarily shaped video objects in MPEG-4 has been proposed. By considering the correlation between the boundary macroblock (MB) and the opaque MB, a priority search algorithm (PSA) for arbitrarily shaped video objects has been proposed. The PSA initially performs motion estimation on all boundary MBs within the bounding box of a VOP. This is in contrast with the conventional raster-scanning approach. The motivation behind the new search strategy is that the opaque MBs which are inside of

moving video objects are correlated highly with the moving boundary MBs. For each opaque MB, if all motion vectors of its neighboring boundary MBs have already been computed, the current opaque MB can take the best-matched one among all its neighboring motion vectors and the zero motion vector (0,0) as the initial centre and employ a conventional fast block matching algorithm. We have also demonstrated that obtaining accurate motion information about boundary MBs is important to improve the performance of the proposed motion estimation algorithm for VOP.

A fast and efficient algorithm for estimating the motion vectors of boundary MBs is suggested, which is referred to as the binary alpha-plane assisted search (BAAS) in this paper. The binary alpha-plane is used for the adjustment of the starting point patterns of the search windows such that a limited number of starting points can still provide a high chance of catching the global minimum in the boundary MBs. Experimental results show that our PSA coupled with the BAAS can reduce the heavy computational burden of the FSA without significantly increasing the prediction error of the motion-compensated frame. The proposed algorithm is significantly better than that of the famous DS and substantially improves the accuracy of the block motion estimation for MPEG-4 video objects.

However, because the BAAS and the PSA are not highly regular, hardware implementation is difficult. On the other hand, as general-purpose processors are becoming more and more powerful, software encoding will likely be possible, since this is a trend in video processing. As a concluding remark, we believe that the results of our work will certainly be useful in the future development of software MPEG-4 codecs.

Chapter 4. New Adaptive Partial Distortion Search Using Clustered Pixel Matching Error Characteristic

4.1 *Introduction*

In order to reduce the computation load, many conventional fast block-matching algorithms have been developed to reduce the set of possible searching points in the search window. All of these algorithms produce some quality degradation of a predicted image. Alternatively, another kind of fast block-matching algorithms which do not introduce any prediction error as compared with the full-search algorithm is to reduce the number of necessary matching evaluations for every searching point in the search window. The partial distortion search is a well-known technique of the second kind of algorithms. In the literature, many researches tried to improve both lossy and lossless block-matching algorithms by making use of an assumption that pixels with larger gradient magnitudes have larger matching errors on average. Base on a simple analysis, it is found that on average, pixel matching errors with similar magnitudes tend to appear in clusters for natural video sequences. By using this clustering characteristic, we propose an adaptive partial distortion search algorithm which significantly improves the computation efficiency of the original partial distortion search. This approach is much better than other algorithms which make use of the pixel gradients. Furthermore, the proposed algorithm is most suitable for motion estimation of both opaque and boundary macroblocks of an arbitrary shaped object in MPEG-4 coding.

Block-based motion compensation technique has been widely used in many modern video coding standards [3, 4]. It is used to reduce the redundancy between successive frames in a video. Motion estimation process is to obtain a motion vector for a target macroblock by using the block matching technique, which minimizes a measure

of matching distortion between the target MB in the current frame and a candidate MB within a search window in a reference frame. The displacement between the candidate MB with the smallest distortion and the target MB will be selected as the resulting motion vector. One of the most frequently used criteria to measure the matching distortion is the sum of absolute difference (SAD). The SAD between a target MB at position (x,y) in the current frame, f_t, and a candidate MB at position ($x+u$, $y+v$), in the reference frame, f_{t-1}, is defined as below.

$$SAD(x, y; u, v) = \sum_{j=0}^{M} \sum_{i=0}^{M} \left| f_t(x+i, y+j) - f_{t-1}(x+i+u, y+j+v) \right| \quad \textbf{(4-1)}$$

where $M \times M$ is size of a block and M is equal to 16 for our consideration; $f_t(\cdot,\cdot)$ and $f_{t-1}(\cdot,\cdot)$ represent pixels intensity in the current frame and the reference frame respectively. This equation is identical to the l_1-norm defined in (2-11) with n = 1.

The simplest block matching motion estimation algorithm is the full search algorithm (FSA). This algorithm can give an optimal solution by exhaustively searching all possible locations within a search window, W. The resulting best motion vector, $(\hat{u}, \hat{v})$, is defined in (2-13). However, this algorithm suffers from heavy computational load. In order to resolve this difficulty, many fast search algorithms have been developed in the past.

In this study, we have investigated and compared the performance of two categories of fast lossless search algorithms. 1) Algorithms in the first category use a reduced complexity distortion measure to save computation, such as partial distortion (PDS) techniques [92, 93]. The PDS [92] reduces the computation complexity by terminating the SAD calculation early when it finds that a partial SAD is already greater than the minimum SAD encountered so far in the searching. In general, the PDS is regarded as a fast full search algorithm because it has identical prediction quality as that of the FSA. 2) The second category makes use of mathematical inequalities to reduce

the computational load; this includes the Successive Elimination Algorithm (SEA) [13]. By making use of the Minkowski's inequality, the SEA eliminates an impossible candidates MB without calculating the SAD.

In addition, we suggest further techniques to improve the searching efficiency of the PDS. Its efficiency also comes from an early termination of the partial SAD. Let us define a generalized form of the partial SAD as shown below,

$$SAD_p(x,y;u,v)=\sum_{j=0}^{p}\sum_{j=0}^{15}\left|f_t(x+k_n,y+l_n)-f_{t-1}(x+k_n+u,y+l_n+v)\right| \quad \textbf{(4-2)}$$

where $\{(k_n,l_n)|n=0,...,j\times 16+i,...,256\}$ is an index set of all pixels in a MB, and p = 0,…,15 which specifies the number of elements for producing the sum of errors for a partial *SAD*. For a given p, there are $16\times(p+1)$ pixels to be accumulated to SAD_p. The index set determines the coordinates and orders of the pixel matching errors to be accumulated to the SAD_p. One simple idea to improve the PDS is to design an adaptive index set such that a pixel with greater matching error is firstly computed, and this error is accumulated to the SAD_p earlier than other pixels. As a result, the SAD calculation can be terminated sooner. In the literature, many researches [96, 106] indicated that pixels with larger gradient magnitudes have larger matching errors on average. They made use of this hypothesis to develop their searching algorithms. An approach is to make use of representative pixels and adaptive matching scan PDS (AMS-PDS)[96] to determine the index set by sorting the gradient magnitude of rows or columns in the target MB of the current frame in descending order. As a result, pixels of a row or column with greater gradient magnitudes will be used to calculate the SAD_p prior to other rows or columns in the MB. However, it can be shown that pixel matching errors with similar magnitudes tend to appear in clusters in natural video sequences. This characteristic is illustrated in Figure 4-3 and discussed in the next Section. In this chapter, we propose an adaptive partial distortion search algorithm by using the

characteristics of these clustered pixel matching errors. This approach is significantly more efficient as compared to algorithms which make use of pixel gradient properties in an adaptive partial distortion search.

In the rest of this charter, we firstly explain and illustrate the characteristics of the pixel errors that tend to form clusters in Section 4.2. Section 4.3 applies these characteristics to develop a new clustered pixel matching error for adaptive partial distortion search algorithm (CPME-PDS). In Section 4.3.1, we establish an analysis to determine an adaptive index set required for the CPME-PDS. Then our proposed CPME-PDS is described in details in Section 4.3.2. It is unavoidable to have a certain amount of overheads for the establishment of the adaptive index set. These overheads are described in Section 4.3.3. Section 4.4 gives the details of our experiments and results. In order to compare the performance of the adaptive PDS based on the cluster pixel matching error characteristic and based on the pixel gradient characteristic, we have also designed another adaptive PDS, the pixel gradients based adaptive partial distortion search algorithm (PG-PDS). The details of the PG-PDS are described in Section 4.4.1. Section 4.4.2 presents the results and analysis of the CPME-PDS comparing to other fast algorithms including the PG-PDS. Finally, concluding remarks are given in Section 4.5.

4.2 *The characteristic of clustered pixel matching error*

The major idea of our proposed CPME-PDS is to design an adaptive index set such that a pixel with greater matching error can be accumulated to the SAD_p sooner than other pixels according to the order indicated by the index set.

For this reason, it is necessary for us to investigate possible spatial distributions of pixel matching errors in a MB. We have found that errors with similar magnitude tend to appear together in clusters. It is because natural images are dominated by low

frequency components. The matching errors of low frequency regions between a target MB and a candidate MB have similar magnitudes and are partitioned by edge pixels of these two MBs. This phenomenon is demonstrated in Figure 4-1. Figure 4-1(a) depicts the matching of a one dimensional (1-D) target MB (thick continuous line) within a 1-D search window (thin dotted line). The corresponding pixel matching errors appear in a cluster form as shown in Figure 4-1(b).

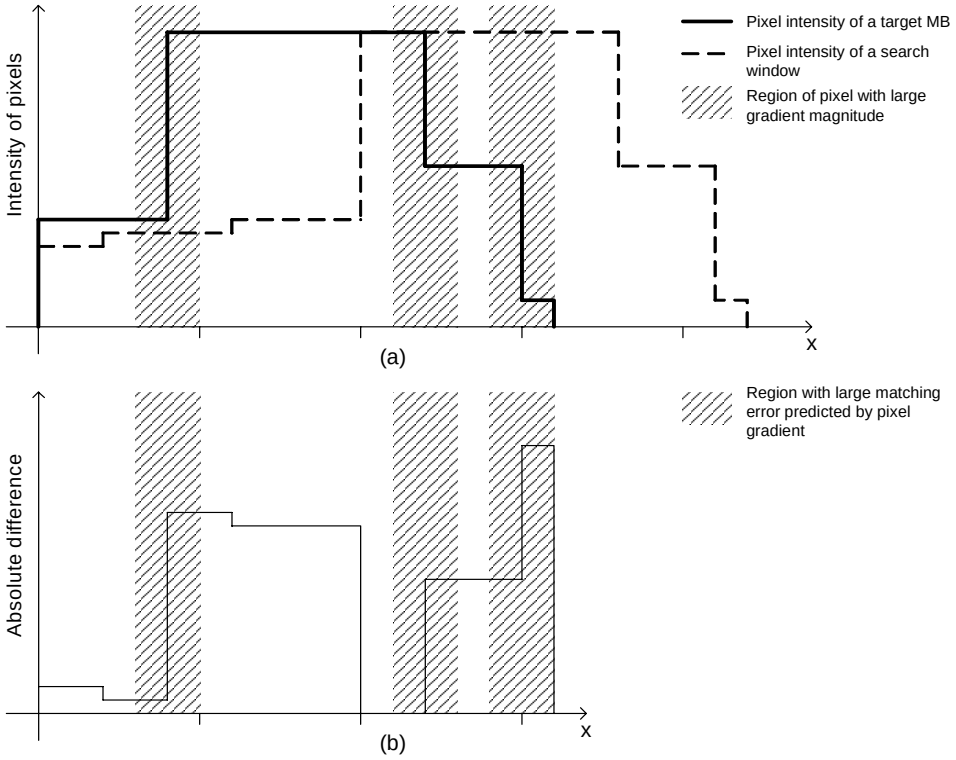


Figure 4-1. (a) Matching of a 1-D target MB within a 1-D search window. (b) Corresponding pixel matching error of the target MB at the current position.

Edges are the most prominent feature in image processing. They are also frequently used to predict pixel matching errors in motion estimation. The prediction is accurate especially near a minimum distortion position. Figure 4-2(b) shows that locations with large pixel matching errors (the hatched region) can be detected by using pixel gradients when the target MB is located near a good candidate MB. However, the result is not good enough in general. In Figure 4-1(b), only pixel matching errors in the hatched region are found, while matching errors outside the hatched region are underestimated. Figure 4-3(a) and (b) are examples of prediction errors in two motion

compensated MBs, which are extracted from the sequence “Football” and video object “Goldfish”, respectively. These examples demonstrate the clustered prediction errors during motion estimation.

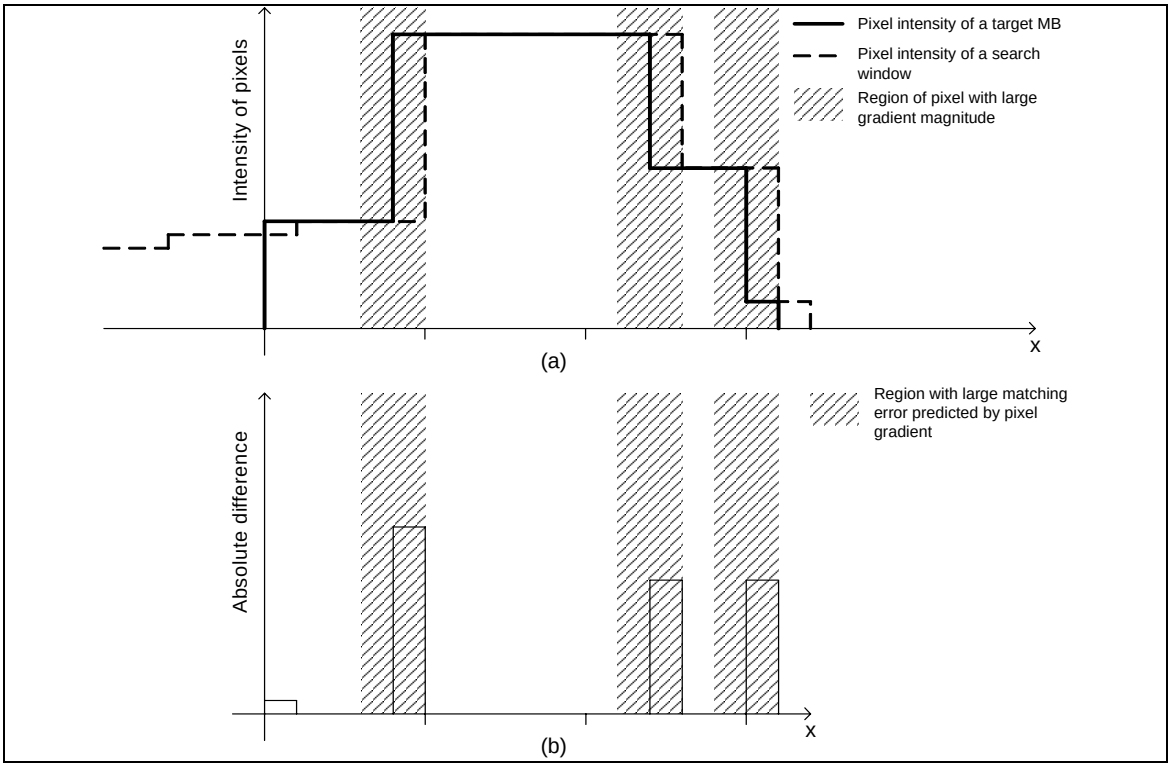


Figure 4-2. (a) Matching of a 1-D target MB within a 1-D search window near a minimum distortion location. (b) Corresponding pixel matching error of the target MB at the current position.

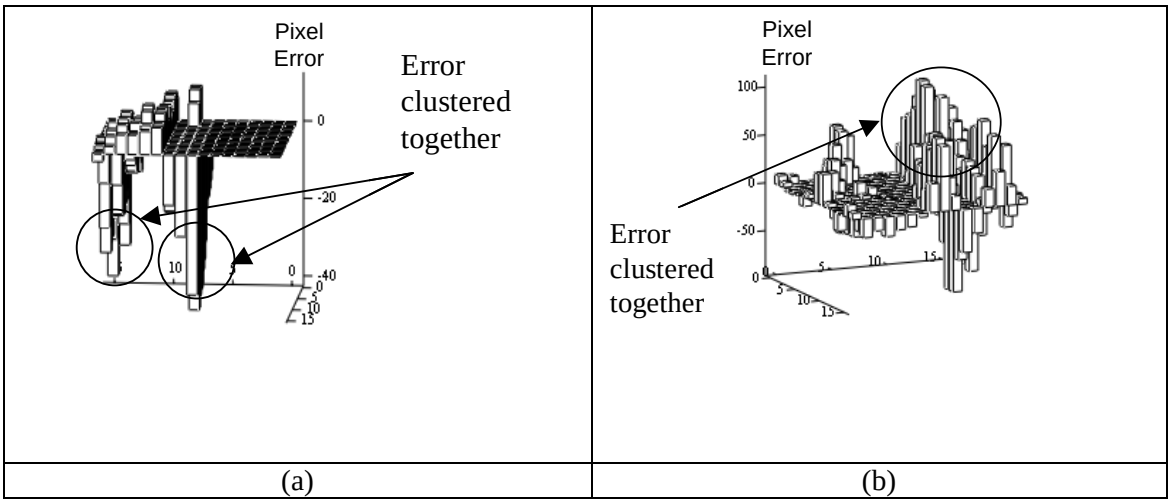


Figure 4-3. Examples of clustering errors in a motion compensated prediction MB of (a) the sequences “Football” and (b) the video object “Goldfish”.

According to the above analysis, we can predict that clustered pixel matching error characteristic can be used to achieve greater advantage in an adaptive partial distortion search.

4.3 *Proposed Algorithm*

4.3.1 Determination of an adaptive index set

For a given target MB, the positions of pixels are represented by an index set, $S = \{(k_n, l_n) | n = 0, ..., N-1\}$, where N is the number of pixels in a MB. For a single pixel at $s_n = (k_n, l_n)$, $s_n \in S$, its matching error is, $e(s_n) = I_t(s_n) - R(s_n)$, where $R(s_n)$ is a random variable which represents the pixel value at s_n of a candidate MB. In the following discussion, the notation for properties relating to the pixel at s_n is indicated by the argument n, and the MB location (x,y) and motion vector (u,v) are dropped for simplicity. Hence, the pixel matching errors are represented by,

$$e(n) = I_t(n) - R(n) \quad \textbf{(4-3)}$$

To improve the saving in computation of a PDS, pixel matching errors with an ideal index set must have the following relation,

$$e(0)^2 \geq ... \geq e(n)^2 \geq ... \geq e(N-1)^2$$

To fulfill the above objective, we have to predict the pixel matching error of each $p(n)$ at location s_n. Hence, the expected values of $p(n)$ must fulfill the following criterion,

$$E\left[p(0)^2 \right] \geq ... \geq E\left[p(n)^2 \right] \geq ... \geq E\left[p(N-1)^2 \right] \quad \textbf{(4-4)}$$

Let us define $p(n) = I_t(n) - m$, where m is a reference value to be used to obtain the predicted pixel matching errors. One possible solution of m is to minimize the expected value of the sum of squares of the differences between $e(n)^2$ and $p(n)^2$,

$$m = \arg\min_{m} \left\{ E\left[\sum_{n=0}^{N-1} \left[\left(I_t(n) - R(n) \right)^2 - \left(I_t(n) - m \right)^2 \right]^2 \right] \right\}$$

i.e.

In solving this equation, we have

$$\frac{d}{dm} E\left[\sum_{n=0}^{N-1} \left[\left(I_t(n) - R(n) \right)^2 - \left(I_t(n) - m \right)^2 \right]^2 \right] = 0 \qquad \textbf{(4-5)}$$

By substituting $R(n) = I_t(n) - e(n)$ into eqn. (4-5), finally we can have a cubic equation,

$$m^3 - 3\overline{I_t} m^2 + \left(3\overline{I_t^2} - \overline{e^2} \right) m + \overline{I_t e^2} - \overline{I_t^3} = 0 \qquad \textbf{(4-6)}$$

where $\overline{e^2} = \frac{1}{N} E\left[\sum_{n=0}^{N-1} e(n)^2 \right]$, and

$$\overline{I_t e^2} = \frac{1}{N} E\left[\sum_{n=0}^{N-1} I_t(n) e(n)^2 \right],$$

The roots of the cubic equation are either all reals or one real and two complex conjugates which depend on the discriminant of the equation. We look for real roots for (4-6), such that *m* can be practically useful. Let us assume that natural images are dominated by low frequency components. Hence, let,

$$\overline{I_t}^a \approx \overline{I_t^a},$$

and $\overline{I_t}\,\overline{e^2} \approx \overline{I_t e^2}$

These are valid only if the image frame under question consists mainly low frequencies and the standard deviation of $I_t(n)$ is small enough. As a result, it gives

$$m \approx \begin{cases} \overline{I_t} \\ \overline{I_t} \pm \sqrt{\overline{e^2}} \end{cases} \qquad \textbf{(4-7)}$$

Mathematical detail is shown in Appendix A. The first approximated root is the mean of pixel values in the target MB. To use this mean as the reference value it already gives a better computational saving when comparing to the PG-PDS, for which we proposed it as a comparison. Intuitively, m is a function of pixel values in a candidate

MB, i.e. $m = m\big(R(n)\big)$. The other roots, $\overline{I_t} \pm \sqrt{\overline{e^2}}$ can also be obtained by the following approximation.

$$\overline{R} = \overline{I_t} - \overline{e} \approx \overline{I_t} \pm \sqrt{\overline{e^2}}$$

It indicates that this solution is an approximation of the mean of pixel values in a candidate MB. Note that our assumption is not always true. However, a shifting of *m* would not affect the criterion in (4-4) dramatically. In fact, the solution of $m = \overline{R}$ can also be obtained directly by minimizing the equation, $E\left[\sum_{n=0}^{N-1}[e(n) - p(n)]^2\right]$ (see appendix A). Hence, this result is used to determine an adaptive index set for the CPME-PDS.

4.3.2 Clustered Pixel Matching Errors for Adaptive Partial Distortion Search (CPME-PDS)

There is another factor which affects the ability of a PDS to reject impossible candidates. The earlier the global minimum is met in a search, the earlier the PDS can terminate a partial SAD to reject the candidates. To achieve this purpose, we use two strategies as shown below.

1. The outward spiral scanning is used to exploit the center-biased motion vector distribution characteristics of the real world video sequence [75].
2. The correlation in the motion field is exploited by using a median predictor of three adjacent blocks, left, top and top right blocks to the current position as the initial searching point of the spiral scanning. We have used the median predictor described in [5].

According to the above considerations and our analytical results, we suggest to use the mean of pixel values in the candidate MB of the initial searching point to compute the reference value, *m*, because we can assume that

$$\overline{I_{t-1}(i+u_{med},\, j+v_{med})} \approx \overline{I_t(i,\, j)}$$

where (u_{med}, v_{med}) = the median predictor. The expected pixel matching error, $p_{exp}(n)$, of each pixel in the target MB is calculated with *m*. The required adaptive index set, *S*, is given by sorting $\left|p_{exp}(n)\right|$ in descending order. The partial SAD in (4-2) is calculated with *S* during the searching in an outward spiral scanning. The CPME-PDS approach can be summarized as follows:

CPME-PDS:

Note that all division operations in the following description are integer division with truncation toward zero for the sake of lower complexity.

Step 1) Determine the median predictor, (u_{med}, v_{med}), of the three adjacent blocks.

Step 2) Calculate the reference value, *m*, with the median predictor, (u_{med}, v_{med}).

$$m = \frac{1}{256}\sum_{j=0}^{15}\sum_{i=0}^{15} I_{t-1}(x+u_{med}+i, y+v_{med}+j) \qquad \textbf{(4-8)}$$

Step 3) Initialize an index set, $S' = \left\{(k'_n, l'_n)\middle| n = 0,..., N-1\right\}$, which represents all pixels of the target MB

Step 4) Calculate the expected absolute pixel matching error, $\left|p_{exp}(n)\right|$, of each pixel in the target MB.

$$\left|p_{exp}(n)\right| = \left|I_t(k'_n, l'_n) - m\right| \qquad \textbf{(4-9)}$$

Step 5) Rearrange the order of set S' to obtain an adaptive index set *S* by sorting the expected absolute pixel matching error, $\left|p_{exp}(n)\right|$, in descending order, such that,

$$S = \left\{(k_n, l_n)\middle| n = 0,..., N-1\right\}$$

The $p_{exp}(n)$ corresponding to the order of the sorted index set, S, has the following feature,

$$\left|p_{\exp}(0)\right| \geq \ldots \geq \left|p_{\exp}(n)\right| \geq \ldots \geq \left|p_{\exp}(N-1)\right|$$

Step 6) Apply the adaptive index set, S, to calculate the partial SAD in (4-2) during the searching in an outward spiral scanning.

Note that the adaptive index set is established on a pixel-based approach. It is straightforward to modify the above procedure for the boundary macroblocks of an arbitrary shaped video object (VO) in MPEG-4 [5]. First, the reference value, m, is calculated after that the repetitive padding is applied to a reference video object plane (VOP). It is only necessary to compute the expected pixel matching error, $p_{exp}(n)$, for opaque pixels in the case of a boundary MB. For an index set, N is equal to the number of pixels in a MB, such that $N = 256$ for an opaque MB, while N = number of opaque pixels in a boundary MB. Second, the partial SAD in (4-2) is rewritten as,

$$SAD_p(x, y;\ u, v) = \sum_{j=0}^{p}\sum_{i=0}^{q}\left|I_t\left(x+k_n, y+l_n\right) - I_{t-1}\left(x+k_n+u, y+l_n+v\right)\right| \qquad \textbf{(4-10)}$$

where

$$p \in \left\{\aleph \,\middle|\, 0,...,\alpha; \alpha = \text{integer division of } N/16 \text{ with truncation toward zero}\right\}$$

$$q = \begin{cases} 15 & , p \neq \alpha \\ N - 16\times\alpha & , p = \alpha \end{cases} \quad \text{and}$$

$\aleph$ is the set of Natural Number.

The design of our algorithm depends upon the clustering properties of the pixel matching errors. Hence the approach is general and it is expected to be useful for both software or hardware realization. Let us consider that for example, most computer architectures tend to be in favour of regular memory pattern access and execution [129]. It is true for the modern Intel processors, say for example, which provide Multimedia Extensions (MMX), Streaming Single Instruction Multiple Data Extensions (SSE) and

SSE2 technologies. These technologies offer a set of instructions for handling a large quantity of data in parallel efficiently. The MMX and SSE instruction set can compare eight bytes or eight pixel values from each of two blocks with a single instruction, thus accelerating the program by almost a factor of four or eight. The SSE2 instruction set can operate data with 16 pixels at one time effectively.

In order to make use of the advantage of clustering characteristic, we may also arrange to sort pixels row by row in a block for SAD_p accumulation. By using an identical reference value, m, we can calculate the expected absolute pixel error of a row of 16 pixels as follows, to determine the accumulating order.

$$p_{\exp} = \sum_{x=0}^{15} \left| I_t(x, l_n') - m \right| \qquad \textbf{(4-11)}$$

Because these instructions demand an increase in memory bandwidth, we have simulated three different situations to evaluate their performances, i.e. sorting a row of 4, 8 and 16 consecutive pixels in a block designated as CPME-PDS_4, CPME-PDS_8 and CPME-PDS_{16} respectively.

4.3.3 Analysis of the Overhead

From the above description, it is shown that the additional computation introduced by the CPME-PDS is the process to construct the adaptive index set for each target MB in the current frame or VOP.

The calculation of the reference value, m, as shown in (4-8) requires 255 additions and one division. For each opaque pixel, (4-9) shows that each expected absolute pixel matching error, $\left| p_{exp}(n) \right|$, needs one absolute operation and one subtraction. Hence, N absolute operations and N subtractions are required for the calculation of $\left| p_{exp}(n) \right|$ for each target MB. In the case of a boundary MB, 256 additional checkings are needed to ensure that only the opaque pixels are involved. To obtain the final adaptive index set, S,

a sorting process is required in step 5 of Section 4.3.2. Because the values of $\left|p_{exp}(n)\right|$ are integers ranging from 0 to 255, the counting sort [130] which has the complexity of $\boldsymbol{O}(N)$ is the most appropriate sorting algorithm in this situation. In general, it requires $2\times N$ increment/decrement operations and z-1 additions, where z is the largest integer in the data set being sorted. For a boundary MB, (4-10) shows that the formulation of SAD_p is different from that of an opaque MB. As shown in (4-10), the computation of α involves one division, and the computation of q when $p=\alpha$ needs one multiplication and subtraction.

