2011

Efficient Motion Estimation and Mode Decision Algorithms for Advanced Video Coding

Mohammed Golam Sarwer
University of Windsor

Efficient Motion Estimation and Mode Decision Algorithms for Advanced Video Coding

by

Mohammed Golam Sarwer

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy at the
University of Windsor

Windsor, Ontario, Canada

2011

Declaration of Co-Authorship / Previous Publication

I. Co-Authorship Declaration

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

This thesis also incorporates the outcome of a joint research undertaken by me under the supervision of Professor Dr. Jonathan Wu. The collaboration is covered in Chapter 3, 4, 5, 6, and 7 of the thesis. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-author was primarily through the provision of valuable suggestions and helping in comprehensive analysis of the simulation results for publication.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This thesis includes 16 original papers that have been previously published/submitted for publication in peer reviewed journals and conferences, as follows:

Thesis Chapter	Publication title/full citation	Publication status*
Chapter 3	Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Efficient Two Step Edge based Partial Distortion Search for Fast Block Motion Estimation" IEEE Transactions on Consumer Electronics, Vol. 55, No. 4, Nov. 2009, pp. 2154-2162.	published
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "Fast Block Motion Estimation by edge based partial distortion search", Proceedings of the IEEE International Conference on Image Processing 2009 (ICIP-2009), Cairo, Egypt, November 7-10, 2009, pp. 1573 - 1576.	published
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "Efficient Partial Distortion Search Algorithm for Block based Motion Estimation", Proceedings of the IEEE CCECE 2009, St John's, NL, pp. 890-893.	published
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Adaptive Variable Block-Size Early motion estimation	published

	termination algorithm for H.264/AVC Video Coding Standard." IEEE Transaction on Circuit and System for Video Technology, volume 19, number 8, August 2009, pp 1196-1201.	
Chapter 4	Mohammed Golam Sarwer, Thanh Minh Nguyen and Q.M. Jonathan Wu, "Fast Motion estimation of H.264/AVC by adaptive early termination", Proceedings of the 10th IASTED International Conference SIP 2008, HI, USA, pp. 140-145.	published
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "Region Based Searching for Early Terminated Motion Estimation Algorithm of H.264/AVC Video Coding Standard", Proceedings of the IEEE CCECE 2009, St John's, NL, pp. 468-471.	published
Chapter 5	Mohammed Golam Sarwer, and Q.M. Jonathan Wu, " An Efficient search range Decision Algorithm for Motion Estimation of H.264/AVC, " International Journal of Circuits, Systems and Signal Processing, vol. 3, issue 4, pp. 173-180, 2009.	published
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "Adaptive Search Area Selection of Variable Block-Size Motion Estimation of H.264/AVC Video Coding Standard", IEEE International Symposium on Multimedia ISM 2009, San Diego, California, USA, December 14-16, 2009, pp. 100-105.	published
Chapter 6	Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Improved Intra Prediction of H.264/AVC" accepted in the book "Video Coding", ISBN 978-953-7619-X-X, IN-TECH publisher.	In Press
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "Enhanced Intra Coding of H.264/AVC Advanced Video Coding Standard with Adaptive Number of Modes", International Conferences on Active Media Technology (AMT 2010), Toronto, Canada, LNCS 6335, pp. 361–372, August 2010.	published
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "A Novel Bit Rate Reduction Method of H.264/AVC Intra Coding", International Congress on Image and Signal Processing (CISP'10), 16-18 October 2010, Yantai, China, pp. 24-28.	published
	Mohammed Golam Sarwer, and Q. M. Jonathan Wu, " Improved DC Prediction for H.264/AVC intra Coding " 2011 International Conference on Communication and Electronics Information - ICCEI 2011, Haikou, China.	Accepted
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Performance Improvement of Intra Coding in H.264/AVC	Submitted

	Advanced Video Coding Standard". Submitted to Journal of Visual Communication and Image Representation.	
Chapter 7	Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Enhanced SATD based Cost Function for Mode Selection of H.264/AVC Intra Coding". Submitted to Springer Journal of Signal, Image and Video Processing.	Submitted
	Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Enhanced low complex cost function of H.264/AVC intra mode decision". International Conference on Multimedia and Signal Processing (CMSP'11), China,	Accepted
	Mohammed Golam Sarwer, Q.M. Jonathan Wu, X. P Zhang "Efficient rate-distortion optimization of H.264/AVC intra coder". Submitted to International Conference on Image Proceeding, 2011	Submitted

I certify that I have obtained permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

H.264/AVC video compression standard achieved significant improvements in coding efficiency, but the computational complexity of the H.264/AVC encoder is drastically high. The main complexity of encoder comes from variable block size motion estimation (ME) and rate-distortion optimized (RDO) mode decision methods.

This dissertation proposes three different methods to reduce computation of motion estimation. Firstly, the computation of each distortion measure is reduced by proposing a novel two step edge based partial distortion search (TS-EPDS) algorithm. In this algorithm, the entire macroblock is divided into different sub-blocks and the calculation order of partial distortion is determined based on the edge strength of the sub-blocks. Secondly, we have developed an early termination algorithm that features an adaptive threshold based on the statistical characteristics of rate-distortion (RD) cost regarding current block and previously processed blocks and modes. Thirdly, this dissertation presents a novel adaptive search area selection method by utilizing the information of the previously computed motion vector differences (MVDs).

In H.264/AVC intra coding, DC mode is used to predict regions with no unified direction and the predicted pixel values are same and thus smooth varying regions are not well de-correlated. This dissertation proposes an improved DC prediction (IDCP) mode based on the distance between the predicted and reference pixels. On the other hand, using the nine prediction modes in intra 4x4 and 8x8 block units needs a lot of overhead bits. In order to reduce the number of overhead bits, an intra mode bit rate reduction method is suggested. This dissertation also proposes an enhanced algorithm to estimate the most probable mode (MPM) of each block. The MPM is derived from the prediction mode direction of neighboring blocks which have different weights according to their positions. This dissertation also suggests a fast enhanced cost function for mode decision of intra encoder. The enhanced cost function uses sum of absolute Hadamard-transformed differences (SATD) and mean absolute deviation of the residual block to estimate distortion part of the cost function. A threshold based large coefficients count is also used for estimating the bit-rate part.

Dedicated to

my mother FIROJA BEGUM, and

my wife DILSHAD HUSSAIN

with love

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to Professor Jonathan Wu, my advisor, for his excellent guidance and invaluable support, which helped me accomplish the doctorate degree and prepared me to achieve more life goals in the future. His total support of my dissertation and countless contributions to my technical and professional development made for a truly enjoyable and fruitful experience. Special thanks are dedicated for the discussions we had on almost every working day during my research period and for reviewing my dissertation.

I would also like to thank my doctoral committee members Dr. Imran Ahmad, Dr. Esam Abdel-Raheem and Dr. Kemal Tepe for their valuable comments and suggestions. Also, thanks to all my colleagues and friends at Computer Vision and Sensing System Laboratory of University of Windsor.

I am indebted to my immediate family and extended family; all of them have patiently supported me throughout my entire academic journey. It is impossible to express my love and appreciation to my wife Dilshad Hussain in a word. Finally my deepest gratitude goes to my father (in the heaven) and mother who have devoted their everything to our education with invariant love and enthusiasm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DVD	Digital Versatile Disk
DSL	Digital Subscriber Line
ITU-T	International Telecommunications Union, Telecommunication Standardization Sector
VCEG	Video Coding Expert Group
ISO	International Organization for Standardization
MPEG	Moving Picture Expert Group
VCD	Video CD
CD-ROM	Compact Disc Read Only Memory
PSTN	Public Switched Telephone Network
AVC	Advanced Video Coding
DMB	Digital Multimedia Broadcasting
DVB-H	Digital Video Broadcasting-Handheld
SD	Standard-Definition
HD	High-Definition
DCT	Discrete Cosine Transform
CABAC	Context-adaptive binary arithmetic coding
RD	Rate-Distortion
RDO	Rate-Distortion Optimization
ME	Motion Estimation
QCIF	Quarter Common Intermediate Format
CIF	Common Intermediate Format
MVD	Motion Vector difference
MPM	Most Probable Mode
BMA	Block-Matching Algorithm
FBMA	Fast Block-Matching Algorithm
IDCP	Improved DC Prediction
SATD	Sum of Absolute Hadamard-Transform Differences

JVT	Joint Video Team
TML	Test Model Long-Term
JM	Joint Model (H.264/AVC)
SVC	Scalable Video Coding
MVC	Multi-view Video Coding
3D	Three Dimension
MB	Macroblock
BP	Baseline Profile
MP	Main Profile
XP	Extended Profile
BDM	Block Distortion Measure
SAD	Sum of Absolute Differences
SSD	Sum of Squared Differences
MV	Motion Vector
QP	Quantization Parameter
MF	Multiplication Factor
VLC	Variable Length Coding
CAVLC	Context-Adaptive Variable Length Coding
FS	Full Search
PDS	Partial Distortion Search
NPDS	Normalized Partial Distortion Search
EPDS	Edge based Partial Distortion Search
3SS	Three Step Search
4SS	Four Step Search
DS	Diamond Search
NDS	New Diamond Search
HEXBS	Hexagon Based Search
CDS	Cross Diamond Search
EHEXBS	Enhanced Hexagon Based Search

APDS	Adjustable Partial Distortion Search
PSNR	Peak Signal to Noise Ratio
HPDS	Hadamard transform based Partial Distortion Search
DHS-NPDS	Dual halfway stop Normalized Partial Distortion Search
FFSSG	Fast Full Search with Sorting by Gradient
TS-EPDS	Two-step Edge based Partial Distortion Search
UMHS	Unsymmetrical-cross Multi-hexagon Grid Search
ET	Early Termination
VBME	Variable Block size Motion Estimation
MVP	Motion Vector Predictor
ASR	Adaptive Search Range
BIP	Bi-directional Intra Prediction
DWP	Distance based Weighted Prediction
IBS	Intra mode Bits Skip
ANM	Adaptive Numbers of Modes
IMBR	Intra Mode Bit Reduction
FHT	Fast Hadamard Transform
SAITD	Sum of Absolute Integer Transform Differences
ESATD	Enhanced Sum of Absolute Integer Transform Differences
IPTV	Internet TV
HEVC	High Efficiency Video Coding

Chapter 1

Introduction

Digital video has taken the place of traditional analogue video in a wide range of applications due to its compatibility with other types of data (such as voice and text). However, digital video contains huge amount of data and despite the increases in processor speed and disc storage capacity. For example, using a video format of 352x240 pixels with 3 bytes of color data per pixel, playing at 30 frames per second, 7.6 Megabytes of disc space is needed for one second of video and it is only feasible to store around 10 minutes of video in a 4.3 Gigabytes Digital Versatile Disk (DVD). When it is transmitted in real time through the internet, it requires a channel with 60.8 mbps. But the data throughput of consumer DSL services typically ranges from 256 Kb/s to 20 Mbit/s in the direction to the customer (downstream), depending on DSL technology, line conditions, and service-level implementation [32].

Video compression technologies are about reducing and removing redundant video data so that a digital video file can be effectively sent over a network and stored on computer disks. With efficient compression techniques, a significant reduction in file size can be

achieved with little or no adverse effect on the visual quality. There are two classes of techniques for image and video compression: lossless coding and lossy coding [1].

Lossless coding techniques compress the image and video data without any loss of information and the compressed data can be decoded exactly the same as original data; however, these techniques obtain a very low compression ratio and result in large files. Consequently, they are appropriate for applications requiring no loss introduced by compression, for example, medical image storage. On the other hand, lossy coding methods sacrifice some image and video quality to achieve a significant decrease in file size and a high compression ratio. Lossy coding techniques are widely used in digital image and video applications due to the high compression ratio provided. The goal of a video compression algorithm is to achieve efficient compression as well as minimizing the distortion introduced by the compression process [2].

1.1 Video compression standards

There are two main standardization bodies for video technology. Firstly, there is a division of the International Telecommunication Union known as the ITU-T. The specific division within the ITU-T which is in charge of multimedia data compression is called Video Coding Experts Group (VCEG). The second organization is the International Organization for Standardization (ISO). Within the ISO, the specific committee which is responsible for data compression is the Moving Picture Experts Group (MPEG) [3].

The ITU-T video coding standards are called recommendations, and they are denoted with H.26x (e.g., H.261, H.262, H.263 and H.264). The ISO/IEC standards are denoted with MPEG-x (e.g., MPEG-1, MPEG-2 and MPEG-4). VCEG and MPEG have researched the video coding techniques for various applications of moving pictures since the early 1985s. Fig. 1.1 shows the progression of video coding standards.

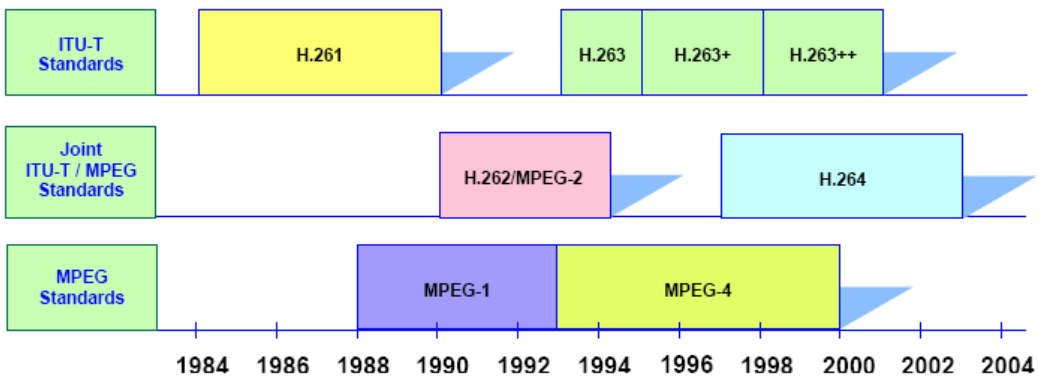


Fig. 1.1 Progression of the ITU-T Recommendations and MPEG standards

Recommendation H.261 [4] was adopted as an international standard by the ITU-T in 1990. It was designed for use in video-conferencing applications at bit rates which are integer multiples of 64 kbps. H.261 operates with very little encoding delay and minimal computing overhead.

In 1992, MPEG released their first standard MPEG-1 [5]. The motivation behind MPEG-1 was the efficient storage and retrieval of video and audio data from a CD-ROM. Video CD (VCD) is a popular implementation of MPEG-1. MPEG-1 also provided the well known audio compression format MPEG-1 Layer 3 (MP3).

In 1994, both VCEG and MPEG jointly developed MPEG-2 [6] as the standard for digital (standard definition) television, and it is currently in widespread use. It was

designed to handle higher resolutions than MPEG-1, as well as interlaced frames (although progressive video is also supported). MPEG-2 borrows many techniques from MPEG-1, with some modifications to handle interlacing. It supports bit-rates in the range of 2 to 10 Mbps. Since MPEG-2 is the standard which is used for encoding Digital Versatile Disks (DVDs), it is perhaps the most widely used multimedia data compression standard.

Following advances in video coding, the ITU-T released H.263 [7] as a standard for use in video telephony in 1995. It is a video coding standard for low bit rate video communication over Public Switched Telephone Network (PSTN) and mobile networks with transmission bit rates of around 10-24kbps or above. H.263 also offers rate, spatial and temporal scalability in a similar way to MPEG-2. Two more versions of H.263 have followed H.263+ [8] in 1998 and H.263++ [9-11] in 2000. While the H.263 series has not been as widely recognized as MPEG, many products use it. For example, many digital cameras use H.263 for capturing video.

The MPEG-4 [12, 13] standard was developed with the goal of being more than just an incremental improvement on the previous two standards. MPEG-4 supports a wide range of bit-rates, but is mainly focused on low bit-rate video. The first version of the standard provides a very low bit-rate video coding, which is actually very similar to baseline H.263 [14]. A fundamental concept in MPEG-4 is the idea of object-based coding. This allows a scene to be described in terms of foreground and background objects, which may be coded independently. However, since the standard only defines how the decoder should operate, there is no prescribed method for the difficult task of

segmenting a scene into its constituent objects. This has resulted in a slow uptake in the use of object-based coding for practical applications.

The ITU-T and ISO established a Joint Video Team to develop a new video compression standard using a “back to basics” approach [15]. In 2003, they proposed the H.264 standard [16-18], which has also been incorporated into MPEG-4 under the name of Advanced Video Coding (AVC). The goal was to compress video at twice the rate of previous video standards while retaining the same picture quality. Due to its improved compression quality, H.264/AVC is quickly becoming the leading standard; it has been adopted in many video coding applications such as the iPod and the Playstation Portable, as well as in TV broadcasting standards such as Digital Video Broadcasting-Handheld (DVB-H) and Digital Multimedia Broadcasting (DMB). Portable applications primarily use the Baseline Profile up to standard-definition (SD) resolutions, while high-end video coding applications such as set-top boxes, Blu-ray and high definition DVD (HD-DVD) use the Main or High Profile at HD resolutions. The Baseline Profile does not support interlaced content; the higher profiles do.

1.2 Statement of the problem

H.264/AVC is the newest international video compression standard. H.264/AVC has been demonstrated to provide significant rate-distortion gains over previous standards, and it is widely accepted as the state-of-the-art in video compression [17, 19]. H.264/AVC has many similar characteristics to previous standards, but some of the main new features are outlined below:

- Up to five reference fames may be used for motion estimation (as opposed to the one or two frames used in previous standards).
- For each 16 × 16 macro-block, variable block size motion estimation is used. This allows a range of different block sizes for motion compensation — from 16 × 16 down to 4 × 4 pixels. Using seven different block sizes and shapes can translate into bit rate savings of more than 15% as compared to using only a 16x16 block size [21]. However, computational complexity of this method is extremely high.
- Motion vectors can be specified to one-quarter pixel accuracy (or one eighth pixel in the case of chrominance components).
- Intra-frame coding is performed using 4×4 blocks, based on a fast integer approximation of the DCT. Spatial prediction within frames is also used to achieve additional de-correlation.
- An adaptive de-blocking filter is used within the motion compensation loop in order to improve picture quality.
- Context-adaptive binary arithmetic coding (CABAC) is employed

All of the above new features improve rate-distortion (RD) performance of the encoder with the expense of extremely high computations. Among the several new features which are introduced by H.264/AVC, the motion estimation (ME) and mode decision process is highly computationally intensive than traditional algorithms. Therefore, the development of efficient algorithms for the ME and mode decision of H.264/AVC is one of the most challenging themes. Table 1.1 shows the total encoding time 100 frames of eight QCIF (176x144) sequences in different video coding standards [20]. It is shown

that total encoding time of H.264/AVC baseline profile is around 106 times of H.263+. It is easy to foresee that the computational complexity will further increase dramatically if higher picture resolutions, for instant CIF (352x288) and CCIR-601 (720x480), are required. Therefore, algorithms to reduce the computational complexity of H.264/AVC without compromising the coding efficiency are indispensable for real time implementation.

Table 1.1: Encoding time of different video coding standards [20]

Standard-compliant encoder	Total encoding time (second)
MPEG-1	26.65
H.263	66.37
H.263+	126.70
H.264 Baseline Profile	13387.83
H.264 Main Profile	19264.53
H.264 Extended Profile	20713.22

The main purpose of this dissertation is to propose some fast motion estimation and mode decision algorithms to reduce the complexity of the H.264/AVC encoder, at the same time without degrading the picture quality as compared to original method.

1.3 Contribution of the dissertation

Within H.264/AVC video coding, motion estimation and rate-distortion optimized mode decisions contributes to the largest gain in compression but both of these are also the most computationally intensive parts. In motion estimation, similarities between different video frames are searched and identified; redundant data are then eliminated or minimized to reduce temporal redundancy within a video sequence. Many fast motion estimation methods have been developed over the last decade, many of these methods come with a complex search flow and with a limited speedup. To reduce the

computation of motion estimation module of H.264/AVC, following contributions has been made.

- This dissertation proposes a novel edge based partial distortion search (EPDS) algorithm [94] which reduces the computation of each distortion measure by using partial distortion search.
- In order to reduce number of search points of EPDS algorithm, two step EPDS is also developed [95].
- An adaptive early termination algorithm is developed to reduce the number of search points that features an adaptive threshold based on the statistical characteristics of rate-distortion (RD) cost function [73].
- A region-based searching strategy based on the orientation of the previously calculated motion vectors is also suggested to further reduce the computation requirement for full search ME [96].
- A search point reduction scheme for the fast motion estimation of H.264/AVC is introduced [73].
- This dissertation also presents a novel adaptive search area selection method by utilizing the information of the previously computed motion vector differences (MVDs) [97].

Rate-distortion optimized intra mode decision is also one of the important coding tools of H.264/AVC encoder. In order to improve performance of conventional intra coding, this dissertation also developed following algorithms.

- An improved DC prediction method for 4x4 intra mode decision is suggested [98].

- In order to reduce the number of overhead bits and computational cost, an intra prediction method is also proposed in this dissertation. In this method, the number of prediction modes for each 4x4 and 8x8 block is selected adaptively [99].
- This thesis also proposes an algorithm to estimate the most probable mode (MPM) of each 4x4 or 8x8 block [100].
- Finally, an enhanced low complexity rate-distortion cost function is introduced for 4x4 intra mode decision of H.264/AVC encoder.

1.4 Organization of the dissertation

In chapter 2, a more detailed survey on the new video coding standard H.264/AVC is provided. In this chapter, H.264/AVC profiles, intra and inter Prediction, variable block size motion estimation, transform coding, quantization, discrete cosine transform (DCT), entropy coding method are discussed. This chapter also reviewed sub pixel motion estimation and rate distortion optimized motion estimation technique adopted by H.264/AVC.

In chapter 3, a novel edge based partial distortion search (EPDS) algorithm which reduces the computation of each distortion measure by using partial distortion search is proposed. In this algorithm, the entire macroblock is divided into different sub-blocks and the calculation order of partial distortion is determined based on the edge strength of sub-blocks. A lossy algorithm is also presented that adaptively changes the early termination threshold for every accumulated partial sum of absolute difference. In this method, only selected numbers of search points are considered for candidate motion

vectors. The proposed method is compared with some well known methods and simulation results are presented.

In chapter 4, an early termination algorithm with adaptive threshold is proposed to reduce the number of search points of motion estimation module of H.264/AVC encoder. The adaptive threshold is developed based on the statistical characteristics of rate-distortion (RD) cost of the current block with that of previously processed block and modes. This chapter also proposed a region based search to further reduce the computation of full search motion estimation. A search point reduction scheme of fast motion estimation of H.264/AVC is also introduced in this chapter.

Chapter 5 presents a novel adaptive search area selection method by utilizing the information of the previously computed motion vector differences (MVDs). The direction of picture movement of the previously computed blocks is also considered for search area selection. In this algorithm, narrow search ranges are chosen for areas in which little motion occurs and wide ranges are chosen for areas of significant motion.

Chapter 6 proposes an improved DC prediction (IDCP) mode based on the distance between the predicted and reference pixels. In order to reduce the number of overhead bits and computational cost, an intra prediction method is also proposed in this chapter. This chapter also proposes an algorithm to estimate the most probable mode (MPM) of each block. A comparison of the rate-distortion performance and speed between different conventional algorithms are also conducted.

In chapter 7, we propose an enhanced low complexity cost function for H.264/AVC intra 4x4 mode selections. Firstly, different fast cost functions recommended by H.264/AVC are presented. Then cause of distortion is analyzed. The enhanced cost

function uses sum of absolute Hadamard-transformed differences (SATD) and mean absolute deviation of the residual block to estimate distortion part of the cost function. A threshold based large coefficients count is also used for estimating the bit-rate part. The proposed cost function is compared with traditional fast cost functions used in H.264/AVC.

Lastly, conclusions and future research directions of this dissertation are given in Chapter 8.

Chapter 2

Overview of H.264/AVC

To provide better compression of video compared to previous standards, H.264/ AVC [16, 27] video coding standard was recently developed by the JVT (Joint Video Team) consisting of experts from VCEG and MPEG. H.264/AVC fulfills significant coding efficiency, simple syntax specifications, and seamless integration of video coding into all current protocols and multiplex architectures. Thus, H.264/AVC can support various applications like video broadcasting, video streaming, video conferencing over fixed, and wireless networks and over different transport protocols [29]. In this chapter, the main features of the H.264/AVC are summarized.

2.1 History

In early 1998 the Video Coding Experts Group (VCEG - ITU-T SG16 Q.6) issued a call for proposals on a project called H.26L [22,23], with the target to double the coding efficiency (which means halving the bit rate necessary for a given level of fidelity) in comparison to any other existing video coding standards for a broad variety of

applications. The first draft (Test Model Long-Term TML-1) design for that new standard was adopted in August 1999 [24]. In December 2001, the Moving Pictures Experts Group (MPEG) joined the standardization process in the Joint Video Team (JVT) to finalize the joint recommendation/standard H.264/AVC. The development started off with the Joint Test Model JM-1 [26]. The JM-1 departed from TML-9 [25]. The status of a Final Draft International Standard was reached in March 2003 [27]. In June 2004, the Fidelity range extensions (FRExt) project was finalized [28]. From January 2005 to November 2007, the JVT was working on an extension of H.264/AVC towards scalability by an Annex (G) called Scalable Video Coding (SVC). From July 2006 to November 2009, the JVT worked on Multiview Video Coding (MVC), an extension of H.264/AVC towards free viewpoint television and 3D television.

2.2 Terminology

Some of the important terminology adopted in the H.264/AVC standard is as follows.

Pictures, frames and fields:

A coded video sequence in H.264/AVC consists of a sequence of *coded pictures.* A coded picture can be represented either an entire *frame* or a single *field* [18]. There are two types of video supported in H.264/AVC; interlaced and progressive [30]. In interlaced video, each video frame is divided into two fields. The first field consists of odd numbered picture line (1,3,5,…) and second field is formed by taking the even lines (2,4,6,….). A progressive video frame is not divided into any sections, the entire picture is coded and transmitted as one unit.

YCbCr color space and 4:2:0 sampling:

The human visual system (HVS) is less sensitive to color than to luminance (brightness) [18]. Video transmission systems can be designed to take advantage of this. The video color space used by H.264/AVC separates a color representation into three components called Y, Cb, and Cr. Component Y is called *luma*, and represents brightness. The two *chroma* components Cb and Cr represent the extent to which the color deviates from gray toward blue and red, respectively. In H.264/AVC as in prior standards, a YCbCr color space is used to reduce the sampling resolution of the Cb and Cr chroma information [17].

4:4:4 sampling means that the three components (Y, Cb and Cr) have the same resolution and hence a sample of each component exists at every pixel position. In the popular 4:2:0 sampling format , *Cb* and *Cr* each have half the horizontal and vertical resolution of *Y* . Because each color difference component contains one quarter of the number of samples in the Y component, 4:2:0 YCbCr video requires exactly half as many samples as 4:4:4 video. H.264 supports coding and decoding of 4:2:0 progressive or interlaced video and the default sampling format is 4:2:0 progressive frames.

Example

Image resolution: 720 × 576 pixels, Y resolution: 720 × 576 samples, each represented with eight bits

4:4:4 Cb, Cr resolution: 720 × 576 samples, each eight bits, total number of bits: 720 × 576 × 8 × 3 = 9 953 280 bits

4:2:0 Cb, Cr resolution: 360 × 288 samples, each eight bits, total number of bits: (720 × 576 × 8) + (360 × 288 × 8 × 2) = 4 976 640 bits

The 4:2:0 version requires half as many bits as the 4:4:4 version.

Macroblock and slices:

H.264/AVC uses block based coding schemes. In these schemes, the pictures are sub-divided into smaller units called macroblocks that are processed one by one, both by the decoder and the encoder. A macroblock (MB) contains coded data corresponding to a 16 × 16 sample region of the video frame (16 × 16 luma samples, 8 × 8 Cb and 8 × 8 Cr samples) . MBs are numbered (addressed) in raster scan order within a frame.

A video picture is coded as one or more slices, each containing an integral number of MBs from 1 (1 MB per slice) to the total number of MBs in a picture (1 slice per picture). The number of MBs per slice need not be constant within a picture. There is minimal inter-dependency between coded slices which can help to limit the propagation of errors. There are five types of coded slice and a coded picture may be composed of different types of slices [18].

- I (Intra) slice: Contains only I MBs. Each block or all MBs is predicted from previously coded data within the same slice.
- P (Predicted) slice: In addition to the coding types of the I slice, some MBs of the P slice can also be coded using inter prediction with at most *one* motion compensated prediction signal per prediction block.

- B (Bi-predictive): In addition to the coding types available in a P slice, some MBs of the B slice can also be coded using inter prediction with *two* motion compensated prediction signals per prediction block.
- SP (Switching P): A so-called switching P slice that is coded such that efficient switching between different precoded pictures becomes possible [31].
- SI (Switching I): A so-called switching I slice that allows an exact match of a MB in an SP slice for random access and error recovery purposes [31].

2.3 H.264/AVC Profiles

While H.264/ AVC standard contains a rich set of video coding tools, not all the coding tools are required for all applications. For example, error resilience tools may not be needed for video stored on a compact disk or on networks with very few errors [33]. Therefore, the standard defines subsets of coding tools intended for different classes of applications. These subsets are called Profiles. There are three Profiles in the first version: Baseline (BP), Main (MP), and Extended (XP) [29]. Baseline Profile is to be applicable to real-time conversational services such as video conferencing and videophone. Main Profile is designed for digital storage media and television broadcasting. Extended Profile is aimed at multimedia services over Internet. Also there are four High Profiles defined in the fidelity range extensions [34] for applications such as content-contribution, content-distribution, and studio editing and post-processing: High (Hi), High 10 (Hi10), High 4:2:2 (Hi422), and High 4:4:4 (Hi444). High Profile is to support the 8-bit video with 4:2:0 sampling for applications using high resolution.

High 10 Profile is to support the 4:2:0 sampling with up to 10 bits of representation accuracy per sample. High 4:2:2 Profile is to support up to 4:2:2 chroma sampling and up to 10 bits per sample. High 4:4:4 Profile is to support up to 4:4:4 chroma sampling, up to 12 bits per sample, and integer residual color transform for coding RGB signal. Table 2.1 shows the features supports in different profiles.

Table 2.1 Features of different profiles

Feature	BP	MP	XP	Hi	Hi10	Hi422	Hi444
B slices	No	Yes	Yes	Yes	Yes	Yes	Yes
SI and SP slices	No	No	Yes	No	No	No	No
Flexible macroblock ordering (FMO)	Yes	No	Yes	No	No	No	No
Data partitioning	No	No	Yes	No	No	No	No
Interlaced coding	No	Yes	Yes	Yes	Yes	Yes	Yes
CABAC entropy coding	No	Yes	No	Yes	Yes	Yes	Yes
8×8 vs. 4×4 transform adaptivity	No	No	No	Yes	Yes	Yes	Yes
Quantization scaling matrices	No	No	No	Yes	Yes	Yes	Yes
Separate C_b and C_r QP control	No	No	No	Yes	Yes	Yes	Yes
Monochrome (4:0:0)	No	No	No	Yes	Yes	Yes	Yes
Chroma formats	4:2:0	4:2:0	4:2:0	4:2:0	4:2:0	4:2:0/4:2:2	4:2:0/4:2:2/4:4:4
Sample depths (bits)	8	8	8	8	8 to 10	8 to 10	8 to 14
Separate color plane coding	No	No	No	No	No	No	Yes
Predictive lossless coding	No	No	No	No	No	No	Yes

2.4 Block diagram of H.264/AVC

H.264/AVC uses hybrid video scheme [18][35] structure. Fig. 2.1 shows the block diagram of H.264/AVC codec. An input frame F_n is processed in units of a macroblock. Each macroblock is coded in intra or inter mode and, for each block in the macroblock, a predicted block P is formed from samples in the current slice that have previously encoded, decoded and reconstructed (uF'n). In inter mode, prediction is formed by motion-compensated prediction from previous reconstructed frames. Motion estimation and motion compensation [36][37] play very important roles in the hybrid coding scheme, since they can greatly reduce the temporal redundancy between adjacent video frames. Generally speaking, the adjacent pictures always share much similar MBs; thus, the current MB can be presented by the other one in the previous frame with very little difference.

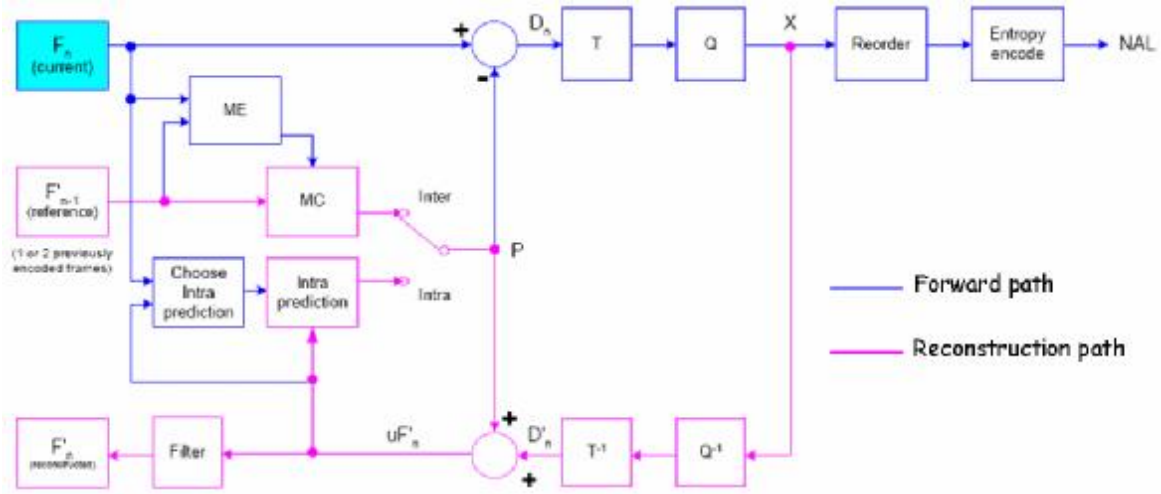


Fig. 2.1 Block diagram of H.264/AVC encoder

The predicted block is subtracted from the current block to produce residual block D_n. The residual data is then discrete cosine transformed (DCT) [38], [39] and quantized for lossy compression to give X which contains the quantized DCT coefficients [40]. The quantized transform coefficients are recorded and then entropy coded.

The encoder decodes a macroblock to provide a reference for further predictions. The coefficients X are scaled (Q^{-1}) and inverse transformed (T^{-1}) to produce a difference block D'_n. The predicted block P is added to D'_n to create a reconstructed block uF'_n. A filter is applied to reduce the effect of blocking distortion and reconstructed reference picture is created from a series of block F'_n.

2.5 Intra Prediction

Intra coding refers to the case where only spatial redundancies within a video picture are exploited. The resulting frame is referred to as an I-frame. I-frames are typically encoded by directly applying the transform to the different macroblocks (MBs) in the frame. Consequently, encoded I-pictures are large in size since a large amount of information is usually present in the frame, and no temporal information is used as part of the encoding process. In order to increase the efficiency of the intra coding process in H.264/AVC, spatial correlation between adjacent MBs in a given frame is exploited [21]. The idea is based on the observation that adjacent MBs tend to have similar properties. Therefore, as a first step in the encoding process for a given MB, one may predict the MB of interest from the surrounding MBs (typically the ones located on top and to the left of the MB of interest, since those MBs would have already been encoded). The difference between the actual MB and its prediction is then coded, which results in fewer bits to represent the MB of interest as compared to when applying the transform directly to the MB itself.

For the luma samples, the prediction block may be formed for each 4x4 subblock, each 8x8 block, or for a 16x16 macroblock [29]. One case is selected from a total of 9

prediction modes for each 4x4 and 8x8 luma blocks; 4 modes for a 16x16 luma block; and 4 modes for each chroma block.

M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

Fig. 2.2 (a) Prediction samples of a 4x4 block

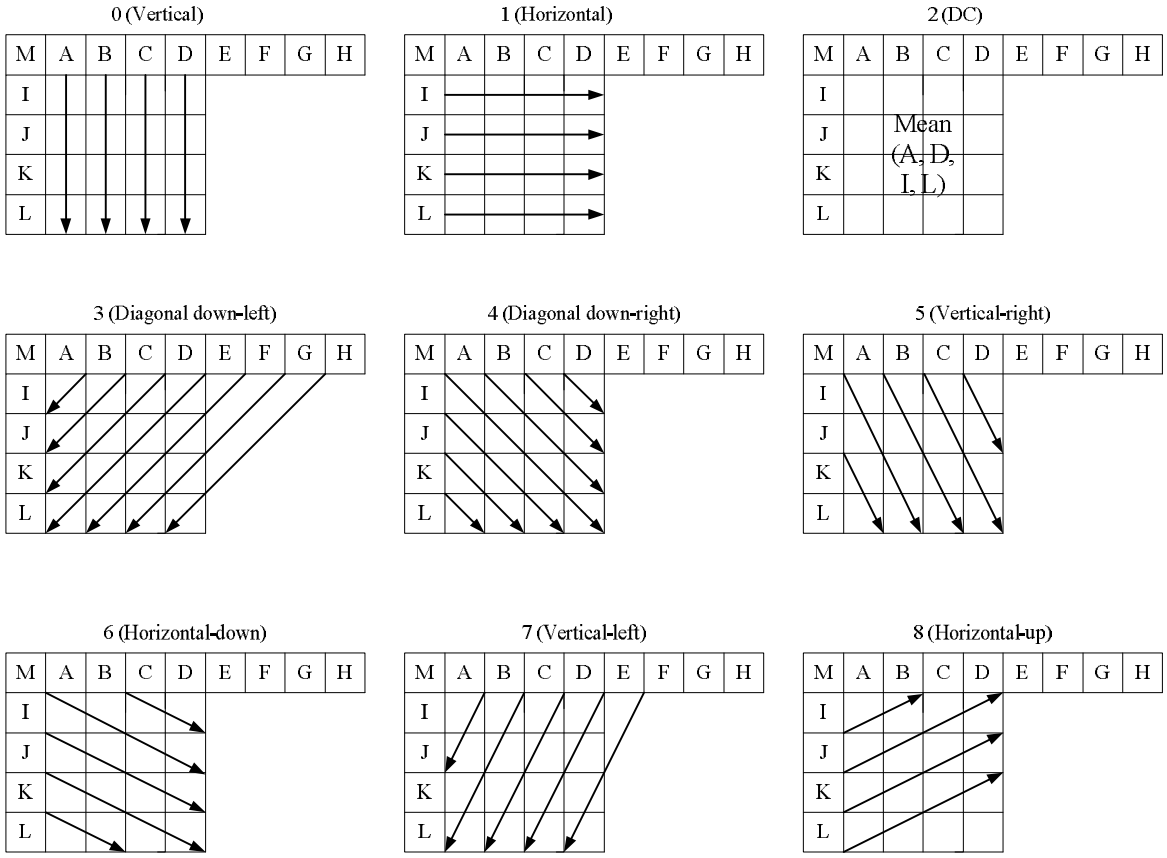


Fig. 2.2 (b) Nine prediction mode of a 4x4 block

Table 2.2 Nine intra 4x4 prediction modes

Mode 0 (Vertical)	The upper samples A, B, C, D are extrapolated vertically.
Mode 1 (Horizontal)	The left samples I, J, K, L are extrapolated horizontally.
Mode 2 (DC)	All samples in P are predicted by the mean of samples A . . . D and I . . . L.
Mode 3 (Diagonal down-left)	The samples are interpolated at a 45◦ angle between lower-left and upper-right.
Mode 4 (Diagonal down-right)	The samples are extrapolated at a 45◦ angle down and to the right.
Mode 5 (Vertical right)	Extrapolation at an angle of approximately 26.6◦ to the left of vertical (width/height = 1/2).
Mode 6 (Horizontal down)	Extrapolation at an angle of approximately 26.6◦ below horizontal.
Mode 7 (Vertical left)	Extrapolation (or interpolation) at an angle of approximately 26.6◦to the right of vertical.
Mode 8 (Horizontal up)	Interpolation at an angle of approximately 26.6◦ above horizontal.

2.5.1 Intra 4×4 Prediction

Fig. 2.2 (a) shows a 4x4 luma block that is to be predicted. For the predicted samples [a, b, . . . ,p] of the current block, the above and left previously reconstructed samples [A, B, . . . ,M] are used according to direction modes. The arrows in Fig. 2.2 (b) indicate the direction of prediction in each mode.

