THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學
Pao Yue-kong Library
包玉剛圖書館

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

Pao Yue-kong Library, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

http://www.lib.polyu.edu.hk

The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

Block-Based Motion Estimation Algorithms for Video Coding Applications

by
Kam Yan Ho, BEng(Hons)

A thesis submitted in partial fulfillment of the requirements for the Degree of Master of Philosophy

November 2006

Pao Yue-kong Library
PolyU • Hong Kong

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)

Kam Yan Ho (Name of student)

Abstract

Block-based motion estimation (ME) is one of the most important parts of video coding process. Over recent years, variable block size motion estimation (VBS-ME) and multiple reference frames motion estimation (MRF-ME) techniques have been included as parts of the H.264/AVC standard to improve the performance of ME. Although they can produce higher video quality as compared with the traditional fixed block size and single reference frame ME, applying them will suffer from heavy computational load so it is necessary to look for fast implementation approaches. There are already many fast algorithms of VBS-ME and MRF-ME developed, but nearly all of them are lossy methods besides the SAD reuse algorithm which is a lossless method of VBS-ME. It is always desirable that some fast algorithms can be developed to give the optimal searching results. To achieve this purpose, we propose respectively a fast approach of VBS-ME and a fast approach of MRF-ME, and both of them can provide exactly the same video quality as compared with the original exhaustive approach.

The proposed VBS-ME makes use of the row-based partial distortion search (PDS) with special scanning orders of the rows of pixels inside the blocks for different block sizes. The partially computed sums of absolute differences (PSAD) of the searched candidate blocks are reused to pre-compute the sum-of-absolute-difference (SAD) values for the unsearched candidates with other block sizes. Therefore, the number of operations can be reduced. In addition, it is found that the locations in the searching region with similar extent of pixel error usually appear in clusters. Since the early rejection of non-best matches in the PDS becomes faster if the pixel errors with larger values are considered in the highest priority for the calculation of distortion values, the special scanning orders are arranged in a dispersed way so that even though

a block matching process starts at a low error region, it will not keep processing at that region and will jump to other locations.

Our proposed MRF-ME is different from the original approach which will find the best match in each reference frame firstly, and then select the optimal one among those matches. Instead, the location of the predicted block will be found within just one stage. This is achieved by allowing the comparisons between the costs of candidate blocks in one reference frame with those in other reference frames while searching for the best candidate block. By using this method, the tentative minimum value of the costs of searched candidate blocks will be stored and will keep on being updated when the searching process scans through all the candidates in different reference frames. Therefore, once the best match in a certain reference frame which has the global minimum cost among all the candidates in different frames has scanned, all remaining unsearched candidates including those in other reference frames can be rejected without fully computing their costs.

To perform a full ME, usually, the motion search is bounded within a search window. Square window is a traditional type of search window and is popularly used. However, it was found after conducting our investigation that nearly all of the motions in video sequences are contained within a star-shaped range but not a square range. This means that it is inefficient to search for motions from positions near the four corners of the square window. We propose a star-shaped search window with about ¼ of the size of a square window. Using the star-shaped window can largely reduce the number of operations, and also the video quality will not be downgraded. It is because those motion vectors corresponding to the positions near the corners of a square window are too far away from the center, and coding them requires a large amount of bits. That is why excluding those motion vectors by using the star-shaped window instead can still

maintain the same level of video quality. On the other hand, when the PDS is applied, usually a spiral search pattern is used, but it is also inadequate according to the star-shaped motion distribution. A star-shaped search pattern is proposed that allows searching the candidate positions along a star-shaped path from the center of the search window to positions far away. According to experimental results, the average coding speed can be increased by using the star-shaped search pattern.

List of Publications

- Published papers

1. Yan-Ho Kam and Wan-Chi Siu, “A Fast Full Search Scheme for Rate-Distortion Optimization of Variable Block Size and Multi-frame Motion Estimation,” IEEE Proc. on Int. Midwest Sym. on Circuits and Systems (MWSCAS), paper 3095, pp. 1-4, Aug. 2006.
2. Yan-Ho Kam and Wan-Chi Siu, “Effective Star-Shaped Search Window and Star-Shaped Pattern for Motion Estimation”, IEEE Proc. on Int. Conf. on Image Tech. and Applications (ICITA), pp. 290-293, Jan. 2007.

Acknowledgements

I am glad to have this opportunity to express my honest gratitude to different people whom I conduct my research work with. First of all, I would like to thank Prof. W.C. Siu, my supervisor, who is the most important person for me to complete my research. His professional and capacious knowledge, abundance of research experiences as well as his patient guidance immeasurably helped me throughout my whole research period. With his constant and kindly support, this study could finally reach the present level.

Sincere thanks are given to my colleagues, Mr. W.H. Wong, Mr. W.L. Hui, Dr. K.T. Fung and Mr. K.Y. Wong. The sharing of their useful advices and programming knowledge greatly contributed to make every success of my work. Their friendly encouragements also accompanied me to walk through much hard time in my study. All of them, actually, made this two years one of the most wonderful time of my life.

I gratefully acknowledge the financial support of the Hong Kong Polytechnic University through the Research Grants Council of the Hong Kong Special Administrative Region which granted me a studentship to make my research work possible.

Last but not least, I am deeply thankful to my family and friends for their love and support. Lacking their understanding, 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 fast method for full variable block size motion estimation is introduced. This algorithm was evolved from the PDS and SAD reuse algorithm. It applies the PDS to obtain the partial SAD values for blocks of one size, and then operates like the SAD reuse algorithm to makes use of those partial SAD values of previously processed blocks to construct the partial SAD values of other blocks with different sizes. As a result, many repeated operations can be saved. It is found that a dispersed processing order of pixels is more favorable for performing the PDS in this algorithm than the traditional top-down processing order during the block matching process because using the dispersed processing order will prevent the process from being trapped into regions of low distortion errors. Experimental results show that the new method is faster than the exhaustive full search (FS) algorithm by about 10 times on average. More details can be found in Section 3.2.1.
2. A new multiple reference frames motion estimation algorithm is proposed to reduce the number of redundant operations. In the PDS of this algorithm, the candidate blocks in different reference frames are allowed to compare with each other. The best candidate block with the known minimum cost among all searched candidates in different reference frames is recorded, and the unsearched candidates in any frame will be rejected immediately once their partial costs are found to be larger than or equal to the cost of the recorded tentatively best candidate. It is different from applying the PDS in MRF-ME traditionally that the candidates in each reference frame can only compare their costs with each other. As a result, some candidates can be rejected sooner, and hence some operations can be skipped.

According to experimental results, it can averagely speed up the multi-frame motion estimation process by about 7 times as compared with the exhaustive FS without any quality degradation. More details can be found in Section 3.2.2.

3. The algorithms mentioned in points 1 and 2 are compatible with each other so that a powerful variable block size and multi-frame ME method can be produced. On average, this ME scheme is about 9.7 times faster than the exhaustive FS approach meanwhile the output video quality is the same. More details can be found in Section 3.3.
4. It is found that the average motion activity of real video objects shows itself a star-shaped distribution. It is different from past study in many literatures that the motions of video objects tend to be in horizontal more than in vertical, and turn out a rectangular distribution. More details can be found in Section 4.2.
5. Based on the star-shaped distribution of motion activity, a star-shaped search window is introduced to replace the traditional square window. The new search window has only about ¼ of the size of a square window but it can cover almost all motion activities contained in the square window. Although some motions do exist in the region not covered by the star-shaped window but covered by the square window, since it is found that the motions in those regions require too many bits to encode, giving up those motions does not affect the quality of the coded sequences. Experimental results have proved that performing full ME with the proposed search window can be about 3.1 times faster both in terms of actual time of ME and the total number of operations on average when compared with using a square window. This new search window can corporate with other full or lossy search schemes to further speed up the ME process. More details can be found in Section 4.3.1.

6. A star-shaped search pattern is proposed which follows also the star-shaped distribution of motion activity. It lets the motion search carried out from the most probable reference position to the least probable position in order to reject the non-best matches earlier. If the star-shaped search window is used together with this new search pattern, the ME process can be further speeded up. More details can be found in Section 4.3.2.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 *Introduction to video compression*

Digital videos have being widely used for various purposes all over the world for many years. Each video sequence is composed of a large number of still pictures which are displayed one by one when the video is being played with a specific frame rate, so the size of each raw video is very large. As the demands of higher visual quality and faster transmission speed of video data become larger and larger, it is impossible to directly store up and make use of raw digital videos without video compression. With video compression or video coding, the redundancies within video sequences can be eliminated. As a result, not only the video size can be diminished so as to let more video data be stored in limited memory size and hence increase the video quality with the same size, the transmission speed of video signal can also be raised so that real-time applications such as video conferencing and broadcasting become smooth. Since an efficient video coding scheme can largely increase the performance of coded videos, a number of researches on video coding techniques have being conducted. For years, many video coding techniques were successfully developed and adopted. Video coding standards are used to standardize the use of video coding techniques in the world. MPEG-1 video [1], MPEG-2 video [2], H.263 [3, 4], MPEG-4 visual [5] and H.264/AVC [6] are some well-known international video standards.

1.2 *History of video coding standards*

The first digital video coding standard for video conferencing was developed by the International Telegraph and Telephone Consultative Committee (CCITT) in 1980s. In early 1991, CCITT finalized a set of coding standards called H.320. The video

coding part is known as H.261 [7] which targets to code video frames in a common intermediate format (CIF). Quarter CIF (QCIF) with lower resolution is only supported for videophone or videoconference on the integrated services digital network (ISDN). This is the first standard using a basic video coding structure which is still being used in recent applications. H.261 adopts the 16x16 MB size for motion estimation & compensation, 8x8 block-based discrete cosine transform (DCT), scalar quantization and 2-D run-level variable length coding (VLC).

In 1988, a working group named as the Moving Picture Expects Group (MPEG) was established under the auspices of International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) to be in charge of the development of standards of digital audio and video coding. In the same year, MPEG started the work of moving picture standardization with a strong focus on real-time decoding the data stored in digital storage media. Their first product is MPEG-1, and its part two is the well-known MPEG-1 video which was finally approved in 1993. MPEG-1 is officially entitled “Information technology – coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s”. As stated in the title, MPEG-1 is responsible for coding digital audio and video scripts. Since the targeted bit-rate is about 1.5 Mbit/s, it is particularly suitable for applications involving storage media such as CD-ROM data retrieval. MPEG-1 video supports bidirectional predictive frames and half-pixel motion estimation and compensation in addition to the coding techniques of H.261. MPEG-1 outperforms H.261 at a bit-rate of 1-2 Mbit/s, but it does not perform well for bit-rate out of this range.

When the development of MPEG-1 was nearly complete, the MPEG committee started the development of MPEG-2 targeting to have a better performance at higher bit-rate of around 4-30 Mbit/s and with more kinds of applications such as television

broadcasting, high-definition TV, broadcast satellite service to home, cable TV distribution, interactive storage media, interpersonal communications, etc in 1992. MPEG-2 video part was developed from a joint project of ISO/IEC and ITU-T, and was finalized in 1995. It was designed to encompass MPEG-1 and support interlaced videos which are not supported by MPEG-1. It also supports hierarchical scalability including spatial scalability, temporal scalability, SNR scalability and data partitioning.

In 1993, CCITT formed the ITU Telecommunications Standardization Sector (ITU-T) to standardize the global telecommunications. In early 1996, a new standard called H.263 worked out by ITU-T was approved. H.263 is the first standard designed specially for very low bit-rate videos, and is widely used for video communication nowadays. The targeted bit-rate of H.263 is about 10-2048 Kbit/s. It has several new technical features including 8x8 block size ME, overlapped block motion compensation, allowing motion vectors pointing outside video frames, 3-D run-level-last VLC, median motion vector predictor, etc. As compared with H.261, H.263 includes also the arithmetic coding, half-pixel ME and bidirectional prediction. At a bit-rate below 30 Kbit/s, H.263 can provide the same visual quality as H.261 but only half or less than half of the bit-rate is required. If the bit-rate is above 80 Kbit/s, the performance of H.263 is much better than H.261.

The second version of H.263, H.263+, was approved in early 1998. This new version contains a number of new optional features. H.263+ is targeted to outperform all existing standards at any bit-rate and frame rate, and supports both the progressive and interlaced video formats and scalability. It is also the first standard that supports telecommunication in error prone environment by offering a high degree of error resilience techniques.

MPEG-4 is the next standard developed by MPEG after finishing the MPEG-1 and MPEG-2. The standardization was started in 1994, and was ended in October 1998. It finally became an international standard in early 1999. Its second version with fully backward compatibility was then finalized in early 2000. The target of MPEG-4 is to code scenes of a number of individual Audio-Video-Object, and is ultimately to fulfill the future trend of interactive multimedia applications. As a result, a set of tools and algorithms for coding audio-visual data were developed so that users can access and manipulate arbitrarily shaped objects in video scenes in the client side. Part two of the standard, MPEG-4 visual, includes many technical features which prior video or image coding standards do not have. They are shape coding of segmented objects, context-based arithmetic coding, shape-adaptive DCT, padding techniques for arbitrarily shaped coding of texture, wavelet coding of image, global motion compensation, reversible VLC for error prone environment, quarter-pixel motion estimation and compensation using Wiener filter, etc. MPEG-4 was designed for any bit-rate, picture format, frame rate and both progressive and interlaced videos.

H.264 advanced video coding standard [6] is the most nearly approved standard proposed by the joint video team (JVT) of the video coding experts group (VCEG) and the MPEG in March 2003. This standard employs and generalizes lots of most effective video coding techniques such as variable block size ME, multiple reference frames ME and directional intra-frame prediction in order to pursue a very high compression ratio and video quality. The details are discussed in Section 2.3.

1.3 *Literature review*

Motion estimation and compensation is a key technique to acquire such a high effectiveness of different video coding standards. The book written by Siu [8] provides

a broad discussion on the techniques of ME. Jain and Jain [9] presented a hybrid coding scheme of image sequence that involves block-based ME and compensation with a fixed block size. The block-based approach is good at predicting translational motions, but it suffers the problem of inaccurate matching for complex motion activities within blocks. Using variable block size such that complex motion activities can be predicted by blocks with different sizes is a normal solution. Chan, Yu and Constantinides [10] used quadtree structure to construct a variable block size model. Besides using square and rectangular blocks, Mahmoud and Bayoumi [11] suggested to partition a frame into equilateral triangle blocks.

The simplest block matching algorithm of ME is full search algorithm (FSA). The FSA exhaustively computes and checks a matching criterion [12-17] for each matching candidate block within a search window, and then gives the optimal solution. However, its heavy computational load is a serious disadvantage. A number of researchers developed various fast approaches to improve the ME. These fast algorithms can be classified into four categories as follows:

1) Techniques based on a reduced set of motion vector candidates

Instead of searching over all possible blocks within the search window, algorithms [9, 15, 18-31] belonging to this category restrict the search over a subset of candidate blocks such that the number of operations can be largely reduced. Most algorithms in this category are based on the uni-modal error surface assumption. Therefore the search starts by evaluating the distortion measure at locations coarsely spread over the search window according to some predefined uniform pattern. Then it is repeated with finer resolution around the search location with the minimum distortion value found from preceding steps. Many famous algorithms are in this category like the 2-D logarithmic search proposed by Jain and Jain [9], the n-step hierarchical search

proposed by Koga et al. [18], the genetic search algorithm proposed by Chow and Liou [19], the diamond search proposed by Tham, Ranganath and Kassim [20], the four-step search proposed by Po and Ma [21], and the motion vector field adaptive fast motion estimation proposed by Hosur and Ma [22], etc.

2) Techniques based on a reduced complexity distortion measure

Algorithms [18, 32-41] in the second category use a reduced complexity distortion measure. Pixel decimation techniques [18, 32-34] including an adaptive pixel decimation method proposed by Chan and Siu [33] making use of the edge features of image blocks are in this category. Other representative algorithms include the partial distortion search (PDS) [35] adopted by most video coding standards, the adaptive PDS proposed by Chan, Siu and Hui [37], the adaptive PDS using clustered pixel matching error proposed by Hui, Siu and Chan [38], and the winner-update strategy proposed by Chen, Hung and Fuh [41].

3) Techniques based on mathematical inequalities

The third category [42-45] applies some mathematical inequalities. Successive elimination algorithm (SEA) proposed by Li and Salari [42] is the most representative one in this group. Another famous algorithm is the multilevel SEA (MSEA) proposed by Gao, Duanmu, Zou and He [43] which was evolved from SEA to reach a much better performance.

4) Techniques based on image features

In the fourth category [46-48], different image features such as edge and gradient information are used to help finding the best matching blocks. The edge oriented search strategy proposed by Chan and Siu [46] surprisingly gives a motion compensated video frame visually even better than that of the full search algorithm.

Another search strategy proposed by Chan and Siu [47] makes use of the gradient of the error-surface to effectively avoid trapping in local minima.

Sullivan [49] proposed a multi-hypothesis motion compensating prediction for hybrid video coders. As long as the predicted pixel values are generated from more than one source, the approaches are generally defined as multi-hypothesis. The bidirectional prediction and overlapped block motion compensation (OBMC) proposed by Nogaki and Ohta [50] to remove blocking artifacts of motion prediction error are some kinds of multi-hypothesis technique.

The emerging H.264/AVC video coding standard [6] allows ME to be performed with variable block size and multiple reference frames instead of traditional fixed block size and single reference frame, which stimulates researchers to broaden their research directions about ME. In H.264, the size of block is ranging from 16x16 to 4x4 such that complex motions within MBs can be captured by using small blocks. On the other hand, because of various factors, image contents inside video frames will move and change in many ways in addition to translational motions. The decoded blocks in the reference frame just before the current frame may not resemble the blocks in the current frame, but better matches may exist in earlier decoded frames. These better matching blocks can be found out by considering some more reference frames. Although using more block sizes and reference frames can result in better video quality, the computational load is also largely increased. The FSA of multi-block size and multi-frame ME involves conducting individual motion search for the blocks with different sizes and in each reference frame to estimate the motions within each MB so the number of operations used in this method is extremely large. Consequently, fast algorithms that make use of the specialties of variable block size and multiple reference frames are demanded for speeding up the process. As there are strong correlations about

image contents and motion activities among the blocks of various sizes within the same MB and among consecutive reference frames, fast algorithms can be developed to exploit these spatial and temporal redundancies.

Fast algorithms of variable block size ME based on the correlation between different block sizes [51-54] include the SAD reuse algorithm embedded in the H.264 reference software [51] which recycles the obtained SAD values of some block sizes to evaluate the SAD values of other block sizes, and the merge and split method proposed by Zhou, Sun and Hsu [52] which uses the obtained MVs of some block sizes to predict the motions of other block sizes. Another type of algorithms of multi- block size ME analyzes the texture of each MB to decide how to partition the block. In these algorithms, the information about edges, smoothness, etc will be considered. The methods proposed by Shen, Zhang, Huang and Li [55] and by Wu, Wu, Lim, Pan, Li and Lin [56] are of this type.

As for the fast algorithms of multiple reference frame ME [57-60], the first type is to reuse the obtained MVs to help predicting the motions of image blocks referring to different reference frames. The short-term and long-term MV reusing methods proposed by Chen, Chiang, Li and Chi [57] and by Hsiao, Lee and Chang [58] belong to the first type. Furthermore, Chang, Au and Yeung [59] suggested that some reference frames can be disabled before searching them by investigating the “sub-pixel locations” of the objects in the current frame and each reference frame. When a MB is going to be encoded, only the reference frames in which the blocks at the same position have the same “sub-pixel location” of the current MB will be searched. This means that the similarity between the current frame and each reference frame is roughly checked to filter out some improbable frames. In addition, Ting, Po and Cheung [60] suggested applying different search patterns and search ranges to different reference frames

according to different situations. The performance of coding videos with variable block size and multiple reference frames ME can be calibrated and optimized by rate-distortion optimization technique [17, 61].

Besides ME, entropy coding is another part of a whole video coding system which can provide compression of video data. Two basic types of entropy coding are variable length coding (VLC) and arithmetic coding. VLC was adopted in MPEG-2 and MPEG-4 to use different coding tables for inter- and intra-predicted blocks. For the H.263, optional intra coding mode and alterative inter VLC mode were added to the VLC coder to provide some adaptation. However, all these methods use a single VLC table to code the transformed blocks, and do not consider the context relationship between transform coefficients. Since the magnitudes of coefficients in low frequency subbands are usually bigger than those in high frequency subbands, the new H.264/AVC standard adopted the context-based VLC (CAVLC) [62] which adaptively selects from multiple VLC tables to reduce the inter-coefficients redundancy and hence raise the performance. Arithmetic coding [63] was firstly implemented in H.263 which applies multiple probability distribution models adaptively. Subsequently, a new and sophisticated arithmetic coding technique called “context-based adaptive binary arithmetic coding” (CABAC) [64] was developed and adopted in the H.264/AVC standard. It combines an adaptive binary arithmetic coder with many context models to adapt to different statistical behaviors, and can result in significant improvement in coding performance.

1.4 *Organization of the thesis*

The rest of the thesis is organized as follows:

Chapter 2 is a review of basic video coding techniques, which includes hybrid video coding modeling, theory and algorithms of block-based motion estimation and compensation, and the up-to-date H.264 advanced video coding standard and its modern coding techniques.

Chapter 3 presents a scheme of variable block size and multiple reference frames motion estimation which aims at achieving lossless and rate-distortion optimized results with smaller operational complexity and also a faster speed. There are two new methods in the scheme which are “partial SAD reusing PDS” (PSADR-PDS) and “common tentative minimum PDS” (CTM-PDS). The experimental results attached have proved that our proposed scheme is better than other existing approaches of multi-block and multi-frame motion estimation which are claimed to be fastest.

In Chapter 4, the observation on motion activity in natural video sequences is discussed. The star-shaped search window and star-shaped search pattern which are developed basing on our observation are also depicted.

Conclusion of the research is given in Chapter 5 in which some suggestions for further development is also included.

Chapter 2. Technical Review

2.1 *Hybrid video coding model*

There are many video coding models used to code and decode video sequences. The most common video coding model is the hybrid model which was adopted in many modern video coding standards including H.263 [3, 4], MPEG-1 [1], MPEG-2 [2], MPEG-4 [5] and H.264 [6]. A simple structure of a hybrid video coding model is shown in Figure 2-1.

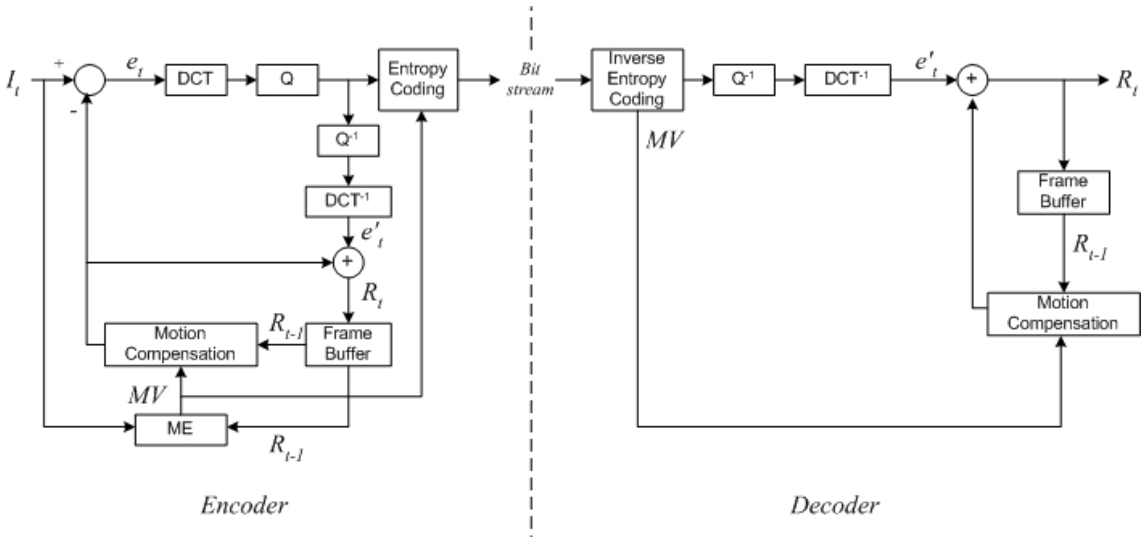


Figure 2-1. Hybrid video coding model

The model consists of an encoder and a decoder. To encode a video sequence, the frames of the video will firstly be classified as various types. There are basically two classes of video frames, and they are intra-frame (I-frame) and inter-frame. As stated in its name, each I-frame is coded by using the information within the frame itself without using other frames. I-frame coding makes use of the coding techniques for still images like JPEG. Therefore, I-frames should have very high quality but they require a large number of bits to represent them so the amount of I-frames of an encoded video

sequence should be small. Another class of frame type is the inter-frame, which is also called predictive frame. It is called predictive since the coding of inter-frames is accomplished by predicting the contents from previously coded frames to exploit the temporal redundancies between frames. As the contents of any two neighboring frames should not differ too much from each other, there exists many repeated data inside neighboring frames. Then it is highly possible for a frame to find out its contents from other coded frames and copy those coded data such that the coding of the actual contents of the current frame can be skipped. Instead, only the residual signal which is the difference between the predicted contents and the original contents will be coded, hence much fewer bits will be required. This is the major concept of motion estimation (ME). The most frequently used type of inter-frame is the predictive frame (P-frame) in which each pixel is predicted from a coded I-frame or a coded P-frame. Another type of inter-frame is the bidirectional predictive frame (B-frame). Its contents are predicted from a coded previous frame and a coded future frame. No B-frame will be used as reference frame of other frames. Figure 2-2 below shows an example depicting clearly the reference frame of each inter-frame. The frames are arranged in display order in the figure. Since B-frames require referring to future frames, the coding order must not be the same as the display order as shown, and in this example, a possible coding order is 1→4→2→3→7→5→6→10→8→9.