Note that all multiplications and divisions mentioned above can be implemented with simple bitwise shift operations. In our analysis, however, each multiplication or division is counted and assumed to be equivalent to 8 additions for simplicity.

4.4 *Experiments*

In Section 4.3, we have proposed the CPME-PDS which makes use of the characteristics of clustered pixel matching errors to improve the searching efficiency of the conventional PDS. In this section, let us modify the adaptive PDS to become pixel gradient based adaptive PDS (PG-PDS) in order to compare saving in computation. A large amount of experimental work has been done. We describe the PG-PDS in brief in the next part. Approaches using representative pixels and adaptive matching scan (AMS-PDS) [96], conventional PDS, PG-PDS and Successive Elimination Algorithm (SEA) [99] were also implemented for the sake of comparison.

4.4.1 Pixel Gradients based Adaptive PDS (PG-PDS)

In the PG-PDS, an adaptive index set, S_{pg}, is obtained based on the magnitude of individual pixel gradient.

For each pixel in an opaque MB, let us express the magnitudes of x-directional gradients, G_x, and y-directional gradients, G_y, as,

$$\begin{aligned} |G_x(x,y)| &= |I_t(x,y) - I_t(x+1,y)| \qquad \text{where } x = 0,1,\ldots,14 \text{ ; } y = 0,1,\ldots,15 \\ |G_y(x,y)| &= |I_t(x,y) - I_t(x,y+1)| \qquad \text{where } y = 0,1,\ldots,14 \text{ ; } x = 0,1,\ldots,15 \end{aligned} \quad \textbf{(4-12)}$$

There are 15×16 = 240 gradient values for each direction. Hence, totally 480 gradient magnitudes need to be found for an opaque MB. These magnitudes are sorted in descending order with a counting sort. The S_{pg} is then established by extracting the pixel's position according to the order of the sorted gradient magnitudes. Obviously, each pixel must appear only once in S_{pg}. A proper checking procedure is needed to prevent double extraction of a pixel, because each pixel involves two directional gradient magnitudes. The adaptive index set, S_{pg}, is applied for the calculation of the partial SAD in (4-2) during the search in an outward spiral scanning.

There are some differences in the implementation of a boundary MB and an opaque MB. The total number of gradient magnitudes in a boundary MB depends on the number of opaque pixels and the shape in the MB. In calculating (4-12), if one of the involved pixel, $I_t(\cdot,\cdot)$, is a transparent pixel, the magnitude of the corresponding gradient is regarded as zero. We have also used pixel gradients in a row to perform row based sorting for comparison. Similar to the CPME-PDS, the average gradient of a row with 4, 8 and 16 consecutive pixels in a block has been used to determine the accumulating order of different rows in a SAD_p.

4.4.2 Experimental Results and Discussion

The analytical result in Section 4.3 suggests that two mean values can be used as the reference value, m, in the CPME-PDS. These are the mean of pixel values in the

target MB, m_1, and the mean of pixel values in the candidate MB of the initial searching point, m_2. In addition to these two mean values, we can also choose a third candidate, $m_3=128$, because we assume that R is a random variable which represents the pixel values in a MB and $m = \overline{R}$.

Table 4-1 compares the computational load of the CPME-PDS with these three reference values. The results show that all three values can successfully improve the efficiency of the conventional PDS. Among these three values, m_2 provides the least computational load. Hence, it confirms our suggestion that the mean of pixel values in the candidate MB of the initial searching point is used as the reference value in the CPME-PDS.

Table 4-1. Comparison of the average numbers of operations per MB of the CPME-PDS for different reference values, *m*.

	PDS	CPME-PDS		
		m_1	m_2	m_3
Video Sequences				
Children	188931	108120	99908	110958
Bream	219654	149539	139461	197122
Arbitrary Shaped Video Objects				
Football	299882	256443	232779	268849
Tabletennis	219509	159596	149103	173683

m_1 = The mean of pixel values in the target MB.
m_2 = The mean of pixel values in the candidate MB of the initial searching point.
m_3 = 128

To evaluate the performance of the clustered pixel matching error for adaptive partial distortion search (CPME-PDS), we implemented six algorithms: (i) the full-search algorithm (FSA), (ii) the conventional partial distortion search (PDS), (iii) the representative pixels and adaptive matching scan PDS (AMS-PDS), (iv) the pixel gradients based adaptive PDS (PG-PDS), (v) the Successive Elimination Algorithm (SEA) and (vi) the proposed CPME-PDS. The outward spiral scan was applied to all six algorithms to exploit the center-biased motion vector distribution characteristics. In addition, a median predictor was used as an initial searching centre to exploit the correlation in the motion field for all tested algorithms. The SEA uses the norm of each

search point to speed up the processing. The norm of each search point is calculated for frame-based sequences and opaque MBs of a video object by using a recursive method suggested in [99]. However, the recursive technique is not suitable for boundary MB of an arbitrarily shaped object. We need to calculate each norm during the search and the required operations are counted as computational load for searching in the realization. Hence, it is seen that the SEA may require much computation for the motion estimation of arbitrarily shaped objects in MPEG-4. Because AMS-PDS is an algorithm developed only suitable for block based motion estimation, experiments which involved arbitrary shaped video objects did not include AMS-PDS. The computational efficiency of the algorithms has been assessed in terms of the number of operations required for the searching. Each addition, subtraction, absolute or checking operation mentioned above was considered as one operation. Each multiplication or division was considered to be equivalent to 8 additions for simplicity. All these operations were counted in runtime during the experiments. Moreover, non-uniform memory access is the major disadvantage of these adaptive PDS algorithms. To evaluate the practical performance, we have also measured the execution time for motion estimation including the required overheads of all tested algorithms for comparison. We performed the experimental work on a standard desktop computer. The configuration of the platform was Intel P-III 600MHz desktop PC with 256M RAM and Windows 2000.

We used a large variety of video sequences and video objects for the evaluation. Sequences “Football”, “Table Tennis”, “Stefan”, “Salesman”, “Foreman”, “Grand Mother”, “Suzie” and “Trevor” were used as test sequences, while “News”, “Children”, “Bream” and “Goldfish” were used to test arbitrary shaped video objects. The format of the above video sequences and video objects are summarized in Table 4-2.

Table 4-2. Format of the tested video sequences and video objects.

Video Sequences		
Sequence	**Image format**	**num. of Tested Frames**
Football	352×240	209
Table Tennis	352×240	299
Stefan	352×240	299
Salesman	352×288	199
Forman	176×144	299
Grand Mother	176×144	299
Suzie	176×144	149
Trevor	176×144	149
Arbitrary Shaped Video Objects		
Video Object	**Source format**	**num. of Tested VOPs**
News	352×288	299
Children	352×288	299
Bream	352×288	299
Goldfish	352×288	299

According to the analysis in Section 4.2, we have shown that the prediction of pixel matching errors based on pixel gradients is only accurate near a minimum distortion position. Results in Figure 4-4 to Figure 4-6 justify our analysis. These figures show the average numbers of operations with different distances from the centre of a search window for the tested algorithms. The average number of operations at distance, *d*, is obtained by,

$$\text{Average no. of operations at distance, } d = \frac{\sum_{u=-d}^{d}\sum_{v=-d}^{d}\begin{pmatrix}\text{No. of operations}\\ \text{to calculate}\\ SAD_p(x,y;u,v)\end{pmatrix} - \sum_{u=-(d-1)}^{d-1}\sum_{v=-(d-1)}^{d-1}\begin{pmatrix}\text{No. of operations}\\ \text{to calculate}\\ SAD_p(x,y;u,v)\end{pmatrix}}{(2d+1)^2-[2(d-1)+1]^2}$$

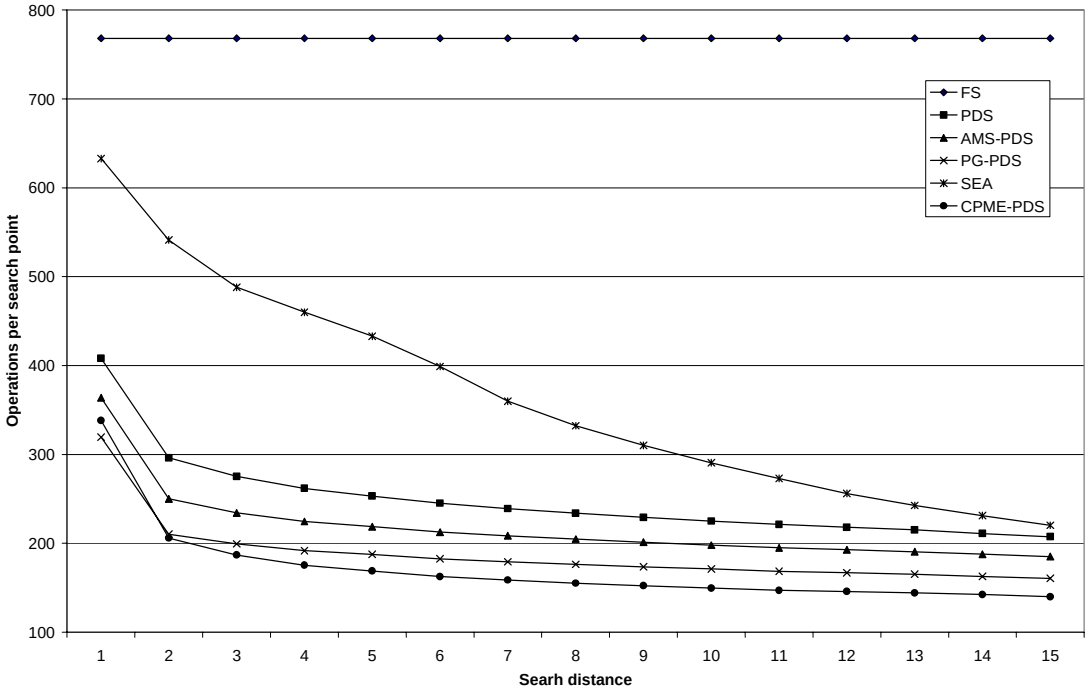


Figure 4-4. Comparison between the computational saving capability of the tested algorithms at different distances from the centre of a search window for "Table Tennis".

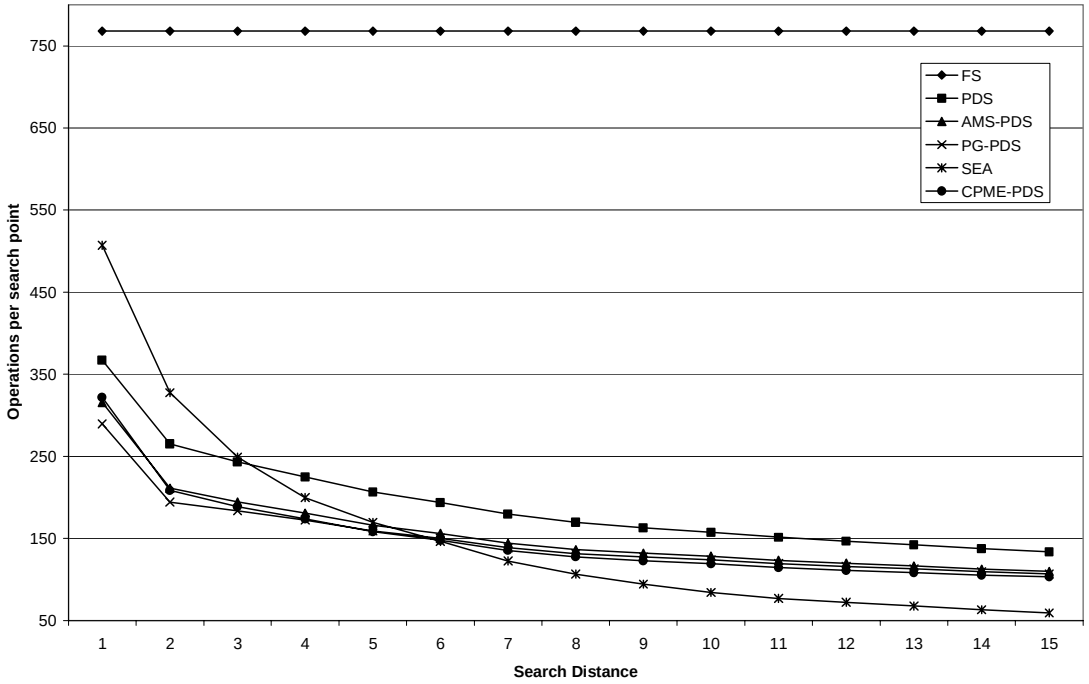


Figure 4-5. Comparison between the computational saving capability of the tested algorithms at different distances from the centre of a search window for "Grand Mother".

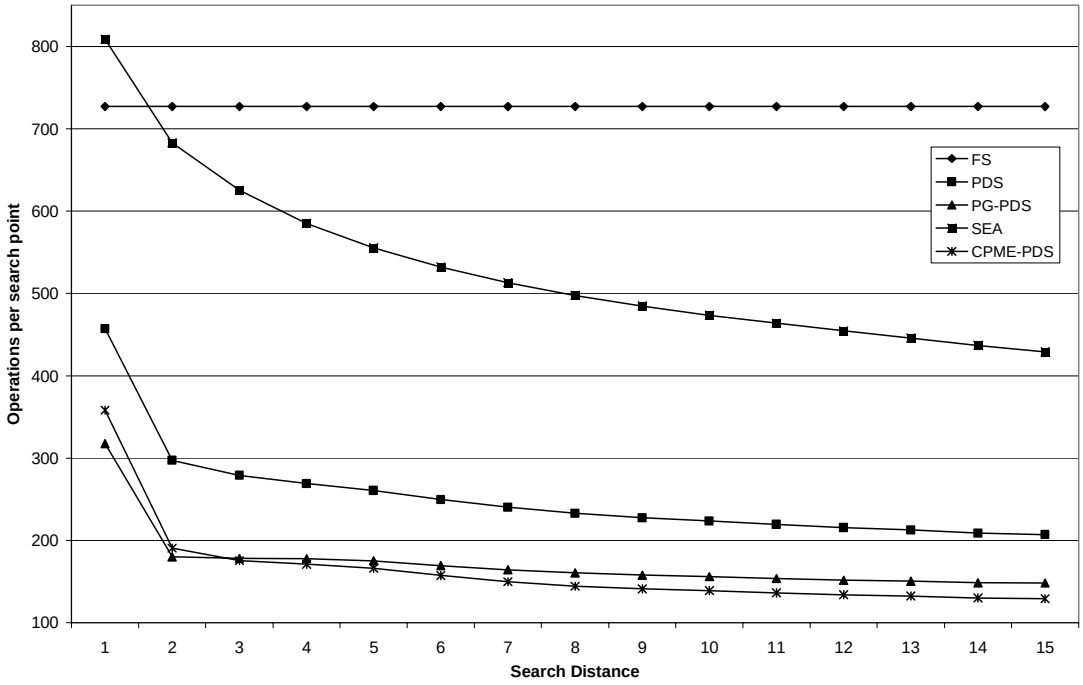


Figure 4-6. Comparison of the computational saving capability of the tested algorithms at different distances from the centre of a search window for "Bream".

For our initial analysis, we just counted the number of operations for the realization. All overheads such as the time for memory access, etc were not considered, since these overheads are usually machine dependent. Three typical results of the selected sequences, including the "Table Tennis", "Grand Mother" and "Bream", are provided in Figure 4-4 to Figure 4-6 respectively. In these Figures, algorithms with the least number of operations per search point at a specified distance, *d*, are shaded in grey. These results confirm our prediction that PG-PDS has a better ability to save computation when a search point is near the minimum distortion position. When the search point location is extended, the performance of CPME-PDS overrides that of PG-PDS. For the "Grand Mother" sequence, however, the SEA provides the best efficiency when the search distance is greater than six. Table 4-3 summarizes the results of all tested sequences. Entries in Table 4-3 give the search distances in which the corresponding algorithms have the least numbers of operations. On the whole, we can see that our approach (the CPME-PDS) always provides the largest range of search distance to have the best performance. This is particular true for object-based sequences and sequences

with high motion activities. On the other hand, the PG-PDS gives the largest computational saving among all tested algorithms with a short search distance, ranging from 0 to 4 on average. Furthermore, the SEA has the best performance for sequences related to video conferencing when the search distance is large enough. The number of operations of the AMS-PDS is about 4% on average smaller than that of the CPME-PDS for the sequence "Suzie" when the search range is smaller than 4. For the "Grand Mother" and "Foreman" sequences, the AMS-PDS is about 3% on average better than that of the CPME-PDS within a search range of 2. For "Football", "Salesman" and "Stefan" sequences, the number of operations of the AMS-PDS is about 2% smaller than that of the CPME-PDS within a unit search range.

Table 4-3. Summary of the computational saving ability of the tested algorithms for different sequences. The entries indicate the search distance at which the corresponding algorithms require the least number of operations.

Video Sequences	FSA	SEA	PDS	AMS-PDS	PG-PDS	CPME-PDS
Football	x	x	x	x	1 - 2	3 - 15
Tabletennis	x	x	x	x	1	2 - 15
Stefan	x	x	x	x	1 - 2	3 - 15
Salesman	x	x	x	x	1 - 3	4 - 15
Foreman	x	15	x	x	1 - 5	6 - 14
Grand mother	x	6 - 15	x	x	1 - 4	5
Suzie	x	9 - 15	x	x	1 - 3	4 - 8
Trevor	x	10 - 15	x	x	1 - 4	5 - 9
Video Objects						
News	x	x	x	invalid	1 - 3	4 - 15
Children	x	x	x	invalid	1 - 2	3 - 15
Bream	x	x	x	invalid	1 - 2	3 - 15
Goldfish	x	x	x	invalid	1 - 4	5 - 15

x – Indicates that the algorithms are not the most efficient one at all search distances.
invalid – AMS-PDS was not designed for the motion estimation of arbitrary shaped video objects.

Let us turn our attention to the overheads, such as sorting processes etc. Table 4-4 lists the average numbers of operations of these overheads for the tested algorithms. In Table 4-5, we have summarized the average number of total operations per search

point in a given search range, *D*, (i.e. $-D \le u, v \le D$). In this case, overheads are also included. In our implementation, quick sort was used as the sorting approach for AMS-PDS. Its complexity is *O(n log n)*, where *n* is equal to 16 in the case of the AMS-PDS algorithm. The selection of a sorting algorithm affects seriously the performance of an adaptive PDS especially if the search window is small. In order to prevent an under-evaluation of the AMS-PDS, the number of operations for its sorting process was not counted and this assumption is made in all of the latter discussion.

Table 4-4. Comparison of the overheads for different algorithms in terms of the average numbers of operations per MB.

Video Sequences	**FSA**	**SEA**	**PDS**	***AMS-PDS**	**PG-PDS**	**CPME-PDS**
Football	0	1158	0	1479	3686	1613
Tabletennis	0	1158	0	1479	3712	1618
Stefan	0	1158	0	1479	3723	1626
Salesman	0	1145	0	1479	3668	1597
Foreman	0	1272	0	1479	3678	1620
Grand mother	0	1272	0	1479	3658	1590
Suzie	0	1272	0	1479	3646	1583
Trevor	0	1272	0	1479	3668	1601
Video Objects						
News	0	1711	0	invalid	3373	1562
Children	0	2757	0	invalid	3196	1565
Bream	0	1614	0	invalid	3449	1588
Goldfish	0	1670	0	invalid	3199	1542

* The numbers of operations for sorting in the AMS-PDS were not counted in the experiments.
invalid – AMS-PDS was not designed for the motion estimation of arbitrary shaped video objects.

Table 4-5. Summary of the computational efficiencies in terms of operations of the tested algorithms for different sequences. The figures indicate the sizes of search window in which the corresponding algorithms require the least number of operations.

Video Sequences	FSA	SEA	PDS	*AMS-PDS	PG-PDS	CPME-PDS
Football	x	x	1 - 2	x	x	3 - 15
Tabletennis	x	x	1	x	x	2 - 15
Stefan	x	x	1	x	x	2 - 15
Salesman	x	x	1 - 2	x	x	3 - 15
Foreman	x	x	1 - 2	x	4 - 7	3, 8 - 15
Grand mother	x	10 - 15	1 - 2	x	x	3 - 9
Suzie	x	15	1 - 2	3	x	4 - 14
Trevor	x	x	1 - 2	3	x	4 - 15
Video Objects						
News	x	x	1	invalid	x	2 - 15
Children	x	x	1	invalid	x	2 - 15
Bream	x	x	1	invalid	x	2 - 15
Goldfish	x	x	1	invalid	x	2 - 15

* The numbers of operations for the sorting in AMS-PDS were not counted in the experiments.
x – Indicates that the algorithms are not the most efficient one at all search distances.
invalid – AMS-PDS was not designed for the motion estimation of arbitrary shaped video objects.

Table 4-5 summarizes the results of algorithms which require the least number of operations with the indicated search range. Generally speaking, CPME-PDS gives the best performance in a search range within 2 to 15 for nearly all sequences. The SEA achieves the best efficiency within a search range from 10 to 15 and 15 for the "Grand mother" and "Suzie" sequences respectively. It is interesting to point out that the PG-PDS offered the best efficiency in the middle search range, within 4 to 8, for the sequence "Foreman". In Table 4-3, we can see that the PG-PDS gives the best computational saving for *d* equal to 0 to 5 in "Foreman", but it requires the largest overheads as shown in Table 4-4. Hence, it needs more computational saving to outperform other algorithms, the situation of which is reflected in the table. The efficiency of CPME-PDS outperforms that of PG-PDS when the search range is extended sufficiently. In terms of the total number of required operations, we have

found that the performance of the PG-PDS is about 1.7% better than the CPME-PDS for only one sequence (the "Foreman" sequence) in the medium search range.

Table 4-6 demonstrates a comparison between the computational efficiency of the tested algorithms in a search window of 15 (i.e. $-15 \le u, v \le 15$). The computational efficiency was compared in terms of the average number of operations per MB and speed-up ratios. It shows that our algorithm, CPME-PDS, can successfully improve the computational efficiency of the conventional PDS and is the best among all other adaptive PDSs. In terms of speed-up ratios, it can achieve a speed-up ranging from 3 to 9 times of the FSA. The SEA gives worse performance as compared to our algorithm, CPME-PDS, for most sequences but achieves better efficiency for sequences on vide conferencing, such as the "Grand mother", and "Suzie".

Table 4-6. Average numbers of total operations per MB of the tested algorithms in a search window with a search range equal to 15 (i.e. $-15 \le u, v \le 15$).

Video Sequences	FSA		SEA		PDS		*AMS-PDS		PG-PDS		CPME-PDS	
	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio
Football	738048	1.00	351897	2.10	299882	2.46	279341	2.64	263057	2.81	232779	3.17
Tabletennis	738048	1.00	290482	2.54	219509	3.36	194414	3.80	170269	4.33	149103	4.95
Stefan	738048	1.00	304156	2.43	241830	3.05	206729	3.57	191230	3.86	169652	4.35
Salesman	738048	1.00	172550	4.28	160614	4.60	140468	5.25	126625	5.83	111693	6.61
Foreman	738048	1.00	129522	5.70	153344	4.81	123309	5.99	110916	6.65	106310	6.94
Grand mother	738048	1.00	98119	7.52	157032	4.70	129139	5.72	126619	5.83	121658	6.07
Suzie	738048	1.00	118877	6.21	170286	4.33	137157	5.38	135321	5.45	121301	6.08
Trevor	738048	1.00	82068	8.99	110638	6.67	93004	7.94	89113	8.28	81240	9.08
Video Objects												
News	689274	1.00	321915	2.14	188931	3.65			147191	4.68	138052	4.99
Children	653006	1.00	478403	1.36	188931	3.46			116043	5.63	99908	6.54
Bream	698661	1.00	464301	1.50	219654	3.18			155870	4.48	139461	5.01
Goldfish	663847	1.00	353271	1.88	226663	2.93			151615	4.38	140129	4.74

Let us evaluate the execution time of all algorithms. The evaluation takes into the account of both computation loads of the algorithms and their overheads. This is, to some extent, CPU dependent. Table 4-7 and Table 4-8 compare the execution time per frame or per VOP of the algorithms with a search range of 15. Table 4-7 clearly shows

that the improvements of all PDS algorithms are reduced. On average, the speedup ratios in terms of the number of operations are decreased by about 33%, 40%, 52% and 48% for the PDS, AMS-PDS, PG-PDS and the CPME-PDS respectively. The SEA is only degraded by about 9%. This phenomenon is caused by two factors. The first factor is that all adaptive PDS algorithms suffer from the problem of non-uniform memory access. Among these three adaptive PDS, the AMS-PDS has the minimum degradation because it makes use of pixel gradients to determine the sorting of rows or columns in a MB. When the column scanning is used in AMS-PDS, non-uniform memory access problem is occurred. On the other hand, during the process of row scanning in the AMS-PDS algorithm, it accumulates pixel errors of consecutive pixels in a row. In our experimental work, we used different implementation techniques for these two situations, such that the non-uniform access problem of AMS-PDS becomes less severe. The second factor occurs in all PDS algorithms. The pipeline structure of modern CPUs improves greatly the performance of computing consecutive data. However, the pipeline flow is interrupted when cache misses or exceptions occur. For PDS, it suffers inherently from branch miss-prediction penalty; say for example it happens in Intel CPUs. Except the “Table Tennis” sequence and the VOs, the SEA requires less computational time as compared to that of the PDS, while the SEA actually requires more operations. Even though the presence of these two drawbacks, our experimental results show that the CPME-PDS gives the best performance for sequences containing high motion activities and the video objects.

Table 4-7. The execution time (seconds) per frame or per VOP of the tested algorithms in a search window with a search range equal to 15 (i.e. $-15 \le u, v \le 15$).

Video Sequences	FSA		SEA		PDS		AMS-PDS		PG-PDS		CPME-PDS	
	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio
Football	1.92	1.00	0.95	2.02	0.96	2.00	0.97	1.98	1.12	1.71	0.93	2.06
Tabletennis	1.92	1.00	0.89	2.16	0.79	2.43	0.77	2.49	0.86	2.23	0.73	2.63
Stefan	1.92	1.00	0.83	2.31	0.84	2.29	0.80	2.40	0.95	2.02	0.78	2.46
Salesman	2.30	1.00	0.60	3.83	0.79	2.91	0.77	2.99	0.97	2.37	0.77	2.99
Foreman	0.58	1.00	0.11	5.27	0.19	3.05	0.18	3.22	0.23	2.52	0.17	3.41
Grand mother	0.58	1.00	0.09	6.44	0.19	3.05	0.18	3.22	0.24	2.42	0.18	3.22
Suzie	0.57	1.00	0.11	5.18	0.20	2.85	0.19	3.00	0.24	2.38	0.18	3.17
Trevor	0.58	1.00	0.08	7.25	0.16	3.63	0.16	3.63	0.21	2.76	0.16	3.63
Video Objects												
News	1.05	1.00	0.51	2.05	0.45	2.32			0.43	2.43	0.38	2.75
Children	0.60	1.00	0.43	1.39	0.27	2.21			0.20	2.99	0.18	3.32
Bream	0.81	1.00	0.53	1.52	0.36	2.24			0.34	2.37	0.29	2.78
Goldfish	1.51	1.00	0.83	1.82	0.74	2.04			0.61	2.48	0.54	2.80

Table 4-8. The execution time (second) per frame or per VOP of the row-based adaptive PDS algorithms in a search window with a search range equal to 15 (i.e. $-15 \le u, v \le 15$).