For mode 0 (vertical) and mode 1 (horizontal), the predicted samples are formed by extrapolation from upper samples [A, B, C, D] and from left samples [I, J, K, L], respectively. For mode 2 (DC), all of the predicted samples are formed by mean of upper and left samples [A, B, C, D, I, J, K, L]. For mode 3 (diagonal-down-left), mode 4 (diagonal-down-right), mode 5 (vertical-right), mode 6 (horizontal-down), mode 7 (vertical-left), and mode 8 (horizontal-up), the predicted samples are formed from a

weighted average of the prediction samples A–M. For example, in the case where Mode 3 (Diagonal-Down-Left prediction) is chosen, the values of a to p are given as follows:

- a is equal to *(A+2B+C+2)/4*
- b, e are equal to *(B+2C+D+2)/4*
- c, f, i are equal to *(C+2D+E+2)/4*
- d, g, j, m are equal to *(D+2E+F+2)/4*
- h, k, n are equal to *(E+2F+G+2)/4*
- l, o are equal to *(F+2G+H+2)/4*, and
- p is equal to *(G+3H+2)/4*.

The remaining modes are defined similarly according to the different directions as shown in Fig. 2.2(b) and Table 2.2. Note that in some cases, not all of the samples above and to the left are available within the current slice: in order to preserve independent decoding of slices, only samples within the current slice are available for prediction. The encoder may select the prediction mode for each block that minimizes the residual between the block to be encoded and its prediction.

2.5.2 Intra 8×8 Prediction

Similar to intra 4x4 block, 8x8 luma block also has 9 prediction mode based on the direction of Fig. 2.2 (b). For prediction of each 8x8 luma block, one mode is selected from the 9 modes, similar to the 4x4 intra-block prediction.

2.5.3 Intra 16×16 Prediction

In Intra 16x16 block, there are four prediction modes: Vertical, Horizontal, DC and Plane prediction, which are listed in Fig. 2.3 and Table 2.3. The 16x16 intra prediction works well in a gently changing area.

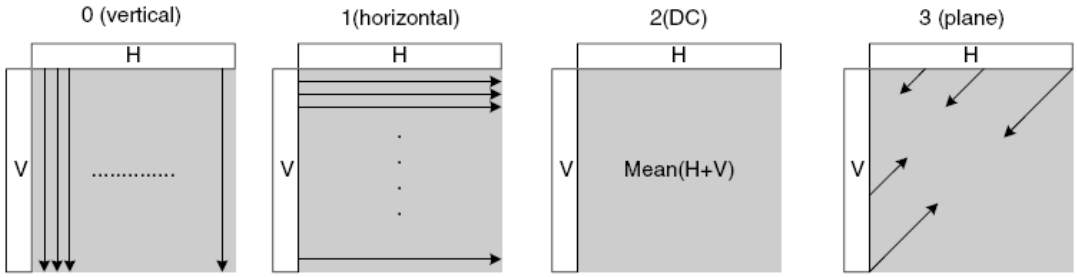


Fig. 2.3 Intra 16x16 prediction modes

Table 2.3 Four intra 16x16 prediction modes

Mode 0 (Vertical)	Extrapolation from upper samples (H)
Mode 1 (Horizontal)	Extrapolation from left samples (V)
Mode 2 (DC)	Mean of upper and left-hand samples (H + V)
Mode 3 (Plane)	A linear 'plane' function is fitted to the upper and left-hand samples H and V. This works well in areas of smoothly-varying luminance.

2.5.4 Intra Croma Prediction

Each chroma component of a macroblock is predicted from chroma samples above and/or to the left that have previously been encoded and reconstructed. The chroma prediction is defined for three possible block sizes, 8x8 chroma in 4:2:0 format, 8x16 chroma in 4:2:2 format, and 16x16 chroma in 4:4:4 format [29]. The 4 prediction modes for all of these cases are very similar to the 16x16 luma prediction modes, except that the order of mode numbers is different: mode 0 (DC), mode 1 (horizontal), mode 2 (vertical), and mode 3 (plane).

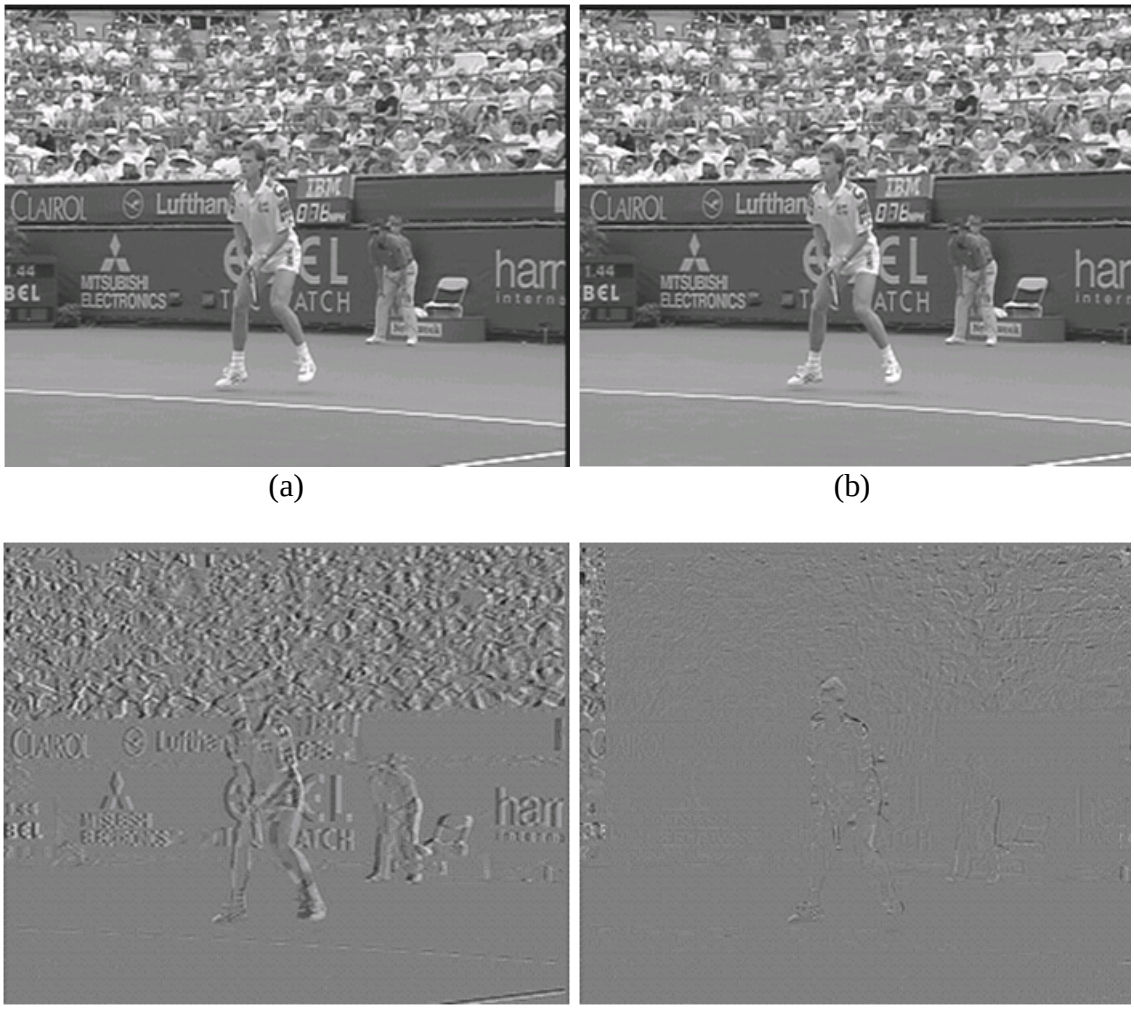


Figure 2.4 The luminance component of 'Stefan' at (a) frame 70, and (b) frame 71. The residual pictures by subtracting the frame 70 from frame 71 (c) without motion compensation, and (d) with motion compensation.

2.6 Inter Prediction

Inter-prediction is used to reduce the temporal correlation with help of motion estimation and compensation. Motion estimation and compensation play an important role in video compression. In general, they can improve the compression efficiency by utilizing the temporal redundancy between adjacent pictures. As depicted in Figure 2.4,

the pictures in (a) and (b) are the frame 70 and 71 captured from the sequence ‘Stefan’. To observe the two pictures carefully, we can see that the frame 71 has a small amount of shift to the left due to the camera panning. Though the two pictures are very similar, if the frame 71 is encoded with reference to frame 70, a large amount of residual data will be left without motion compensation, as shown in Figure 2.4 (c). However, if the panning motion is compensated by finding the displacement with motion estimation, the remaining residual data will be very little, as shown in Figure 2.4 (d). For this reason, the necessary information to be coded is substantially reduced, and the huge volume of video data can be effectively compressed in a very high compression ratio.

2.6.1 Basic assumptions of motion estimation

Motion estimation is a procedure to locate an object of a current frame from a reference frame. The object size can be as small as a pixel, or as large as a frame, but typically a rectangular block of medium size is used. The shift of the object is basically induced by the motion field which can be estimated by the information in spatial and temporal domains, such as the variances of illumination, the orientation of edges, the distribution of colors, and so on [20]. In general, there are some basic assumptions that most motion estimation algorithms count on:

1. the illumination is constant along the motion path; and
2. the occlusion problem is not present.

These two assumptions confine the complex interactions between motion and illumination to a simple model. The former ignores the problem of illumination changing over time that causes optical flow but not necessary for motion. The latter

neglects the typical problems of scene change and uncovered background in which the optical flow is interrupted and not exist for reference. Although the simplified model is not perfect for real-world video contents, the assumptions still hold in most cases.

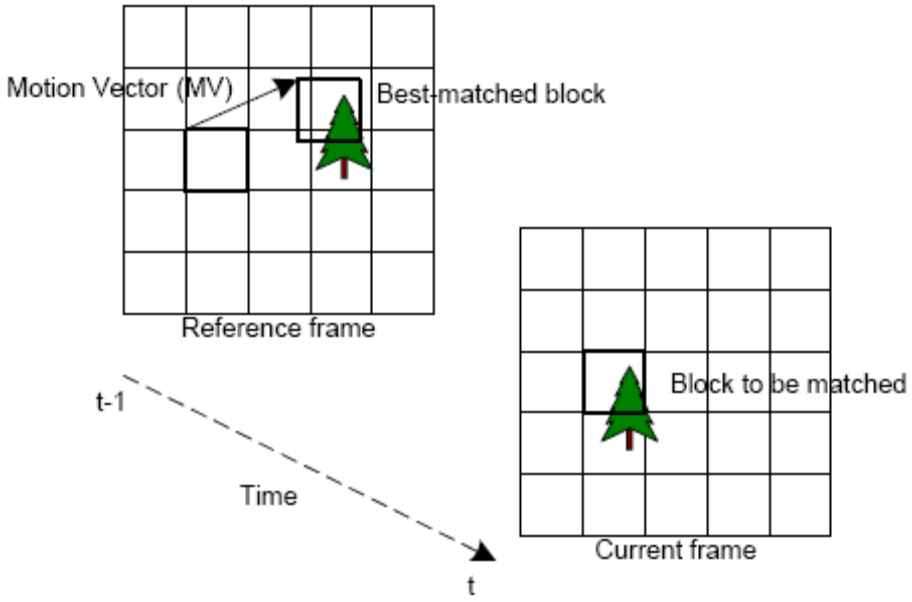


Figure 2.5 Block matching motion estimation.

2.6.2 Block based Motion Estimation

Block based motion estimation is the most popular and practical motion estimation method in video coding. Standards like the H.26X series and the MPEG series use block based motion estimation. Fig. 2.5 shows how it works. Each frame is divided into square blocks. For each block in the current frame, a search is performed on the reference frame to find a matching based on a block distortion measure (BDM). One of the most popular BDMs is the sum of absolute differences (SAD) between current block and candidate block. The motion vector (MV) is the displacement from the current block to the best-matched block in the reference frame. Usually a search window is defined to

confine the search. Suppose a MB has size $N \times N$ pixels and the maximum allowable displacement of a MV is ± w pixels in both horizontal and vertical directions, there are $(2w+1)^2$ possible candidate blocks inside the search window. Fig. 2.6 shows a search point inside a search window.

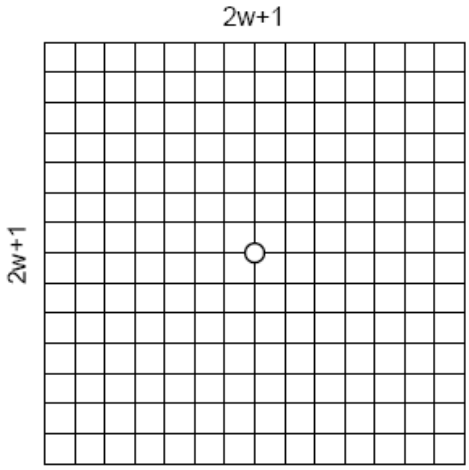


Figure 2.6 a search point in a search window

A matching between the current MB and one of the candidate MBs is referred as a point being searched in the search window. If all the points in a search window are searched, the finding of a global minimum point is guaranteed. The MV pointing to this point is the optimum MV because it provides the optimum block distortion measure (BDM). This is the simplest block matching algorithm (BMA) and is named Full Search (FS) or exhaustive search. To calculate the BDM of 1 search point using Sum of Absolute Differences (SAD), for a MB of size 16×16 pixels, 3×16×16-1=767 operations are needed (subtract, absolute, and add for each pixel). If FS is used to search all the points in a search window of size ± 7 pixels, total of $(7\times2+1)^2 \times767 = 172575$ operations will be needed for one single MB. For newer video coding standards such as the

H.264/AVC, which uses variable block-size encoding and multiple reference frames, the number of operations for motion estimation will be even larger.

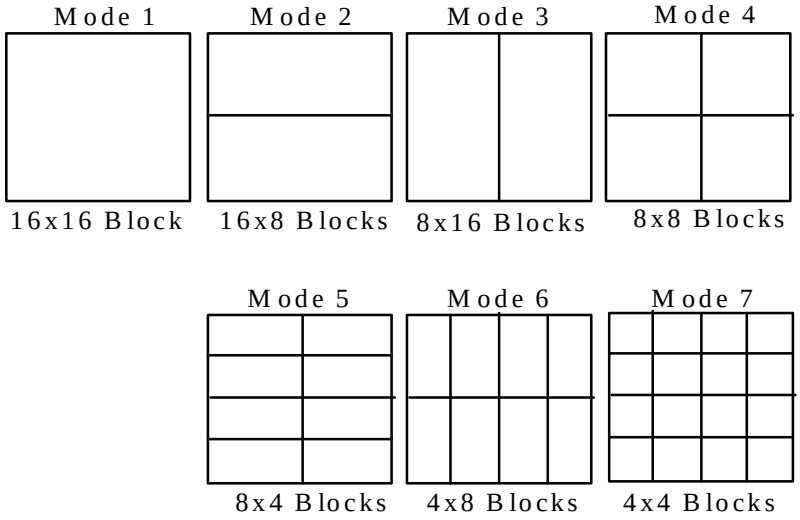


Fig. 2.7 Block sizes for motion estimation of H.264/AVC

2.6.3 Variable Block size Motion Estimation

In order to best represent the motion information, H.264/AVC allows partitioning a macroblock (MB) into several blocks with variable block size, ranging from 16 pixels to 4 pixels in each dimension. For example, one MB of size 16x16 may be kept as is, decomposed into two rectangular blocks of size 8x16 or 16x8, or decomposed into four square blocks of size 8x8. If the last case is chosen (i.e. four 8x8 blocks), each of the four 8x8 blocks can be further split to result in more sub-macroblocks. There are four choices again, i.e. 8x8, 8x4, 4x8 and 4x4. The possible modes of different block sizes are shown in Fig. 2.7. Each block with reduced size can have its individual motion vectors to estimate the local motion at a finer granularity. Though such finer block sizes incur overhead such as extra computation for searching and extra bits for coding the

motion vectors, they allow more accurate prediction in the motion compensation process and consequently the residual errors can be considerably reduced, which are usually favorable for the final RD performance.

2.6.4 Sub-Pixel Motion Estimation

The inter-prediction process can form segmentations for motion representation as small as 4 x 4 luma samples in size, using motion vector accuracy of one-quarter of the luma sample. Sub-pel motion compensation can provide significantly better compression performance than integer-pel compensation, at the expense of increased complexity. Quarter-pel accuracy outperforms half-pel accuracy. Especially, sub-pel accuracy would increase the coding efficiency at high bitrates and high video resolutions. In the luma component, the sub-pel samples at half-pel positions are generated first and are interpolated from neighboring integerpel samples using a 6-tap FIR filter with weights (1, -5, 20, 20, -5, 1)/32. Once all the half-pel samples are available, each quarter-pel sample is produced using bilinear interpolation between neighboring half- or integer-pel samples. For 4:2:0 video source sampling, 1/8 pel samples are required in the chroma components (corresponding to 1/4 pel samples in the luma). These samples are interpolated (linear interpolation) between integer-pel chroma samples. Sub-pel motion vectors are encoded differentially with respect to predicted values formed from nearby encoded motion vectors. Detail of sub-pel motion estimation is found in reference [17] and [18].

2.6.5 Multiple Reference Frame Motion Compensation

As the name implies, this concept uses more than one reference frame for prediction. In H.264/AVC, each MB can be predicted using any previously decoded frame in the sequence, which enlarged the search space two to five times. This new feature is very effective for inter frame prediction of the following cases:

1. Motion that is periodic in nature. For example, a flying bird with its wings going up and down. The wings are best predicted from a picture where they are in similar position, which is not necessarily the preceding picture.
2. Alternating camera angles that switch back and forth between two different scenes.
3. Occlusions: once an object is made visible after occlusion, it is beneficial to do prediction from the frame where the object was last visible.

2.6.6 Motion vector prediction

Since the encoder and decoder both have access to the same information about the previous motion vectors, the encoder can take advantage of this to further reduce the dynamic range of the motion vector it sends. Encoding a motion vector for each partition can cost a significant number of bits, especially if small partition sizes are chosen. Motion vectors for neighboring partitions are often highly correlated and so each motion vector is predicted from vectors of nearby, previously coded partitions. A predicted vector, MVp, is formed based on previously calculated motion vectors and motion vector difference (MVD), the difference between the current vector and the

predicted vector, is encoded and transmitted. The method of forming the prediction MVp depends on the motion compensation partition size and on the availability of nearby vectors [17].

2.6.7 Rate-distortion optimized Motion estimation

Earlier encoders typically computed the sum of absolute differences (SAD) between the current block and candidate blocks and selected simply the motion vector (MV) yielding the least distortion. However, this often will not give the best image quality for a given bit rate, because it may select long motion vectors that need many bits to transmit. It also does not help determining how subdivision should be performed, because the smallest blocks will always minimize the distortion, even if the multiple MVs may use larger amount of bits and increase the bit rate. For this reason, H.264/AVC uses the cost function J, rather than SAD, as the measure of prediction error in selecting the best matching block [19, 41, 42]. The RD cost function J is defined as

$$J(mv, l) = SAD(s, c(mv)) + l\, R(mv - pmv) \tag{2.1}$$

where $mv = (mv_x, mv_y)^T$ is the current MV, $pmv = (pmv_x, pmv_y)^T$ is the predicted MV, and $SAD(s, c(mv))$ is the sum of absolute differences between current block s and candidate block c for a given motion vector mv, l is the Lagrangian multiplier which is a function of quantization parameter (QP) and $R(mv - pmv)$ is the number of bits to code the MVD. In H.264, the Lagrange multiplier for motion estimation is empirically calculated by the following formula [43]:

$$l = 0.92 \times 2^{(QP-12)/6} \tag{2.2}$$

2.7 Integer Transform and Quantization

Similar to previous video coding standards, H.264/AVC utilizes transform coding of the prediction residual. However, in H.264/AVC, the transformation is applied to 4x4 blocks, and instead of a 4x4 discrete cosine transform (DCT), a separable integer transform with similar properties as a 4x4 DCT is used [18]. Let us assume **X** is a 4x4 residual block, then the forward transform matrix **Y** is computed as follows [17]:

$$Y = W \otimes \boldsymbol{E}_f \tag{2.3}$$

$$\text{with} \quad W = C_f X C_f^T \tag{2.4}$$

Where $\otimes$ indicates that each element of **W** is multiplied by the scaling factor in the same position in matrix E_f. C_f is integer transform matrix and E_f is the scaled matrix which are defined as follows

$$\boldsymbol{C}_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{E}_f = \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

with $a = \frac{1}{2}$, $b = \sqrt{\frac{2}{5}}$

The basic forward quantiser [17] operation is:

$$Z_{ij} = round\left(\frac{Y_{ij}}{Q_{step}}\right) \tag{2.5}$$

where Y_{ij} is a coefficient of the transformed block computed by (2.3), Q_{step} is a quantizer step size and Z_{ij} is a quantised coefficient. The rounding operation here need not round to the nearest integer; for example, biasing the 'round' operation towards smaller integers can give perceptual quality improvements. In order to avoid the division and floating point operation of (2.5), quantization operation of H.264/AVC is computed from W_{ij} instead of Y_{ij} as follows [17]

$$|Z_{ij}| = (|W_{ij}|.MF + f) >> qbits \tag{2.6}$$

$$sign(Z_{ij}) = sign(W_{ij}) \tag{2.7}$$

where, >> indicates a binary shift right. W_{ij} is calculated by (2.4) and

$$qbits = 15 + floor(QP/6) \tag{2.8}$$

$$f = \begin{cases} 2^{qbits}/3 \text{ for intra block} \\ 2^{qbits}/6 \text{ for inter block} \end{cases} \tag{2.9}$$

The first six values of multiplication factor MF (for each coefficient position) used by the H.264/AVC reference software encoder are given in Table 2.4. MF doubles in size for every increment of 6 in quantization parameter (QP).

Table 2.4 Multiplication factor MF [17]

QP	MF		
	Positions (i,j) (0,0), (2,0), (2,2), (0,2)	Positions (i,j) (1,1), (1,3), (3,1), (3,3)	Other positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

2.8 Entropy Coding

In H.264/AVC, two methods of entropy coding are supported [44]. The simpler entropy coding method uses a single infinite-extent codeword table for all syntax elements except the quantized transform coefficients. Thus, instead of designing a different variable length coding (VLC) table for each syntax element, only the mapping to the single codeword table is customized according to the data statistics. The single codeword table chosen is an exp-Golomb code with very simple and regular decoding properties. For transmitting the quantized transform coefficients a more efficient method called Context-Adaptive Variable Length Coding (CAVLC) [45] is employed. In this scheme, VLC tables for various syntax elements are switched depending on already transmitted syntax elements. Since the VLC tables are designed to match the corresponding conditioned statistics, the entropy coding performance is improved in comparison to schemes using a single VLC table.

The efficiency of entropy coding can be improved further if the Context-Adaptive Binary Arithmetic Coding (CABAC) is used [46]. Encoding with CABAC consists of three stages—binarization, context modeling and adaptive binary arithmetic coding. Fig. 2.8 shows a high level block diagram of CABAC encoder showing these various stages and their interdependence.

CABAC uses four basic types of tree structured codes tables for binarization. Since these tables are rule based, they do not need to be stored. The four basic types are the unary code, the truncated unary code, the kth order exp-golomb code, and, the fixed-length code.

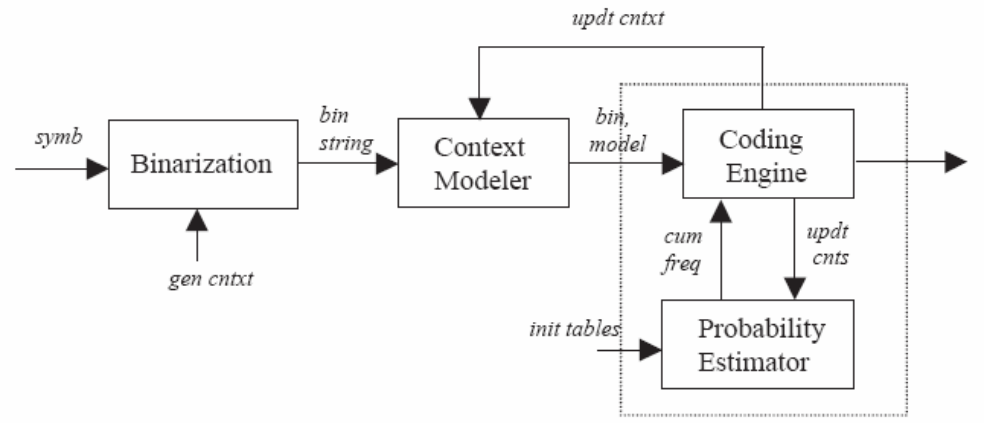


Fig. 2.8 CABAC encoder block diagram

CABAC also uses four basic types of context models based on conditional probability. The first type uses a context template that includes up to two past neighbors to the syntax element currently being encoded. For instance modeling may use a neighbor immediately before and an immediately above the current element, and further, the modeling function may be based on bin-wise comparison with neighbors. The second type of context model is used only for syntax elements of MB type and sub-MB type and uses prior coded i-th bins for coding of i-th bin. The third and fourth types of context models are used for residual data only and are used for context categories of different block types. The third type does not rely on past coded data but on the position in the scanning path, and the fourth type depends on accumulated levels.

The arithmetic coding engine used is similar to that used by other arithmetic coders and generally involves, based on the probability of a symbol to be coded to perform recursive subdivision to fractional accuracy of an existing interval that initially spans the range from 0 to 1. Since the multiplication operation used in interval subdivision is a reason for increased complexity of arithmetic coders, a multiplication free state transition table based approach is used by CABAC.

Chapter 3

Edge based Partial Distortion Search

In video coding, block based full search motion estimation algorithm has been widely used, but it suffers from high computational requirements. In order to reduce the computations, this chapter proposes a novel edge based partial distortion search (EPDS) algorithm which reduces the computation of each distortion measure by using partial distortion search. In this algorithm, the entire macroblock (MB) is divided into different sub-blocks and the calculation order of partial distortion is determined based on the edge strength of sub-blocks. This algorithm adaptively changes the early termination threshold for every accumulated partial sum of absolute difference. In the proposed method, only selected numbers of search points are considered for candidate motion vectors. Simulation results show that the proposed method offers a remarkable improvement in computational speed when compared to full search (FS) and normalized partial distortion search (NPDS) algorithms.

3.1 Literature Review

Motion estimation (ME) has a key role in compression of video sequences. Motion estimation and motion compensation have been widely used in many video coding standards, such as MPEG-1/2/3 and H.261/263/264, to remove the temporal redundancy between successive frames in video sequences. Many different motion estimation methods are investigated and reported in the literature. Among them, block matching algorithms (BMAs) are the most popular motion estimation methods owing to their simplicity and effectiveness. The direct BMA is the full search (FS) algorithm, which exhaustively searches all possible locations within the search window to find the best-matched block. The full search algorithm generally obtains optimum results, but it suffers from heavy computational load. Therefore, the development of an efficient and fast algorithm for ME is one of the most challenging themes. Various algorithms have been proposed to reduce the computational complexity of ME module such as three step search (3SS) [47], diamond search (DS) [48], 2-d logarithm search [37], and four step search (4SS) [68]. These algorithms utilize monotonic property of block distortion measure (BDM) value used to estimate motion vector. However, this property does not always hold real-world video sequences, especially with high and irregular motion videos at low frame rates. As a consequence, the video quality degradation introduced by these algorithms is relatively high [50]. Recently many BMAs utilize stationary or quasi-stationary characteristics of video sequences i.e. neighbouring pictures vary slowly so many motion vectors (MVs) are either zero or within ± 1 pixels. Well known examples are new diamond search (NDS) [69], [48], hexagon based search (HEXBS)

[70], enhanced HEXBS (EHEXBS) [71], and cross-diamond search (CDS) [72]. One way to reduce the complexity of ME process is to terminate the ME calculation early. Efforts have been made to explore the early termination algorithms in motion estimation [53, 73, 65, 64, 74]; in [53] a zero-motion detection algorithm compares sum of absolute difference (SAD) of two blocks with a predefined threshold to determine the stationary block and skip the remaining search points. A fast motion estimation algorithm by adaptively changing the early termination threshold for H.264/AVC video coding standard is proposed in [73]. A number of algorithms [75-79] were presented in literature to make the ME module faster.

Another approach for the fast algorithms is focused on reducing complexity during the calculation of the matching criterion in each macroblock. The partial distortion search (PDS) [80] algorithm reduces the computational complexity by terminating the measuring distortion (sum of absolute difference: SAD) calculation early when it finds that a partial SAD is greater than the minimum SAD encountered so far during search. In normalized partial distortion search (NPDS), partial distortion and the current minimum distortion are normalized on the number of checked pixels before comparison [81]. However, NPDS has a limitation on the maximum computational reduction and visual quality performance. In order to improve the efficiency of PDS algorithms, several approaches are presented in the literature. Cheng and Po [82] extended NPDS to an adjustable PDS (APDS) so that it is capable of adjusting the prediction accuracy against the searching speed by a quality factor. Simulations show that APDS reduces computation up to 38 times with a relatively larger degradation in PSNR performance, when compared to that of full search.

The enhanced normalized partial distortion search based on the block similarity is presented in [83] and an adaptive matching scan strategy to quickly reject unnecessary candidates is introduced in [84]. A sorting based partial distortion based motion estimation in presented in [85], wherein pixel positions are sorted according to the gradient magnitude between adjacent pixels. Since gradient is performed by pixel by pixel operation, it involves more overhead computations. A clustered pixel matching error for adaptive PDS is proposed in [86] in which matching order is determined by using the characteristics of clustered pixels matching error. A two stage sorting based PDS algorithm by using the characteristic of pattern similarity matching error is described in [87, 88]. The basic idea in [89] is to eliminate invalid candidates at an early stage by predicting a total matching error between matching and candidate blocks. However this prediction is based on the linear model which results in significant degradation of image quality. A ME algorithm that adaptively calculates the early termination threshold for the accumulated SAD is developed in [59] in order to reduce the complexity of H.264/AVC. A Hadamard transform based partial distortion search algorithm (HPDS) [90] is proposed that utilizes sum of DC and AC Hadamard transform coefficients in order to determine the calculation order of partial differences. Hadamard transform results in large computational cost that renders the algorithm inefficient for real time implementation. The dual halfway stop normalized PDS (DHS-NPDS) is introduced in [93] and simulation results shows that the PSNR performance degradation of DHS-NPDS algorithm is large in comparison to full search especially for high motion video sequences.

3.2 Partial Distortion Search (PDS)

The full search algorithm finds the most similar block to a matching block within a given search range of the reference frames. The similarity between blocks is measured by the block matching error and is often computed by the SAD between matching and candidate block. SAD is defined as

$$SAD(x, y; mx, my) = \sum_{i=1}^{M} \sum_{j=1}^{M} \left| f_t(x+i, y+j) - f_{t-1}(x+i+mx, y+j+my) \right| \tag{3.1}$$

where $M \times M$ is the size of the block, M is equal to 16 for a MB, (x,y) is the position of the MB being coded, *mx* and *my* are the horizontal and vertical component of candidate motion vector (MV) and $f_t(.,.)$ and $f_{t-1}(.,.)$ represent the luminance pixel intensity of the current frame and reference frame, respectively. The full search algorithm for ME can give an optimum solution by exhaustively searching all possible locations within a search window. The resulting best motion vector (*mvx* ,*mvy*) is

$$(mvx, mvy) = \arg \min_{(\text{mx, my})} SAD(x, y; mx, my) \tag{3.2}$$

where $-R \le mx, my \le R$ is a set of all possible locations in a search window, and R is the maximum possible displacement of the *MV*.

Partial distortion search is a technique used to introduce early termination in the calculation of SAD [80]. In block-matching ME, a block of $M \times M$ pixels can be divided into a number of small groups, k, such that the distortion of the k-th group, d_k, can be measured. The k-th partial SAD to check during the matching is

$$d_k = \sum_{i=1}^{k} \sum_{j=1}^{M} \left| f_t(x+i, y+j) - f_{t-1}(x+i+mx, y+j+my) \right| \tag{3.3}$$

The partial SAD d_k, which is the matching error accumulated for every period is computed and compared with the minimum SAD already found (with another candidate vector). Once this is larger than the minimum SAD (SAD_{min}) at each period, the candidate block cannot be the most similar block regardless of the rest of the incomplete matching computations. Therefore, the PDS algorithm can find and remove impossible candidates before complete matching error calculation of candidate block. In (3.3), the matching is performed by row by row and the test is performed after every row.

In order to improve the performance of PDS, a sorting based approach called fast full search with sorting by gradient (FFSSG) is introduced in [85]. In this method, the highest contributions to SAD are found early by creating an index set. The average gradient of each pixel (i,j) in a MB is calculated as follows:

$$\left|G[f(i,j)]\right| = \frac{1}{8}\sum_{m=-1}^{1}\sum_{n=-1}^{1}\left|f(i,j) - f(i+m, j+n)\right| \tag{3.4}$$

After calculating the gradient of each pixel, an adaptive index set is derived by sorting all gradient values. The pixel with the highest gradient is accumulated first. After accumulation of every sixteen pixels, the comparison with SAD_{min} is performed. This algorithm reduces about 15% computation of PDS. However, it suffers from overhead computations since gradient calculation of each pixel requires 24 operations (7 additions, 8 subtractions, 1 division, 8 absolute). Although division operations can be implemented by right shift, extra computations are also incurred in sorting of gradient values.

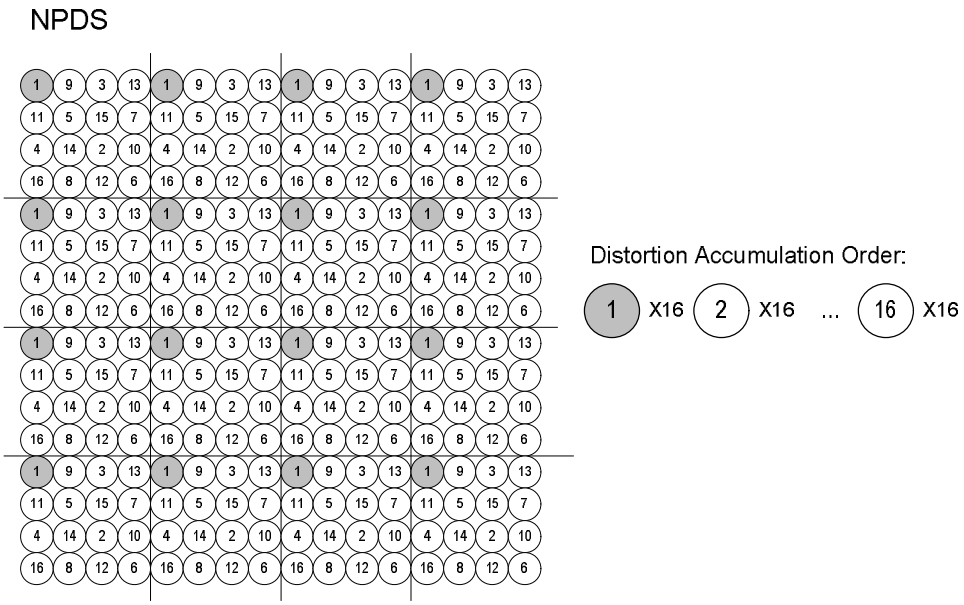


Fig. 3.1 NPDS sub-sampled groups.

3.3 Normalized Partial Distortion Search (NPDS)

Normalized partial distortion search (NPDS) is proposed to push the rejection earlier [81]. A normalized cumulative partial distortion is defined to replace the original cumulative partial distortion d_k as:

$$D_{norm} = d_k \times \frac{16}{k} \tag{3.5}$$

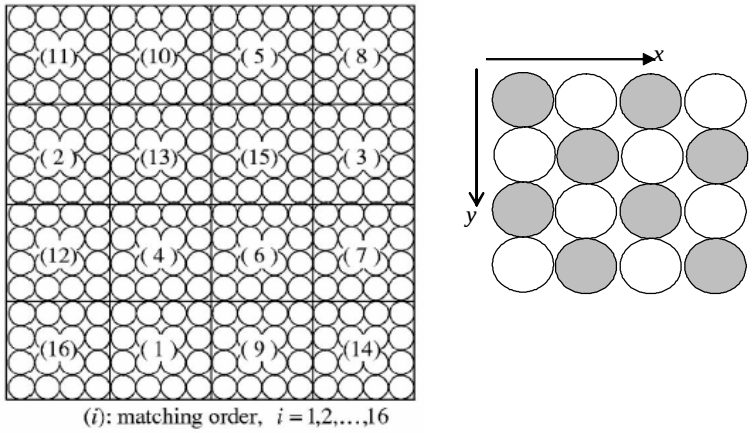
Possibility of rejection in early stages increases greatly after normalization. However, it also introduces erroneous rejection as distortion does not uniformly distribute inside a block. Therefore, NPDS groups pixels in an evenly spaced manner so that the correlation between adjacent pixels is delocalized and a more accurate representation of full distortion is obtained by a partial distortion d_k. Fig. 3.1 demonstrates the grouping of the pixels and the calculation order of the partial distortion d_1 to d_{16}. The macroblock is

divided into sixteen groups, where the group of d_1 is highlighted by the grey color. The group size actually implants the theoretical speedup limit of the algorithm. The earliest rejection only requires summing the distortions of sixteen pixels, and results in an utmost 16 times improvement. About 10 to 12 times computation reduction is practically achieved. However, PSNR performance of the NPDS is not sufficient for video quality.

3.4 Proposed Edge based Partial Distortion Search (EPDS)

3.4.1 EPDS

The matching strategy, that is, the order in which pixels within a block are picked up to compute SAD affects the speed of the ME. In fact if the highest contributions to SAD are found early, then the distortion bound may be reached after small number of differences and partial sum calculation can be stopped. The proposed algorithm divides the entire target MB into different sub-blocks as shown in Fig. 3.2(a). In Fig. 3.2(a), the 16x16 MB is divided into 16 4x4 sub blocks.



(a) (b)

Fig. 3.2 (a) Partition of a MB (b) Pixels of a 4x4 sub-block.

The main idea is based on the assumption that a higher complex sub-block results in a large distortion; sub-blocks with high detail produce large error as compared to homogeneous sub-blocks. In order to classify the complexity of sub-block, we have used edge strength. Fig. 3.2(b) shows the pixels of k-th sub-block. The edge strength is calculated as follows:

Horizontal edge strength,

$$E_{KH} = \left| \sum_{y=1}^{4} I(1,y) - \sum_{y=1}^{4} I(3,y) \right| + \left| \sum_{y=1}^{4} I(2,y) - \sum_{y=1}^{4} I(4,y) \right| \tag{3.6}$$

Vertical edge strength,

$$E_{KV} = \left| \sum_{x=1}^{4} I(x,1) - \sum_{x=1}^{4} I(x,3) \right| + \left| \sum_{x=1}^{4} I(x,2) - \sum_{x=1}^{4} I(x,4) \right| \tag{3.7}$$

Edge strength, $$E_K = \sqrt{E_{KH}{}^2 + E_{KV}{}^2} \tag{3.8}$$

Here *I(x,y)* is the pixel intensity of *(x,y)* position of the k-th sub-block which is clearly shown in Fig. 3.2(b). Calculation of edge strength results in overhead computations and each horizontal and vertical edge strength evaluation requires 17 operations [(6 additions + 1 subtraction + 1 absolute operation)*2 + 1 addition]. In addition to these, the computation intensive square and square root operations are also required for calculation of edge strength in (3.8) for k-th sub-block. In order to reduce the computation of edge strength, we have considered only grey pixels (clearly shown in Fig. 3.2(b)) in calculation of horizontal edge strength and only white pixels in vertical edge strength.

The approximated edge strength is calculated as follows:

$$E_{KH} \approx \left|I(1,1)+I(1,3)-I(3,1)-I(3,3)\right|+\left|I(2,2)+I(2,4)-I(4,2)-I(4,4)\right| \quad (3.9)$$

$$E_{KV} \approx \left|I(2,1)+I(4,1)-I(2,3)-I(4,3)\right|+\left|I(1,2)+I(3,2)-I(1,4)-I(3,4)\right| \quad (3.10)$$

$$E_K \approx E_{KH} + E_{KV} \quad (3.11)$$

In such a way the overhead computation of edge strength is reduced to 19 operations for each sub-block. From some experiments, it is found that these approximations do not significantly affect the average partial SAD computations. After calculation of all E_k , the partial SAD is calculated in the order of block with greatest E_k to smallest E_k. This enables ME more quickly in the SAD calculation of a candidate position. The P'th accumulated partial distortion is defined as:

$$D_p = \sum_{k=1}^{p} SAD_k \ , \quad (3.12)$$

where SAD_k is the sum of the absolute differences of k-th 4x4 sub-block. Firstly, E_k of all 16 sub-block is calculated by (3.11) and calculation order is determined based on the edge strength. During SAD computation if partial SAD (D_p) is greater than the SAD_{min}, it terminates the remaining partial SAD computation and jumps to the next search position. EPDS require a sorting phase in which a vector of 16 elements must be ordered according to edge strength E_K. The small numbers of data (only 16) in the array allows the use of fast sorting technique based on insertion sorting [91]. Every iterations of insertion sort removes an element from input data, inserting it into the correct position in the already-sorted list, until no input elements remain.