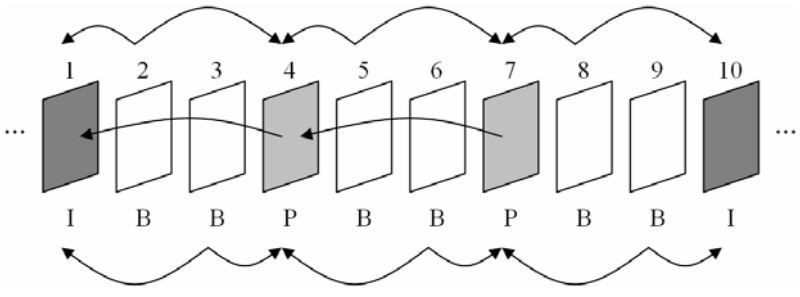


Figure 2-2. Reference frames of each inter-frame

Besides the classification of video frames, normally, each video frame will be divided into a number of blocks with equal size, and then the coding will be applied to each image block one by one. In Figure 2-1, the input block is indicated as I_t, and the reconstructed block from the decoder is indicated as R_t.

The coding of an image block of an I-frame can be accomplished by following the horizontal path at the top of the block diagram. The first step of the coding process is transform coding. It is an image conversion process that transforms the input block from spatial domain to frequency domain. Precision error will be introduced after this step. The most popular transform coding method is the Discrete Cosine Transform (DCT). The DCT has high energy compaction power and data decorrelation power so that the energy of an image block can be compacted into relatively few coefficients. Although DCT does not achieve any compression, the output transform coefficients are more suitable for compression by using run-length encoding.

The next step after DCT is Quantization. The transform coefficients will be divided by different quantization factors. Usually the quantization factors for higher frequency coefficients are larger, and hence the high frequency signals will be depressed while the low frequency signals will be remained. Since human eyes are more sensitive to low frequency signals, and high frequency signals are relatively not important, compression can be acquired by cutting some of those unimportant signals. Quantization error as well as precision error will be introduced after executing quantization.

The quantized coefficients will then be converted into a string of bits by entropy coding. Entropy coding is a lossless process for compression. There are many methods to conduct entropy coding such as fixed length codeword assignment, arithmetic coding

[63] and run length coding. Fixed length codeword assignment is the simplest method that assigns different quantized coefficients with the largest number of bits. Therefore, it can only produce the worst compressed results. Differently, arithmetic coding and run length coding use variable length codeword according to the knowledge of the probabilities of all possible events. Short codeword is assigned to event with high probability of occurrence, thus a higher compression ratio can be obtained. It is important that the codebooks in the encoder and in the decoder should be the same, otherwise data mismatch will appear. During the entropy coding process, the image data will continuously outputted as bit strings. These bit strings can be transmitted via channels to decoders, or be stored up as compressed data.

In the decoder, the received video signals are used to reconstruct the original video sequences. It can be realized by carrying out the inverse processes of entropy coding, quantization and transform coding.

The procedure to code an image block of a P-frame is depicted by the whole diagram in Figure 2-1. Firstly, motion estimation (ME) is performed to obtain a prediction of the image block from a previously coded frame stored in the frame buffer. The difference in location between the current block and its prediction is represented by a motion vector (MV). Then a residual block, e_t, is determined, which is composed of the absolute values of the pixel differences between the current image block and the predicted block. Instead of encoding the pixels of the image block just like encoding an intra-block, the residual error between the image block and its prediction is encoded through DCT, quantization and entropy coders. The information of the vertical and horizontal components of the MV is also coded by entropy coding, and transmitted to the decoder. Since the frame buffer in the decoder also stores the previously coded frame, once a MV is received by the decoder, the predicted block can be copied from

the frame buffer, and used as part of the reconstructed block. To compensate the distortion between the copied block and the original image block, the encoded residual signal has to be decoded, and subsequently be added to the copied block. A copy of the reconstructed block should be stored in the frame buffer for the decoding process of the next video frame. Likewise, the frame buffer in the encoder should also keep the reconstructed block to make ME possible for the next frame. That is why there is a decoder embedded inside the encoder.

2.2 *Block matching algorithm of motion estimation*

There are a lot of methods [9, 15, 18-48] to implement the ME from the exploitation of temporal redundancies. The block matching algorithm (BMA) is a widely adopted method by modern video coding standards such as the H.263, MPEG-1, MPEG-2, MPEG-4 and H.264. In the BMA, each video frame will be broken down into non-overlapping marcoblocks (MB) with a size of 16x16. Not only for the block matching process, each MB will be coded individually also for the transform coding and entropy coding process. Then the motion of each MB will be estimated by searching a region of pre-determined size called the search window within the reference frame. This is to look for the most similar 16x16 block that needs the minimum number of bits to code the residual signal. Commonly, the search window is a square-shaped region containing candidate blocks. The searching process is based on matching the candidate blocks in the reference frame with the MB in the current frame. To perform BMA, a number of different methods are available in the literatures. The full search algorithm (FSA) is the simplest approach which can give the optimal solution by exhaustively examining all possible candidate blocks. However, its heavy computation load is really

a serious problem in real-time applications. In order to resolve the difficulties, many fast approaches were developed.

Based on the matching criteria, search schemes and the search area, a number of researchers investigated and developed a large number of fast algorithms of block-based ME [9, 15, 18-48]. Generally, there are four categories of these fast search algorithms. The algorithms in the first category [9, 15, 18-31] reduce the number of search points inside the search window to reduce the computations. The main challenge of these algorithms is that the search can be easily trapped into local minima. A possible solution is to refer to the motion field while deciding the set of search points. Another way to reduce the probability of being trapped into local minima is the hierarchical or multi-resolution techniques [65, 66]. Algorithms in the second category [18, 32-41] reduce the complexity of computing the distortion values. The pixel decimation techniques [18, 32-34] sub-sample the current MB and the candidate blocks while block matching between two blocks is proceeding so that fewer operations of computing the SAD values are required. The PDS techniques [35-40] terminate the computation of SAD values intermediately in case the partial SAD values are found to be greater than or equal to the already found best SAD value. In the third category, the algorithms [42-45] reduce the computational effort by applying some mathematical inequalities. The representative of this group is the successive elimination algorithm (SEA) [42]. It makes use of the Minkowski's inequality to filter out some impossible candidate blocks without calculating the SAD values. The fourth category [46-48] makes use of the edge information inside each block to assist finding the best matching block since object edge is a prominent feature in image processing.

There is another classification of fast ME algorithms that they will either be classified as lossless approaches [35-43, 45] or lossy approaches [9, 15, 18-34, 44, 46-

48]. The lossless algorithms can result in an optimal set of MVs with the best peak signal-to-noise ratio (PSNR) and output bit-rate, which is just the same as the FSA since all candidate blocks are still considered. As for the lossy algorithms, many of them can be very fast, and may be able to reach tens times of encoding speed of a lossless algorithm. However, the resultant video quality will be dropped as a trade-off result. Some of this type of algorithms skips considering some candidate blocks, while some involves approximation in the calculation of the matching criteria. In the following sub-sections, some typical algorithms of block-based ME will be reviewed in detail. Section 2.2.1 will introduce the full search algorithm. Equations and figures are provided. Section 2.2.2 will focus on the lossless approaches of ME. Section 2.2.3 will concentrate on some lossy algorithms.

2.2.1 Full search algorithm (FSA)

In FSA, the distortion between the target MB in the current frame and each of its candidate blocks in the reference frame will be computed. There are many choices of distortion measure. Two most common distortion measures are the sum of absolute differences (SAD) and the sum of squared differences (SSD). Let $c(.,.)$ be a pixel value of the current frame, $r(.,.)$ be a pixel value of the reference frame, (i,j) be the location of the top-left pixel of the current MB, (a,b) be the position of a pixel within an image block, and (x,y) be a candidate MV. The formulation of the SSD between the current MB and a reference block is:

$$SSD(x,y)=\sum_{b=0}^{15}\sum_{a=0}^{15}\left[c(i+a,\,j+b)-r(i+x+a,\,j+y+b)\right]^2 \qquad \textbf{(2-1)}$$

The formulation of the SAD between the current MB and a reference block is:

$$SAD(x,y)=\sum_{b=0}^{15}\sum_{a=0}^{15}\left|c(i+a,\,j+b)-r(i+x+a,\,j+y+b)\right| \qquad \textbf{(2-2)}$$

It is assumed that SAD is used as the distortion measure within this thesis, and the maximum and minimum values of x and y are ±*D*. After obtaining the SAD values of all candidate references, these values will then be compared to find out which candidate block should be used as the prediction of the current MB. The selected candidate should have the minimum SAD value, and a MV (*u*,*v*) will be used to indicate the predicted block as stated below.

$$(u,v) = \arg \min_{-D \le x,y \le D} [SAD(x,y)] \quad \textbf{(2-3)}$$

After obtaining the MV (*u*,*v*), the residual error, *e*, for the current MB can be obtained by subtracting the pixel values of the predicted block from the pixel values of the current MB as stated below.

$$e(a,b) = c(i+a, j+b) - r(i+u+a, i+v+b) \quad \textbf{(2-4)}$$

Figure 2-3 below shows the target MB, candidate block, search window and candidate MV in graphical form for explaining the concept of block-based ME.

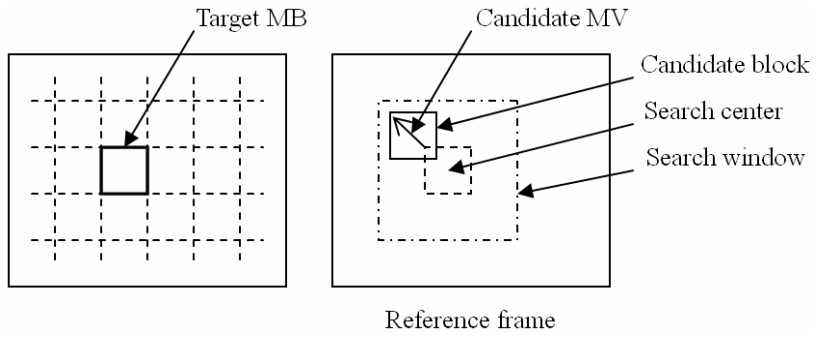


Figure 2-3. Block-based full search

2.2.2 Fast lossless approaches

Fast lossless algorithms are very favorable because of both the fast processing speed and the quality preservation. In section 2.2.2.1, the famous PDS [35] will be discussed. A modification of PDS called “clustered pixel matching error for adaptive

partial distortion search (CPME-PDS)" [38] will be introduced in section 2.2.2.2. It looks into the error surface to further speeds up the PDS by adaptively setting a suitable processing order of pixels for matching each current MB with its candidates. In addition, the successive elimination algorithm (SEA) [42] will be given in section 2.2.2.3.

2.2.2.1 Partial distortion search (PDS)

To reject a candidate block, the FSA will compute the SAD value of that candidate block, and compare it with the minimum SAD value of another candidate encountered before in the search within the search window. Since SAD value is the sum of many values corresponding to different pixels, for a candidate block, the distortion contributed by a certain amount of pixels may already go beyond the border of rejection. In other words, it is no need to compute the distortions in all pixels inside the candidate block and sum them up for rejecting the candidate. The main idea of PDS just follows this concept. It divides the computation of each SAD value into several stages. Instead of processing all the pixels of a block within one time, only a certain amount of pixels will be considered at each stage, and so incompletely computed SAD values will be output. Those intermediate SAD values are termed partial SAD (PSAD). After each stage, the incompletely computed SAD value will be compared with the lowest SAD value found so far, and checking whether the incomplete SAD is greater or not. The candidates under consideration can be safely rejected if their PSAD values are larger than the previous minimum SAD. Otherwise, more pixels can then be involved in the processing to see whether the candidate is still better than the tentatively selected best reference block. By operating in this way, the accumulation of SAD values for non-best matches can be halfway terminated before the SAD values are fully computed, and hence many unnecessary operations can be omitted. Note that it is ineffective to check

the PSAD value each time the distortion value for a single pixel is added to. A sensible approach is to allow checking once a row of pixels inside the candidate block is processed. This is also called the row-based PDS. The PSAD value after *p* consecutive rows of pixels have been processed becomes

$$PSAD_p(x,y)=\sum_{b=0}^{p}\sum_{a=0}^{15}\left|c(i+a,\,j+b)-r(i+x+a,\,j+y+b)\right|. \qquad \textbf{(2-5)}$$

Originally, the candidates within the search window can be searched one by one in the raster scanning order. In this order, the blocks in each row will be accessed from left to right, and the rows will be accessed from top to bottom as illustrated in Figure 2-4. This kind of scanning order is suitable for the FSA due to its simple realization and easy data accessing.

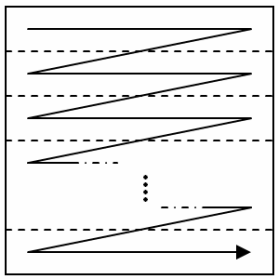


Figure 2-4. Raster scanning order of candidate blocks within a search window

However, the raster scanning order may not optimize the processing speed when the PDS is adopted as the ME scheme. In the PDS, the smaller is the SAD value of the searched candidate, the earlier will be the rejections of the remaining unsearched non-best candidates. This is because the candidates can be rejected once their PSAD values are larger than the SAD value of the tentative best match. If a candidate having a very small SAD value is found at the beginning of the search, the ME process will be completed very soon. In most real-world video sequences, the motion activities of objects and different parts of the backgrounds show a center-biased motion vector

distribution [24]. This means that MVs tend to concentrate at the center of the search window, and the population of MVs will decrease continuously as the distance between a candidate block and the search center increases. To take advantage from this characteristic of video sequences, a spiral scanning pattern as shown in Figure 2-5 can be used. This pattern will start the search from the search center in which the probability of founding the global minimum is highest, and then the search will be conducted from locations near the center to the farthest locations. In other words, using the spiral scanning pattern can make the candidates with small SAD values be checked earlier to form a faster PDS.

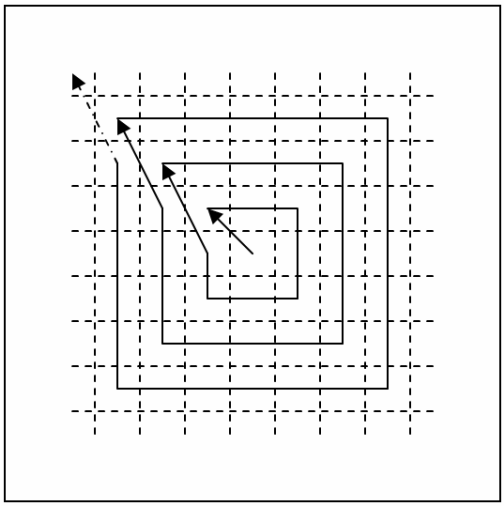


Figure 2-5. Spiral scanning pattern

2.2.2.2 Clustered pixel matching error for adaptive partial distortion search (CPME-PDS)

There are two way to raise the processing speed of the PDS. The first is to use a scanning pattern of candidate blocks that can let the candidates requiring smaller SAD costs be considered earlier than other candidates. This can be achieved by using such as the spiral search pattern introduced in Section 2.2.2.1. Another way to shorten the processing time of the PDS is to make the accumulation of partial SAD costs of non-

optimal candidates faster so that they can be accumulated beyond the terminating point earlier too. One possible approach is using a special pixel matching order during calculating the matching error between the current block and a candidate block that will match the pixels with large distortion values first. The raster scanning order - it processes the pixels from left to right in each row and from the top row to the bottom row - is the traditional order of matching the pixels inside the current block and its candidate, but it cannot fulfill the expectation of improving the accumulation of SAD values. The CPME-PDS [38] tries to adaptively determine a good pixel matching order for each motion search in advance of matching the candidate blocks with the current block so that the pixels with larger matching error can be considered first in general.

It has been found that similar distortion errors between a current block and its candidate blocks tend to appear in clusters instead of appearing individually. Therefore, after determining the high error region and low error region of the current block from one of its candidate, the pixel matching order can be defined and used for matching with that candidate as well as all other candidates around that it. To ensure a high accuracy of the pixel matching order, the candidate in the center of the search window is used to help finding different error regions of the current block since it is highly probable that the candidate block in the search center resemble the best match inside the search window. First of all, a reference value, m, which is the mean of the pixel values of the candidate block in the search center with coordinate (i,j) in the reference frame will be computed by using the following formulation:

$$m = \frac{1}{16 \times 16} \sum_{b=0}^{15} \sum_{a=0}^{15} r(i + a, j + b) \qquad \textbf{(2-6)}$$

Then the expected absolute pixel matching error of each pixel of the current block, $p_{exp}(a,b)$, will be calculated according to eqn. (2-7) shown below:

$$p_{\exp}(a,b) = \left| c(i+a,\, j+b) - m \right| \qquad \textbf{(2-7)}$$

where $0 \le a,\ b \le 15$. By sorting the expected absolute pixel matching error values decreasingly, the pixel matching order can also be defined. The order is from the pixel with the largest $p_{exp}(a,b)$ through the pixel with the smallest $p_{exp}(a,b)$. This order will then be applied while computing the SAD costs of all reference blocks within the search window. This kind of definition of the pixel matching order is called the pixel-based approach.

Alternative approach is the row-based approach in which the expected absolute pixel matching errors of the pixels in the same row will be summed together to obtain the degree of error of each row of pixels. The summed error values will be sorted such that an order of rows of pixels can be determined. Then during matching the current block with its candidate matches, the rows of pixels will be processed in that order. In each row of pixels, the pixels will be processed from the leftmost to the rightmost.

It seems that the row-based approach is less accurate than the pixel-based approach. In fact, using the pixel-based approach can make the PDS to be completed with fewer operations, but it is proven that the actual processing time of the row-based approach is less than that of the pixel-based approach for software application with processors which employ the regular memory pattern access and execution. That is why the row-based approach should not be excluded.

2.2.2.3 Successive elimination algorithm (SEA)

Other than the PDS and its modifications, SEA [42] is another well-known fast method of lossless ME scheme. The concept of the SEA is based on the Minkowski's inequality. For two values *A* and *B*, they must obey the Minkowski's inequality,

$$\left\| A \right| - \left| B \right\| \le \left| A - B \right| . \qquad \textbf{(2-8)}$$

Regarding the block matching problem, let A and B be $c(i+a,j+b)$ and $r(i+x+a,j+y+b)$ respectively. Therefore, eqn. (2-8) can be expressed as

$$\left\|c(i+a,\, j+b)\right| - \left|r(i+x+a,\, j+y+b)\right\| \le \left|c(i+a,\, j+b) - r(i+x+a,\, j+y+b)\right| . \quad \textbf{(2-9)}$$

By taking summation of both sides in eqn. (2-9),

$$\begin{aligned} &\left|\sum_{b=0}^{15}\sum_{a=0}^{15}\left|c(i+a,\, j+b)\right| - \sum_{b=0}^{15}\sum_{a=0}^{15}\left|r(i+x+a,\, j+y+b)\right|\right| \\ &\le \sum_{b=0}^{15}\sum_{a=0}^{15}\left|c(i+a,\, j+b) - r(i+x+a,\, j+y+b)\right| \end{aligned} . \quad \textbf{(2-10)}$$

Eqn. (2-10) is equivalent to the following two inequalities:

$$\begin{aligned} &\sum_{b=0}^{15}\sum_{a=0}^{15}\left|c(i+a,\, j+b)\right| - \sum_{b=0}^{15}\sum_{a=0}^{15}\left|r(i+x+a,\, j+y+b)\right| \\ &\le \sum_{b=0}^{15}\sum_{a=0}^{15}\left|c(i+a,\, j+b) - r(i+x+a,\, j+y+b)\right| \end{aligned} \quad \textbf{(2-11)}$$

and

$$\begin{aligned} &\sum_{b=0}^{15}\sum_{a=0}^{15}\left|r(i+x+a,\, j+y+b)\right| - \sum_{b=0}^{15}\sum_{a=0}^{15}\left|c(i+a,\, j+b)\right| \\ &\le \sum_{b=0}^{15}\sum_{a=0}^{15}\left|c(i+a,\, j+b) - r(i+x+a,\, j+y+b)\right| \end{aligned} . \quad \textbf{(2-12)}$$

In the above inequalities, the first term in eqn. (2-11) and the second term in eqn. (2-12) in the left hand side of the inequalities is the sum norm of the current block. The second term in eqn. (2-11) is equal to the first term in eqn. (2-12) in the left hand side of the inequalities, which is the sum norm of a candidate in the reference frame. The sum in the right hand side of the inequalities is, in fact, the SAD between the current block and the candidate block. For simplicity, eqn. (2-11) and eqn. (2-12) can be rewritten as

$$C - R(x,y) \le SAD(x,y) \quad \textbf{(2-13)}$$

and

$$R(x,y) - C \le SAD(x,y) . \quad \textbf{(2-14)}$$

Originally, to decide whether a candidate match will be chosen as the new tentative best match, its SAD cost will be compared with the SAD cost of the existing tentative best match that must be the minimum value among the SAD costs of all previously searched

candidate blocks. Let the MV corresponding to the existing tentative best match be (u,v). The candidate block with MV, (x,y), will replace the position of the existing tentative best match if

$$SAD(x, y) < SAD(u, v). \quad \textbf{(2-15)}$$

It must be true if the candidate match is better than the existing best match that

$$C - R(x, y) < SAD(u, v) \quad \textbf{(2-16)}$$

and

$$R(x, y) - C < SAD(u, v). \quad \textbf{(2-17)}$$

The above two conditions can be combined and rewritten as

$$C - SAD(u, v) < R(x, y) < C + SAD(u, v) \quad \textbf{(2-18)}$$

This means that it is not required to compute the SAD costs of all candidate blocks to find out which of them are better than the tentative best matches found before. Instead, for each candidate block, only an R value is needed to be computed and checked if it matches the condition in eqn. (2-18). Since calculating an R value requires much fewer operations than calculating a SAD value, a large number of operations can be saved for those non tentative best matches. Only when the R values match the condition, the SAD costs of the corresponding candidates are required to be computed. In addition, different from SAD values, R values not only can be used for the block matching for the current block, they can also be reused for subsequent blocks in the current frame that will search the repeated locations in the reference frame. Therefore some R values of the candidate locations inside the search window may not needed to be computed for carrying out the block matching for a block in the current frame in each time, and it is true that the computational complexity is further reduced.

The processing speed of SEA is also affected by the means of obtaining the sum norm of candidate blocks in the reference frame. A fast method to compute the sum norms can greatly shorten the operating time. It is suggested to compute the sum norms

of all blocks in the reference block at once other than compute them when they are in need. Since the same pixels are contained in neighboring candidate blocks, the sum of the pixel values of most pixels within a block can be reused in the calculation of the sum norm of neighboring blocks. In practice, the first step of a good method is to compute the sums of pixel values of all vertical strips of 16 pixels in all positions of the reference frame. These sums can be called strip sums. Assume that the strip sums of the vertical strips at the top of the reference frame are computed first, the strip sums of remaining vertical strips can be computed not by directly summing the pixel values together by taking 15 addition operations. Since each vertical strip overlaps its upper vertical strip in 15 pixels, the sum of pixel values of a vertical strip can be obtained by subtracting a pixel value from and also adding another pixel value to the strip sum of its upper vertical strip. Only two operations are involved in the calculation. After getting the sums of pixel values of vertical strips, every 16 of them can be combined to form the sum norm of a candidate block. A better way is to compute the sum norm of the leftmost candidate block in each row in the reference frame first so that the sum norm of each remaining candidate block can simply be formed by subtracting a strip sum from and also adding another strip sum to the sum norm of its left candidate block.

2.2.3 Fast lossy approaches

Fast lossy algorithms sacrifice some degree of quality of the coded sequences for getting ME done within very short period of time. The n-step hierarchical search (n-SHS) [18] and the diamond search (DS) [20] are two most typical types of fast lossy search methods, and will be discussed in Section 2.2.3.1 and 2.2.3.2 respectively. The hexagon-based search (HEXBS) [30] will also be included in Section 2.2.3.3 to be compared with the DS.

2.2.3.1 N-step hierarchical search (n-SHS)

The n-SHS divides the whole procedure into n steps that 9 positions will be considered in each step, and the positions to be searched in each step are decided just after completing the previous step. The loose to tight strategy is adopted that the candidate positions in the first step are widely separated, and the distances separating the candidate positions will become shorter and shorter as the process proceeds. In the first step, the positions to be searched are just based on the size of the search window. The distance between any two candidate positions is halved for the second step, and it will be further halved for the third step, and so on, so the total number of steps also depends on the size of the search window. In each step, the SAD costs of the candidate positions will be checked, and then the position which has the minimum cost will become the center point of the next step. The SAD values computed in previous steps will be stored in memory. If some candidate positions are considered in previous steps, then no repeated computation is needed. An example of the search pattern of the n-SHS can be found in Figure 2-6. In this example, the size of the search window is (±6,±6), so a three-step search is performed.

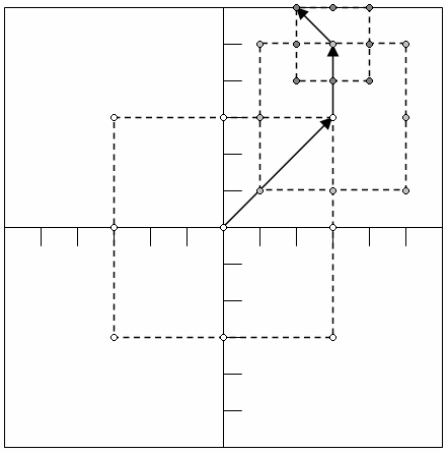


Figure 2-6. An example of performing n-SHS

There is an obvious disadvantage of the n-SHS. Since a relatively large search pattern is used in the first step, the efficiency of finding the best matches with small motions will be low. It is against the fact that most real-world videos have a center-biased motion vector distribution [24]. It has been found on average that 80% of image blocks are stationary with MV = (0,0), and 90-99% of image blocks have MVs within (±3,±3).