Video Sequences	PG-PDS$_4$		PG-PDS$_8$		PG-PDS$_{16}$		CPME-PDS$_4$		CPME-PDS$_8$		CPME-PDS$_{16}$	
	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio
Football	0.88	2.18	0.91	2.11	0.86	2.23	0.80	2.40	0.83	2.31	0.80	2.40
Tabletennis	0.61	3.15	0.64	3.00	0.62	3.10	0.61	3.15	0.64	3.00	0.62	3.10
Stefan	0.71	2.70	0.74	2.59	0.70	2.74	0.64	3.00	0.68	2.82	0.65	2.95
Salesman	0.66	3.48	0.69	3.33	0.67	3.43	0.60	3.83	0.62	3.71	0.60	3.83
Foreman	0.15	3.87	0.15	3.87	0.15	3.87	0.14	4.14	0.14	4.14	0.14	4.14
Grand mother	0.16	3.63	0.16	3.63	0.16	3.63	0.15	3.87	0.16	3.63	0.15	3.87
Suzie	0.16	3.56	0.17	3.35	0.17	3.35	0.15	3.80	0.16	3.56	0.16	3.56
Trevor	0.13	4.46	0.14	4.14	0.13	4.46	0.13	4.46	0.13	4.46	0.12	4.83
Video Objects												
News	0.37	2.83	0.39	2.68	0.38	2.75	0.35	2.99	0.36	2.90	0.35	2.99
Children	0.22	2.72	0.23	2.60	0.23	2.60	0.19	3.15	0.20	2.99	0.20	2.99
Bream	0.30	2.69	0.32	2.52	0.31	2.60	0.28	2.88	0.29	2.78	0.28	2.88
Goldfish	0.59	2.56	0.62	2.44	0.61	2.48	0.54	2.80	0.56	2.70	0.55	2.75

In addition to the approach using pixel-based sorting, we have also tested the row-based sorting approach for the adaptive PDS algorithms by making use of pixel-gradients (PG-PDS) or clustering characteristics (CPME-PDS). This is regarded as a

compromise between the sorting approach and the problem of non-uniform memory access. This approach is not valid for other algorithms in our discussion. Table 4-8 summarizes the performances in terms of the execution time per frame or per VOP. The number of consecutive pixels in a sorted row is indicated by the subscript *r*, such as PG-PDS$_r$ and CPME-PDS$_r$. Note that after this modification, the PG-PDS$_r$ is very closed to the approach of AMS-PDS. Comparing Table 4-8 with Table 4-7, it shows that the row-based sorting can efficiently improve both PG-PDS and CPME-PDS. It confirms that the approach using clustering characteristics is more effective than the pixel gradient for adaptive PDS technique. It gives better performance as compared to the gradient-based approach for all tested sequences. Table 4-7 also indicates that the SEA can attain superior results for video sequences on conferencing. Table 4-8 shows that CPME-PDS$_4$ is able to achieve the best performance for the remaining sequences except the "Children", for which the pixel-based CPME-PDS gives the best computational time. On average, the SEA and CPME-PDS$_4$ provide 3.42 and 3.38 times speedup when comparing to the FSA. These experimental results confirm that CPME-PDS$_4$ is most suitable for motion estimation of sequences containing high motion activities and arbitrarily shaped video objects.

4.5 *Conclusions*

We have proposed an adaptive partial distortion search algorithm entitled as the Clustered Pixel Matching Error for adaptive Partial Distortion Search (CPME-PDS). The algorithm makes use of the phenomenon that pixel matching errors in a MB with similar magnitude tend to appear together in a cluster in natural video sequences. We have demonstrated that this is a popular phenomenon for relatively large search windows. According to this phenomenon, we have found that both mean of pixel values in a target MB and mean of pixel values in a candidate MB are good references to

predict the magnitude of each pixel matching error in the target MB. Hence, the mean of pixel values in the initial candidate MB at the centre of a search window has been used to calculate a reference value and to construct an adaptive index set. As a result, the pixel matching error with larger magnitude can be accumulated to the SAD_p sooner than others and the SAD calculation can be terminated at an early stage. We have evaluated the efficiency of the CPME-PDS in two measures, the total number of operations and the execution time per frame or per VOP in motion estimation.

In terms of the number of operations, our experimental results show that for a small maximum allowable search range, such as $D = 1$ or some cases of $D = 2$, the conventional PDS is still the best algorithm due to the overheads of fast algorithms. However, in a reasonably longer maximum allowable search range, $D = 2$ to $D = 15$, the computational efficiency of the CPME-PDS outperforms other algorithms for coding sequences with high motion activities and arbitrarily shaped objects. In the case of a large search window, $D = 15$, our experimental results show that the CPME-PDS can have a speed-up of 3 to 9 as compared with FSA, depending upon the contents of the coded video sequences. Hence, the proposed CPME-PDS is generally the best among all algorithms. The major advantages of CPME-PDS are its high efficiency and conceptual simplicity. Comparing to other adaptive PDS, it requires less overheads.

When motion estimation time per frame or per VOP is used for evaluation, the performance of CPME-PDS is degraded slightly due to the problem of non-uniform memory access. Nevertheless, the CPME-PDS is still able to provide the best efficiency for sequences with high motion activities and video object encoding. We have modified the CPME-PDS into a row-based algorithm in order to remedy the non-uniform memory access problem. For example, a row of 4 consecutive pixels with larger prediction errors is accumulated to the SAD_p sooner than other rows. Experimental results show that the conventional SEA provides a speed-up of about 3.42 times when

comparing to the FSA, and it performs the best for video conferencing sequences. Meanwhile, our row-based CPME-PDS, CPME-PDS$_4$ can speed up the search for about 3.38 times as compared to the FSA on average, and furthermore the CPME-PDS$_4$ outperforms all other tested algorithms (including the SEA) for encoding sequences with high motion activities and arbitrarily shaped video objects.

Chapter 5. New Mixed Spatial-DCT-based Coding of the Motion Prediction Error Frame of Video Objects and Frames

5.1 *Introduction*

In the past decade, various waveform coding techniques such as the ones using the Discrete Cosine Transform (DCT), subband/wavelet and vector quantization have been developed for video coding. It is well known that subband/wavelet is superior to DCT in still image coding. DCT based coding is still popular and researchers are in favour of it for video coding [124]. For example, the DCT is widely used in modern video compression standards, such as the ITU-T H.263, the ISO MPEG-1, the ISO MPEG-2 and the ISO MPEG-4, to achieve high compression efficiency. However, using the DCT based coding to perform compression, the motion compensated prediction error is far from optimal. The statistical properties of the errors are different from that of natural images. The redundancies of errors which are synthetically generated by the process of motion compensation cannot be exploited successfully by the DCT [117,118]. Moreover, the BMC assumes that the motion between successive frames is purely translational. This is not true for most of the real world video sequences. All factors including deformation of foreground objects, non-translational motion and irregular light variation, will make the BMC fail. The prediction error is generally concentrated in a clustered portion of the image even if the FSA is employed. It leads to a scattering of the DCT coefficients and makes the compression inefficient.

To resolve the inefficiency of coding the prediction errors in the DCT domain, in this chapter, we introduce a new Mixed Spatial-DCT-based Coding technique (MSDC). The MSDC divides a prediction error MB into two components. Each component is characterized by its own spatial correlation. One component is then coded by using the

binary bit plane coding and variable length coding techniques (VLC), and the second component is coded by using the traditional DCT-based method. Results of our experimental work show that the MSDC can achieve a better compression efficiency for the prediction error when compared with the traditional DCT-based coding technique. It can effectively compress the prediction errors of arbitrary shaped video objects and video sequences with moderate to high motion activities. Furthermore, enhanced video codecs can easily be obtained by embedding the MSDC for high quality video applications.

In the rest of this Chapter, we firstly illustrate and discuss the observation of spatial characteristics of the motion compensated prediction errors. Problems and properties related to the Full Search and other fast search algorithms are discussed in Section 5.2. Making use of our observations, we will develop a new Mixed Spatial-DCT-based Coding Scheme (MSDCS). Section 5.3 describes the proposed algorithm in details. The performance of the proposed algorithm is then compared with the traditional DCT-based coding in terms of the compression efficiency and execution time using our computer. A large variety of video sequences and video objects have been used for the evaluation. Section 5.4 gives the details of our experimental results and the analysis. Finally, some concluding remarks are given in Section 5.5.

5.2 *Characteristics of the motion compensated prediction error*

Making use of motion compensation, modern video coding standards usually can attain efficient video compression. In most cases, it is also assumed that the DCT-based coding is applicable to process the prediction error if proper motion estimation is used in the encoder side. However, as noted before, a possible failure of BMC comes from the fact that there is a special distribution of the prediction error. It destroys the energy

compaction ability of the DCT. Hence, investigating the distribution of the BMC prediction error is important to us for improving the compression efficiency of an encoder.

MPEG-4 supports coding of arbitrary shaped objects with a set of tools. An arbitrarily shaped video object plane (VOP) is partitioned into a number of MBs. The MBs, which only partially filled with opaque pixel values, are called partial MBs. For a reference arbitrarily shaped VOP, its transparent region must be padded to form a rectangular shape, such that block-based motion estimation/compensation of the partial MBs can be processed efficiently. MPEG-4 makes use of a repetitive padding technique [52] to pad an arbitrarily shaped VOP. The transparent region of the reference VOP is padded by replicating the boundary pixel values of the VOP towards the exterior. It results in a special constant intensity line patterns. This technique can effectively compensate the motion of video objects with low motion activity. However, for objects with moderate to high motion activities, it is unreliable to use padding to predict the pixel values of a partial MB. Consequently, the block-based motion compensation is not efficient and the prediction error concentrates at this padded region. Moreover, inaccurate video object segmentation may also make the padding inappropriate. A segmented video object plane which contains part of a moving object and part of a still background is generally the result of inaccurate segmentation. Hence, replicating the wrong edge pixel values in the repetitive padding cannot compensate the motion of video objects efficiently. Examples of inaccurate video object segmentation are shown in Figure 5-1. Figure 5-1(a) illustrates a repetitively padded VOP, the “Goldfish”. The padded region is clearly unable to compensate the motion of the objects due to luminance variation at the edge region of the video object. Figure 5-1(b) is an example of inaccurately segmented object in the sequence “Stefan”. The edge of the object is

segmented with part of the still background and thus inefficient padding result is available.

(a) **(b)**

Figure 5-1. Examples of inaccurate video object segmentation (a) a repetitively padded "Goldfish" and (b) an inaccurately segmented object in sequence "Stefan".

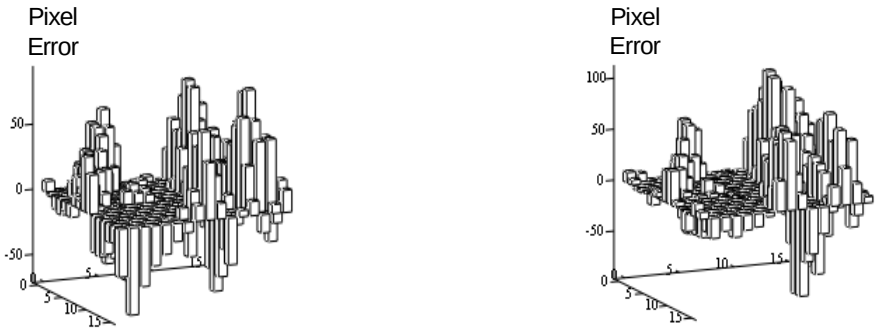
Figure 5-2(a) depicts a typical example, for which the prediction errors concentrate at the padded region in a MB. In this simulation we have used a fast full search algorithm, the partial distortion search (PDS), for motion estimation. The situation of using other fast search algorithms is even more serious. The prediction errors of the MB, when MVFAST was used in our simulation, are shown in Figure 5-2(b). It is found that a larger region that contains clustered prediction errors are found when compared to that of the result as shown in Figure 5-2(a). Furthermore, the magnitudes of the errors are also larger in this case.

(a) FS **(b) MVFAST**

Figure 5-2. Examples of clustered errors in a motion compensated prediction MB of the video object "GoldFish" by (a) FS and (b) MVFAST.

The clustering effect of prediction errors occurs not only in partial MBs. It is well known that natural scenes are dominated by low frequency components. Furthermore, a natural scene often consists of different correlated pixel regions. These low frequency regions are separated by edges. Motion estimation is a matching of a target MB with a reference frame. Hence, prediction errors of motion estimation can be regarded as regions due to the differences between the smooth parts of the target MB and the reference MB. In the meantime, the correlated regions are partitioned by the edges of the two MBs. Hence in this analysis, we can assume that the clustered prediction error is a characteristic of motion prediction errors. We further make a postulate that an error MB at a local minimum containing some regions with small errors and other regions with large errors. On the other hand, the error MB at a global minimum is a MB where large error regions shrink to some tiny portion or disappeared. However, large clustered prediction errors also occur frequently in sequences with high motion activity, even though full search algorithm is employed. This undesirable result is due to poor prediction of BMC of irregular motion activities.

Figure 5-3(a) and Figure 5-3(b) are examples of MB's prediction errors in which the PDS and MVFAST are used as motion estimation for sequence "Football", respectively. These examples demonstrate that the clustered prediction errors give significant effect for frame based video coding. As mentioned before, the distribution of these errors scatters the DCT coefficients and increases the bit-rate of DCT based coding results.



(a) FS **(b) MVFAST**

Figure 5-3. Examples of clustered errors in a motion compensated prediction MB of the sequence "Football" by (a) FS and (b) MVFAST.

5.3 *Proposed Algorithm*

The clustered error decreases the efficiency of DCT based compression. Fortunately, errors join together in clusters. This fact reflects that some spatial redundancies remain in this prediction error. We make use of these remaining redundancies to improve the compression efficiency of a DCT-based encoder. In the following, a detailed description of our proposed algorithm is given.

5.3.1 Separation of the prediction error MB into two components

Our algorithm starts by separating a prediction error into two components. Each component is then characterized by its own spatial correlation. Hence, we can apply different compression techniques to these two components based on their spatial characteristics. According to our discussion in Section 5.2, we know that large prediction errors tend to cluster together. This clustering property motivated us to separate large prediction errors from an error block. When the magnitudes of the remaining error block are small and correlated, the traditional DCT coding can be applied directly. Moreover, the clustering property of error components enables us to treat them as an arbitrary shaped object plane. We suggest using the context-based

arithmetic encoding (CAE) [45,46], which is a technique included in the MPEG-4 for binary alpha plane coding, to code separated error components.

A MB consists of four 8×8 pixels blocks. To separate the prediction errors, $E_i(x,y)$ of each block into two components, $Ec_i(x,y)$ and $Ed_i(x,y)$, we use a simple approach which involves a thresholding accompanied with some modulus operations. Let us define,

$$E_i(x,y) = Ec_i(x,y) + Ed_i(x,y), \qquad i = 0 \ldots 3 \tag{5-1}$$

where $$Ed_i(x,y) = \begin{cases} E_i(x,y) & if\ |E_i(x,y)| < TS \\ \langle E_i(x,y) \rangle_{TS} & otherwise \end{cases} \tag{5-2}$$

$$Ec_i(x,y) = E_i(x,y) - Ed_i(x,y)$$

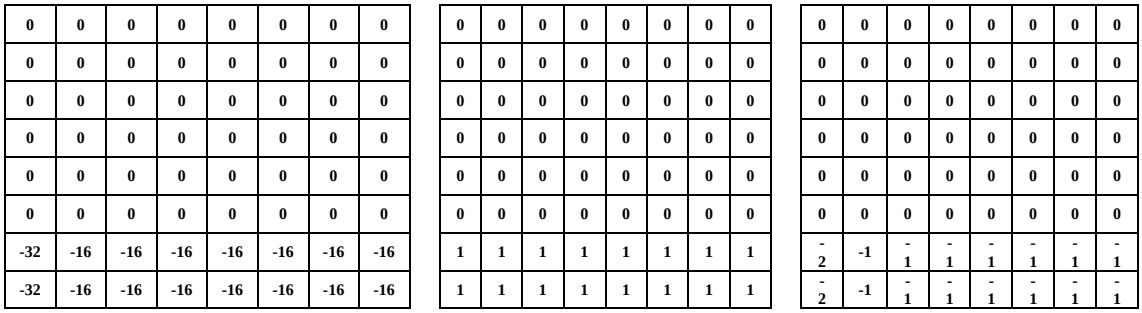
TS is a pre-defined threshold to detect peak errors and acts as a scaling factor for $Ec_i(x,y)$.

$\langle \cdot \rangle$ denotes modulus operator

It is obvious that the absolute values of error components in $Ed_i(x,y)$ are smaller than the pre-defined threshold value, *TS*. This separation cuts down abrupt peak errors from $E_i(x,y)$, and thus the traditional DCT becomes more efficient to code the resulting $Ed_i(x,y)$. We can then separate coding of the second error component, $Ec_i(x,y)$ into two parts, namely (i) shape and position of clustered errors, and (ii) magnitudes of the errors. The values of $Ec_i(x,y)$ are multiples of the threshold, *TS*, because of the modulus operation in eqn. (5-2). Hence, we only need to code $Ec_i'(x,y)$, that is the quotients of $Ec_i(x,y)$ divided by *TS*.

or $$Ec_i'(x,y) = Ec_i(x,y)/TS \tag{5-3}$$

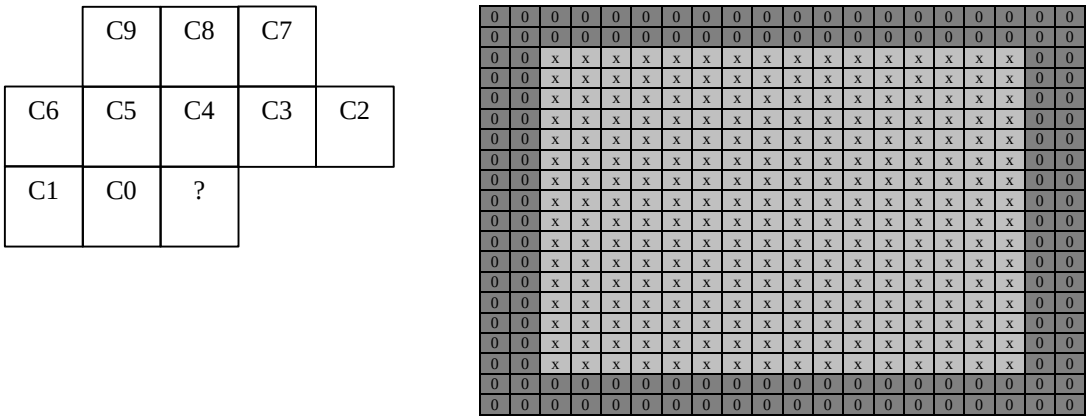
Figure 5-4 demonstrates the shape of a sample $Ec_i(x,y)$ and it's corresponding quotients, $Ec_i'(x,y)$, for the coding to be described below.



(a) $Ec_i(x,y)$ **(b) Shape of $Ec_i(x,y)$** **(c) $Ec'_i(x,y)$**

Figure 5-4. The decomposition of an example Eci (x,y) into it's shape for CAE and the corresponding quotients, Ec'i (x,y) for VLC coding.

5.3.2 Compression of the clustering regions with context-based arithmetic encoding (CAE) and variable length coding (VLC)



(a) **(b)**

Figure 5-5. (a) The INTRA template for context construction in CAE. (b) Binary alpha block with extended border, "x"s denote contents of the shape.

We use context-based arithmetic encoding (CAE) [45,46] to code the shape of the separated error component, $Ec_i(x,y)$, in a MB, which is adopted in MPEG-4 for binary alpha plane coding. A brief introduction of the CAE is given below. The shape of $Ec_i(x,y)$ is compressed with CAE in the intra mode. A block with 16×16 pixels, which contains the shape of four $Ec_i(x,y)$ blocks, is called a binary alpha block (BAB). A template of ten pixels is used to define a context number, $C = \sum_{k=0}^{9} c_k \times 2^k$. Figure 5-5(a) illustrates the intra template for the context number construction. The context number is used to index a probability table provided in MPEG-4 for a current pixel. Then, we use

the indexed probability to drive an arithmetic encoder. When encoding a BAB, a border of width equal to 2 is extended from the current BAB for context number construction. The pixel positions in the extended border are padded with zero as depicted in Figure 5-5(b) in our algorithm.

The magnitudes of errors in $Ec_i(x,y)$ are firstly divided by *TS* as shown in Figure 5-4(c). We encode the quotients with variable length codes, and raster scan is used to scan the MB from left to right and top to bottom. The code words for our VLC are listed in Table 5-1. Actually Table 5-1 is part of the variable length codes for DCT coefficients in MPEG-2 with the only difference that the meanings of code words have been modified.

Table 5-1. Variable length codes for E'2(x,y).

$E'_2(x,y)$	Variable length code
1	1s
2	11s
3	011s
4	0100 s
5	0101 s
6	0010 1s
7	0011 1s
8	0011 0s
9	0001 10s
10	0001 11s
11	0001 01s
12	0001 00s
13	0000 110s
14	0000 100s
15	0000 111s
The last bit 's' denotes the sign of a value	

5.3.3 Mode determination between Mixed Spatial-DCT-Based Coding and traditional DCT-based coding

The proposed Mixed Spatial-DCT-Based Coding technique is a way to exploit the redundancies remained in the error MBs. These redundancies are due to inefficient block based motion estimation and the repetitive padding technique. The traditional DCT-based coding is still useful for some MBs. In other words, the MSDC provides an additional mode to code the motion compensated prediction error for each MB. We add a mode bit to distinguish between the traditional DCT mode and the MSDC mode in the resulting bitstream.

In order to exploit the benefits from these two modes, one simple method is to test exhaustively the efficiency of the traditional DCT coding and that of the MSDC for each MB. Nevertheless, the computational load introduced in this exhaustive comparison is a major problem that we need to resolve. It is obvious that we do not need to test a MB when all pixel errors in the MB are smaller than the threshold, *TS*. Experimental results show that this simple technique can save about 21% of the computational load caused by exhaustive testing.

It is always attractive for us to study the possibility of improving the efficiency of the MSDC for a MB. If we can predict the possibility accurately, we need only to perform the efficiency test to MBs that the MSDC can possibly be able to improve the coding efficiency. The spatial distribution of peak pixel errors and their magnitudes are factors which affect improvement of the MSDC. It is because these two factors directly affect the required number of bits of the CAE and variable length coding. This consideration leads us to simply assume that the possibility of using the MSDC is proportional to the sum of quantized peak errors, $SQPE_i$, of each block.

$$SQPE_i = \sum_{x=0}^{8}\sum_{y=0}^{8}\left|E_i(x,y)\right| / TS\,, \qquad i = 0\ldots3 \tag{5-4}$$

Hence, only if the corresponding $SQPE_i$ is greater than or equal to a speed-up threshold, p, the block of error, $E_i(x,y)$, will be separated into two components using (5-1) to (5-3). In this case, we can avoid to compute the efficiency comparison between the DCT-based coding and the MSDC if the condition in (5-5) is fulfilled.

$$SQPE_i < p \qquad \forall i \tag{5-5}$$

Apparently, the decrease in compression efficiency is a result of increasing p. If p is set to 1, this implies to obtain identical results as the exhaustive test. The mode which requires a small number of bits for a MB is the resulting mode in our implementation. The influence of the speed-up factor p ranging from 1 to 4 has been examined in our experiments. Details of our experimental work will be described in Section 5.4.

5.3.4 Mixed Spatial-DCT-Based Coding Scheme (MSDCS)

The above description gives some guidelines for our proposed Mixed Spatial-DCT-Based Coding Scheme (MSDCS). The scheme makes use of the MSDC to exploit the redundancies remained in appropriate error macroblocks and the mentioned mode decision technique to speed up the process. The block diagram of the MSDCS is given in Figure 5-6. The procedure is summarized as shown below.

Step 1: Code the input motion compensated prediction error, $E_i(x,y)$ in a MB with traditional DCT-based coding, and count the required number of bits for the MB as B_{DCT}.

Step 2: Calculate $SQPE_i$ of each block with (5-4). If the condition in (5-5) is satisfied, go to step 12.

Step 3: Calculate $Ed_i(x,y)$ from $E_i(x,y)$ according to (5-2). Set threshold value TS to 16 in our simulation which has been determined experimentally.

Step 4: Calculate $Ec_i'(x,y)$ using (5-3).

Step 5: Create the shape of clustered errors with the four $Ec_i'(x,y)$s in the MB.

Step 6: Code the shape of the clustered errors with CAE, which is described in Section 5.3.2, and count the required number of bits as B_{CAE}.

Step 7: Code the magnitudes of $Ec_i'(x,y)$s with VLC using Table 5-1, and count the required number of bits for the MB as B_{VLC}.

Step 8: Code the $Ed_i(x,y)$ with the traditional DCT-based coding, and count the required number of bits for the MB as $B_{m\text{-}DCT}$.

Step 9: Calculate the total required number of bits for the MSDC, B_{MSDC}

where $B_{MSDC} = B_{CAE} + B_{VLC} + B_{m-DCT}$.

Step 10: If $B_{DCT} \le B_{MSDC}$, go to step 12.

Step 11: Set the mode bit to represent MSDC mode, including the mode bit, and use the results of step 6, 7 and 8 to form the resulting bitstream; then go to step 1 for next MB.

Step 12: Set the mode bit to represent the DCT mode, and make use of the result of step 1 for bitstream formation, and go to step 1 for next MB.

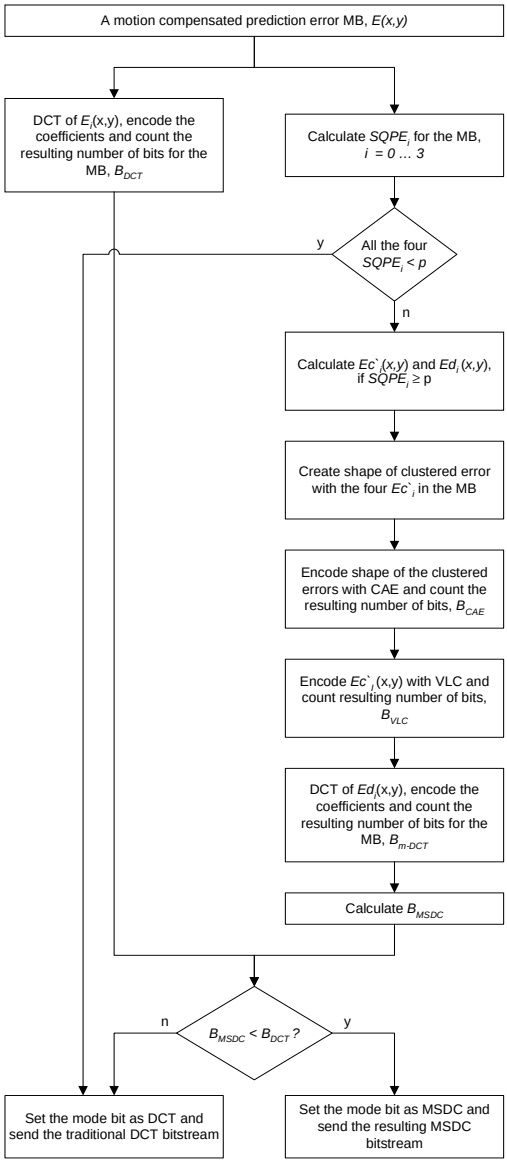


Figure 5-6. Block diagram of the proposed Mixed Spatial-DCT-based Coding scheme.

5.4 *Experiments*

In Section 5.3, we have proposed the MSDCS which exploits the redundancies remained in the error MBs to improve the efficiency of the traditional error coding. In order to evaluate the coding efficiency of the MSDCS, a large amount of experimental works have been done. We used a large variety of video sequences and video objects for the evaluation. Sequences “Table Tennis”, “Football”, and “Templete” were used as test sequences, while “Goldfish”, “Weather”, “Segmented Stefan”, and “Children” were used to test coding performance for arbitrary shaped video objects. The format of the above video sequences and video objects are summarized in Table 5-2.