Fig. 3.3 reports the total number of operations required for three different ME methods for *Foreman* video sequence. In this test, total 20 frames are encoded and similar results were found for other sequences. In our implementation, counting sort [92] was used as the sorting approach for FFSSG. In terms of total number of required operations, we have found that the proposed EPDS is about 4% faster than the FFSSG without loss of PSNR performances. Although computational performance of EPDS is better than FFSSG, still it is not comparable to search point reduction scheme. In order to further reduce computations, an enhanced EPDS approach is introduced in the next section.

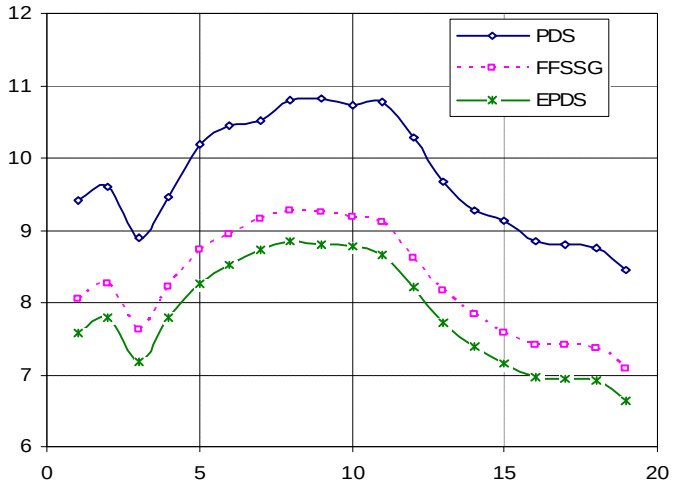


Fig. 3.3 Number of total operation of each frame of the tested algorithms of *Foreman* (X-axis: Frame number; Y-axis: No. of operation in 10^7)

3.4.2 Enhanced EPDS

Fig. 3.4 shows the flow diagram of the proposed enhanced EPDS algorithm. The enhanced algorithm adopts the similar approach of EPDS. However, during comparison we use a threshold value SAD_{th} instead of minimum SAD.

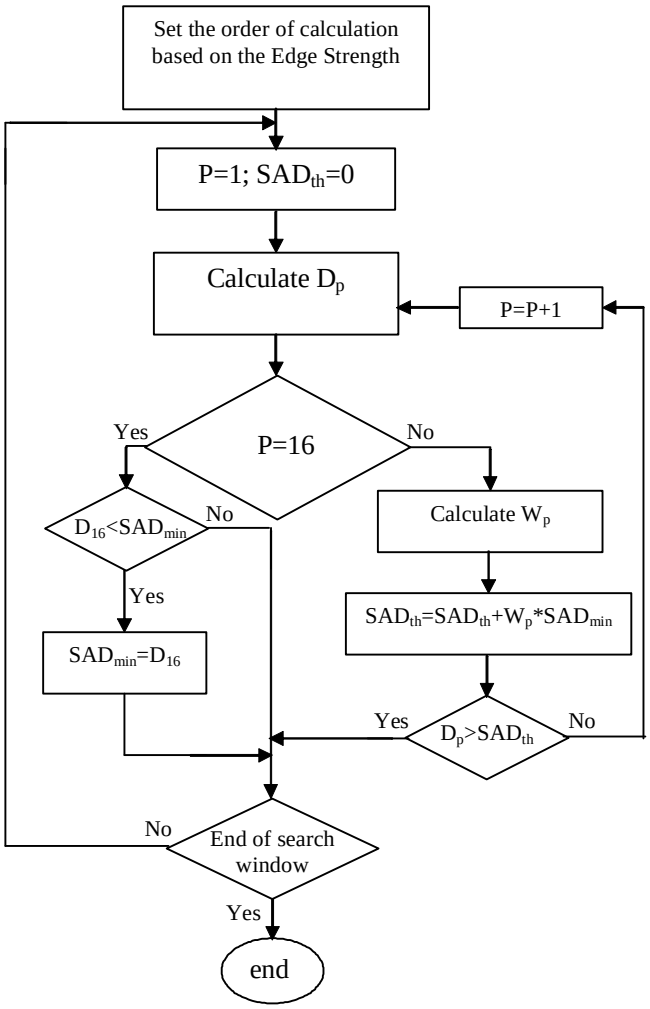


Fig. 3.4 Flow diagram of enhanced EPDS algorithm

The comparison starts from $p=1$ and proceeds towards $p=16$, and comparison is stopped if partial distortion is greater than the threshold $SAD_{th.}$ At the end of comparison ($p=16$), if D_{16} is smaller than the SAD_{min}, then this candidate motion vector becomes a new current minimum point. Comparison with SAD_{th} reduces computational complexity through high rejection of impossible candidates at an early stage. The adaptive threshold for *p*-th partial distortion is obtained using the following identity:

$$SAD_{th(p)} = SAD_{th(p-1)} + W_p * SAD_{\min} \quad \text{for p=1,2,3…16} \qquad (3.13)$$

and $SAD_{th(0)} = 0$

Here SAD_{min} is the current minimum distortion and W_p is the weighting factor for p-th partial distortion. W_p represents the contribution of the p-th sub-block to SAD calculation. It is reasonable to say that if the edge strength of a block is higher, then the contribution of this block in SAD computation is also higher. Therefore, we can conclude that edge strength of the k-th sub-block E_k, is proportional to distortion. From this observation, W_p is defined as

$$W_p = E_p / ES \tag{3.14}$$

Where, ES is the edge strength of the entire MB. ES can be calculated by adding all E_K as follows:

$$ES = \sum_{P=1}^{16} E_P \tag{3.15}$$

3.5 Two-step Edge based Partial Distortion Search (TS-EPDS)

In order to further reduce computations we propose a two-step edge based PDS (TS-EPDS) algorithm. At first we predict the best motion vector by searching some selected search points in the search window. It is understandable that SADs of the neighbouring search points are highly similar due to overlapping area between neighbouring search points. As an example consider that the current search point is (*i,j*) and for the search point (*i-1, j*), all pixel errors except the left most column are exactly same as that of (*i,j*) i.e. all columns except the left are overlapped. Fig. 3.5 shows the value of SAD over the search area of a macroblock of high motion *Stefan* video sequence. In this MB, the best motion vector is at (5, 0) position. It is clearly shown that the SAD differences between

neighbouring search points are very low. A low value implies that SAD of neighbouring search points of best motion vector is close to the minimum SAD and lower than the SAD of farther search points. This observation confirms that the best point of some selected points (as shown in grey points in Fig. 3.6) should be close to the best point of the full search method.

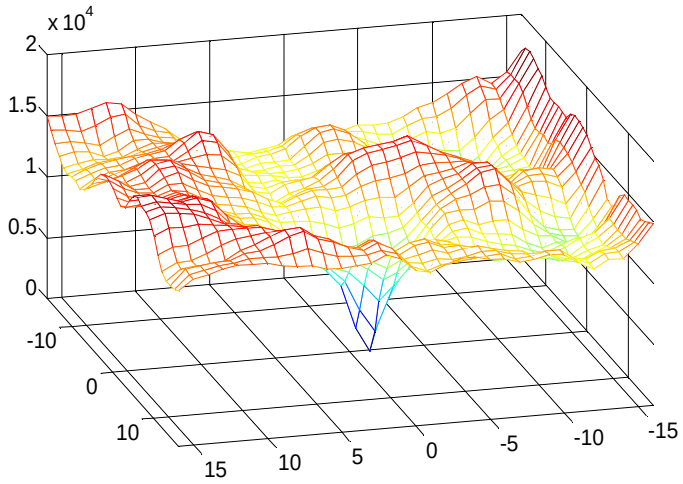


Fig. 3.5 A plot of SAD values over the search area

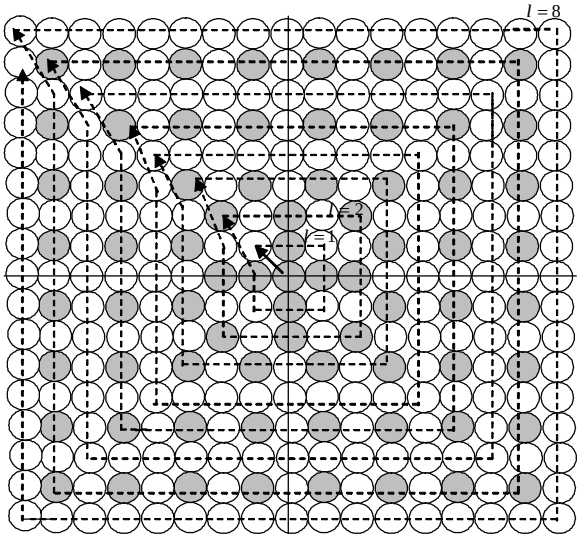


Fig. 3.6 Selected search points in the proposed method.

The spiral search method used in conventional algorithms starts searching from the center of the search region and moves the candidate position to estimate motion on a pixel by pixel basis. The spiral search algorithm performs block matching from the innermost spiral level to the outmost spiral level l. In the proposed algorithm, we have avoided all search points which belong to the even spiral levels except $l = 0,2$. Since in a natural video sequences most of the motion vectors are center biased, avoiding $l = 0,2$ is not an efficient way. Therefore, all even values of l from l = 4 to l = search range (in fig. 3.6 search range is 8) are skipped. In this chapter, these even spiral levels are called rejected spiral levels and the rest of the levels are defined as accepted spiral levels. In order to further reduce the search point, we have only searched some selected points in the accepted spiral level. The candidate MVs of the first step of proposed method are highlighted with grey colour in Fig. 3.6. In such a way, we can drastically reduce search points.

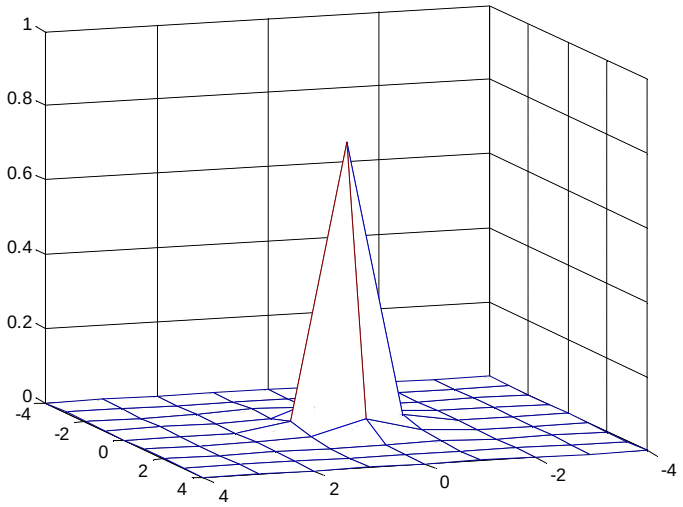


Fig. 3.7 Probability distribution of differential MVs

In order to illustrate the error introduced by searching only selected points , various kinds of standard video sequences are used to generate the true motion vectors found by FS and the motion vectors found by searching the grey points. Fig. 3.7 demonstrates the probability distribution plot of the differential motion vectors. Furthermore, from our observation, the probabilities of differential motion vectors located in $\pm D$ square window are as follows. 76.63 % for D=0, 95.28% for D=1, 96.90% for D=2 and so on. Such a higher center biased characteristic implies that the true motion vector is very near to the motion vectors found by searching selected grey points in Fig. 3.6.

From our extensive experiments, we have also observed that some of the true motion vectors are extremely close to the second best motion vector found by searching grey points in Fig 3.6. In order to improve the video quality of the proposed method, we have also considered this MV as a candidate in future steps. The proposed algorithm can be summarized as follows:

Step: 1

Search the grey points of Fig. 3.6 using the enhanced EPDS algorithm. Find two points that have minimum SAD and second minimum SAD.

Step: 2

Select the value of D.

(a) Select the point with minimum SAD as the search center. Search the points (which are not searched in step 1) within $\pm D$ search area by using enhanced EPDS.

(b) Select the point with second minimum SAD as the search center. Search the points (which are not searched in step 1 and step 2(a)) within $\pm D$ search area by using enhanced EPDS.

It is clear that a smaller value of D produces better complexity reduction. However, more blocks are incorrectly selected at the same time, which results in a significant loss in image quality. Therefore, there is a tradeoff between performance and complexity. From simulations, it is found that D=2 is a better selection in terms of both of performance and complexity.

In order to further reduce the search points, we have used a stopping condition at the first step of TS-EPDS. The objective of this stopping condition is to decide whether a search point has met the SAD criterion so that the best search point for the step 1 can be terminated early without performing an exhaustive search. In EPDS, the search begins from the matching candidate at the origin and then moves outward spirally. For the origin, we always compute the full distortion D_{16} and for other search points, if the accumulated distortion is larger than the current minimum, these search points are rejected without searching the rest of the pixels. In this dissertation, the search points for which full distortion D_{16} is smaller than the current SAD_{min}, are defined as accepted search points. After searching each accepted search point of step 1, D_{16} is compared against the threshold value and the search of the step1 is terminated if this D_{16} is lower than the threshold value. If D_{16} of an accepted search point satisfies (3.16), the search point is taken as the best search point of step1 and the remaining searches of step1 are skipped.

$$D_{16} < Th \tag{3.16}$$

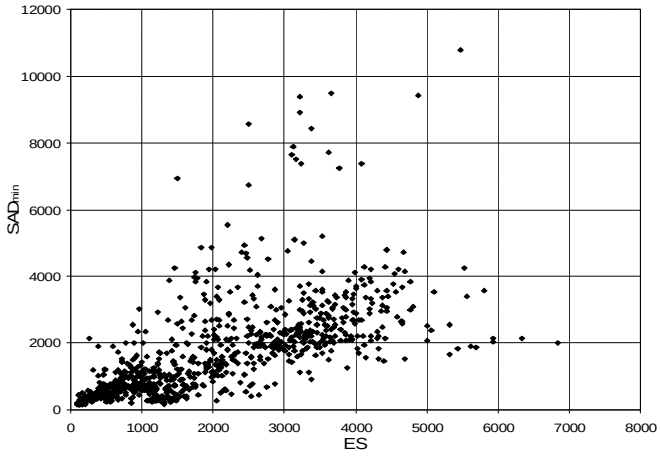


Fig. 3.8 Plot of SAD_{min} vs ES

Fixed threshold is not efficient for all types of video sequences and different parameter settings. It is already proved that an MB with high detail produces large value of minimum SAD. Therefore, threshold can be defined as a function of edge strength of an entire MB (ES) as defined in (3.15). Fig 3.8 shows the plot of SAD_{min} with ES, which demonstrates an increase in threshold value (*Th*) with ES. In this figure, data is collected by applying FS algorithm at different types of video sequences with different resolutions. The threshold value (Th) can be defined as a linear function of ES

$$Th = a + b \times ES \tag{3.17}$$

Where, b (=0.63) is the slope of the best fitted straight-line of data samples in Fig. 3.8 and a (=264) is the constant. In addition to the above mentioned stopping criteria, we have also used another stopping criteria outlined in step1 to detect the zero MV early. After searching half spiral levels, if we find that the current minimum is still at (0, 0), this MB is more likely to be associated with zero motion vector. If the condition is satisfied we skip the search points of the rest of the spiral levels. The overall algorithm of proposed TS-EPDS is summarized in Fig. 3.9.

For each Macroblock
% Initialization:
ES=0; SAD_{min}=124578965478; No_accepted_point=0; $a = 264$; $b = 0.63$; D=2;
For each partial distortion K (=1 to 16)
Calculate E_K by (3.11).
$ES=ES+E_K$;
endfor
Find the adaptive index set by insertion sorting of 16 E_k
For each partial distortion K (=1 to 16)
Calculate $W_K=E_K/ES$;
endfor
$Th = a + bES$
% Step 1:
For each search point SP (grey points of Fig. 3.6)
$SAD_{th(0)} = 0;$
D_P=0;
For each partial distortion P (= 1 to 16)
Calculate SAD_P;
$D_p=D_P+SAD_P$;
if ($P \neq 16$)
$SAD_{th(p)} = SAD_{th(p-1)} + W_p * SAD_{\min}$
if(D_P>$SAD_{th(p)}$) break the for loop and go to next search point;
endif
elseif (D_P<=SAD_{min})
$SAD_{min}=D_P$
No_accepted_point++;
Accepted_point(No_accepted_point)=SP;
% First stopping criteria
if(D_p<Th) go to step 2
end if
endif
endfor
% another stopping criteria.
if (the SP is within the spiral level which is greater than half of the search range & No_accepted_point =1), go to step 2
end if
endfor
% Step 2:
% Here center (1) is the best search point in step 1 and center (2) is the second best search point of step 2. Second best search points exist when
%No_accepted_point is greater than one. When only one accepted point was found, only search around the best search point.
if (No_accepted_point>1) high =2;
center (1)= Accepted_point(No_accepted_point);
center (2)= Accepted_point(No_accepted_point-1);
else high=1;
center (1)= Accepted_point(No_accepted_point);
endif

For i = 1 to high,
Choose center (i) as a center of the search area.
For each Search point SP (which are not searched in step 1) within the $\pm D$ search area.
$SAD_{th(0)} = 0;$
D_P=0;
For each partial distortion P (= 1 to 16)
Calculate SAD_P;
$D_p=D_P+SAD_P$;
if ($P \neq 16$)
$SAD_{th(p)} = SAD_{th(p-1)} + W_p * SAD_{\min}$;
if(D_P>$SAD_{th(p)}$) go to next search point;
endif
elseif (D_P<=SAD_{min})
$SAD_{min}=D_P$;
Best_SP=SP;
endif
endfor
endfor
endfor
return Best_SP;
endfor

Fig. 3.9 Overall algorithm of the proposed TS-EPDS

Table 3.1 Experimental results (FS shows full PSNR in dB)

Method		**FS**	**3SS [47]**	**4SS [68]**	**PDS [80]**	**FFSSG [85]**	**NPDS [81]**	**DHS-NPDS [93]**	**TS-EPDS**
Foreman (CIF)	PSNR	33.41	-0.89	-0.68	0	0	-0.20	-0.26	-0.02
	Speed-up	1	12.71	11.82	3.71	4.62	10.91	36.52	55.85
Stefan (CIF)	PSNR	25.75	-1.59	-1.99	0	0	-0.15	-0.99	-0.03
	Speed-up	1	12.13	12.86	3.40	3.89	10.80	32.53	39.51
Akiyo (CIF)	PSNR	42.80	-0.16	-0.07	0	0	-0.12	-0.03	-0.01
	Speed-up	1	13.11	12.36	8.74	10.05	11.46	105.15	247.91
Mobile (CIF)	PSNR	23.92	-0.28	-0.21	0	0	-0.18	-0.12	-0.02
	Speed-up	1	12.03	12.97	3.47	4.83	11.23	43.28	66.15
News (CIF)	PSNR	37.36	-0.25	-0.2	0	0	-0.20	-0.15	-0.02
	Speed-up	1	12.18	12.24	6.75	8.95	11.39	70.48	161.61
Hall (CIF)	PSNR	34.35	-0.13	-0.12	0	0	-0.19	-0.13	-0.01
	Speed-up	1	12.71	12.03	3.22	3.72	11.12	39.76	49.84
Claire (QCIF)	PSNR	42.94	-0.01	-0.01	0	0	-0.01	-0.01	0
	Speed-up	1	11.98	11.47	5.71	6.91	11.43	75.49	196.11
Foreman (QCIF)	PSNR	32.83	-0.46	-0.3	0	0	-0.17	-0.08	-0.01
	Speed-up	1	12.63	11.13	5.55	6.75	11.21	42.11	109.50
Carphone (QCIF)	PSNR	34.01	-0.22	-0.2	0	0	-0.14	-0.08	-0.03
	Speed-up	1	12.88	12.79	5.17	6.33	11.14	42.10	110.01
Average	PSNR	34.15	-0.44	-0.42	0	0	-0.15	-0.20	-0.01
	Speed-up	1	12.48	12.18	5.08	6.22	11.18	54.15	115.16

3.6 Simulation Results

In order to verify the effectiveness of our proposed algorithm, a large variety of video sequences are used for evaluation. Each sequence has 100 frames, hence, they represent a wide range of motion contents and video formats. All simulations are conducted under Windows Vista operating system, with Pentium 4 2.2 G CPU and 1 G RAM.

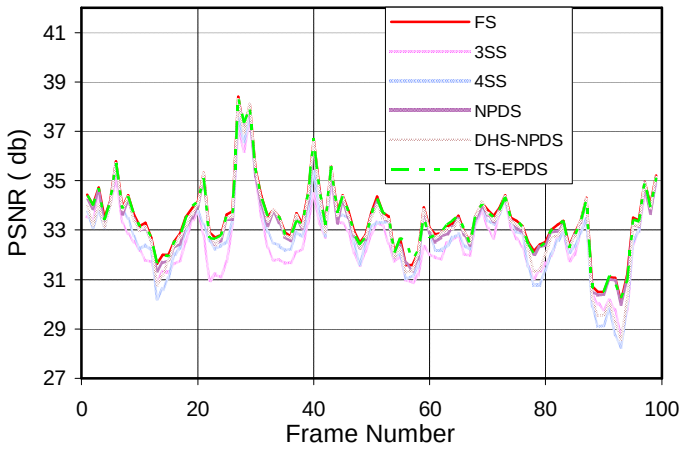
3.6.1 Experiments in Motion Estimation Package

In the experiment, the proposed method is integrated in a ME developing package in MATLAB environment. The block size is selected as 16x16, and the size of the search window is ± 16. We have compared our proposed method against FS, 3SS, 4SS, PDS,

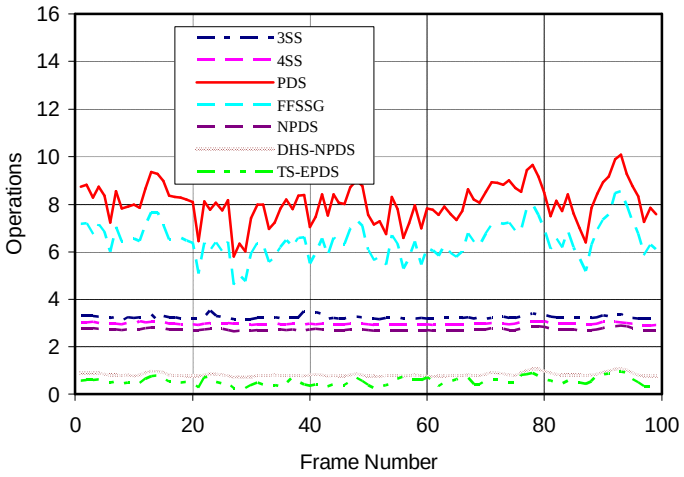
FFSSG, NPDS and DHS-NPDS in terms of PSNR and number of operations. Reduction in computational complexity is based on total operations involved in each method.

Table 3.1 demonstrates the PSNR and speed up performance of eight different algorithms. The PSNR performance of PDS and FFSSG are exactly same to FS. We have seen that the proposed TS-EPDS has the highest average PSNR amongst other methods. The average PSNR degradation is about 0.01 dB as compared to FS. Table 3.1 also shows the speed performance of different ME algorithms and the speed up is computed as $operations\_of\_FS / operations\_of\_BMA$. Note that multiplication and division operations are multiplied by weight of 8 since each multiplication/division operations requires more computations than addition/subtraction. The speed up of the proposed TS-EPDS is quite significant for various motion videos and on average the proposed method is 115 times faster than FS, about 10 times faster than NPDS and 2 times faster than the DHS-NPDS. The speed improvement is more pronounced and utmost 247 times of FS for low motion sequences such as Akiyo and Claire. Basically, these large computation reductions are direct consequence of early termination method. Most of the motion vectors of these sequences are around the search center so the best search points are found after few search operations. The speed up factor for high motion sequences such as Foreman and Stefan is lower than that of low motion sequences. The PSNR degradation is very low (about 0.02 dB) whereas many fast BMAs such as 3SS, 4SS, DHS-NPDS result in larger PSNR degradations of up to 1.99 dB. Frame by frame comparison of Foreman sequence for both PSNR and number of operations is presented in Fig. 3.10. As evident from this figure the proposed method offers significant savings in computational cost and obtains PSNR performance that closely matches the FS

method. The subjective video quality of the sequence Stefan is reflected in Fig. 3.11. In this example, the white line of the down-right corner of the image can only be detected by FS, NPDS and TS-EPDS algorithm. It is also shown that the video quality of the proposed algorithm is to a great extent similar to FS as compared to other algorithms.



(a)



(b)

Fig. 3.10 Frame by frame comparison for Foreman CIF sequences (a) Average PSNR in dB (b) average total operation per frame* 10^{-7}

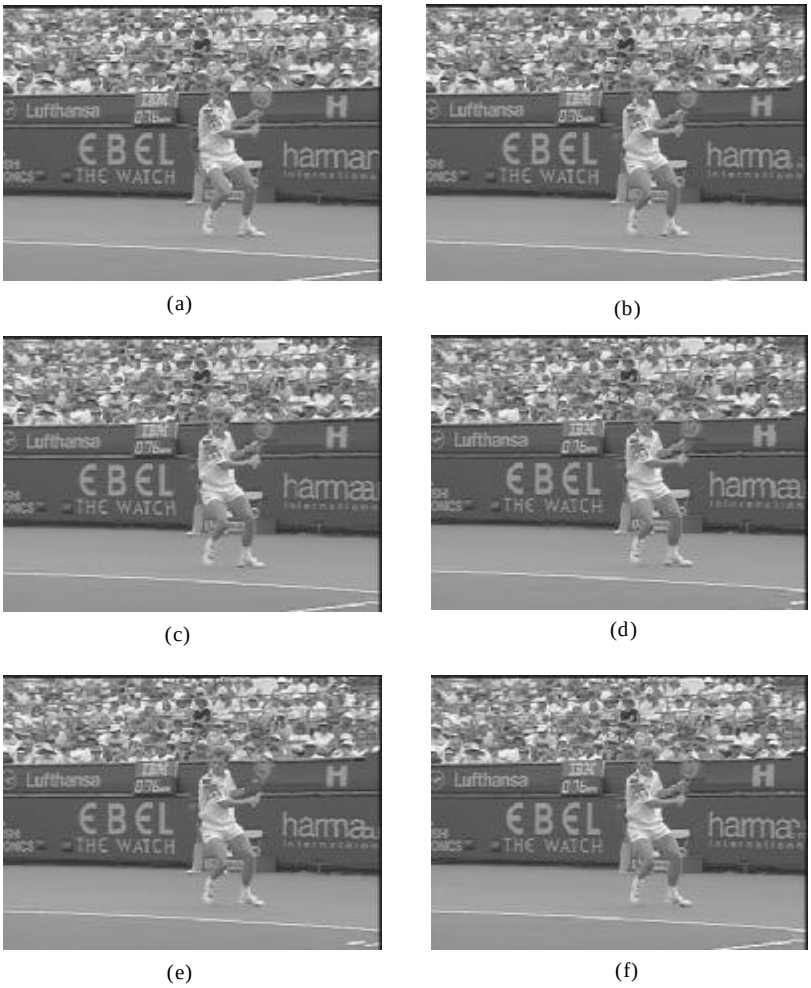


Fig. 3.11 The 7th motion compensated predicted frame for the “Stefan” CIF video using different BMAs: (a) original frame; (b) FS, PSNR = 25.32 dB; (c) NPDS, PSNR = 25.05 dB; (d) DHS-NPDS, PSNR = 24.28 dB; (e) 3SS, PSNR = 24.96 dB; and (f) TS-EPDS, PSNR = 25.29 dB.

3.6.2 Experiments in H.264/AVC

Although we used SAD as the cost of this paper, our method is also applicable using Lagrangian cost scheme. In this experiment, the proposed method is integrated with H.264/AVC reference software version 9.6 [66]. The simulation conditions are: number

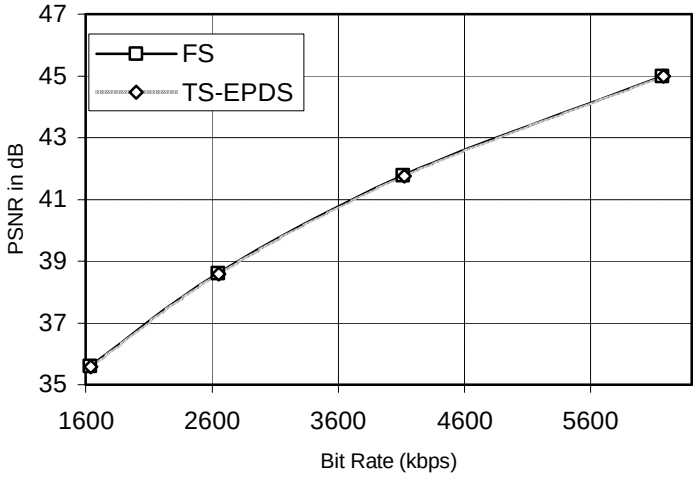
of reference frames is one; Hadamard transform is OFF, rate-distortion (RD) optimization is ON, search range is 16; only 16×16 mode is used; frame rate is 30 fps; and quantization parameters are 16, 20, 24 and 28. The average PSNR differences ($\Delta PSNR$), and the average bit rate differences ($\Delta Rate$%) are calculated according to the numerical averages between RD curves derived from the FS and other algorithms, respectively [67]. The speed up is computed as the ratio of the total operations of integer pixel ME of FS to that of BMAs.

Table 3.2 Experimental results in H.264/AVC

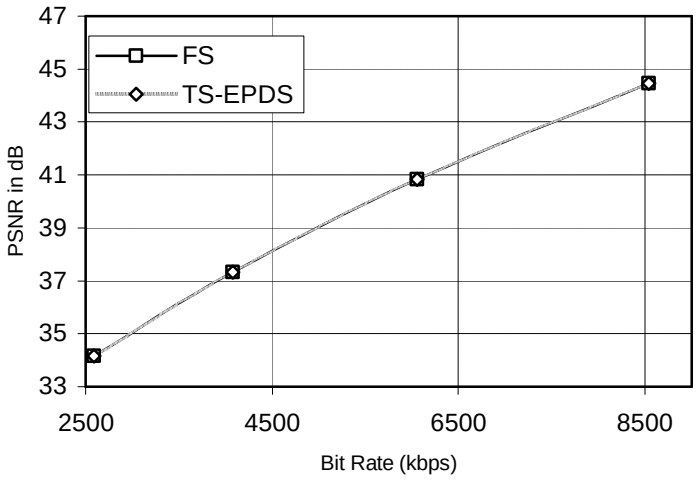
Method		PDS	NPDS	TS-EPDS
Foreman (CIF)	ΔPSNR	0	-0.03	-0.001
	ΔRate%	0	1.95	0.037 %
	Speed-up	3.69	10.14	77.18
Stefan (CIF)	ΔPSNR	0	-0.13	-0.027
	ΔRate%	0	3.81	0.789 %
	Speed-up	3.59	10.46	65.94
Akiyo (CIF)	ΔPSNR	0	-0.03	-0.001
	ΔRate%	0	1.58	0.007 %
	Speed-up	7.54	10.34	257.7
Mobile (CIF)	ΔPSNR	0	-0.05	-0.008
	ΔRate%	0	1.16	0.019 %
	Speed-up	3.62	11.20	102.29
News (CIF)	ΔPSNR	0	-0.04	-0.005
	ΔRate%	0	1.43	0.018 %
	Speed-up	5.65	10.21	155.24
Hall (CIF)	ΔPSNR	0	-0.04	-0.002
	ΔRate%	0	2.91	0.090 %
	Speed-up	4.55	10.60	76.56
Average	ΔPSNR	0	0.053	0.007
	ΔRate%	0	2.14	0.160 %
	Speed-up	4.77	10.49	122.48

The comparison of PDS, NPDS and TS-EPDS methods with FS in terms of PSNR, bit-rate, and speed-up are shown in Tables 3.2. We have seen that average PSNR reduction and bit rate increment of TS-EPDS is very negligible which are 0.007 dB and 0.16%,

respectively. The TS-EPDS is about 122 times faster than FS and can achieve higher PSNR and lower bit rate as compared to NPDS. Fig. 3.12 shows the comparison of RD curves of *Stefan* and *Mobile* video sequence. The RD curve of TS-EPDS method is closely matched with FS RD curve.



(a) Stefan



(b) Mobile

Fig. 3.12. RD curves of FS and proposed method

3.7 Summary

In this chapter, we proposed a new block matching algorithm by sorting the edge strength of square sub-blocks. By using a calculation order according to block complexity, we can obtain faster elimination of impossible candidate vectors than PDS algorithms. The enhanced EPDS algorithm adaptively changes the weighting factors of early termination threshold. In order to further reduce the complexity, a TS-EPDS algorithm is successfully developed. Early search termination can effectively obtain better motion vectors for different types of blocks. Experimental results confirmed that the proposed algorithm can achieve both higher PSNR and lower computational complexity than many motion estimation algorithms, such as 3SS, 4SS, NPDS and DHS-NPDS.

Chapter 4

Early Terminated Motion Estimation

The variable block-size motion estimation process is the H.264/AVC encoder's most time-consuming function. This chapter proposes reducing the complexity of the motion estimation process with an early termination algorithm that features an adaptive threshold based on the statistical characteristics of rate-distortion (RD) cost regarding current block and previously processed blocks and modes. The motion estimation process is terminated when the computed RD cost dips below an adaptive threshold that depends on the RD cost of previously calculated macroblocks (MBs) and modes. This means that most motion searches can be stopped early, with a large number of search points saved. A region-based search is also suggested to further reduce the computation required for full search motion estimation. The proposed early termination method will work with any fast, variable block-size motion estimation algorithm, and experiments were also carried out with the fast-motion estimation algorithm adopted by H.264/AVC. A search point reduction scheme for the fast-motion estimation of H.264/AVC is also introduced, and the experimental results are illustrated.

4.1 Challenges and Literature Survey

Motion estimation for each 16x16 MB can be performed using a number of different block sizes and shapes. In H.264/AVC, inter frame motion estimation is performed for different sizes such as 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4. These were illustrated in Fig. 2.7. Using seven different block sizes and shapes can translate into bit rate savings of more than 15% as compared to using only a 16x16 block size [21]. However, the motion estimation process is extremely complex. For each search point, H.264/AVC computes RD cost function J as the measure of prediction error. The RD cost function J is defined as

$$J(mv, l) = SAD(s, c(mv)) + l\,R(mv - pmv) \tag{4.1}$$

where mv, pmv, $SAD(s, c(mv))$, l and $R(mv - pmv)$ are defined in (2.1).

As shown in Fig. 2.7, the total number of blocks for all modes is 41. Each block can have its individual motion vector. If the encoder selects one reference frame, to encode a MB, there are total 41 MVs should be computed. If full search method is utilized, the number of search points for a macroblock is $41(2W+1)^2$, where, W is the search range. Therefore, the complexity and computation load of H.264/AVC increase drastically compared to any previous standards.

Various algorithms, such as three step search [47], diamond search [48] and 2-d logarithm search [37], have been proposed to reduce the computational complexity of the ME module, as well as a number of research explorations studying fast algorithms and ME mode decision [49-52] in an attempt to accelerate H.264/AVC video coding.

In order to reduce the computation, the Hybrid Unsymmetrical-cross Multi-hexagon Grid Search (UMHS) method [50, 64] was adopted by the H.264/AVC reference software as a fast integer-pel motion estimation algorithm for its good RD performance. This method includes four steps with different kinds of search pattern.

One way to reduce the complexity of the ME process is to terminate the ME calculation early in the process. At present, many efforts have been made to comprehensively explore the use of early termination algorithms in ME and mode decision for H.264/AVC video coding [53-63]. The well-known zero-motion detection algorithm [53] compares sum of absolute difference (SAD) of two blocks with a predefined threshold to determine the stationary block and then skips the remaining search points. A fast motion estimation algorithm by adaptively changing the early termination threshold for the current accumulated partial SAD is proposed in [59]. An improvement on fast motion estimation method adopted by H.264/AVC reference software is presented in [58]. The variable block-size zero motion detection and variable block-size best motion detection algorithms compare the rate-distortion cost function of the two blocks instead of the SAD to detect the zero motion blocks or best motion vector for the variable block-size H.264/AVC video coding is proposed in [54]. However, this algorithm is using fixed threshold values that are difficult to determine good values for remaining the high accuracy with different sequences and parameters setting. This method searches either only one search point or all of the search points in the search range. Therefore, computation reduction is not so high.

4.2 Algorithm of Proposed Early Termination

The objective of early termination is to decide whether a search point has met the RD cost criterion, so that the best search point for the current block can be terminated early without trying all of the search points. After searching each search point, the current RD cost J is compared against the threshold value of the specific block type, and the search is terminated if this RD cost J is lower than the threshold value. We define thresholds T_m $(m=1, 2…7)$ for seven block sizes. If the cost of a search point satisfies (4.2), this search point is taken as the best one and the remaining searches can be skipped.

$$J < T_m \text{, for } m=1, 2,\ldots..7 \tag{4.2}$$

Thus, most of the motion searches can be stopped early and then a large number of search points can be skipped. The procedure of the proposed fast motion estimation based on early termination algorithm is outline as follows.

```
Assume the current macroblock is MB(x,y)
For each mode m (m = 1, 2, …, 7) of the MB(x, y)
    Calculate the adaptive threshold, Tm
       For each block in the macroblock
                        For each search point
                                If J < Tm, this search point is best point and go to the next  block
                                end if
                        End for
       End for
End for
Set best mode to be the mode with minimum cost
```

The key issue becomes how to determine the thresholds. It is clear that the larger the thresholds are, the more search points can be skipped, but more blocks are often incorrectly selected at the same time, which results in significant image quality loss. A

tradeoff between performance and complexity must be made. In practice, preventing a loss in video quality is more important than a minor increase in complexity, so before the thresholds are selected, a statistical analysis of RD cost is presented in the next section.

4.3 Statistical Analysis of RD Cost function

While RD costs vary largely for different modes, the RD costs of neighboring MBs and their collocated MBs in previous frames are highly correlated. Our first series of experiments investigated whether or not previously calculated RD cost values could be used as a prediction metric for the early termination technique. Correlation coefficients were calculated between RD-cost values, using full search ME, and the correlation coefficient (r) of two random variables, X and Y, is defined by:

$$r(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} \tag{4.3}$$

where $Cov(X,Y)$ is the co-variance between X and Y and $Var(X)$ and $Var(Y)$ are the variances of X and Y, respectively.

The processing order for the different modes in the H.264/AVC ME process is 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. This allows us to estimate the RD costs of lower size blocks from higher size blocks with the same MB. In the case of the 16x16 mode,

information about MV and RD costs for other modes with the same MB are not available, but the RD costs of neighboring MBs and collocated MBs in previous frames had already been computed.

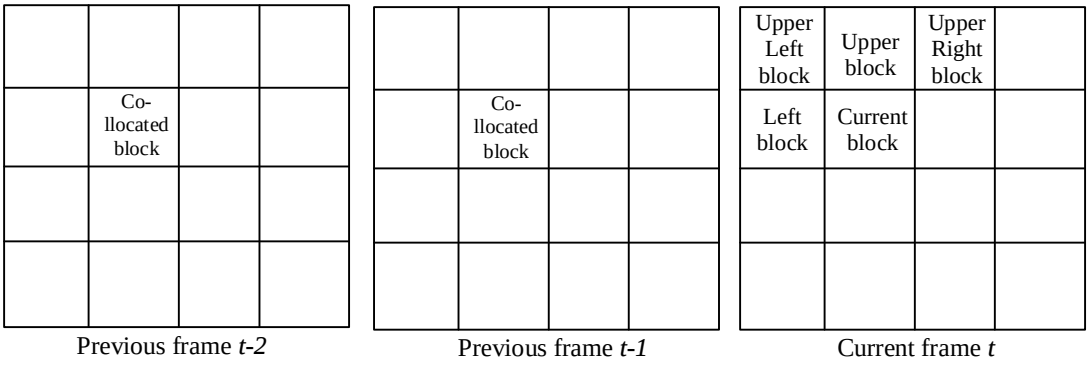


Fig. 4.1 Neighboring and collocated blocks

Consider the current frame number to be t. Fig. 4.1 shows the neighboring macroblocks of frame t and collocated macroblocks of previous frames. Table 4.1 (a) shows the correlation coefficients of RD costs between the 16x16 mode of the current MB and the 16x16 mode of the candidate MBs. The data is collected by encoding the first 100 frames of the mentioned video sequences with QP equal to 28, 36 and 40. The average correlation coefficients are presented, and RD costs are clearly highly correlated with the collocated block in the immediately previous frame. The second highest correlation is with the collocated MB in frame $t-2$, which implies that the RD cost of the best search point for the 16x16 mode remains similar between consecutive frames and provides a good basis for predicting the RD cost of the current frame MB, which can then be used for early termination.