2.2.3.2 Diamond search (DS) and hexagon-based search (HEXBS)

For DS [20], there are two different search patterns as shown in Figure 2-7. The reference positions inside a unit circle with radius = 2 pixels are used to compose the search patterns. The first pattern is called the large diamond search pattern (LDSP) which contains 9 search points. Eight of them in the periphery form a diamond shape so that this pattern is called “diamond search pattern” and also the search scheme is called “diamond search”. Another pattern is the small diamond search pattern (SDSP), and comprises of 5 search points forming a smaller diamond shape. To begin with, the LDSP is applied at the search center. The SAD values of the corresponding 9 search points will be checked to find out the block with the minimum cost. If the minimum SAD value is not found at the center point, the center of the LDSP will move to that position having the minimum SAD value, and then the new 9 search points will be checked as what the first step has done. This process will be repeated until the center point of the LDSP contains the minimum SAD value among all searched points. Then the LDSP will be switched to the SDSP. Among the 5 search points of the SDSP, the block located in the point yielding the minimum SAD value will be the final selected block.

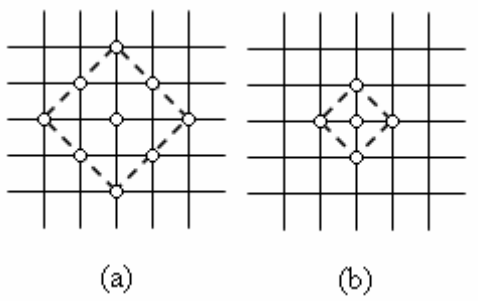


Figure 2-7. The search patterns of DS: (a) LDSP; (b) SDSP

The DS starts the search from the search center so it matches well with the center-biased motion vector distribution. Since the SAD value at the center of the LDSP keeps on decreasing, convergence is guaranteed, and the searching process will never become a closed loop that it must reaches an end point. However, there are some problems about the DS. Regarding the LDSP, the 8 checking points around the center point have different distances from the center point. The distances from the points at the vertices of the diamond are 2 pixels from the center point, but that from the points at the faces of the diamond are $\sqrt{2}$. This means that the searching time will vary with the moving direction of the contents of image blocks even though the moving speed is the same. In addition, there are different numbers of new candidate points after the diamond moving in different directions so that the searching speed is also affected by this factor. If the diamond moves horizontally or vertically, there will be 5 new candidates. On the other hand, if the diamond moves diagonally, there will be only 3 new candidates.

In order to rectify the flaws of the DS, search patterns which approximate to a unit circle are preferred. The HEXBS [30] was proposed to meet this goal, which uses exactly the same mechanism as the DS except the search patterns. The two search patterns adopted in the HEXBS are shown in Figure 2-8. As for the large hexagonal pattern, 7 checking points are contained, and the distances between an outer point and

the center point is just ranging from $\sqrt{5}$ to 2 pixels. No matter which direction the hexagon moves, there are always 3 additional candidate points.

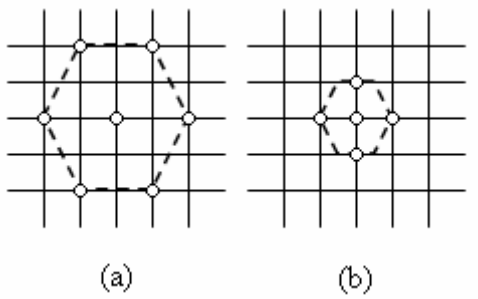


Figure 2-8. The search patterns of HEXBS: (a) large hexagonal pattern; (b) small hexagonal pattern

Comparing the processing time between these two algorithms, the results show that the HEXBS can achieve about 40% of improvement over the DS. Nevertheless, since monotonically varying error surface is assumed in this kind of fast searching schemes, the problem of being trapped into local minima occurs easily in both algorithms, especially when the coding of video sequences with high motion activity. Besides, both algorithms do not show good performance if the motions between video frames are large because they always start the search from the search center, and move the pattern to faraway locations slowly along a long path.

2.3 *H.264 advanced video coding standard*

In Section 2.3.1, a brief introduction of the H.264 advanced video coding (AVC) standard [6] will be given. The features of this standard will be introduced in a glance in this section. However, the concepts and applications of one of the important part of H.264 – rate-distortion optimization (RDO) [17, 62] will be described in details in Section 2.3.2. Sections 2.3.3 and 2.3.4 include some fast algorithms which are suitable

to be implemented in H.264 for variable block size ME and for the multiple reference frames ME.

2.3.1 Overview

H.264 is the most advanced international video coding standard, and was approved by ITU-T as MPEG-4 part 10 advanced video coding. In the past, the operating speed was more important than video quality in the past, especially for real-time video processing such as video conferencing. With the rapid development of processor speed and memory size, computers can handle a larger amount and more complex tasks within a limited time. Therefore, it is possible to pursue better video quality by using a more precise coding scheme while the coding speed is kept very fast. In early 1998, the video coding expects group (VCEG) called for proposals on a project called H.26L targeting to double the coding efficiency of other video coding standards for different kinds of applications. The first draft of the expected standard was finished and adopted in October 1999. In December 2001, the VCEG and the MPEG formed a joint video team (JVT) in order to finalize the draft of the standard, and the H.264/AVC standard was resulted and submitted in March 2003.

In the H.264/AVC standard, only the decoder part is standardized that the bit-stream and syntax are put under restrictions. The decoding process was defined to ensure each decoder produce similar output if an encoded bit-stream conforming to the limitations of the standard is given. Since the encoding process is not restricted, crude encoding techniques may be used so that the end-to-end reproduction quality is not guaranteed.

The standard employs many complex techniques in order to advance the coding efficiency. It supports variable block size and shape [68] for motion estimation and

compensation to enhance the prediction accuracy of block-based ME method and also the compression ratio. There is a large range of the selection of block sizes which varies from 16x16 down to 4x4 pixels. It also allows multiple numbers of reference frames for ME [69] to carry out motion searches from more than one decoded previous frame stored in the decoder. Improved prediction results can be obtained in return. To optimize the results of a variety of technologies adopted by the standard, the rate-distortion optimization (RDO) [17, 62] technique is used, which is essential to make further improvement of the coding efficiency.

2.3.2 Rate-distortion optimized video coding

RDO is a very important feature adopted in H.264/AVC standard. It controls each coding result to an optimal point that the PSNR and the output bit-rate are in balance. This feature is crucial to carrying out good ME, coding block size selection, reference frames selection and intra-prediction. To get into the concept of RDO, an introduction is given in Section 2.3.2.1. The applications of RDO in diverse parts of video coding including general ME, multiple reference frames ME and block size selection method are explained respectively from Section 2.3.2.2 to 2.3.2.4.

2.3.2.1 Introduction to rate-distortion optimization

PSNR and bit-rate are good measures of the qualities of coded video sequences. PSNR reveals the degree of distortion of the image contents between coded sequences and their original representations. There exists a relationship of the PSNR and distortion that one will have high value while the other one is in low value. It is always desirable that both the distortion and the bit-rate are low for a video sequence after encoding as this means that a compressed video with high visual quality can be stored without the

requirement of large memory size, or can be transmitted rapidly through a low capacity channel. Moreover, it seems as if this case does not occur in reality. With the same coding tools in the encoder, different encoder strategies will have different operating points as shown in Figure 2-9. In the figure, it can be found that it is impossible to minimize either the distortion or bit-rate without sacrificing the other. The optimal results are represented by a curve instead of an optimal point. In this curve, if low distortion is obtained, the bit-rate must be high. Inversely, if the encoder focuses on depressing the bit-rate, the distortion will become large. There is a trade-off between low distortion and low bit-rate. Although there are a lot of "optimal" points in the curve, there exists only one point that can be known as the optimal among all those possible points if certain requirements or applications are upheld. This brings the problem of RDO. Some rate-distortion optimizing techniques such as Lagrangian optimization and dynamic programming were developed to help finding out the optimal point.

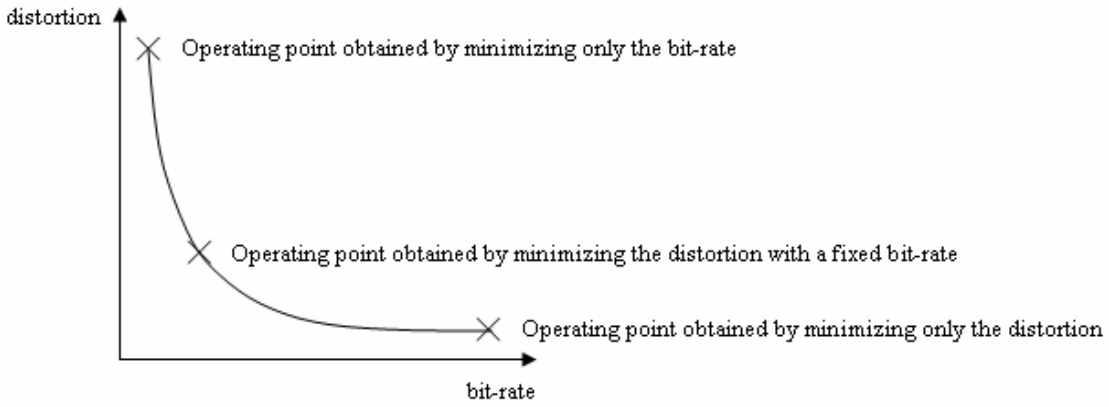


Figure 2-9. Rate-distortion plot of an optimized encoder

The Lagrangian optimization technique brings various stages of the encoder to their optimal points by minimizing a cost function in the form of

$$J = D + \lambda \cdot R \qquad \textbf{(2-19)}$$

where *D* is a distortion term, *R* is a rate term, and λ is a Lagrangian multiplier which is a non-negative real number keeping a balance between *D* and *R*,. The minimum value of *J* will change in accordance with the value of λ. For a fixed value of λ, there must exist a unique solution of the minimum value of *J*, and it can be determined by putting different pairs of *D* and *R* into eqn. (2-19), provided that the lower bounds of *D* and *R* are not independent of each other. In graphical, Figure 2-10 shows that the tangents touching at different points of the boundary between the achievable and the non-achievable regions of the R-D performance have different slope values. The value of λ, in fact, is the negative slope of the tangents of the boundary. This means that there is only one optimal point which can be found as long as λ is fixed. Smaller the value of λ is, more horizontal the tangent is, and the point with a smaller D and a larger R will be the optimal point. Similarly, larger the value of λ is, more vertical the tangent is, and the point with a larger D and a smaller R will be the optimal point.

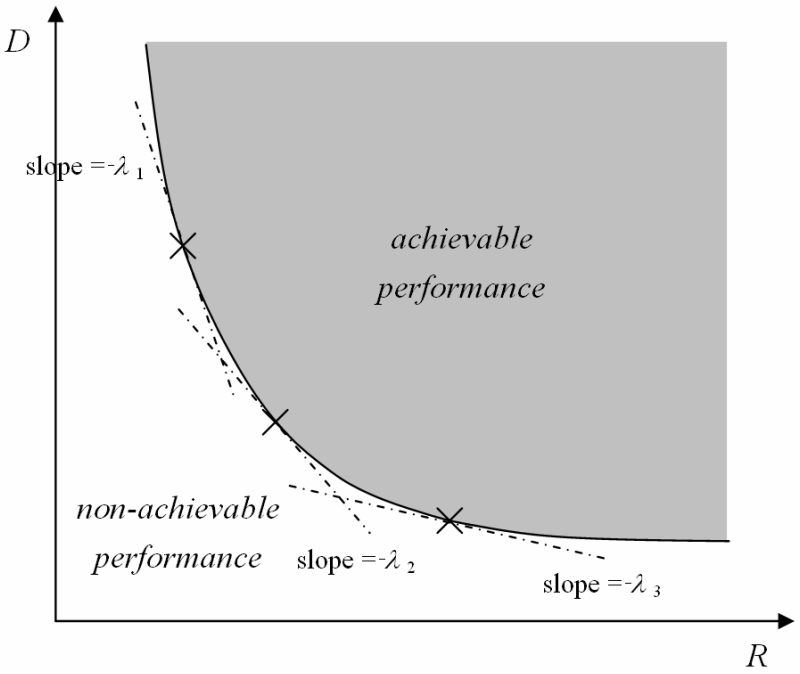


Figure 2-10. The value of λ decides the optimal R-D point

Generally, the optimal operating point of an encoder can be controlled by adjusting the quantization parameter (QP). Applying a larger QP means wanting to sacrifice more about the visual quality to pursue a lower bit-rate. Inversely, setting a smaller QP value is equivalent to allowing a higher bit-rate in exchange for a higher visual quality. Since the possible values of QP are usually discrete integers, the optimal operating points corresponding to different QP values are also discrete as depicted in Figure 2-11. To make the Lagrangian optimization applicable to video coding, the terms in eqn. (2-19) must be related to variables and parameters in the coding issues. Because of the same nature of QP and Lagrangian multiplier in eqn. (2-19), the Lagrangian multiplier should be a function of QP.

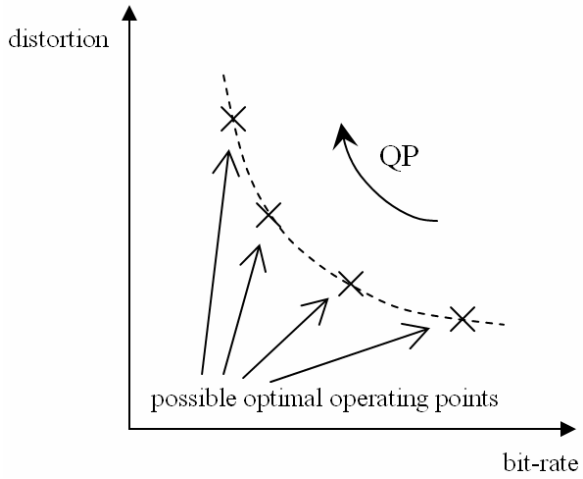


Figure 2-11. Discrete optimal operating points correspond to different QP

2.3.2.2 RDO for general motion estimation

Originally, ME of an image block is carried out by finding a substitute block from a reference frame which has the minimum distortion from the original block. As mentioned before, not only the visual quality determines the performance of a coding process, but also the bit-rate does. Although taking the original approach can result in the best visual quality, since the information about MVs are required to be coded into

bit-stream in addition to coding the block differences, a large number of bits may be required if the magnitudes of MVs are large, which leads to a high bit-rate and hence lower the coding efficiency. Therefore, it is better to include the bit-rate in the block matching criterion besides the distortion value to make ME a RDO process. Assume that SAD is the distortion measure, the cost function of ME now becomes

$$J_{motion}(m) = SAD(m) + \lambda_{motion} \cdot R_{motion}(m), \quad \textbf{(2-20)}$$

where m is a candidate MV of a reference block, $SAD(m)$ is the SAD value between the original block and the reference block pointed by m, λ_{motion} is a Lagrangian multiplier, and $R_{motion}(m)$ is the approximated number of bits which are required when the information of m is entropy coded. In H.264/AVC, the value of λ_{motion} can be written as

$$\lambda_{motion} = \sqrt{0.85 \cdot 2^{(QP-12)/3}}, \quad \textbf{(2-21)}$$

which is controlled by the quantization parameter, QP, that can be any integer between 0 and 51.

The following is an example demonstrating the use of RDO for ME. In the example, the first step of an integer-pixel three step search of a 4x4 block is performed. Figure 2-12 shows the pixel values of the 4x4 block, and Figure 2-13 shows the pixel values of the search window. In Figure 2-13, the crosses indicate the top-left corners of the nine candidate blocks with 4x4 sizes. Let the QP be 28, the value of the λ_{motion} should be 5.85. After calculating the SAD values of the nine candidate blocks and their corresponding R_{motion} values, the J_{motion} values of the candidate blocks can be obtained by using eqn. (2-20), and hence Table 2-1 is formed. According to the table, the block at the position pointed by m = (6,3) is the best match as it has the minimum J_{motion} value among all candidates.

61	45	76	30
103	94	58	44
41	3	38	31
80	97	4	37

Figure 2-12. Pixel values of the current 4x4 block in the example

	0	1	2	3	4	5	6	7	8	9
0	27	47	74	112	8	85	93	120	50	42
1	86	34	35	72	43	35	45	74	55	110
2	62	216	36	39	54	57	98	81	49	96
3	95	76	78	198	29	36	71	30	70	106
4	99	96	96	50	30	88	48	211	33	23
5	191	53	97	62	47	66	66	48	74	59
6	175	53	34	64	3	230	67	80	6	39
7	211	84	249	41	6	61	90	240	2	89
8	73	234	74	93	84	30	54	29	49	52
9	48	30	33	7	53	57	76	46	98	94

Figure 2-13. Pixel values of the search window in the example

Table 2-1. SAD, R_{motion} and J_{motion} values of the nine candidate blocks in the example

m	*SAD*	$R_{motion}(m)$	$J_{motion}(m)$
(0,0)	763	2	813.54
(3,0)	657	6	692.12
(6,0)	714	8	770.54
(0,3)	771	6	806.12
(3,3)	879	10	898.71
(6,3)	493	12	534.12
(0,6)	1029	8	1085.54
(3,6)	808	12	849.12
(6,6)	653	14	715.54

2.3.2.3 RDO for multiple reference frames motion estimation

One of the important features of the H.264/AVC standard is multiple reference frames ME [69]. For the traditional single reference frame ME, the nearest decoded frame will be used as the unique reference frame of the current frame, so the better predicted blocks in prior decoded frames will be overlook. Accordingly, instead of searching just one reference frame, it is suggested to search more decoded frames before

the current frame to look for a better prediction which requires a smaller cost function. The prediction accuracy of ME can be improved by allowing more reference frames because part of the video objects or part of the background in the current frame may sometimes not exist in the nearest preceding frame. There are many possible reasons that the best matches cannot be found from the nearest reference frame. For example, an object may be blocked by some foreground objects, or it may just occasionally move outside the frame, or may even somehow disappear suddenly. It is also possible that the object or background is in the nearest reference frame, but its contents are distorted. In this case, better references of it may be found from other reference frames [59].

Each reference frame is indexed, and the indexing is from the frame which is nearest to the current frame in the coding order to the farthest frame. In order to let the decoder correctly extract the predicted block from the right reference frame to decode each image block, it is necessary to inform the decoder the index of the reference frame in which the predicted block locates. Therefore an index representing the information of the reference frame should be coded and transmitted for coding each image block.

Since the reference frame index of a farther reference frame is larger, coding it will generate more bits than coding the index of a nearer frame. If RDO is not implement in multiple reference frames ME, that is the block matching criterion is the distortion value only, the ME engine may select from a far reference frame a candidate block which has the minimum distortion cost, but coding the index of that reference frame will require many bits. As a result, the overall coding performance is not optimized. That is the reason why RDO is so important to multiple reference frames ME. The arrangement of the indexing is reasonable as it is found that the probability of finding a best match from a nearer frame is larger so this arrangement can reduce the

amount of bits. Assume that the SAD is used to measure the distortion between blocks, generally, the RD cost of each candidate block in each reference frame is

$$J_{motion}(m,r) = SAD(m,r) + \lambda_{motion} \cdot \left[R_{motion}(m) + R_{ref}(r)\right], \quad \textbf{(2-22)}$$

where r is a reference frame index, $SAD(m,r)$ is the SAD value between the original block and the reference block with a candidate MV, m, in a reference frame indexed by r, and $R_{ref}(r)$ is the approximated number of bits which are required when the information of r is entropy coded.

2.3.2.4 RDO for block size selection

Variable block size (VBS) motion estimation and compensation [68] is another special feature of the H.264/AVC standard. Due to the large size of a MB, the probability that a MB contains more than one objects moving in different velocities is large. Using just one fixed block size and one motion vector to describe the motion of each 16x16 region is insufficient. This is the reason that VBS are proposed so that blocks of different sizes are used. With the VBS-ME, more accurate predictions of video objects and the background can be resulted. The H.264 standard supports seven block sizes with a tree-structured hierarchical MB partition as displayed in Figure 2-14 instead of the unique 16x16 block size adopted in many former standards like MPEG-2 and H.263. Each MB can be partitioned into 16x16, 16x8, 8x16 or 8x8 blocks, and each 8x8 block can be further partitioned into 8x8, 8x4, 4x8 or 4x4 blocks. This allows different objects in a MB estimate their own MVs and have their own predicted blocks instead of share the same MV.

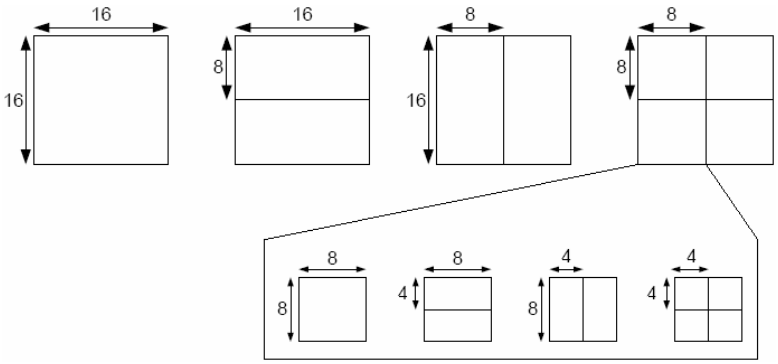


Figure 2-14. Possible MB partitions adopted in the H.264 standard

Using different block sizes can be treated as applying different coding modes. To carry out the variable block size ME for a MB, motion searches will be performed with different coding modes so that several sets of MVs are obtained. Afterwards, the best coding mode will be determined by comparing the cost functions corresponding to various coding modes.

There is an important point relating to the use of variable block size ME. When a MB undergoes variable block size ME, if the formulation of the cost function involves just the distortion term, the cost functions obtained by using smaller blocks must be lower than those obtained by using larger blocks. This is because it is always true that smaller blocks must find better representations than larger blocks. Moreover, using smaller blocks requires coding more MVs, and hence more bits may be required. Therefore, there exists a balance point between the image quality and the bit-rate. To find out the block size that fits the balance point, RD cost can be used as the formulation of the cost function. If SSD is the distortion measure, the RD cost of each coding mode is suggested to be

$$J_{mode}(I) = [SSD_Y(I) + SSD_U(I) + SSD_V(I)] + \lambda_{mode} \cdot R_{mode}(I), \quad \textbf{(2-23)}$$

where I is a coding mode in a set {16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4}, $SSD_Y(I)$, $SSD_U(I)$ and $SSD_V(I)$ are the SSD values between respectively the Y, U and V

components of the original MB and those of the MB reconstructed from entropy coded data of the predicted blocks in mode I, λ_{mode} is a Lagrangian multiplier, $R_{mode}(I)$ is the approximated number of bits which are required when the MB is coded in mode I. In H.264/AVC, the value of λ_{mode} would be

$$\lambda_{mode} = 0.85 \cdot 2^{(QP-12)/3}. \quad \textbf{(2-24)}$$

2.3.3 Fast algorithms of variable block size motion estimation

Although variable block size ME can provide better qualities of compressed videos than traditional ME, it requires much computational effort which largely increases the burden of the processor. The number of computations is almost seven times because MEs take place for seven block sizes instead of a unique block size.

There are many fast realizations of variable block size ME [51-56] already developed including lossless and lossy approaches, especially in recent years. In succession, three effective fast algorithms will be reviewed. Firstly, Section 2.3.3.1 will introduce the SAD reuse algorithm [51]. The merge and split method [52], and the texture and local motion activity analyzing method [55] will then be described respectively in Section 2.3.3.2 and 2.3.3.3.

2.3.3.1 SAD reuse algorithm

SAD reuse algorithm is a lossless method to compute the SAD costs of all candidate matching blocks of the blocks with different block sizes within each MB in a faster manner. In H.264/AVC standard, the partitioning of MB is designed in such a way that each block is composed of two smaller blocks, except for the smallest 4x4 blocks. For example, an 8x8 block comprises two 8x4 or 4x8 blocks. Therefore, the SAD values corresponding to two adjoining smaller blocks may be useful for

computing the SAD values for a larger block. In other words, the processing order must be from the smallest block size to the largest block size. To start with, the SAD values for all 4x4 blocks are needed to be computed through a usual way. Then the SAD values for blocks with other block sizes are obtained through the steps shown in Figure 2-15. Each SAD value requires just one addition only, and block matching is no longer to be performed for in total 6 block sizes so the number of operations is much less as compared with the original approach.

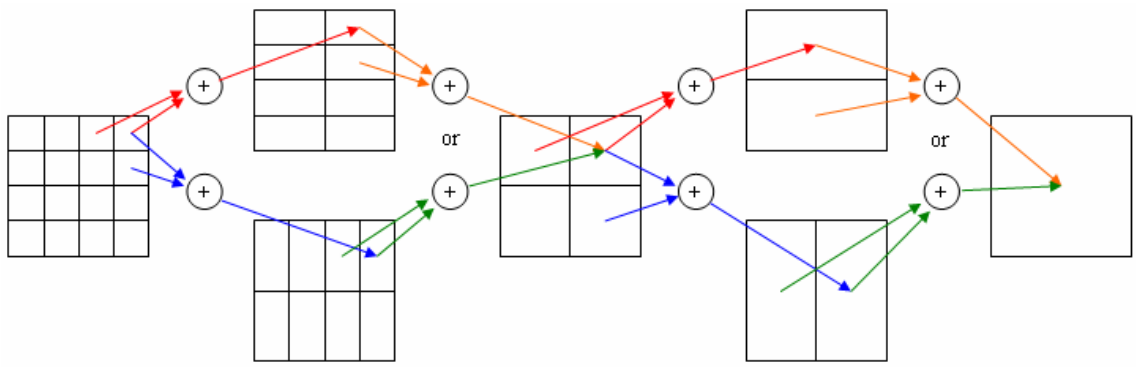


Figure 2-15. SAD reuse algorithm

2.3.3.2 Merge and split method

The search center is the center of the region in reference frame in which the search pattern applies to carry out ME. It is an important component of ME since a good search center which is close to the best match of the current block can increase the accuracy of the search and can allow diminishing the size of the search region. In prior video coding standards, search center is just the same as the position of the block to be coded in the current frame. Tourapis et al. [78] suggested that the search center can be the median, the mean, or the one corresponding to the minimum distortion of the MVs of spatial and/or temporal neighboring blocks. Then the vector from the position of the current block to the predicted search center can be called predicted MV (PMV).

However, using the motions of neighboring blocks to predict the motion of a block is based on the assumption that the MV field is homogeneous. The assumption may be invalid in case there are many local motion activities and small moving objects.