Table 5-2. Format of the tested video objects and sequences

Arbitrary Shaped Video Objects		
Video Object	**Source format**	**Number of Tested VOPs**
Goldfish	352×288	209
Weather	352×288	299
Segmented Stefan	352×240	299
Children	352×288	299
Video Sequences		
Sequence	**Image format**	**Number of Tested Frames**
Table Tennis	352×240	299
Football	352×240	209
Tempete	352×288	299

The optimization of a coding system is essentially a multi-dimensional problem. The key issues concerned in this problem are: bit-rate, quality (PSNR), speed-up (or computational gain), algorithmic complexity, memory size and bandwidth. There is always a trade-off among all these five key factors. It is the reason that fast motion estimation algorithms have attracted a lot of attention in the past decade. In order to evaluate the compatibility of the MSDCS, we have compared the coding results of our proposed scheme and the traditional DCT-based prediction results of the PDS which is a fast full search algorithm and the MVFAST which is a lossy fast algorithm. For the optional mode of the MVFAST, no early elimination of search was used in all of our experiments. Hence, the motion estimation of a MB can only be terminated if a local/global minimum has reached.

We performed the simulations on a standard desktop computer. The configuration of the platform was Intel P-III 600MHz desktop PC with 256M RAM and Windows 2000. The software platform used and modified was based on MPEG-4 VM14.0, Microsoft C++ implementation package [131,132]. The coding type was “IPPP…” for

all tested sequences and video objects. Each macroblock in P-Frame/P-VOP was coded in inter mode with one motion vector only. The motion vector was estimated by the PDS or MVFAST with half pixel accuracy. In our simulation, no frame skipping was applied for all experiments.

To determine the value of TS, we have performed a set of experiments to find the separation of large errors from an error block in order to have the best rate distortion performance. At the beginning, two thresholds T and S have been used to separate large errors from an error block and to scale the large error components. The separation is formulated as shown below,

$$E(x, y) = Ec(x, y) + Ed(x, y)$$

$$Ed(x, y) = \begin{cases} E(x, y) & if\ |E(x, y)| \le T \\ \langle E(x, y) - T \rangle_S & otherwise \end{cases}$$

$$Ec(x, y) = E(x, y) - Ed(x, y)$$

The experiments were done by using with T ranging from 16 to 52 and S ranging from 16 to 36, all with an increment of 4. We have found that if T and S = 16, this results in a degradation of about 0.03dB PSNR performance in the best video quality and the PSNR is 49.48dB. However, it provides 10% decrease in bitrate, from about 195 to 177.5 kbits per VOP when comparing to the traditional DCT technique. At a quality of about 42.8dB PSNR, with T and S = 16 only about 0.2dB PSNR degradation results, whilst about 13% of the bitrate is saved. The required bitrate reduces from 92 to 81.5kbits per VOP. At a video quality of about 37.78dB PSNR, the case with T and S = 16 not only provides about 6% saving in bitrate and it also results in 0.05db PSNB improvement. Hence, we have grouped T and S to be a single threshold TS and set it to 16 in all of our experiments.

Table 5-3. Comparison of the additional execution time per frame with different speed up factor p in the MSDCS.

	PDS	MVFAST	Additional computational time of the MSDCS				
			p = 0	p = 1	p = 2	p = 3	p = 4
Arbitrary Shaped Video Objects							
Goldfish	0.334	0.023	0.056	0.043	0.040	0.038	0.036
Weather	0.091	0.008	0.025	0.017	0.014	0.013	0.011
Segmented	0.065	0.003	0.008	0.008	0.008	0.008	0.008
Children	0.118	0.008	0.022	0.020	0.018	0.017	0.016
Video sequences							
Table Tennis	0.356	0.021	0.078	0.047	0.040	0.036	0.033
Football	0.496	0.025	0.082	0.062	0.056	0.052	0.050
Tempete	0.426	0.026	0.098	0.083	0.076	0.071	0.067

To analyze the proposed mode prediction method, we evaluated the additional computational load of the MSDCS in terms of execution time with a different speed-up threshold, *p*. Table 5-3 summarizes the execution time of the PDS, MVFAST and additional computation of the MSDC with p ranging from 1 to 4. Note that exhaustive comparison between the two modes for every MB is performed with *p* being set to zero. Our experimental results show that, with $p = 1$, about 21% of the introduced computational load can be avoided on average and without any degradation of rate-distortion performance. More computation can be saved with increasing *p*. According to our experiments, on average, by setting $p = 4$, a further 21% of the introduced computation can be avoided when comparing to the situation of $p = 1$, and only about 0.3% of bit rate was increased, with same PSNR coding quality. The comparison is depicted in Figure 7 to Figure 11.

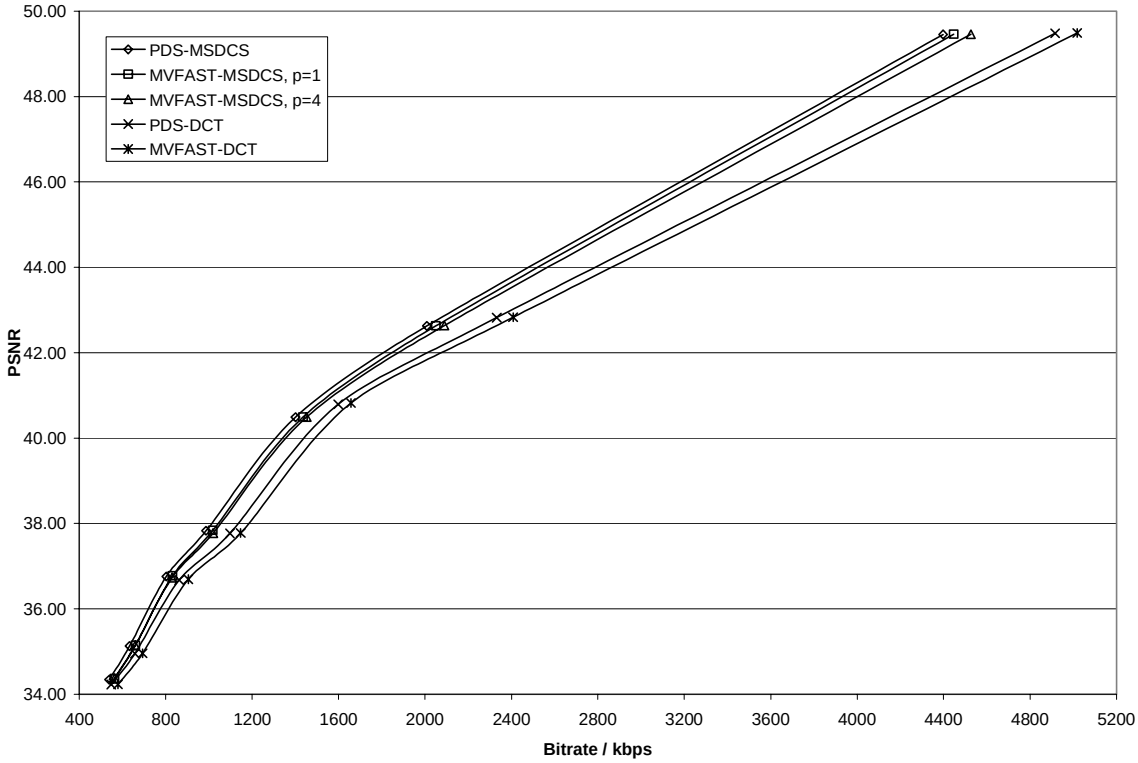


Figure 5-7. Coding performance for the video object "Goldfish" with different quantization parameter, Qp ranged from 1 to 7.

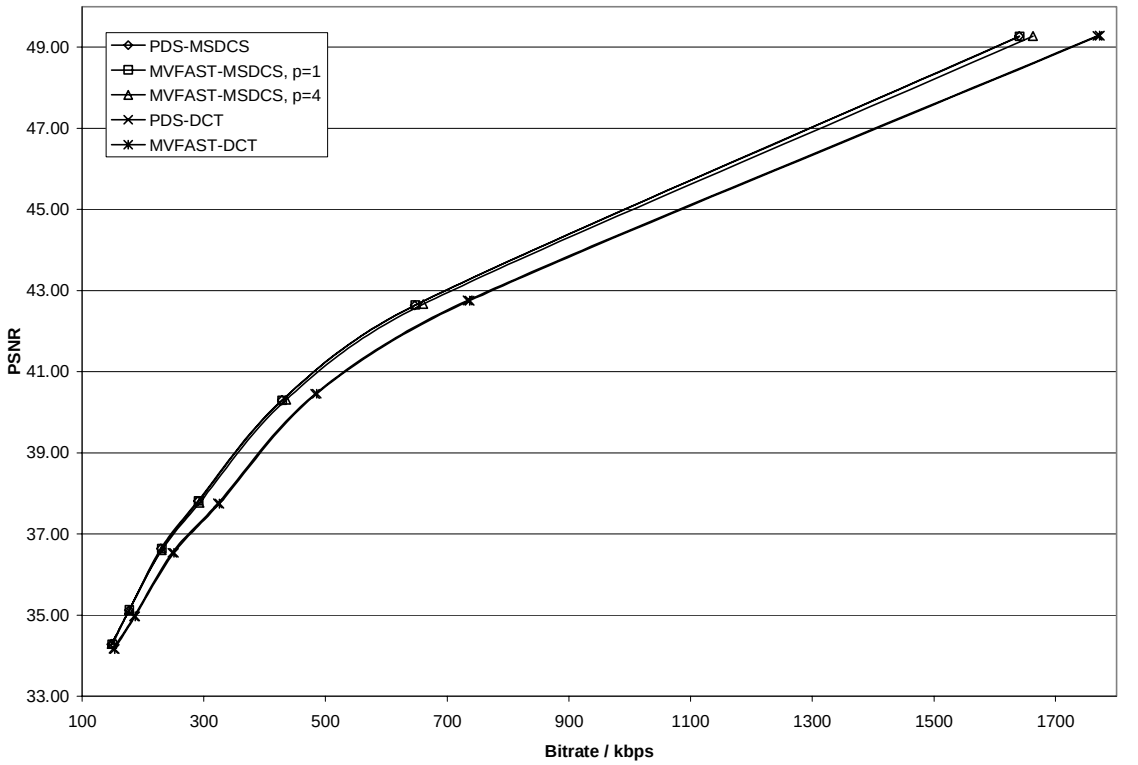


Figure 5-8. Coding performance for the video object "Weather" with different quantization parameter, Qp ranged from 1 to 7.

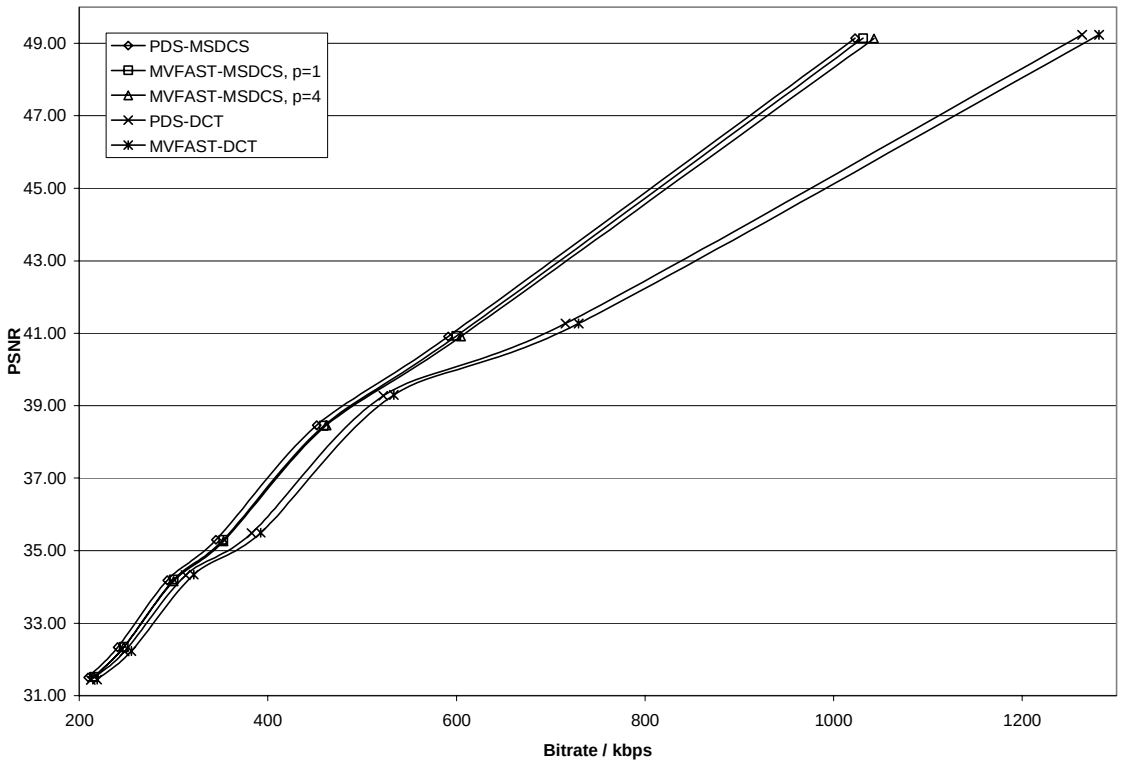


Figure 5-9. Coding performance for the video object "Segmented Stefan" with different quantization parameter, Qp ranged from 1 to 7.

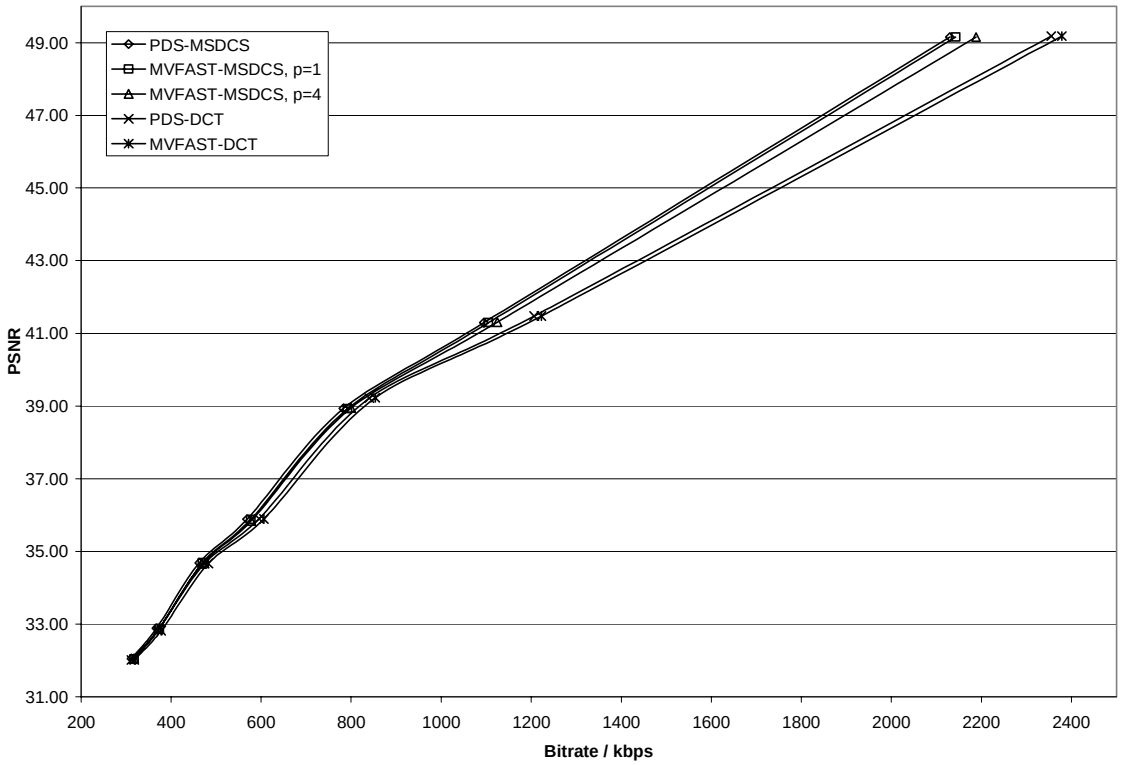


Figure 5-10. Coding performance for the video object "Children" with different quantization parameter, Qp ranged from 1 to 7.

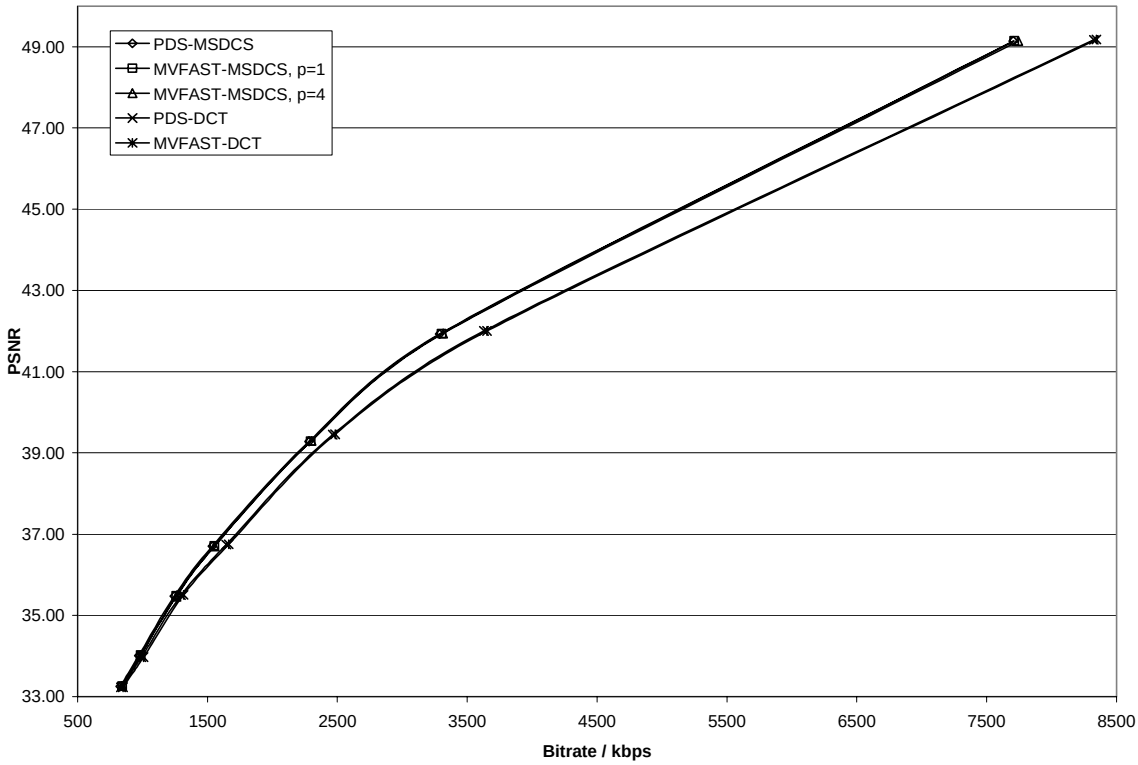


Figure 5-11. Coding performance for the sequence "Table Tennis" with different quantization parameter, Qp ranged from 1 to 7.

The coding efficiency was evaluated for the luminance component only. We compared the rate-distortion performance of the DCT-based coding to that of the MSDCS by varying the quantization parameter, Qp. Figure 5-7 to 5-11 depict the rate-distortion performances of all tested video objects and sequences. The results show that the MSDCS can outperform the traditional DCT-based coding especially in the high video quality region. This advantage vanishes gradually with increasing value of Qp. The reason is that the bitstream of MSDC consists of three components, bits of CAE, VLC and quantized DCT coefficients. Only the last component is a function of Qp. The bits required for coding CAE and VLC remain fairly constant; hence for low bitrate situation they dominate the bit rate requirement. However, experimental results as shown in Figure 5-7 to 5-10 still confirm that for the tested video objects, the MSDCS can achieve better rate-distortion performance comparing to the traditional method for Qp ranging from 1 to 6. On average, with the same PSNR, the MSDCS saves about 5% to 12% of the bitrate, depending upon the contents of the tested video objects. The

improvement given by the MSDCS in the tested sequences is not as significant as the situation in the tested video objects. The corresponding bitrate saving ranges from 1% to 6% on average.

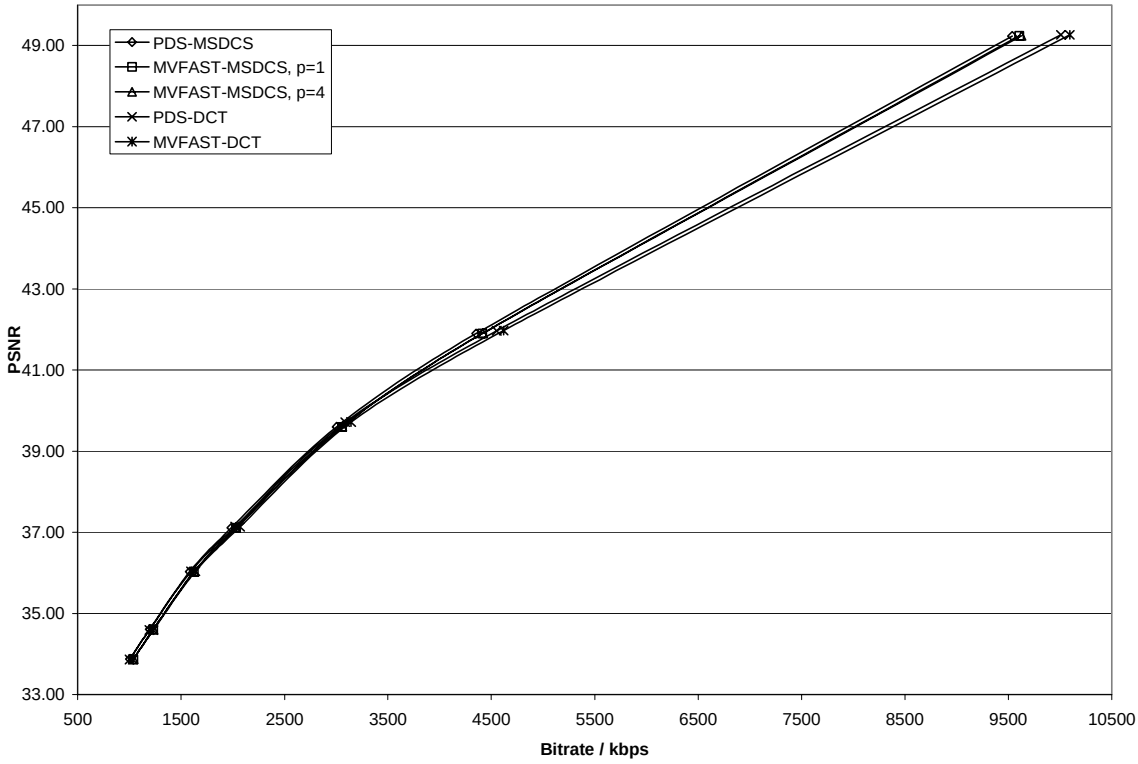


Figure 5-12. Coding performance for the sequence "Football" with different quantization parameter, Qp ranged from 1 to 7.

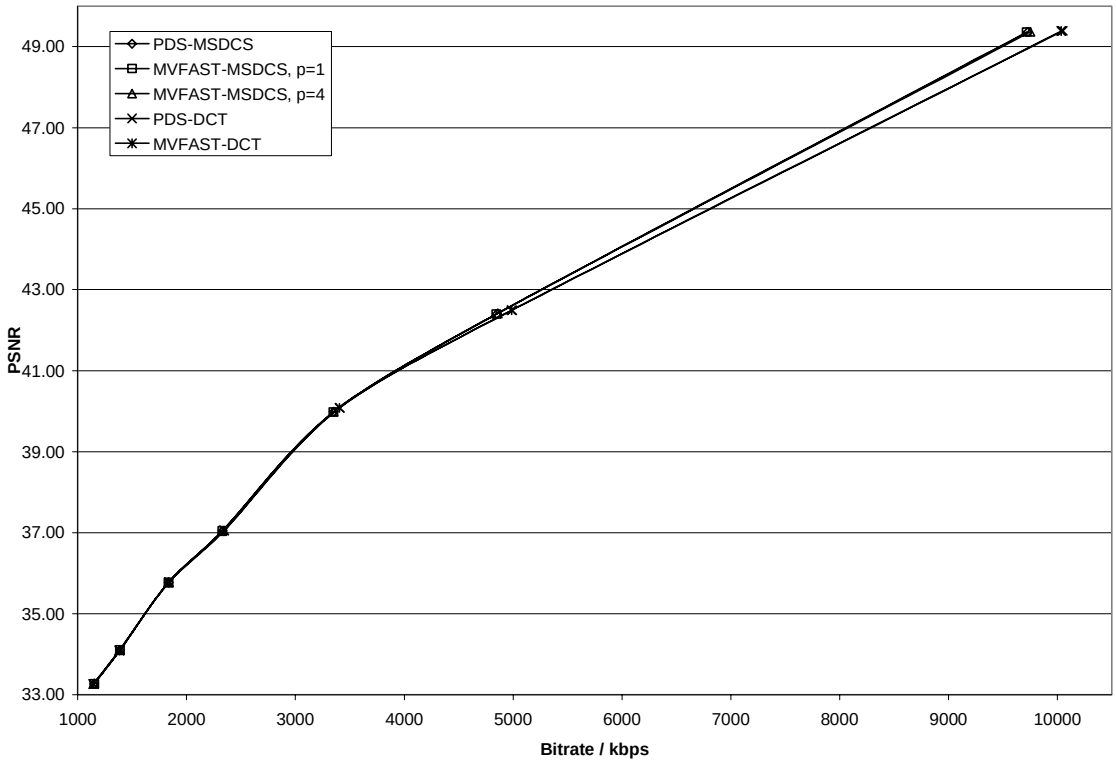


Figure 5-13. Coding performance for the sequence "Tempete" with different quantization parameter, Qp ranged from 1 to 7.

Figure 5-11 to 5-13 show that the MSDCS can still obtain better rate-distortion performance in high video quality situation, when Qp ranges from 1 to 3. From Figure 5-11, sequence "Table Tennis" gives the best performance among three tested sequences. High motion activity and complex background with camera motion in the "Table Tennis" are the main reasons of this result. These factors are the causes of clustered prediction error and thus the DCT-based coding for these error MBs can be improved by the MSDC. From our simulation results, it is verified that the MSDC successfully improves the coding efficiency of MBs with clustered prediction errors. Hence, the proposed MSDCS is very suitable for coding arbitrary shaped objects or sequences with complex motion activity in high video quality application.

Moreover, it is worth to point out that the MSDCS accompanied with the MVFAST provides similar, even much better rate-distortion performance when comparing to the results of PDS with traditional DCT-based coding in all of our experiments. In addition, the required computational time is reduced significantly. We

have found that about 80% of computational time and 7% of bitrate saving are achievable for all tested video objects on average.

5.5 *Conclusions*

In this Chapter, we have proposed a new algorithm, the Mixed Spatial-DCT-Based Coding Scheme, for coding motion compensated prediction errors. The algorithm makes use of the phenomenon that pixel matching errors in some MBs tend to appear together in a cluster form. The reasons of this phenomenon include inefficient block based motion estimation, inaccurate segmented video objects, results of repetitive padding in MPEG-4, and high and complex motion activities. The proposed algorithm exploits the redundancies remained and caused by these phenomena in an error MB. Our experimental results show that the algorithm successfully improves the coding efficiency of the traditional DCT-based coding for MBs with clustered prediction errors. On average, with reasonable additional computation, the proposed algorithm saves up to 12% of the bit rates comparing to the results of the DCT-based coding when the same PSNR is used for all testing video objects and sequences with high video quality. The average required time for additional computation is about 1.9 to 2.4 times of the execution time for MVFAST. When comparing to the fast full search algorithm, PDS, only 13% to 19% of the execution time of the PDS is required. It is lesser than the required execution time if the optimal coding performance is used in a traditional coding system. The major advantages of the MSDCS include conceptual simplicity, and well defined CAE and VLC technique in the modern video codec. This also allows an easy way to improve an existing video codec by embedding the proposed MSDCS into it. Generally speaking, the proposed algorithm is extremely suitable for coding arbitrary shaped video objects or sequences with complex motion activity for high video quality coding applications.