Table 4.1 (a) RD cost correlation between 16x16 mode of current MB and 16x16 mode of the candidate MB

Sequence	Collocated block in frame $t-1$	Collocated block in frame $t-2$	Left	Upper	Upper right	Upper left
Stefan	0.95	0.90	0.74	0.75	0.58	0.61
Akiyo	0.98	0.97	0.49	0.71	0.44	0.38
Mobile	0.93	0.92	0.39	0.22	0.18	0.09
Foreman	0.90	0.83	0.51	0.36	0.21	0.23
Carphone	0.92	0.89	0.50	0.52	0.22	0.34
Salesman	0.98	0.96	0.65	0.39	0.31	0.25
Claire	0.98	0.97	0.59	0.82	0.42	0.51

Table 4.1 (b) RD cost correlation of 16x8 block size motion estimation

Sequence	16x16 mode of Collocated block in frame $t-1$	16x8 mode of Collocated block in frame $t-2$	16x16 mode of current block at current frame t
Stefan	0.86	0.80	0.91
Akiyo	0.90	0.82	0.91
Mobile	0.74	0.65	0.83
Foreman	0.74	0.65	0.85
Carphone	0.80	0.71	0.88
Salesman	0.86	0.79	0.89
Claire	0.93	0.88	0.95

Table 4.1 (c) RD cost correlation of 8x16 block size motion estimation

Sequence	16x16 mode of Collocated block in frame $t-1$	8x16 mode of Collocated block in frame $t-1$	16x16 mode of current block at current frame t	16x8 mode of current block at current frame t
Stefan	0.84	0.83	0.90	0.81
Akiyo	0.86	0.75	0.87	0.80
Mobile	0.77	0.68	0.83	0.71
Foreman	0.79	0.71	0.86	0.73
Carphone	0.79	0.72	0.87	0.77
Salesman	0.88	0.82	0.91	0.83
Claire	0.89	0.82	0.91	0.87

The correlation of RD costs between other modes and previously computed modes are presented in Table 4.1 (b-g). Tables 4.1 (b-d) reveals that the RD cost of the 16x8, 8x16 and 8x8 modes are highly correlated with the 16x16 mode of the same MB. The value of correlation is about 0.9, which is understandable because the 16x16 mode is decomposed into two rectangular blocks (8x16 or 16x8) or four 8x8 square blocks.

Table 4.1(d) RD cost correlation of 8x8 block size motion estimation

Sequence	16x16 mode of Collocated block in frame $t-1$	8x8 mode of Collocated block in frame $t-1$	16x16 mode of current block at current frame t	16x8 mode of current block at current frame t	8x16 mode of Collocated block in frame t
Stefan	0.76	0.68	0.81	0.75	0.80
Akiyo	0.78	0.61	0.89	0.72	0.70
Mobile	0.62	0.49	0.87	0.57	0.59
Foreman	0.64	0.50	0.89	0.60	0.63
Carphone	0.70	0.56	0.87	0.69	0.70
Salesman	0.78	0.60	0.81	0.74	0.75
Claire	0.85	0.75	0.87	0.83	0.79

Table 4.1(e) RD cost correlation of 8x4 block size motion estimation

Sequence	16x16 mode of Collocated block in frame $t-1$	16x16 mode of current block at current frame t	16x8 mode of current block at current frame t	8x16 mode of Collocated block in frame t	8x8 mode of current block in current frame t
Stefan	0.71	0.76	0.74	0.75	0.90
Mobile	0.52	0.56	0.47	0.49	0.83
Foreman	0.57	0.61	0.53	0.55	0.86
Akiyo	0.72	0.73	0.66	0.64	0.91
Carphone	0.62	0.69	0.62	0.62	0.88
News	0.62	0.76	0.69	0.67	0.93
Salesman	0.71	0.73	0.67	0.68	0.88
Claire	0.81	0.83	0.80	0.77	0.95

Similarly, from Table 4.1(e-g), the RD costs of the 8x4, 4x8 and 4x4 modes are highly correlated with the 8x8 mode of the same MB, so the threshold value of lower size modes can be defined as a function of the best RD cost of either the 16x16, or the 8x8 block.

Table 4.1(f) RD cost correlation of 4x8 block size motion estimation

Sequence	16x16 mode of Collocated block in frame $t-1$	16x16 mode of current block at current frame t	16x8 mode of current block at current frame t	8x16 mode of Collocated block in frame t	8x8 mode of current block in current frame t	8x4 mode of current block in current frame t
Stefan	0.71	0.77	0.70	0.76	0.94	0.83
Mobile	0.52	0.56	0.47	0.49	0.93	0.69
Foreman	0.58	0.63	0.53	0.58	0.89	0.78
Akiyo	0.70	0.71	0.65	0.63	0.89	0.81
Carphone	0.63	0.70	0.61	0.63	0.88	0.79
News	0.67	0.74	0.66	0.66	0.91	0.84
Salesman	0.72	0.75	0.67	0.70	0.90	0.80
Claire	0.80	0.82	0.78	0.75	0.93	0.88

Table 4.1(g) RD cost correlation of 4x4 block size motion estimation

Sequence	16x16 mode of current block in current frame t	16x8 mode of current block at current frame t	8x16 mode of current block at current frame t	8x8 mode of Collocated block in frame t	8x4 mode of current block in current frame t	4x8 mode of current block in current frame t
Stefan	0.71	0.67	0.69	0.85	0.79	0.84
Mobile	0.67	0.49	0.51	0.82	0.78	0.79
Foreman	0.53	0.45	0.49	0.85	0.67	0.73
Akiyo	0.66	0.59	0.59	0.81	0.74	0.74
Carphone	0.63	0.55	0.57	0.88	0.71	0.74
News	0.70	0.62	0.62	0.85	0.79	0.78
Salesman	0.68	0.61	0.64	0.80	0.72	0.75
Claire	0.79	0.76	0.72	0.88	0.84	0.84

4.4 Threshold Selection

An analysis of the above section shows that the RD cost of the best MV from the 16x16 mode is highly correlated with that of the collocated MBs in the previous frames. The threshold for the 16x16 mode can be derived as a linear model of the RD costs of collocated MBs. Additionally, it is reasonable to say that a complex MB produces a large RD cost value compared to a simple MB. In the case of a complex MB, the difference in RD costs between the collocated MBs is also higher. Conversely, if the difference in RD costs between the collocated MBs in the previous frames is larger, the RD cost of the target MB should be larger. These observations prompt the threshold for the 16x16 mode motion estimation to be defined as follows:

$$T_1 = \frac{a\,J_{t-1} + b\,J_{t-2}}{a+b} + ed \tag{4.4}$$

where $d = \left| J_{t-1} - J_{t-2} \right|$ and a , b , e are the weighting factors. Here, J_{t-1} and J_{t-2} are the RD cost of the best MV for the 16x16 mode of collocated MB in frame *t-1* and *t-2*, respectively. d represents the absolute difference of RD costs between collocated MBs in the previous frames. In order to set the weighting factors a , b , e we have done several experiments for different video sequences (*Akiyo, Foreman, Stefan, Mobile, Carphone, Salesman*) with QCIF format at different QP values and have observed a rate-distortion performance in these video sequences at different weighting factor combinations. $a = 3, b = 1$ and $e = 1/2$ produced better rate-distortion performance and to avoid the division operation, the threshold value in (4.4) is implemented by shift right as follows:

$$T_1 = (3J_{t-1} + J_{t-2}) >> 2 + \left| J_{t-1} - J_{t-2} \right| >> 1 \tag{4.5}$$

It has already been proven that the RD cost of the best MV of the 16x8, 8x16 and 8x8 modes correlate highly with the 16x16 mode from the same MB. Similarly, the RD costs of the best MV from the 8x4, 4x8 and 4x4 modes correlate highly with the 8x8 mode from the same MB. Based on these facts, the thresholds for other modes are defined as:

$$T_m = C_m + g \quad \text{for } m = 2,3,4,5,6 \text{ and } 7 \tag{4.6}$$

$$\text{where, } C_m = \frac{J_{16x16}}{S_m} \text{ for } m = 2,3, \text{and } 4 \text{ (16x8,8x16 and 8x8 modes)}$$

$$\text{and } C_m = \frac{J_{8x8}}{S_m} \text{ for } m = 5,6, \text{and } 7 \text{ (8x4,4x8 and 4x4 modes)}$$

where the scale factor $S_m = 2$ for the 16x8,8x16,4x8 and 8x4 *(m=2,3,5,6)* modes and $S_m = 4$ for the 8x8 and 4x4 *(m=4,7)* modes. g is the penalty function due to the difference between the actual best RD cost of the corresponding mode and C_m. g is defined as:

$$g = \left| Actual \cos t - C_m \right| \tag{4.7}$$

Fig. 4.2 shows the relationship between C_m and g for two different types of video sequences, and similar results were found for other types of sequences. The figure shows that the penalty function g is very low, almost similar in region 1 ($C_m < z$). The value of g is increasing with C_m in region 2 ($C_m \geq z$), which suggests that g is approximated as follows:

$$g = A \quad \text{for } C_m < z \tag{4.8}$$

$$g = BC_m + D \quad \text{for } C_m \geq z$$

where B is the slope of the best fitted straight-line and D is the constant. A is the mean value of g within the range $C_m = 0$ to z. Simulations proved that the value of z increases from low motion sequences to high motion sequences, always within the range of 400 to 700. Low motion sequences (such as Akiyo) put z around 400, while high motion and complex sequences (such as Stefan) put z around 700. The Foreman sequence can represent scenes that possess relatively medium motion, which means that we can select $z = 500$ based on the Foreman and apply them to other sequences. Experimental results have confirmed that this assumption does not affect the encoding performances, because the variation of g around z is very low. Table 4.2 shows the value of A, B and D for different types of video sequences. Data is collected by encoding 100 frames of different video sequence types with different QP values. The values of A, B and D vary with the content of the video sequences. The value of A increases with the content of the motion of sequences. High motion and complex sequences have a higher A value compared to low motion and simple sequences. It is clear that the larger the A value, the more search points can be skipped, but more blocks are incorrectly selected as a result. We choose the three different combinations (minimum, maximum and mean) of A and observe the RD performances and computation savings of the different video sequences. We have also encoded some video sequences that do not appear in Table 4.2. The mean value of A produced better results in terms of both computation and quality, and we have applied a similar

technique to find the B and D values. Three different combinations (minimum, maximum and mean) were tested, and we found that both the RD performance and the computation were not greatly affected by the combinations, which is understandable because only a small percentage of sample values fall in region 2. Finally, we choose the mean value of A, B and D. In order to avoid the floating-point operation, we chose the nearest integer values of A and D. Additionally, the mean value of B (=0.13) is approximated as $1/8$, which can be implemented by shift right operator. So the value of g becomes:

$$g = 50 \quad \text{for } C_m < 500 \tag{4.9}$$

$$g = (C_m >> 3) + 45 \quad \text{for } C_m \geq 500$$

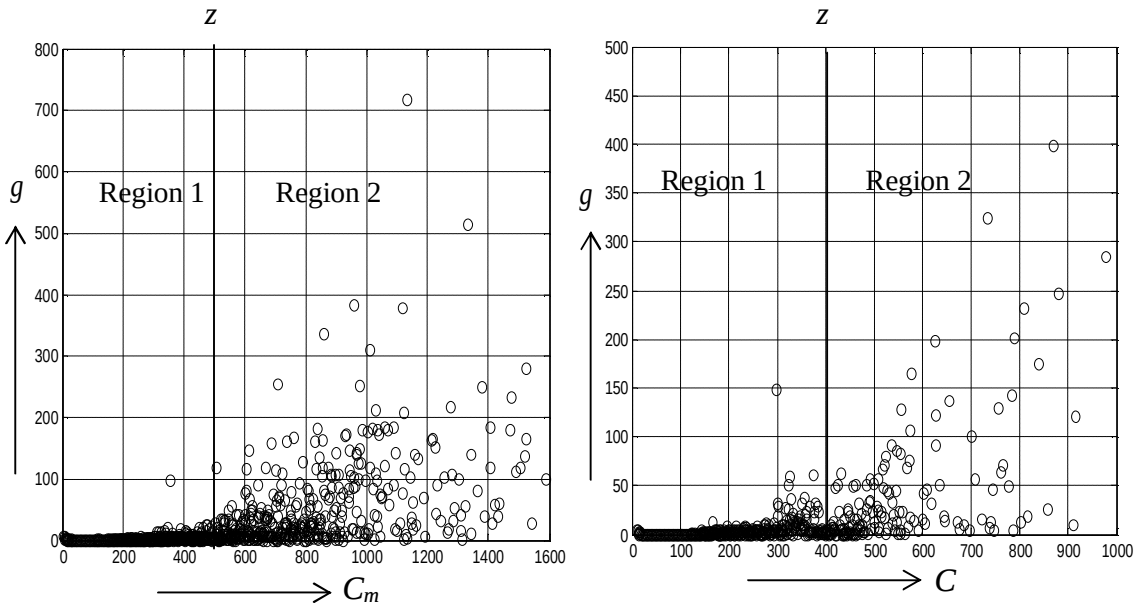


Fig 4.2 Variation of g with C_m at QP=28 (a) Foreman (b) Akiyo

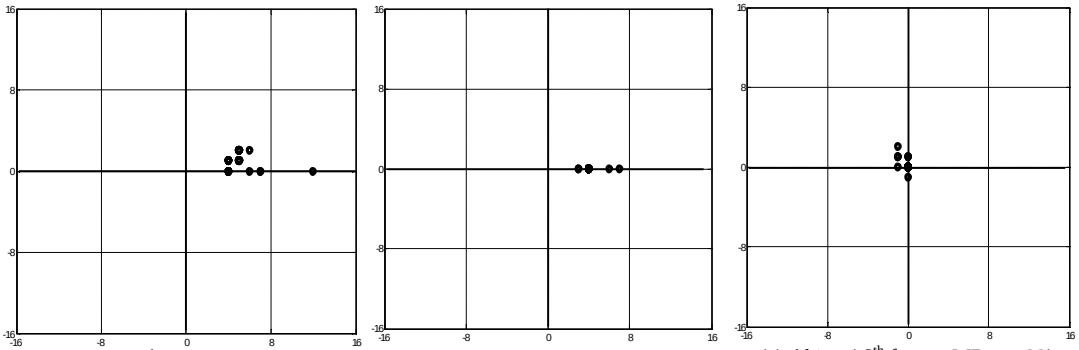
Table 4.2 value of A, B and D of (4.8) for different video sequences

Sequence	QP	A	B	D
Stefan	28	43.12	0.19	20.88
	36	63.00	0.14	46.86
	40	90.03	0.12	87.63
Foreman	28	17.02	0.09	7.46
	36	59.05	0.07	58.75
	40	80.09	0.06	97.87
Akiyo	28	16.88	0.18	8.62
	36	37.93	0.15	59.96
	40	47.87	0.11	99.27
Mobile	28	57.00	0.21	49.13
	36	78.29	0.16	91.73
	40	94.39	0.14	137.29
Carphone	28	31.13	0.19	6.65
	36	53.38	0.16	37.35
	40	69.64	0.14	64.15
Salesman	28	21.59	0.16	10.36
	36	34.10	0.12	37.85
	40	40.27	0.09	64.62
Claire	28	20.78	0.18	14.24
	36	35.91	0.14	34.30
	40	49.01	0.11	67.46
Average		49.55	0.13	45.38

4.5 Region-based Search Order for Full Search ME

The full search algorithm utilized in H.264/AVC applies a pixel-based spiral search method. The spiral search algorithm compares a block of the current frame with blocks from the previous frame inside the search area. The search area is centered at the same coordinates as the previous frame block and the search process of the search space moves outward in a spiral. This method suits video sequences with a motion vector distribution that is close to the search center, but for high motion sequences, the spiral search method is not more efficient [65]. This problem is addressed by dividing the search window into different regions and representing each region with a square of NxN

size [65], with the size of the square related to the complexity reduction. Additionally, the early termination algorithm presented in the previous section is based on the inter-mode correlation, so in order to apply the region-based search, a modification is necessary. Fig. 4.3 displays the 41 motion vectors of a low motion MB (*Akiyo*) and a high motion sequence (*Stefan*). All the motion vectors are more directionally oriented for the high motion sequences, so that dividing the search window into a square pattern is not as suitable as it is for low motion sequences.



(a) Stefan (6th frame: MB no: 40) (b) Stefan (6th frame: MB no: 2) (c) Akiyo (6th frame: MB no: 22)

Fig. 4.3 41 motion vectors of a MB

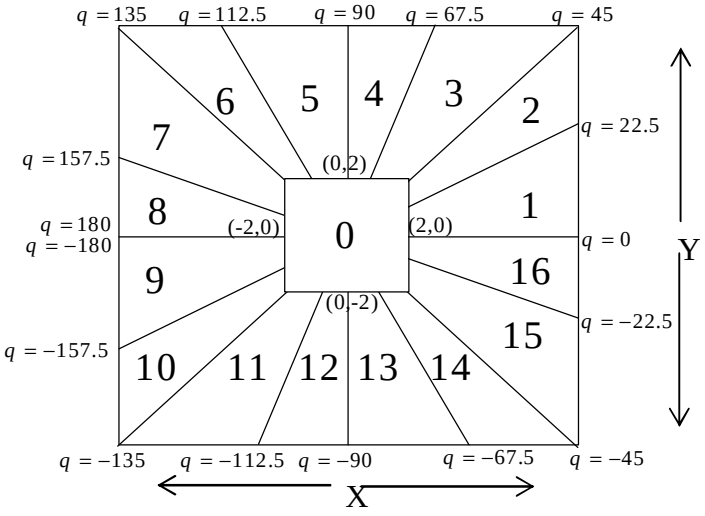


Fig. 4.4 Partitioned search range

Based on this observation, we have divided the search window into 17 different regions as shown in Fig. 4.4. Region 0 represents the low motion vectors. In this region, $|X| <= 2$ and $|Y| <= 2$. Regions 1-16 are based on the angle q, which is defined as follows:

$$q = \tan^{-1}\frac{Y}{X}, \text{ where } |X| > 2 \text{ and } |Y| > 2 \tag{4.10}$$

Table 4.3 % of the event "*MV at target mode=MV with previously calculated mode*"

Target mode	Previously computed mode						
	Collocated MB in previous frame	Current MB in current frame					
	16x16	16x16	16x8	8x16	8x8	8x4	4x8
16x16	77.5	-	-	-	-	-	-
16x8	72.6	82.3	-	-	-	-	-
8x16	67.2	82.0	76.2	-	-	-	-
8x8	61.7	76.4	73.2	74.1	-	-	-
8x4	59.8	73.6	71.2	69.8	82.4	-	-
4x8	58.2	73.4	70.4	69.4	82.6	75.4	-
4x4	59.1	71.4	68.1	68.4	79.2	73.8	77.2

The goal of this algorithm is to reduce the number of search points in the early termination method, so the starting search region is a key factor of the resulting computation reduction. After the search range is partitioned, an adaptive search starts in the most probable region. All the successive video frames are correlated in a temporal direction, so we can expect most of the picture movements to follow in the same direction. Table 4.3 gives the MV percentage for the previously calculated modes. This analysis collected data from five different types of video sequences (Foreman, Akiyo,

Stefan, Mobile and Carphone) to produce the average results presented in Table 4.3. We have seen that the MV of the 16x8, 8x16 and 8x8 modes correlate highly with the 16x16 mode, while the MV of the 4x8, 8x4 and 4x4 modes correlate more with the 8x8 mode of the current MB. Based on this analysis, Table 4.4 presents the most probable region and MV correspondence with different modes.

Table 4.4 Selection of most probable region

Mode	**Most Probable Motion vector**	**Most Probable region**
16x16 mode	Motion vector of the collocated macroblock in the previous frame	The region at which the MV of the collocated macroblock in the previous frame
16x8, 8x16 and 8x8 mode	Motion vector of 16x16 mode	The region at which the MV of the 16x16 mode of the current macroblock
4x8, 8x4 and 4x4 mode	Motion vector of 8x8 mode	The region at which the MV of the 8x8 mode of the current macroblock

Table 4.5 Percentage of most probable region

Sequence	**Percentage**
Foreman	90.69 %
Akiyo	99.34 %
Mobile	98.30 %
Carphone	95.02 %
Claire	97.95 %
Silent	96.17 %
News	98.42 %
Container	99.38 %
Stefan	87.33 %
Average	95.84%

In order to justify this method, we have done some experiments with different video sequences at different QP factors. We have calculated the percentage of the event "*most*

probable region is the best region" and the results are presented in Table 4.5. Approximately 95% of MVs fall within the most probable regions.

For high motion MB, the most probable region should be between regions 1-16, as shown in Fig. 4.4. The encoder starts to search from the most probable region and then moves to search region 0. The order of the remaining search regions is based on the angle between the most probable region and the other regions, which must be calculated by (4.11) beforehand:

$$\Delta q = |q_m - q_R| \quad \text{for R=1,2,3 ……16, } R \neq \text{ most probable region} \qquad (4.11)$$

where q_m is the direction of the most probable motion vector, and q_R is the mean direction of region R, which is calculated as follows:

$$q_m = \tan^{-1}\left(\frac{MV_{my}}{MV_{mx}}\right) \qquad (4.12)$$

$$q_R = \frac{q_{R(\max)} + q_{R(\min)}}{2} \qquad (4.13)$$

Here, MV_{mx} and MV_{my} are horizontal and vertical component of the most probable motion vector found in Table 4.4. For example, the most probable region is 3 and $q_m = 50^0$, followed by a search order of 3,0,2,4,1,5,16,6,15,7,14,8,13,9,12,10 and 11, which becomes slightly different when the most probable region is 0. In this case, the search starts from region 0 and the order of the remaining 16 search regions is calculated based on (4.11).

The process of constructing the region order for each mode of an MB in the current frame becomes an additional computation introduced by the region-based search. The value of q_R in (4.13) is constant throughout the whole encoding process, and q_R is calculated one time before encoding begins. Although the total number of modes in an MB is 7, the arc-tan operator in (4.12) should be calculated 3 times for each MB because MV_{mx} and MV_{my} in (4.12) are updated 3 times (clearly shown in Table 4.4). Table 4.3 illustrates how about 77% of the MV for the 16x16 mode of the current MB is the same as that of the 16x16 mode for the collocated MB in the previous frame, while about 76% of the MV for the 8x8 mode is equal to that of the 16x16 mode from the current MB. Since the arc-tan operator is time consuming, we have used these observations to reduce the number of times the search order must be updated. This updated search order procedure is implemented as follows:

For each MB:

- Calculate (4.11) for 16x16 mode and store the search order in memory.
- Perform the 16x16 mode motion estimation.
- If the MV of the 16x16 mode does not equal that of the previous frame's 16x16 mode, update the search order for 16x8, 8x16 and 8x8 modes based on (4.11).
- Perform the ME for the 16x8, 8x16 and 8x8 modes.
- If the MV of the 8x8 mode does not equal that of the 16x16 mode, update the search order for the 4x8, 8x4 and 4x4 modes based on (4.11).
- Perform the ME for the 4x8, 8x4 and 4x4 modes.

End for

It should be noted that the search center of the proposed region-based search algorithm must be at (0, 0) MV. Some existing motion estimation methods use motion vector prediction to predict the search center and reduce the computation, but the proposed region-based method is not suitable for these methods because the proposed method uses two types of patterns that depend on the content of motion. The proposed method uses a square pattern for low motion, and a directional pattern for high and medium motion. If an MB has a predicted MV of (8, 6) the proposed method would use a directional pattern, but if the search center prediction method were merged with the proposed region-based search, an inefficient square pattern would be used for this MB. That's why the proposed method must have a search center of (0, 0) MV.

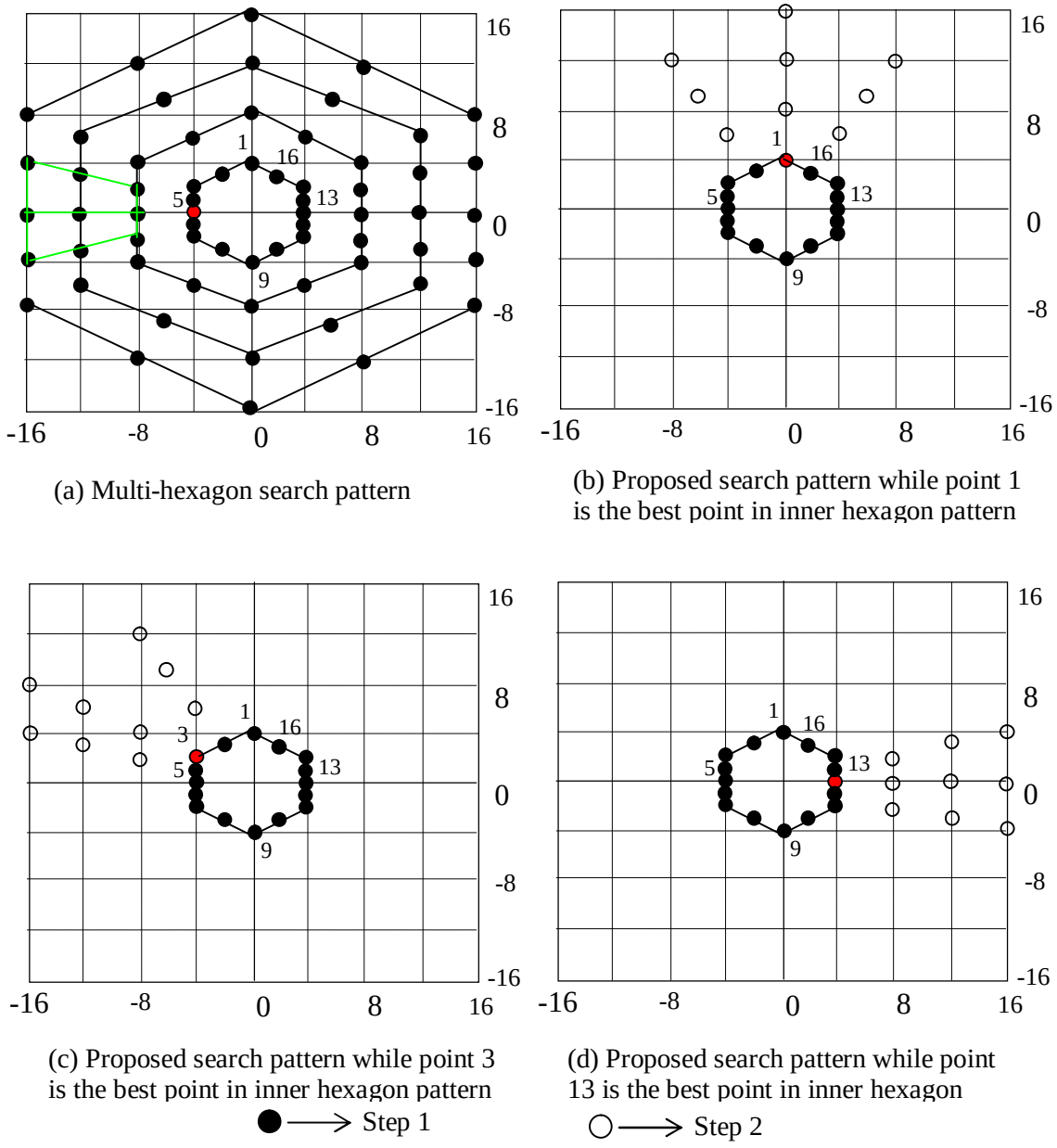
Table 4.6 Rate of four different hexagons % (order: inner to outer)

Sequence(CIF)	Hexagon 1	Hexagon 2	Hexagon 3	Hexagon 4
Foreman	79.58	10.01	6.25	4.16
Stefan	62.89	16.21	12.12	8.78
Akiyo	74.23	12.26	7.78	5.73
Bus	60.14	17.11	12.32	10.43
Mobile	58.82	17.19	14.98	9.01
Flower	53.11	19.89	13.25	13.75
Paris	64.02	15.56	11.22	9.20
Silent	72.31	12.24	9.16	6.29
Average	65.63	15.05	10.88	8.41

4.6 Search Point reduction of Multi-Hexagon Grid Search

A multi-hexagon grid search strategy is taken in fast integer pel motion estimation in JVT [50]. This algorithm has four different steps. In the third step, this method uses four different hexagons as shown in Fig. 6(a). Encoder searches all of the 64 points and the point with minimum RD-cost is chosen as best motion vector. Table 4.6 shows the rate

(= *100*Number of best motion vector in a particular hexagon/Total number of best motion vector in all hexagon*) of 4 different hexagons. The order of the hexagon is chosen from inner to outer.



(a) Multi-hexagon search pattern

(b) Proposed search pattern while point 1 is the best point in inner hexagon pattern

(c) Proposed search pattern while point 3 is the best point in inner hexagon pattern

(d) Proposed search pattern while point 13 is the best point in inner hexagon

● ⟶ Step 1

○ ⟶ Step 2

Fig. 4.5 Proposed search pattern

It is shown that about 65% (on average) of motion vectors fall in most inner hexagon. For this reason, we choose the most inner hexagon as a most significant hexagon and first search all of the 16 points of this hexagon. It is reasonable to say that RD costs of neighboring search points are similar. Assume, the best search point in most inner hexagon is point 5 which is indicated as red color in Fig 4.5(a). The best points of other hexagons should be one of the neighboring points of point 5 in inner hexagon. Based on this idea, the proposed method only searches 9 neighboring points other hexagons. The green line in Fig. 4.5(a) shows the neighboring points of point 5 of most inner hexagon. Fig. 4.5 (b-d) shows the proposed search pattern with different best point in the inner hexagon. The proposed method first searches the 16 points in the inner hexagon to find the best point with a minimum RD cost, then the 9 neighboring points to that best point are searched.

4.7 Simulation Results

To evaluate the performance of the proposed method, JM 9.6 [66] reference software is used in the simulations and different types of video sequences are used as test materials.

Table 4.7 Simulation conditions

GOP structure	IPPP
Quantization parameters	16/20/24/28/32
Number of reference frame	1
Hadamard transform	OFF
Search range	16
Symbol mode	CAVLC
RDO optimization	ON
Frame rate	30/60 fps
Number of frames	100

A group of experiments were carried out on test sequences with different quantization parameters. All simulations were conducted using the Windows Vista operating system, with a Pentium 4 2.2 G CPU and 1 G RAM. The simulation conditions are listed in Table 4.7. Detailed parameters setting are given in Appendix *A*. Comparison results were produced and tabulated based on the average difference in ME time (ΔT %), average difference in search points (ΔSP %), the average PSNR differences ($\Delta PSNR$) and the average bit rate differences ($\Delta Rate$ %). PSNR and bit rate differences were calculated according to numerical averages between RD curves derived from the original and proposed algorithms, respectively. The detailed procedure for calculating these differences can be found in [67]. In order to evaluate the ME time reduction, ΔT (%) is defined as follows:

$$\Delta T\% = \frac{T_{original} - T_{proposed}}{T_{original}} \times 100\% \tag{4.14}$$

where, $T_{original}$ denotes the ME time of the JM 9.6 encoder and $T_{proposed}$ is the ME time of the encoder in the proposed early termination method. The motion search time of an algorithm could vary depending on the simulation environment. To eliminate this influence, the search points used in the different algorithms were ones that could be used as an alternative criterion. The percentage of average search point reduction is calculated as follows:

$$\Delta SP\% = \frac{SP_{original} - SP_{proposed}}{SP_{original}} \times 100\% \tag{4.15}$$

where, $SP_{original}$ and $SP_{proposed}$ are the average number of search points per MB of the original method and the proposed method, respectively.

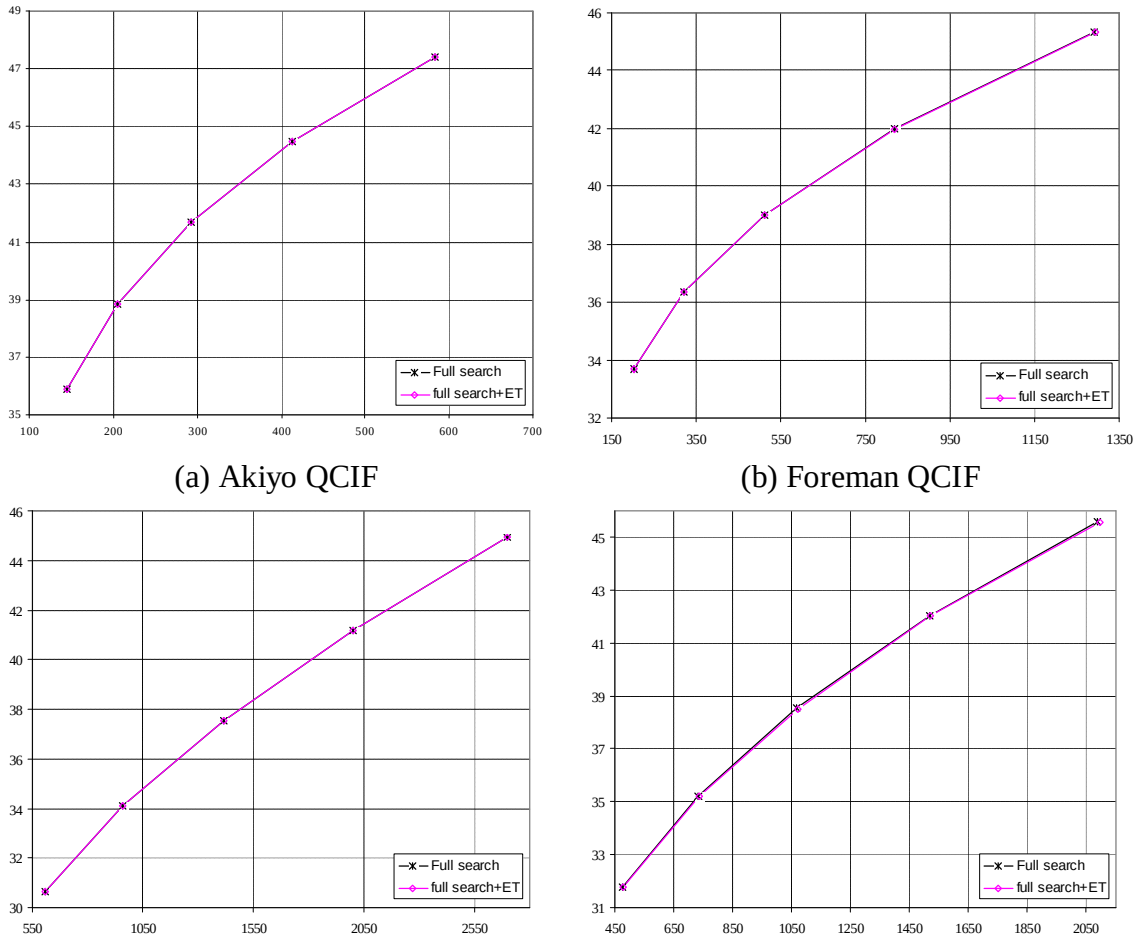
Table 4.8 Comparison with full search motion estimation

Format	Sequence	ΔPSNR	ΔRate%	Δ*T* %	Δ*SP* %
QCIF (30 fps)	Akiyo	0.007	0.017	81.76	97.61
	Claire	0.005	0.138	80.34	97.25
	Carphone	0.011	0.353	74.33	92.03
	Coastguard	0.003	0.070	74.75	93.11
	Container	0.006	0.186	76.44	94.43
	Foreman	0.002	0.004	75.43	93.48
	Mobile	0.003	0.056	73.93	87.24
	Stefan	0.025	0.566	77.86	89.53
CIF (60 fps)	News	0.009	0.351	79.44	98.07
	Paris	0.013	0.529	76.15	94.22
	Bus	0.024	0.892	79.10	90.72
	Hall	0.003	0.707	77.77	96.78
Average		0.009	0.322	77.27	93.70

4.7.1 Experiments with Full Search (FS) ME

This experiment applies the proposed early termination (ET) method to the full search motion estimation of H.264/AVC. We chose twelve sequences with motion activities varying from small, to large and the comparison results were tabulated in Table 4.8, which shows how the proposed algorithm yields 77% ME time savings on average, compared to a full search ME with a negligible PSNR reduction (0.009 dB) and bit rate increment (0.32%). Table 4.8 also lists the search point's comparison between the proposed method and the full search method. The reduction percentage of the search points (93%) is clearly not in accordance with the reduction percentage of the ME time (77%) because in addition to block matching operations, there are other extra time-consuming aspects of fast algorithms (e.g., reading and writing memories and switching). For slow motion sequences like *Akiyo, Claire* and *News,* the proposed algorithm saves about 80% of ME time and 97% of search points. The computation

reduction is high because most of the motion vectors for these types of sequences are around the search center, and the encoder gets the best point after searching only a few of the points. The RD curves of full search ME and full search with proposed early termination (ET) is presented in Fig. 4.6. We can see that the RD curves of proposed method closely match with original full search method.



(a) Akiyo QCIF (b) Foreman QCIF

(e) Mobile QCIF (f) Stefan QCIF

Fig. 4.6 RD curves of full search method and proposed early termination method.

4.7.2 Experiments with UMHS

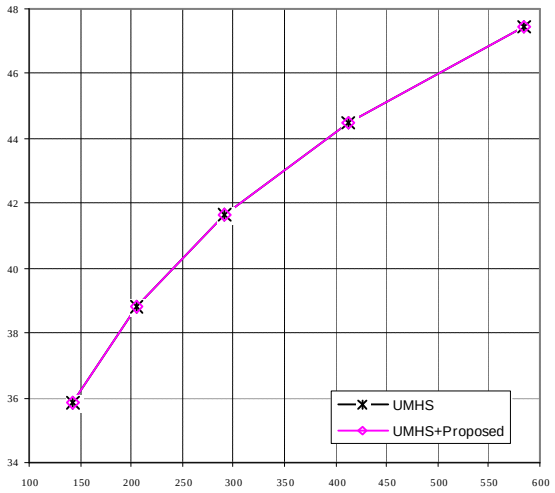
This experiment applies our early termination method to the UMHS [50, 64] of H.264/AVC, and the comparison results are tabulated in Table 4.9. UMHS is used as a

benchmark, and Table 4.9 shows how the computational complexity of the proposed scheme is much less than that of the UMHS. The proposed scheme saves about 31% of ME time and 68% of search points compared to UMHS. The rate distortion performance degradation is also very negligible. PSNR decrement is about 0.006 db, and the bit rate increment is about 0.16%. The comparisons of RD curves are presented in Fig. 4.7. The RD curves of proposed method are closely matched with that of UMHS.