Considering variable block size ME, since the blocks of different sizes contain the same contents of the MB, there is a strong correlation among the MVs of them, and the correlation should be stronger than that among neighboring blocks. The merge and split method [52] makes use of this kind of correlation to predict the search center so its accuracy is much better. The prediction process starts from the 8x8 block size, and it separates into a bottom-up merging process and a top-down splitting process. The merging and the splitting processes are demonstrated in Figure 2-16.

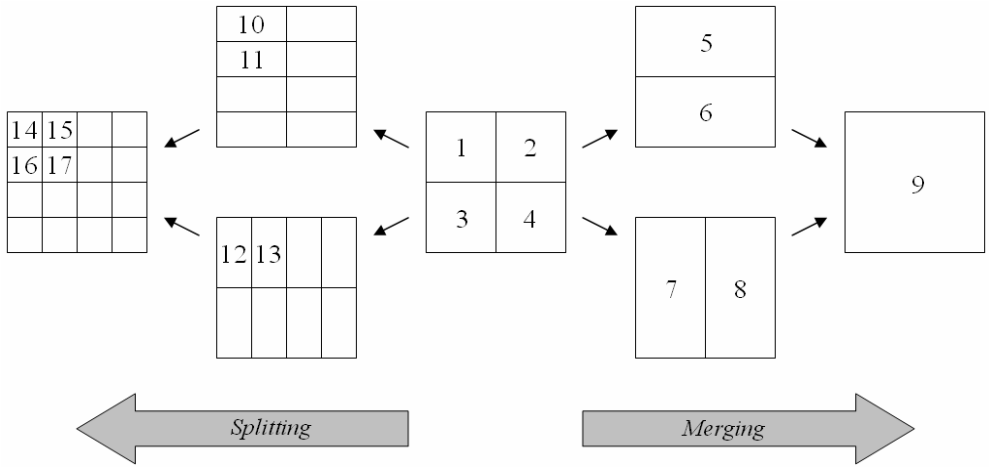


Figure 2-16. The merging and the splitting processes of predicting PMVs

The PMV of a 16x8 block or an 8x16 block is just the average of the MVs of relative 8x8 blocks:

$$\begin{cases} PMV_5 = (MV_1 + MV_2)/2 \\ PMV_6 = (MV_3 + MV_4)/2 \\ PMV_7 = (MV_1 + MV_3)/2 \\ PMV_8 = (MV_2 + MV_4)/2 \end{cases} \quad \textbf{(2-25)}$$

As a 16x16 block contains both 16x8 and 8x16 blocks, its PMV is calculated by taking the mean of the MVs of relative 16x8 and 8x16 blocks:

$$PMV_9 = \left(MV_5 + MV_6 + MV_7 + MV_8\right)/4 \quad \textbf{(2-26)}$$

The MV of each 8x8 block is used as the PMVs of the 8x4 and 4x8 blocks within the same 8x8 block. For instance, the PMVs of blocks '10'~'13' in Figure 2-15 are equal to the MV of block 1. Since each 4x4 block is covered by an 8x4 block and a 4x8 block, the PMV of a 4x4 block is the average of the MVs of the 8x4 and 4x8 block covering it:

$$\begin{cases} PMV_{14} = \left(MV_{10} + MV_{12}\right)/2 \\ PMV_{15} = \left(MV_{10} + MV_{13}\right)/2 \\ PMV_{16} = \left(MV_{11} + MV_{12}\right)/2 \\ PMV_{17} = \left(MV_{11} + MV_{13}\right)/2 \end{cases} \quad \textbf{(2-27)}$$

2.3.3.3 Texture and local motion activity analyzing method

The original way to carry out variable block size ME requires considering all block sizes, but it is computationally very expensive. A large amount of operations can be saved if we can detect which block modes are impossible in a prior stage, and skip these block modes. Shen et al. [55] suggested that analyzing the texture of MBs can provide clues to reject some block modes.

If the texture of a MB is found to be smooth, then the whole MB must belongs to a video object or the background, and only the biggest 16x16 block size is needed to be processed. The other six block modes can be safely skipped. The roughness of a MB can be revealed by the variance which is calculated as

$$Var = \sum_{b=0}^{15}\sum_{a=0}^{15}\left|c\left(i+a, j+b\right) - \bar{c}\right|, \quad \textbf{(2-28)}$$

where $\bar{c}$ is the mean of the pixel intensities of the current MB. A MB is classified as smooth if its variance is smaller than an empirical threshold value.

Besides the texture of MBs, local motion activity is another kind of information that can be used to help simplifying the variable block size ME process [53]. As mentioned before, since the blocks of different sizes cover the same contents of a MB, if several neighboring blocks contain the same or very similar MV (say for example, seven 4x4 blocks contain the same MV, and the MV of the remaining 4x4 block has difference of just one pixel), it is highly probable that the larger block covering these blocks has the MV of these blocks. For the same MV, coding a larger block is better than coding a number of smaller blocks inside the larger block because of relatively smaller overheads. Therefore in this case, those smaller blocks will be merged into a larger block, and the MV of smaller blocks will be used as the final MV of the larger block, and also no further motion search of other block sizes is necessary. Consequently, it is possible to obtain the optimal search results while motion searches are only performed for the smallest 4x4 block mode.

2.3.4 Fast algorithms of multiple reference frames motion estimation

Multiple reference frames ME considers more than one reference frame to increase the quality of coded sequences, but the processing time is directly proportional to the number of reference frames allowed. In order to relieve the heavy computational burden, some fast algorithms were developed. Since high temporal correlations exist among successive coded frames, it is good to reuse the motion vectors of previous frames to predict the motion of the current block [57, 58]. The details of short-term

motion vectors reusing is in Section 2.3.4.1. Then in Section 2.3.4.2, the long-term motion vectors reusing scheme will also be introduced.

2.3.4.1 Short-term motion vectors reuse algorithm

In the conventional scheme of multiple reference frames ME, the search windows in various reference frames have their centers all located at the same position of the block to be coded in the current frame. However, as the objects in video frames keep on moving, they may be at very different positions at different time instant even if the motion is slow. It is also likely that the spatial position of an object in two different frames is farther away as the time difference becomes larger. Therefore, if the search window in each reference frame is placed at the position of the current block, the probability that the objects are outside the search window will be larger. As mentioned early, good search centers or PMV is essential to the accuracy of motion searches, and they should be found such that a smaller size of search windows can be used to reduce the number of search points.

To roughly predict the motion of an image block in different reference frames, the MVs of the blocks in previously coded frames can be used. Short-term MVs reusing involves reusing the MVs of the nearest previous frame only. Figure 2-17 shows how to reuse the obtained MVs in this scheme. Firstly, motion search is performed in the nearest reference frame with a relatively large search window to obtain an accurate motion of the current block, and the motion is represented by MV_1^t. Since the motions of different image blocks in the nearest reference frame t-1 with reference to earlier reference frames were already estimated after reference frame t-1 was coded, the MVs belonging to the position of the targeted object pointed by MV_1^t in reference frame t-1 can be added to MV_1^t to become the predicted MVs of the object pointing to other

reference frames. As the predicted block in reference frame *t*-1 can be in arbitrary position, but only the blocks in specific positions were coded and have MVs, the strategy is to select a dominant block which covers most portion of the predicted block in reference frame *t*-1, and reuse only the MVs of the dominant block. Let the MVs of the dominant block corresponding to reference frames *t*-2 to *t*-*N* be MV_1^{t-1} to MV_{N-1}^{t-1}, the PMVs of the current block corresponding to different reference frames are

$$PMV_r = MV_1^t + MV_{r-1}^{t-1}, \quad \textbf{(2-29)}$$

where $2 \le r \le N$ which is a reference frame index. Then a smaller search window can be applied to the motion search in each of the remaining reference frames.

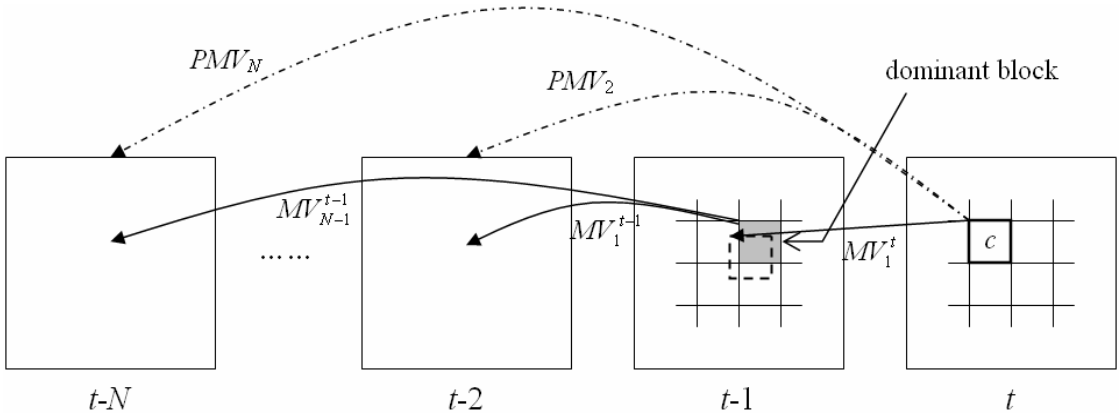


Figure 2-17. Short-term motion vectors reusing

2.3.4.2 Long-term motion vectors reuse algorithm

Long-term MVs reusing scheme is an alternative way to determine the PMVs of current block. The short-term MVs reusing needs storing and making use of a set of MVs of just the nearest reference frame, whereas the long-term MVs reusing needs storing and using the first MVs of a set of continuous reference frames. Noted that both schemes require the same memory size for the overheads as well as same number of operations to compute the PMVs. Figure 2-18 illustrates the mechanism of long-term

MVs reusing. Except for searching the first reference frame *t*-1, the PMV corresponding to each reference frame will be determined just before searching that reference frame.

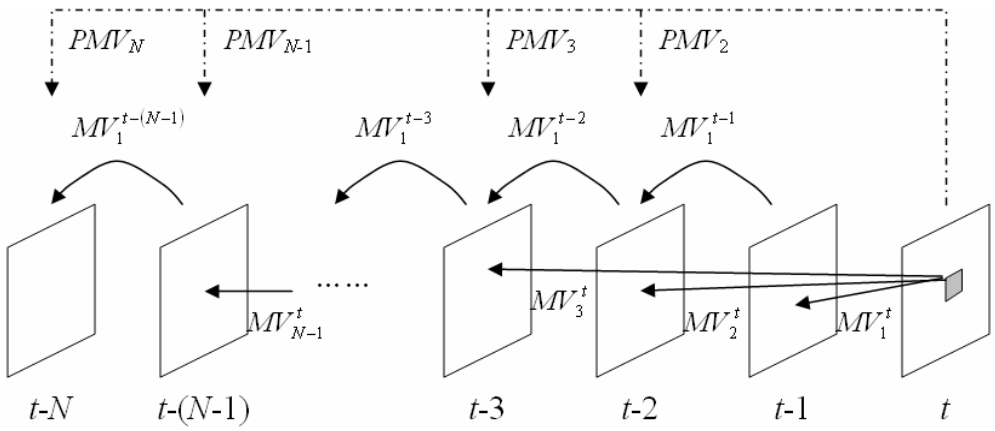


Figure 2-18. Long-term motion vectors reusing

First of all, the reference frame *t*-1 is searched with a relatively large search window; hence a motion vector, MV_1^t, is resulted. Then the predicted motion vector for reference frame *t*-2, PMV_2, is calculated by adding up MV_1^t and MV_1^{t-1} which is the MV of the dominant block in reference frame *t*-1 pointing to frame *t*-2. After that, reference frame *t*-2 is searched with a smaller search window centered at the newly found search center pointed by PMV_2, and then a motion vector, MV_2^t, is outputted. Similarly, PMV_3 is computed by adding up MV_2^t and MV_1^{t-2} which is the MV of the dominant block in reference frame *t*-2 pointing to frame *t*-3. This same process is repeated until the final reference frame is searched. The way to compute PMVs in this scheme is summarized as follow:

$$PMV_r = MV_{r-1}^t + MV_1^{t-(r-1)} \qquad \textbf{(2-30)}$$

where $2 \le r \le N$ which is a reference frame index.

Chapter 3. A Fast Full Search Scheme for Rate-Distortion Optimization of Variable Block Size and Multi-frame Motion Estimation

3.1 *Introduction*

Although the original approaches of the variable block size motion estimation (VBS-ME) and the multiple reference frame motion estimation (MRF-ME) can produce coded video sequences with modified visual qualities and smaller compressed sizes, they require much computational effort that the coding process will be very time consuming. To solve this problem, it is necessary to relieve the effort by using some fast algorithms. This chapter proposes a new ME scheme included in two different parts to effectively speed up respectively the VBS-ME and the MRF-ME processes without downgrading the video quality.

The first method combines the merits of both the traditional PDS and the SAD reuse algorithm. It speeds up the VBS-ME process by reusing the incompletely computed SAD values obtained during the normal PDS processes of different block sizes so that no repeated computation of errors will happen. The second method lets the motion searches in different reference frames jointly uses a common tentative minimum of SAD, which can speed up the rejection of candidate blocks while the MRF-ME is proceeding. A number of redundant operations can be saved accordingly. Since these two methods are designed to be compatible with each other, they can be combined to form an effective motion estimation scheme for both VBS-ME and MRF-ME. It has been found that the combined scheme is faster than the conventional full search approach by 9.74 times on average without any objective quality degrading. This proves

that the proposed algorithm outperforms other fast approaches of lossless VBS-ME and MRF-ME like the PDS and the SAD reuse algorithm.

The details of the mechanism of this new ME scheme is presented in Section 3.2. In Section 3.3, the experimental results of this scheme and other approaches are shown, discussed and compared. Then a conclusion is drawn in Section 3.4.

3.2 *Proposed Scheme*

The proposed fast and lossless scheme of the VBS & MRF-ME is a combination of two newly developed approaches called “partial SAD reusing PDS” (PSADR-PDS) and “common tentative minimum PDS” (CTM-PDS). The PSADR-PDS is designed for advancing the VBS-ME while the CTM-PDS focuses on improving the MRF-ME. In this section, the theories and mechanisms of these two ME approaches are presented one by one.

3.2.1 Partial SAD reusing PDS for variable block size motion estimation

3.2.1.1 Background

In the conventional approach of the VBS-ME, ME is applied to each block size independently to find out the best matches of the blocks with different block sizes, and then the set of partitions within the current MB that minimizes the rate-distortion (RD) cost [17, 62] will be selected.

It is assumed that the SAD is used as the distortion measure for ME. If the PDS is used, the non-best matches for the current block are rejected once their PSAD values accumulate beyond a tentative minimum SAD, and therefore much of the processing time can be saved. On the other hand, since the blocks with different block sizes are overlapped with each other within a MB as shown in Figure 3-1, (for example a 16x16

block is composed of two 16x8 blocks, and the 16x8 blocks are also composed of other blocks with smaller sizes) the same contents of a MB will undergo MEs within the same search regions for seven times. Finally, the sum of pixel differences between the pixels of the current MB and the pixels in the reference frame is computed repeatedly for many times. It is extremely redundant. In other words, the speed of executing VBS-ME can be largely improved if these redundant operations are saved by reusing the previously computed data for the computation of distortion values in the remaining motion searches. This is the basic concept of the existing SAD reuse algorithm [51].

As discussed in Section 2.3.3.1, the first step of the SAD reuse algorithm is to conduct exhaustive searches for the 4x4 blocks to obtain the SAD values in all candidate positions in the reference frame. Then some pairs of these computed SAD values can be added up to produce a SAD for a 4x8 block and a SAD for an 8x4 block. Likewise, the SAD values of 4x8 blocks and 8x4 blocks can then be reused to produce the SAD values of other larger blocks. In the same sense, all the SAD values for different block sizes can be obtained without performing any further block matching. Since only 4x4 block size requires carrying out block matching, a large amount of computational operations is avoided. The following summarizes how a SAD value of a block is obtained by reusing the SAD values of two smaller blocks with just one addition operation:

$$\begin{cases} SAD_{4\times8}(i,m:0\le i\le 3)=SAD_{4\times4}(i,m)+SAD_{4\times4}(i+4,m) \\ SAD_{4\times8}(i,m:4\le i\le 7)=SAD_{4\times4}(i+4,m)+SAD_{4\times4}(i+8,m) \\ SAD_{8\times4}(i,m:0\le i\le 7)=SAD_{4\times4}(i\times2,m)+SAD_{4\times4}(i\times2+1,m) \\ SAD_{8\times8}(i,m:0\le i\le 3)=SAD_{4\times8}(i\times2,m)+SAD_{4\times8}(i\times2+1,m) \\ SAD_{8\times16}(i,m:0\le i\le 1)=SAD_{8\times8}(i,m)+SAD_{8\times8}(i+2,m) \\ SAD_{16\times8}(i,m:0\le i\le 1)=SAD_{8\times8}(i\times2,m)+SAD_{8\times8}(i\times2+1,m) \\ SAD_{16\times16}(m)=SAD_{16\times8}(0,m)+SAD_{16\times8}(1,m) \end{cases}, \quad \textbf{(3-1)}$$

where *i* is the index of a current block shown in Figure 3-1, and *m* is a position in the reference frame.

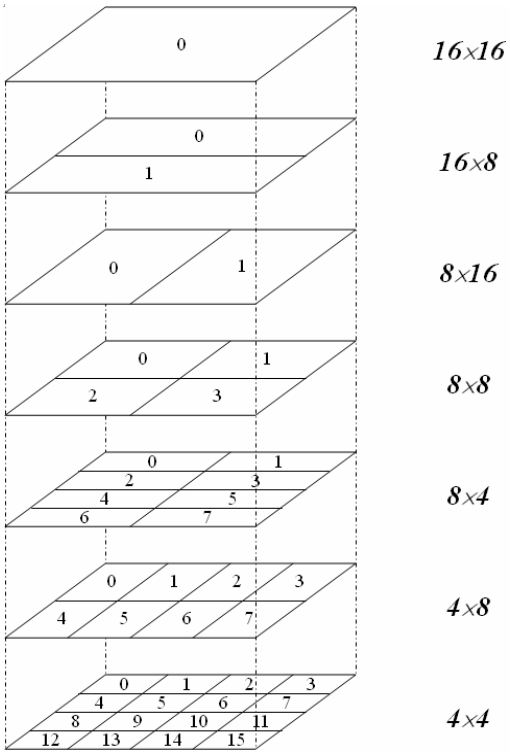


Figure 3-1. Blocks with different sizes are overlapped with each other within a MB

Though the SAD reuse algorithm is so powerful, however, it can only cooperate with exhaustive search algorithm but not the faster PDS approach. It is because when the PDS is performed, full SAD values will be computed only for the best matches but not all the candidate blocks. Just the PSAD values will be computed for other candidate blocks. Then the SAD values of larger blocks cannot be obtained by using the SAD reuse algorithm without the full SAD values of smaller blocks. To make the data reusing possible for the PDS, a method called “partial SAD reusing PDS” (PSADR-PDS) is proposed.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure 3-2. Sixteen non-overlapping 4x4 regions inside a MB

3.2.1.2 Theory of the Proposed Approach

In the PSADR-PDS, each 16x16 MB in current frame is carved up into sixteen non-overlapping 4x4 regions as shown in Figure 3-2. The row-based PDS is adopted as the block matching method because it is simple to be realized and is proved to be faster than the pixel-based PDS. The order of block sizes to be processed in the PSADR-PDS is

16x16 → 16x8 → 8x16 → 8x8 → 8x4 → 4x8 → 4x4.

During executing the PDS for a block, other than the SAD and PSAD values of the whole block, the SAD and PSAD values of each small 4x4 region covered by the current block are also stored up (For example, for the top 16x8 block within the MB, the '0'~'7' 4x4 regions are involved; for the bottom 16x8 block, the '8'~'15' 4x4 regions are involved). For each of these small 4x4 regions indexed by *i*, there is a buffer, $buf_i(m)$, to store the distortion values corresponding to different positions in the reference frame indicated by *m*. The stored buffer values can be reused for the subsequent block sizes to pre-compute the SAD and PSAD values in advance of performing PDS. Each pre-computed distortion value for a block is just the summation of relevant buffer values. For example, as shown in Figure 3-3, a distortion value of the top 16x8 block corresponding to a candidate MV, *m*, is the summation of the buffer values of regions '0'~'7' which are all corresponding to the same MV, *m*. The basic principle is very similar to that of the existing SAD reuse algorithm. It should be noted that a full SAD

value for the current block will be obtained if the buffer values used in the summation are all full SAD values. Otherwise, only a PSAD value will be resulted. After the pre-computed distortion values corresponding to different candidate MVs of the current block are obtained, the candidate positions will be checked one by one by comparing their pre-computed distortion values with the tentative minimum SAD. If it is found that some candidate positions have pre-computed distortion costs which are larger than or equal to the tentative minimum, these candidates can be rejected immediately, and further accumulation of the distortion values is not required. If not, the accumulation of the distortion values between the 4x4 regions of the current block and those candidate blocks will carry on until completion or till their partial values are larger than or equal to the tentative minimum. The mechanism of pre-computing each SAD or PSAD value for block of different sizes is concluded as follows:

- 16x8 or 8x16 block – summing up 8 corresponding buffer values
- 8x8 block – summing up 4 corresponding buffer values
- 8x4 or 4x8 block – summing up 2 corresponding buffer values
- 4x4 block – taking a corresponding buffer value

Since the 16x16 block size is the first block size to be processed, no reusing of buffer values is needed for it.

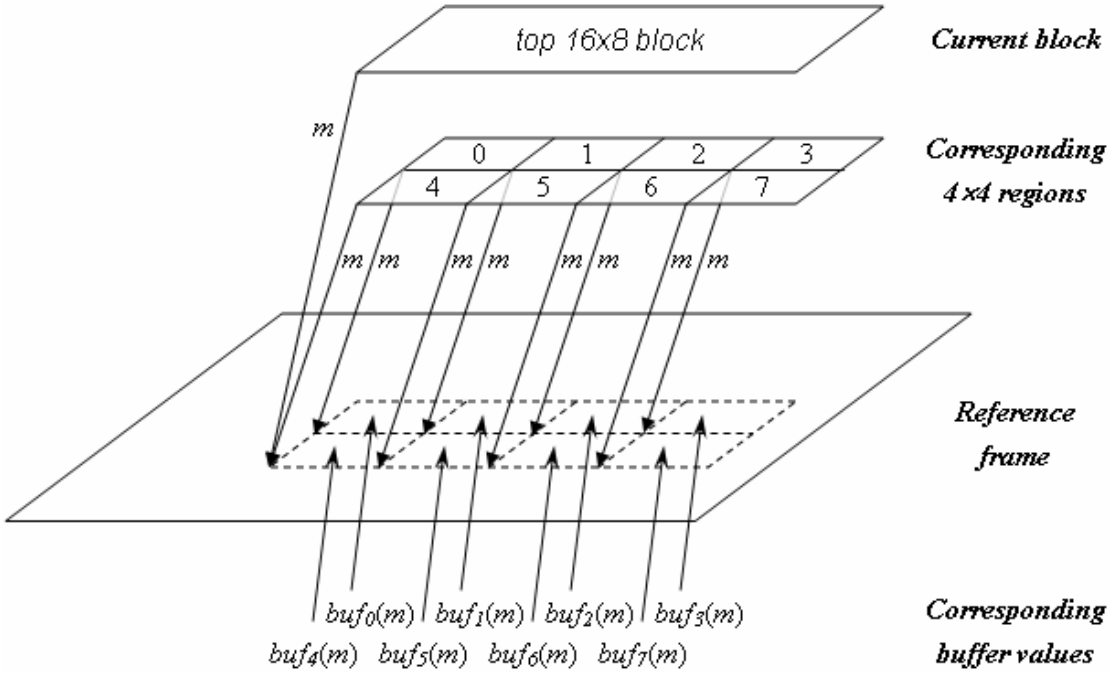


Figure 3-3. A distortion value of the top 16x8 block corresponding to a candidate MV, *m*, is the summation of the buffer values of regions '0'~'7' which are all corresponding to the same MV, *m*

To avoid repeatedly calculating the previously computed pixel differences, the system will record the matching status between the 4x4 regions of the current MB and various positions in the reference frame. For a specific position in the reference frame, matching status means which rows of pixels of a 4x4 region are matched with the pixels of the reference frame. By checking the matching status, the processed rows can be skipped when the 4x4 regions are matching with the matched regions in the reference frame again. Therefore, the advantage comes from the fact that the PDS for each block size except the 16x16 can start from an intermediate stage instead of the very beginning in which the distortions of all the pixels are needed to be computed.

The procedure of the proposed PSADR-PDS is summarized as follows:

1. Set all buffer values to be zero.
2. Let the current block size be 16x16, and jump to step 7.
3. Update buffer values, *buf*, with the SAD and PSAD obtained after PDS.
4. Update the matching statuses.

5. Change the current block size to the next.
6. Pre-compute the SAD and PSAD values for the current block size by using *buf*.
7. Perform PDS for the current block size. For each candidate position, start the accumulation of distortion value from the pre-computed value, and skip the matched rows of pixels according to the matching status.
8. If the current block size is 4x4, end the process. Otherwise, jump to step 3.

Figure 3-4 is a flow chart showing the procedure of the PSADR-PDS.