Charter 6. Extended analysis of motion-compensated frame difference for block-based motion prediction error

6.1 *Introduction*

Transform coding method with motion-compensated prediction, abbreviated as hybrid interframe coding, is one of the most essential methods for modern video coding standards. In this method, motion-compensated prediction errors in a block are transform coded by the DCT. To limit the generated amount of bits, a quantizer carries out quantization to discard some part of transform coefficients.

Most of the work for the design and optimization of the hybrid video codecs is carried out experimentally. A proper theoretical treatment of motion-compensated video coding is still valuable for the design of state-of-the-art video codecs, even though it requires many assumptions and simplifications for the analysis of a complicated system processing real-world signals. Furthermore, even an approximate theory can provide useful insights in the underlying mechanisms of the video codecs. In 1987, the first comprehensive rate-distortion analysis of motion-compensated prediction (MCP) was presented [20]. This theoretical framework leads motion-compensated video coding away from heuristics and toward an engineering science. Afterward, a lot of research activities investigate this subject in depth and develop many different techniques for efficiency improvement [21, 23-27, 115, 116, 119, 125, 126]. These techniques include motion-compensation with fractional-pixel accuracy [23], overlapped block motion compensation [24-27], loop filtering technique [126], etc.

To design an optimal coding algorithm, a signal-source model that is sufficiently accurate to reflect the practical signal characteristics is required. The first-order Markov process has been proven to be a successful model for still image analysis. This model is

accurate for the correlation relationship for smooth image, and it even works fairly well for the first few steps in more active images. By using the model, the correlation coefficient ρ for natural image is often suggested as 0.95. Hence, we consider that the DCT is a very close approximation of the optimum KLT and widely employed for video coding. However, for motion-compensated errors signal, the situation is very different. It has been observed that the MCP errors at block boundaries tend to be larger than those at block centers [25]. It means that they are space-dependent and the assumption of wide-sense stationary (WSS) is not valid. As a result, it is inaccurate to employ the simple Markov model for the MCP errors.

In [115], Chen and Pang proposed a compound covariance model (CP model) theoretically and demonstrated that the DCT performs nearly optimal in intraframe coding. Nevertheless, this investigation still assumed that the prediction errors are WSS across a block. For the objective of reduction and equalization of the MCP errors across a block, overlapped block motion compensation (OBMC) has therefore developed. In [116], it confirms that means and standard deviations of the errors may change significantly from block to block. Hence Niehsen and Brünig, proposed another different compound covariance model (NB model) empirically, which takes the OBMC into account. According to their experimental results, they claimed that their model closely fit the characteristic of practical signals. The major disadvantage of this model is it lack theoretical basis, and thus using of this model for other analytical purposes is limited. To resolve the problem, we assume that a net deformation of pixels in a block along a certain direction is a more general situation. We improve the CP model to propose a covariance model analytically by making use of this assumption. Our proposed model reflects the characteristics of practical prediction errors fairly well and comparable to the empirical model proposed in [116]. Moreover, our model explains the deviation of the compound model [115].

The outline of this study is as follow. Section 6.2 presents the derivation of a mathematical model for autocorrelation of block-based motion prediction error. Our derivation is originated from a similar direction as that of [125], such that it is easy to show that the compound model [115] can be obtained from our model analytically. This work is also given in Section 6.2. Section 6.3 shows our simulation results, and comparing to other models. The experimental results given from [115, 116, 125] verify the accuracy of our model. The last Section 6.4 summarizes and concludes this study.

6.2 *Modeling of the autocorrelation of block-based motion prediction error*

The simplicity and analytical tractability make the AR(1) as a popular model in still image and image sequence processing. Our model is also derived based on a first-order Markov statistics (or first order Autoregressive AR(1)) model with image correlation coefficient equal to ρ.

For a block of pixels $f_t(i,j)$ in a frame at time t, the block-based motion compensation uses a matched block $f_{t-1}(i+u,j+v)$ in a reference frame at time t-1 for prediction. Hence, the motion prediction error is given by,

$$e(i,j) = f_t(i,j) - f_{t-1}(i+u, j+v) \tag{6-1}$$

where (u,v) represent the motion vector of the block.

The autocorrelation function, $C_e(I, J)$ of the prediction error is then given by,

$$\begin{aligned} C_e(I,J) &= E\left[\left\{f_t(i,j) - f_{t-1}(i+u, j+v)\right\}\times \right. \\ &\qquad \left. \left\{f_t(i+I, j+J) - f_{t-1}(i+u+I, j+v+J)\right\}\right] \\ &= 2C_f(I,J) - 2C_{f,t}(I-u, J-v) \end{aligned} \tag{6-2}$$

where $C_f(\cdot,\cdot)$ is the autocorrelation function of with correlation coefficient ρ.

$C_{f,t}(\cdot,\cdot)$ is the cross-correlation function between a frame at time t and the reference frame at time t-1.

We assume that the matched block $f_{t-1}(i+u,j+v)$ can be approximated to the current block $f_t(i, j)$ with reasonable deformation. That is

$$f_t(i+m_x, j+n_y) \approx f_{t-1}(i+u, j+v) \quad \textbf{(6-3)}$$

The deformation vector (m_x,n_y) represent the deformation of each pixels in the current block. Note that, in this modeling, the magnitudes of m_x and n_y are not related to that of the u and v directly. They only depend on the matching or correlation between the matched block, $f_{t-1}(i+u, j+v)$ and the current block, $f_t(i, j)$. By substituting (6-3) into (6-2), we can have,

$$C_{f,t}(I-u, J-v) \approx E\left[f_t(i+m_x, j+n_y) f_t(i+I, j+J)\right] = C_{f,t}(I-m_x, J-n_y) \quad \textbf{(6-4)}$$

The reasons for the block deformation include failure of block-based motion model for moving parts, light variation, inaccurate of motion compensation (i.e. inaccuracy due to digitized image), and noise. We regard the deformation vector is a pair of independent random variables, and the expectation value of the autocorrelation function, $C_e(I, J)$ is represented as,

$$E[C_e(I,J)] = 2C_f(I,J) - 2E\left[C_{f,t}(I-m_x, J-n_y)\right] \quad \textbf{(6-5)}$$

By applying separable 2-dimensional AR(1) model, we get

$$C_f(I,J) = \sigma_f{}^2 \rho^{|I|} \rho^{|J|} \quad \textbf{(6-6)}$$

where $\sigma_f{}^2$ is variance of the pixels value in AR(1) model.

and

$$E\left[C_{f,t}(I-m_x, J-n_y)\right] = \sigma_f{}^2 E\left[\rho^{|I-m_x|} \rho^{|J-n_y|}\right] \quad \textbf{(6-7)}$$

For the sake of simplicity, a separable autocorrelation model is our objective. Hence, (6-7) along x-axis is expressed as

$$E\left[C_{f,t}(I-m_x, J-n_y)\right] = \sigma_f^2 E[\rho^{|I-m_x|}] E[\rho^{|J-n_y|}]; \qquad I \geq 0, J = 0 \quad \textbf{(6-8)}$$

The $E\left[\rho^{|I-m_x|}\right]$ can be computed as

$$E\left[\rho^{|I-m_x|}\right] = \int p(m_x)\rho^{|I-m_x|}dm_x \qquad \textbf{(6-9)}$$

where $p(m_x)$ is the probability density function (pdf) of m_x.

At this stage, we make an assumption that pixels in a deformed block tend to deform along a definite direction rather than deformed randomly. In other words, a mean vector of the deformation vectors (m_x,n_y) is not regarded to be zero according to our assumption. This assumption is based on the translational nature of part of an object, partial rotation of a moving part, zooming and inaccurate of motion compensation. Figure 6-1 illustrates the idea of our assumption using a one-dimensional example.

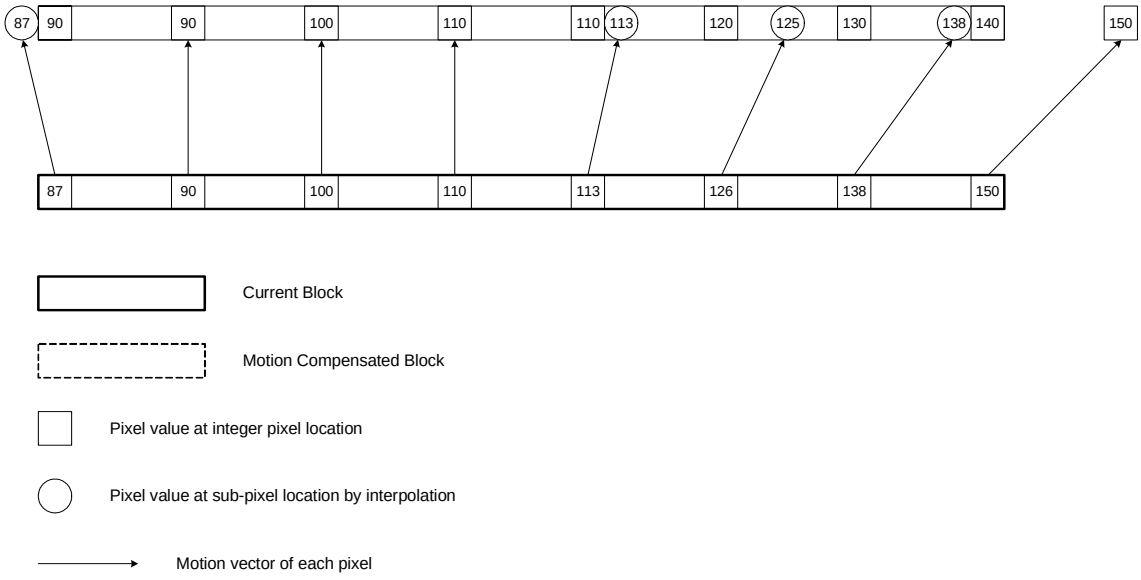


Figure 6-1. Illustration of an assumption that pixels in a deformed block tend to deform along a definite direction

It demonstrates that part of a curent block is not predicted accurately enough, because not all pixels in the block translated in the same direction and their moving distances are not identical. However, they still present some motion tendency. Hence, we refine (6-9) with the above consideration. Then (6-9) becomes

$$E\left[\rho^{|I-m_x|}\right] = \int p(m_x,\mu_x)\rho^{|I-m_x|}dm_x \qquad \textbf{(6-10)}$$

where μ_x is the x-component of the mean deformation vector.

In fact, (6-7) represents the variance-normalized cross-correlation function between the matched block and the current frame in terms of the image correlation coefficient, ρ. It is a matching or correlation measure of the matched block in the current frame conceptually. The assumed block deformation makes the cross-correlation function, $E\left[C_{f,t}\left(I-m_x, J-n_y\right)\right]$ and the error autocorrelation function depend on the direction of the mean deformation vector. However, an error autocorrelation function must be an even function with respect to I. To remedy this directional dependence, we only consider the absolute value of μ_x. Then we further assume that μ_x is randomly distributed. It gives

$$E\left[\rho^{|I-m_x|}\right]=\iint p\left(m_x\middle\|\mu_x\right|)p(\mu_x)\rho^{\|I|-m_x|}dm_x d\mu_x \qquad \textbf{(6-11)}$$

where $p(\mu_x)$ is probability density function of μ_x.

Without loss of generality, we use Gaussian distributions to represent the conditional distribution function, $p\left(m_x\middle\|\mu_x\right|)$ and the pdf, $p(\mu_x)$. The $E\left[\rho^{|I-m_x|}\right]$ is then given by

$$E\left[\rho^{|I-m_x|}\right]=\frac{2}{\sigma_\mu\sqrt{2\pi}}\frac{1}{\sigma_{mv}\sqrt{2\pi}}\int_0^\infty e^{\frac{-\mu_x^2}{2\sigma_\mu^2}}\int_{-\infty}^\infty e^{\frac{-(m_x-|\mu_x|)^2}{2\sigma_{mv}^2}}\rho^{\|I|-m_x|}dm_x d\mu_x \qquad \textbf{(6-12)}$$

where σ_μ is the standard deviation of the mean deformation vector.

σ_{mv} is the standard deviation of the deformation vectors in a single block.

Assuming that the block deformation in the x- and y- dimension can be modeled with same standard deviations σ_μ and σ_{mv}, the error autocorrelation function along x-direction in (6-5) can be expressed as

$$E\left[C_e(I,J)\right]=2\sigma_f{}^2\left\{\rho^{|I|}\rho^{|J|}-E[\rho^{|I-m_x|}]E[\rho^{|J-n_y|}]\right\};\quad I\geq 0, J=0 \qquad \textbf{(6-13)}$$

and the variance-normalized autocorrelation function of the block prediction error is given by

$$\frac{E[C_e(I,0)]}{E[C_e(0,0)]} = \frac{\rho^{|I|}\rho^{|0|} - E[\rho^{|I-m_x|}]E[\rho^{|-n_y|}]}{1 - E[\rho^{|m_x|}]E[\rho^{|n_y|}]} \;\; ; \quad I \geq 0,\ J = 0 \tag{6-14}$$

The result of (6-14) can be obtained by numerical calculation. However, this form is not convenient for analytical purpose. We have described the derivation of its approximated form below.

We use an expected value of the mean deformation vector in (6-11), instead of using the pdf, $p(\sigma_x)$ to approximate the autocorrelation model. The expected value of the mean deformation vector is

$$\overline{\mu}_x(\sigma_\mu) = \frac{2}{\sigma_\mu\sqrt{2\pi}}\int_0^\infty \mu_x e^{\frac{-\mu_x^{\,2}}{2\sigma_\mu^{\,2}}} d\mu_x = \sigma_\mu\sqrt{\frac{2}{\pi}} \tag{6-15}$$

We rewrite (6-12) by using this expected value.

$$\begin{aligned} E\left[\rho^{|I-m_x|}\right] &\approx \frac{1}{\sigma_{mv}\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{\frac{-(m_x-|\overline{\mu}_x|)^2}{2\sigma_{mv}^{\,2}}+||I|-m_x|\ln\rho} dm_x \\ &= \frac{1}{\sigma_{mv}\sqrt{2\pi}}\left[\int_{-\infty}^{|I|} e^{\frac{-(m_x-|\overline{\mu}_x|)^2}{2\sigma_{mv}^{\,2}}+(m_x-|I|)\ln\rho} dm_x \right. \\ &\qquad \left. + \int_{|I|}^{\infty} e^{\frac{-(m_x-|\overline{\mu}_x|)^2}{2\sigma_{mv}^{\,2}}+(|I|-m_x)\ln\rho} dm_x \right] \end{aligned} \tag{6-16}$$

These two integration is expressed involving error function, *erf*(*z*). Finally, we have

$$\begin{aligned} E\left[\rho^{|I-m_x|}\right] &\approx \frac{1}{2}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left\{\rho^{(|I|-|\overline{\mu}_x|)}\left[1+erf\left(\frac{|I|-|\overline{\mu}_x|+\sigma_{mv}^{\,2}\ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right]\right. \\ &\qquad \left. + \rho^{-(|I|-|\overline{\mu}_x|)}\left[1-erf\left(\frac{|I|-|\overline{\mu}_x|-\sigma_{mv}^{\,2}\ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right]\right\} \\ &= \rho^{|I|}\frac{1}{2}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left\{\rho^{|\overline{\mu}_x|}\left[1+erf\left(\frac{|I|-|\overline{\mu}_x|+\sigma_{mv}^{\,2}\ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right]\right. \\ &\qquad \left. + \rho^{-(2|I|-|\overline{\mu}_x|)}\left[1-erf\left(\frac{|I|-|\overline{\mu}_x|-\sigma_{mv}^{\,2}\ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right]\right\} \\ &= \rho^{|I|}\widetilde{R}(I,\overline{\mu}_x,\sigma_{mv},\rho) \end{aligned} \tag{6-17}$$

Now, using the 6-17), we can approximate the variance-normalized autocorrelation function of the block prediction error which given by (6-14) as

$$\frac{E[C_e(I,0)]}{E[C_e(0,0)]} \approx \tilde{C}_{e,\bar{\mu}_x,\sigma_{mv}}(I,0) = \rho^{|I|}\rho^{|0|}\frac{1-\tilde{R}(I,\bar{\mu}_x,\sigma_{mv},\rho)\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)}{1-\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)}$$
$$= \rho^{|I|}\rho^{|J|}\Re(I,J,\bar{\mu}_x,\sigma_{mv},\rho)\Big|_{J=0} \quad \textbf{(6-18)}$$

Finally, we express a separable 2-D variance-normalized autocorrelation function as,

$$\frac{E[C_e(I,0)]E[C_e(0,J)]}{E[C_e(0,0)]^2} = \rho_x^{|I|}\rho_y^{|J|}\Re(I,0,\bar{\mu}_x,\sigma_{mvx},\rho_x)\Re(0,J,\bar{\mu}_y,\sigma_{mvy},\rho_y) \quad \textbf{(6-19)}$$

A detailed derivation of (6-19) is shown in Appendix B-B.1.

In fact, the function $\Re(I,J,\bar{\mu}_x,\sigma_{mv},\rho)\Big|_{J=0}$ is equivalent to the role of $A(a)$ in [125] in 1-D case, which is shown by (2-39).

Moreover, we can show that the compound covariance model, (2-39), proposed in [125] can be obtained from (6-18). The model of (2-39) is a one-dimensional case that derived based on a translational motion model and a composite motion-estimation-error probability density function (pdf) consisting of a uniform pdf for the granular estimation errors and an impulse pdf for the background regions with zero estimation errors.

To derive the (2-39), we assume that a video sequence with very low motion activity is under simulation. For this reason, block-based motion model can properly compensate the motion of each block between successive frames, and thus $\bar{\mu}_x$ and σ_{mv} are set equal to zero and 0.5 respectively. Using the properties of *erf*(z),

$$erf(z) = \begin{cases} \frac{2}{\sqrt{\pi}}e^{-z^2}z\left[1+\frac{2z^2}{1\cdot 3}+\frac{(2z^2)^2}{1\cdot 3\cdot 5}+\cdots\right] & if\ \ z<<1 \\ 1 & if\ \ z>>1 \end{cases}$$

, an approximation of (6-17) can be obtained. For |*I*| equal to zero, we express the $erf(\cdot)$ in (6-17) as the first term of its expanded series, and for |*I*| greater or equal to one, the $erf(\cdot)$ is expressed as unity. Hence, (6-17) is approximated as

$$E\left[\rho^{|I-m_x|}\right] \approx \rho^{|I|}\frac{1}{2}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left\{\left[1+\frac{2}{\sqrt{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\delta(|I|)+(1-\delta(|I|))\right]\right.$$
$$\left. + \left[1-\frac{2}{\sqrt{\pi}}e^{-\left(\frac{-\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\cdot\frac{-\sigma_{mv}\ln\rho}{\sqrt{2}}-(1-\delta(|I|))\right]\right\}$$
$$= \rho^{|I|}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left[1+\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\sigma_{mv}\ln\rho\cdot\delta(|I|)\right] \quad \textbf{(6-20)}$$

Let $\beta = \sqrt{\frac{2}{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\sigma_{mv}\ln\rho$, and substituted into (6-20). As a result,

$$\tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(I,0) \approx \rho^{|I|}\frac{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\delta(|I|)\right]\cdot[1+\beta]}{1-e^{(\sigma_{mv}\ln\rho)^2}[1+\beta]^2}$$
$$= \rho^{|I|}\left[A+(1-A)\delta(|I|)\right] \quad \textbf{(6-21)}$$

Where

$$A = \frac{1-e^{(\sigma_{mv}\ln\rho)^2}[1+\beta]}{1-e^{(\sigma_{mv}\ln\rho)^2}[1+\beta]^2} \quad \textbf{(6-22)}$$

The numerical value of A is equal to 0.497 with ρ = 0.95 and σ_{mv} = 0.5. (2-39) is arrived. The detailed mathematical deduction of (2-39) is shown in Appendix B-B.2.

In Matrix form,

$$COV = \begin{bmatrix} \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(0,0) & \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(1,0) & \cdots & \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(N-1,0) \\ \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(1,0) & \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(0,0) & \cdots & \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(N-2,0) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(N-1,0) & \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(N-2,0) & \cdots & \tilde{\tilde{C}}_{e,\bar{\mu}_x,\sigma_{mv}}(0,0) \end{bmatrix} \quad \textbf{(6-23)}$$

It is equivalent to the corresponding Toeplitz form of eqn. (2-30), which is shown in eqn. (2-34).

6.3 *Simulation Results*

Let us consider an analytical treatment of the motion-compensated frame differences (MCFD). A covariance model that can represent the empirical covariance result accurate enough is required. Hence, it is necessary to verify a covariance model by comparing its fitness to empirical results. We made use of the experimental results form [115, 116 and 125] to compare the fitness of our model with that of the CP model and the WNMB model. The empirical result of "Trevor" sequence extracted from [125] and the result of sequence "Miss America" in [115] are used to evaluate the accuracy of our model and compared to the CP model in 2-D situation. Besides, we have compared the accuracy of our model with that of the WNMB model in 1-D case by using the statistical result of a number of standard sequences in [116].

Figure 6-2 depicts the 3-D plot of an autocorrelation function of MCFD signals generated from Trevor images [125]. The spatial autocorrelation function of the Trevor image has a high value with ρ = 0.99 [125].

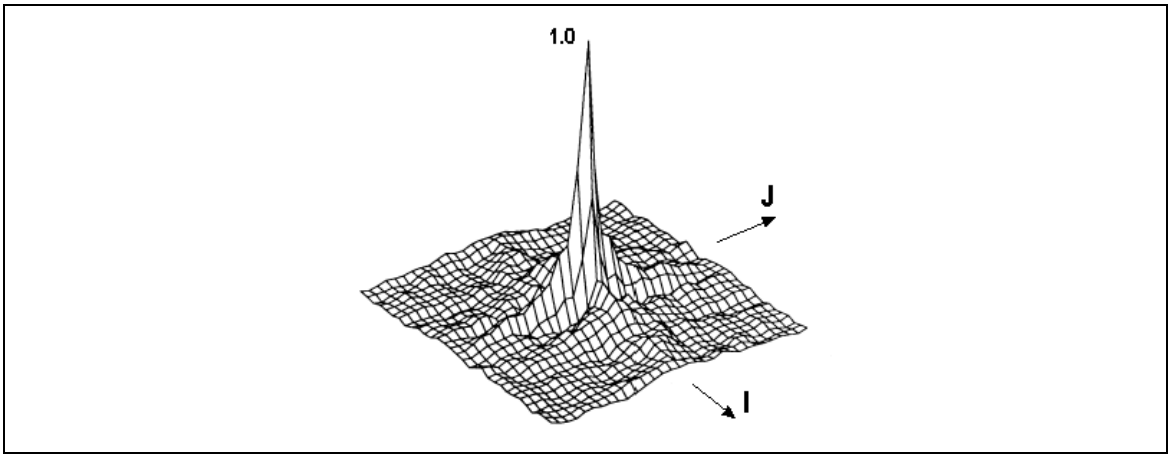
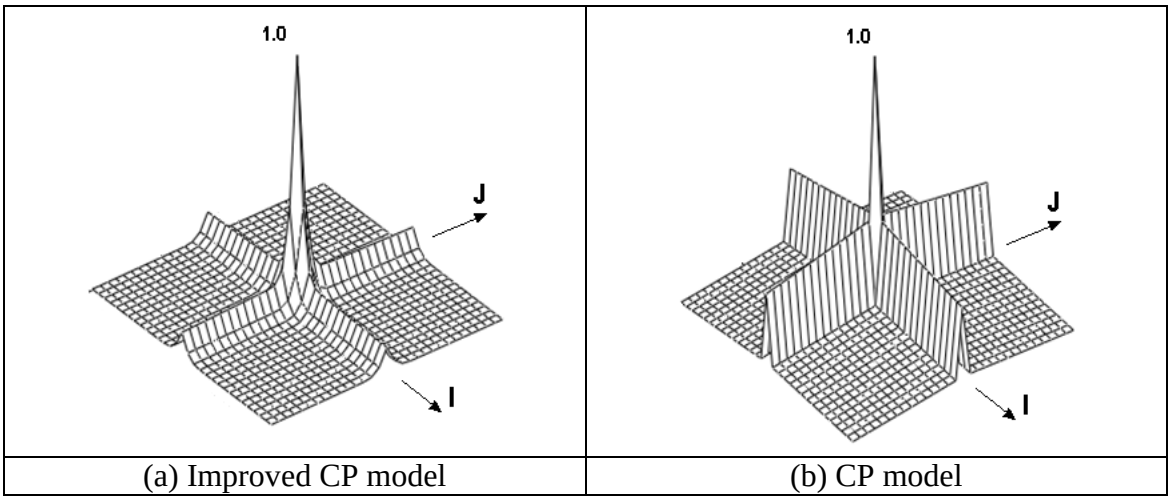


Figure 6-2. Autocorrelation function of MCFD signals generated from Trevor sequence [122].

The result of our improved model is plotted in Figure 6-3(a) by substituting ρ_x = ρ_y = 0.99, σ_x = σ_y = 1.15, and μ_x = μ_y = 0.75 into (6-19). We have plotted the result of CP model with ρ_x = ρ_y = 0.99 in Figure 6-3(b) for comparison. According to the prediction from CP model, the autocorrelation function is decreasing slowly for $I, J \geq 1$. It deviates significantly from the empirical result. However, the improved model can

represent the autocorrelation function more accurately when comparing to the original CP model. It correctly represents the rapidly decreasing autocorrelation for $I, J \geq 1$.



(a) Improved CP model (b) CP model

Figure 6-3. Representation of the autocorrelation function of Trevor's MCFD signals by separable models (a) our improved model, (b) original CP model.

Let us use another experimental result, the autocorrelation function of MCFD of the sequence "Miss America" in [115], to evaluate the accuracy of our model and that of the original CP model. The pixel correlation coefficients are 0.88 and 0.80 in the vertical direction (indicated by arrow J) and horizontal direction (indicated by arrow I) [115]. The corresponding autocorrelation function of the MFCD is shown in Figure 6-4.