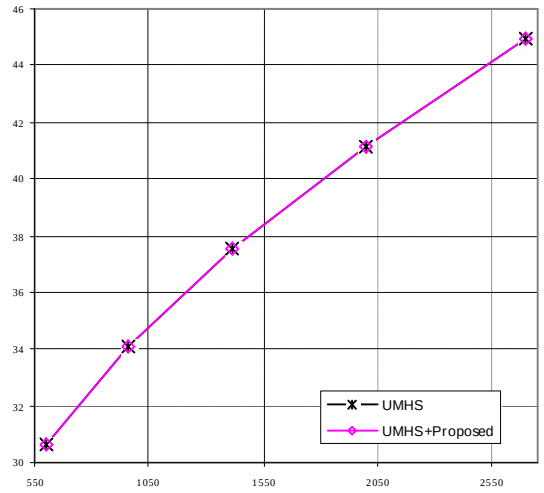
Table 4.9 Comparison with UMHS

Format	Sequence	Δ PSNR	Δ Rate%	Δ*T* %	Δ*SP* %
QCIF (30 fps)	Stefan	0.014	0.285	31.73	64.59
	Foreman	0.004	0.154	26.46	66.43
	Akiyo	0.002	0.052	27.80	69.40
	Mobile	0.001	0.029	32.44	62.08
	Carphone	0.006	0.184	29.93	66.24
	Claire	0.005	0.173	33.82	73.58
	Coastguard	0.009	0.205	30.46	67.98
	Container	0.005	0.130	35.11	74.96
CIF (30 fps)	Foreman	0.011	0.414	28.56	69.07
	Mobile	0.005	0.010	31.65	64.47
	Coastguard	0.002	0.077	35.40	71.14
	Container	0.009	0.027	29.08	68.06
	Flower	0.012	0.227	29.65	65.13
CIF (60 fps)	News	0.003	0.106	27.64	73.07
	Paris	0.008	0.203	31.11	71.79
	Bus	0.014	0.252	31.42	70.11
	Hall	0.008	0.357	37.02	72.97
Average		0.006	0.169	31.13	68.88

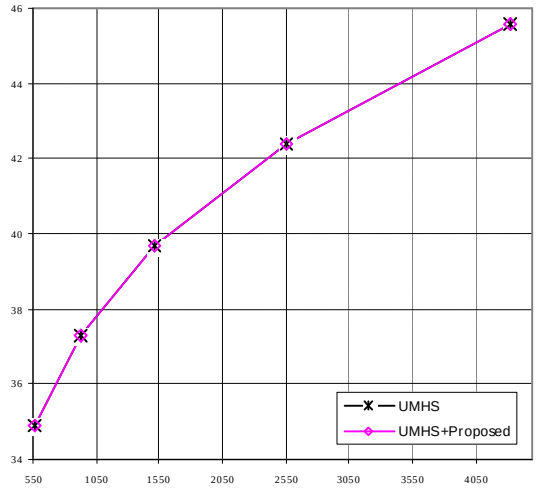
Table 4.10 compares the proposed method with the early termination methods described in [54] and [58].The RD performance of the proposed method proves better for most sequence types, but for the complexity of a low motion sequence like *Claire*, other methods are more efficient despite significant performance degradation. The proposed method is more efficient in terms of quality, bit rate and complexity for medium and high motion sequences.



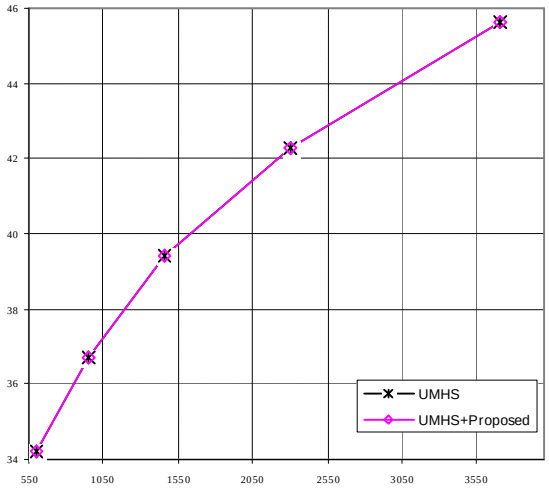
(a) Akiyo (QCIF)



(b) Mobile (QCIF)



(c) Foreman (CIF)



(d) Container (CIF)

Fig. 4.7 RD curves of FME and proposed method.

Table 4.10 Comparison with other methods at 30 fps

Sequence (QCIF)	Δ PSNR			Δ Rate%			Δ*SP* %		
	Proposed	ET [54]	ET [58]	Proposed	ET [54]	ET [58]	Proposed	ET [54]	ET [58]
Stefan	0.014	0.15	0.057	0.285	3.05	1.16	64.59	24.31	32.14
Mobile	0.001	0.10	0.064	0.029	1.92	1.22	62.08	35.17	35.21
Claire	0.005	0.13	0.055	0.173	3.89	1.60	73.58	81.16	82.14
Coastguard	0.009	0.14	0.058	0.205	3.64	1.47	67.98	58.20	44.59
Average	0.007	0.13	0.058	0.173	3.12	1.36	67.06	49.71	48.52

4.8 Summary

This chapter develops a simple and fast early termination method for the motion estimation of H.264/AVC. The adaptive thresholds are based on the correlation between the RD costs of current blocks and neighboring and collocated blocks. This correlation with other modes is also considered during threshold selection. A method based on the region-based search order is presented for a full search ME and a fast algorithm for a multi-hexagon grid search is also developed. The simulation results verified that the proposed technique is suitable for H.264/AVC full search and fast motion estimations, and the proposed scheme allows most of the search points to be skipped in the early stages of the motion estimation process. The proposed method saves about 93% and 69% of search points, compared to H.264/AVC's typical full search and fast motion estimations, respectively. The proposed algorithm can also be applied to any other conventional, fast motion estimation method to further reduce its computation.

Chapter 5

Adaptive Search Area Selection

Variable block size motion estimation (VBME) is a new feature introduced in H.264/AVC video coding standard. VBME plays a significant rule in achieving outstanding performance in compression efficiency and video quality. However, the VBME is the most time consuming part of H.264/AVC encoder. In order to reduce computations in motion estimation module, this chapter presents a novel adaptive search area selection method by utilizing the information of the previously computed motion vector differences (MVDs). The direction of picture movement of the previously computed blocks is also considered for search area selection. In this algorithm, narrow search ranges are chosen for areas in which little motion occurs and wide ranges are chosen for areas of significant motion. Experimental results show that the proposed algorithm provides significant improvement in coding speed with negligible objective quality degradation compared to the full search motion estimation method adopted by H.264/AVC reference software.

5.1 Literature Review

As a result of rapid development of digital techniques and increasing use of internet, image and video compression plays a more and more important role in our life. H.264/AVC [16] is the newest video coding standard which offers a significant performance improvement over previous video coding standards such as H.263++ and MPEG-4 part 2. But the complexity of the encoder, especially motion estimation (ME) module, is extremely high.

One of the ways to reduce the complexity of ME process is to adaptively change the search range [101-107]. In [101], the size of search range for each block is adaptively controlled according to the initial cost at the search center, and then the full search algorithm is performed to find the optimal matching block in the adjusted search range. The summation of absolute value of previously computed motion vectors and summation of prediction errors are used to change the search range adaptively [104]. In [105], the search range is determined at each block level by using local statistics of neighboring blocks motion vectors (MVs). Unlike in [104] the MVs are directly used irrespective of their size, in [106], three MVs are firstly used together to calculate two statistics, and then based on these statistics, decision on search range is made. However, the fact is, high probability of large MV does not mean a large search window is required, because the search window may not be centered on the (0, 0) vector, it may centered on the search center placed by median Motion Vector Predictor (MVP).

This chapter proposes an efficient adaptive search area selection algorithm of H.264/AVC encoder. In this method, instead of MVs, motion vector differences

(MVDs) are used. The inter mode correlation is also considered to select the search area. Based on the MVDs of previously computed modes, the search area is changed in block level. Search area is also inspired by the orientation of the previously computed MVs.

5.2 Proposed Adaptive Search Area Selection

In H.264/AVC video coding, the search range is fixed throughout the encoding process. This constant search range is inefficient for slow motion macroblocks because the distance between best MV and search center is very small. But for large motion MBs, a large search range is more efficient. Therefore, an adaptive search range decision algorithm is necessary for H.264/AVC video coding.

Since an object usually occupies more than one block of an image, motions of the neighbouring blocks are highly correlated. Also due to the inertia of the moving objects, there is a correlation among motion vectors of blocks of consecutive frames. Consequently if the MVs of the previously computed MBs are all high then current MB is highly probable to have large MV. However, the fact is, high probability of large MV does not mean a large search window is required, because the search window is not centered on the (0, 0) vector, it is centered on the search center placed by median motion vector predictor (MVP). For example, if the MVs of all adjacent MBs is (12, 11) and the MV of current MB is also (12, 11), the real situation is that the search center is calculated out as (12, 11) thus a search range of zero will be large enough for finding the correct MV (12, 11). Based on this idea, the previously computed motion vector differences (MVDs) are used to predict the search area of a current MB or mode. In the

proposed method, the adaptive search range (ASR) is defined as the weighted average of the previously calculated MV differences.

Table 5.1 Previously computed motion vectors (MVx, MVy) and corresponding weighting factors (Wi) of equation (5.1) and (5.2)

Mode	**Previously computed MVs and corresponding weighting factor (w_i)**
16x16	Collocated (2), up (2), left (2), up-left (1), up-right (1)
16x8	16x16 (2), Collocated (2), up (2), left (1), up-left (1)
8x16	16x16 (2), 16x8 (2), Collocated (2), up (1), left (1)
8x8	16x16 (2), 16x8 (2), 8x16(2), Collocated (1), up (1)
8x4	8x8 (2), 16x16 (2), 16x8 (2), 8x16(1), Collocated (1)
4x8	8x8 (2), 8x4 (2), 16x16 (2), 16x8 (1), 8x16(1)
4x4	8x8 (2), 8x4 (2), 4x8 (2), 16x16 (1), 16x8 (1)

Weighted average of horizontal and vertical components of most correlated MVs and the adaptive search range (ASR) are calculated as follows

$$Av\_MVD_x = \left\lceil \frac{\sum_{i=1}^{5} w_i (MV_{x(i)} - MVP_x)}{\sum_{i=1}^{5} w_i} \right\rceil \tag{5.1}$$

$$Av\_MVD_y = \left\lceil \frac{\sum_{i=1}^{5} w_i (MV_{y(i)} - MVP_y)}{\sum_{i=1}^{5} w_i} \right\rceil \tag{5.2}$$

$$ASR = \max(\left|Av\_MVD_x\right|, \left|Av\_MVD_y\right|) + \Delta R \tag{5.3}$$

Where, (MV_x, MV_y) are the horizontal and vertical components of previously computed MVs , (MVP_x , MVP_y) is the motion vector predictor, (Av_MVD_x, Av_MVD_y) is the weighted average motion vector difference, ASR is the adaptive search range, w is the weighting factors, and ΔR is the penalty function. The notation $\lceil\ \rceil$ denotes rounding

up to next integer. Table 5.1 shows the candidate MVs and their corresponding weighting factors for ASR calculation of different modes.

In H.264/AVC motion estimation process, the processing order of different modes is 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. So, during motion estimation process of 16x8 mode, the information about the best MV of 16x16 mode is available. Similarly while processing 4x4 block, the best MVs and best RD costs of all other modes are already computed. So we can estimate the search area of lower size block from motion information of the higher size block of same MB. In case of 16x16 mode, the information about MV of other modes for same MB is not available. But MVs of neighboring MBs and collocated MBs in the previous frames are already computed.

In Table 5.1, the corresponding weighting factors of the set of the MVs are given within braces. Motion vector of 16x16 mode is highly correlated with that of collocated, up and left MB as compared to up-left and up-right MVs [108]. Therefore, weighting factors of collocated, up and left modes should be larger than that of up-left and up-right MVs. Similarly, motion vector of 4x4 mode is highly correlated with that of 8x8, 8x4, and 4x8 mode [108]. That's why the weighting factors of these modes are higher than other modes. Similar observation is applied to select weighing factors of all other modes.

In order to select a penalty function ΔR, we have used the initial cost of the search center (J_{MVP}). If J_{MVP} is low, it means that MVP is probably close to best MV. On the other hand, if J_{MVP} is high, it means that MVP is not so correlated with current MV, and thus the size of the search range should be larger. Based on this observation ΔR is defined as follows:

$$\Delta R = \begin{cases} 0 & \text{if } E <= 2 \\ 1 & \text{otherwise} \end{cases} \tag{5.4}$$

and $$E = \frac{J_{MVP}}{M \times N} \tag{5.5}$$

where E is the average pixel error, J_{MVP} is the RD cost of the search center and MxN is the size of the block.

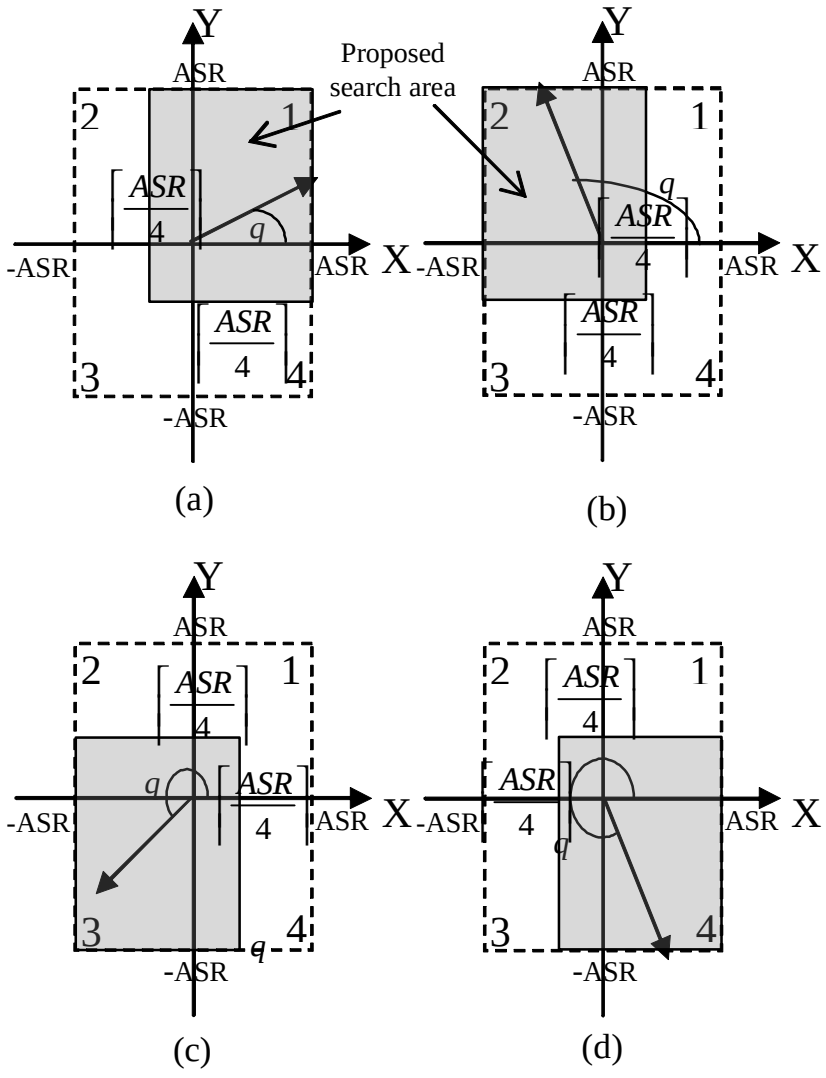


Fig. 5.1 Proposed search area with (a) most probable quadrant = 1, (b) most probable quadrant = 2, (c) most probable quadrant = 3, (d) most probable quadrant =4

Dotted squares in Fig. 5.1 indicate adaptive search window. Search window is square which implies that the search range is same in both horizontal and vertical directions but motion vectors of real video sequences are directional oriented. If all of the correlated blocks move in a particular direction, the movement of current block should be also in the same direction. Therefore a square search window is not efficient and in order to tackle this issue, we have used orientation of weighted average motion vector, calculated as follows:

$$q = \tan^{-1}(Av\_MVD_y / AV\_MVD_x) \tag{5.6}$$

If q falls in first quadrant, the MV of current block is most likely to be in the first quadrant. In this paper, the quadrant at which q falls is called the most probable quadrant. Therefore, search ranges of other quadrants are reduced from ASR to $\lceil ASR/4 \rceil$. Here $\lceil\ \rceil$, means rounding up to integer value. Grey area in Fig. 5.1 shows the modified search window for different values of most probable quadrants.

5.3 Simulation Results

In order to verify the effectiveness of the proposed adaptive search area selection method, the proposed algorithm is integrated in JM 9.6 [66] reference software. Different types of video sequences with CIF format are used as test materials. Each sequence has 100 frames; hence, they represent a wide range of motion contents. The simulation environment and parameter setting are tabulated in Table 5.2 and rests of the parameters are settled as the default of the encoder.

Table 5.2 Simulation Conditions

GOP Structure	IPPPP../IBPBP..
Quantization Parameters	12/16/20/24
Number of Reference Frame	1
Hadamard Transform	OFF
Search range for FS	16
Symbol mode	CAVLC
RDO optimization	ON
Frame rate	30/60 fps
Number of frames	100

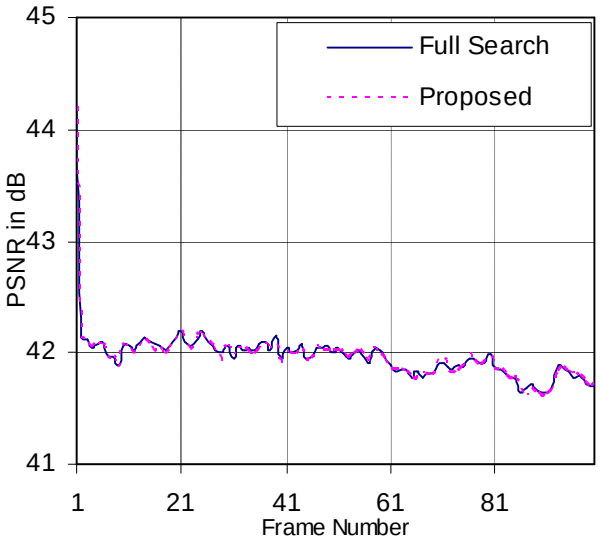
All the simulations are conducted under Windows Vista operating system, with Pentium 4 2.2 G CPU and 1 G RAM. The comparison results are produced and tabulated based on the average difference of integer pixel ME time, the average PSNR differences ($\Delta PSNR$), and the average bit rate differences ($\Delta Rate\%$). PSNR and bit rate differences are calculated according to the numerical averages between RD curves derived from the original and proposed algorithm, respectively [67].

5.3.1 Comparison with full search ME

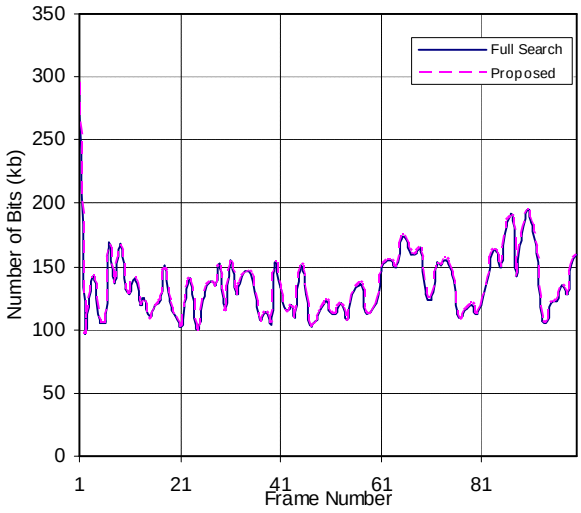
In this experiment, the proposed method is compared with full search ME of H.264/AVC. Both IPPPP and IPBPBP sequences are used in simulations. Comparison results with IPPPP sequences are tabulated in Table 5.3. The positive values mean increments whereas negative values mean decrements. The complexity reduction $\Delta T_1\%$ is calculated as follows and average values presented.

$$\Delta T_1\% = \frac{T_{proposed} - T_{FS}}{T_{FS}} \times 100\% \tag{5.7}$$

where, T_{FS} denotes the integer pixel ME time of the JM 9.6 encoder with full search ME and $T_{proposed}$ is the integer pixel ME time of the encoder in the proposed method.



(a)

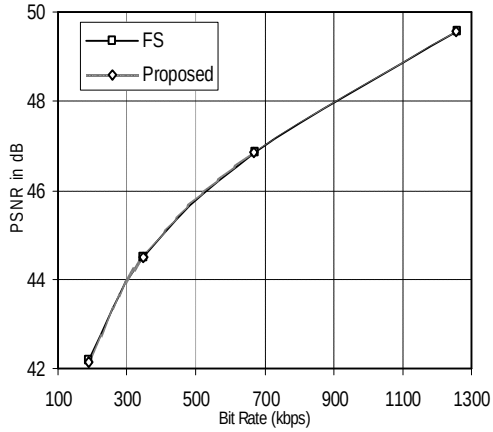


(b)

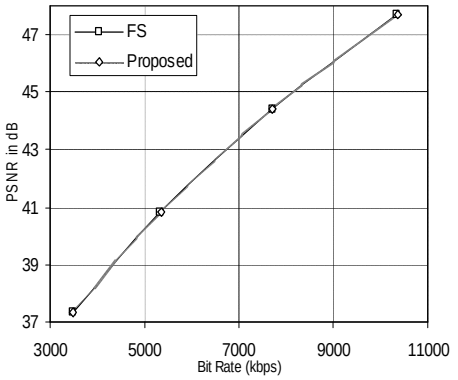
Fig. 5.2 Frame by frame comparison of the proposed method with FS motion estimation of Flower video sequence (a) PSNR Comparison (b) Bit rate Comparison

Fig. 5.2 shows the frame by frame comparison of the proposed method with FS with IPPPP.. sequences in terms of PSNR and number of bits. Both PSNR and bit rate of the proposed method are almost same as FS. In these plots, 100 frames of *Flower* video

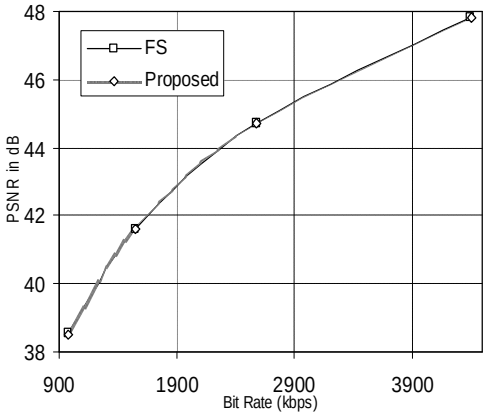
sequence are encoded with quantization parameter equal to 20. The PSNR and rate of the first frame is higher than others because first frame is encoded with intra-coding and remaining frames are encoded with inter-coding.



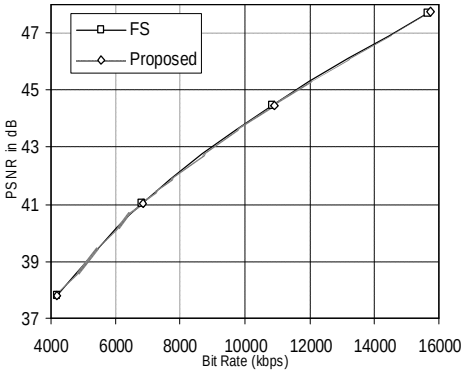
(a) Akiyo@30 fps



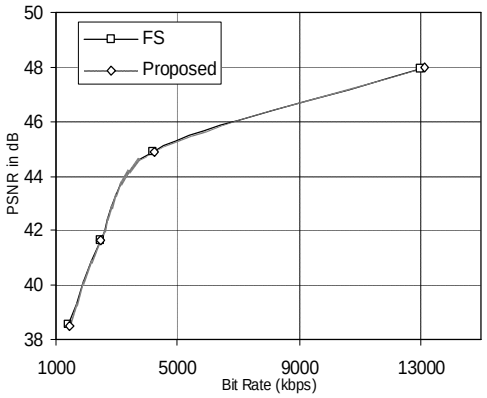
(b) Mobile@30 fps



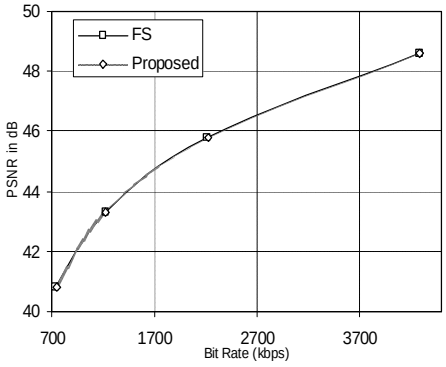
(c) Paris@30 fps



(d) Bus@60 fps

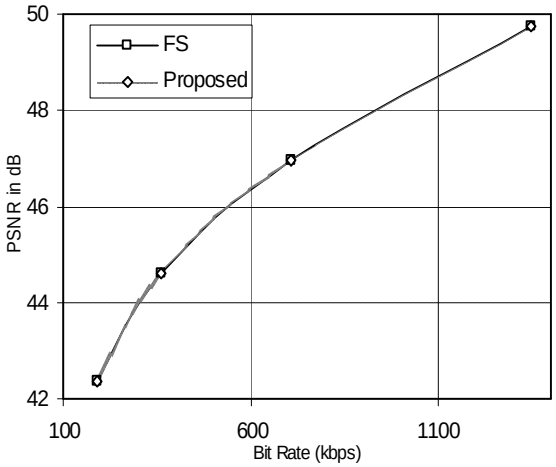


(e) Tennis@60 fps

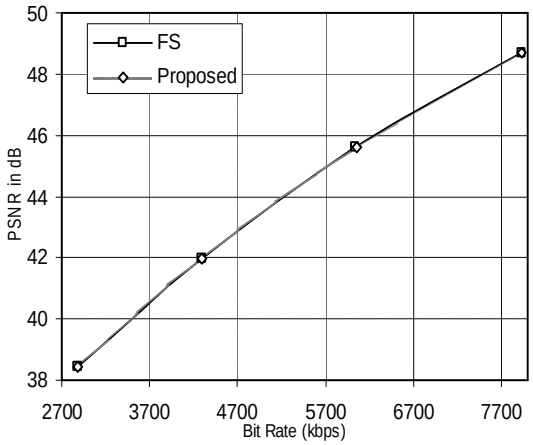


(f) News@60 fps

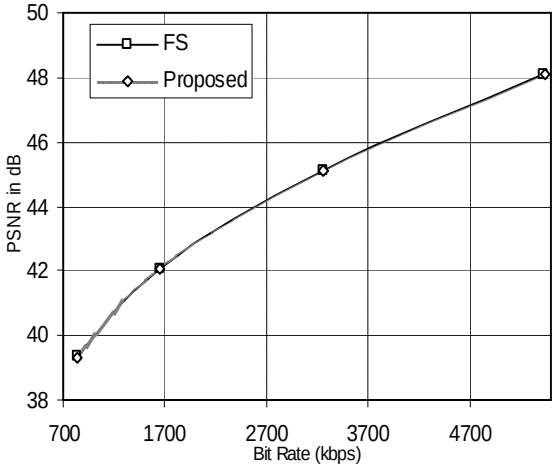
Fig. 5.3 Rate-distortion (RD) curves of different sequences with IPPP.. frames



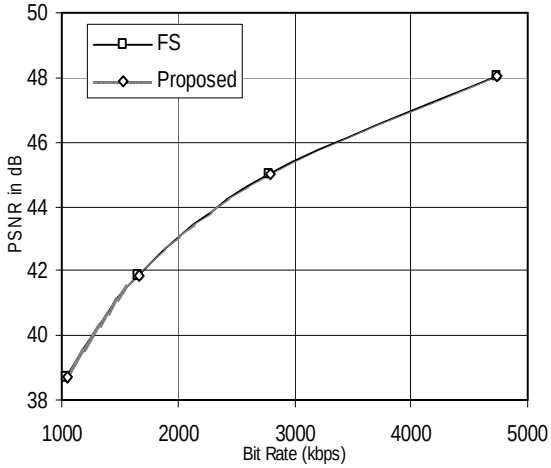
(a) Akiyo@30 fps



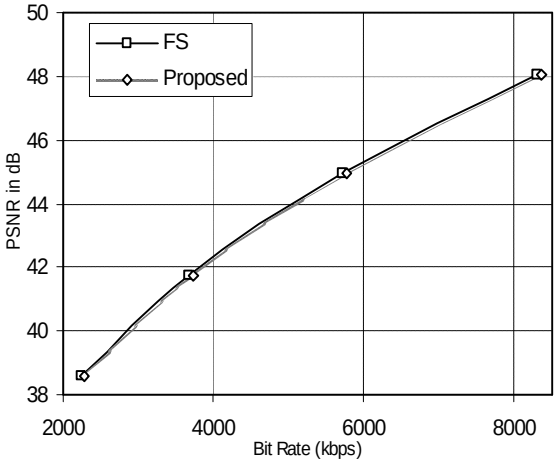
(b) Flower@30 fps



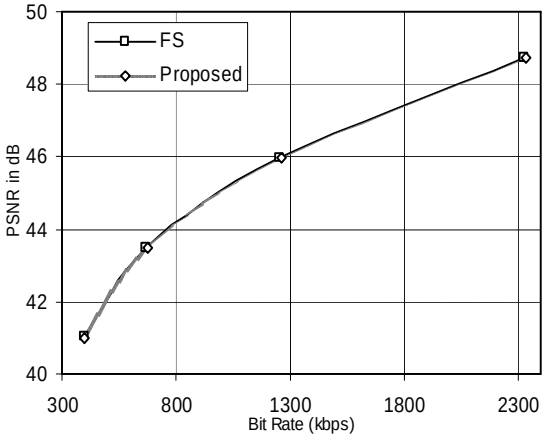
(c) Foreman@30 fps



(d) Paris@30 fps



(e) Stefan@30 fps



(f) News@30 fps

Fig. 5.4 Rate-distortion (RD) curves of different sequences with IPBPBPB.. frames

From Table 5.3, it is clear that the proposed algorithm yields up to 92% of integer pixel ME time savings, compared to a full search ME of IPPP frames with a negligible PSNR reduction (0.015 db) and bit rate increment (0.47%). The speed up of the proposed algorithm is quite significant for various motion videos and on average the proposed method saves 84% computational cost in comparison with full search ME. The computation reduction is more pronounced and an utmost 92% of FS for low motion sequence such as Akiyo and News. Most of the motion vectors of these sequences are around the search center so the adaptive search range is low.

Table 5.3 Comparison with full search ME with IPPP.. sequences

Frame rate	Sequence	ΔPSNR	ΔRate%	ΔT_1%
30 fps	Foreman	-0.003	0.178 %	-82.28 %
	Paris	-0.018	0.591 %	-84.24 %
	Stefan	-0.023	0.689 %	-74.79 %
	Flower	-0.015	0.317 %	-81.27 %
	Akiyo	-0.007	0.005 %	-92.65 %
	Mobile	-0.002	0.060 %	-86.57 %
60 fps	News	-0.017	0.773 %	-88.37 %
	Bus	-0.031	0.860 %	-82.95 %
	Tennis	-0.019	0.797 %	-84.98 %
Average		-0.015	0.474 %	-84.23 %

Table 5.4 Comparison with full search ME with IPBPBP.. sequences

Sequence	ΔPSNR	ΔRate%	ΔT_1%
Foreman	-0.010	0.618	-81.26
Paris	-0.034	1.140	-86.28
Stefan	-0.046	1.206	-73.18
Flower	-0.013	0.280	-82.24
Akiyo	-0.006	0.359	-92.84
News	-0.041	1.910	-85.25
Average	-0.025	0.918	-83.50

The RD curves for full search ME and the proposed method of different sequences at dissimilar frame rates are presented in Fig. 5.3. In this figure IPPP sequences are used. We can see that the RD curves of proposed method closely follow the original full search method. As shown in Table 5.4, the proposed method saves about 83% of computation of IBPBPBPBP sequences with negligible degradation in RD performance. The RD curves comparison of B frame coding is presented in Fig. 5.4. The RD curves of the proposed method closely match with the original standard.

5.3.2 Comparison with other methods

In this experiment, the proposed method is compared with two different methods in terms of RD performance and complexity. The complexity reduction $\Delta T_2\%$ is calculated using (5.8) and average results presented.

$$\Delta T_2\% = \frac{T_{proposed} - T_{oth}}{T_{oth}} \times 100\% \tag{5.8}$$

where, T_{oth} denotes the integer pixel ME time of the JM 9.6 encoder with other method and $T_{proposed}$ is the integer pixel ME time of the encoder in the proposed method. Table 5.5 shows the comparison of our proposed method with search range decision method presented in [104] and [106]. It is shown that as compared with reference [104], our algorithm reduces bit rate by about 2.58 % and achieve 0.16 dB improvements in PSNR on average with up to 26% computation reduction. The proposed algorithm not only saves up to 36% computation time of [106] but also improves the rate-distortion performance.

Table 5.5 (a) Comparison with [104]

Sequence	Δ PSNR	Δ Rate%	ΔT_2 %
Foreman	0.18	-3.77 %	-26.05 %
Paris	0.19	-2.23 %	-22.27 %
Stefan	0.10	-2.93 %	-25.28 %
Flower	0.13	-2.76 %	-20.24 %
Akiyo	0.20	-1.24 %	-8.17 %
Average	0.16	-2.58 %	-20.40 %

Table 5.5 (b) Comparison with [106]

Sequence	Δ PSNR	Δ Rate%	ΔT_2 %
Foreman	0.03	-1.47 %	-36.34 %
Paris	0.03	-1.24 %	-23.58 %
Stefan	0.06	-1.90 %	-18.88 %
Flower	0.01	-0.31 %	-21.81 %
Akiyo	0.005	-0.28 %	-18.84 %
Average	0.135	-1.04 %	-23.89 %

5.4 Summary

In this chapter, we have proposed an adaptive search range decision algorithm for variable block size motion estimation strategy based on motion vector differences of neighboring and collocated blocks. The adaptive search range method determines the size of the search range dynamically. The proposed method is very simple and efficient and experimental results confirm that the proposed method not only cuts down the computational complexity of other methods in literature but also improves coding efficiency and picture quality as well. The proposed method can be easily applied to many mobile video application areas such as a digital camera.

Chapter 6

Improved Intra Prediction

In H.264/AVC intra coding, each 4x4 luma block is predicted as either one of the eight directional prediction modes or DC mode. DC mode is used to predict regions with no unified direction and the predicted pixel values are same and thus smooth varying regions are not well de-correlated. In order to address this issue, this chapter proposes an improved DC prediction (IDCP) mode based on the distance between the predicted and reference pixels. On the other hand, using the nine prediction modes in intra 4x4 and 8x8 block units can reduce spatial redundancies, but it may needs a lot of overhead bits to represent the prediction mode of each 4x4 or 8x8 block. In order to reduce the number of overhead bits and computational cost, an intra prediction method is also proposed in this chapter. In this method, the number of prediction modes for each 4x4 and 8x8 block is selected adaptively. This chapter also proposes an algorithm to estimate the most probable mode (MPM) of each block. The MPM is derived from the prediction mode direction of neighboring blocks which have different weights according to their positions. The proposed methods not only improve the Rate-Distortion (RD) performance but also reduce the computational complexity of H.264/AVC intra encoder.

6.1 Literature Review and Challenges

The Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG has finalized a new standard for the coding of natural video known as H.264/AVC [16, 18]. The performance of H.264/AVC is superior to all conventional video coding standards in terms of peak-signal-to-noise ratio (PSNR) at same bit rate. New and advanced techniques are introduced in this new standard, such as intra prediction for I-frame encoding, multi-frame inter prediction, small block-size transform coding, context-adaptive binary arithmetic coding (CABAC), de-blocking filtering, etc. H.264/AVC offers a rich set of prediction patterns for intra prediction, i.e. nine prediction modes for 4x4 luma block, nine prediction modes for 8x8 luma block and four prediction modes for 16 x 16 luma block. However, the rate-distortion (RD) performance of the intra frame coding is still lower than that of inter frame coding. Hence intra frame coding usually requires much larger bits than inter frame coding which results in buffer control difficulties and/or dropping of several frames after the intra frames in real-time video. Thus the development of an efficient intra coding technique is an important task for overall bit rate reduction and efficient streaming.

A number of researchers have proposed methods to improve the RD performance of intra coder [109-122]. A hybrid block matching is proposed in [109] where a block searching scheme is used instead of simply performing directional prediction, this algorithm is time consuming. A new intra coding method for sample predictor creation by template matching in a region of reconstructed pixels is presented in [110]. This concept is further evaluated and refined in [111]. An extended Markovian prediction scheme that comprises of two additional prediction methods exploiting self similar

properties of encoder texture is introduced in [112]. Improved performance is achieved at the cost of substantial increase in complexity, especially taking into account the decoder side of the Markovian prediction. A block based extra/inter-polation method by changing the sub-block coding order in a Macroblock (MB) is proposed in [113]. Since three additional directional predictions are utilized, the computational complexity of intra prediction increases drastically and the number of overhead bits required to represent the prediction mode also increases. Bi-directional intra prediction (BIP) [114] method combines two existing prediction modes to form bi-directional prediction modes. If all possible combination of two prediction modes are used, then there will be a total of 45 intra modes i.e. 36 BIP modes and 9 existing modes. Too many intra modes not only increase the computational complexity but also increase the number of overhead bits required to code the modes due to the use of longer codeword. A simplified BIP prediction method allows limited set of BIP modes [115]. In order to improve the performance of DC prediction, a distance based weighted prediction (DWP) method is introduced in [116, 117]. DWP considered that correlation exist in vertical and horizontal directions only. But in natural images, smoothly variations of pixels are not limited to combination of horizontal and vertical directions only. The predicted pixels may be correlated with any orientations. In [118], a recursive prediction scheme and an enhanced block matching algorithm prediction scheme are designed and integrated into the state-of-the-art H.264/AVC framework to provide a new intra coding model. A spatial gradient based intra prediction is presented in [119] that use multiple reference line to reduce the residual energy but the total increase in computational cost was about 39 times of conventional encoder.

H.264/AVC uses rate-distortion optimization (RDO) technique to get the best coding mode out of nine prediction modes in terms of maximizing coding quality and minimizing bit rates. This means that the encoder has to code the video by exhaustively trying all of the nine mode combinations. The best mode is the one having the minimum rate-distortion (RD) cost. In order to compute RD cost for each mode, the same operation of forward and inverse transform/quantization and entropy coding is repetitively performed. All of these processing explains the high complexity of RD cost calculation. Therefore, computational complexity of encoder is increased drastically. Using nine prediction modes in intra 4x4 and 8x8 block unit for a 16x16 macroblock (MB) can reduce spatial redundancies, but it may needs a lot of overhead bits to represent the prediction mode of each 4x4 and 8x8 block. Fast intra mode decision algorithms were proposed to reduce the number of modes that needed calculation according to some criteria [123-125]. An intra mode bits skip (IBS) method based on adaptive single-multiple prediction is proposed in order to reduce not only the overhead mode bits but also computational cost of the encoder [126]. If the neighboring pixels of upper and left blocks are similar, only DC prediction is used and it does not need prediction mode bits or else nine prediction modes are computed. But the IBS method suffers with some drawbacks a) the reference pixels in up-right block are not considered for similarity measure. If variance of reference pixels of upper and left blocks is very low, diagonal-down-left and vertical-left-modes are not similar to all other modes. But IBS considered all modes produce similar values. In this case, only DC prediction mode is not enough to maintain good PSNR and compression ratio. b) In IBS, each block is divided into two categories, either DC modes or all 9 modes. That's why; the

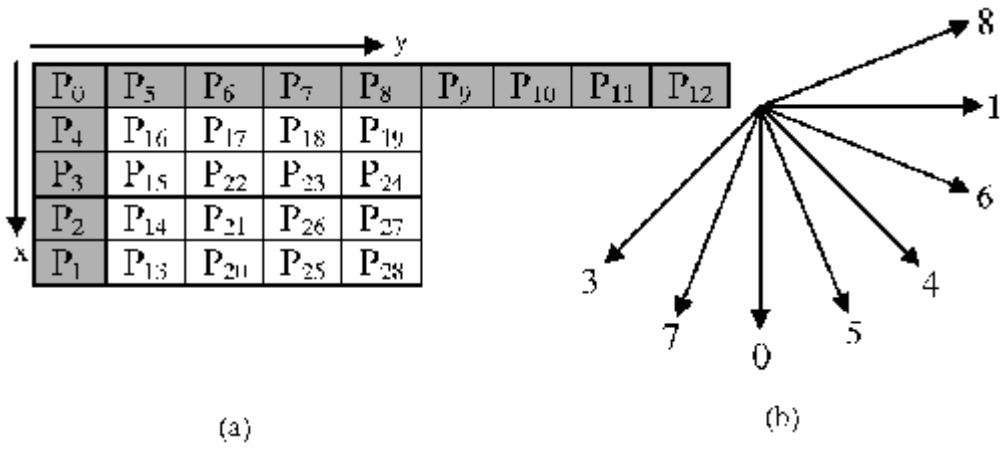
performance improvement is not significant for very complex sequences such as *Stefan* because only small amount of blocks are predicted by DC mode for these types of sequences. c) also computational expensive square operations are used in variance and threshold calculation which is hardware inconvenient in both encoder and decoder side. In order to reduce the intra mode bits, methods for estimating the most probable mode (MPM) are presented in [127, 128]. But the performance improvements are not significant.

In this chapter, we propose an improved DC prediction method to reduce the number of residual bits. An efficient method to reduce the number of intra mode bits and computational complexity of encoder is also proposed.