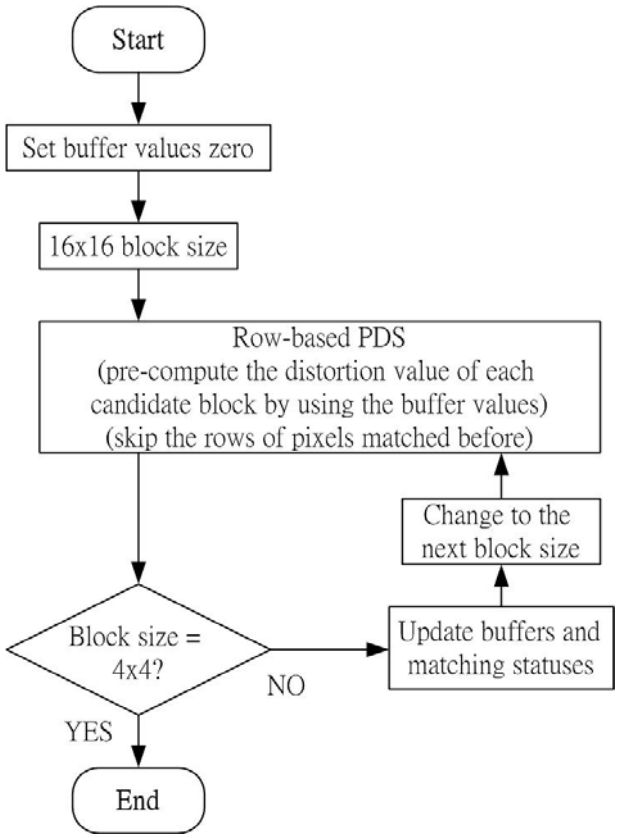


Figure 3-4. Flow chart of PSADR-PDS

3.2.1.3 Further Analysis

In fact the customary bottom-up order (4x4 → 4x8 → 8x4 → 8x8 → 8x16 → 16x8 → 16x16) can be used, but it is incompatible with the operating procedures of the row-based PDS. Since each small block undergoes the PDS independently, the

processed rows of pixels inside different small blocks will differ from each other; thus the pixels in some rows inside each larger block which is composed of two smaller blocks may be midway processed as illustrated in the example in Figure 3-5. In this figure, the processed pixels are shadowed. Assuming that the next block size to be processed in this example is 8x4, it is found that the row-based PDS is difficult to be applied to the 8x4 blocks. Likewise, it is also difficult for the row-based PDS to proceed for other block sizes. Inversely, by using the suggested top-down order, the row-based PDS can be applied without any problem to all block sizes.

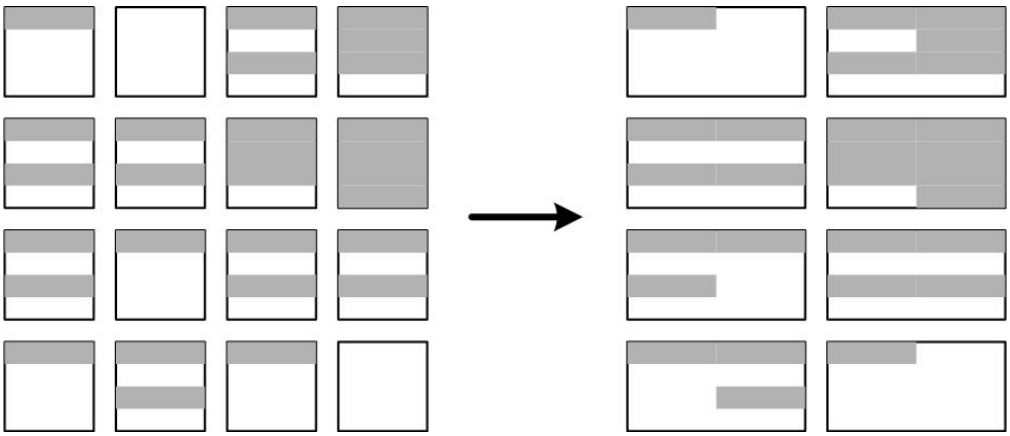


Figure 3-5. A MB containing incompletely processed rows of pixels (the processed pixels are shadowed). Assuming that the next block size is 8x4, it is difficult for the row-based PDS to be applied to the 8x4 blocks

In the traditional row-based PDS, the processing order of the rows of pixels is from the top row to the bottom row inside the current block. Figure 3-6 shows this traditional top-down order for a 16x16 block. However, this order does not suit the proposed PSADR-PDS as it lets the PDS bias specific contents of the current block. By investigating the average spatial distribution of pixel matching errors [38], it is found that distortion errors with similar magnitude tend to appear together in a cluster. The effect of this phenomenon can be described by considering two different cases:

1) While matching the current block with a non-best match, if the error for the first row of pixels is very small, the errors for a few more rows below are likely to be small too. Therefore, if the top-down processing order is used, the rejection of non-best matches requires many operations. This is because the partial distortion values of the non-best matches requires combining a lot of small pixel errors to become a sufficiently large values which are larger than or equal to the tentative minimum SAD. In other words, it is needed to process many pixels before rejecting a non-best match.
2) Conversely, if the error for the first row is very large, it is highly probable that the errors for the subsequent several rows are also very large. Therefore, the rejection of non-best matches would happen much earlier in this case.

As the variation of the speed of performing the row-based PDS will be very large, it is highly unfavorable to apply the top-down processing order and the row-based PDS at the same time.

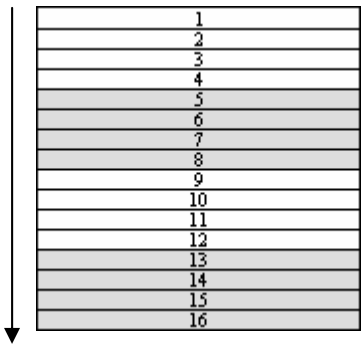
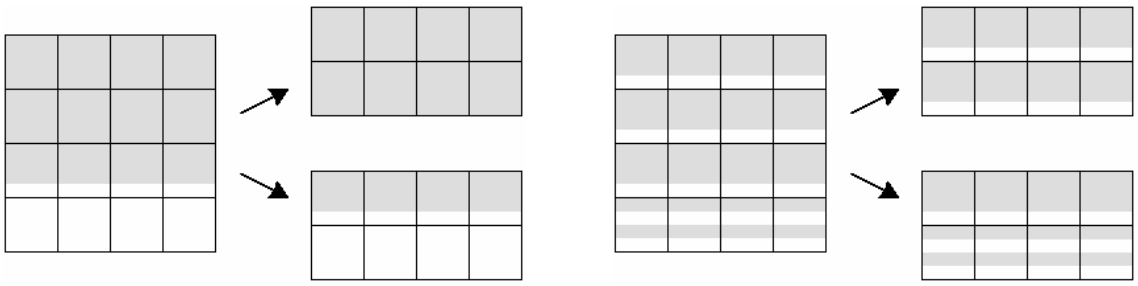


Figure 3-6. Traditional top-down processing order of row of pixels of the PDS for a 16x16 block

On the other hand, the top-down processing order bias towards processing the upper part of a block firstly and the lower part is despised. As a result, more pixels will be processed in the upper part than in the lower part while matching with non-best matches. Since the proposed PSADR-PDS makes use of the pixel errors to pre-compute

the distortion values for small block sizes, when the errors for these processed pixels are reused for two vertically separated small blocks like the 16x8 blocks in Figure 3-7(a), it is likely that more pixel errors will be shared with the upper block than the lower block as more processed pixels are inside the upper block. Consequently, it is probable that the pre-computed distortion values for the lower block are smaller than the tentative minimum SAD so that operations are needed to further accumulate the distortion values. Differently, some pixel errors reused by the upper block may be wasteful since the sufficiently large pre-computed distortion values may be obtained by reusing fewer pixel errors. Due to the unfair distribution of pixel errors, the reduction of redundant operations is far from the optimal.



(a) (b)

Figure 3-7. (a) If top-down processing order of the PDS is used, the upper half of the block will have more computed pixel errors (in gray). (b) If regular processing order of the PDS is used, both halves of the block will have similar number of computed pixel errors

To make the row-based PDS more efficient in the proposed PSADR-PDS approach, and to make the reusing scheme of computed distortion values more effective, the traditional top-down processing order of the row-based PDS is replaced by a dispersed processing order as shown in Figure 3-8. By using this proposed order, one row of pixels of the current block will be considered for every four rows in each vertical

scan. It operates in a dispersed manner so that it will not focus either on a high error region or a low error region. Furthermore, it makes the progresses of accumulating the distortion values for different 4x4 regions of the current MB almost even. Almost the same number of pixel errors will be distributed to the upper and the lower halves of each large block, and thus the over-processing of the upper half block and the under-processing of the lower half block will not be boosted. A large amount of unnecessary operations can be avoided. Figure 3-7(a) shows that if 11 rows of pixels of the large block were processed by using the traditional top-down order, 8 of them will be contained in the upper half block but the lower half block has only 3 of them. The difference is 5 rows. In contrast to Figure 3-7(b) in which the dispersed processing order is used, the upper half block has 6 processed rows this time while the remaining 5 rows belong to the lower half block. The difference is just one row. For the traditional top-down order, the maximum value of the difference in the number of processed rows of the upper and the lower halves of a block is 8; as for the dispersed processing order, the maximum value is 2 which is much smaller. Table 3-1 shows a comparison of the total number of operations (additions + subtractions + comparisons + taking absolute values) in billion for performing the proposed PSADR-PDS for the integer-pixel VBS and MRF-ME between using the traditional top-down processing order and using the dispersed processing order. The results proved that the proposed dispersed processing order is more effective than the top-down order. The average speedup brought by the proposed order is 1.42 time.

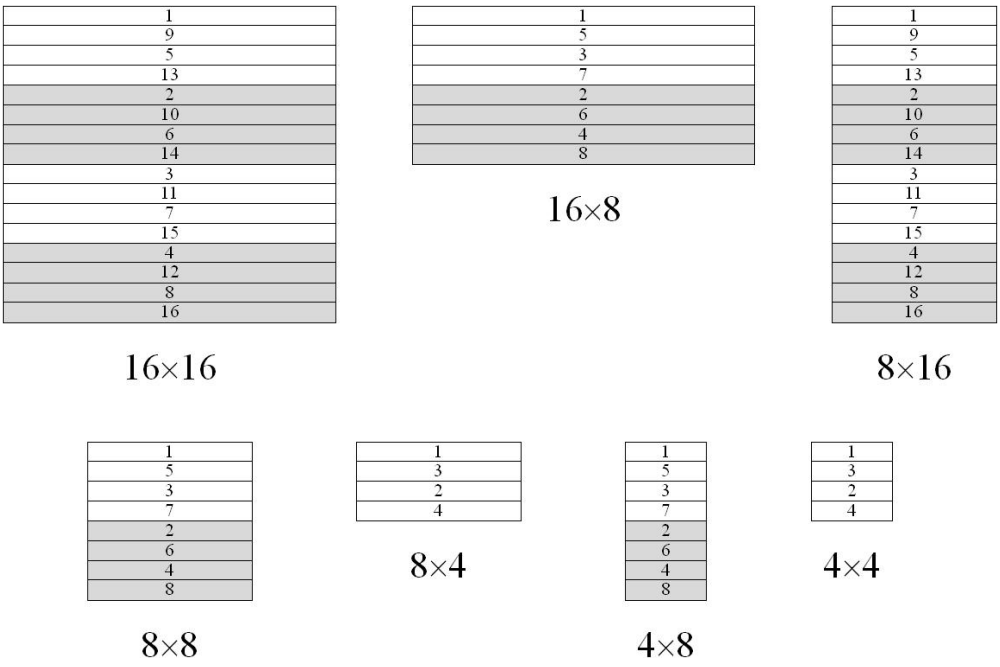


Figure 3-8. Proposed dispersed processing order of the row-based PDS for seven block sizes

Table 3-1. Total number of operations (additions + subtractions + comparisons + taking absolute values) in billion for performing the PSADR-PDS for integer ME between using the top-down processing order and using the dispersed processing order

Sequence	Top-down Order	Dispersed Order	Speedup
Car phone	43.19	30.95	1.40
Claire	35.72	26.73	1.34
Coastguard	59.38	42.02	1.41
Container	47.40	33.09	1.43
Erik	42.05	29.50	1.43
Foreman	47.28	32.29	1.46
Grandmother	45.88	31.36	1.46
Hall monitor	38.80	27.68	1.40
Miss America	38.70	30.41	1.27
Mother daughter	42.47	29.10	1.46
News	44.25	29.57	1.50
Salesman	48.33	30.79	1.57
Silent	47.68	31.15	1.53
Suzie	49.19	35.62	1.38
Table tennis	57.29	44.03	1.30
Trevor	48.58	35.21	1.38
Average speedup:			1.42

3.2.2 Common tentative minimum PDS for multiple reference frames motion estimation

3.2.2.1 Background

In the principle of the MRF-ME, each block of a current frame will search for its motion compensated block from a set of decoded frames. To perform the MRF-ME, the

conventional way is to firstly conduct individual ME for the current block in each reference frame. It is assumed that the SAD is used as the distortion measure. The best match in each reference frame should be the block with the minimum RD cost [17, 62] as shown below:

$$J_{motion}(m,r) = SAD(m,r) + \lambda_{motion} \cdot Rate_{motion}(m), \quad \textbf{(3-2)}$$

where m is a candidate MV, r is the reference frame index of the reference frame in which the candidate block is, λ_{motion} is a Lagrangian multiplier, and $Rate_{motion}(m)$ is the approximated number of bits for coding the candidate MV, m.

In the RD optimized ME scheme, J_{motion} is used in place of SAD value as the matching criterion for checking whether the candidate blocks in reference frames should be rejected. If the PDS is applied, partial J_{motion} (PJ_{motion}) values stated below will be outputted during the block matching:

$$PJ_{motion}(m,r) = PSAD(m,r) + \lambda_{motion} \cdot Rate_{motion}(m), \quad \textbf{(3-3)}$$

where PJ_{motion} is the sum of a PSAD value and a weighed rate term. A candidate block can be rejected once its PJ_{motion} cost stores up beyond the tentative minimum of the J_{motion} costs of searched candidate blocks, and then the incomplete computation of the SAD value for this candidate block can be stopped. As a result, a number of redundant operations are saved.

The basic way to implement the PDS algorithm in the MRF-ME is to carry out individual PDS for the current block in each reference frame. After the best matching blocks of the current block in different reference frames are found, the next step is to decide which one of these candidates should be used for motion compensation. This step is known as reference frame selection since it is equal to selecting the reference frame for the current block from a number of possible frames. Normally the blocks in the current frame to be encoded can refer to distinct reference frames, but in the H.264

reference model [51], the sub-blocks within the same 8x8 region in the current frame must use the same reference frame. Therefore, the ways to decide the reference frames are different for two different groups of block sizes.

1) For each 16x16, 16x8, 8x16 or 8x8 block, the reference frame is decided by comparing the RD costs of different reference frames stated in eqn. (3-4), and then the reference frame corresponding to the minimum RD cost is selected.

$$J_{ref}(r) = J_{motion\_best}(r) + \lambda_{motion} \cdot Rate_{ref}(r), \quad \textbf{(3-4)}$$

where $J_{motion\_best}(r)$ is the J_{motion} cost of the best match found from reference frame r, and $Rate_{ref}(r)$ is the cost of the reference frame index r. It should be noted that $Rate_{ref}(r)$ will become larger as r increases.

2) As for the 8x4, 4x8 or 4x4 blocks inside an 8x8 region, their reference frames are identical, and are decided by comparing the RD costs of different reference frames stated in eqn. (3-5). The combination of the best matches of these sub-blocks with the minimum RD cost will be chosen as the motion compensated blocks.

$$J_{ref\_8\times8region}(r) = \sum_{block=0}^{p-1}\left[J_{motion\_best}(block, r)\right] + \lambda_{motion} \cdot Rate_{ref}(r), \quad \textbf{(3-5)}$$

where $J_{motion\_best}(block,r)$ is the J_{motion} cost of the best match of a sub-block indexed by *block* found from reference frame r. p is the number of sub-blocks inside the 8x8 region. It should be 2 for the 8x4 or 4x8 block size, and should be 4 for the 4x4 block size.

If the MRF-ME is performed by carrying out individual PDS in each reference frame, the candidates in each reference frame can compare only with the searched candidates in the same reference frame for judging whether they should be rejected, and so the individual best match in each reference frame must be determined. However, MRF-ME aims only at choosing one best match among all candidates in all reference frames for each current block. This means that as long as the best one among all

candidate blocks is found, all the remaining candidates including those individual best matches in other reference frames can be rejected without completely compute their costs. It is redundant and unnecessary to determine the best matches in all reference frames, and also fully compute their costs.

3.2.2.2 Theory of the proposed approach

To have a better approach of applying the PDS in the MRF-ME, a new method called “common tentative minimum PDS” (CTM-PDS) is proposed. It lets the candidates in any reference frame compare with the searched candidates in any reference frame. While performing the PDS for a current block, a buffer will store and update the minimum value of the costs of searched candidate blocks among all reference frames, and then the partial costs or full costs of the remaining unsearched candidate blocks will be compared with the stored temporary minimum value. The candidates having partial costs which are larger than or equal to the stored minimum cost will be rejected immediately. By adopting the common tentative minimum, the determination of the motion compensated block for the current block no longer requires two stages as described by eqn. (3-2) ~ (3-5). It can be done in just one stage.

In the previous approach which conducts the PDS in each reference frame individually, the RD cost and partial RD cost to be computed are defined in eqn. (3-2) and (3-3) respectively. Using this definition, the cost of each candidate block cannot compare with the costs of candidate blocks in other reference frame as they do not include the costs of reference frame indices. Therefore, the proposed CTM-PDS must adopt another definition of the cost function so as to make the candidates in different reference frames comparable.

1) For 16x16, 16x8, 8x16 or 8x8 block size

For each 16x16, 16x8, 8x16 or 8x8 block, the RD cost and partial RD cost of each of its candidate block are defined respectively as

$$J_{blk}(m,r) = J_{motion}(m,r) + \lambda_{motion} \cdot Rate_{ref}(r) \text{ and} \quad \textbf{(3-6)}$$

$$PJ_{blk}(m,r) = PJ_{motion}(m,r) + \lambda_{motion} \cdot Rate_{ref}(r). \quad \textbf{(3-7)}$$

where the J_{motion} and PJ_{motion} terms are the same as those stated in eqn. (3-2) and (3-3) respectively. The new formulations also include the $Rate_{ref}$ term which varies for different reference frames so the comparison among the candidate block in different reference frames is now possible.

Figure 3-9 is an example showing the difference in performance of carrying out MRF-ME by applying the original PDS and by applying the proposed CTM-PDS. In this example, three reference frames are used, and the candidates in different reference frames indexed by r are organized side by side along the "candidates"-axis in a searching order. The J_{blk} costs of the candidate blocks are plotted by using the solid line in black. The dotted lines in red indicate the tentative minimum cost stored in buffer. For those candidates corresponding to that the solid lines are higher than the dotted red lines in the figure, they can be rejected before their J_{blk} costs are fully computed. These candidates are marked in deep grey above the "candidates"-axis, while the others are marked in tinged grey. The shadowed areas represent the difference between the full J_{blk} cost and the partial J_{blk} cost of each candidate required for judging whether the candidate can be rejected. For candidates with a larger shadowed area, generally, more operations can be saved by the proposed approach for computing their costs. According to the results, the shadowed area becomes larger if CTM-PDS is applied instead of the conventional PDS approach. Therefore, using CTM-PDS can turn out a smaller number of operations and more candidates rejected before their J_{blk} costs are fully computed.

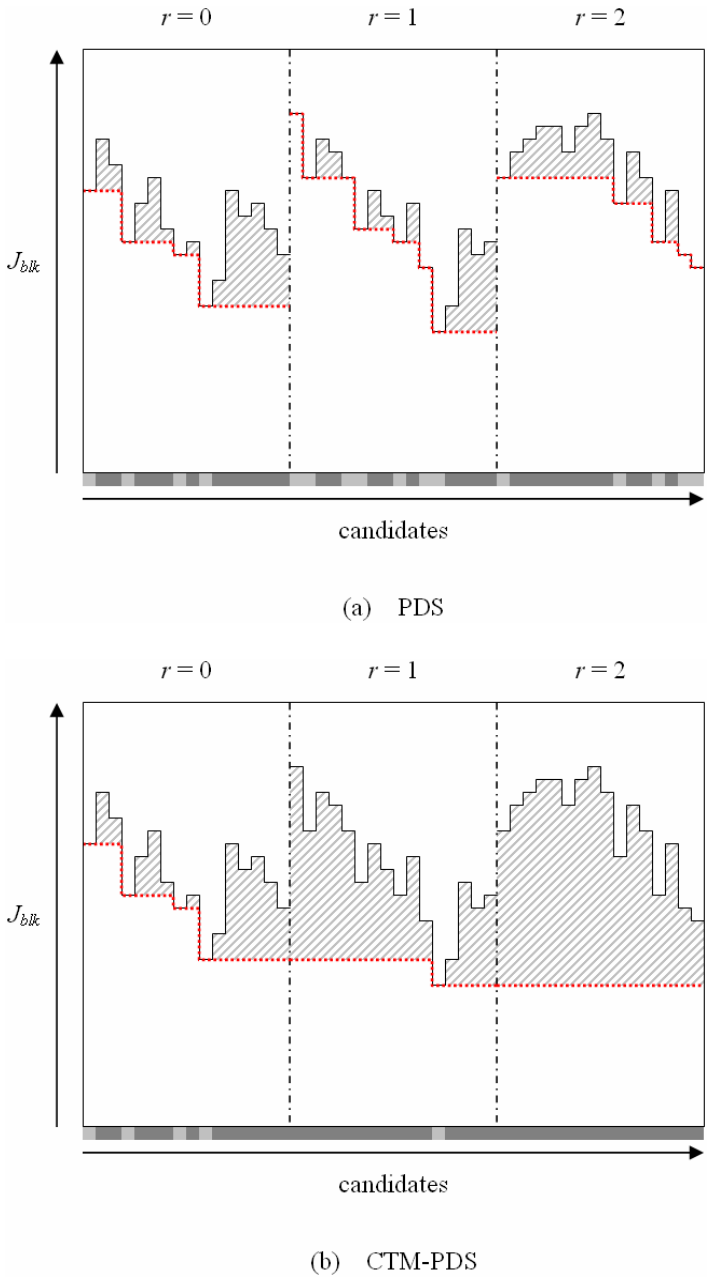


Figure 3-9. Examples of MRF-ME: (a) Original PDS vs (b) CTM-PDS

2) For 8x4, 4x8 or 4x4 block size

For 8x4, 4x8 or 4x4 block size, the motion compensated blocks of the sub-blocks within the same 8x8 region must come from the same reference frame. That is, a reference frame should be selected for each 8x8 region in place of each sub-block. In this case, the common tentative minimum is still applicable, but it should be defined as

the cost of a combination of candidate matches of all the sub-blocks, and this cost must be the minimum value among the costs of all the examined combinations in different reference frames. It should be noted that the candidate matches in each combination must come from the same reference frame.

In practice, the motion search of all the sub-blocks within an 8x8 region rather than of each individual sub-block will be conducted in each reference frame one by one. This means that the reference frame will not be changed before all the sub-blocks have searched all their candidate matches in a reference frame. For a reference frame, the PDS of each sub-block will be conducted one by one. During the PDS of a sub-block conducted in a reference frame, if the common tentative minimum is not yet formed, the candidate blocks will be rejected once their J_{motion} costs or partial J_{motion} costs are larger than or equal to the minimum of the J_{motion} costs of the searched candidates. Then after the common tentative minimum is formed, the candidate blocks will be rejected when the full RD values or partial RD values defined respectively in eqn. (3-8) and (3-9) are larger than or equal to the common tentative minimum. The minimum $J_{8\times8region}$ cost that corresponds to a considered set of best matches of all sub-blocks will be stored as the common tentative minimum.

$$J_{8\times8region}(m,r\mid i)=J_{motion}(m,r\mid i)+\sum_{block}\left[J_{motion\_best}(block,r)\right]+\lambda_{motion}\cdot Rate_{ref}(r) \quad \textbf{(3-8)}$$

$$PJ_{8\times8region}(m,r\mid i)=PJ_{motion}(m,r\mid i)+\sum_{block}\left[J_{motion\_best}(block,r)\right]+\lambda_{motion}\cdot Rate_{ref}(r), \quad \textbf{(3-9)}$$

where i is the index of the current sub-block, $J_{motion}(m,r|i)$ and $PJ_{motion}(m,r|i)$ are respectively the J_{motion} cost and partial J_{motion} cost for the current sub-block indexed by i, and *block* indicates the index of other sub-blocks which have found their best matches in reference frame r.

In case all the candidate blocks of the current sub-block for a specific reference frame are rejected, this means that the RD cost of any combination of the best matches of sub-blocks found from this reference frame must not smaller than the common tentative minimum, so it is not necessary to carry out the PDS for the remaining sub-blocks in this reference frame, and the whole reference frame can be rejected. As a result, many unworthy operations can be saved.

Furthermore, the CTM-PDS can be combined perfectly with the PSADR-PDS to form a very effective VBS and MRF-ME scheme. This can be realized by just replacing the “row-based PDS” block in Figure 3-4 by the “row-based CTM-PDS”.

3.3 *Experimental results*

To compare the performance of the proposed approaches with that of other algorithms, 16 QCIF video sequences are encoded by using the H.264/AVC verification model JM9.0 [51] embedded with different ME approaches. Each video sequence contains 150 frames, and was encoded in IPPP… format. The search range was limited to ±16 pixels, and the quantization parameter was set to 30. All the 7 various block sizes, 5 reference frames as well as intra prediction modes were enabled. The predicted MV of the 16x16 block is used as the predicted MVs of other block sizes for each MB. In addition, sub-pixel ME was excluded from the coding processes.

Table 3-2 is a speed comparison among the exhaustive full search (FS), the traditional PDS, the SAD reuse, the proposed PSADR-PDS, the CTM-PDS and the combined scheme of PSADR-PDS and CTM-PDS in terms of the total number of operations (additions + subtractions + comparisons + taking absolute values) in billion for performing integer VBS and MRF-ME. The speedup ratios of each scheme are obtained by comparing its total numbers of operations with those of the FS scheme.

Table 3-3 shows the speed comparison in terms of actual ME time. The detailed number of operations required to encode different video sequences by applying different ME methods can be found in Tables 3-4 ~ 3-9.

According to the results in Table 3-2, it can be found that the proposed PSADR-PDS approach can bring a speedup of 8.08 times on average as compared with FS, and also the proposed CTM-PDS approach has an average speedup of 4.38 times. If the combined scheme of the PSADR-PDS and CTM-PDS is used, a larger average speedup of 8.86 times can be obtained. In contrast, the traditional PDS approach and the SAD reuse algorithm can only give 3.63 and 3.99 times of average speed improvement respectively. The two proposed approaches and the combined scheme have faster speeds than other algorithms.