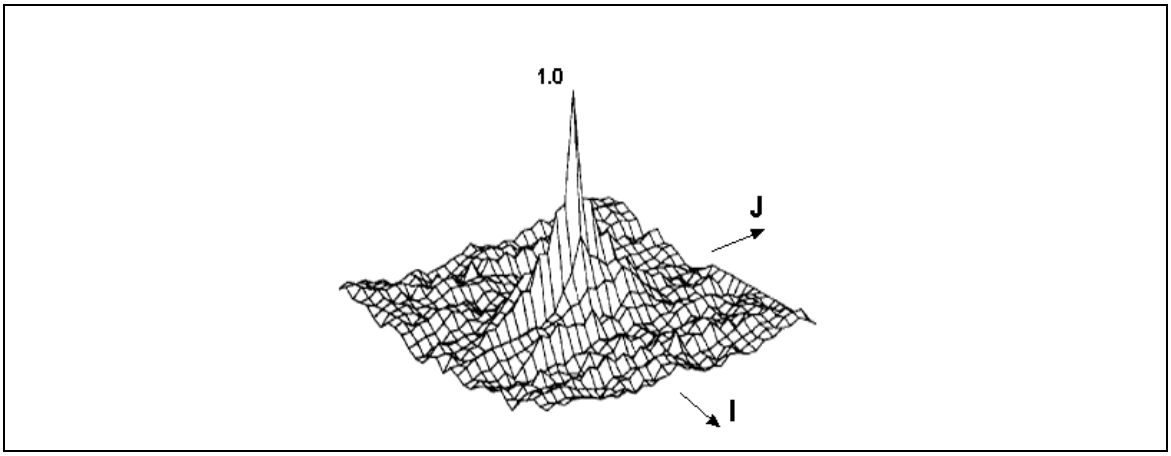
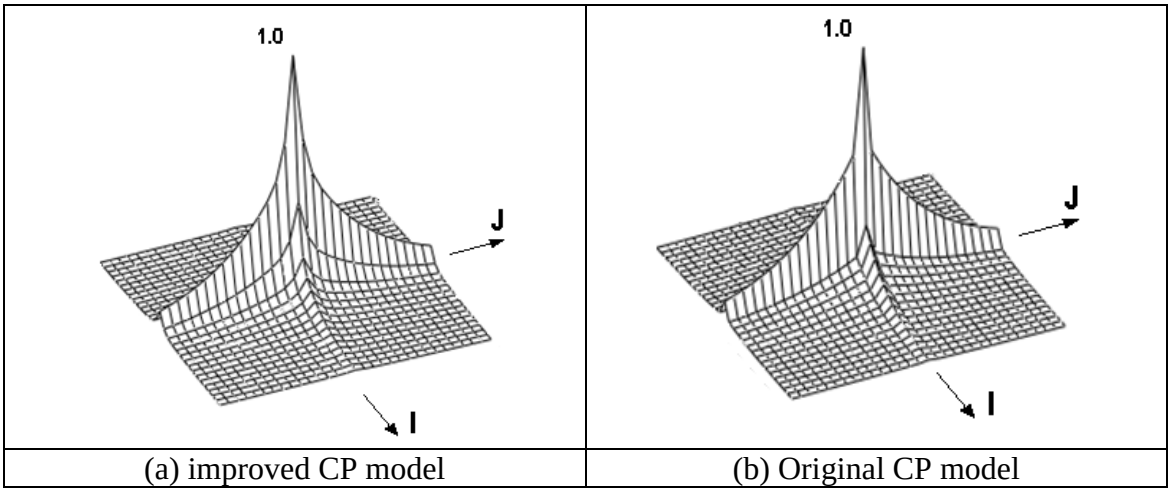


Figure 6-4. Autocorrelation function of MCFD of the sequence Miss America.

Figure 6-5(a) illustrates the representation of the autocorrelation function of Miss America's MCFD signals by our improved model with the following parameters ρ_x = 0.80, ρ_y = 0.88, $\sigma_x = \sigma_y$ = 1.15, and $\mu_x = \mu_y$ = 0 in (6-19). Meanwhile, Figure 6-5(b)

gives the result of the original CP model with $\rho_x = 0.80$ and $\rho_y = 0.88$. Figure 6-5 shows that both models can compare favorably with the experimental results in Figure 6-4. Because, the original CP model does not concern practical motion properties of a video sequence, it assumes that the block based motion estimation can successfully compensate motion of each pixel in a block. This assumption is only valid in sequence with very slow motion activities, such as the Miss America. However, we can still find that, in the I-direction, the CP model shows a sudden drop of autocorrelation value at I=1, which is not correct when compared to the empirical result in Figure 6-4. The improved model can represent the trend of the autocorrelation function in both directions correctly.



(a) improved CP model (b) Original CP model

Figure 6-5. Representation of the autocorrelation function of Miss America's MCFD signals by separable models (a) improved CP model, (b) original CP model.

In [116], the authors used four MPEG test sequences to verify and proposed their 1-D NB model, which empirically fitted in the sense of the l_1-norm. Their experimental results are shown in Figure 6-6. Figure 6-6 gives the maximal, minimal and mean of the normalized autocorrelation functions of the MFCD at different pixel distances. We use these results to evaluate the fitness of the original 1-D CP model, the NB model and the improved CP model. The original 1-D CP model, the NB model and the improved CP model are given in (2-39), (2-40) and (6-18) respectively. In this simulation, we set ρ_x =

0.95, σ_x = 1, and μ_x = 1 in (6-18). We chose these values for σ_x and μ_x in order to fit the experimental results close enough. The value of ρ_x = 0.95 is chosen identical to the original CP model.

In Figure 6-6, we can find that the original CP model does not fit accurately the empirical autocorrelation of the MCFD incorrectly. The CP model decreases slowly with pixel distances, $I \geq 1$, which is different from the empirical results significantly. The NB model fits the experimental results closely, because the required parameters of the NB model are chosen to fit the experimental result in the l_1-norm sense. However, this model is purely empirical and without any theoretical foundation. The usage of this model for analytical purposes is seriously limited. On the other hand, the improved CP model successful represents the autocorrelation function of the MCFD as shown in Figure 6-6. Moreover, the improved model is derived from the simple first order Markov model and including the consideration of a net deformation of pixels in a block due to imperfect block-based motion compensation. Hence, the improved CP model is suitable for analytical design and investigation of signal decomposition algorithms in motion compensation. For instance, using of σ_x = 1, and μ_x = 1 indirectly suggests that the motion compensation errors in a block is not distributed uniformly. It is a property of the motion compensation error that induces a widely study of the OBMC algorithms in video coding. Moreover, it also explains the deviation of the CP model from empirical results, which is due to the lack of concern about practical block-based motion compensation in an encoder.

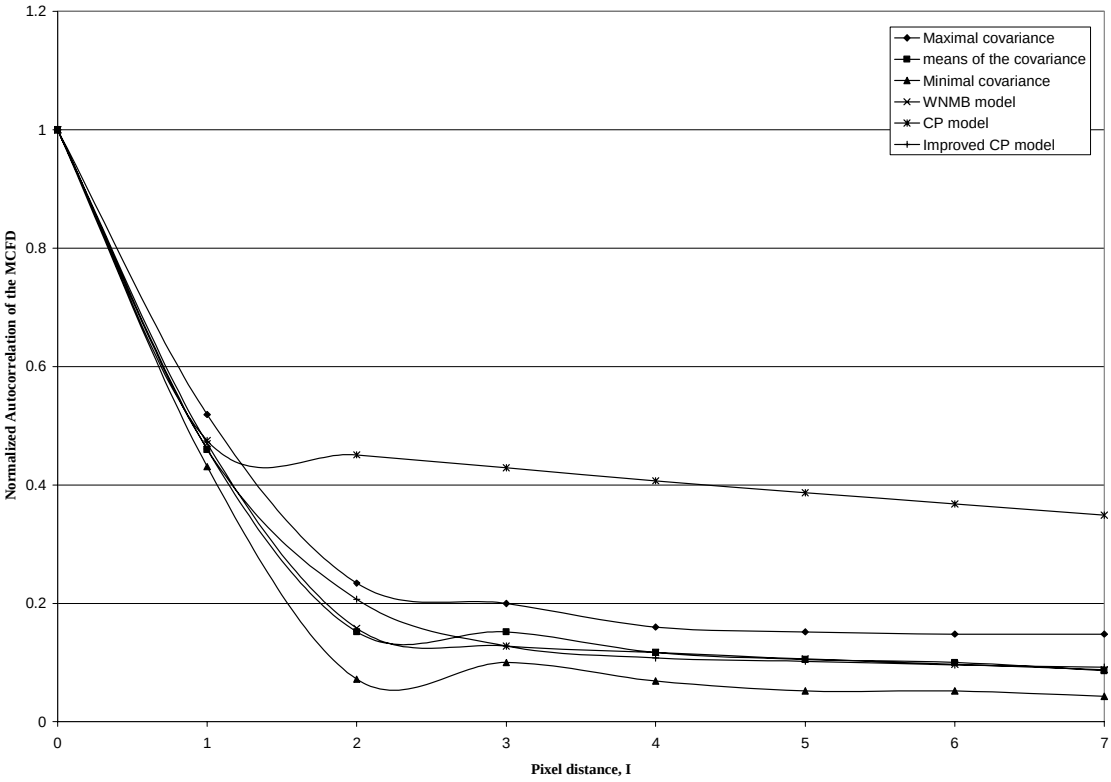


Figure 6-6. Comparison of the predicted autocorrelation function by the tested models to the experimental results.

6.4 *Conclusion*

A proper theoretical treatment of motion-compensated video coding is valuable for the design of most advanced video coding system, even though it requires to have a number of assumptions and simplifications for the analysis of real-world signals.

In this study, we have shown that the first order Markov model can be used to derive an approximate separable autocorrelation model for the block based motion compensation difference signal. In the derivation, we have assumed that a net deformation of pixels is directional in general situation rather than a uniform error distribution in a block. We have improved the original CP model by proposing a covariance model analytically making use of this assumption. Simulation results show that, the improved CP model can describe the characteristics of the MCFD signals

accurately. We have also found that the concern of imperfect block-based motion compensation is one important step to study the motion-compensated coder; otherwise the autocorrelation function of the MCFD signals cannot be expressed correctly. As a result, we can utilize this improved model to provide some useful insights for analytical design and investigation of video signal decomposition algorithms.

Chapter 7. Conclusion

7.1 *Conclusion on this investigation*

In this multimedia era, different applications and services require the communication and interactive functions in the form of text, audio, image and video. However, the transmission and storage requirements of these multimedia data are very critical, especially for the video data. Consequently, these requirements motivate the rapid development of video compression standards, such as the ITU-T H.261, the ITU-T H.263, H.264 the ISO MPEG-1, the ISO MPEG-2 and the ISO MPEG-4. Both types of standards using block-based motion estimation techniques to exploit the temporal redundancies from frame to frame in order to achieve the purpose of high compression ratio. However, the amount of computation for motion estimation may take up to about 90% of the execution time of the whole encoding system on average if the conventional Full Search Algorithm is employed. Hence, there is a huge need for low computational complexity motion estimation techniques which do not significantly degrade the image quality. In addition, the motion estimation and compensation techniques used in MPEG-4 can be seen as an extension of the standard MPEG block matching techniques for image objects with arbitrary shapes. Nevertheless, the situation of motion estimation in the boundary region of an arbitrary shaped object is more complex. Severe optimizations are necessary. Hence we concentrate on developing different efficient motion estimation algorithms for a modern video encoder.

In Chapter 3, we propose a new priority search algorithm (PSA) for motion estimation in MPEG-4. For an arbitrarily shaped object, we perform motion estimation on all boundary MBs first in contrast to the conventional raster-scanning approach. The motivation behind the new search strategy is that opaque MBs which are inside a

moving video object are correlated highly with moving boundary MBs. We estimate motion vectors (MVs) of the opaque MBs by taking the best-matched MV among all its neighboring MVs and the zero MV as the initial centre. For searching, we make use of conventional fast block matching algorithms such as the Diamond Search. Our experimental results show that this strategy can provide good searching results if the motion vectors in the boundary MBs truly represent the moving video object. Hence, a novel fast search algorithm is thus designed for the boundary MBs, which is referred to as the binary alpha-plane assisted search (BAAS). Note that the error surface of a boundary MB is more complex than that of an opaque MB due to the repetitive padding. The assumption that a distortion function increases monotonically as the search location moves away from the global minimum is not valid. However, it is still reasonable for us to assume that it is monotonic in a small neighborhood around the global minimum. We use a binary alpha-plane to examine a number of candidate points, which are distributed uniformly in a search window. We can select a limited number of starting points but this still provides a high chance of catching the global minimum for the boundary MBs. The reason for us to incorporate the binary alpha-plane in BAAS is to reduce the required computational load. The information of the binary alpha-plane can be exploited by simple bitwise operations. These operations require less computation as compared to the operations for Sum of Absolute Difference (SAD). Experimental results show that, when compared to conventional methods, our PSA coupled with the BAAS can reduce the heavy computational burden in motion estimation without significantly increasing the prediction error of the motion-compensated frame. The proposed algorithm can produce a motion-compensated VOP that are tied more closely to the video object. It is significantly better than that of the famous DS and substantially improves the accuracy of block motion estimation for MPEG-4 video objects. On average, the proposed algorithm can speed up the motion estimation about 23 times in terms of the total

number of operations when compared to the FSA. We believe that results of our work will certainly be useful in the future development MPEG-4 codecs.

Many conventional fast algorithms including our PSA proposed in Chapter 3 produce some quality degradation of a predicted image. Alternatively, another kind of fast algorithms that do not introduce any prediction error as compared with the full-search algorithm have been widely studied in the coding field. The partial distortion search (PDS) is a well-known technique of this kind of algorithms. In Chapter 4, we propose an adaptive partial distortion search entitled as the Clustered Pixel Matching Error for adaptive Partial Distortion Search (CPME-PDS) which significantly improves the computation efficiency of the original PDS. In the literature, many researchers assumed that pixels with larger gradient magnitudes have larger matching errors on average and make used of this assumption to improve block-matching algorithms. On the other hand, we have found that on average, pixel matching errors with similar magnitudes tend to appear in clusters for natural video sequences. Our experimental results show that, by using this clustering characteristic, the CPME-PDS gives much better computational efficiency than other algorithms which make use of the pixel gradients, especially for encoding sequences with high motion activities and arbitrarily shaped video objects. In the CPME-PDS, we used the mean of pixel values in the initial candidate MB at the centre of a search window to determine the order of each pixel matching errors that accumulate to a partial SAD. As a result, pixels matching error with larger magnitudes can be accumulated to SAD_p sooner than others and the SAD calculation can be terminated at an early stage. We have evaluated the efficiency of the CPME-PDS in two measures, the total number of operations and the execution time per frame or per VOP in motion estimation with a wide variety of sequences. In terms of the number of operations, the CPME-PDS and the original PDS can have a speed-up of 3.93 and 5.71 on average as compared with FSA, respectively. When motion estimation

time per frame or per VOP is used for evaluation, the performance of CPME-PDS is degraded slightly due to the problem of non-uniform memory access. Hence, we have modified the CPME-PDS into a row-based algorithm in order to remedy the non-uniform memory access problem. For example, a row of 4 consecutive pixels with larger prediction errors is accumulated to the SAD_p sooner than other rows. Experimental results show that our row-based CPME-PDS_4 and the original PDS can speed up the search for about 3.38 times and 2.56 as compared to the FSA on average. Hence, we conclude that the CPME-PDS improves the original PDS significantly.

After the processes of motion estimation, the motion prediction error is then coded by using the discrete cosine transform (DCT) to achieve high compression efficiency. The DCT is widely used in modern video compression standards. A major merit of the DCT is its capability in high energy compaction for still natural images. Hence, DCT based coding is popular for video coding. Nevertheless, the motion prediction error frame is not a natural image but synthetically generated by the process of motion compensation. This process degrades the energy compaction efficiency of the DCT. As a result, using the DCT based coding to perform compression for the motion prediction error is far from optimal.

In Chapter 5, we study the spatial distribution of the prediction errors resulting from either the full-search motion estimation or other fast search algorithms. We have then proposed a new algorithm, the Mixed Spatial-DCT-Based Coding Scheme (MSDCS) to resolve the inefficiency of coding the prediction error in the DCT domain. The algorithm makes use of the phenomenon that pixel matching errors in some MBs tend to appear together in a cluster form. The reasons of this phenomenon include inefficient block based motion estimation, inaccurate segmented video objects, results of repetitive padding in MPEG-4, and high and complex motion activities. The MSDCS exploits the redundancies remained and caused by these phenomena in an error MB.

The proposed algorithm divides a prediction error MB into two components. Each component is characterized by its own spatial correlation. One component is then coded by using the binary bit plane coding and variable length coding techniques (VLC), and the second component is coded by using the traditional DCT-based method. Our experimental results show that the algorithm successfully improves the coding efficiency of the traditional DCT-based coding for MBs with clustered prediction errors. On average, with reasonable additional computation, the proposed algorithm saves up to 12% of the bit rates comparing to the results of the DCT-based coding when the same PSNR is used for all testing video objects and sequences with high video quality. The average required time for additional computation is about 1.9 to 2.4 times of the execution time for MVFAST. When comparing to the fast full search algorithm, the PDS, only 13% to 19% of the execution time of the PDS is required. It is less than the required execution time, such that the optimal coding performance can be obtained in a traditional coding system. The major advantages of the MSDCS include conceptual simplicity, and well defined CAE and VLC techniques for a modern video codec. This also allows an easy way to improve an existing video codec by embedding the proposed MSDCS into it. Generally speaking, the proposed algorithm is extremely suitable for coding arbitrary shaped video objects or sequences with complex motion activity for high video quality coding applications.

In the past decade, various transform coding techniques such as the one using DCT, subband/ wavelet and vector quantization have been developed for video coding. Among these coding techniques, the DCT based coding is still the most popular for various video coding standards. For a natural image, correlation coefficient $\rho = 0.95$ is often suggested, and the discrete cosine transform (DCT) is often employed for its energy compaction capability. Because the DCT is a very close approximation of the theoretically mean-square optimal Karhunen-Loève Transform for an first order

autoregressive, AR(1), process with ρ = 0.95. However, the statistical properties of motion prediction errors (MPE) are different from that of natural images. Some researchers are still questioning that there is no theoretical basis that can clearly predict the suitability of the DCT for encoding the motion prediction errors. Moreover, a proper theoretical treatment of motion prediction error is valuable for the design of most advanced video coding system, even though we require making many assumptions and simplifications for the analysis of the real-world signals.

In 1993, Chen and Pang [115] proposed a compound covariance model (CP model) theoretically and demonstrated that the DCT performs nearly optimal as the situation in the intraframe coding. Nevertheless, this investigation assumed that the prediction errors are wide-sense stationary across a block. It can only describe the characteristics of the MPE across a block in sequences containing low motion activities. In 1999, Niehsen and Brünig [116] proposed another compound covariance model (NB model) empirically and showed that means and standard deviations of the errors may change significantly from block to block. According to their experimental results, their model can closely fit the characteristic of practical signals. The major disadvantage of this model that it lacks a theoretical basis, and thus using of this model for other analytical purposes is limited.

In Chapter 6, we have shown that the first order Markov model can be used to derive an approximated separable autocorrelation model for the block based motion compensation difference signal. In the derivation, we assumed that a net deformation of pixels is directional in general situation rather than a uniform error distribution in a block. We have improved the original CP model to propose a covariance model analytically by making use of this assumption. Simulation results show that, the improved CP model can describe the characteristics of the MCFD signals accurately. We have found that the concern of imperfect block-based motion compensation is one

important step to study the motion-compensated coder; otherwise the autocorrelation function of the MCFD signals cannot be expressed correctly. As a result, we can use this improved model to provide useful insights for an analytical design and investigation of video signal decomposition algorithm.

7.2 *Future research directions*

We can classify the results obtained in this research work into two parts. The first part is a study of fast motion estimation algorithms. The second part relates to the transform coding of motion prediction errors. However, considering to apply these techniques to modern coding system or real applications, further study and extension of our work are necessary.

In Chapter 3, we have proposed a lossy fast motion estimation algorithm. Besides, in Chapter 4, a lossless fast algorithm has also been developed. These algorithms can significantly speed-up the motion estimation for arbitrarily shaped objects in MPEG-4. However, the computational requirement of the future video encoder will become more critical. The newest international video coding standard, H.264/AVC (or MPEG-4 part 10) allows using more than one prior coded picture as reference for motion-compensated prediction. We may construct a block of prediction signals from a weighted average of two motion-compensated prediction values. Moreover, the standard supports to partition a macroblock into smaller block sizes of 16×16, 16×8, 8×8, 8×4 pixels. We expecte that the traditional motion estimation algorithm need to be enhanced to provide acceptable efficient. Other than using the edge feature that we studied in Chapter 3 and clustering characteristic used in Chapter 4, we believe that color feature and object texture are useful properties to develop efficient algorithm for the H.264/AVC. For instance, we may use a matching condition of texture in a block as a

criterion to determine the partition of a macroblock and the requirement of weighted averaging for a block reconstruction.

In Chapter 5, we have shown that a suitable transform algorithm accompanied with a fast lossy motion estimation algorithm can achieve equivalent or better result in both rate-distortion and computational efficiency, when comparing to the optimal performance in a tradition video coding system that uses the full search algorithm. It suggests that we may treat the optimization of fast motion estimation and transform coding of the motion prediction error as a whole system. Subsequently, we can compensate the rate-distortion performance and speed-up factor between a transform algorithm and a fast motion estimation to optimize a video coding system. We consider that the contribution of our work in Chapter 5 is indeed significant, but not complete, because it is not suitable for low bit-rate applications. Hence, it is good to extend our study into this region.

A signal-source model that is accurate enough to describe the practical signal characteristics is an important issue for us to fulfill the above purpose. The auto correlation model for motion prediction error proposed in Chapter 6 provides some useful insights to analyze a complicated system for real signals. Nevertheless, we need to further extend the model such that it is more suitable for applicable practical applications. First, it is needed to exploit the relationship between the model parameters to obtain results of different motion estimation algorithms. Second, one may study the coding efficiency of different transform algorithms related to the model parameters. Hence, a wide variety of further work can be done as a future development of the present investigation.

Appendix A:

A.1 Solution of the reference value, m, in the clustered pixel matching error for adaptive partial distortion search algorithm (CPME-PDS) (Method 1)

$$m = \arg\min_{m} \left\{ E\left[\sum_{n=0}^{N-1} \left[\left(I_t(n) - R(n) \right)^2 - \left(I_t(n) - m \right)^2 \right]^2 \right] \right\}$$

In solving this equation, we have

$$\frac{d}{dm} E\left[\sum_{n=0}^{N-1} \left[\left(I_t(n) - R(n) \right)^2 - \left(I_t(n) - m \right)^2 \right]^2 \right] = 0 \qquad \textbf{(4-1)}$$

$$E\left[\sum_{n=0}^{N-1} \frac{d}{dm} \left[\left(I_t(n) - R(n) \right)^2 - \left(I_t(n) - m \right)^2 \right]^2 \right] = 0$$

$$E\left[\sum_{n=0}^{N-1} 2\left[\left(I_t(n) - R(n) \right)^2 - \left(I_t(n) - m \right)^2 \right] 2\left(I_t(n) - m \right) \right] = 0$$

$$E\left[\sum_{n=0}^{N-1} \left[\left(I_t(n)^2 - 2I(n)R(n) + R(n)^2 \right) - \left(I_t(n)^2 - 2mI(n) + m^2 \right) \right] \left(I_t(n) - m \right) \right] = 0$$

$$E\left[\sum_{n=0}^{N-1} \left[\left(R(n)^2 - m^2 \right) - 2\left(R(n) + m \right) I(n) \right] \left(I_t(n) - m \right) \right] = 0$$

$$E\left[\sum_{n=0}^{N-1} \left[2mI(n)^2 + 2mI(n)R(n) - 3m^2 I(n) - 2R(n)I(n)^2 - mR(n)^2 + R(n)^2 I(n) + m^3 \right] \right] = 0$$

$$E\left[\sum_{n=0}^{N-1} m^3 \right] - 3E\left[\sum_{n=0}^{N-1} I(n)m^2 \right] + E\left[\sum_{n=0}^{N-1} \left(2I(n)^2 + 2I(n)R(n) - R(n)^2 \right) m \right]$$

$$+ E\left[\sum_{n=0}^{N-1} \left(R(n)^2 I(n) - 2R(n)I(n)^2 \right) \right] = 0$$

By substituting $R(n) = I_t(n) - e(n)$ into eqn. (4-1), finally we can have a cubic equation,

$$256m^2 - 3m^2\sum_{n=0}^{N-1} I(n) + m \cdot E\left[\sum_{n=0}^{N-1}\left(2I(n)^2 + 2I(n)\left(I(n)-e(n)\right) - \left(I(n)-e(n)\right)^2\right)\right] \quad \textbf{(4-2)}$$

$$+ E\left[\sum_{n=0}^{N-1}\left(\left(I(n)-e(n)\right)^2 I(n) - 2\left(I(n)-e(n)\right)I(n)^2\right)\right] = 0$$

$$256m^2 - 3m^2\sum_{n=0}^{N-1} I(n) + m \cdot E\left[\sum_{n=0}^{N-1}\left(2I(n)^2 + 2I(n)^2 - 2I(n)e(n) - I(n)^2 + 2I(n)e(n) - e(n)^2\right)\right]$$

$$+ E\left[\sum_{n=0}^{N-1}\left(I(n)^3 - 2I(n)^2 e(n) + I(n)e(n)^2 - 2I(n)^3 + 2I(n)^2 e(n)\right)\right] = 0$$

$$m^3 - 3\overline{I_t}m^2 + \left(3\overline{I_t^2} - \overline{e^2}\right)m + \overline{I_t e^2} - \overline{I_t^3} = 0$$

where $\overline{e^2} = \frac{1}{N} E\left[\sum_{n=0}^{N-1} e(n)^2\right]$, and

$$\overline{I_t e^2} = \frac{1}{N} E\left[\sum_{n=0}^{N-1} I_t(n)e(n)^2\right],$$

The roots of the cubic equation are either all reals or one real and two complex conjugates which depend on the discriminant of the equation.

$$a_1 = -3\overline{I_t};\ a_2 = \left(3\overline{I_t^2} - \overline{e^2}\right);\ a_3 = \overline{I_t e^2} - \overline{I_t^3}$$

$$Q = \frac{3a_2 - a_1^2}{9} = \frac{9\overline{I_t^2} - 3\overline{e^2} - 9\overline{I_t}^2}{9} = \left(\overline{I_t^2} - \overline{I_t}^2\right) - \frac{3\overline{e^2}}{3^2} \approx -3\left(\frac{\overline{e}}{3}\right)^2$$

$$R = \frac{9a_1a_2 - 27a_3 - 2a_1^3}{54}$$

$$= \frac{9\left(-3\overline{I_t}\right)\left(3\overline{I_t^2} - \overline{e^2}\right) - 27\left(\overline{I_t}\,\overline{e^2} - \overline{I_t^3}\right) - 2(-27)\overline{I_t}^3}{54} \approx 0$$

Discriminant, $D=Q^3+R^2$ $D = Q^3 + R^2 \approx -27\left(\frac{\overline{e}}{3}\right)^6$

$\because D < 0 \ \Rightarrow$ all roots are real and unequal roots

We look for real roots for (4-2), such that *m* can be practically useful. The above proof assumes that natural images are dominated by low frequency components. Hence, we let,

$$\overline{I_t}^a \approx \overline{I_t^a},$$

and $$\overline{I_t}\,\overline{e^2} \approx \overline{I_t e^2}$$

These are valid only if the image frame under question consists mainly low frequencies and the standard deviation of $I_t(n)$ is small enough. As a result, it gives

$$m_1 = 2\sqrt{-Q}\frac{\sqrt{3}}{2} - \frac{a_1}{3} = \overline{I_t} + \sqrt{\overline{e^2}} \quad \textbf{(4-3)}$$

$$m_2 = 2\sqrt{-Q}\left(\frac{-\sqrt{3}}{2}\right) - \frac{a_1}{3} = \overline{I_t} - \sqrt{\overline{e^2}}$$

$$m_3 = \frac{-a_1}{3} = \overline{I_t}$$

$$m \approx \begin{cases} \overline{I_t} \\ \overline{I_t} \pm \sqrt{\overline{e^2}} \end{cases}$$

The first approximated root is the mean of pixel values in the target MB. To use this mean as the reference value it already gives a better computational saving when comparing to the PG-PDS, for which we proposed it as a comparison. Intuitively, m is a function of pixel values in a candidate MB, i.e. $m = m\left(R(n)\right)$. The other roots, $\overline{I_t} \pm \sqrt{\overline{e^2}}$ can also be obtained by the following approximation.

$$\overline{R} = \overline{I_t} - \overline{e} \approx \overline{I_t} \pm \sqrt{\overline{e^2}}$$

It indicates that this solution is an approximation of the mean of pixel values in a candidate MB.