6.2 Review of H.264/AVC Intra Prediction

Intra prediction in H.264/AVC is conducted in spatial domain, by referring to neighboring samples of previously coded blocks which are to the left and/or above the block to be predicted. For the luma samples, intra prediction may be formed for each 4x4 block or for each 8x8 block or for a 16x16 macroblock. There are a total of 9 optional prediction modes for each 4x4 and 8x8 luma block; 4 optional modes for a 16x16 luma block. The prediction of a 4x4 block is computed based on the reconstructed samples labeled P_0-P_{12} as shown in Fig. 6.1 (a). The grey pixels (P_0-P_{12}) are reconstructed previously and considered as reference pixels of the current block. For correctness, 13 reference pixels of a 4x4 block are denoted by P_0 to P_{12} and pixels to be predicted are denoted by P_{13} to P_{28}. Mode 2 is called DC prediction in which all pixels (labeled P_{13} to P_{28}) are predicted by $(P_1+P_2+P_3+P_4+P_5+P_6+P_7+P_8+4)/8$ and mode 0

specifies the vertical prediction mode in which pixels (labeled P_{13}, P_{14}, P_{15} and P_{16}) are predicted from P_5, and the pixels (labeled P_{17}, P_{22}, P_{21}, and P_{20}) are predicted from P_6, and so on. The remaining modes are defined similarly according to the different directions as shown in Fig. 6.1 (b).



y

P_0	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}	P_{22}	P_{23}	P_{24}
P_8	a1	b1	c1	d1	e1	f1	g1	h1								
P_7	a2	b2	c2	d2	e2	f2	g2	h2								
P_6	a3	b3	c3	d3	e3	f3	g3	h3								
P_5	a4	b4	c4	d4	e4	f4	g4	h4								
P_4	a5	b5	c5	d5	e5	f5	g5	h5								
P_3	a6	b6	c6	d6	e6	f6	g6	h6								
P_2	a7	b7	c7	d7	e7	f7	g7	h7								
P_1	a8	b8	c8	d8	e8	f8	g8	h8								

x

(c)

Fig. 6.1 (a) Prediction samples of a 4x4 block (b) direction of prediction modes of 4x4 and 8x8 blocks (c) prediction samples of a 8x8 block

The reconstructed reference pixels and pixels to be predicted of a 8x8 block are shown in Fig. 6.1 (c). The directional pattern of a 8x8 block is exactly same as that of a 4x4 block which is shown in Fig. 6.1 (b). The best mode is the one that has the minimum RD cost and this cost is expressed as,

$$J_{RD} = SSD + l \cdot R \tag{6.1}$$

where, SSD is the sum of squared difference between the original block and the reconstructed block . R is the true bits needed to encode the block and λ is an exponential function of the quantization parameter (QP). After the best mode is acquired, it will be encoded into the compressed bit stream. The choice of intra prediction mode for each block must be signaled to the decoder and this could potentially require a large number of bits especially for 4x4 and 8x8 blocks due to the large number of modes. Hence the best mode is not directly encoded into the compressed bit stream. Intra modes for neighboring blocks are highly correlated and for example if a previously-encoded block was predicted using mode 2, it is likely that the best mode for current block is also mode 2. To take advantage of this correlation, predictive coding is used to signal 4x4 and 8x8 intra modes.

For current 4x4 or 8x8 block, a mode is predicted based on the modes of upper and left blocks and this mode is defined as the most probable mode (MPM). In the standard of H.264/AVC, the MPM is inferred according to the following rules; if the left neighboring block or the up neighboring block is unavailable, the MPM is set to 2(DC) or else the MPM is set to the minimum of the prediction mode of left neighboring block and the up neighboring block. For intra prediction according to each prediction mode, the encoder uses the condition of the MPM with a flag to signal the prediction mode. If the MPM is the same as the prediction mode, the flag is set to “1” and only one bit is needed to signal the prediction mode. When the MPM and prediction mode is different, the flag is set to “0” and additional 3 bits are required to signal the intra prediction mode. Encoder has to spend either 1or 4 bits to represent the intra mode.

6.3 Proposed Improved DC Prediction for 4x4 block

In H.264/AVC, DC mode is used to predict regions with no unified direction and the predicted values of all pixels are same. But the correlation that exists between predicted pixels and reference pixels are not absolutely considered to predict the DC prediction mode. Thus, the prediction signal generated by DC prediction is not well matched to the original signal, and a large number of bits is required for encoding the difference between the predicted and original signal. It is well known that Gaussian-like distribution can approximate local intensity variations in smooth image region. Original DC prediction always considers the reference pixels are belong to upper and left reconstructed blocks. But the correlation between neighboring pixels would be attenuated while the distance is increased and negligible when pixels are far apart. Therefore, prediction accuracy is degraded if all of the pixels of a 4x4 block are predicted from upper and left blocks.

In order to improve the performance of DC prediction of a 4x4 block, this section proposes an improved DC prediction (IDCP) method based on distance between reference and predicted pixels. Let us consider a 4x4 block as shown in Fig. 6.1 (a). In original DC prediction of H.264/AVC all of the pixels P_{13} to P_{28} of the 4x4 block are predicted from the neighboring reconstructed pixels P_1 to P_8. In the proposed IDCP method, pixels P_{13} to P_{19} are predicted from reconstructed pixels P_1 to P_8, P_{20} to P_{24} are predicted from previously predicted pixels P_{13} to P_{19}, P_{25} to P_{27} are calculated from pixels P_{20} to P_{24} and the reference pixels of P_{28} are P_{25} to $P_{27.}$ The predicted pixels P_r are estimated by following equation.

$$P_r = \left\lfloor \left(\sum_{i=m}^{n} W_i P_i \right) / \left(\sum_{i=m}^{n} W_i \right) \right\rfloor, \text{ where } r = 13 \text{ to } 28. \quad (6.2)$$

Here P_r is the pixel to be predicted and P_i is the reference pixel and $\lfloor \ \rfloor$ is the truncation operation to generate an integer value. Table 6.1 show the value of m and n for different predicted pixels r and W_i is the weight of i-th reference pixel. If the distance between predicted pixels P_r and reference pixels P_i is lower, the correlation between P_r and P_i is higher, and W_i is also higher. Therefore, weighting factor W_i is inversely proportional to the distance and defined as follows.

$$W_i = \lfloor 8 / D \rfloor \quad (6.3)$$

where D is the distance between reference pixel and pixel to be predicted and defined as $D = |B_{r,x} - B_{i,x}| + |B_{r,y} - B_{i,y}|$. $B_{r,x}$ and $B_{r,y}$ are the x and y positions of the predicted pixel P_r and similarly $B_{i,x}$ and $B_{i,y}$ are the x and y positions of the reference pixel P_i. In our method, the predicted value of the neighboring pixels obtained in the previous step is used to predict current pixels. Since some of the predicted values are based on other predicted values, the order of the calculation of predicted values are P_{13} to P_{28} as shown in four steps in Table 6.1.

Table 6.1 Value of m and n of (2) with different predicted pixels

Steps	Predicted pixels r	Reference pixels i	m	n
Step 1	r= 13 to 19	i= 1 to 8	1	8
Step 2	r= 20 to 24	i= 13 to 19	13	19
Step 3	r = 25 to 27	i= 20 to 24	20	24
Step 4	r= 28	i= 25 to 27	25	27

Let us assume, we would like predict pixel P_{26} as shown in Fig. 6.1 (a). According to third row of Table 6.1, in order to predict P_{26}, the reference pixels are P_{20}, P_{21}, P_{22}, P_{23}, and P_{24}. By substituting the value of r, m and n in (6.2) we get

$$P_{26} = \left\lfloor \frac{W_{20}P_{20} + W_{21}P_{21} + W_{22}P_{22} + W_{23}P_{23} + W_{24}P_{24}}{W_{20} + W_{21} + W_{22} + W_{23} + W_{24}} \right\rfloor \tag{6.4}$$

Consider that the position of predicted pixel P_{26} is $(B_{26,x}, B_{26,y})$ and therefore, as shown in Fig. 6.1(a) the position of reference pixels $(B_{i,x}, B_{i,y})$, i=20 to 24, are $(B_{26,x}+1, B_{26,y}-1)$, $(B_{26,x}, B_{26,y}-1)$, $(B_{26,x}-1, B_{26,y}-1)$, $(B_{26,x}-1, B_{26,y})$ and $(B_{26,x}-1, B_{26,y}+1)$, respectively . By substituting these values in (6.3), we can calculate the weighting factors W_{20} to W_{24}. It is shown that in order to calculate (6.3), computationally expensive division operation is necessary. In order to lessen computations, W_i is simplified as follows.

$$W_i = 2^{K_i} \quad \text{where} \quad K_i = \begin{cases} 3 & \text{if } D = 1 \\ 2 & \text{if } D = 2 \\ 1 & \text{if } D = 3 \text{ or } 4 \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

By substituting (6.5) into (6.2) and replacing multiplication operation by shifting,

$$P_r = \left\lfloor \frac{(\sum_{i=m}^{n} P_i << K_i)}{\sum_{i=m}^{n} W_i} \right\rfloor \quad \text{, where } r = 13 \text{ to } 28 \tag{6.6}$$

The division and truncation operation of (6.6) is inconvenient for hardware implementation. We have seen that for a particular value of r, the denominator of (6.6) is constant throughout the encoding and decoding process and independent on the intensity of the pixels. In order to avoid computational expensive division operator, (6.6) is approximated as follows:

$$P_r = [(\sum_{i=m}^{n} P_i << K_i) + C_r \times P_{r,left}] >> 5 \quad (6.7)$$

where $C_r = 2^5 - \sum_{i=m}^{n} W_i$ and value of C_r is given in Table 6.2. $P_{r,left}$ is the immediate left pixel of P_r. For example, $P_{r,left} = P_{15}$ for r=22. Obviously, (6.7) is faster than (6.2) and hardware inconvenient division and truncation operations are avoided. According to (6.7), each 4x4 block needs 125 shifting, 109 addition and 16 multiplication operator for improved DC prediction.

Table 6.2 Value of C_r with different values of r

r	C_r
13, 19, 20, 24	11
14, 18	8
15, 17	5
16, 22	0
21, 23	6
25, 27, 28	12
26	4

6.4 Proposed Intra Mode Bit Reduction (IMBR) for 4x4 and 8x8 block

6.4.1 Adaptive numbers of modes (ANM)

Although H.264/AVC intra coding method provides good compression ratio, owing to the use of nine prediction modes of 4x4 and 8x8 luma blocks, its computational complexity increases drastically. Using nine prediction modes in intra 4x4 and 8x8 block unit for a 16x16 MB can reduce the spatial redundancies, but it may needs a lot of overhead bits to represent the prediction mode of each 4x4 or 8x8 block. Based on the

variation of neighboring pixels, the proposed method classifies a block as one of three different cases.

N	N	N	N	N	N	N	N	N
N	P_{16}	P_{17}	P_{18}	P_{19}				
N	P_{15}	P_{22}	P_{23}	P_{24}				
N	P_{14}	P_{21}	P_{26}	P_{27}				
N	P_{13}	P_{20}	P_{25}	P_{28}				

Fig. 6.2 Case 1: All of the reference pixels have similar value

6.4.1.1 Case 1

As shown in Fig. 6.2, if all reference pixels are same, the predicted values of nine directional predictions are same. In this case, it does not need to calculate the entire prediction modes. In this case, only DC mode can be used for encoder and decoder prediction and the prediction mode bit can be skipped. In order to classify the 4x4 block in this category, variance s_1 and threshold T_1 are calculated. If variance s_1 of all of the neighboring pixels is less than the threshold T_1, only DC prediction mode is used and it does not need the prediction mode bits. The variance s_1 and mean m_1 are defined as,

$$s_1 = \sum_{i=1}^{12} |P_i - m_1| \quad \text{and} \quad m_1 = \left\lfloor (\sum_{i=1}^{12} P_i)/12 \right\rfloor \tag{6.8}$$

where P_i is the i-th pixel of Fig. 6.1(a) and m_1 is the mean value of block boundary pixels. In order to avoid computational expensive division and truncation operations, m_1 is replaced by weighted mean value (m_1') as

$$m_1' = (\sum_{i=1}^{4} P_i + (\sum_{i=5}^{8} P_i) << 1 + \sum_{i=9}^{12} P_i) >> 4 \quad \text{and} \quad s_1 = \sum_{i=1}^{12} |P_i - m_1'| \tag{6.9}$$

Here << and >> are the shift left and shift right operator, respectively. In order to set the threshold T_1, we have done several experiments for four different types of video sequences (*Mother & Daughter*, *Foreman*, *Bus* and *Stefan*) with CIF format at different QP values. *Mother & Daughter* represents simple and low motion video sequence. *Foreman* and *Bus* contain medium detail and represent medium motion video sequences. *Stefan* represents high detail and complex motion video sequence. By changing the threshold, we observed the RD performance and found that threshold T_1 is independent on the type of video sequence but depends on the QP values. Fig. 6.3 shows the variation of selected threshold T_1 with QP values. The original threshold curve is generated by averaging the threshold values of all four sequences for each QP. By using the polynomial fitting technique, the generalized threshold value T_1 is approximated as follows:

$$T_1 = \begin{cases} QP + 12 & \text{if } QP \le 24 \\ 5QP - 90 & \text{Otherwise} \end{cases} \tag{6.10}$$

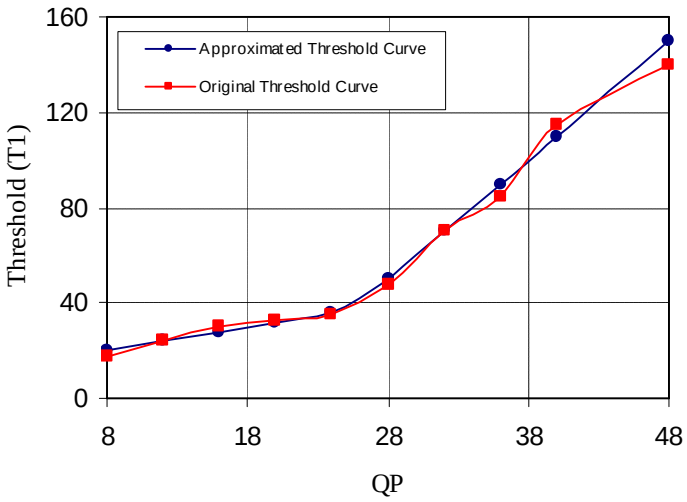


Fig. 6.3 Variation of threshold T_1 with QP

P_0	N	N	N	N	N	N	N	N
P_4	P_{16}	P_{17}	P_{18}	P_{19}				
P_3	P_{15}	P_{22}	P_{23}	P_{24}				
P_2	P_{14}	P_{21}	P_{26}	P_{27}				
P_1	P_{13}	P_{20}	P_{25}	P_{28}				

Fig. 6.4 Case 2: The reference pixels of up and up-right block have similar value

6.4.1.2 Case 2

As shown in Fig. 6.4, if all of the reference pixels of up and up-right blocks are same, vertical, diagonal-down-left, vertical-left, vertical-right and horizontal-down modes produce the same prediction values. That's why, in the proposed method we have chosen only one mode from this set and it is the vertical prediction mode.

If variance s_2 of the neighboring pixels of up and up-right blocks is less than the threshold T_2, four prediction modes (vertical, horizontal, diagonal-down-right and horizontal-up) are used and one of them is selected through RDO process. Instead of using 3 bits of original encoder, each of the four prediction mode is represented by 2 bits. In this case, the binary representations of prediction modes are recorded as shown in Table 6.3 and hence a significant amount of mode bits are saved. Threshold T_2 is selected in the same way as T_1. T_2 also depends on the QP and better results were obtained at

$$T_2 = \lfloor (2T_1 / 3) \rfloor \approx (T_1 >> 1 + T_1 >> 3) \tag{6.11}$$

The variance s_2 and mean m_2 are defined as,

$$s_2 = \sum_{i=5}^{12} |P_i - m_2| \quad \text{and} \quad m_2 = (\sum_{i=5}^{12} P_i) >> 3 \tag{6.12}$$

where m_2 is the mean value of block boundary pixels of top and top-right blocks.

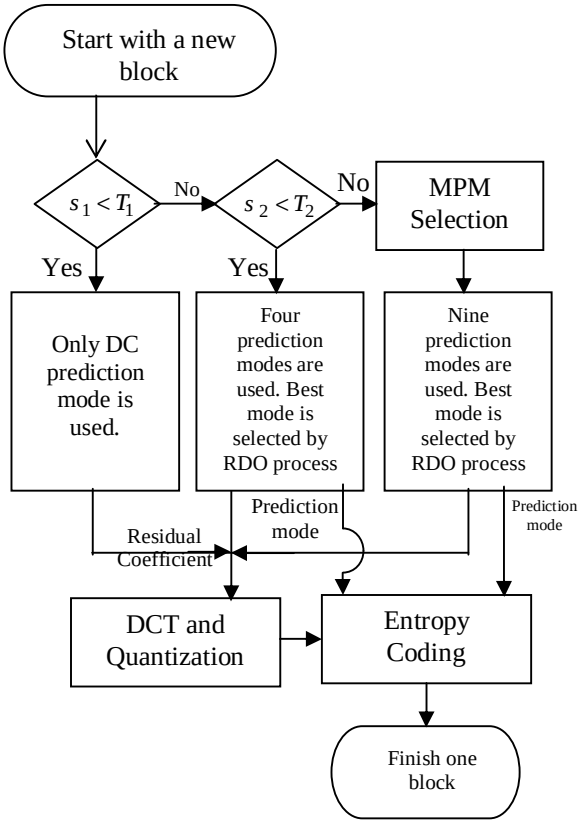


Fig. 6.5 Flow diagram of proposed intra mode bits reduction (IMBR) method

Table 6.3 Binary representation of modes of case 2

Mode	Binary representation
Vertical	00
Horizontal	01
Diagonal-down-right	10
Horizontal-up	11

The flow diagram of the proposed IMBR method is presented in Fig. 6.5. The variance s_1 and threshold T_1 are calculated at the start of the mode decision process and if the variance is less than the threshold ($s_1 < T_1$) only DC prediction mode is used. In this case computational expensive RDO process is skipped and a lot of computations are

saved. In addition, no bit is necessary to represent intra prediction mode because only one mode is used. In the decoder side, if $s_1 < T_1$, decoder understands that DC prediction mode is the best prediction mode. On the other hand, if $s_1 < T_1$ is not satisfied, encoder calculates the variance s_2 and threshold T_2. If $s_2 < T_2$, vertical, horizontal, diagonal-down-right and horizontal-up modes are used as candidate modes in RDO process. A substantial saving in computations is achieved using 4 prediction modes instead of 9 modes of the original RDO process. The best mode is the mode which has the smallest rate-distortion cost. In order to represent the best mode, 2 bits are sent to the decoder and Table 6.3 shows the four prediction modes with corresponding binary representations. As shown in Table 6.3, if the diagonal-down-right mode is selected as the best mode, the encoder sends "10" to the decoder. In this category, only 2 bits are used to represent the intra prediction mode whereas 3 bits are used in the original encoder. Consequently a large number of intra prediction mode bits are saved. If $s_2 < T_2$ is not satisfied, nine prediction modes are used as the candidate mode and one of them is selected through the RDO process, as in H.264/AVC.

In this case, based on the MPM either 1 or 4 bits are allocated to represent the intra prediction mode. The new prediction mode numbers are recorded and compared against H.264/AVC in Table 6.4. Since diagonal-down-left, vertical-right, horizontal-down and vertical-left predictions modes are not utilized in the previous cases, the probability of these modes are high in this case and thus these modes are defined as small numbers. Similarly mode numbers for other modes are higher value.

Table 6.4 Prediction modes recording of the proposed method

Mode	Mode number H.264/AVC	Mode number Proposed
Diagonal-down-left	3	0
Vertical-right	5	1
Horizontal-down	6	2
Vertical-left	7	3
Vertical	0	4
Horizontal	1	5
DC	2	6
Diagonal-down-right	4	7
Horizontal-up	8	8

Since 8x8 intra prediction also uses 9 prediction modes, the proposed IMBR method is also applied to 8x8 intra prediction mode. Assume P_i is the i-th reconstructed pixel of Fig. 6.1 (c). Variances and thresholds of a 8x8 block are defied as

$$m'_{1_{8x8}} = (\sum_{i=1}^{8} P_i + (\sum_{i=9}^{16} P_i) << 1 + \sum_{i=17}^{24} P_i) >> 5 \quad \text{and} \quad s_{1_{8x8}} = \sum_{i=1}^{24} \left| P_i - m_{1_{8x8}}{}' \right| \tag{6.13}$$

$$m_{2_{8x8}} = (\sum_{i=9}^{24} P_i) >> 4 \text{ and } s_{2_{8x8}} = \sum_{i=9}^{24} \left| P_i - m_{2_{8x8}} \right| \tag{6.14}$$

$$T_{1_{8x8}} = \begin{cases} 2QP + 24 & \text{if } QP \le 24 \\ 10QP - 180 & \text{Otherwise} \end{cases} \quad \text{and } T_{2_{8x8}} = (T_{1_{8x8}} >> 1 + T_{1_{8x8}} >> 3) \tag{6.15}$$

Table 6.5 Percentage of different MBs

Sequences	4x4 block %	8x8 block %	16x16 block %
Akiyo	38.82	50.70	10.49
Foreman	64.92	31.73	3.35
Stefan	83.16	13.55	3.29
Container	56.11	19.41	24.48
Claire	36.20	18.01	45.79
Mobile	99.89	0.11	0
Coast Guard	38.26	60.01	1.73
Grand Mother	41.05	36.98	21.97
Average	57.30	28.81	13.89

Table 6.6 (a) Percentage of different categories of 4x4 blocks (only 4x4 mode is enabled)

Sequences	Case 1 %	Case 2 %	Case 3 %
Akiyo	52.39	17.45	30.14
Foreman	38.50	21.35	40.15
Stefan	28.96	18.34	52.70
Container	56.51	14.46	29.03
Claire	67.19	7.83	24.98
Mobile	16.28	16.94	66.78
Coast Guard	44.49	26.14	29.37
Grand Mother	54.27	17.29	28.43
Average	44.82	17.47	37.69

Table 6.6 (b) Percentage of different categories of 8x8 blocks (only 8x8 mode is enabled)

Sequences	Case 1 %	Case 2 %	Case 3 %
Akiyo	44.96	19.94	35.10
Foreman	31.66	19.72	48.62
Stefan	29.02	17.52	53.45
Container	50.71	14.17	35.13
Claire	56.60	5.52	37.88
Mobile	16.22	16.78	66.99
Coast Guard	54.51	22.06	23.44
Grand Mother	52.33	18.15	29.52
Average	42.00	16.73	41.27

The percentage of MBs choosing different intra modes are tabulated in Table 6.5. In this simulation, all of the frames are encoded by intra coding. It is shown only 13.89% MBs are encoded with 16x16 intra modes. In case of complex sequences (such as ***Stefan, Coastguard, Mobile***), this percentage is negligible. In addition to this, the number of bits to represent 16x16 intra mode is not so high. Therefore, if we apply proposed technique in 16x16 mode the performance improvements will not be significant but the computation of the decoder side will be increased. That's why the proposed methods are

not implemented in 16x16 mode. Table 6.6 shows the percentage of different categories of 4x4 and 8x8 block. In Table 6.6 (a), only 4x4 mode is enabled and in Table 6.6(b) only 8x8 mode is enabled. Experiments have shown that about 44% of the 4x4 blocks belong to case 1 and calculate only one mode and about 17.47% 4x4 blocks used four modes and spend 2 bits to represent intra coding mode. In addition to it about 37% of the 4x4 blocks and 41% of 8x8 blocks still calculate 9 prediction modes and spend either 1 or 4 bits to represent the prediction mode number. If the MPM is the best mode only 1 bit is used; otherwise 4 bits are required. Therefore, if we can develop a more accurate method to estimate the MPM of 4x4 and 8x8 block, a significant percentage of blocks will require only 1 bit for mode information.

The proposed adaptive number of mode selection method reduces the computation of the encoder significantly but increase the computation of the decoder. In order to compute s_1 of a 4x4 block, decoder needs extra 34 additions, 12 absolute operations and 2 shifting operations. Some of the blocks also need to compute s_2, which need more 22 additions, 8 absolute operations and 1 shifting operation. Similarly for an 8x8 block, $s_{1_{8x8}}$ bears extra 70 additions, 24 absolute operations and 2 shifting operations. In the same fashion, 46 additions, 16 absolute and 1 shifting operations are required to compute $s_{2_{8x8}}$. Since there is no division and floating point operations, the proposed method is convenient for hardware implementation. Simulation results confirm that the complexity increment of decoder is very low.

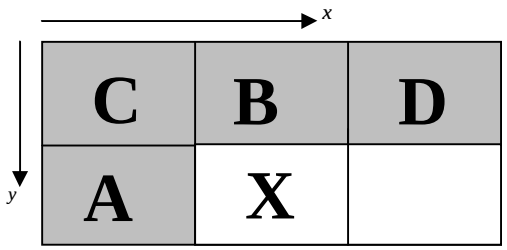


Fig. 6.6 Current and neighboring blocks

Table 6.7 Mode directions (q_m)

Mode	Direction
Vertical	$p/2$
Horizontal	0
Diagonal-down-left	$3p/4$
Diagonal-down-right	$p/4$
Vertical- right	$3p/8$
Horizontal-down	$p/8$
Vertical-left	$5p/8$
Horizontal-up	$-p/8$

6.4.2 Selection of Most Probable Mode (MPM)

Natural video sequences contain a lot of edges and these edges are usually continuous thus indicating that the prediction direction of neighboring blocks and that of current block is also continuous. Let us consider that X is the current block as shown in Fig. 6.6 and four neighboring blocks are denoted as A, B, C and D. So if the upper block (B) is encoded with vertical mode, the mode of current block is more likely to be vertical mode. Similarly, if mode of up-left block (C) is diagonal-down-left mode (mode 4 in Fig. 6.1(b)), then the mode of the current block is more likely to be diagonal-down-left mode. If the direction from the neighboring block to the current block is identical to the prediction mode direction of the neighboring block, there is a high possibility that the best prediction mode of the current block is also identical to the prediction mode

direction. Based on this idea, the weight of the proposed MPM method is proportional to the absolute difference between block direction and mode directions. The mode direction (q_m) is calculated based on the direction of Fig. 6.1 (b) and tabulated in Table 6.7. The block direction (q_B) and block distance (D_B) are calculated by the following set of equations.

$$q_B = \tan^{-1}\frac{y_c - y_n}{x_c - x_n} \tag{6.16}$$

$$D_B = |y_c - y_n| + |x_c - x_n| \tag{6.17}$$

Where, (x_c, y_c) and (x_n, y_n) are the position of current and neighboring block, respectively. The mode of the neighboring block is denoted as M_n . Weight is also dependent on the distance between the current block and neighboring block. If the distance between the current and neighboring block is higher, the correlation between the blocks is lower and weight is also low. Based on these observations weight of neighboring mode M_n is calculated as

$$W(M_n) = \min[0, \frac{a}{D_B} - b\,|q_B - q_m|] \tag{6.18}$$

where a and b are the proportionally constant and min(P,Q) means minimum value between P and Q. Based on simulation, a and b are selected as 6 and $\frac{8}{p}$.

Instead of using two neighboring blocks A and B in the original H.264/AVC encoder, the proposed method utilizes the prediction mode used in the four neighboring blocks (A, B, C and D). The weight of the prediction mode of each neighboring block is calculated and updated by adding the weight of same mode. Since, DC has no unified

direction, if the neighboring mode is DC, the weight corresponding to this block is set to 0. The weight of each prediction mode is counted up and find out the mode with highest weight $W_{\max}$. If the maximum weight $W_{\max}$ is very low, it seems that there is no continuation of edges. In this case, possibility of DC prediction mode to be the best mode is higher. If maximum weight $W_{\max}$ is less than a threshold T_{MPM}, the MPM is the DC mode; otherwise the MPM is the mode with maximum weight $W_{\max}$. Following is the step by step algorithm of the proposed method.

Step 1: Initialize weight (W) of each mode to zero.

Step 2:

For each of the four neighboring blocks (A, B, C and D),

If neighboring mode $M_n = DC$, $W(M_n)\ +=\ 0$.

Otherwise

a) Calculate block direction, q_B and D_B

b) Find mode direction of the neighboring mode M_n from Table 6.7.

c) Calculate weight of neighboring mode:

$$W(M_n)\ +=\ \min[0,\ \frac{a}{D_B} - b\,|q_B - q_m|]$$

End of block

Step 3: Find the maximum weight $W_{\max}$ and the mode that has maximum weight.

Step 4: If maximum weight $W_{\max}$ is less than T_{MPM}, the most probable mode is the DC mode; otherwise MPM is the mode with maximum weight $W_{\max}$.

In order to find the threshold T_{MPM}, we have done some simulations. Four different types of video sequences (*Mother & Daughter, Foreman, Bus* and *Stefan*) were encoded by changing the value of T_{MPM} from 1 to 10 and RD performances were observed. Better results were found at $T_{MPM} = 5$. In order to reduce the computation of (6.18), a / D_B and q_B of neighboring blocks A, B, C and D are pre-calculated and stored in a Table. For example, a / D_B is 6 for block A and B, and equal to 3 for block C and D. q_B is equal to 0, $p/2$, $p/4$, and $-p/4$ for block A, B, C and D, respectively.

It is clear that proposed MPM selection method introduce some overhead computations both of the encoder and decoder side. Experimental results confirm that the overhead is very negligible. In addition to that the encoder does not need to calculate the MPM for all blocks. The MPM is calculated only for the blocks which are classified as case 3 in Table 6.6 and the percentage of this type of blocks is around 40% on average.

6.5 Simulation Results

The performance of the proposed method is evaluated by using JM 12.4 [129] reference software in simulation. Different types of video sequences with different resolutions are used as test materials. A group of experiments were carried out on the test sequences with different quantization parameters (QPs). All simulations are conducted under Windows Vista operating system, with Pentium 4 2.2 G CPU and 1 G RAM. Simulation conditions are (a) QPs are 28, 36, 40, 44 (b) entropy coding: CABAC (c) RDO on (d) frame rate: 30 fps and (e) number of frames: 100. The comparison results are produced

and tabulated based on the average difference in the total encoding ($\Delta T_1\%$) and decoding ($\Delta T_2\%$) time, the average PSNR differences ($\Delta PSNR$), and the average bit rate difference ($\Delta R\%$). PSNR and bit rate differences are calculated according to the numerical averages between RD curves derived from original and proposed algorithm, respectively [67]. The encoding ($\Delta T_1\%$) and decoding ($\Delta T_2\%$) complexity is measured as follows

$$\Delta T_1\% = \frac{T_{p_{enc}} - T_{o_{enc}}}{T_{o_{enc}}} \times 100\% \tag{6.19}$$

$$\Delta T_2\% = \frac{T_{p_{dec}} - T_{o_{dec}}}{T_{o_{dec}}} \times 100\% \tag{6.20}$$

where, $T_{o_{enc}}$ and $T_{o_{dec}}$ are the total encoding and decoding time of the JM 12.4, respectively. $T_{p_{enc}}$ and $T_{p_{dec}}$ are the total encoding and decoding time of the proposed method, respectively.

6.5.1 Experiments with 4x4 intra modes only

In this experiment all frames are intra coded only 4x4 mode is enabled. Each video sequence is encoded with seven different methods (DWP [116], MPM [128], IBS [126], Proposed IDCP only, proposed IMBR: ANM only, proposed IMBR: MPM only and proposed IMBR+IDCP) and the performance comparisons are presented in Table 6.8. In these tables, a positive value indicates increment and a negative value represents decrement.

Table 6.8(a) PSNR performances of proposed methods (only 4x4 modes, All I frames)

Sequences	DWP [116]	MPM [128]	IBS [126]	Proposed IDCP only	Prop IMBR: ANM only	Prop IMBR: MPM selection only	IMBR + IDCP
	Δ PSNR	Δ PSNR	Δ PSNR	Δ PSNR	Δ PSNR	Δ PSNR	Δ PSNR
Grand Mother (QCIF)	0.04	0.01	0.37	0.11	0.41	0.09	0.50
Sales man (QCIF)	0.02	0.01	0.32	0.12	0.39	0.03	0.47
Stefan (QCIF)	0.01	0.02	0.10	0.09	0.19	0.03	0.23
Container (QCIF)	0.04	0.01	0.09	0.11	0.15	0.09	0.20
Carphone (QCIF)	0.04	0.14	0.66	0.07	0.79	0.18	0.86
Silent (CIF)	0.02	0.05	0.35	0.07	0.40	0.12	0.48
Bus (CIF)	0.003	0.05	0.11	0.13	0.12	0.07	0.20
Hall (CIF)	0.02	0.08	0.32	0.10	0.37	0.12	0.47
Mobile Calendar (HD-1280x720)	0.03	0.02	0.19	0.06	0.25	0.07	0.31
Average	0.02	0.04	0.28	0.10	0.341	0.09	0.413

Table 6.8(b) Bit rate performances of proposed methods (only 4x4 modes, All I frames)

Sequences	DWP [116]	MPM [128]	IBS [126]	Proposed IDCP only	Prop IMBR: ANM only	Prop IMBR: MPM selection only	IMBR + IDCP
	ΔR %	ΔR %	ΔR %	ΔR %	ΔR %	ΔR %	ΔR %
Grand Mother (QCIF)	-1.43	-0.92	-15.46	-4.64	-16.44	-3.61	-20.21
Sales man (QCIF)	-0.21	-0.46	-12.99	-6.64	-13.57	-1.70	-19.01
Stefan (QCIF)	-0.28	-0.65	-2.76	-1.37	-5.87	-1.56	-7.03
Container (QCIF)	-1.73	-0.40	-3.12	-3.48	-5.35	-2.54	-6.98
Carphone (QCIF)	-1.06	-3.69	-18.40	-2.14	-22.38	-4.77	-23.98
Silent (CIF)	-1.07	-2.53	-15.40	-2.46	-17.35	-5.23	-19.88
Bus (CIF)	-0.17	-2.02	-3.83	-4.03	-4.58	-1.90	-6.03
Hall (CIF)	-0.53	-3.43	-8.69	-2.69	-9.87	-3.28	-12.50
Mobile Calendar (HD-1280x720)	-2.46	-0.75	-6.84	-2.77	-9.31	-3.51	-10.93
Average	-0.99	-1.65	-9.70	-3.35	-11.64	-3.12	-14.06

Table 6.8(c) Encoder Complexity comparisons of proposed methods (only 4x4 modes, All I frames)

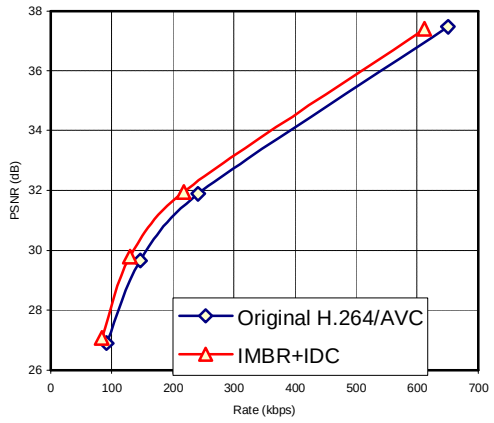
Sequences	DWP [116]	MPM [128]	IBS [126]	Proposed IDCP only	Prop IMBR: ANM only	Prop IMBR: MPM selection only	IMBR + IDCP
	ΔT_1 %	ΔT_1 %	ΔT_1 %	ΔT_1 %	ΔT_1 %	ΔT_1 %	ΔT_1 %
Grand Mother (QCIF)	2.22	0.76	-39.79	4.35	-52.14	0.91	-43.81
Sales man (QCIF)	1.17	1.19	-31.22	3.70	-37.99	1.30	-32.66
Stefan (QCIF)	1.01	1.29	-17.99	3.46	-25.47	1.64	-19.17
Container (QCIF)	2.23	1.28	-31.38	4.41	-39.55	1.75	-34.11
Carphone (QCIF)	1.29	2.20	-33.81	4.27	-46.01	3.21	-41.12
Silent (CIF)	1.34	1.09	-35.85	4.77	-45.87	1.32	-39.72
Bus (CIF)	1.82	1.09	-16.49	4.74	-32.04	1.18	-24.49
Hall (CIF)	1.35	0.97	-38.84	3.16	-48.5	1.42	-44.16
Mobile Calendar (HD-1280x720)	2.62	1.76	-27.69	4.39	-34.7	2.06	-31.61
Average	1.67	1.29	-30.34	4.13	-40.25	1.64	-34.54

Table 6.8(d) Decoder Complexity comparisons of proposed methods (only 4x4 modes, All I frames)

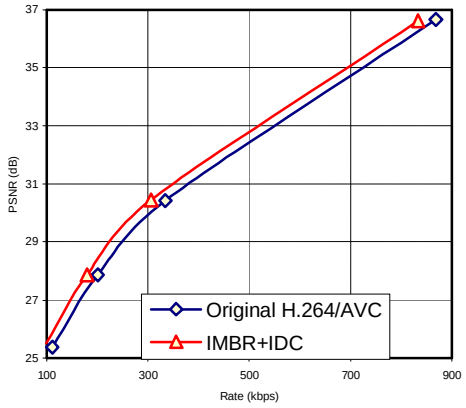
Sequences	DWP [116]	MPM [128]	IBS [126]	Proposed IDCP only	Prop IMBR: ANM only	Prop IMBR: MPM selection only	IMBR + IDCP
	ΔT_2 %	ΔT_2 %	ΔT_2 %	ΔT_2 %	ΔT_2 %	ΔT_2 %	ΔT_2 %
Grand Mother (QCIF)	2.25	0.91	2.09	4.39	1.99	1.15	5.17
Sales man (QCIF)	0.04	0.41	1.19	1.12	0.91	0.72	2.55
Stefan (QCIF)	0.39	1.09	0.39	0.54	0.33	1.19	1.76
Container (QCIF)	0.79	0.55	0.45	1.49	0.41	0.68	3.42
Carphone (QCIF)	1.18	0.82	0.42	2.65	0.39	0.96	4.27
Silent (CIF)	0.99	0.86	1.74	2.81	1.53	0.98	3.67
Bus (CIF)	1.29	1.15	0.99	2.58	0.82	1.27	4.73
Hall (CIF)	0.35	0.72	1.45	1.71	1.18	0.83	3.69
Mobile Calendar (HD-1280x720)	1.28	0.93	1.55	1.84	1.59	0.96	3.73
Average	0.95	0.82	1.14	2.12	1.01	0.97	3.67

As shown in Tables 6.8(a) and 6.8(b), the proposed IDCP improves 0.10 dB PSNR and reduces bit rate by 3.35% whereas DWP improves PSNR by only 0.02 dB and reduces bit rate by only 0.99% of the original encoder. In terms of computation, it is clear from Tables 6.8 (c) and 6.8 (d), the proposed IDCP increases the encoding and decoding time by 4.13% and 2.12%, respectively. In case of IBS method, the average PSNR improvement is about 0.28 dB and average bit rate reduction is about 9.7%. Whereas in our proposed IMBR: ANM only method, the average PSNR improvement is about 0.34 dB and average bit rate reduction is about 11.64%. The proposed IMBR: ANM only method also reduces the computation of the original encoder by 40%. Although this method introduces some extra computations of the decoder side, the simulation results of Table 8(d) confirm that, the computational overhead of the decoder is very low (about 1.01%). Although our proposed ANM method calculated two variances, the decoding time of proposed ANM is faster than that of IBS method. This is because our method does not need any computational expensive square operators. The performance of the proposed MPM selection is also tabulated in Table 8. It is clear form Table 6.8 (a-b) that the improvement of proposed MPM method is higher than that of MPM in [128] in terms of both video quality and bit rate. From Tables 8(c) and 8(d) we have seen that computation performance of both of the proposed MPM and MPM in [23] is almost similar. If we combine our IMBR and IDCP method together, about 14.06% bit rate reduction is achieved along with a 0.41 dB improvement in PSNR in the expense of 3.67% increment of decoding time. Out of all video sequences listed in Table 8, the best RD performance improvement was accomplished for *Car Phone* video sequence; bit rate reduction is about 24% (from Table 6.8 (b)) and PSNR improvement is 0.86

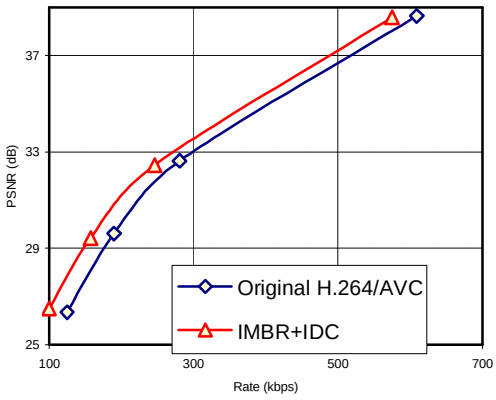
dB(from Table 6.8 (a)). This is understandable because most of the blocks of this sequence are classified as either case 1 or case 2. The PSNR improvement and bit rate reduction of worst case (*Bus*) is 0.20 dB and 6.03%, respectively.