According to the speed comparison in terms of actual ME time in Table 3-3, the SAD reuse algorithm has an average speedup ratio of 8.73 which is much higher than the theoretical result which is only 3.63 times. This is because the SAD reuse algorithm requires very few comparing operations, but requires more additions, subtractions and operations for taking absolute values than all other fast ME approaches. Since the actual time duration of a comparing operation is longer than that of a addition/subtraction or a operation for taking a absolute value, the speedup in actual time contributed by the reduction of the number of comparing operations is much higher than the slow-down contributed by the increase of the same number of additions/subtractions or operations for taking absolute values.

Although the SAD reuse algorithm is very fast, the proposed PSADR-PDS is still faster in terms of actual ME time. The PSADR-PDS approach has an average speedup of 10.04 times. In terms of actual time, the average speedup ratio of the CTM-PDS is 6.90, and that of the combined scheme of the two proposed approaches is 9.74.

Although the average speedup of the combined scheme is a little bit lower than that of using the PSADR-PDS individually, the combined scheme is the fastest approach for a certain number of video sequences. Therefore, the average speed of performing ME will be higher by using the proposed PSADR-PDS or the combined scheme of the PSADR-PDS and CTM-PDS.

Since the proposed approaches are lossless methods, encoding video sequences by using them will turn out the same PSNR ratios and output bit-rates. In other words, no quality degradation is introduced by adopting the proposed ME schemes.

Table 3-2. Total number of operations (additions + subtractions + comparisons + taking absolute values) in billion for performing integer ME by using schemes including exhaustive FS, traditional PDS, SAD reuse, PSADR-PDS, CTM-PDS and the combined scheme of PSADR-PDS + CTM-PDS

Sequence	FS		PDS		SAD reuse	
	# operations in billion	speedup ratio	# operations in billion	speedup ratio	# operations in billion	speedup ratio
Car phone	260.66	1.00	67.60	3.86	65.74	3.96
Claire	182.86	1.00	42.73	4.28	66.22	2.76
Coastguard	363.25	1.00	134.40	2.70	65.33	5.56
Container	258.99	1.00	72.54	3.57	65.77	3.94
Erik	251.63	1.00	65.08	3.87	65.38	3.85
Foreman	307.04	1.00	84.63	3.63	65.44	4.69
Grandmother	238.16	1.00	67.06	3.55	65.75	3.62
Hall monitor	242.72	1.00	49.52	4.90	65.37	3.71
Miss America	173.21	1.00	49.14	3.52	67.16	2.58
Mother & daughter	238.19	1.00	64.07	3.72	65.63	3.63
News	256.67	1.00	65.23	3.93	65.58	3.91
Salesman	284.76	1.00	72.41	3.93	65.04	4.38
Silent	288.10	1.00	73.43	3.92	65.08	4.43
Suzie	262.40	1.00	84.39	3.11	65.96	3.98
Table tennis	303.13	1.00	122.90	2.47	66.19	4.58
Trevor	277.34	1.00	86.82	3.19	65.90	4.21
Average speedup:		1.00		3.63		3.99

Sequence	PSADR-PDS		CTM-PDS		PSADR + CTM-PDS	
	# operations in billion	speedup ratio	# operations in billion	speedup ratio	# operations in billion	speedup ratio
Car phone	30.95	8.42	54.51	4.78	27.82	9.37
Claire	26.73	6.84	35.64	5.13	24.53	7.46
Coastguard	42.02	8.65	103.75	3.50	36.54	9.94
Container	33.09	7.83	64.22	4.03	31.46	8.23
Erik	29.50	8.53	52.97	4.75	27.06	9.30
Foreman	32.29	9.51	68.04	4.51	28.74	10.68
Grandmother	31.36	7.59	59.52	4.00	29.51	8.07
Hall monitor	27.68	8.77	42.70	5.68	26.05	9.32
Miss America	30.41	5.70	40.47	4.28	27.30	6.34
Mother & daughter	29.10	8.19	54.20	4.39	26.75	8.90
News	29.57	8.68	55.25	4.65	27.27	9.41
Salesman	30.79	9.25	63.36	4.49	29.12	9.78
Silent	31.15	9.25	61.04	4.72	28.56	10.09
Suzie	35.62	7.37	65.45	4.01	31.55	8.32
Table tennis	44.03	6.89	97.86	3.10	39.06	7.76
Trevor	35.21	7.88	68.35	4.06	31.32	8.85
Average speedup:		8.08		4.38		8.86

Table 3-3. Total time for performing integer ME by using schemes including exhaustive FS, traditional PDS, SAD reuse, PSADR-PDS, CTM-PDS and the combined scheme of PSADR-PDS + CTM-PDS

Sequence	FS		PDS		SAD reuse	
	ME time (sec)	speedup ratio	ME time (sec)	speedup ratio	ME time (sec)	speedup ratio
Car phone	777.17	1.00	196.94	3.95	73.08	10.64
Claire	778.46	1.00	172.64	4.51	81.03	9.61
Coastguard	763.92	1.00	272.81	2.80	85.91	8.89
Container	760.81	1.00	201.67	3.77	87.89	8.66
Erik	742.99	1.00	191.81	3.87	84.47	8.80
Foreman	728.86	1.00	213.39	3.42	84.56	8.62
Grandmother	717.92	1.00	195.61	3.67	89.97	7.98
Hall monitor	725.69	1.00	174.63	4.16	86.46	8.39
Miss America	734.18	1.00	182.16	4.03	89.86	8.17
Mother & daughter	745.29	1.00	191.66	3.89	88.29	8.44
News	762.29	1.00	191.37	3.98	87.29	8.73
Salesman	759.62	1.00	198.26	3.83	86.20	8.81
Silent	761.73	1.00	196.70	3.87	87.30	8.73
Suzie	767.18	1.00	218.09	3.52	90.48	8.48
Table tennis	777.04	1.00	270.81	2.87	93.60	8.30
Trevor	770.39	1.00	217.49	3.54	91.62	8.41
Average speedup:		1.00		3.73		8.73

Sequence	PSADR-PDS		CTM-PDS		PSADR + CTM-PDS	
	ME time (sec)	speedup ratio	ME time (sec)	speedup ratio	ME time (sec)	speedup ratio
Car phone	65.87	11.80	83.21	9.34	74.23	10.47
Claire	58.72	13.26	63.46	12.27	65.69	11.85
Coastguard	86.52	8.83	177.27	4.31	93.02	8.21
Container	74.12	10.26	121.10	6.28	81.79	9.30
Erik	70.16	10.60	104.56	7.11	73.99	10.04
Foreman	78.04	9.34	132.00	5.52	76.35	9.55
Grandmother	75.21	9.55	116.97	6.14	80.37	8.93
Hall monitor	70.31	10.32	90.75	8.00	75.04	9.67
Miss America	71.48	10.27	83.34	8.81	73.06	10.05
Mother & daughter	71.64	10.40	106.13	7.02	72.84	10.23
News	73.66	10.35	109.42	6.97	72.84	10.47
Salesman	77.20	9.84	124.92	6.08	76.72	9.90
Silent	77.21	9.87	120.85	6.30	76.14	10.00
Suzie	86.79	8.84	128.16	5.99	81.89	9.37
Table tennis	101.08	7.69	179.59	4.33	96.49	8.05
Trevor	81.80	9.42	131.09	5.88	78.70	9.79
Average speedup:		10.04		6.90		9.74

Table 3-4. Detailed number of operations in billion required to carry out FS

Sequence	+ / –	Taking absolute value	Comparison	Total	Speedup ratio
Car phone	170.86	85.43	4.36	260.66	1.00
Claire	119.33	59.67	3.86	182.86	1.00
Coastguard	238.60	119.30	5.35	363.25	1.00
Container	169.82	84.91	4.26	258.99	1.00
Erik	164.90	82.45	4.28	251.63	1.00
Foreman	201.52	100.76	4.76	307.04	1.00
Grandmother	155.98	77.99	4.19	238.16	1.00
Hall monitor	159.07	79.54	4.11	242.72	1.00
Miss America	112.92	56.46	3.83	173.21	1.00
Mother & daughter	156.00	78.00	4.20	238.19	1.00
News	168.25	84.12	4.30	256.67	1.00
Salesman	186.88	93.44	4.44	284.76	1.00
Silent	189.07	94.54	4.49	288.10	1.00
Suzie	171.99	85.99	4.42	262.40	1.00
Table tennis	198.94	99.47	4.72	303.13	1.00
Trevor	181.87	90.93	4.54	277.34	1.00
Average speedup:					1.00

Table 3-5. Detailed number of operations in billion required to carry out PDS

Sequence	+ / –	Taking absolute value	Comparison	Total	Speedup ratio
Car phone	40.71	20.35	6.54	67.60	3.86
Claire	25.09	12.55	5.09	42.73	4.28
Coastguard	82.75	41.38	10.27	134.40	2.70
Container	44.03	22.02	6.49	72.54	3.57
Erik	39.13	19.57	6.38	65.08	3.87
Foreman	51.31	25.65	7.67	84.63	3.63
Grandmother	40.55	20.27	6.25	67.06	3.55
Hall monitor	29.28	14.64	5.60	49.52	4.90
Miss America	29.29	14.65	5.20	49.14	3.52
Mother & daughter	38.46	19.38	6.24	64.07	3.72
News	39.22	19.61	6.39	65.23	3.93
Salesman	43.75	21.87	6.78	72.41	3.93
Silent	44.35	22.17	6.91	73.43	3.92
Suzie	51.49	25.75	7.15	84.39	3.11
Table tennis	76.01	38.00	8.88	122.90	2.47
Trevor	52.91	26.46	7.45	86.82	3.19
Average speedup:					3.63

Table 3-6. Detailed number of operations in billion required to carry out SAD reuse algorithm

Sequence	+ / –	Taking absolute value	Comparison	Total	Speedup ratio
Car phone	41.75	20.29	3.71	65.74	3.96
Claire	41.99	20.29	3.95	66.22	2.76
Coastguard	41.54	20.29	3.50	65.33	5.56
Container	41.76	20.29	3.72	65.77	3.94
Erik	41.56	20.29	3.53	65.38	3.85
Foreman	41.59	20.29	3.56	65.44	4.69
Grandmother	41.75	20.29	3.71	65.75	3.62
Hall monitor	41.56	20.29	3.52	65.37	3.71
Miss America	42.45	20.29	4.42	67.16	2.58
Mother & daughter	41.69	20.29	3.65	65.63	3.63
News	41.67	20.29	3.63	65.58	3.91
Salesman	41.39	20.29	3.36	65.04	4.38
Silent	41.41	20.29	3.38	65.08	4.43
Suzie	41.85	20.29	3.82	65.96	3.98
Table tennis	41.97	20.29	3.94	66.19	4.58
Trevor	41.82	20.29	3.79	65.90	4.21
Average speedup:					3.99

Table 3-7. Detailed number of operations in billion required to carry out PSADR-PDS

Sequence	+ / –	Taking absolute value	Comparison	Total	Speedup ratio
Car phone	21.11	6.00	3.84	30.95	8.42
Claire	18.28	4.74	3.70	26.73	6.84
Coastguard	28.60	9.33	4.08	42.02	8.65
Container	22.58	6.65	3.86	33.09	7.83
Erik	20.14	5.57	3.79	29.50	8.53
Foreman	21.99	6.40	3.90	32.29	9.51
Grandmother	21.43	6.14	3.79	31.36	7.59
Hall monitor	18.90	5.02	3.75	27.68	8.77
Miss America	20.80	5.86	3.75	30.41	5.70
Mother & daughter	19.88	5.46	3.76	29.10	8.19
News	20.18	5.59	3.81	29.57	8.68
Salesman	21.01	5.96	3.81	30.79	9.25
Silent	21.25	6.07	3.83	31.15	9.25
Suzie	24.30	7.42	3.90	35.62	7.37
Table tennis	29.99	9.95	4.09	44.03	6.89
Trevor	23.99	7.28	3.93	35.21	7.88
Average speedup:					8.08

Table 3-8. Detailed number of operations in billion required to carry out CTM-PDS

Sequence	+ / –	Taking absolute value	Comparison	Total	Speedup ratio
Car phone	32.53	16.26	5.73	54.51	4.78
Claire	20.69	10.34	4.62	35.65	5.13
Coastguard	63.53	31.76	8.46	103.75	3.50
Container	38.81	19.41	6.00	64.22	4.03
Erik	31.53	15.77	5.67	52.97	4.75
Foreman	40.96	20.48	6.61	68.04	4.51
Grandmother	35.81	17.90	5.81	59.52	4.00
Hall monitor	24.99	12.49	5.22	42.70	5.68
Miss America	23.83	11.91	4.72	40.47	4.28
Mother & daughter	32.38	16.19	5.64	54.20	4.39
News	32.96	16.48	5.81	55.25	4.65
Salesman	38.07	19.04	6.25	63.36	4.49
Silent	36.57	18.28	6.19	61.04	4.72
Suzie	39.58	19.79	6.08	65.45	4.01
Table tennis	60.23	30.11	7.51	97.86	3.10
Trevor	41.34	20.67	6.35	68.35	4.06
Average speedup:					4.38

Table 3-9. Detailed number of operations in billion required to carry out PSADR + CTM-PDS

Sequence	+ / –	Taking absolute value	Comparison	Total	Speedup ratio
Car phone	19.03	5.16	3.62	27.82	9.37
Claire	16.83	4.18	3.52	24.53	7.46
Coastguard	24.92	7.79	3.83	36.54	9.94
Container	21.49	6.20	3.77	31.46	8.23
Erik	18.51	4.89	3.66	27.06	9.30
Foreman	19.64	5.44	3.67	28.74	10.68
Grandmother	20.19	5.62	3.70	29.51	8.07
Hall monitor	17.82	4.56	3.68	26.05	9.32
Miss America	18.72	5.00	3.58	27.30	6.34
Mother & daughter	18.32	4.81	3.63	26.75	8.90
News	18.64	4.95	3.67	27.27	9.41
Salesman	19.90	5.48	3.74	29.12	9.78
Silent	19.52	5.33	3.71	28.56	10.09
Suzie	21.58	6.29	3.68	31.55	8.32
Table tennis	26.65	8.55	3.86	39.06	7.76
Trevor	21.39	6.22	3.71	31.32	8.85
Average speedup:					8.86

3.4 *Conclusion*

In this chapter we have proposed two new, lossless and fast approaches of VBS-ME and MRF-ME to improve the performance of rate-distortion optimized video encoding process. They can be easily adopted in the recent H.264/AVC standard. The proposed PSADR-PDS method integrates the concept of the existing SAD reuse algorithm with the row-based PDS method to store and reuse the PSAD values obtained previously so as to make the block matching processes for the VBS-ME faster. Besides, we have also proposed the CTM-PDS approach which accelerates the MRF-ME process by making the tentative minimum value in the PDS independent of reference frame so that the condition of rejecting the non-best solutions in any reference frame becomes looser. It allows the rejections of candidates sooner. Furthermore, a very effective ME scheme is produced by combining these two proposed approaches together. This new ME scheme is substantially better than other representative fast full schemes such as the PDS and the SAD reuse algorithm for performing the VBS and MRF-ME. Experimental results show that the combined scheme has averagely 9.74 times of the processing speed of the exhaustive FS method, and meanwhile the qualities of the encoded sequences remain the same.

Chapter 4. Effective Star-Shaped Window and Star-Shaped Pattern for Motion Estimation

4.1 *Introduction*

One important factor that controls the computational complexity is the size and shape of the search window. For slow video sequences, since most of the best matches can be found near the search center, a smaller search window is enough, while a larger window size is more suitable for fast sequences. In previous ME schemes, the motion activity of video objects is assumed to have a square distribution. That is why square search window was adopted all along. Recently, it has been suggested that the movement of video objects in horizontal appears more frequently than the movement in vertical in natural video sequences so that a rectangular search window was proposed. However, after our investigations, it was found that the average motion activity of actual video objects shows itself a star-shaped distribution other than a rectangular distribution.

This chapter proposes respectively the new definitions of search window and search pattern for full ME schemes which can improve the speed of video encoding process. The proposed search window has a star shape which can cover nearly all best matches contained in the traditional square window by using just a small region. On the other hand, the proposed search pattern provides a star-shaped searching order which can yield faster results in the PDS process as compared with the spiral pattern. On average, more than 3 folds speedup of the PDS process can be achieved by replacing the traditional square window and the spiral pattern with the star-shaped window and the star-shaped pattern respectively while no quality degradation arises on average.

In Section 4.2, the motion activities in video sequences are firstly analyzed. Then the derivations of the star-shaped search window and the star-shaped search pattern are given in Section 4.3. Experimental results are shown and discussed in Section 4.4. Finally, Section 4.5 gives a conclusion of the investigation.

4.2 *Motion activity in natural video sequences*

The average motion activity within each 33x33 square region inside the 16 QCIF video sequences (car phone, Claire, coastguard, container, Erik, foreman, grandmother, hall monitor, miss America, mother & daughter, news, salesman, silent, Suzie, table tennis, and Trevor) with 150 frames can be represented by the average of the probability distributions of the 16 QCIF sequences of finding the best matches from different positions inside each 33x33 square region in different reference frames. Figure 4-2 shows the probability distribution (in percentage) clearly, and Figure 4-1 indicates the positions with different probabilities by different gray levels. In Figure 4-1, the positions with probabilities smaller than 0.0106% are represented in white color.

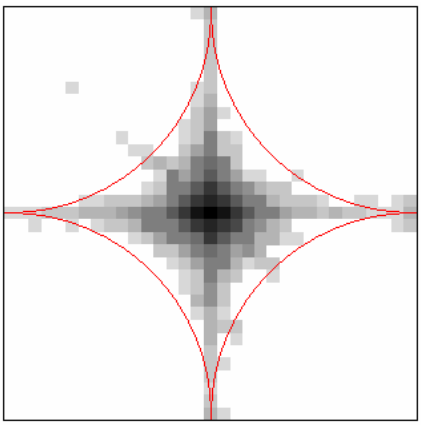


Figure 4-1. Average probability distribution (in gray scale) of finding the best matches (the deeper is the gray-level, the higher is the probability)

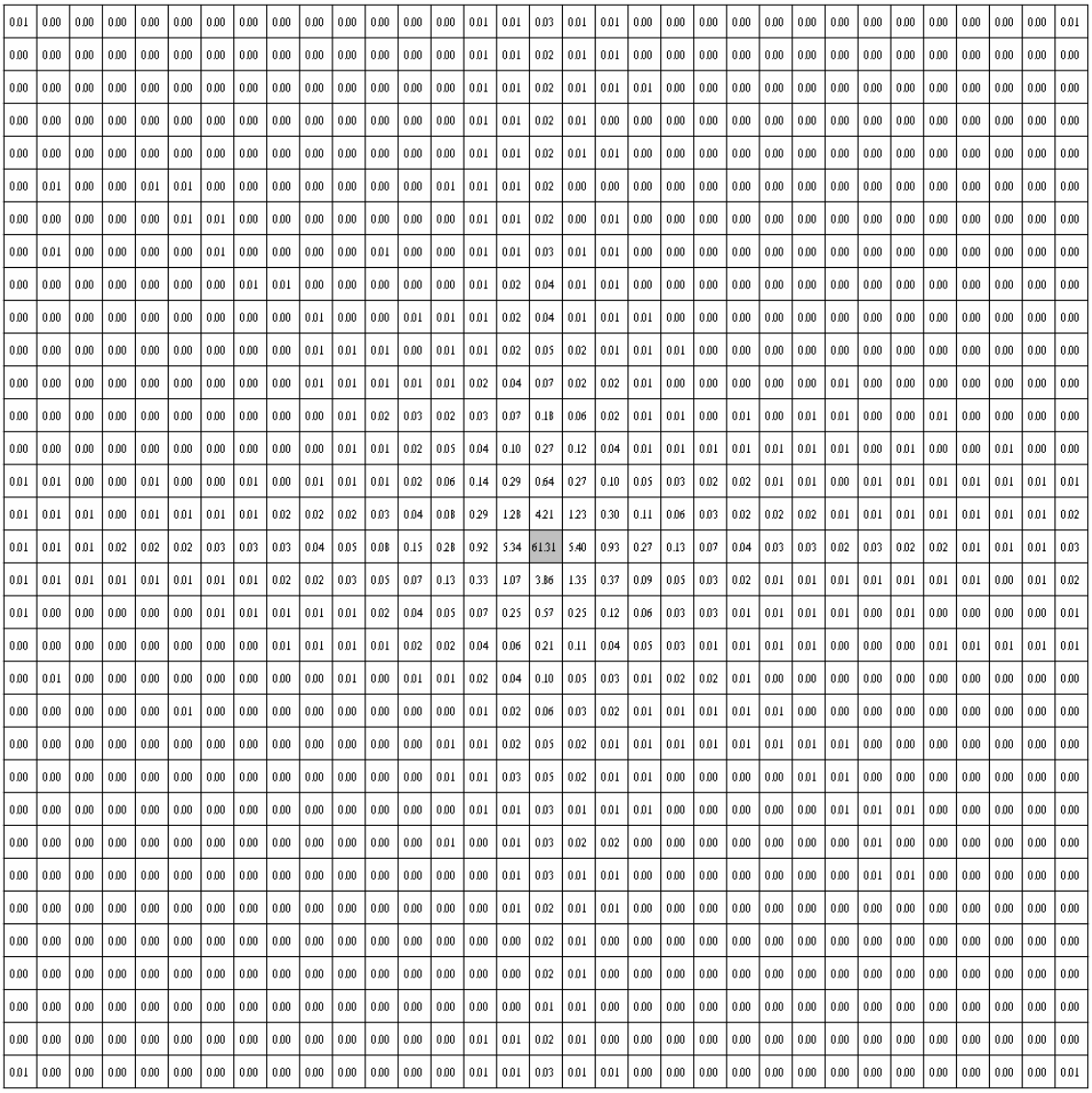


Figure 4-2. Average probability distribution (in percentage) of finding the best matches from different positions inside each 33x33 square region in different reference frames in 16 QCIF video sequences with 150 frames (The position in gray is the center of the square search regions)

From these figures, it can be found that the probability is largest at the center, and it decreases sharply in the nearby area of the center, and then decreases gradually away from the center. Finally, the positions belonging to similar probabilities form a star-shaped distribution. In addition, 82% of total positions inside a 33x33 square region have probabilities which are smaller than 0.0106%, which occupies only 2.7% of the total hit rate of successfully finding the best matches. As this average probability distribution was produced by using QCIF sequences, it represents the average motion

activity of QCIF sequences only. The motion activity of sequences with other frame sizes should be represented by other average probability distributions for which we believe are different from but bear some similarities with the present distribution pattern. The present distribution represents the average motion activity of many QCIF video sequences since the 16 sequences used to construct the probability distribution contain various common motions and camera movements including pan, tilt, translation, etc.

4.3 *Proposed scheme*

Apart from the size of a search window, the shape of the search window is also a significant factor that affects the performance of the ME process. If a search window covering some positions that the corresponding blocks in the reference frame are seldom the best matches, many operations are wasted for considering those blocks. Accordingly, a good definition of the shape of the search window should be based on the motion activity existing in natural video sequences.

Although the square window is commonly used as the search window for ME, it contains a large amount of wasteful positions which involve the majority of the total number of operations, hence it is inefficient. To improve the effectiveness of the search window, a star-shaped search window which matches the general motion activity is proposed.

4.3.1 Star-shaped search window

Since the motion activity in natural video sequences shows a star-shaped distribution, an effective search window should also be star-shaped as shown in Figure 4-3. Let the coordinate of the center of the window be $(i,j) = (0,0)$, and the four sides of the window are defined as sections of the sides of four uniform circles as

$$\begin{cases} (i+r)^2+(j+r)^2=r^2 & for\ \ -r\le i\le 0\ \ and\ \ -r\le j\le 0 \\ (i+r)^2+(j-r)^2=r^2 & for\ \ -r\le i\le 0\ \ and\ \ 0<j\le r \\ (i-r)^2+(j+r)^2=r^2 & for\ \ 0<i\le r\ \ and\ \ -r\le j\le 0 \\ (i-r)^2+(j-r)^2=r^2 & for\ \ 0<i\le r\ \ and\ \ 0<j\le r \end{cases}, \qquad \textbf{(4-1)}$$

where r is the radius of each uniform circle which also controls the size of the search window.

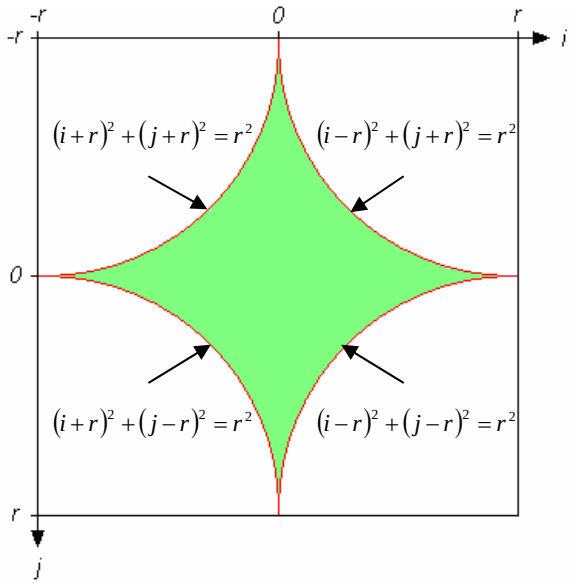


Figure 4-3. The proposed design of a star-shaped search window

Based on this design, a set of search windows with different sizes can be produced. Figure 4-4 gives an example of star-shaped search window with r = 16. To save the computation for considering unproductive positions during ME, a star-shaped window with $r = a$ should be used rather than a [-a,a] square window. Generally, about 75% of points inside the square window can be saved if a star-shaped window is used instead, and the star-shaped window can cover more than 95% of best matches included by the square window.