A.2 Solution of the reference value, m, in the CPME-PDS (Method 2)

In fact, the solution of $m = \overline{R}$ can also be obtained directly by minimizing the equation,

$$E\left[\sum_{n=0}^{N-1}[e(n)-p(n)]^2\right] \quad \textbf{(A2-1)}$$

To minimizing the equation (A2-1), we substituting $e(n)=I_t(n)-R(n)$ and $p(n)=I_t(n)-m$ into the equation,

$$m=\arg\min_m\left\{E\left[\sum_{n=0}^{N-1}[(I_t(n)-R(n))-(I_t(n)-m)]^2\right]\right\}$$

Hence, m is obtained by solving the equation,

$$\frac{d}{dm}E\left[\sum_{n=0}^{N-1}[(I_t(n)-R(n))-(I_t(n)-m)]^2\right]=0$$

$$\frac{d}{dm}E\left[\sum_{n=0}^{N-1}[(I_t(n)-R(n))^2-2(I_t(n)-R(n))(I_t(n)-m)+(I_t(n)-m)^2]^2\right]=0$$

$$E\left[\sum_{n=0}^{N-1}[2(I_t(n)-R(n))-2(I_t(n)-m)]\right]=0$$

$$E\left[\sum_{n=0}^{N-1}[m-R(n)]\right]=0$$

$$Nm-E\left[\sum_{n=0}^{N-1}R(n)\right]=0$$

$$m=\frac{E\left[\sum_{n=0}^{N-1}R(n)\right]}{N}=\overline{R}$$

Appendix B:

B.1 Detailed derivation of the proposed compound covariance model for motion prediction error

The proposed model is derived based on a first-order Markov statistics (or first order Autoregressive AR(1)) model with image correlation coefficient equal to ρ.

For a block of pixels $f_t(i,j)$ in a frame at time t, the block-based motion compensation uses a matched block $f_{t-1}(i+u,j+v)$ in a reference frame at time t-1 for prediction. Hence, the motion prediction error is given by,

$$e(i,j) = f_t(i,j) - f_{t-1}(i+u, j+v) \quad \textbf{(6-1)}$$

where (u,v) represent the motion vector of the block.

The autocorrelation function, $C_e(I, J)$ of the prediction error is then given by,

$$\begin{aligned} C_e(I,J) &= E\left[\left\{f_t(i,j) - f_{t-1}(i+u, j+v)\right\}\times \right. \\ &\qquad \left.\left\{f_t(i+I, j+J) - f_{t-1}(i+u+I, j+v+J)\right\}\right] \\ &= 2C_f(I,J) - 2C_{f,t}(I-u, J-v) \end{aligned} \quad \textbf{(6-2)}$$

where $C_f(\cdot,\cdot)$ is the autocorrelation function of a frame with correlation coefficient ρ.

$C_{f,t}(\cdot,\cdot)$ is the cross-correlation function between a frame at time t and the reference frame at time t-1.

(* Note that, more precisely,

$$\begin{aligned} C_e(I,J) &= E\left[\left\{f_t(i,j) - f_{t-1}(i+u, j+v)\right\}\times \right. \\ &\qquad \left.\left\{f_t(i+I, j+J) - f_{t-1}(i+u+I, j+v+J)\right\}\right] \\ &= 2C_f(I,J) - C_{f,t}(I-u, J-v) - C_{f,t}(I+u, J+v) \end{aligned}$$

However, if $f_t(\cdot,\cdot) \approx f_{t-1}(\cdot,\cdot)$ and (u, v) is a pair of random variable with -ve

or +ve values, one could expresses it as (6-2). Moreover, these two different expressions will go to identical result because of even function characteristics used in (6-6) *)

We assume that the matched block $f_{t-1}(i+u,j+v)$ can be approximated to the current block $f_t(i, j)$ with reasonable deformation. That is

$$f_t(i+m_x, j+n_y) \approx f_{t-1}(i+u, j+v) \quad \textbf{(6-3)}$$

Hence, first-order Markov statistics model with correlation coefficient, ρ , can be involved. The deformation vector (m_x,n_y) represent the deformation of each pixels in the current block. Note that, in this modeling, the magnitudes of m_x and n_y are not related to that of the u and v directly. They only depend on the matching or correlation between the matched block, $f_{t-1}(i+u, j+v)$ and the current block, $f_t(i, j)$. By substituting (6-3) into (6-2), we can have,

$$C_{f,t}(I-u, J-v) = E[f_{t-1}(i+u, j+v) f_t(i+I, j+J)] = C_{f,t}(I-m_x, J-n_y) \quad \textbf{(6-4)}$$

$$C_{f,t}(I-u, J-v) \approx E[f_t(i+m_x, j+n_y) f_t(i+I, j+J)] = C_{f,t}(I-m_x, J-n_y)$$

By regarding (m_x and n_y) is a pair of independent random variables, and the expectation value of the autocorrelation function, $C_e(I, J)$ is represented as,

$$E[C_e(I,J)] = 2C_f(I,J) - 2E[C_{f,t}(I-m_x, J-n_y)] \quad \textbf{(6-5)}$$

By applying separable 2-dimensional AR(1) model, we get

$$C_f(I,J) = \sigma_f{}^2 \rho^{|I|} \rho^{|J|} \quad \textbf{(6-6)}$$

where $\sigma_f{}^2$ is variance of the pixels value in AR(1) model.

and

$$E[C_{f,t}(I-m_x, J-n_y)] = \sigma_f{}^2 E\left[\rho^{|I-m_x|} \rho^{|J-n_y|}\right] \quad \textbf{(6-7)}$$

For the sake of simplicity, a separable autocorrelation model is our objective. Hence, (6-7) along x-axis is expressed as

$$E\left[C_{f,t}\left(I-m_x, J-n_y\right)\right]=\sigma_f^2 E[\rho^{|I-m_x|}]E[\rho^{|J-n_y|}]; \qquad I \geq 0, J = 0 \tag{6-8}$$

The $E\left[\rho^{|I-m_x|}\right]$ can be computed as

$$E\left[\rho^{|I-m_x|}\right]=\int p(m_x)\rho^{|I-m_x|}dm_x \tag{6-9}$$

where $p(m_x)$ is the probability density function (pdf) of m_x.

At this stage, we make an assumption that pixels in a deformed block tend to deform along a definite direction rather than deformed randomly. In other words, a mean vector of the deformation vectors (m_x,n_y) is not regarded to be zero according to our assumption. Then (6-9) becomes

$$E\left[\rho^{|I-m_x|}\right]=\int p(m_x,\mu_x)\rho^{|I-m_x|}dm_x \tag{6-10}$$

where μ_x is the x-component of the mean of the deformation vector.

An error autocorrelation function must be an even function with respect to *I*. To remedy this directional dependence, we only consider the absolute value of μ_x. Then we further assume that μ_x is randomly distributed. It gives

$$E\left[\rho^{|I-m_x|}\right]=\iint p\left(m_x\middle\|\mu_x\right|)p(\mu_x)\rho^{\|I|-m_x|}dm_x d\mu_x \tag{6-11}$$

where $p(\mu_x)$ is probability density function of μ_x.

Without loss of generality, we use Gaussian distributions to represent the conditional distribution function, $p\left(m_x\middle\|\mu_x\right|)$ and the pdf, $p(\mu_x)$. The $E\left[\rho^{|I-m_x|}\right]$ is then given by

$$E\left[\rho^{|I-m_x|}\right]=\frac{2}{\sigma_\mu\sqrt{2\pi}}\frac{1}{\sigma_{mv}\sqrt{2\pi}}\int_0^\infty e^{\frac{-\mu_x^2}{2\sigma_\mu^2}}\int_{-\infty}^{\infty} e^{\frac{-(m_x-|\mu_x|)^2}{2\sigma_{mv}^2}}\rho^{\|I|-m_x|}dm_x d\mu_x \tag{6-12}$$

where σ_μ is the standard deviation of the mean deformation vector.

σ_{mv} is the standard deviation of the deformation vectors in a single block.

Assuming that the block deformation in the x- and y- dimension can be modeled with same standard deviations σ_μ and σ_{mv}, the error autocorrelation function along x-direction in (6-5) can be expressed as

$$E[C_e(I,J)] = 2\sigma_f{}^2\left\{\rho^{|I|}\rho^{|J|} - E[\rho^{|I-m_x|}]E[\rho^{|J-n_y|}]\right\}; \quad I \geq 0,\ J = 0 \tag{6-13}$$

and the variance-normalized autocorrelation function of the block prediction error is given by

$$\frac{E[C_e(I,0)]}{E[C_e(0,0)]} = \frac{\rho^{|I|}\rho^{|0|} - E[\rho^{|I-m_x|}]E[\rho^{|-n_y|}]}{1 - E[\rho^{|m_x|}]E[\rho^{|n_y|}]}; \quad I \geq 0,\ J = 0 \tag{6-14}$$

The result of (6-14) can be obtained by numerical calculation. However, this form is not convenient for analytical purpose. We have described the derivation of its approximated form below.

We use an expected value of the mean deformation vector in (6-11), instead of using the pdf, $p(\mu_x)$ to approximate the autocorrelation model. Hence,

$$E[\mu_x(\sigma_\mu)] = \bar{\mu}_x(\sigma_\mu) = \frac{2}{\sigma_\mu\sqrt{2\pi}}\int_0^\infty \mu_x e^{\frac{-\mu_x{}^2}{2\sigma_\mu{}^2}} d\mu_x = \sigma_\mu\sqrt{\frac{2}{\pi}} \tag{6-15}$$

We rewrite (6-12) by using this expected value.

$$\begin{aligned} E\left[\rho^{|I-m_x|}\right] &\approx \frac{1}{\sigma_{mv}\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{\frac{-(m_x-|\bar{\mu}_x|)^2}{2\sigma_{mv}{}^2}+||I|-m_x|\ln\rho} dm_x \\ &= \frac{1}{\sigma_{mv}\sqrt{2\pi}}\left[\int_{-\infty}^{|I|} e^{\frac{-(m_x-|\bar{\mu}_x|)^2}{2\sigma_{mv}{}^2}+(|I|-m_x)\ln\rho} dm_x\right. \\ &\qquad \left. + \int_{|I|}^{\infty} e^{\frac{-(m_x-|\bar{\mu}_x|)^2}{2\sigma_{mv}{}^2}+(m-|I|)\ln\rho} dm_x\right] \end{aligned} \tag{6-16}$$

These two integration is expressed involving error function, *erf*(z).

where $erf(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-x^2} dx$

1st term of LHS of (6-16)

$$\frac{1}{\sigma_{mv}\sqrt{2\pi}}\left(\int_{-\infty}^{|I|} e^{\frac{-\left(m_x-|\bar{\mu}_x|\right)^2}{2\sigma_{mv}{}^2}+\left(|I|-m_x\right)\ln\rho} dm_x\right)$$

$$=\frac{1}{\sigma_{mv}\sqrt{2\pi}}\left(\int_{-\infty}^{|I|} e^{\frac{-m_x{}^2+2|\bar{\mu}_x|m_x-|\bar{\mu}_x|^2}{2\sigma_{mv}{}^2}+\frac{-2\sigma_{mv}{}^2 m_x\ln\rho+2\sigma_{mv}{}^2|I|\ln\rho}{2\sigma_{mv}{}^2}} dm_x\right)$$

$$=\frac{1}{\sigma_{mv}\sqrt{2\pi}}\left(\int_{-\infty}^{|I|} e^{\frac{-m_x{}^2}{2\sigma_{mv}{}^2}+\frac{2|\bar{\mu}_x|m_x-2\sigma_{mv}{}^2 m_x\ln\rho}{2\sigma_{mv}{}^2}+\frac{-|\bar{\mu}_x|^2+2\sigma_{mv}{}^2|I|\ln\rho}{2\sigma_{mv}{}^2}} dm_x\right)$$

$$=\frac{1}{\sigma_{mv}\sqrt{2\pi}}\left(\int_{-\infty}^{|I|} e^{\frac{-m_x{}^2}{2\sigma_{mv}{}^2}+\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)m_x}{\sigma_{mv}{}^2}+\frac{-|\bar{\mu}_x|^2+2\sigma_{mv}{}^2|I|\ln\rho}{2\sigma_{mv}{}^2}} dm_x\right)$$

$$\because \frac{-m_x{}^2}{2\sigma_{mv}{}^2}+\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)m_x}{\sigma_{mv}{}^2}=-\left(\frac{m_x}{\sqrt{2}\sigma_{mv}}-\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)}{\sqrt{2}\sigma_{mv}}\right)^2+\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)^2}{2\sigma_{mv}{}^2}$$

$$=\frac{1}{\sigma_{mv}\sqrt{2\pi}}\left(\int_{-\infty}^{|I|} e^{-\left(\frac{m_x}{\sqrt{2}\sigma_{mv}}-\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)}{\sqrt{2}\sigma_{mv}}\right)^2+\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)^2}{2\sigma_{mv}{}^2}+\frac{-|\bar{\mu}_x|^2+2\sigma_{mv}{}^2|I|\ln\rho}{2\sigma_{mv}{}^2}} dm_x\right)$$

$$=\frac{1}{\sigma_{mv}\sqrt{2\pi}}e^{\left(|I|-|\bar{\mu}_x|\right)\ln\rho+\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left(\int_{-\infty}^{|I|} e^{-\left(\frac{m_x}{\sqrt{2}\sigma_{mv}}-\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)}{\sqrt{2}\sigma_{mv}}\right)^2} dm_x\right)$$

$$\text{let } \zeta=\frac{m_x}{\sqrt{2}\sigma_{mv}}-\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)}{\sqrt{2}\sigma_{mv}}\ ;\ d\zeta=\frac{dm_x}{\sqrt{2}\sigma_{mv}}\ ;\ I'=\frac{|I|}{\sqrt{2}\sigma_{mv}}-\frac{(|\bar{\mu}_x|-\sigma_{mv}{}^2\ln\rho)}{\sqrt{2}\sigma_{mv}}$$

$$=\frac{1}{\sqrt{\pi}}e^{\left(|I|-|\bar{\mu}_x|\right)\ln\rho+\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left(\int_{-\infty}^{I'} e^{-(\zeta)^2} d\zeta\right)$$

$$=\frac{1}{\sqrt{\pi}}e^{\left(|I|-|\bar{\mu}_x|\right)\ln\rho+\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left(\int_{-\infty}^{0} e^{-(\zeta)^2} d\zeta+\int_{0}^{I'} e^{-(\zeta)^2} d\zeta\right)$$

$$\because \int_{-\infty}^{0} e^{-(\zeta)^2} d\zeta=\frac{\sqrt{\pi}}{2}\ \text{and}\ erf(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-x^2} dx$$

$$= \frac{1}{2} e^{(|I|-|\bar{\mu}_x|)\ln\rho + \frac{(\sigma_{mv}\ln\rho)^2}{2}} \left(1 + erf(z)\right)$$

$$= \frac{1}{2} e^{(|I|-|\bar{\mu}_x|)\ln\rho + \frac{(\sigma_{mv}\ln\rho)^2}{2}} \left(1 + erf\left(\frac{|I|}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| - {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}}\right)\right)$$

2nd term of LHS of (6-16)

$$\frac{1}{\sigma_{mv}\sqrt{2\pi}} \left(\int_{|I|}^{\infty} e^{\frac{-(m_x - |\bar{\mu}_x|)^2}{2{\sigma_{mv}}^2} + (m_x - |I|)\ln\rho} dm_x \right)$$

$$= \frac{1}{\sigma_{mv}\sqrt{2\pi}} \left(\int_{|I|}^{\infty} e^{\frac{-{m_x}^2 + 2|\bar{\mu}_x| m_x - |\bar{\mu}_x|^2}{2{\sigma_{mv}}^2} + \frac{2{\sigma_{mv}}^2 m_x \ln\rho - 2{\sigma_{mv}}^2 |I|\ln\rho}{2{\sigma_{mv}}^2}} dm_x \right)$$

$$= \frac{1}{\sigma_{mv}\sqrt{2\pi}} \left(\int_{|I|}^{\infty} e^{\frac{-{m_x}^2}{2{\sigma_{mv}}^2} + \frac{2|\bar{\mu}_x| m_x + 2{\sigma_{mv}}^2 m_x \ln\rho}{2{\sigma_{mv}}^2} + \frac{-|\bar{\mu}_x|^2 - 2{\sigma_{mv}}^2 |I|\ln\rho}{2{\sigma_{mv}}^2}} dm_x \right)$$

$$= \frac{1}{\sigma_{mv}\sqrt{2\pi}} \left(\int_{|I|}^{\infty} e^{\frac{-{m_x}^2}{2{\sigma_{mv}}^2} + \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho) m_x}{{\sigma_{mv}}^2} + \frac{-|\bar{\mu}_x|^2 - 2{\sigma_{mv}}^2 |I|\ln\rho}{2{\sigma_{mv}}^2}} dm_x \right)$$

$$\because \frac{-{m_x}^2}{2{\sigma_{mv}}^2} + \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho) m_x}{{\sigma_{mv}}^2} = -\left(\frac{m_x}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}} \right)^2 + \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)^2}{2{\sigma_{mv}}^2}$$

$$= \frac{1}{\sigma_{mv}\sqrt{2\pi}} \left(\int_{|I|}^{\infty} e^{-\left(\frac{m_x}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}} \right)^2 + \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)^2}{2{\sigma_{mv}}^2} + \frac{-|\bar{\mu}_x|^2 - 2{\sigma_{mv}}^2 |I|\ln\rho}{2{\sigma_{mv}}^2}} dm_x \right)$$

$$= \frac{1}{\sigma_{mv}\sqrt{2\pi}} e^{(|\bar{\mu}_x| - |I|)\ln\rho + \frac{(\sigma_{mv}\ln\rho)^2}{2}} \left(\int_{|I|}^{\infty} e^{-\left(\frac{m_x}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}} \right)^2} dm_x \right)$$

let $\zeta = \frac{m_x}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}}$ and $d\zeta = \frac{dm_x}{\sqrt{2}\sigma_{mv}}$; $I'' = \frac{|I|}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}}$

$$= \frac{1}{\sqrt{\pi}} e^{(|\bar{\mu}_x| - |I|)\ln\rho + \frac{(\sigma_{mv}\ln\rho)^2}{2}} \left(\int_{I''}^{\infty} e^{-(\zeta)^2} d\zeta \right)$$

$$= \frac{1}{2} e^{(|\bar{\mu}_x| - |I|)\ln\rho + \frac{(\sigma_{mv}\ln\rho)^2}{2}} \left[1 - erf\left(I''\right)\right]$$

1st term of LHS of (6-16)

$$= \frac{1}{2} e^{(|I| - |\bar{\mu}_x|)\ln\rho + \frac{(\sigma_{mv}\ln\rho)^2}{2}} \left[1 + erf\left(\frac{|I|}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| - {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}}\right)\right]$$

2nd term of LHS of (6-16)

$$= \frac{1}{\sqrt{\pi}} e^{(|\bar{\mu}_x| - |I|)\ln\rho + \frac{(\sigma_{mv}\ln\rho)^2}{2}} \left[1 - erf\left(\frac{|I|}{\sqrt{2}\sigma_{mv}} - \frac{(|\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho)}{\sqrt{2}\sigma_{mv}}\right)\right]$$

Finally, we have

$$\begin{aligned} E\left[\rho^{|I - m_x|}\right] &\approx \frac{1}{2} e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}} \left\{ \rho^{(|I| - |\bar{\mu}_x|)} \left[1 + erf\left(\frac{|I| - |\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right] \right. \\ &\quad \left. + \quad \rho^{-(|I| - |\bar{\mu}_x|)} \left[1 - erf\left(\frac{|I| - |\bar{\mu}_x| - {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right] \right\} \\ &= \rho^{|I|} \frac{1}{2} e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}} \left\{ \rho^{|\bar{\mu}_x|} \left[1 + erf\left(\frac{|I| - |\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right] \right. \\ &\quad \left. + \quad \rho^{-(2|I| - |\bar{\mu}_x|)} \left[1 - erf\left(\frac{|I| - |\bar{\mu}_x| - {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right] \right\} \\ &= \rho^{|I|} \tilde{R}(I, \bar{\mu}_x, \sigma_{mv}, \rho) \end{aligned} \quad \textbf{(6-17)}$$

where

$$\begin{aligned} \tilde{R}(I, \bar{\mu}_x, \sigma_{mv}, \rho) &= \frac{1}{2} e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}} \left\{ \rho^{|\bar{\mu}_x|} \left[1 + erf\left(\frac{|I| - |\bar{\mu}_x| + {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right] \right. \\ &\quad \left. + \quad \rho^{-(2|I| - |\bar{\mu}_x|)} \left[1 - erf\left(\frac{|I| - |\bar{\mu}_x| - {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right] \right\} \end{aligned}$$

Now, using the 6-17), we can approximate the variance-normalized autocorrelation function of the block prediction error which given by (6-14) as

$$\frac{E[C_e(I,0)]}{E[C_e(0,0)]} \approx \tilde{C}_{e,\bar{\mu}_x,\sigma_{mv}}(I,0) = \rho^{|I|}\rho^{|0|}\frac{1-\tilde{R}(I,\bar{\mu}_x,\sigma_{mv},\rho)\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)}{1-\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)}$$
$$= \rho^{|I|}\rho^{|J|}\Re(I,J,\bar{\mu}_x,\sigma_{mv},\rho)\Big|_{J=0} \quad \textbf{(6-18)}$$

Finally, we express a separable 2-D variance-normalized autocorrelation function as,

$$\frac{E[C_e(I,0)]E[C_e(0,J)]}{E[C_e(0,0)]^2} = \rho_x^{|I|}\rho_y^{|J|}\Re(I,0,\bar{\mu}_x,\sigma_{mvx},\rho_x)\Re(0,J,\bar{\mu}_y,\sigma_{mvy},\rho_y) \quad \textbf{(6-19)}$$

B.2 Deduce the compound covariance model, CP Model (2-39), from (6-18).

To derive the (2-39), we assume that a video sequence with very low motion activity is under simulation, and thus $\overline{\mu}_x$ and σ_{mv} are set equal to zero and 0.5 respectively. Using the properties of *erf*(z),

$$erf(z) = \begin{cases} \frac{2}{\sqrt{\pi}} e^{-z^2} z\left[1 + \frac{2z^2}{1\cdot 3} + \frac{(2z^2)^2}{1\cdot 3\cdot 5} + \cdots\right] & if \;\; z << 1 \\ 1 & if \;\; z >> 1 \end{cases}$$

, an approximation of (6-17) can be obtained. For |*I*| equal to zero, we express the $erf(\cdot)$ in (6-17) as the first term of its expanded series, and for |*I*| greater or equal to one, the $erf(\cdot)$ is expressed as unity. Hence, (6-17) is approximated as

$$E\left[\rho^{|I-m_x|}\right] = \rho^{|I|}\frac{1}{2} e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}} \left\{\rho^{|\overline{\mu}_x|}\left[1 + erf\left(\frac{|I| - |\overline{\mu}_x| + {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right] + \rho^{-(2|I| - |\overline{\mu}_x|)}\left[1 - erf\left(\frac{|I| - |\overline{\mu}_x| - {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)\right]\right\}$$

$\because$ $\overline{\mu}_x$=0, $\rho^{|\overline{\mu}_x|} = 1$ and

$$erf\left(\frac{|I| - |\overline{\mu}_x| + {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right) \approx \frac{2}{\sqrt{\pi}} e^{-\left(\frac{{\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right)^2} \frac{{\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}} \delta(|I|) + [1 - \delta(|I|)]$$

1st term is contributed by $I = 0$ and the remained terms are for $I > 0$

similarly ,

$$-erf\left(\frac{|I| - |\overline{\mu}_x| - {\sigma_{mv}}^2 \ln\rho}{\sigma_{mv}\sqrt{2}}\right) \approx -\left\{\frac{2}{\sqrt{\pi}} e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2} \left(\frac{-\sigma_{mv}\ln\rho}{\sqrt{2}}\right)\delta(|I|) + [1 - \delta(|I|)]\right\}$$

$$E\left[\rho^{|I-m_x|}\right] \approx \rho^{|I|}\frac{1}{2}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left\{\left[1+\frac{2}{\sqrt{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\delta(|I|)+(1-\delta(|I|))\right]\right.$$
$$\left. + \quad \rho^{-2|I|}\left[1-\frac{2}{\sqrt{\pi}}e^{-\left(\frac{-\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\cdot\frac{-\sigma_{mv}\ln\rho}{\sqrt{2}}\delta(|I|)-(1-\delta(|I|))\right]\right\}$$

for the 2nd middle bracket term in the braces,

$$\rho^{-2|I|}\left[1+\frac{2}{\sqrt{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\cdot\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\delta(|I|)-(1-\delta(|I|))\right]=0 \quad \text{, if } I\neq 0$$

if $I = 0$

$$E\left[\rho^{|I-m_x|}\right] \approx \rho^{|I|}\frac{1}{2}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left\{\left[1+\frac{2}{\sqrt{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right]\right.$$
$$\left. + \quad \left[1+\frac{2}{\sqrt{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\cdot\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right]\right\}$$
$$= \rho^{|I|}\frac{1}{2}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left\{\left[2+\frac{4}{\sqrt{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right]\right.$$

Hence, for $I \geq 0$

$$E\left[\rho^{|I-m_x|}\right]=\rho^{|I|}e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left[1+\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\sigma_{mv}\ln\rho\cdot\delta(|I|)\right] \qquad \textbf{(6-20)}$$

Let $\beta=\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\sigma_{mv}\ln\rho}{\sqrt{2}}\right)^2}\sigma_{mv}\ln\rho$, and substituted into **(6-20)**.

Now, we have

$\tilde{R}(I,\bar{\mu}_x,\sigma_{mv},\rho)=e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left[1+\beta\cdot\delta(|I|)\right]$ and substituted into (6-18)

$$\frac{E[C_e(I,0)]}{E[C_e(0,0)]} = \tilde{C}_{e,\bar{\mu}_x,\sigma_{mv}}(I,0) = \rho^{|I|}\rho^{|0|}\frac{1-\tilde{R}(I,\bar{\mu}_x,\sigma_{mv},\rho)\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)}{1-\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)\tilde{R}(0,\bar{\mu}_x,\sigma_{mv},\rho)}$$

$$\tilde{C}_{e,\bar{\mu}_x,\sigma_{mv}}(I,0) = \rho^{|I|}\frac{1-\left[e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left[1+\beta\cdot\delta(|I|)\right]\right]\left[e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left[1+\beta\right]\right]}{1-\left[e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left[1+\beta\right]\right]\left[e^{\frac{(\sigma_{mv}\ln\rho)^2}{2}}\left[1+\beta\right]\right]}$$

$$= \rho^{|I|}\frac{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\delta(|I|)\right]\cdot\left[1+\beta\right]}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}$$

$$= \rho^{|I|}\frac{1-e^{(\sigma_{mv}\ln\rho)^2}\left\{1+\beta+\beta\delta(|I|)+\beta^2\delta(|I|)\right\}}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}$$

$$= \rho^{|I|}\frac{1-e^{(\sigma_{mv}\ln\rho)^2}(1+\beta)-e^{(\sigma_{mv}\ln\rho)^2}\left\{\beta\delta(|I|)+\beta^2\delta(|I|)\right\}}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}$$

$$= \rho^{|I|}\frac{1-e^{(\sigma_{mv}\ln\rho)^2}(1+\beta)-e^{(\sigma_{mv}\ln\rho)^2}\left\{(1+\beta)^2-1-\beta\right\}\delta(|I|)}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}$$

$$= \rho^{|I|}\frac{1-e^{(\sigma_{mv}\ln\rho)^2}(1+\beta)+(1-1)\delta(|I|)-e^{(\sigma_{mv}\ln\rho)^2}\left\{(1+\beta)^2-(1+\beta)\right\}\delta(|I|)}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}$$

$$= \rho^{|I|}\frac{1-e^{(\sigma_{mv}\ln\rho)^2}(1+\beta)+1\delta(|I|)-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2\delta(|I|)-1\delta(|I|)+e^{(\sigma_{mv}\ln\rho)^2}(1+\beta)\delta(|I|)}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}$$

As a result,

$$\begin{aligned}\tilde{C}_{e,\bar{\mu}_x,\sigma_{mv}}(I,0) &= \rho^{|I|}\left\{\frac{1-e^{(\sigma_{mv}\ln\rho)^2}(1+\beta)}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}+\frac{\left[1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2\right]\delta(|I|)}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}\right.\\ &\qquad\left.+\frac{-\left[1-e^{(\sigma_{mv}\ln\rho)^2}(1+\beta)\right]\delta(|I|)}{1-e^{(\sigma_{mv}\ln\rho)^2}\left[1+\beta\right]^2}\right\}\\ &= \rho^{|I|}\left[A+(1-A)\delta(|I|)\right]\end{aligned} \qquad \textbf{(6-21)}$$

Where

$$A = \frac{1 - e^{(\sigma_{mv} \ln \rho)^2} [1 + \beta]}{1 - e^{(\sigma_{mv} \ln \rho)^2} [1 + \beta]^2} \quad \textbf{(6-22)}$$

The numerical value of A is equal to 0.497 with ρ = 0.95 and σ_{mv} = 0.5. And CP Model is arrived.