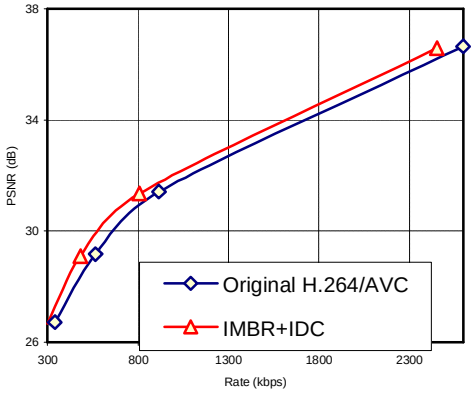
(a) Grand Mother (QCIF)



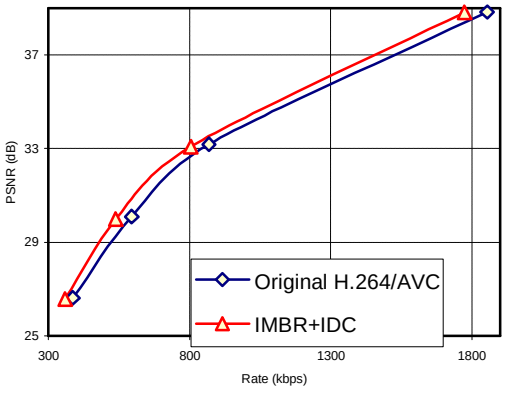
(b) Salesman (QCIF)



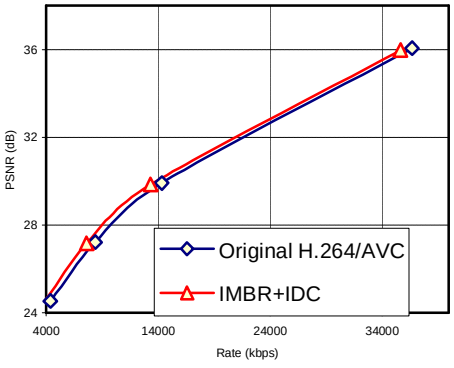
(c) Carphone (QCIF)



(d) Silent (CIF)



(e) Hall (CIF)



(f) Mobile Calendar (HD)

Fig. 6.7 RD curves of proposed method (only 4x4 mode, all I frames)

The rate-distortion (RD) curves of six different types of video sequences are plotted in Fig. 6.7. It is shown that RD curve of our proposed method is always superior to that of the original H.264/AVC encoder. The proposed method saves about 34.54% computations of original H.264/AVC encoder in the expanse of 3.67% of decoding computations increments. This small computation increment of decoder side is allowable because improvement of RD performance and encoding complexity is very significant.

Table 6.9 Experimental results of proposed methods (All I frames, all Intra modes)

Sequence	$\Delta PSNR$ in dB	ΔR %	ΔT_1 %	ΔT_2 %
Grand Mother (QCIF)	0.37	-15.30	-35.1	4.46
Sales man (QCIF)	0.43	-14.98	-27.7	2.03
Stefan (QCIF)	0.22	-6.27	-17.6	1.53
Container (QCIF)	0.16	-4.59	-29.5	1.99
Carphone (QCIF)	0.81	-19.81	-32.6	3.89
Silent (CIF)	0.39	-14.28	-35.2	3.51
Bus (CIF)	0.19	-5.78	-19.1	4.67
Hall (CIF)	0.32	-10.17	-33.7	1.94
Mobile Calendar (HD-1280x720)	0.23	-13.16	-28.8	2.09
Average	0.35	-11.59	-28.8	2.90

6.5.2 Experiments with all intra modes

In this experiment all 100 frames are encoded by intra coding and all intra modes (4x4, 8x8, and 16x16) are enabled. The results are tabulated in Table 6.9. Here proposed IDCP method is applied in 4x4 block and IMBR method is implemented in 4x4 and 8x8 blocks. Since only small amount of MBs are encoded with 16x16 modes (clearly shown in Table 6.5), the proposed methods are not implemented in 16x16 mode for computational difficulties. We have seen that the average gain is in the range of 0.35

dB PSNR and 11.59% bit rate saving, with a maximum for sequence *Carphone* with 0.81 dB and 19.81%. We have seen that the proposed method reduces 28.8% computation of original encoder. The computation increment of decoder side is very low and that is 2.90% on average.

6.6 Summary

In this chapter, we proposed two methods to improve the RD performance of H.264/AVC intra encoder. At first, a distance based improved DC prediction is utilized to better represent smooth region in sequences. Then an intra mode bit rate reduction scheme for representing the intra prediction mode is described. In the proposed IMBR method the number of prediction modes of a block is selected based on the variance in neighboring pixels. For most of the blocks, either 1 or 2 bits are required to encode the intra prediction mode information. An efficient technique for selecting the MPM is also developed based on the continuity of the edge direction. In comparison to original intra coder, the proposed method saves about 11.59% bits and improves the video quality by 0.35 dB, on an average. The proposed methods not only improve the RD performance but also reduce computational complexity of H.264/AVC intra encoder with the little expense of decoder computations.

Chapter 7

Enhanced SATD based cost function

To achieve the highest coding efficiency, H.264/AVC uses rate-distortion optimization technique. That means the encoder has to code the video by exhaustively trying all the mode combinations including the different intra- and inter-prediction modes. Therefore, the complexity and computation load of video coding in H.264/AVC increase drastically compared to any previous standards. In this chapter, we propose an enhanced low complexity cost function for H.264/AVC intra 4x4 mode selections. The enhanced cost function uses sum of absolute Hadamard-transformed differences (SATD) and mean absolute deviation of the residual block to estimate distortion part of the cost function. A threshold based large coefficients count is also used for estimating the bit-rate part. The proposed method improves the rate-distortion performance of the conventional fast cost functions while maintaining low complexity requirement. As the results, the encoding process can be significantly accelerated with use of the proposed cost function. Simulation results confirm that the proposed method achieves better coding performance as compared with the Sum of Absolute Differences (SAD) and SATD based cost functions recommended in H.264/AVC.

7.1 Cost functions of H.264/AVC intra prediction

H.264/AVC intra prediction uses 9 prediction modes for a 4x4 luma block. To take the full advantages of all modes, the H.264/AVC encoder can determine the mode that meets the best rate-distortion (RD) tradeoff using rate-distortion optimization (RDO) mode decision scheme. The best mode is the one having minimum RD cost and this cost is expressed as

$$J_{RD} = SSD + l \cdot R \tag{7.1}$$

where the SSD is the sum of squared difference between the original blocks **S** and the reconstructed block **C**, and it is expressed by

$$SSD = \sum_{i=1}^{4} \sum_{j=1}^{4} (S_{ij} - C_{ij})^2 \tag{7.2}$$

where S_{ij} and C_{ij} are the (i, j)th elements of the current original block **S** and the reconstructed block **C**. In (7.1), the R is the true bits needed to encode the block and λ is an exponential function of the quantization parameter (QP). In [41], a strong connection between the local Lagrangian multiplier and the QP was found experimentally as

$$l = 0.85 \times 2^{(QP-12)/3} \tag{7.3}$$

This was done by fixing a λ and for each macroblock selecting the QP that minimized J_{RD}.

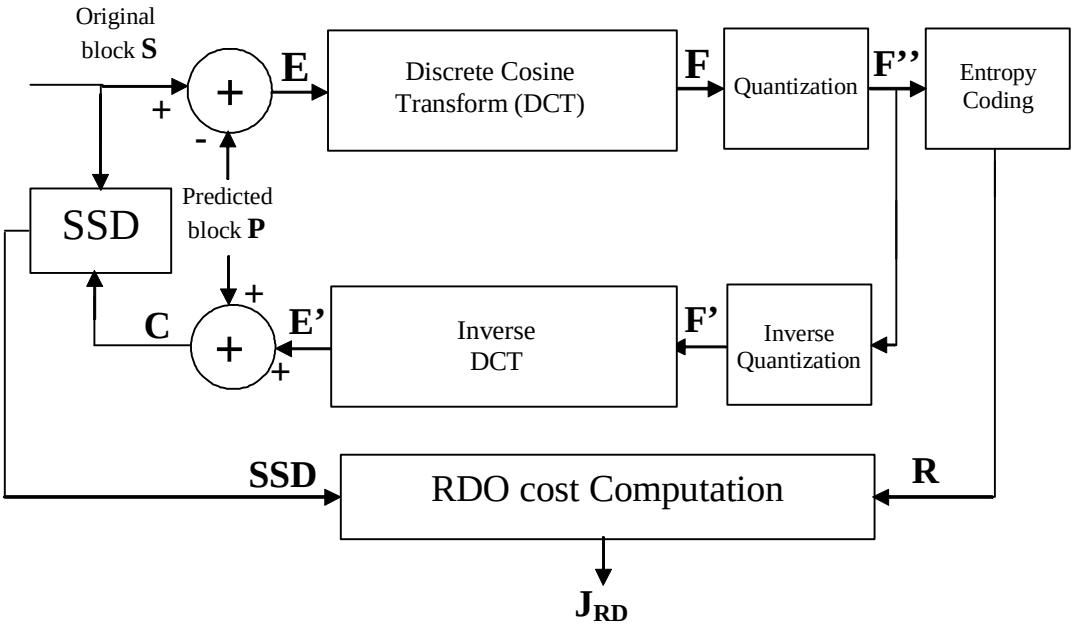


Fig. 7.1 Computation of Rate-distortion (RD) cost function

Fig. 7.1 shows the flow diagram of RD cost computation. It is found that the rate-distortion function J_{RD} introduces a lot of computation in real encoding as it requires the following computations:

1. Compute the predicted block: **P**
2. Compute the residual block: **E** = **S** – **P**
3. Discrete Cosine Transform (DCT) of the residual block: **F** = DCT(**E**)
4. Quantize the transformed residual block: **F**" = Q(**F**)
5. Entropy coding of the quantized and transformed residual block to determine the bit-rate for encoding the block: R = EC(**F**")
6. Inverse quantize the quantized and transformed residual block: **F**' = Q^{-1}(**F**")

7. Inverse Discrete Cosine Transform (DCT) of the de-quantized block: **E'** = DCT^{-1}(**F'**)
8. Compute the reconstructed image block: **C** = **E'** + **P**
9. Compute SSD between **S** and **C** by using 7.2
10. Calculate the cost function : $J_{RD} = SSD + \lambda \cdot R$

The H.264/AVC encoder computes this rate-distortion optimization process for every macroblock with all possible modes. All of these processing explains the high computational complexity of J_{RD} cost calculation. Hence, the cost function will make H.264/AVC impossible to be realized in real-time applications.

To accelerate the coding process, H.264/AVC provides a fast SAD-based cost function:

$$J_{SAD} = SAD + l_1 \cdot 4P \tag{7.4}$$

where SAD is sum of absolute difference between the original block **S** and the predicted block **P**. The λ_1 is also approximate exponential function of the QP which is almost the square of λ, and the P equal to 0 for the most probable mode and 1 for the other modes. The SAD is expressed by

$$SAD = \sum_{i=1}^{4} \sum_{j=1}^{4} \left| S_{ij} - P_{ij} \right| \tag{7.5}$$

where S_{ij} and P_{ij} are the *(i, j)*th elements of the current original block **S** and the predicted block **P**, respectively. This SAD-based cost function saves a lot of computations as the distortion part is based on the differences between the original block and the predicted block instead of the reconstructed block. Thus, the processes of image block transformation, quantization, inverse quantization, inverse transformation and reconstruction of image block can all be saved. In addition, the rate part is pre-defined by constants either equal 4 or 0. Thus, the Context-adaptive variable-length coding (CAVLC) or Context-adaptive binary arithmetic coding (CABAC) can also be saved. However, the expense of the computation reduction usually comes with quite significant degradation of coding efficiency.

To achieve better rate-distortion performance, H.264/AVC also provided an alternative SATD-based cost function:

$$J_{SATD} = SATD + l_1 \cdot 4P \tag{7.6}$$

where SATD is sum of absolute Hadamard-transformed difference between the original block **S** and the predicted block **P**, which is given by

$$SATD = \sum_{i=1}^{4} \sum_{j=1}^{4} \left| h_{ij} \right| \tag{7.7}$$

where h_{ij} are the (i, j)th element of the Hadamard transformed image block **H**. The Hadamard transformed block **H** is defined as

$$\mathbf{H} = \mathbf{T}_H(\mathbf{S}-\mathbf{P})\mathbf{T}_H^T = \mathbf{T}_H(\mathbf{E})\mathbf{T}_H^T \tag{7.8}$$

with

$$\mathbf{T}_H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{7.9}$$

Experimental results show that the J_{SATD} could achieve better rate-distortion performance than the J_{SAD}, but it requires more computation due to the Hadamard transformation. If the fast Hadamard transform (FHT) is used, the computational requirement of the J_{SATD} can reduce half of the computation. However, the overall rate-distortion performance is still lower that the optimized J_{RD}. In order to reduce the computation of RD optimization several rate distortion models are proposed in literature [131-133, 135-136]. To improve the RD performance of J_{SATD}, an improved cost function for intra 4x4 mode decisions was proposed in [131]. In this cost function, sum of absolute integer transform differences (SAITD) is used in distortion part and a rate prediction algorithm is used in rate part. The major drawback of this cost function is that it requires performing the true integer transform. Even though fast transformation algorithm was proposed to perform SAITD but the overall complexity is still quite high. To reduce the complexity of rate-distortion cost computation, a fast bit rate estimation technique is proposed to avoid the entropy coding method during intra and inter mode decision of H.264/AVC [132]. But this cost function still need to calculate computation

expensive SSD. A bit rate estimation method based on standard deviation of transform coefficient is proposed in [134]. In this chapter, an enhanced SATD cost function is proposed which use the mean absolute deviation of the residual block in addition to SATD for the distortion part and a threshold is used to determine the large coefficients of the Hadamard transformed block for rate estimations.

7.2 The Cause of Sum of Square Differences (SSD)

Before we propose the new cost function, we first present the major cause of the sum square differences (SSD) between the original block and reconstruction block in the rate-distortion cost function. Mathematically, the original block **S** and reconstructed block **C** can be expressed as

$$\mathbf{S} = \mathbf{E} + \mathbf{P} \tag{7.10}$$

$$\mathbf{C} = \mathbf{E}' + \mathbf{P} \tag{7.11}$$

where the **P** is the predicted block, **E** is the residual block and **E**' is the reconstructed residual block. Then, the SSD can be expressed as [130]

$$\mathbf{SSD} = \|\mathbf{S} - \mathbf{C}\|_F^2 = \|\mathbf{E} + \mathbf{P} - \mathbf{E}' - \mathbf{P}\|_F^2 = \|\mathbf{E} - \mathbf{E}'\|_F^2 = \|\mathbf{F} - \mathbf{F}'\|_F^2 \tag{7.12}$$

where $\| \ \|_F$ is the Frobenius norm. and **F** and **F**' are the transformed residual block and inverse quantized-transformed residual block. That means the spatial-domain SSD is equivalent to DCT domain SSD. Based on this relationship, we can calculate the R-D cost in transform domain with $J_{RD} = SSD(\boldsymbol{F}, \mathbf{F}') + l \cdot R$ for saving the computation of

inverse DCT transform of image block . However, this computational reduction is very limited. The quantization is applied in the transformed coefficients of **F**. Thus, the cause of the SSD is due to the quantization errors in the DCT transformed residual block **F'**. This formation explains why the SATD can perform better than the SAD for the distortion part estimation for the rate-distortion cost function. It is because SATD uses the sum of the absolute coefficient values of **H** to estimate the distortion part, and property of the Hadamard transform is quite close to the Integer Transform used in H.264/AVC, thus the accuracy of SATD is much better than SAD. In this chapter, we use Hadamard transfrom coefficients to define an enhanced cost function, which maintain similar complexity of the SATD but with better rate-distortion performance.

7. 3 Enhanced SATD based Cost Function

It is reasonable to say that a complex block produces a large distortion value compared to a simple block. Therefore, residual block with high detail has larger distortion value than homogeneous block. Let us consider two residual 4x4 blocks $E_{\mathbf{A}}$ and $E_{\mathbf{B}}$.

$$\boldsymbol{E_A} = \begin{bmatrix} 0 & 10 & 8 & 10 \\ 9 & 7 & 4 & 10 \\ 1 & 10 & 11 & 4 \\ 19 & 6 & 15 & 7 \end{bmatrix} \text{ and } \boldsymbol{E_B} = \begin{bmatrix} 22 & 22 & 22 & 22 \\ 22 & 22 & 22 & 22 \\ 20 & 20 & 20 & 20 \\ 22 & 22 & 22 & 22 \end{bmatrix}$$

Assume *QP=24*. If we compute SSD based on Fig. 7.1, it is found that $SSD_A = 173$ and $SSD_B = 48$. That means distortion of block **A** is higher than that of block **B**. This is understandable because block **A** contains larger detail than block **B**. But if we calculate

SATD of A and B by (7.7), we found that both of the residual blocks produce same SATD value $SATD_A = SATD_B = 368$. In order to address this issue, the distortion part of the proposed cost function is calculated as follows

$$ESATD = SATD + a \times s(\boldsymbol{E}) \tag{7.13}$$

where *ESATD* is enhanced SATD and a is a constant value and $s(\boldsymbol{E})$ is the absolute deviation of the residual block **E** which represent the variation of pixel values. $s(\boldsymbol{E})$ is defined as

$$s(\boldsymbol{E}) = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}\left|E_{ij} - m\right| \tag{7.14}$$

$$\text{with } m = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}\left|E_{ij}\right| = \frac{h_{11}}{16} = (h_{11} >> 4)$$

Where h_{11} is the (1,1)th co-efficient of Hadamard transformed matrix **H** which is already computed for SATD calculation. The proposed cost function also uses a rate-predictor for estimating the rate for encoding the residual block instead of just using a penalty cost (4P) for the unfeasible modes. The rate-predictor is based on the property of the context-based adaptive variable length coding (CAVLC) entropy coder. To avoid DCT, quantization and CAVLC encoding processes, the proposed rate-predictor only uses the total number of the non-zero quantized Hadamard Transform coefficients (T_{bc}) which can be obtained by perform simple threshold and counter operations. Based on

the property of the CAVLS's VLC tables [16], the larger T_{bc} will produce more encoded bits. With the penalty cost (4P) from J_{SAD}, the overall estimated rate R_e is defined as

$$R_e = 3T_{bc} + 4P \tag{7.15}$$

The total number of the non-zero quantized Hadamard Transform coefficients (T_{bc}) is calculated as follows:

Step1: $T_{bc} = 0$;

Step 2: For i=1 to 4

For j=1 to 4

If ($|h_{ij}| \geq Q_{step}$) then $T_{bc} = T_{bc} + 1$

End for

End for

Q_{step} is the quantization step size used in H.264/AVC encoder. Values of Q_{step} with different QP are given in Table 7.1 [17]. Q_{step} doubles in size for every increment of 6 in QP. Therefore the proposed cost function is defined as

$$J_{ESATD} = ESATD + l_1 R_e \tag{7.16}$$

Table 7.1 Quantization step sizes Q_{step} in H.264/AVC codec

QP	Q_{step}
0	0.625
1	0.6875
2	0.8125
3	0.875
4	1
5	1.125
6	1.25
7	1.375
8	1.625
9	1.75
10	2
. . .	. . .
30	20
. . .	. . .
42	80
. . .	. . .
51	224

By putting value of ESATD and R_e, the cost function becomes

$$J_{ESATD} = SATD + a \times s\,(\boldsymbol{E}) + l_1(3T_{bc} + 4P) \tag{7.17}$$

In order to find the value of a , we have done some simulations with different types of sequences with different QPs and better results were found at $a = 1.25$. As explained in the previous section, the cause of the SSD is due to the quantization error of the integer transform coefficients of the residual block **E**, which can be estimated by sum of

absolute coefficients of **H** in SATD. It is well understandable that high frequency coefficients of Hadamard transform coefficients are insignificant and bears low values.

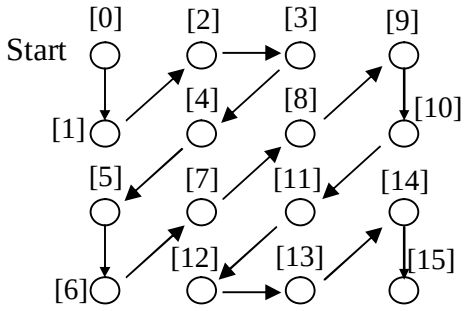


Fig. 7.2 Zig-zag scan and corresponding frequency of **H**

Fig. 7.2 shows the zig-zag scan and corresponding value of frequency of Hadamard Transform matrix **H. H** can be redefined in terms of frequency as follows.

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} = \begin{bmatrix} I_0 & I_2 & I_3 & I_9 \\ I_1 & I_4 & I_8 & I_{10} \\ I_5 & I_7 & I_{11} & I_{14} \\ I_6 & I_{12} & I_{13} & I_{15} \end{bmatrix} \tag{7.18}$$

where I_f is the Hadamard Transform coefficient of frequency f. After Hadamard transform, the high frequency coefficient usually has small energy. For this reason, the proposed cost function calculates only 10 low frequency Hadamard co-efficients instead of calculating all 16 co-efficient. Therefore the proposed cost function becomes,

$$J_{ESATD} = SATD' + a \times s(\boldsymbol{E}) + l_1(3T'_{bc} + 4P) \tag{7.19}$$

$$\text{with, } SATD' = \sum_{f=0}^{9} \left| I_f \right|$$

The step by step computation of proposed cost function is provided below:

1. Compute the predicted block: **P**
2. Compute the residual block: **E** = **S** – **P**
3. Compute only 10 low frequency coefficients of Hadamard transform matrix of the residual block: **H** = HT(**E**)
4. Calculate the $SATD'$ and T'_{bc}

   Set $SATD' = 0$ and $T'_{bc} = 0$

   For f=0 to 9

   $SATD' = SATD' + |I_f|$;

   If ($|I_f| \geq Q_{step}$) $T'_{bc} = T'_{bc} + 1$;

   End for
5. Calculate $s(\boldsymbol{E})$ by (7.14)
6. Calculate $J_{ESATD} = SATD' + a \times s(\boldsymbol{E}) + l_1(3T'_{bc} + 4P)$

7. 4 Simulation Results

The performance of proposed cost function was tested using the first 100 frames from nine different video sequences (*Akiyo*, *Claire*, *News, Container, Foreman, Stefan, Mother_daughter, Silent,* and *Hall*). They correspondingly represent different kinds of video: slow motion, medium motion and fast motion. For example "*Akiyo*" and "*News*"

are sequence of low spatial detail that contains low changes in motion. "*Foreman*" is a sequence with medium changes in motion and contains dominant luminance changes. "*Stefan*" contains panning motion and high spatial color detail. "*Container*" has dominant horizontal motion but also it contains a mixture of oscillating and slow motion in part of the slices. The experiment was carried out in the JVT JM 12.4 [129] encoder and the test parameters are listed as below:

-CAVLC is enabled;

-Frame rate is 30;

-All frames are I frame;

-Intra 4x4 prediction only;

-QPs are 30/36/42/48

-Number of frames: 100

7.4.1 Rate-distortion performance comparison

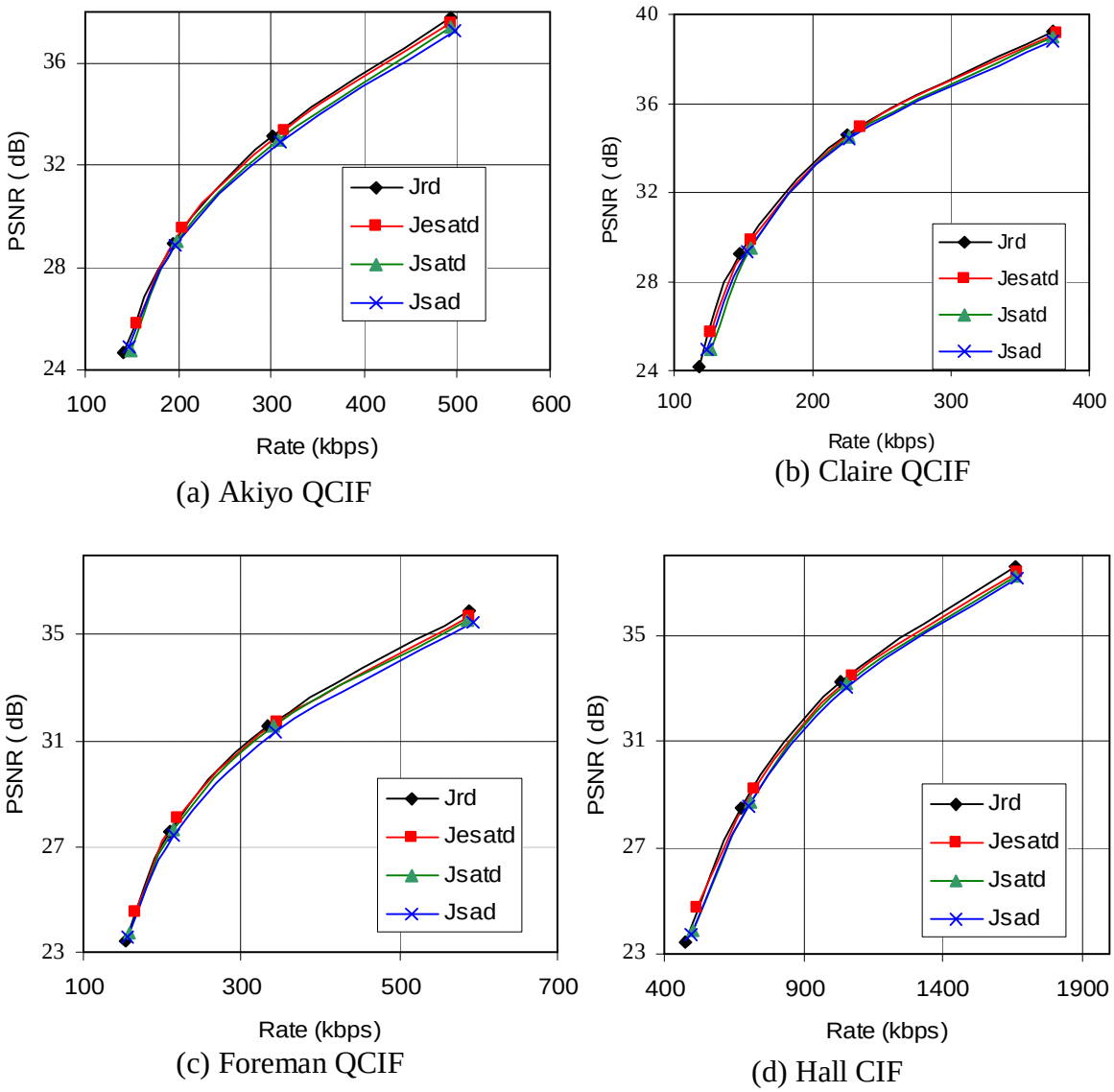
In these experiments, four cost functions (J_{RD}, J_{SAD}, J_{SATD}, and J_{ESATD}) are simulated. The PSNR and bit rate comparisons of the proposed cost function are tabulated in Table 7.2. PSNR and bit rate differences are calculated according to the numerical averages between RD curves derived from the original and proposed algorithm, respectively. The detail procedure to calculate these differences can be found in [67]. The positive values mean increments whereas negative values mean decrements. It is shown that RD performance degradation for both J_{SAD} and J_{SATD} are significant. In case of J_{SATD} the

average PSNR reduction is about 0.31 dB and average bit rate increment is about 7.10%. Whereas in our proposed method, the average PSNR reduction is about 0.13 dB and average bit rate increment is about 3.62%.

Table 7.2 PSNR and bit rate Comparison

Sequence	$\Delta PSNR$ in dB			ΔR%		
	J_{SAD}	J_{SATD}	J_{ESATD}	J_{SAD}	J_{SATD}	J_{ESATD}
Akiyo QCIF	-0.39	-0.37	-0.12	7.49	6.86	2.71
Foreman QCIF	-0.37	-0.21	-0.07	8.46	4.76	2.34
Container QCIF	-0.36	-0.30	-0.18	8.83	7.26	4.29
Claire QCIF	-0.33	-0.31	-0.01	5.96	5.64	1.52
Stefan QCIF	-0.60	-0.47	-0.27	16.84	12.99	7.43
News QCIF	-0.49	-0.42	-0.19	10.79	9.11	4.82
Mother_daughter QCIF	-0.29	-0.22	-0.07	6.90	4.86	2.13
Silent CIF	-0.28	-0.21	-0.12	8.90	6.20	4.82
Hall CIF	-0.36	-0.35	-0.14	6.41	6.29	2.59
Average	-0.38	-0.31	-0.13	8.95	7.10	3.62

The coding results of proposed method are very similar to actual RDO method. It is clear that our proposed cost function always perform better than J_{SATD} and J_{SAD}. The worst case is encoding of ***Stefan*** video sequence. For ***Stefan***, the proposed method increase the bit rate of about 7.43% whereas J_{SATD} generate around 12.99% of bit rate increment. The RD curves of four different types of video sequences are presented Fig. 7.3. It is shown that RD curve of proposed method is much closed to RD optimized curve and better than J_{SATD} and J_{SAD} for all type of sequences.



(a) Akiyo QCIF

(b) Claire QCIF

(c) Foreman QCIF

(d) Hall CIF

Fig. 7.3 Rate-distortion (RD) curves of four different cost functions

7.4.2 Complexity comparison

In the proposed method RD cost calculation is not required therefore, computation of DCT, quantization, entropy coding, inverse quantization and inverse DCT have saved.

In order to evaluate complexity reduction of proposed method as compared to original JM encoder, ΔT_1 (%) is defined as follows

$$\Delta T_1 = \frac{T_{original} - T_{proposed}}{T_{original}} \times 100\% \tag{7.20}$$

where, $T_{original}$ and $T_{proposed}$ denote the total encoding time of the JM 12.4 encoder with J_{RD} and with proposed cost function, respectively.

Table 7.3 Complexity comparison

Sequence	ΔT_1%		
	J_{SAD}	J_{SATD}	J_{ESATD}
Akiyo (QCIF)	86.76	85.92	85.76
Foreman (QCIF)	87.33	86.40	86.17
Container QCIF	85.49	84.65	83.99
Claire QCIF	84.03	83.15	82.49
Stefan QCIF	88.88	88.17	87.36
News QCIF	85.66	84.99	84.30
Mother_daughter QCIF	84.17	83.38	82.57
Silent CIF	87.43	86.99	86.67
Hall CIF	90.80	86.39	85.78
Average	86.73	85.56	85.01

Table 7.4 Comparison with J_{SAITD} [131]

Sequence	$\Delta PSNR$	ΔR%	ΔT_2 %
Akiyo (QCIF)	0.10	-1.93	16.24
Foreman (QCIF)	0.04	-0.89	18.69
Container QCIF	0.05	-1.48	14.23
Claire QCIF	0.11	-1.85	12.77
Stefan QCIF	0.08	-1.99	11.60
News QCIF	0.12	-2.66	13.62
Mother_daughter QCIF	0.09	-1.87	12.98
Silent CIF	0.04	-1.22	21.35
Hall CIF	0.06	-1.15	22.59
Average	0.10	-1.67	16.00

The complexity reductions of J_{SAD}, J_{SATD}, and J_{ESATD} are tabulated in Table 7.3. The proposed algorithm reduced about 85.01% of total encoding time compared to rate-

distortion optimized cost function J_{RD}. The computational reduction of proposed method is almost similar with J_{SATD}. However, the rate-distortion performance of proposed method is much better as compared with $J_{SATD.}$

7.4.3 Comparison with other method

In this experiment, the proposed method is compared with fast SAITD based cost function (J_{SAITD}) proposed in [131] in terms of rate distortion performance and complexity. Table 7.4 shows the comparison result. In Table 7.4, PSNR and bit rate performances are calculated based on [67] and Complexity reduction is calculated as follows:

$$\text{Complexity reduction } (\Delta T_2\,\%) = \frac{T_{ref[131]} - T_p}{T_{ref[131]}} \times 100\% \tag{7.21}$$

Where $T_{ref[131]}$ and T_p are the total encoding time of the method presented in [131] and proposed method, respectively. From the comparison result, it is shown that our proposed method reduced the bit rate of about 1.67% and increases the PSNR of about 0.10 dB on average. The proposed cost function not only improves the RD performance but also about 16% faster than that of J_{SAITD}.

7. 5 Summary

In this chapter, a simple and fast cost function based on SATD is proposed for intra mode decision of H.264/AVC. The distortion part of the cost function is estimated based on the SATD and mean absolute deviation of residual block. If mean absolute deviation is higher, the distortion is also higher. A bit rate predictor part also is introduced. Bit rate is predicted based on the number of large Hadamard Transformed coefficients. A block with large number of Hadamard Transformed coefficient produce large bit rate. The experimental results verified that the proposed technique is very suitable for intra mode decision of H.264/AVC. With the proposed scheme, DCT, quantization, inverse-quantization, inverse DCT operations can be skipped during the mode decision process. The proposed technique reduces encoding time by 85% with acceptable performance degradation. The RD performance and computational complexity of this algorithm is also better than methods stated in [131].

Chapter 8

Conclusion and Future Works

8.1 Concluding Remarks

Significant advances in digital video compression and communication methods make it possible to deliver high-quality video at low bit rates for today's networks. This state-of-the-art technology enables many possible networked multimedia, which include digital television, video over the Internet, wireless and mobile video, third generation (3G) cell phones with video capabilities, and the coming generation of Internet TV (IPTV). H.264/AVC, the latest video-coding standard, offers a solution to achieve high compression, but with an enormous increase in computational complexity. The main complexity of encoder comes from variable block size motion estimation (ME) and rate-distortion optimized (RDO) mode decision method.

Within video coding, motion estimation contributes to the largest gain in compression but is also the most computationally intensive part of it. In motion estimation, similarities between different video frames are searched and identified; redundant data are then eliminated or minimized to reduce temporal redundancy within a video

sequence. Many fast motion estimation methods have been developed over the last decade, most of these methods come with a complex search flow and with a limited speedup. In this dissertation, three different ways to reduce the computation of ME module were suggested.

At first, this dissertation proposed a novel edge based partial distortion search (EPDS) algorithm which reduces the computation of each distortion measure. Instead of calculating full distortion, distortion was calculated partially. At first the 16x16 MB was divided into 16 4x4 sub-blocks and calculation order of sub-blocks was sorted based on the edge strength of each block. The sub-block with larger edge strength was produced more distortion. This algorithm adaptively changed the early termination threshold for every accumulated partial sum of absolute differences. In addition to partial distortion, only selected numbers of search points were tested. Simulation results showed that the proposed method offered a remarkable improvement in computational speed when compared to full search (FS). The proposed method is 115 times faster than FS and PSNR degradation of the proposed algorithm is negligible and in the region of 0.01 dB.

Secondly, an adaptive early termination algorithm was suggested to reduce the number of search points that features an adaptive threshold based on the statistical characteristics of rate-distortion (RD) cost regarding current block and previously processed blocks and modes. After calculating RD cost of each search point, the RD cost was compared with an adaptive threshold. If RD cost is less than the threshold value, this search point was considered as the best search point for this block. In this way, most of the motions searched were stopped early and a lot of computation was saved. Based on the orientation of the previously calculated motion vectors, a region-

based search was also introduced. A search point reduction scheme for the fast motion estimation of H.264/AVC was also introduced, and the experimental results illustrated how the proposed method reduces ME times for full search and fast motion estimation by about 77 and 31%, respectively, despite the insignificant degradation of RD performance.

Thirdly, in order to reduce computations in motion estimation module, this dissertation also presented a novel adaptive search area selection method by utilizing the information of the previously computed motion vector differences (MVDs). In this method, motion search area was selected adaptively. A block with low motion has low search area and a block with large motion has the large search area. Experimental results showed that the proposed algorithm provides significant improvement in coding speed with negligible objective quality degradation compared to the full search motion estimation method adopted by H.264/AVC reference software. The proposed algorithm saved up to 92% of integer pixel ME time savings, compared to a full search ME of IPPP frames with a negligible PSNR reduction (0.015 db) and bit rate increment (0.47%).

This dissertation also developed some algorithms to improve the performance of H.264/AVC intra coder. An improved DC prediction (IDCP) mode based on the distance between the predicted and reference pixels was suggested. The proposed IDCP method improved the performance of original encoder by 0.10dB PSNR gain and 3.35% bit rate reduction. An adaptive number of mode selection method was also introduced to reduce the number of overhead bit to represent intra mode. In this method, around 44%

(shown in Table 6.6(a)) blocks does not need any bits to represent intra-mode. Two bits are required for 17% of blocks. The proposed methods not only improved the Rate-Distortion (RD) performance but also reduced the computational complexity of H.264/AVC intra encoder. Experimental results confirmed that the proposed methods saved 11.64% bit rate, improved the video quality by 0.34 dB on average, require 40.25% less computations than H.264/AVC intra encoder. The complexity increment of decoder side is very low. An enhanced MPM method was also presented in this dissertation. The MPM was derived from the prediction mode direction of neighboring blocks which have different weights according to their positions. This MPM selection method improved PSNR by 0.09 dB and bit rate by 3.12%.

Due to the use of large number of intra and inter prediction mode, the rate-distortion optimization (RDO) process is another time consuming module of H.264/AVC encoder. The most complex part of RDO mode decision methods includes the computations of the sum of squared difference (SSD) between the original and reconstructed image blocks and context-based variable length entropy coding (CAVLC) of the block. In this dissertation, a Hadamard transform-domain rate-distortion optimization accelerator based on SATD and CAVLC-based rate estimation algorithm were developed. In addition to SATD, mean absolute deviation was also used to estimate distortion part of the cost function and a threshold based large coefficients count was used for estimating the bit-rate part. Simulation results demonstrated that the proposed algorithm reduces about 85% of total encoding time with negligible degradation of coding performance.

8.2 Future Works

The motion estimation and mode decision algorithms presented in this dissertation can be improved in a variety of ways.

- Multi-view video receives many attentions in these years, because it can support a wide range of applications, such as 3D video communication and free view point video. To improve the coding efficiency, multi-view video coding not only employs temporal predictions but also predictions between different views. Inter-view prediction is also another time consuming part of multi-view video coding. The fast motion estimation methods developed in this dissertation can be extended to reduce the computation of inter-view prediction of multi-view video coding (MVC).
- The adaptive number of mode selection method and most probable mode selection method described in Chapter 6 can be extended to improve the compression performance of conventional inter-prediction method. After improvements, the developed method may be integrated in upcoming High Efficiency Video Coding (HEVC) standard [137]. HEVC aims to substantially improve coding efficiency compared to H.264/AVC High Profile, i.e. reduce bitrate requirements by half with comparable image quality, probably at the expense of increased computational complexity. We believe future research of this dissertation will play important role to successfully develop upcoming HEVC standard.

References

[1] A. M. Tekalp, “Digital Video Processing”, Prentice Hall, 1995.

[2] Y. Wang, J. Ostermann, Y. Q. Zhang, “Video processing and communications”, Prentice Hall, New Jersey, 2001

[3] MPEG website: http://www.mpeg.org.

[4] CCITT. Video Codec for Audiovisual services at p × 64 kbit/s, CCITT Recommendation H.261, 1990.

[5] ISO/IEC. MPEG-1: Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s - Video, Geneva, 1993.

[6] ISO/IEC JTC1 and ITU-T. MPEG-2/H.262: Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video, November 1994.

[7] ITU-T. Video Coding for Low Bit-rate Communication, ITU-T Recommendation H.263, March 1996.

[8] ITU-T. Video Coding for Low Bit-rate Communication, ITU-T Recommendation H.263, version 2 (H.263+), January 1998.

[9] ITU-T Recommendation H.263 - Annex U, "Video Coding for low bit rate communication - Annex U: Enhanced reference picture selection mode," Nov. 2000.

[10] ITU-T Recommendation H.263 - Annex V, "Video Coding for low bit rate communication - Annex V: Data-partitioned slice mode," Nov. 2000.

[11] ITU-T Recommendation H.263 - Annex W, "Video Coding for low bit rate communication - Annex W: Additional supplemental enhancement information specification," Nov. 2000.