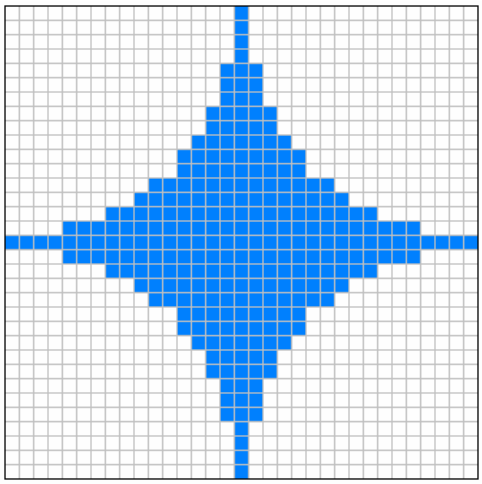


Figure 4-4. Star-shaped search window with r = 16

Besides the complexity of carrying out the ME process, the coding performance of a hybrid video encoder is also determined by the output bit-rate. Since the information of the horizontal and vertical components of each MV will be coded, and the coded results will then be appended to the output bit-stream, it may be even more efficient to give up some "best matches" in the block matching process sometimes if their corresponding MVs require too many bits for coding in order to reduce the overall bit-rate [17]. By using the VLC [62], the total number of bits required for coding the information of the u and v components of a MV (u,v) for $-63 \leq u, v \leq 63$ are shown in Figure 4-5. The required number of bits becomes larger and larger for the positions far from the center, and the pattern of positions requiring similar number of bits forms also a star-shaped distribution. This implies that those insignificant positions included by the square window not only have very small probabilities of being the best matches, but also require a large number of bits for coding their corresponding MVs. Therefore, although replacing the square window by the star-shaped window will exclude some best matches and hence decrease the video quality, this bad influence will be cancelled out as fewer bits will be used to code the MVs. This further supports the value of the proposed star-shaped window.

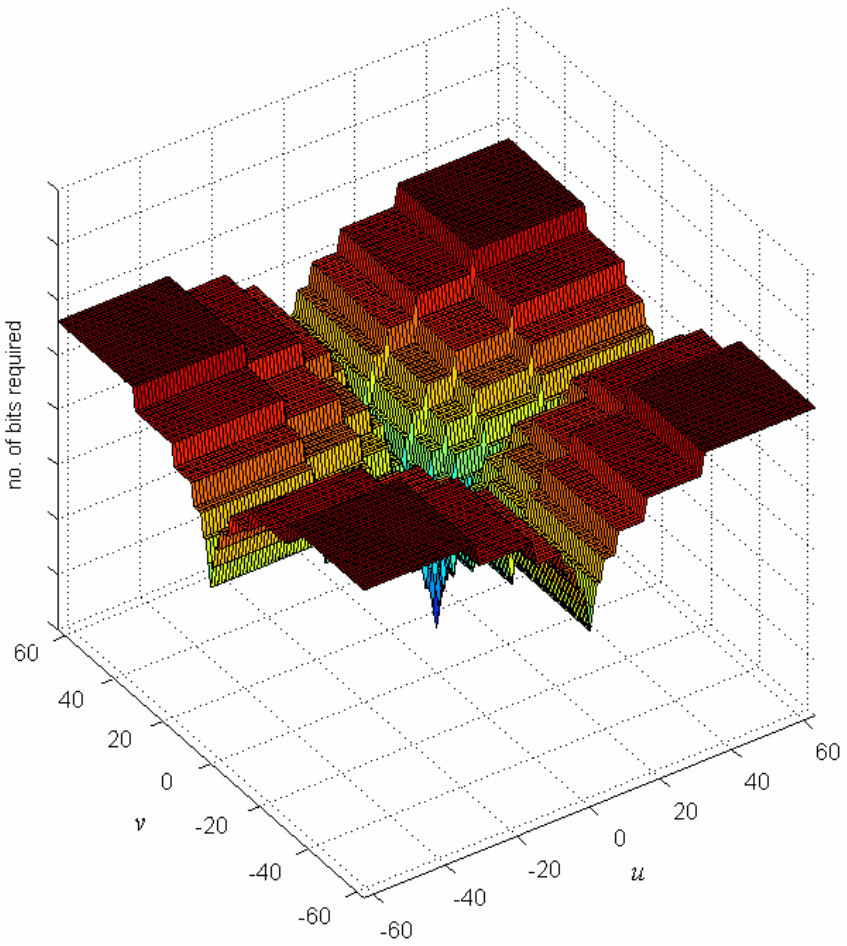


Figure 4-5. Number of bits required for coding a MV (*u*,*v*)

4.3.2 Star-shaped search pattern

As discussed in Section 2.2.2.1, applying a good search pattern inside a search window can speed up the PDS. A widely used search pattern is called the spiral scanning pattern. In fact, an effective search order should start from a position having the highest probability of requiring the minimum distortion cost, and then move along a path to go through all other positions from high probability to low probability. According to the star-shaped distribution of motion activity mentioned before as shown in Figure 4-1, the conventional spiral search pattern does not meet the expectation since it assumes that the distribution of motion activity is in a square shape. In other words, making use of a star-shaped search pattern that matches with the star-shaped motion

distribution in addition to the proposed star-shaped search window is preferable for performing the PDS.

Figure 4-6 illustrates the proposed star-shaped search pattern. Let r be the distance between the farthest position and the center of the star-shaped search window. The searching path is sectioned into $r + 1$ stars with different sizes. Each star is defined as follows:

$$\begin{cases} (i+k)^2+(j+k)^2=k^2 & for \ -k \le i \le 0 \ \ and \ \ -k \le j \le 0 \\ (i+k)^2+(j-k)^2=k^2 & for \ -k \le i \le 0 \ \ and \ \ 0 < j \le k \\ (i-k)^2+(j+k)^2=k^2 & for \ 0 < i \le k \ \ and \ \ -k \le j \le 0 \\ (i-k)^2+(j-k)^2=k^2 & for \ 0 < i \le k \ \ and \ \ 0 < j \le k \end{cases} \quad \textbf{(4-2)}$$

Applying this search pattern will start the search from a point star with $k = 0$ located at the center of the search window where the probability of being the best matched is the highest. The remaining stars will then be considered one by one, from $k = 1$ to r, for searching for the best match from all candidate positions in these stars. A possible search order for $r = 16$ is shown in Figure 4-7.

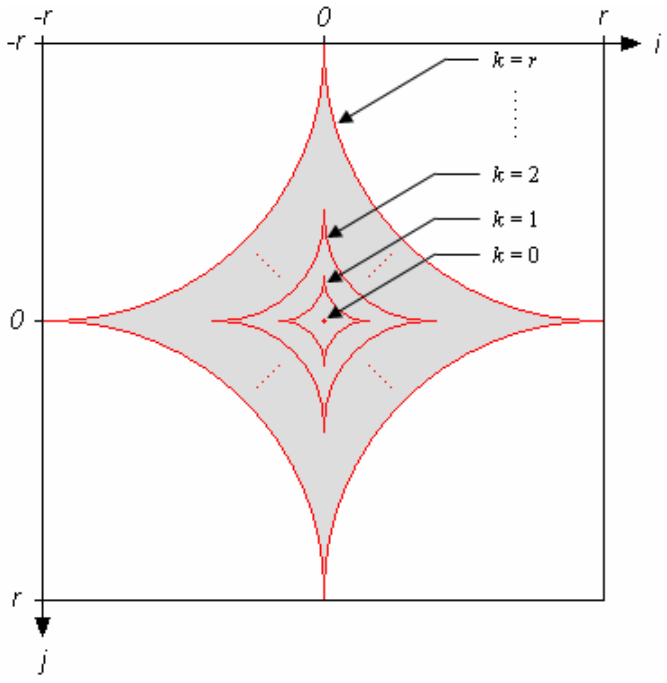


Figure 4-6. The proposed design of a star-shaped search pattern

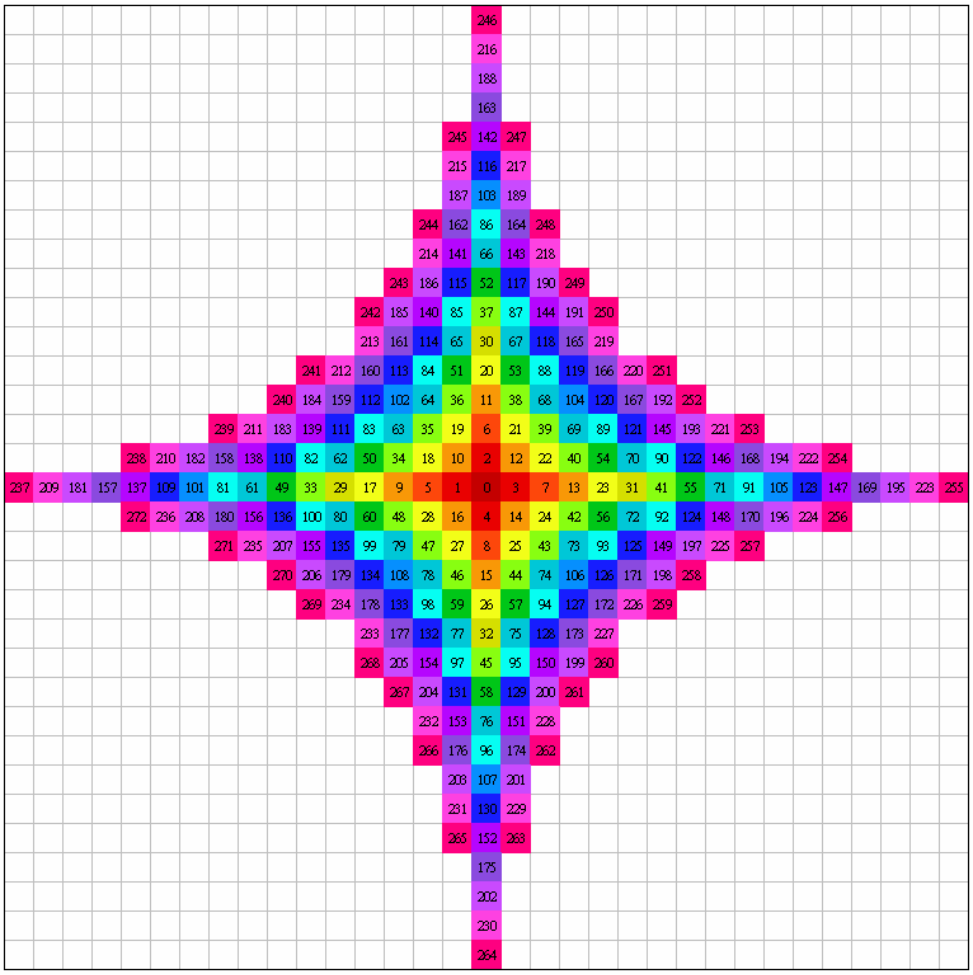


Figure 4-7. Possible star-shaped search pattern for r = 16

4.4 *Experimental results*

The experimental work was carried out by using the H.264/AVC verification model JM9.0 [51] as a platform to obtain the performance of encoding 16 QCIF sequences. All test sequences contains 150 frames. 16x16 block-size was used for ME, and one reference frame was allowed. In addition, The RDO feature of the verification model was turned off. All sequences were encoded in IPPP… format.

The performance of performing the PDS by using different approaches is compared. The first approach is a traditional approach that applies a ±16-pel square window with the spiral pattern; the second approach uses the proposed star-shaped

search window with r = 16 and the spiral pattern; the third one makes use of the proposed star-shaped search window with r = 16 as well as the proposed star-shaped pattern. Table 4-1 and Table 4-2 show the comparisons of respectively the ME time and the total number of operations (additions + subtractions + comparisons + taking absolute values) in million required for different approaches using the PDS and a quantization parameter (QP) of 30. Note in these two tables that the result in bold in each row is the best of the results obtained from the three different approaches for a relevant video sequence. Besides, the detailed number of operations required to apply PDS by different approaches to the testing sequences are listed in Table 4-3 to Table 4-6.

Table 4-1. Execution time (ms) of ME corresponding to four different approaches

Sequence	PDS					PDS + SEA	
	±16-pel square window	Proposed star-shaped search window with r = 16					
	Spiral pattern	Spiral pattern		Proposed star-shaped pattern			
	ME time	ME time	Speedup	ME time	Speedup	ME time	Speedup
Car Phone	3472	**1249**	**2.780**	1267	2.740	830	4.183
Claire	2904	1169	2.484	**1047**	**2.774**	706	4.113
Coastguard	4654	1314	3.542	**1066**	**4.366**	999	4.659
Container	3654	**1316**	**2.777**	1338	2.731	1011	3.614
Erik	3154	**1034**	**3.050**	1125	2.804	605	5.213
Foreman	3123	**921**	**3.391**	1162	2.688	795	3.928
Grandmother	4259	**1466**	**2.905**	1581	2.694	805	5.291
Hall Monitor	2905	**798**	**3.640**	1016	2.859	451	6.441
Miss America	5434	1690	3.215	**1614**	**3.367**	1387	3.918
M & D	3058	1392	2.197	**938**	**3.260**	577	5.300
News	3372	1251	2.695	**1020**	**3.306**	623	5.413
Salesman	3248	972	3.342	**812**	**4.000**	690	4.707
Silent	3190	**934**	**3.415**	1001	3.187	608	5.247
Suzie	4981	**1437**	**3.466**	1479	3.368	1031	4.831
Table Tennis	5404	**1579**	**3.422**	1591	3.397	1288	4.196
Trevor	3374	**1108**	**3.045**	1342	2.514	859	3.928
Average			3.085		**3.128**		4.686

Table 4-2. Total number of operations (additions + subtractions + comparisons + taking absolute values) in million required to carry out ME corresponding to four different approaches

Sequence	PDS					PDS + SEA	
	±16-pel square window	Proposed star-shaped search window with r = 16					
	Spiral pattern	Spiral pattern		Proposed star-shaped pattern			
	#Operations	#Operations	Speedup	#Operations	Speedup	#Operations	Speedup
Car Phone	3548.66	**1163.23**	**3.051**	1163.46	3.050	597.61	5.938
Claire	3195.22	**1165.71**	**2.741**	1170.33	2.730	556.18	5.745
Coastguard	4508.82	1323.46	3.407	**1315.86**	**3.427**	1030.45	4.376
Container	4201.61	1362.28	3.084	**1359.27**	**3.091**	824.39	5.097
Erik	3360.53	1091.99	3.077	**1090.23**	**3.082**	538.96	6.235
Foreman	3316.78	1053.28	3.149	**1052.01**	**3.153**	543.10	6.107
Grandmother	4493.65	1397.15	3.216	**1396.19**	**3.219**	818.86	5.488
Hall Monitor	2887.95	**957.80**	**3.015**	960.00	3.008	478.02	6.042
Miss America	5865.57	1815.14	3.231	**1813.58**	**3.234**	1225.23	4.787
M & D	3524.16	1153.79	3.054	**1152.40**	**3.058**	622.12	5.665
News	3489.30	1172.88	2.975	**1172.44**	**2.976**	561.90	6.210
Salesman	3417.32	1041.46	3.281	**1041.41**	**3.281**	597.61	5.718
Silent	3169.12	1015.11	3.122	**1013.80**	**3.126**	486.59	6.513
Suzie	4873.22	1556.30	3.131	**1555.79**	**3.132**	951.78	5.120
Table Tennis	5734.07	1713.05	3.347	**1709.38**	**3.354**	1320.81	4.341
Trevor	3988.32	1306.06	3.054	**1300.16**	**3.068**	807.84	4.937
Average			3.121		**3.124**		5.520

Table 4-3. Detailed number of operations in million required to carry out the PDS with a ±16-pel square window and the spiral pattern

Sequence	+ / –	Taking absolute value	Comparison	Total
Car Phone	2307.00	1153.50	88.16	3548.66
Claire	2076.19	1038.09	80.94	3195.22
Coastguard	2934.05	1467.02	107.75	4508.82
Container	2733.42	1366.71	101.48	4201.61
Erik	2184.14	1092.07	84.32	3360.53
Foreman	2155.57	1077.79	83.43	3316.78
Grandmother	2924.14	1462.07	107.44	4493.65
Hall monitor	1875.52	937.76	74.67	2887.95
Miss America	3820.08	1910.04	135.44	5865.57
M & D	2291.00	1145.50	87.66	3524.16
News	2268.23	1134.12	86.95	3489.30
Salesman	2221.23	1110.62	85.48	3417.32
Silent	2059.14	1029.57	80.41	3169.12
Suzie	3172.02	1586.01	115.19	4873.22
Table tennis	3734.21	1867.11	132.76	5734.07
Trevor	2594.13	1297.06	97.13	3988.32

Table 4-4. Detailed number of operations in million required to carry out the PDS with a proposed star-shaped window (r = 16) and the spiral pattern

Sequence	+ / –	Taking absolute value	Comparison	Total
Car Phone	757.03	378.52	27.68	1163.23
Claire	758.65	379.33	27.73	1165.71
Coastguard	861.67	430.83	30.95	1323.46
Container	885.06	442.53	34.69	1362.28
Erik	710.50	355.25	26.23	1091.99
Foreman	685.23	342.61	25.44	1053.28
Grandmother	909.79	454.90	32.46	1397.15
Hall monitor	622.87	311.44	23.49	957.80
Miss America	1182.77	591.38	40.99	1815.14
M & D	750.86	375.43	27.49	1153.79
News	763.33	381.66	27.88	1172.88
Salesman	677.51	338.75	25.20	1041.46
Silent	660.30	330.15	24.66	1015.11
Suzie	1013.73	506.87	35.71	1556.30
Table tennis	1116.09	558.05	38.90	1713.05
Trevor	850.31	425.15	30.60	1306.06

Table 4-5. Detailed number of operations in million required to carry out the PDS with a proposed star-shaped window (r = 16) and the proposed star-shaped pattern

Sequence	+ / –	Taking absolute value	Comparison	Total
Car Phone	757.18	378.59	27.69	1163.46
Claire	761.68	380.83	27.83	1170.33
Coastguard	856.71	428.35	30.80	1315.86
Container	885.06	442.53	31.69	1359.27
Erik	709.36	354.68	26.19	1090.23
Foreman	684.40	342.20	25.41	1052.01
Grandmother	909.17	454.58	32.44	1396.19
Hall monitor	624.31	312.16	23.54	960.00
Miss America	1181.75	590.87	40.96	1813.58
M & D	749.96	374.98	27.46	1152.40
News	763.04	381.52	27.87	1172.44
Salesman	677.47	338.74	25.20	1041.41
Silent	659.44	329.72	24.63	1013.80
Suzie	1013.40	506.70	35.70	1555.79
Table tennis	1113.70	556.85	38.83	1709.38
Trevor	846.45	423.23	30.48	1300.16

Table 4-6. Detailed number of operations in million required to carry out the SEA + PDS with a proposed star-shaped window (r = 16) and the proposed star-shaped pattern

Sequence	+ / –	Taking absolute value	Comparison	Total
Car Phone	396.41	180.56	20.64	597.61
Claire	369.55	167.14	19.49	556.18
Coastguard	678.28	321.50	30.68	1030.45
Container	544.31	254.52	25.56	824.39
Erik	358.03	161.38	19.55	538.96
Foreman	360.73	162.72	19.65	543.10
Grandmother	540.53	252.63	25.70	818.86
Hall monitor	318.39	141.56	18.07	478.02
Miss America	805.82	385.26	34.16	1225.23
M & D	412.28	188.51	21.32	622.12
News	373.12	168.93	19.85	561.90
Salesman	196.08	180.41	21.11	597.61
Silent	323.83	144.28	18.48	486.59
Suzie	627.39	296.05	28.34	951.78
Table tennis	867.95	416.32	36.53	1320.81
Trevor	533.50	249.11	25.22	807.84

From Table 4-1 and Table 4-2, it is found that using the proposed star-shaped search window instead of a square window can effectively speed up the PDS processes for all test sequences. The average speedups are 3.085 times in terms of the actual time and 3.121 times in terms of the total number of operations. Such a big acceleration is resulted since a large number of unimportant positions existing in the square window are disregarded. Even better results can be obtained by adopting the proposed star-shaped search pattern in addition to the star-shaped search window. Experimental results show that speedups of 3.128 times in terms of the actual time and of 3.124 times in terms of the total number of operations are acquired on average.

Although making use of the proposed star-shaped search window and the star-shaped search pattern brings a large improvement in ME speed, the qualities of the encoded sequences are not degraded at all. Table 4-7 and Table 4-8 show the resultant PSNR and the output bit-rates obtained by carrying out different approaches of the PDS for the “Carphone” and the “Silent” sequences with various QP. Referring to Figure 4-8 which plots the rate-distortion curves of encoding the “Carphone” and the “Silent” sequences, the proposed approaches result in a similar rate-distortion performance as the

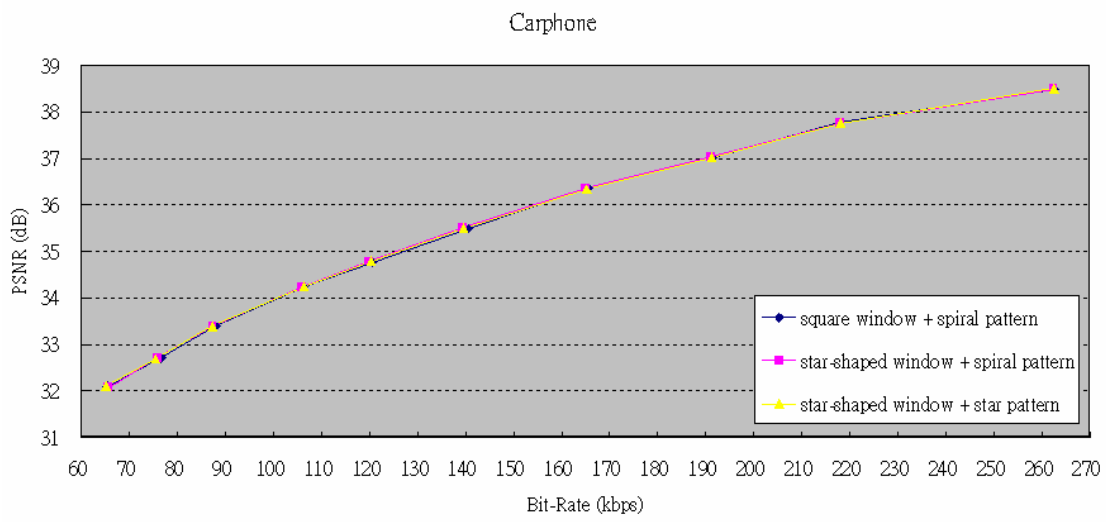
original PDS with a square window and the spiral pattern. Resemble results are found for other test sequences. This is because using the star-shaped window instead of the square window does not require examining those positions that their relative MVs need a large number of bits to be coded. Though some better matches do exist in those ignored positions, not using them saves the bits for encoding and hence maintains the original qualities of the encoded sequences.

Table 4-7. Resultant PSNR (dB) and output bit-rates (kbits/s) obtained by carrying out different approaches of the PDS for the "Carphone" sequence with various QP

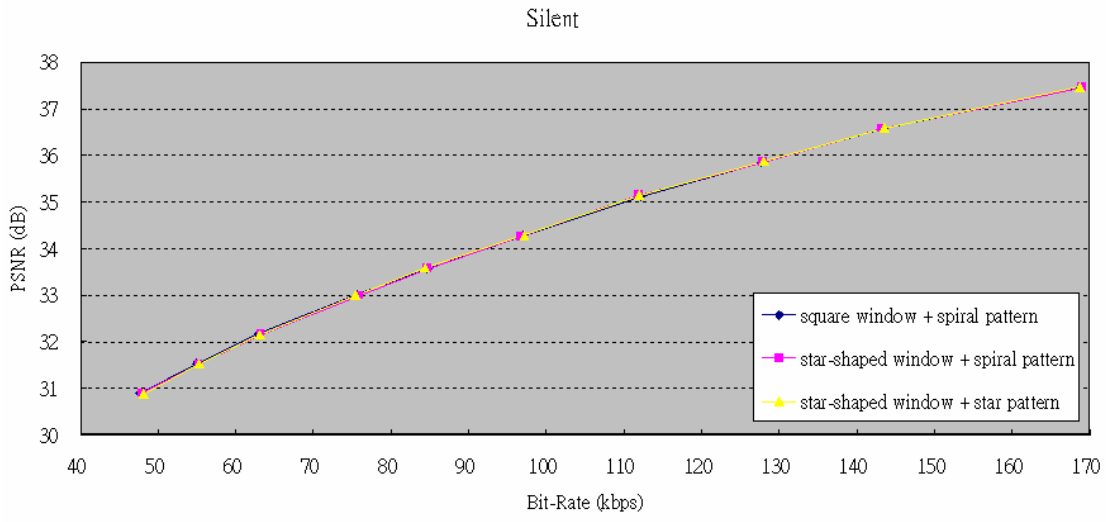
QP	±16-pel square window		Proposed star-shaped search window with r = 16			
	Spiral pattern		Spiral pattern		Star-shaped pattern	
	PSNR	Bit-rate	PSNR	Bit-rate	PSNR	Bit-rate
25	38.485	262.709	38.494	262.419	38.501	262.533
26	37.769	217.851	37.758	217.960	37.762	218.085
27	37.018	191.483	37.033	191.131	37.020	191.189
28	36.365	165.459	36.363	164.984	36.339	165.150
29	35.496	140.571	35.506	139.470	35.500	139.544
30	34.772	120.507	34.787	119.862	34.792	120.238
31	34.246	106.334	34.235	105.968	34.254	106.358
32	33.385	87.946	33.386	87.315	33.389	87.214
33	32.710	76.616	32.711	75.814	32.694	75.448
34	32.099	65.357	32.071	65.608	32.106	65.147

Table 4-8. Resultant PSNR (dB) and output bit-rates (kbits/s) obtained by carrying out different approaches of the PDS for the "Silent" sequence with various QP

QP	±16-pel square window		Proposed star-shaped search window with r = 16			
	Spiral pattern		Spiral pattern		Star-shaped pattern	
	PSNR	Bit-rate	PSNR	Bit-rate	PSNR	Bit-rate
25	37.467	168.749	37.464	169.002	37.470	168.720
26	36.588	143.477	36.583	143.285	36.596	143.539
27	35.854	127.701	35.862	127.858	35.877	128.029
28	35.128	112.154	35.148	111.923	35.149	111.944
29	34.284	96.998	34.261	96.768	34.293	97.157
30	33.566	84.632	33.573	84.736	33.590	84.382
31	33.025	75.638	32.995	75.952	33.001	75.419
32	32.164	62.882	32.162	63.246	32.150	63.112
33	31.534	54.992	31.520	55.224	31.526	55.274
34	30.895	47.709	30.902	47.941	30.880	48.094



(a)



(b)

Figure 4-8. Rate-distortion curves for different approaches of performing the PDS for the (a) "Carphone" and (b) the "Silent" sequences

Apart from the PDS, the proposed star-shaped search window can be applied to any other fast lossless or lossy algorithm of ME such as the CPME-PDS, the SEA and the block-based gradient descent search algorithm while the high qualities of encoded sequences can be remained. Furthermore, if the SEA is combined with the PDS and the proposed star-shaped search window and star-shaped pattern are used, a very effective full ME approach can be formed. The results in Table 4-1 and Table 4-2 prove that this

approach can averagely achieve actual time acceleration of 4.686 times and 5.52 times of speedup in terms of the total number of operations as compared with performing solely the PDS with a square window and the spiral pattern.