References

[1] ITU-T, “Video Codec For Audiovisual Services At P×64 Kbit/s”, ITU-T Recommendation H.261, March 1993.

[2] ISO/IEC, “Information technology -- coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- part 2: video”, ISO/IEC 11172-2 (MPEG-1), March 1993.

[3] ITU-T and ISO/IEC, “Information technology -- generic coding of moving pictures and associated audio information: video”, ITU-T Recommendation H.262 - ISO/IEC 13818-2 (MPEG-2), November 1994.

[4] ITU-T, “Video coding for low bit rate communication”, ITU-T Recommendation H.263, Version 1, November 1995; Version 2, January 1998.

[5] ISO/IEC, “Information technology -- coding of audio-visual objects: Part 2 visual”, ISO/IEC 14496-2 (MPEG-4), October 1998.

[6] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)

[7] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard and Ajay Luthra, “Overview of the H.264/AVC Video Coding Standard”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003.

[8] Jaswant R. Jain, Anil K. Jain, “Displacement Measurement and Its Application in Interframe Image Coding”, IEEE Transactions on Communications, Vol. COM-29, pp1799-1808, December 1984.

[9] Peter Strobach, “Tree-Structured Scene Adaptive Coder”, IEEE Transactions on Communications, Vol. 38, No. 4, pp. 477-486, April 1990.

[10] Li Jin, Lin Xinggang and Wu Youshou, “Color image sequence coding with variable block size motion compensation”, Proceedings, 1993 IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering, TENCON '93. Vol. 2, Issue: 0, pp. 974-977 , October 1993.

[11] Gary J. Sullivan and Richard L. Baker, “Efficient Quadtree Coding of Images and Video”, IEEE Transactions on Image Processing, Vol. 3, No. 3, pp. 327-331, May 1994.

[12] Hanan A. Mahmoud and Magdy A. Bayoumi, “An efficient low-bit rate adaptive mesh-based motion compensation technique”, 2000 The 7th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2000. Vol. 1, pp. 491-494, December 2000.

[13] Hans Georg Musmann, Michael Hötter and Jörn Ostermann, “Object-oriented Analysis-synthesis Coding of Moving Images”, Signal Processing: Image Communication 1, pp.117-138, 1989.

[14] Yutaka Yokoyama, Yoshihiro Miyamoto and Mutsumi Ohta, “Very Low Bit Rate Video Coding Using Arbitrarily Shaped Region-Based Motion Compensation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 6, pp. 500-507, December 1995.

[15] Yuichiro Nakaya and Hiroshi Harashima, “Motion compensation based on spatial transformations”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, 3 , pp. 339 -356, 366-7, June 1994.

[16] M. Ghanbari, S. de Faria, I. N. Goh and K. T. Tan, “Motion Compensation for Very Low Bit-Rate Video”, Signal Processing: Image Communication, Vol. 7, pp. 567-580, 1995.

[17] A. Murat Tekalp, Yucel Altunbasak, and Gozde Bozdagi, “Two- Versus Three-Dimensional Object-Based Video Compression”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 2, pp. 391-397, April 1997.

[18] Christoph Stiller, “Object-Based Estimation of Dense Motion Fields”, IEEE Transactions on Image Processing, Vol. 6, No. 2, pp. 234-250, February 1997.

[19] Soo-Chul Han and Christine I. Podilchuk, “Video Compression With Dense Motion Fields”, IEEE Transactions on Image Processing, Vol. 10, No. 11, pp. 1602-1612, November 2001.

[20] Bernd Girod, “The Efficiency of Motion-Compensating Prediction for Hybrid Coding of Video Sequences”, IEEE Journal on Selected Areas in Communications, Vol. SAC-5, No. 7, pp. 1140-1154, August 1987.

[21] Bernd Girod, “Why B-Pictures Work: A Theory of Multi-Hypothesis Motion-Compensated Prediction”, Proceedings, International Conference on Image Processing 1998, ICIP 98., Vol. 2, pp. 213-217,October 1998.

[22] ISO/IEC JTC1/SC29/WG11 N3908, “MPEG-4 Video Verification Model version 18.0”, January 2001.

[23] Bern Girod, “Motion-Compensation Prediction with Fractional-Pel Accuracy”, IEEE Transactions on Communications, Vol. 41, pp. 604-612, April 1993.

[24] S. Nogaki and M. Ohta, “An overlapped block motion compensation for high quality motion picture coding”, Proceedings, International Symposium on Circuits and Systems, Vol. 1, pp. 184-187, May 1992.

[25] Michael T. Orchard and Gary J. Sullivan, “Overlapped block motion compensation: An estimation-theoretic approach”, IEEE Transaction on Image Processing, Vol. 3, pp. 693-699, May 1994.

[26] Bo Tao and Michael T. Orchard, “A parametric Solution Optimal Overlapped Block Motion Compensation”, IEEE Transactions on Image Processing, Vol. 10, No. 3, pp. 341-350, March 2001.

[27] Wentao Zheng, Yoshiaki Shishikui, Masahide Naemura, Yasuaki Kanatsugu, and Susumu Itoh, “Analysis of Space-Dependent Characteristics of Motion-Compensated Frame Differences Based on a Statistical Motion Distribution Model”, IEEE Transactions on Image Processing, Vol. 11, pp. 377-386, April 2002.

[28] Barry G. Haskell, Atul Puri, and Arun N. Netravali, “Digital video: an introduction to MPEG-2”, New York, N.Y. : Chapman & Hall, 1997.

[29] Roland Mech and Michael Wollborn, “A Noise Robust Method for Segmentation of Moving Objects in Video Sequences”, Proceedings, International Conference on Acoustic, Speech and Signal, Munich, pp. 2657-2660, April 1997.

[30] Til Aach, André Kaup and Rudolf Mester, “Statistical model-based change detection in moving video”, Signal Processing, Vol. 31, No. 2, pp. 165-180, March 1993.

[31] A. Neri, S. Colonnese, G. Russo, “Video Sequence Segmentation for Object-based Coders using Higher Order Statistics”, ISCAS ‘97, Hongkong, June 1997.

[32] Philippe Salembier and Montse Pardàs, “Hierarchical Morphological Segmentation for Image Sequence Coding”, IEEE Transactions on Image Processing, Vol.3, No.5, pp. 639-651, September 1994.

[33] Luc Vincent and Pierre Soille, “Watersheds in digital spaces: an efficient algorithm based on immersion simulations“, IEEE Transactions on PAMI, Vol.13, No. 6, pp. 583-598, June 1991.

[34] Aydin Alatan, Levent Onural, Michael Wollborn, Roland Mech, Ertem Tuncel and Thomas Sikora, “Image Sequence Analysus for Emerging Interactive Multimedia Services – The COST 211 Framework”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 8, No.7, pp. 802-813, November 1998.

[35] Michael Kass, Andrew Witkin, and Demetri Terzopoulos, “Snakes: Active Contour Models”, Internalional Journal of Computer Vision, Vol. 1, pp. 321-331, January 1988.

[36] Muriel Gastaud and Michael Barlaud, “Video segmentation using active contours on a group of pictures”, Image Processing, 2002, Proceedings, 2002 International Conference on, Vol. 2, pp. 81 – 84, 22-25 Sept. 2002.

[37] Shijun Sun, David R. Haynor, and Yongmin Kim, “Semiautomatic Video Object Segmentation Using VSnakes”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, pp. 75 – 82, Jan. 2003.

[38] Noboru Yamaguchi, Takashi Ida and Toshiaki Watanabe, “A binary shape coding method using modified MMR”, Proceedings, International Conference on Image Processing 1997, Vol. 1, pp. 504-507, Oct. 1997.

[39] Shi Hwa Lee, Dae-Sung Cho, Yu-Shin Cho, Sehoon Son, E.S. Jang, Jae-Seob Shin, and Yang Seok Seo, “Binary shape coding using 1-D distance values from baseline”, Proceedings, International Conference on Image Processing 1997, Vol. 1, pp. 508-511, Oct. 1997.

[40] Chil-Cheang Ma, Mei-Juan Chen, and Po-Yuen Cheng, “Efficient shape coding algorithm for object-oriented video in MPEG-4”, International Conference on Consumer Electronics 1999, ICCE 1999, pp. 174-175, June 1999.

[41] Yin-Tsung Hwang, Yun-Chiang Wang, and Shi-Shen Wang, “An efficient shape coding scheme and its codec design”, IEEE Workshop on Signal Processing Systems 2001, pp. 225-232, Sept. 2001.

[42] Janez Zaletelj, and Jurij F. Tasič, “B-spline optimization and its application to video object shape coding”, Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis 2001, ISPA 2001, pp. 80-85, June 2001.

[43] Huitao Luo, “Efficient Image-Dependent Object Shape Coding”, Proceedings, International Conference on Image Processing 2002, Vol. 1, pp. 169-172, Sept. 2002.

[44] Frank Bossen, and Touradi Ebrahimi, “A simple and efficient binary shape coding technique based on bitmap representation”, IEEE International Conference on Acoustics, Speech, and Signal Processing 1997, ICASSP-97, Vol. 4, pp. 3129-3132 ,April 1997.

[45] N. Brady, F. Bossen, and N. Murphy, “Context-based arithmetic encoding of 2D shape sequences”, Proceedings, International Conference on Image Processing 1997, Vol. 1, pp. 29-32, Oct. 1997.

[46] Jörn Ostermann, “Efficient encoding of binary shapes using MPEG-4”, Proceedings, International Conference on Image Processing 1998, ICIP 98, Vol. 1, pp. 295-298, Oct. 1998.

[47] Mei-Juan Chen, Yuan-Pin Hsieh, and Yu-Pin Wang, “Multi-resolution shape coding algorithm for MPEG-4”, IEEE Transactions on Consumer Electronics, Vol. 46, Issue: 3, pp. 505-513, Aug. 2000.

[48] Tzu-Ming Liu, Bai-Jue Shieh and Chen-Yi Lee, “An efficient modeling codec architecture for binary shape coding”, IEEE International Symposium on Circuits and Systems 2002, ISCAS 2002, Vol. 2, pp. 316-319, May 2002.

[49] Aggelos K. Katsaggelos, Lisimachos P. Kondi, Fabian W. Meier, Jörn Ostermann, and Guido M. Schuster, “MPEG-4 and Rate-Distortion-Based Shape-Coding Techniques”, Proceedings of the IEEE, Vol. 86, No. 6, pp. 1126-1154, June 1998.

[50] Detlev Marpe, Gabi Blättermann, Guido Heising, and Thomas Wiegand, “Video Compression Using Context-Based Adaptive Arithmetic Coding”, Proceedings, International Conference on Image Processing, 2001, Vol. 3, pp. 558-561,October 2001.

[51] Jae-Beom Lee, Jin-Soo Cho, and Alexandros Eleftheriadis, “Optimal Buffered Compression and Coding Mode Selection for MPEG-4 Shape Coding”, IEEE Transactions On Image Processing, Vol. 10, No. 5, pp. 686-700, May 2001.

[52] Wei-Ge Chen, Chuang Gu, Ming-Chieh Lee, “Repetitive and morphological padding for object-based video coding”, Proceedings of International Conference on Image Processing, ICIP'97, Vol. 1, pp. 373-376, Oct. 1997.

[53] E. A. Edirisnghe, J. Jiang and C. Grecos, “Shape Adaptive Padding for MPEG-4”, IEEE Transactions on Consumer Electronics, Vol. 46, No.3, pp. 514-520,August 2000.

[54] Jie Chen and K. J. Ray Liu, “Transform Domain Motion Estimation without Macroblock-based Repetitive Padding for MPEG-4 Video”, Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, 1998, ISCAS '98, Vol. 4, pp. 130-133, 31 May-3 June 1998.

[55] André Kaup, “Adaptive low-pass extrapolation for object-based texture coding of moving video”, Proceeding, Visual Communications and Image Processing ‘97, SPIE, Vol. 3024, pp. 731–741, February 1997.

[56] André Kaup, “Object-Based Texture Coding of Moving Video in MPEG-4”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 1, pp. 5-15, February 1999.

[57] Guobin Shen, Bing Zeng, and Ming Lei Liou, “Arbitrarily Shaped Transform Coding Based on a New Padding Technique”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, pp. 67-79, January 2001.

[58] Michael Gilge, Thomas Engelhardt, and Ralf Mehlan, “Coding of arbitrarily shaped image segments based on a generalized orthogonal transform”, Signal Processing: Image Communication, Vol. 1, pp. 153–180, October 1989.

[59] Thomas Sikora and Béla Makai, “Shape-Adaptive DCT for Generic Coding Video”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 1, pp. 59-62, February 1995.

[60] Thomas Sikora, Sven Bauer, and Béla Makai, “Efficiency of Shape-Adaptive 2-D Transforms for Coding of Arbitrarily Shaped Image Segments”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 3, pp. 254-258, June 1995.

[61] M. Bi, S. H. Ong and Y. H. Ang, “Comment on “Shape-Adaptive DCT for Generic Coding of Video””, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No.6, pp. 686-688,December 1996.

[62] Peter Kauff, Béla Makai, Stefan Rauthenberg, Ulrich Gölz, Jan L. P. De Lameillieure, and Thomas Sikora, “Functional Coding of Video Using a Shape-Adaptive DCT Algorithm and an Object-Based Motion Prediction Toolbox”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 1, pp. 181- 196, February 1997.

[63] Peter Kauff and Klaas Schüür, “Shape-Adaptive DCT with Block-Based DC Separation and DC Correction”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 8, No. 3, pp. 237-242, June 1998.

[64] Yoshiaki Shishikui and Shinichi Sakaida, “Region Support DCT (RS-DCT) for Coding of Arbitrarily Shaped Texture”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 5, pp. 320-330, May 2002.

[65] Joo-Hee Moon, Ji-Heon Kweon, and Hae-Kwang Kim, “Boundary Block-Merging (BBM) Technique for Efficient Texture Coding of Arbitrarily Shaped Object”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 1, pp. 35-43, February 1999.

[66] H. Gharavi and Mike Mills, “Blockmatching Motion Estimation Algorithms – New Result”, IEEE Transactions on Circuits and Systems, Vol. 37, No. 5, pp. 649-651, May 1990.

[67] Mei-Juan Chen, Liang-Gee Chen, Tzi-Dar Chiueh, and Yung-Pin Lee, “A New Block-Matching Criterion for Motion Estimation and its Implementation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 3, pp. 231-236, June 1995.

[68] Arild Fuldseth and Tor A. Ramstad, “A New Error Criterion for Block Based Motion Estimation”, Proceedings, International Conference on Image Processing 1995, Vol. 3, pp. 188-191, October 1995.

[69] S. Kappagantula and K. R. Rao, “Motion Compensated Interframe Image Prediction”, IEEE Transaction on Communications, Vol. COM-33, No. 9, pp. 1011-1015, September 1985.

[70] Bern Girod, “Rate-Constrained Motion Estimation”, Proceedings of the SPIE Conference on Visual Communications and Image Processing, pp.1026-1034, September 1994.

[71] Yair Shoham, Allen Gersho, “Efficient bit allocation for an arbitrary set of quantizers [speech coding]”, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 36, Issue: 9, pp. 1445-1453, September 1988.

[72] Gary J. Sullivan and Thomas Wiegand, “Rate-distortion optimization for video compression” IEEE Signal Processing Magazine, Vol. 15, Issue: 6, pp. 74-90, November 1998.

[73] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion compensated interframe coding for video conferencing,” Proceedings. National Telecommunication Conference, pp. G5.3.1-5.3.5, November 29-December 3, 1981.

[74] Ram Srinivasan and K. R. Rao, “Predictive Coding Based on Efficient Motion Estimation,” IEEE Transactions on Communications, Vol. COM-33, pp. 888-896, September 1985.

[75] Reoxiang Li, Bing Zeng and Liou, M.L., “A New Three-Step Search Algorithm for Block Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, No. 4, pp. 438-442, August 1994.

[76] A. Puri, H. M. Hang, and D. L. Schilling, “An efficient block-matching algorithm for motion compensated coding,” Proceedings. IEEE ICASSP 1987, pp. 25.4.1-25.4.4., 1987.

[77] Keith Hung-Kei Chow and Ming L. Liou, “Genetic Motion Search Algorithm for Video Compression”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 3, No. 6, pp. 440-445, December 1993.

[78] Lurng-Kuo Liu and Ephraim Feig, “A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 4, pp. 419-422, August 1996.

[79] Mohammed E. Al-Mualla, C. Nishan Canagarajah and David R. Bull, “Simplex Minimization for Single- and Multiple-Reference Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 12, pp. 1209-1220, December 1998.

[80] Shan Zhu and Kai-Kuang Ma, “A New Diamond Search Algorithm for Fast Block Matching Motion Estimation”, Proceedings, International Conference on Information, Communication and Signal Processing (ICICS’97), pp. 292-296, September. 1997.

[81] Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath and Ashraf Ali Kassim, “A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 8, No. 4, pp. 369-377, August 1998.

[82] Lai-Man Po and Wing-Chung Ma, “A Novel Four-Step Search Algorithm for Fast Block Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 3, pp. 313-317, June 1998.

[83] Fang-Hsuan Cheng and San-Nan Sun, “New Fast and Efficient Two-Step Search Algorithm for Block Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 7, pp. 977-983, October 1999.

[84] Ce Zhu, Xiao Lin and Lap-Pui Chau, “Hexagon-Based Search Pattern for Fast Block Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 5, pp. 349-355, May 2002.

[85] Prabhudev Irappa Hosur and Kai-Kuang Ma, “Motion Vector Field Adaptive Fast Motion Estimation,” Second International Conference on Information, Communications and Signal Processing (ICICS ’99), December 1999.

[86] Alexis M. Tourapis, Oscar C. Au and Ming L. Liou, “Highly Efficient Predictive Zonal Algorithms for Fast Block-Matching Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 10, pp. 934-947, October 2002.

[87] M. Bierling, “Displacement estimation by hierarchical block matching” Proceedings, VCIP’88, Vol. 1001, pp. 942–951, 1988.

[88] K. Metin Uz, Martin Vetterli, and Didier J. LeGall, “Interpolative multiresolution coding of advanced television with compatible subchannels” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 1, No. 1, pp. 86–99, March 1991.

[89] Bede Liu and André Zaccarin, “New Fast Algorithm for the Estimation of Block Motion Vectors”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 3, No. 2, pp. 148-157, April 1993.

[90] Y. L. Chan and W. C. Siu, “New Adaptive Pixel Decimation for Block Motion Vector Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 1, pp. 113-118, February 1996.

[91] Yankang Wang, Yanqun Wang, and Hideo Kuroda, “A Globally Adaptive Pixel-Decimation Algorithm for Block-Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 6, pp.1006-1011, September 2000.

[92] ITU-T recommendation H.263 software implementation, Digital Video Coding Group, Telenor R&D, 1995.

[93] S. Eckart and C. Fogg, ISO/IEC MPEG-2 software video codec, Proceedings SPIE 2419, pp. 100-118, 1995.

[94] Chok-Kwan Cheung and Lai-Man Po, “A Hierarchical Block Motion Estimation Algorithm Using Partial Distortion Measure”, Image Processing, 1997. Proceedings., International Conference on, Vol. 3, pp. 606-609, 1997.

[95] 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, Vol. 10, No. 3, pp. 417-422, April 2000.

[96] Jong-Nam Kim and Tae-Sun Choi, “A Fast Full-Search Motion Estimation Algorithm Using Representative Pixels and Adaptive Matching Scan”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 7, pp. 1040-1048, October 2000.

[97] Yui-Lam Chan, Wan-Chi Siu and Ko-Cheung Hui, “Block Motion Estimation Using Adaptive Partial Distortion Search”, Proceedings, the 2002 IEEE International Conference on Multimedia and Expo, ICME 2002, Vol. I, pp. 477-480, August 2002.

[98] Yong-Sheng Chen, Yi-Ping Hung and Chiou-Shann Fuh, “Fast Block Matching Algorithm Based on the Winner-Update Strategy”, IEEE Transactions on Image Processing, Vol. 10, No. 8, pp. 1212-1222, August 2001.

[99] W. Li and E. Salari, “Successive Elimination Algorithm for Motion Estimation”, IEEE Transactions on Image Processing, Vol. 4, No. 1, pp.105-107, January 1995.

[100] Jing-Yi Lu, Kuang-Shyr Wu and Ja-Chen Lin “Fast Full Search in Motion Estimation by hierarchical use of Minkowski’s inequality”, Pattern Recognition, Vol. 31, No. 7, pp. 945-952, 1998.

[101] X. Q. Gao, C. J. Duanmu, C. R. Zou and Z. Y. He, "Multi-level successive elimination algorithm for motion estimation in video coding", Proceedings, 1999 IEEE International Symposium on Circuits and Systems, ISCAS '99, Vol. 4, pp.227 -230, 1999.

[102] Ken Sauer and Brain Schwartz, “Efficient block motion estimation using integral projections”, IEEE Transactions on Image Processing, Vol. 6, No. 5, pp. 513-518, October 1996.

[103] Yih-Chuan Lin and Shen-Chuan Tai, “Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression”, IEEE Transactions on Communications, Vol. 45, No. 5, pp. 527-531, May 1997.

[104] Yui-Lam Chan and Wan-Chi Siu, “Edge Oriented Block Motion Estimation for Video Coding”, IEE Proceedings - Vision, Image and Signal Processing, Vol. 144, No. 3, pp.136-144, June 1997.

[105] Yui-Lam Chan and Wan-Chi Siu, “An Efficient Search Strategy for Block Motion Estimation Using Image Features”, IEEE Transactions on Image Processing, Vol. 10, No. 8, pp. 1223-1238, August 2001.

[106] Bo Tao and Orchard M.T., "Gradient-based residual variance modeling and its applications to motion-compensated video coding", IEEE Transactions on Image Processing, Vol. 10, No. 1, pp. 24 -35, January 2001.

[107] Fabio Cavalli, Rita Cucchiara, Massimo Piccardi and Andea Prati, “Performance Analysis of MPEG-4 Decoder and Encoder”, IEEE Region 8 International Symposium on Video / Image Processing and Multimedia Communications, pp. 227-231, June 2002.

[108] Krit Panusopane and Xuemin Chen, “A fast motion estimation method for MPEG-4 arbitrarily shaped objects”, Proceedings, 2000 International Conference on Image Processing, ICIP 2000, Vol. 3, pp. 624-627, September 2000.

[109] Thomas Wiegand, Xiaozheng Zhang and Bernd Girod, “Long-Term Memory Motion-Compensated Prediction”, IEEE Transactions On Circuits And Systems For Video Technology, Vol. 9, No. 1, pp. 70-84, February 1999.

[110] Thomas Wiegand, Bo Lincoln and Bernd Girod, “Fast Search for Long-Term Memory Motion-Compensated Prediction”, Proceedings, International Conference on Image Processing 1998, Vol. 3, pp. 619-622,1998.

[111] Thomas Wiegand, Eckehard Steinbach and Bern Girod, “Long-Term Memory Prediction Using Affine Motion Compensation”, Proceedings, International Conference on Image Processing 1999, Vol. 1, pp.51-55, 1999.

[112] R. J. Clarke, “Transform Coding of Images”, Academic Press, 1985.

[113] K. R. Rao and P. Yop, “Discrete Cosine Transform Algorithms, Advantages, Applications”, Academic Press, 1990.

[114] Masahide Kaneko, Yoshinori Hatori and Atsushi Koike, “Improvements of Transform Coding Algorithm for Motion-Compensated Interframe Prediction Errors-DCVT/SQ Coding”, IEEE Journal on Selected Areas in Communications, Vol. SAC-5, No. 7, pp. 1068-1078, August 1987.

[115] Chi-Fa Chen and K. K. Pang, “The Optimal Transform of Motion-Compensated Frame Difference Images in a Hybrid Coder”, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 40, No. 6, pp. 393-397, June 1993.

[116] Wolfgang Niehsen and Michael Brünig, “Covariance Analysis of Motion-Compensated Frame Difference”, IEEE Transactions on Circuits and Systems For Video Technology, Vol. 9, No. 4, pp. 536-539, June 1999.

[117] S.M.M. de Faria and M. Ghanbari, "Low bit-rate video coding with spatio-temporal geometric transforms ", IEE Proceedings-Vision, Image and Signal Processing, Vol. 143, No. 3, pp. 164-170, June 1996.

[118] Seung Chul Yoon, Krishna Ratakonda and Narendra Ahuja, “Low bit-rate video coding with implicit multiscale segmentation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 7, pp. 1115-1129, Oct. 1999.

[119] Bo Tao and Michael T. Orchard, “Prediction of second-order statistics in motion-compensated video coding”, Proceedings, 1998 International Conference on Image Processing, ICIP 98, Vol. 3, pp. 910-914, October 1998.

[120] ISO/IEC JTC1/SC29/WG11 5477, “MPEG-4 Video Verification Model version 14.0”, December 1999.

[121] ISO/IEC JTC1/SC29/WG11 N3675, “Draft of MPEG-4 Optimization Model Version 2.0”, October 2000.

[122] ISO/IEC TR 14496-7, Information technology -- Coding of audio-visual objects -- Part 7: Optimized reference software for coding of audio-visual objects.

[123] R. J. Stevens, A. F. Lehar and F. H. Preston, “Manipulation and Presentation of Multidimensional Image Data Using the Peano Scan”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 5, pp. 520-526, September 1983.

[124] Zixiang Xiong, Kannan Ramchandran, Michael T. Orchard and Ya-Qin Zhang, “A Comparative Study of DCT and Wavelet Based Coding”, Proceedings, 1998 IEEE International Symposium on Circuits and Systems, ISCAS '98., Vol. 4, pp. 273-276 , 31 May-3 June 1998.

[125] Chi-Fa Chen and K. K. Pang, “Hybrid Coders with Motion Compensation”, Multidimensional Systems and Signal Processing, Vol. 3, No. 3, pp. 241-266, 1992.

[126] Yung-Lyul Lee, Hyun Wook Park, “Loop filtering and post-filtering for low-bit-rates moving picture coding”, Signal Processing: Image Communication 16, pp. 871-890, 2001.

[127] Peter Kuhn, “Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation”, Kluwer Academic Publishers, London, 1999.

[128] M. I. Sezan and R L. Lagendijk. Motion analysis and image sequence processing, Kluwer Academic Publishers, London, 1990.

[129] Peter Kuhn, “Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation”, Kluwer Academic Publishers, 1999.

[130] Hosam M. Mahmoud, “Sorting: A Distribution Theory”, John Wiley & Sons, 2000.

[131] http://www.iso.ch/iso/en/ittf/PubliclyAvailableStandards/14496-5_Compressed_directories/

[132] http://megaera.ee.nctu.edu.tw/mpeg/