4.5 *Conclusion*

In this chapter, the definition of an efficient search window for any FS schemes and the definition of an efficient search pattern for the PDS have been discussed. Subsequently, a star-shaped search window and a star-shaped search pattern are proposed to improve the speed of full ME. The star-shaped search window follows the shape of the distribution of positions with similar motion activities. It covers nearly all highly probable positions of a traditional square window while its size is only about 1/4 of the size of the square window. The star-shaped search pattern lets the search proceeds in positions in descending order of the probabilities of being the best match such that even earlier rejection of candidate blocks with larger disaffinity for the PDS can be achieved. Experimental results show that a speedup of the processing time of more than three times on average is achieved when the PDS is performed with a star-shaped window in place of a square window, whilst the qualities of the encoded sequences are maintained. In addition, the overall speed for ME is further increased when the star-shaped search pattern is applied as well.

Chapter 5. Conclusion

5.1 *Conclusion of the works*

In this multimedia epoch, numerous applications involve accessing digital data in the form of text, image, audio and video. Nevertheless, the transmission and storage requirements of these multimedia data are very large, especially of video data. Therefore, a number of video coding standards were developed to make video data smaller in storage size and faster in transmission. These standards employ block-based motion estimation (ME) techniques to exploit the temporal redundancies between frames to achieve compression. However, due to the increasing demand of video quality, the size of video sequences became larger and larger, and hence keeping a high transmission speed is definitely a challenge. Prior standards such as H.261, MPEG-1 and MPEG-2 are no longer sufficient for satisfying the requirements today. To achieve larger compression ratio, some modern coding standards like H.263, MPEG-4 and H.264 which are designed to adopt novel video coding techniques were produced. Since the motion estimation and compensation parts always occupy the largest portion of the whole encoding time, there is a vast need of developing new techniques with low computational complexity to replace former techniques such as the full search algorithm (FSA). Our research also concentrates on fast motion estimation and compensation algorithms with very low or even no image deterioration.

In Chapter 3, we have proposed a new scheme of performing the variable block size (VBS) and multiple reference frames (MRF) ME. It can complete the process with fewer computations and hence reduce the processing time, and no drop of the qualities of coded sequences arises. For VBS-ME, a “partial SAD reusing PDS” (PSADR-PDS) approach was proposed. We apply the partial distortion search (PDS) for the 16x16

block size first, and store the sums of absolute differences (SAD) and the partial SAD (PSAD) values for each non-overlapping 4x4 region inside the 16x16 MB during the PDS. Then before we start the PDS for the next block size, we reuse those stored distortion values to pre-compute the full or partial distortion values for the blocks with the next block size. This can reduce the number of redundant operations that repeatedly compute the pixel differences computed already. We keep on recording the matching history of each 4x4 region throughout the ME processes of different block sizes so that the distortion values for any block size other than 16x16 can be pre-computed by using this method. To upgrade the effectiveness of the proposed approach, we also proposed a dispersed processing order to replace the traditional top-down order used in the PDS to calculate the pixel differences. By using the dispersed order, either the low error regions or the high error regions of the error surfaces will not be emphasized so the variation of the processing speed can be more stable. Moreover, using it can also remedy the problem of the over-processing of the upper half of blocks and the under-processing of the lower half of blocks. Our experimental results show that the dispersed order can bring an average speedup of 1.42 times of the ME time in terms of total number of operations. When it compares with the FSA, our proposed PSADR-PDS approach has a speedup ratio of 10.04 times in terms of actual time, which is significantly better than the conventional PDS and SAD reuse algorithm.

The “common tentative minimum PDS” (CTM-PDS) is our proposed new approach of MRF-PDS. In order to make the rejection of the non-best matches of image blocks in different reference frames faster, we do not use the conventional two-stage approach that performs the PDS in each reference frame independently to obtain the best result of each frame first, and then compares these results to choose the most optimal one. Differently, the proposed approach lets the unsearched candidates compare

with the candidate which is the optimal one among all searched candidates in different reference frames. This can be realized by adopting a common tentative minimum cost which is the cost function of the tentatively optimal candidate found during motion searching in various reference frames. The common tentative minimum is used as the rejecting criterion of all candidate blocks. It is looser than the original local tentative minimum of each reference frame. As a result, many worthless operations can be saved. According to our experimental results, on average, the proposed method is faster than the FSA with a speedup of actual time of 6.90 times. We have combined the PSADR-PDS method with the CTM-PDS method to produce a powerful ME scheme, and have found that it can averagely speed up the VBS and MRF-ME process for 9.74 times. We believe that the results of our work can certainly be useful in the future development of video codecs.

In Chapter 4, we have firstly studied the motion activity of natural video sequences. It has been found in general that the amount is larger for the motions with smaller magnitudes. Moreover, the motions with the same probability of appearance show a star-shaped distribution. To reduce the number of computations for finding the motions of image blocks during ME, we proposed a star-shaped search window to replace the traditional square window so that the searching process can concentrate only on the locations with large probabilities of successfully finding the motions. We suggest replacing a square window by a star-shaped window with only 25% of the size of the square window. Since the star-shaped window includes fewer positions, experimental results proved that the coding performance will not be dropped. It is because the motions outside the star-shaped window have too large magnitudes that coding the information of these motions will generate too many bits, and hence the compression ratio will drop. On average, applying the star-shaped window in place of the square

window to perform the PDS has an average acceleration of about 3.1 times of the ME time in terms of both the number of operations and actual processing time. Since just the shape and size of the search window are changed, any fast full search algorithms such as the MSEA can be used together with the star-shaped window to make the ME process faster.

Other than the star-shaped search window, we have also proposed a star-shaped search pattern which guides the searching order of positions inside the search window. The proposed order follows the general trend of the probability of successfully finding the motions, and starts from the search center and then moves in a star-shaped pattern away from the center. It can help advancing the obtaining of a low tentative minimum cost which is used for the rejection of unsearched candidate blocks during the PDS. The earlier a small tentative minimum cost is acquired, the sooner the unsearched non-best matches can be rejected. Therefore, the PDS process can be done quicker.

5.2 *Future research directions*

Among various components of video coding techniques, ME is the most essential part contributing to the compression ratio. However, it is really unfavorable that it requires a very heavy computational load. In recent video standards such as the H.264/AVC, variable block size and multiple reference frames approaches of ME were introduced to further strengthen the compression power of ME, but meanwhile the processing complexity was also largely increased. Therefore, except for targeting a greater compression ratio, it is also important to put our focus on conducting researches into the ways to speed up the ME process. Other than lossy methods, we also aim at developing some lossless algorithms of fast ME scheme with or without variable block size and multiple reference frames.

Concerning applying the variable block size ME for a MB, if we can know which types of partitioning of the MB are impossible in advance, much computational effort can be avoided. We believe that the edge information of the contents of the MB can give us some clues to reject some partition modes. For the existing seven block sizes ME scheme, since the boundaries of the possible small blocks inside a MB appear in some predefined positions only, we can just detect whether edges are there or not, and need not to check the other positions. In case we find that the boundaries of some small blocks do not contain any edge, we can prove that those boundaries are located at positions with smooth texture so the contents of these blocks and the blocks next to them should belong to the same video object. Therefore, it is not likely that partitioning the MB at those positions will provide an improved result, and hence we can safely discard those small blocks. Another possible way to determine whether a region inside a MB should not be cut into two is to compare the average pixel values of the two halves of the region. If the average pixel values are very similar, we can conclude that the contents of the two halves should belong to the same video object, and skip this cutting. Although these methods cannot be completely accurate, the error should be very small, and it is still valuable to try these methods.

Since the number of bits required for coding a MB depends on not only the residue error, but also on other factors such as the MVs, reference frame indices, etc, the traditional way that the block matching criterion used in ME considers only the distortion value is not appropriate. Using RD value which involves also the influence of MVs and reference frame indices as the matching criterion instead of the distortion value is a better way. For years, SAD etc are used as the measure of the distortions between image blocks and their predictions. Assume that the rate term of RD value is fixed, we normally choose the predictions with the smallest distortion values, because it

is assumed that choosing them will generate the least number of bits, and lead to the highest visual quality. However this is not always true in real situations. This is because the number of bits for the contents of residue error is in fact controlled by the run-level coding but not the distortion values. There are some cases that coding two different predictions with the same distortion value generate different number of bits. Therefore, we are interested to study the relationship between the patterns of residue blocks inputted to the run-level coding and the corresponding number of generated bits. The research results must be useful at developing matching criteria that can make the compression ratio better.

References

[1] 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), Mar. 1993.

[2] ITU-T and ISO/IEC JTC 1, “Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Video,” ITU-T Recommendation H.262 and ISO/IEC 13818-2 (MPEG-2), Nov. 1994.

[3] ITU-T, “Video Coding for Low Bit Rate Communication,” ITU-T Recommendation H.263 Version 1, Nov. 1995.

[4] ITU-T, “Video Coding for Low Bit Rate Communication,” ITU-T Recommendation H.263 Version 2, Jan. 1998.

[5] ISO/IEC, “Information Technology – Coding of Audio-Visual Objects: Part 2 Visual,” ISO/IEC 14496-2 (MPEG-4 Visual Version 1), Apr. 1999.

[6] Joint Video Team of ITU-T and ISO/IEC JTC 1, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC),” Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Doc. JVT-G050, Dec. 2003.

[7] ITU-T, “Video Codec for Audiovisual Services at P×64 Kbit/s,” ITU-T Recommendation H.261, Mar. 1993.

[8] W. C. Siu, “Digital Video Coding Technologies, for Multimedia Communications and Storage,” McGraw-Hill, Aug. 2007. (Chapter 6: Motion Estimation)

[9] J. R. Jain, A. K. Jain, “Displacement Measurement and Its Application in Interframe Image Coding,” IEEE Trans. on Commun.,” vol. COM-29, pp. 1799-1808, Dec. 1981.

[10] M. H. Chan, Y. B. Yu and A. G. Constantinides, “Variable Size Block Matching Motion Compensation with Applications to Video Coding”, IEE Proc. on Commun., Speech and Vision, vol. 137, pt. 1, no. 4, pp. 205-212, Aug. 1990.

[11] H. A. Mahmoud and M. A. Bayoumi, “An Efficient Low-Bit Rate Adaptive Mesh-Based Motion Compensation Technique,” IEEE Proc. Int. Conf. Electronics, Circuits and Systems (ICECS), vol. 1, pp. 491-494, Dec. 2000.

[12] H. Gharavi and Mike Mills, “Blockmatching Motion Estimation Algorithms – New Result,” IEEE Trans. on Circuits and Systems, vol. 37, pp. 649-651, May 1990.

[13] M. J. Chen, L. G. Chen, T. D. Chiueh and Y. P. Lee, “A New Block-Matching Criterion for Motion Estimation and Its Implementation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 5, no. 3, pp. 231-236, Jun. 1995

[14] A. Fuldseth and T. A. Ramstad, “A New Error Criterion for Block Based motion Estimation,” IEEE Proc. Int. Conf. on Image Processing (ICIP), vol. 3, pp. 188-191, Oct. 1995.

[15] S. Kappagantula and K. R. Rao, “Motion Compensated Interframe Image Prediction,” IEEE Trans. on Commun., vol. COM-33, no. 9, pp. 1011-1015, Sep. 1985.

[16] B. Girod, “Rate-Constrained Motion Estimation,” SPIE Proc. Conf. on Visual Commun. And Image Processing, pp. 1026-1034, Sep. 1994.

[17] G. J. Sullivan and R. L. Baker, “Rate-distortion optimization for video compression,” IEEE Signal Processing Mag., vol.15, issue 6, pp.74-90, Nov. 1998.

[18] T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro, “Motion Compensated Interframe Coding for Video Conferencing,” Proc. National Telecommun. Conf., pp. G5.3.1-5.3.5, Nov.-Dec. 1981.

[19] Keith H. K. Chow and M. L. Liou, “Genetic Motion Search Algorithm for Video Compression,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 3, no. 6, pp. 440-445, Dec. 1993.

[20] J. Y. Tham, S. Ranganath, M. Ranganath and A. A. Kassim, “A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 8, no. 4, pp. 369-377, Aug. 1998.

[21] L. M. Po and W. C. Ma, “A Novel Four-Step Search Algorithm for Fast Block Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 6, no. 3, pp. 313-317, Jun. 1998.

[22] P. I. Hosur and K. K. Ma, “Motion Vector Field Adaptive Fast Motion Estimation,” IEEE Proc. Int. Conf. on Information, Commun. and Signal Processing (ICICS), Dec. 1999.

[23] R. Srinivasan and K. R. Rao, “Predictive Coding Based on Efficient Motion Estimation,” IEEE Trans. on Commun., vol. COM-33, pp. 888-896, Sep. 1985.

[24] R. Li, B. Zeng and Liou, M. L., “A New Three-Step Search Algorithm for Block Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 4, no. 4, pp. 438-442, Aug. 1994.

[25] A. Puri, H. M. Hang and D. L. Schilling, “An Efficient Block-Matching Algorithm for Motion Compensated Coding,” IEEE Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), pp. 25.4.1-25.4.4., 1987.

[26] L. K. Liu and E. Feig, “A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 6, no. 4, pp. 419-422, Aug. 1996.

[27] M. E. Al-Mualla, C. N. Canagarajah and David R. Bull, “Simplex Minimization for Single- and Multiple-Reference Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 11, no. 12, pp. 1209-1220, Dec. 1998.

[28] S Zhu and K. K. Ma, “A New Diamond Search Algorithm for Fast Block Matching Motion Estimation,” IEEE Proc. Int. Conf. on Information, Commun. and Signal Processing (ICICS), pp. 292-296, Sep. 1997.

[29] F. H. Cheng and S. N. Sun, “New Fast and Efficient Two-Step Search Algorithm for Block Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 9, no. 7, pp. 977-983, Oct. 1999.

[30] C. Zhu, X. Lin and L. P. Chau, “Hexagon-Based Search Pattern for Fast Block Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 12, no. 5, pp. 349-355, May 2002.

[31] A. M. Tourapis, Oscar C. Au and M. L. Liou, “Highly Efficient Predictive Zonal Algorithms for Fast Block-Matching Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 12, no. 5, pp. 934-947, Oct. 2002.

[32] B. Liu and A. Zaccarin, “New Fast Algorithm for the Estimation of Block Motion Vectors,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 3, no. 2, pp. 148-157, Apr. 1993.

[33] Y. L. Chan and W. C. Siu, “New Adaptive Pixel Decimation for Block Motion Vector Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 6, no. 1, pp. 113-118, Feb. 1996.

[34] Y. Wang, Y. Wang and H. Kuroda, “A Globally Adaptive Pixel-Decimation Algorithm for Block-Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 10, no. 6, pp. 1006-1011, Sep. 2000.

[35] S. Eckart and C. Fogg, “ISO/IEC MPEG-2 software video codec,” Proc. SPIE, vol. 2419, pp. 100-118, 1995.

[36] J. N. Kim and T. S. Choi, “A Fast Full-Search Motion Estimation Algorithm Using Representative Pixels and Adaptive Matching Scan,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 10, no. 7, pp. 1040-1048, Oct. 2000.

[37] Y. L. Chan, W. C. Siu and K. C. Hui, “Adaptive Partial Distortion Search for Block Motion Estimation,” Journal of Visual Communication and Image, vol. 15, pp. 489-506, Nov. 2004.

[38] K. C. Hui, W. C. Siu, and Y. L. Chan, “New adaptive partial distortion search using clustered pixel matching error characteristic,” IEEE Trans. on Image Processing, vol. 14, no. 5, pp. 597–607, May 2005.

[39] C. K. Cheung and L. M. Po, “A Hierarchical Block Motion Estimation Algorithm Using Partial Distortion Measure,” IEEE Proc. Int. Conf. on Image Processing (ICIP), vol. 3, pp. 606-609, 1997.

[40] C. K. Cheung and L. M. Po, “Normalized Partial Distortion Search Algorithm for Block Motion Estimation,” IEEE Trans. on Circuits and Systems for Video Tech., vol. 10, no. 3, pp. 417-422, Apr. 2000.

[41] Y. S. Chen, Y. P. Hung and C. S. Fuh, “Fast Block Matching Algorithm Based on the Winner-Update Strategy,” IEEE Trans. on Image Processing, vol. 10, no. 8, pp. 1212-1222, Aug. 2001.

[42] W. Li and E. Salari, “Successive Elimination Algorithm for Motion Estimation,” IEEE Trans. on Image Processing, vol. 4, no. 1, pp. 105-107, Jan. 1995.

[43] X. Q. Gao, C. J. Duanmu, C. R. Zou and Z. Y. He, “Multi-Level Successive Elimination Algorithm for Motion Estimation in Video Coding,” IEEE Proc. Int. Sym. on Circuits and Systems (ISCAS), vol. 4, pp. 227-230, May 1999.

[44] K. Sauer and B. Schwartz, “Efficient Block Motion Estimation Using Integral Projections,” IEEE Trans. on Image Processing, vol. 6, no. 5, pp. 513-518, Oct. 1996.

[45] Y. C. Lin and S. C. Tai, “Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression,” IEEE Trans. on Commun., vol. 45., no. 5, pp. 527-531, May 1997.

[46] Y. L. Chan and W. C. Siu, “Edge Oriented Block Motion Estimation for Video Coding,” IEE Proc. – Vision, Image and Signal Processing, vol. 144, no. 3, pp. 136-144, Jun. 1997.

[47] Y. L. Chan and W. C. Siu, “An Efficient Search Strategy for Block Motion Estimation Using Image Features,” IEEE Trans. on Image Processing, vol. 10, no. 8, pp. 1223-1238, Aug. 2001.

[48] B. 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, Jan. 2001.

[49] G. J. Sullivan, “Multi-hypothesis Motion Compensation for Low Bit-rate Video Coding,” IEEE Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), vol. 5, pp. 437-440, Apr. 1993.

[50] S. Nogaki and M. Ohta, “An Overlapped Block Motion Compensation for High Quality Motion Picture Coding,” IEEE Proc. Int. Sym. on Circuits and Systems, vol. 1, pp. 184-187, May 1992.

[51] H.264/AVC Reference Software: http://iphome,hhi.de/suehring/tml/download/.

[52] Z. Zhou, M. T. Sun and Y. F. Hsu, “Fast Variable Block-Size Motion Estimation Algorithms Based on Merge and Split Procedures for H.264/MPEG-4 AVC,” IEEE Proc. Int. Sym. on Circuits and Systems (ISCAS), vol. 3, pp. 725-728, May 2004.

[53] Andy Chang, Peter H. W. Wong, Y. M. Yeung and Oscar C. Au, “Fast Multi-Block Selection for H.264 Video Coding,” IEEE Proc. Int. Sym. on Circuits and Systems (ISCAS), vol. 3, pp. 817-820, May 2004.

[54] J. Lee and B. Jeon, “Fast Mode Decision for H.264 with Variable Motion Block Sizes,” Proc. Int. Sym. on Comp. and Info. Sciences (ISCIS), pp. 723-730, Nov. 2003.

[55] Y. F. Shen, D. M. Zhang, C. Huang and J. T. Li, “Fast mode selection based on texture analysis and local motion activity in H264/JVT,” IEEE Proc. Int. Conf. on Commun., Circuits and Systems (ICCCAS), vol. 1, pp. 539-542, Jun. 2004.

[56] D. Wu, S. Wu, K. P. Lim, F. Pan, Z. G. Li and X. Lin, “Block Inter Mode Decision for Fast Encoding of H.264,” IEEE Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), vol. 3, pp. 181-184, May 2004.

[57] M. J. Chen, Y. Y. Chiang, H. J. Li and M. C. Chi, “Efficient Multi-Frame Motion Estimation Algorithms for MPEG-4 AVC/JVT/H.264,” IEEE Proc. Int. Sym. on Circuits and Systems (ISCAS), vol. 3, pp. 737-740, May 2004.

[58] Y. H. Hsiao, T. H. Lee and P. C. Chang, “Short/Long-Term Motion Vector Prediction in Multi-Frame Video Coding System,” IEEE Proc. Int. Conf. on Image Processing (ICIP), vol. 3, pp. 1449-1452, Oct. 2004.

[59] Andy Chang, Oscar C. Au and Y. M. Yeung, “A Novel Approach to Fast Multi-Frame Selection for H.264 Video Coding,” IEEE Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), vol. 2, pp. 704-707, May 2003.

[60] C. W. Ting, L. M. Po and C. H. Cheung, “Center-Biased Frame Selection Algorithms for Fast Multi-Frame Motion Estimation in H.264,” IEEE Proc. Int. Conf. on Neural Networks and Signal Processing (ICNNSP), vol. 2, pp. 1258-1261, Dec. 2003.

[61] A. Ortega and K. Ramchandran, “Rate-distortion methods for image and video compression: An Overview,” IEEE Signal Processing Mag., vol. 15, pp. 23-50, Nov. 1998.

[62] G. Bjontegaard and K. Lillevold, “Context-Adaptive VLC Coding of Coefficients,” Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Doc. JVT-C028, May 2002.

[63] I. H. Witten et al, “Arithmetic Coding for Data Compression,” Comm. of the ACM, 30 (6), pp. 520-541, 1987.

[64] D. Marpe, H. Schwartz and T. Wiegand, “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard,” IEEE Trans. on Circuits and Systems for Video Tech., 13 (7), pp. 620-636, July 2003.

[65] M. Bierling, “Displacement Estimation by Hierarchical Block Matching,” Proc. VCIP, vol. 1001, pp. 942-951, 1988.

[66] K. M. Uz, M. Vetterli and D. J. LeGall, “Interpolative Multiresolution Coding of Advanced Television with Compatible Subchannels,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 1, no. 1, pp. 86-99, Mar. 1991.

[67] Iain E. G. Richardson, “H.264 and MPEG-4 Video Compression,” John Wiley and Sons, 2003.

[68] A. Puri, H. M. Hang and D. L. Schilling, “Interframe Coding with Variable Block-Size Motion Compensation,” IEEE Proc. Global Commun. Conf. (GLOBECOM), pp. 65-67, Nov. 1987.

[69] T. Wiegand, X. Zhang and B. Girod, “Block-Based Hybrid Video Coding Using Motion-Compensated Long-Term Memory Prediction,” Proc. of Picture Coding Sym., pp. 153-158, Sep. 1997.

[70] J. Boyce, “Adaptive Reference Picture Weighting Using Reference Picture Index,” Doc. JVT-D122, Jul. 2002.

[71] J. Boyce, “Changes to Adaptive Reference Picture Weighting,” Doc. JVT-E060, Oct. 2002.

[72] J. Boyce, “Weighted Prediction Clean-Up,” Doc. JVT-F034, Dec. 2002.

[73] Y. Kikuchi and T. Chujoh, “Improved Multiple Frame Motion Compensation Using Frame Interpolation,” Doc. JVT-B075, Feb. 2002.

[74] A. Hallapuro and M. Karczewicz, “Low Complexity Transform and Quantization – Part 1: Basic Implementation,” JVT Doc. B-038, Feb. 2001.

[75] M. Karczewicz and R. Kurceren, “A Proposal for SP-Frames,” Doc. VCEG-L-27, Jan. 2001.

[76] M. Karczewicz and R. Kurceren, “Improved SP-Frame Encoding,” Doc. VCEG-M-73, Apr. 2001.

[77] R. Kurceren and M. Karczewicz, “New Marcoblock Modes for SP-Frames,” Doc. VCEG-O-47, Dec. 2001.

[78] A. M. Tourapis, Oscar C. Au and M. L. Liou, “Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) – Enhancing Block Based Matching Estimation,” SPIE Proc. Visual Commun. Image Processing (VCIP), vol. 4310, pp. 883-892, Dec. 2000.

[79] C. Cafforio and F. Rocca, “Methods for Measuring Small Displacements of Television Images,” IEEE Trans. on Info. Theory, vol. IT-22, no. 5, pp. 573-579, Sep. 1976.

[80] “ITU-T Recommendation H.263 Software Implementation,” Digital Video Coding Group, Telenor R & D, 1995.

Glossary

AVC	Advanced video coding
B-frame	Bidirectional predictive frame
BMA	Block matching algorithm
CPME-PDS	Clustered pixel matching error for adaptive partial distortion search
CTM-PDS	Common tentative minimum partial distortion search
DCT	Discrete cosine transform
DS	Diamond search
HEXBS	Hexagon-based search
FS	Full search
FSA	Full search algorithm
I-frame	Intra frame
LDSP	Large diamond search pattern
MB	Marcoblock
ME	Motion estimation
MPEG	Moving Picture Expects Group
MRF	Multiple reference frames
MRF-ME	Multiple reference frames motion estimation
MSEA	Multilevel successive elimination algorithm
MV	Motion vector
n-SHS	n-step hierarchical search
P-frame	Predictive frame
PDS	Partial distortion search
PMV	Predicted motion vector
PSAD	Partial sum-of-absolute-difference
PSADR-PDS	Partial sum-of-absolute-difference reusing partial distortion search
PSNR	Peak signal-to-noise ratio
QCIF	Quarter common intermediate format
QP	Quantization parameter
RD	Rate-distortion
RDO	Rate-distortion optimization
SAD	Sum-of-absolute-difference
SDSP	Small diamond search pattern
SEA	Successive elimination algorithm
SSD	Sum-of-squared-difference
VBS	Variable block size
VBS-ME	Variable block size motion estimation
VLC	Variable length coding