[12] ISO/IEC JTC1/SC29/WG11. MPEG-4 Video Verification Model: Version 18.0, WG11 Document N3908, Pisa, January 2001. http://www. chiariglione.org/mpeg/working_documents.htm#MPEG-4.

[13] F. Pereira, editor. Signal Processing: Image Communication, volume 15. European Association for Signal Processing (EURASIP), January 2000. Tutorial Issue on the MPEG-4 Standard.

[14] I.E.G. Richardson. Video Codec Design: Developing Image and Video Compression Systems. John Wiley and Sons Ltd., 2002.

[15] UB Video. Emerging H.26L Standard - White Paper. Technical report, UBVideo Inc., Vancouver, BC, Canada, February 2002.

[16] ITU-T and ISO/IEC JTC1. Advanced Video Coding for Generic Audiovisual Services, ITU-T Recommendation H.264 ISO/IEC 14496-10 AVC, 2003.

[17] I.E.G. Richardson. H.264 and MPEG-4 Video Compression. John Wiley and Sons Ltd., 2003.

[18] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC Video Coding Standard. IEEE Transactions on Circuits and Systems for Video Technology, 13(7):560–576, July 2003.

[19] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G.J. Sullivan. Rate-Constrained Coder Control and Comparison of Video Coding Standards. IEEE Transactions on Circuits and Systems for Video Technology, 13(7):688–703, July 2003.

[20] Ting, Chi Wang, "Applications of fast low-distortion motion estimation algorithms in video coding" MPhil Thesis, City University of Hong Kong, 2005.

[21] "Emerging H.264 Standard: Overview and TMS320DM642-Based Solutions for real time video application," White paper, Available: http://focus.ti.com/pdfs/vf/vidimg/ubvideo_h26l_whitepaper_v24.pdf

[22] G. Sullivan (Rapporteur), Progress report and agenda items for Question 15 – advanced video coding," in SG16/Q15, Document Q15status. Geneva, Switzerland: ITU-T, Jan. 1998.

[23] G. Sullivan (Rapporteur), Meeting report of the tenth meeting (meeting J) of the ITU-T Q.15/16 advanced video coding experts group," in SG16/Q15, 9. Meeting, Document Q15-J78. Osaka, Japan: ITU-T, May 2000.

[24] G. Bjontegaard (Editor), H.26L test model long term 1," in SG16/Q15 VCEG, 8. Meeting, Document Q15-H36. Berlin, Germany: ITU-T, Aug. 1999.

[25] T. Wiegand (Editor), H.26L test model long term 9," in SG16/Q6 VCEG, 14. Meeting, Document VCEG-N83. Santa Barbara, CA: ITU-T, Sept. 2001.

[26] T. Wiegand (Editor), Joint model number 1, revision 1," in JVT, 1st Meeting, Document JVT-A003. Pattaya, Thailand: ITU-T, ISO/IEC, Dec. 2001.

[27] T. Wiegand (Editor) and G. Sullivan (Editor), Joint draft international standard (FDIS) of joint video speci_cation (ITU-T rec. H.264 ISO/IEC 14496-10 AVC)," in JVT, 7th Meeting, Document JVT-G050. Pattaya, Thailand: ITU-T, ISO/IEC, Mar. 2003.

[28] H.264/MPEG-4 AVC, From Wikipedia, the free encyclopedia, available online: http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[29] Soon-kak Kwon a, A. Tamhankar b, K.R. Rao c, "Overview of H.264/MPEG-4 part 10", Journal of Visual Communiaction and Image representation, vol. 17, 2006, pp. 186-216

[30] Andrew Gibson, " The H.264 Video compression standard", MSc thesis, Queens University, Kingston, Ontario, Canada, August 2002.

[31] M. Karczewicz and R. Kurçeren: "The SP and SI Frames Design for H.264/AVC," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, July 2003, pp. 637-644.

[32] Wikipedia- Broadband Internet Access http://en.wikipedia.org/wiki/Broadband_Internet_access#DSL_.28ADSL.2FSDSL.29\

[33] Atul Puria, Xuemin Chenb, Ajay Luthrac, "Video coding using the H.264/MPEG-4 AVC compression standard", Journal of Signal Processing: Image Communication, vol. 19 (2004), pp. 793–849

[34] G. Sullivan, P. Topiwala, A. Luthra, "The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions", in: SPIE Conference on Applications of Digital Image Processing XXVII, 2004.

[35] H. G. Musmann, P. Pirsh, and H. J. Grallert, "Advances in picture coding," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 523-548, Apr. 1985

[36] E. Chan, S. Panchanathan, "Motion estimation architecture for video compression", *IEEE Trans. Consumer Electro.*, vol. 39, pp. 292-297, Aug. 1993

[37] J.R.Jain and A.K.Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Communications*, vol. COM-29, pp. 1799-1808, Dec. 1981

[38] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Trans. Computers*, vol. 23, no. 1, pp. 90-93, Jan. 1974

[39] K. R. Rao and R. Yip, "Discrete cosine transform – algorithms, advantages, applications," New York: Academic Press, 1990

[40] H.S. Malvar, A. Hallapuro, and M.Karzewicz, " Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. Circuits Syst. Video. Technol.*, vol. 13, pp 598-603, July 2003.

[41] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Mag.*, vol. 15, pp. 74-90, Nov. 1998.

[42] M. C. Chen and A. N. Willson, Jr. "Rate-distortion optimal motion estimation algorithms for motion-compensated transform video coding," *IEEE Trans. Circuits Syst. Video Technol.*, 147-158, vol. 8, Apr. 1998.

[43] Jun Zhang , Xiaoquan Yi, Nam Ling, Weijia Shang, "Bit rate distribution for motion estimation in H.264 coding", IEEE Trans. On Consumer Electronics, vol. 52, no. 2, pp. 606-610, May 2006.

[44] Mohammed Golam Sarwer, " Fast Rate-distortion optimized mode decision of H.264/AVC video coding standard", MPhil thesis, Dept. of Electronic Engineering, City University of Hong Kong, 2007.

[45] Q. Chen, Y. He, "A fast bits estimation method for rate-distortion optimization in H.264/AVC", *Proc. Picture Coding Syst. (PCS 2004)*, paper No. 35, Dec. 2004

[46] D. Marpe, H.Schwarz, and T Wiegand, "Context-based adaptive binary coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video. Technol.*, vol. 13, no 7, pp. 620-636, July 2003.

[47] R.Li, B. Zeng, and M.L. Liuo, " A new three-step search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Tech.., vol. 4, no. 4, pp.438-442, August 1994.

[48] S. Zhu and K-K. Ma, " A new diamond search algorithm for fast block matching motion estimation," IEEE Trans. Image Processing, col. 92. no. 2, pp. 287-293, Feb. 2000.

[49] Yu-Kuang Tu, Jar-Ferr Yang, Ming-Ting Sun, and Yuesheng Tsai, " Fast variable block motion estimation for efficient H.264/AVC encoding." Signal Processing: Image Communication, vol. 20, pp. 595-623, 2005.

[50] Zhibo Chen, Peng Zhou, Yun He, "Fast Integer Pel and Fractional Pel Motion Estimation for JVT," Joint Video Team (JVT) Docs, JVT-F017, Dec. 2002.

[51] Feng Pan, Hongtao, and Zhiping Lin, "Scalable fast rate-distortion optimization for H.264/AVC," EURASIP journal of applied signal processing, vo. 2006, Article ID 37175, pages1-10.

[52] Yao-Chung Lin, and Torsten Fink, Erwin Beller, "Fast mode decision for H.264 based on rate-distortion cost estimation," ICASSP-2007, pp 1137-1140

[53] J. F. Yang, S. H. Chang, and C. Y. Chen, "Computation reduction for motion search in low rate video coders," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, pp. 948-951, Oct. 2002.

[54] L. Yang , K. Yu, J. Li, S. Li "An effective variable block size early termination algorithm for H.264 video coding," IEEE Trans. Circuits Syst. Video Tech.., vol. 15, no. 6, pp.784-788, June 2005.

[55] Sung-Eun Kim and Jong-Ki Han, "Efficient motion estimation using a modified early termination algorithm in H.264," IEICE Trans. Inf. & Syst., vol. E 88-D, no. 7. pp 1707-1715, July, 2005.

[56] Chung-Yen Su, and Shu-Li Chang, " Adaptive early termination for fast H.264 Video Coding", 9th IEEE International symposium on Multimedia 2007, pp. 72-76

[57] Gwo-Long Li, and Mei-Juan Chen, "Adaptive search range decision and early termination for multiple reference frame motion estimation for H.264," IEICE Trans. Commun., vol. E-89B, no. 1, January, 2006, pp. 250-253

[58] Xiaozhong Xu and Yun He, "Improvement on fast motion estimation strategy for H.264/AVC," IEEE Trans. Circuits Syst. Video Tech.., vol. 18, no. 3, pp.285-293, March-2008

[59] Esam A. Al Qarallah and Tian-Sheuan Chang, " Fast variable block size motion estimation by adaptive early termination," IEEE Trans. Circuits Syst. Video Tech.., vol. 16, no. 8, pp.1021-1026, August-2006

[60] Xiao Sul, Sweta Singh, and Yan Bai, "Local reference with early termination in H.264 Motion estimation," ICME 2007, pp.384-387

[61] Hasan F. Ates, and Yucel Altunbasak, "Rate-distortion and complexity optimized motion estimation for H.264/AVC video coding", IEEE Trans. Circuits Syst. Video Tech.., vol. 18 no. 2 pp.159-171, Feb-2008.

[62] Chun-Man Mak, Chi-Keung Fong, and Wai-Kuen Cham, "Fast Motion Estimation for H.264/AVC in Walsh–Hadamard Domain", IEEE Trans. Circuits Syst. Video Tech.., vol. 18 no. 6 pp.735-745, June-2008.

[63] Wenqi You, Yang Song, Takeshi Ikenaga, and Satoshi Goto, "A High Quality Fast Motion Estimation Algorithm for H.264/AVC", 2008 Congress on Image and Signal Processing, pp. 375-379.

[64] Z. Chen, P. Zhou, Y. He, and G. Wang, " Fast motion estimation for JVT", JVT-G016, March 2003.

[65] Soonjong Jin, Sang-Jun Park, and Jechang Jeong, "Adaptive fast full search Algorithm using partitioned region and optimized search order", IEEE Transaction on Consumer Electronics, vol. 53, no. 4, Nov. 2007, pp.-1703-1711

[66] http://iphome.hhi.de/suehring/tml/, JVT Reference Software version 9.6

[67] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," presented at the 13th VCEG-M33 Meeting, Austin, TX, April 2001.

[68] L.M. Po and W.C. Ma, " A novel four step search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 3, pp.-313-317, June 1996.

[69] J. Y. Tham, S. Ranganath, M. Ranganth, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 4, pp. 369–377, Aug. 1998.

[70] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 5, pp. 349–355, May 2002.

[71] C. Zhu, X. Lin, and L.-P. Chau, "Enhanced Hexagon search for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 14, no. 10, pp. 1210–1214, Oct. 2004.

[72] C. H. Cheung and L. M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 12, pp. 1168–1177, Dec. 2002.

[73] Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Adaptive Variable Block-Size Early motion estimation termination algorithm for H.264/AVC Video Coding Standard." IEEE Transaction on Circuit and System for Video Technology, volume 19, number 8, August 2009, pp 1196-1201.

[74] A. Tourapis, O. C. Au, and M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 10, pp. 934–947, Oct. 2002.

[75] B. Demir and S. Erturk, "Block motion estimation using adaptive modified two-bit transform, " IET Image Process., Vol. 1, No. 2, pp. 215-222, June 2007.

[76] S. W. Liu, S. D. Wei, and S. H. Lai, "Fast Optimal Motion Estimation Based on Gradient-Based Adaptive Multilevel Successive Elimination," IEEE Trans. Circuits Syst. Video Technol., vol. 18, no. 2, pp. 263-267, Feb 2008.

[77] K. B. Kim, Y. G. Jeon, and M. C. Hong, "Variable Step Search Fast Motion Estimation for H.264/AVC Video Coder," IEEE Trans. Consumer Electron., Vol. 54, No. 3, pp. 1281-1286, Aug. 2008.

[78] R. K. Purwar, N. Prakash, N. Rajpal, "A Quality Based Motion Estimation Criterion for Temporal Coding of Video," 2009 Seventh International Conf. on Advances in Pattern Recognition, pp. 61-54.

[79] W. You, Y. Song, T. Ikenaga, and S. Goto, "A High Quality Fast Motion Estimation Algorithm for H.264/AVC", 2008 Congress on Image and Signal Process., pp. 375-379, May 2008.

[80] S. Eckart and C. Fogg, ' ISO/IEC MPEG-2 software video codec," Proc. SPIE, vol. 2419, 100-118 (1995).

[81] C. K. Cheung and L. M. Po, " Normalized partial distortion algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 10, no. 3, pp-417-422, April-2000.

[82] C. K. Cheung and L. M. Po, "Adjustable partial distortion search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 1, pp. 100–110, Jan. 2003.

[83] W. G. Hong, and T. M. Oh, “Enhanced partial distortion search algorithm for block motion estimation,” Electronic Letters, 39(15), pp.1112-1113, 2003.

[84] J.N. Kim, S.C. Byun, Y. H. Kim, and B. H. Ahn, “ Fast full search motion estimation algorithm using early detection of impossible candidate vectors,” IEEE Trans. Signal Process., vol. 50, no. 9, pp. 2355-2365, Sep 2002.

[85] B. Montrucchio, and D. Quaglia, “ New sorting based lossless motion estimation algorithms and a partial distortion elimination performance analysis,” IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 2, pp-210-220 , Feb. 2005.

[86] K. C. Hui, W. C. Siu, and Y.L.Chan, “ New adaptive partial distortion search using clustered pixel matching error characteristics”, IEEE Trans. on image proc., vol. 14, no. 5, pp. 597-607, May 2005.

[87] C. C. Wang, and C. J. Lo, “ Using two stage sorting based partial distortion search for motion estimation in H.264/AVC,” Optical Engineering, 46(9), 097002, Sep., 2007.

[88] C. C. Wang, C. J. Lo, C. W. Yu, “ Efficient motion estimation using sorting based partial distortion search,” proc. IEEE Intl. conf. Multimedia & expo., Toronto, Canada, pp. 153-156, July 2006.

[89] S.L.Shin, S. Lee, and J.S.Oh, “Fast partial difference elimination algorithm based on block matching error prediction,” OE letters, 46(4), 040503, April 2007.

[90] S. Jin, H. Lee, and J. Jeong, “ Hadamard transform based fast partial distortion elimination algorithm for lossless and lossy motion estimation,” 2008 Congress of Image and Signal Process., Sanya, China , pp. 201-205, May 2008.

[91] Donald Knuth, "The Art of Computer Programming", Volume 3: Sorting and Searching, Second Edition. Addison-Wesley, 1998. ISBN 0-201-89685-0., pp.80–105.

[92] T. H. Cormen, C. E. Leiserson, R.L. Rivest, and C. Stein., "Introduction to Algorithms", Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7, pp.168–170.

[93] X. Yi and N. Ling, "Improved Normalized Partial Distortion Search with dual-halfway-stop for rapid block motion estimation", IEEE Trans. on Multimedia, vol. 9, no. 5, pp. 995-1003, Aug. 2007.

[94] M. G. Sarwer, and Q..M. Jonathan Wu, "Fast Block Motion Estimation by edge based partial distortion search", Proceedings of the IEEE International Conference on Image Processing 2009 (ICIP-2009), Cairo, Egypt, November 7-10, 2009, pp. 1573 - 1576.

[95] M. G. Sarwer**,** and Q.M. Jonathan Wu "Efficient Two Step Edge based Partial Distortion Search for Fast Block Motion Estimation" ***IEEE Trans. on Consumer Electronics***, Vol. 55, No. 4, Nov. 2009, pp. 2154-2162.

[96] M. G. Sarwer, and Q.M. Jonathan Wu, "Region Based Searching for Early Terminated Motion Estimation Algorithm of H.264/AVC Video Coding Standard", Proceedings of the IEEE CCECE 2009, St John's, NL, pp. 468-471.

[97] M. G. Sarwer, and Q.M. Jonathan Wu, "Adaptive Search Area Selection of Variable Block-Size Motion Estimation of H.264/AVC Video Coding Standard", IEEE International Symposium on Multimedia ISM 2009, San Diego, California, USA, December 14-16, 2009, pp. 100-105.

[98] M. G. Sarwer, and Q.M. Jonathan Wu, “Enhanced Intra Coding of H.264/AVC Advanced Video Coding Standard with Adaptive Number of Modes”, accepted in 2010 International Conferences on Active Media Technology (AMT 2010), Toronto, Canada, August 2010.

[99] M. G. Sarwer, and Q.M. Jonathan Wu “Improved Intra Prediction of H.264/AVC” accepted in the book "*Video Coding*", ISBN 978-953-7619-X-X, IN-TECH publisher.

[100] M. G. Sarwer, and Q.M. Jonathan Wu, “A Novel Bit Rate Reduction Method of H.264/AVC Intra Coding”, International Congress on Image and Signal Processing (CISP'10), 16-18 October 2010, Yantai, China, pp. 24-28.

[101] Si-Woong Lee, Seong-Mo Park, and Hyun-soo Kang, “ Fast Motion estimation with adaptive search range adjustment”, Optical Engineering Letter, vol. 46, no. 4, April 2007.

[102] Z. Chen, Y. Song, T. Ikenaga, and S. Goto, “ Adaptive search range algorithms for variable block size motion estimation in H.264/AVC”, IEICE Trans. Fundamental, vol. E91-A, no.4, pp. 1015-1022, April 2008.

[103] S. Goel, Y. Ismail, and M.A. Bayoumi “Adaptive search window size algorithm for fast motion estimation in H.264/AVC standard”, 48th Midwest Symposium on Circuits and Systems, 2005, Volume, Issue 7-10 Aug. 2005 Page(s): 1557 – 1560.

[104] T. Yamada, M. Ikekawa, I. Kuroda, “Fast and accurate motion estimation algorithm by adaptive search range and shape selection,” Proc. ICASSP, vol.2, pp. 897-900, March-2005.

[105] M. C. Hong and H.H. Oh, "Range Decision for Motion estimation of VCEG-N33," JVT B022, Switherland, February 2002.

[106] X. Xu and Y.HE, "Modification of Dynamic Search range for JVT," JVT Q088-L, USA, Oct. 2005.

[107] Tian Song , K. Ogata, K. Saito, T. Shimamoto, "Adaptive Search Range Motion Estimation Algorithm for H.264/AVC", IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007, 27-30 May 2007, pp. 3956 - 3959

[108] M.G. Sarwer, T. M. Nguyen, Q.M.J. Wu, " Fast Motion estimation of H.264/AVC by adaptive early termination," Proceedings of the 10th IASTED International Conference Signal and Image Processing (SIP 2008), August 18-20, 2008, HI, USA, pp. 140-145.

[109] J. Yang, B. Yin, Y. Sun, and N. Zhang, A Block- Matching based intra frame prediction for H.264/AVC, Proceeding of ICME 2006, July 2006, pp. 705-708.

[110] T. K. Tan, C. S. Boon, and Y. Suzuki, Intra Prediction by template matching, Proceeding of ICIP 2006, Atlanta, GA, USA, October 2006, pp. 1693-1696.

[111] T. K. Tan, C. S. Boon, and Y. Suzuki, Intra prediction by averaged template matching predictors, in Proc. IEEE Consumer Communications & Networking Conference' 07, Jan. 2007, pp. 405-409.

[112] J. Balle, and M. Wien, Extended texture prediction for H.264/AVC intra coding, in Proc. IEEE ICIP 2007, September 2007, pp. VI-93-VI-96.

[113] T. Shiodera, A. Tanizawa, T. Chujoh, Block based extra/inter-polating prediction for intra coding, in Proc. IEEE ICIP 2007, September 2007, pp. VI-445-VI-448.

[114] T. Shiodera, A. Tanizawa, and T. Chujoh, Bidirectional intra prediction, ITU-T SG16/Q.6 VCEG, VCEG-AE14, Marrakech, Morocco, Jan. 2007.

[115] Y. Ye, and M. Karczewicz, Improved H.264 Intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning, in Proc. IEEE ICIP 2008, October 2008, pp. 2116-2119.

[116] S. Yu, Y. Gao, J. Chen, and J. Zhou, Distance based weighted prediction for H.264 Intra Coding, in Proc. ICALIP 2008, July 2008, pp. 1477-1480.

[117] L. Wang, L. M. Po, Y. M. S. Uddin, K. M. Wong, and S. Li, A novel weighted cross prediction for H.264 intra coding, IEEE ICME 2009, June 2009, pp. 165-168.

[118] L. Song, Y. Xu, C. Xiong, and L. Traversoni, Improved intra-coding method for H.264/AVC, EURASIP Journal of advanced signal processing, volume 2009, Article ID 328958, doi: 10.1155/2009/328958.

[119] S. Matsuo, S. Takamura, and Y. Yashima, Intra prediction with spatial gradients and multiple reference lines, Picture Coding Symposium 2009, PCS 2009, May 2009, pp. 1-4.

[120] K. L. Tang, K. N. Ngan, Enhancement techniques for intra block matching, in Proc. IEEE ICME 2007, July 2007, pp. 420-423.

[121] C. S. Park, and S. J. Ko, Estimation-based intra prediction algorithm for H.264/AVC, OE Letters, vol. 48, no. 3, March 2009, pp. 030506-1-3.

[122] A. Robert, I. Amonou, and B. P. Popescu, Improving intra mode coding in H.264/AVC through block oriented transforms, in Proc. IEEE workshop on Multimedia Signal Processing 2006, October 2006, pp. 382-386.

[123] M. G, Sarwer, L. M. Po and J. Wu, Fast Sum of Absolute Transformed Difference based 4x4 Intra Mode Decision of H.264/AVC Video Coding Standard, Elsevier Journal of Signal Process.: Image Commun., Vol. 23,no. 8, Sep.2008, pp. 571-580.

[124] A. C. Tsai, A. Paul, J. C. Wang, and J. F. Wang, Intensity gradient technique for efficient Intra prediction in H.264/AVC, IEEE Trans. Circuits and Systems for Video Technology, vol. 18, no. 5, May 2008, pp. 694-698.

[125] B. G. Kim, Fast selective intra mode search algorithm based on adaptive thresholding scheme for H.264/AVC encoding, IEEE Trans. Circuits and Systems for Video Technology, vol. 18, no. 1, Jan. 2008, pp. 127-133.

[126] D. Y. Kim, K. H. Han, and Y. L. Lee, Adaptive single-multiple prediction of H.264/AVC intra coding, IEEE Trans. Circuits and Systems for Video Technology, vol. 20, no. 4, April 2010, pp. 610-615.

[127] D. Y. Kim, D. K. Kim, and Y. L. Lee, A New Method for Estimating Intra Prediction Mode in H.264/AVC, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E91-A, June 2008 pp. 1529-1532,.

[128] Jinho Lee, Jin Soo Choi, Jinwoo Hong, and Haechul Choi, Intra-mixture Prediction Mode and Enhanced Most Probable Mode Estimation for Intra Coding in H.264/AVC, Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM '09, August 2009, pp. 1619-1622.

[129] JM reference software 12.4, http://iphome.hhi.de/suehring/tml/download/old_jm/jm12.4.zip

[130] L. M. Po and K. Guo, "Transform-Domain Fast Sum of the Squared Difference Computation for H.264 Rate-Distortion Optimization," IEEE Trans. Circuits Syst. Video Tech., vol. 17, No. 6, pp. 765-773, June 2007.

[131] C.H. Tseng, H.M. Wang, and J. F. Yang, "Enhanced Intra 4x4 Mode Decision for H.264/AVC Coders," IEEE Trans. Circuits Syst. Video Tech., vol. 16, no. 8, pp.1027-1032, August 2006.

[132] Mohammed Golam Sarwer, and Lai Man Po "Fast bit rate estimation for mode decision of H.264/AVC" IEEE Trans. Circuits Syst. Video Tech., vol. 17, no. 10, October 2007, pp 1402-1407.

[133]Yu-Kuang, Jar-Ferr Yang, and Ming-Ting Sun, "Efficient rate-distortion estimation for H.264/AVC coders." IEEE Trans. Circuits Syst. Video Tech., vol. 16, no. 5, pp.600-611, May 2006.

[134] Yu-Ming Lee, Yu-Ting Sun, and Yinyi Lin "SATD-Based Intra Mode Decision for H.264/AVC Video Coding", IEEE Trans. Circuits Syst. Video Tech., vol. 20, no. 3, pp.463-469, March 2010.

[135] O. Wang, D. Zhao, W. Gao, and S. Ma "Low complexity RDO mode decision based on a fast coding-bits estimation model for H.264/AVC". IEEE International Symposium on Circuits and Systems, 2005, vol. 4, 3467 – 3470, (2005).

[136] J. Hahm, C. M. Kyung, "Efficient CABAC Rate Estimation for H.264/AVC Mode Decision" IEEE Trans. Circuits Syst. Video Tech., vol. 20, no. 2, 310-316 (2006).

[137] Joint Collaborative Team on Video Coding (JCT-VC) http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/jctvc.aspx

Appendix A

Encoder Configuration

In H.264/AVC reference software encoder parameters are adjusted by *encoder.cfg* file. The follow shows part of the *encoder.cfg* file of JM9.6 reference software.

```
###########################################################################
# Files
###########################################################################
InputFile           = "news_cif.yuv"       # Input sequence
InputHeaderLength    = 0     # If the inputfile has a header, state it's length in byte here
StartFrame          = 0     # Start frame for encoding. (0-N)
FramesToBeEncoded    = 100     # Number of frames to be coded
FrameRate           = 30.0   # Frame Rate per second (0.1-100.0)
SourceWidth          = 352   # Frame width
SourceHeight         = 288   # Frame height
TraceFile           = "trace_enc.txt"
ReconFile           = "test_rec.yuv"
OutputFile          = "test.264"

###########################################################################
# Encoder Control
###########################################################################
ProfileIDC            = 77  # Profile IDC (66=baseline, 77=main, 88=extended; FREXT
                            Profiles: 100=High, 110=High 10, 122=High 4:2:2, 144=High
                            4:4:4, for params see below)
LevelIDC            = 40  # Level IDC   (e.g. 20 = level 2.0)
IntraPeriod         =  4  # Period of I-Frames (0=only first)
IDRIntraEnable       =  0  # Force IDR Intra  (0=disable 1=enable)
QPISlice            =  28  # Quant. param for I Slices (0-51)
QPPSlice            = 28  # Quant. param for P Slices (0-51)
FrameSkip            =  0  # Number of reference frames to be skipped in input (e.g 2 will
                            code every third frame)
ChromaQPOffset       =  0  # Chroma QP offset (-51..51)
```

```
UseHadamard          =  0  # Hadamard transform (0=not used, 1=used)
SearchRange          = 16  # Max search range
NumberReferenceFrames =  1  # Number of previous frames used for inter motion search (1-16)
PList0References     =  0  # P slice List 0 reference override (0 disable, N <= NumberReferenceFrames)
Log2MaxFrameNum      =  0  # Sets log2_max_frame_num_minus4 (0-3 : based on FramesToBeEncoded, >3 : Log2MaxFrameNum - 4)
MbLineIntraUpdate    =  0  # Error robustness(extra intra macro block updates)(0=off, N: One GOB every N frames are intra coded)
RandomIntraMBRefresh =  0  # Forced intra MBs per picture
InterSearch16x16     =  1  # Inter block search 16x16 (0=disable, 1=enable)
InterSearch16x8      =  1  # Inter block search 16x8  (0=disable, 1=enable)
InterSearch8x16      =  1  # Inter block search  8x16 (0=disable, 1=enable)
InterSearch8x8       =  1  # Inter block search  8x8  (0=disable, 1=enable)
InterSearch8x4       =  1  # Inter block search  8x4  (0=disable, 1=enable)
InterSearch4x8       =  1  # Inter block search  4x8  (0=disable, 1=enable)
InterSearch4x4       =  1  # Inter block search  4x4  (0=disable, 1=enable)
UseFME               =  0  # Use fast motion estimation (0=disable, 1=enable)

###############################################################################
# B Slices
###############################################################################
NumberBFrames        =  0  # Number of B coded frames inserted (0=not used)
QPBSlice             = 35 # Quant. param for B slices (0-51)
BRefPicQPOffset      =  0  # Quantization offset for reference B coded pictures (-51..51)
DirectModeType       =  1  # Direct Mode Type (0:Temporal 1:Spatial)
DirectInferenceFlag  =  1  # Direct Inference Flag (0: Disable 1: Enable)
BList0References     =  3  # B slice List 0 reference override (0 disable, N <= NumberReferenceFrames)
BList1References     =  3  # B slice List 1 reference override (0 disable, N <= NumberReferenceFrames)
                           # 1 List1 reference is usually recommended for normal GOP Structures.
                           # A larger value is usually more appropriate if a more flexible
                           # structure is used (i.e. using PyramidCoding)
BReferencePictures   =  0  # Referenced B coded pictures (0=off, 1=on)
PyramidCoding        =  0  # B pyramid (0= off, 1= 2 layers, 2= 2 full pyramid, 3 = explicit)
ExplicitPyramidFormat = "b1r30b3r20b2e34b0e30b4r22" # Explicit Enhancement GOP. Format is {FrameDisplay_orderReferenceQP}.
                           # Valid values for reference type is r:reference, e:non reference.
```

```
PyramidRefReorder      =  1  # Reorder References according to Poc distance for
                             PyramidCoding (0=off, 1=enable)
PocMemoryManagement    =  1  # Memory management based on Poc Distances for
                             PyramidCoding (0=off, 1=on)

###############################################################################
# SP Frames
###############################################################################
SPPicturePeriodicity  =  0  # SP-Picture Periodicity (0=not used)
QPSPSlice             = 28  # Quant. param of SP-Slices for Prediction Error (0-51)
QPSP2Slice            = 27  # Quant. param of SP-Slices for Predicted Blocks (0-51)
###############################################################################
# Output Control, NALs
###############################################################################
SymbolMode             =  0  # Symbol mode (Entropy coding method: 0=UVLC,
                             1=CABAC)
OutFileMode           =  0  # Output file mode, 0:Annex B, 1:RTP
PartitionMode         =  0  # Partition Mode, 0: no DP, 1: 3 Partitions per Slice

###############################################################################
# CABAC context initialization
###############################################################################
ContextInitMethod     =  1    # Context init (0: fixed, 1: adaptive)
FixedModelNumber      =  0    # model number for fixed decision for inter slices ( 0,
                             1, or 2 )

###############################################################################
# Interlace Handling
###############################################################################

PicInterlace          =  0    # Picture AFF    (0: frame coding, 1: field coding, 2:adaptive
                              frame/field coding)
MbInterlace            =  0    # Macroblock AFF (0: frame coding, 1: field coding,
                             2:adaptive frame/field coding, 3:combined with
                             PicInterlace=0, to do frame MBAFF))
IntraBottom           =  0    # Force Intra Bottom at GOP Period

###############################################################################
# Weighted Prediction
###############################################################################
WeightedPrediction     =  0    # P picture Weighted Prediction (0=off, 1=explicit
                               mode)
WeightedBiprediction   =  0    # B picture Weighted Prediciton (0=off, 1=explicit
                             mode,  2=implicit mode)
UseWeightedReferenceME  =  0    # Use weighted reference for ME (0=off, 1=on)
```

```
###########################################################################
# Loop filter parameters
###########################################################################
LoopFilterParametersFlag = 0       # Configure loop filter (0=parameter below ingored,
                                     1=parameters sent)
LoopFilterDisable        = 0       # Disable loop filter in slice header (0=Filter, 1=No
                                     Filter)
LoopFilterAlphaC0Offset  = 0     # Alpha & C0 offset div. 2, {-6, -5, ... 0, +1, .. +6}
LoopFilterBetaOffset     = 0     # Beta offset div. 2, {-6, -5, ... 0, +1, .. +6}

###########################################################################
# Error Resilience / Slices
###########################################################################

SliceMode            =  0   # Slice mode (0=off 1=fixed #mb in slice 2=fixed #bytes in
                              slice 3=use callback)
SliceArgument        = 50   # Slice argument (Arguments to modes 1 and 2 above)

num_slice_groups_minus1 = 0  # Number of Slice Groups Minus 1, 0 == no FMO, 1 ==
                             two slice groups, etc.
slice_group_map_type   = 0       # 0:  Interleave, 1: Dispersed,    2: Foreground with left-
                                   over,
                    # 3:  Box-out,    4: Raster Scan   5: Wipe
                    # 6:  Explicit, slice_group_id read from SliceGroupConfigFileName
slice_group_change_direction_flag = 0     # 0: box-out clockwise, raster scan or wipe
                                            right,
                         # 1: box-out counter clockwise, reverse raster scan or wipe left
                           slice_group_change_rate_minus1    = 85   #
SliceGroupConfigFileName         = "sg0conf.cfg"   # Used for slice_group_map_type 0,
                                                     2, 6

UseRedundantSlice      = 0     # 0: not used, 1: one redundant slice used for each slice
                                 (other modes not supported yet)

###########################################################################
# Search Range Restriction / RD Optimization
###########################################################################
RestrictSearchRange  =  2  # restriction for (0: blocks and ref, 1: ref, 2: no restrictions)
RDOptimization       =  1  # rd-optimized mode decision
                             # 0: RD-off (Low complexity mode)
                             # 1: RD-on (High complexity mode)
                          # 2: RD-on (Fast high complexity mode - not work in FREX
                             Profiles)
                          # 3: with losses
DisableThresholding  =  0  # Disable Thresholding of Transform Coefficients (0:off,
                             1:on)
```

```
DisableBSkipRDO       =  0  # Disable B Skip Mode consideration from RDO Mode
                            decision (0:off, 1:on)

LossRateA             = 10  # expected packet loss rate of the channel for the first partition,
                            only valid if RDOptimization = 2
LossRateB             =  0  # expected packet loss rate of the channel for the second
                            partition, only valid if RDOptimization = 2
LossRateC             =  0  # expected packet loss rate of the channel for the third partition,
                            only valid if RDOptimization = 2
NumberOfDecoders      = 30  # Numbers of decoders used to simulate the channel, only
                            valid if RDOptimization = 2
RestrictRefFrames     =  0  # Doesnt allow reference to areas that have been intra updated
                            in a later frame.

###################################################################################
# Additional Stuff
###################################################################################

UseConstrainedIntraPred =  0  # If 1, Inter pixels are not used for Intra macroblock
                            prediction.
LastFrameNumber       =  0  # Last frame number that have to be coded (0: no effect)
ChangeQPI             = 16  # QP (I-slices)  for second part of sequence (0-51)
ChangeQPP             = 16  # QP (P-slices)  for second part of sequence (0-51)
ChangeQPB             = 18  # QP (B-slices)  for second part of sequence (0-51)
ChangeQPBSRefOffset   =  2  # QP offset (stored B-slices)  for second part of
                            sequence (-51..51)
ChangeQPStart         =  0  # Frame no. for second part of sequence (0: no second part)
NumberofLeakyBuckets  =  8                # Number of Leaky Bucket values
LeakyBucketRateFile   = "leakybucketrate.cfg"  # File from which encoder derives
                                                rate values
LeakyBucketParamFile  = "leakybucketparam.cfg" # File where encoder stores
                                                leakybucketparams

NumberFramesInEnhancementLayerSubSequence  = 0  # number of frames in the
                                                  Enhanced Scalability
                                                  Layer(0: no Enhanced
                                                  Layer)
NumberOfFrameInSecondIGOP             = 0  # Number of frames to be coded in the
                                             second IGOP
SparePictureOption    =  0  # (0: no spare picture info, 1: spare picture available)
SparePictureDetectionThr =  6  # Threshold for spare reference pictures detection
SparePicturePercentageThr = 92  # Threshold for the spare macroblock percentage
PicOrderCntType       = 0   # (0: POC mode 0, 1: POC mode 1, 2: POC mode 2)
```

```
###########################################################################
#Rate control
###########################################################################

RateControlEnable    =     0   # 0 Disable, 1 Enable
Bitrate              =  10000000   # Bitrate(bps)
InitialQP            =    35   # Initial Quantization Parameter for the first I frame
                               # InitialQp depends on two values: Bits Per Picture,
                               # and the GOP length
BasicUnit            =    10   # Number of MBs in the basic unit
                               # should be a fractor of the total number
                               # of MBs in a frame
ChannelType          =     0   # type of channel( 1=time varying channel; 0=Constant
                                 channel)

###########################################################################
#FREXT stuff
###########################################################################

YUVFormat            = 1     # YUV format (0=4:0:0, 1=4:2:0, 2=4:2:2, 3=4:4:4)
RGBInput             = 0     # 1=RGB input, 0=GBR or YUV input
BitDepthLuma         = 8     # Bit Depth for Luminance (8...12 bits)
BitDepthChroma       = 8     # Bit Depth for Chrominance (8...12 bits)
CbQPOffset           = 0     # Chroma QP offset for Cb-part (-51..51)
CrQPOffset           = 0     # Chroma QP offset for Cr-part (-51..51)
Transform8x8Mode     = 0     # (0 : only 4x4 transform, 1: allow using 8x8 transform
                               additionally, 2: only 8x8 transform)
ResidueTransformFlag = 0     # (0: no residue color transform 1: apply residue color
                               transform)
ReportFrameStats     = 0     # (0:Disable Frame Statistics 1: Enable)

###########################################################################
#Q-Matrix (FREXT)
###########################################################################
QmatrixFile          = "q_matrix.cfg"
ScalingMatrixPresentFlag = 0   # Enable Q_Matrix  (0 Not present, 1 Present in SPS, 2
                                 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag0  = 3   # Intra4x4_Luma    (0 Not present, 1 Present in SPS, 2
                                 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag1  = 3   # Intra4x4_ChromaU (0 Not present, 1 Present in SPS, 2
                                 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag2  = 3   # Intra4x4_chromaV (0 Not present, 1 Present in SPS, 2
                                 Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag3  = 3   # Inter4x4_Luma    (0 Not present, 1 Present in SPS, 2
                                 Present in PPS, 3 Present in both SPS & PPS)
```

```
ScalingListPresentFlag4  = 3    # Inter4x4_ChromaU (0 Not present, 1 Present in SPS, 2
                                  Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag5  = 3    # Inter4x4_ChromaV (0 Not present, 1 Present in SPS, 2
                                  Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag6  = 3    # Intra8x8_Luma    (0 Not present, 1 Present in SPS, 2
                                  Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag7  = 3    # Inter8x8_Luma    (0 Not present, 1 Present in SPS, 2
                                    Present in PPS, 3 Present in both SPS & PPS)

###########################################################################
#Lossless Coding (FREXT)
###########################################################################
QPPrimeYZeroTransformBypassFlag = 0    # Enable lossless coding when qpprime_y is
                                         zero (0 Disabled, 1 Enabled)
```

Vita Auctoris

Mohammed Golam Sarwer was born in Comilla, Bangladesh. He received his B.Sc. in Electrical and Electronic Engineering and M.Phil degree in electronic engineering from Khulna University of Engineering and Technology and City University of Hong Kong in 2001 and 2007, respectively. He was working as a full time faculty member in Khulna University of Engineering and Technology from 2001 to 2005. Currently, he is working as a Postdoctoral research fellow in the department of Electrical and Computer engineering, Ryerson University, Canada. He has published 39 peer-reviewed papers in international journals and conferences. His research interest includes video and image coding, video retrieval, video surveillance system, scalable and multi-view video coding, 3D TV etc.

Mr. Sarwer is the student member of IEEE and the member of International Association of Engineers (IAENG). In 2003, he won the **prime minister gold medal** due to outstanding academic performance of undergraduate level. In 2009 and 2010, he won prestigious Fredrick Atskin graduate award, University of Windsor, due to outstanding academic performance of the PhD level. He has also received prestigious Ontario Graduate Scholarship (OGS) for last year of this PhD study. He is also winner of **Outstanding Graduate Research Award**, top University's internal recognition award, from University of Windsor. He is also received Ontario Ministry of Research and Innovation Postdoctoral Fellowship.

List of Publications

Book Chapters

1. Mohammed Golam Sarwer, and Q.M. Jonathan Wu “Improved Intra Prediction of H.264/AVC” accepted in the book "Video Coding", ISBN 978-953-7619-X-X, IN-TECH publisher.
2. Mohammed Golam Sarwer, Lai Man Po and Q.M. Jonathan Wu “Bit Rate Estimation for cost function of H.264/AVC.” Multimedia, Kazuki Nishi (Ed.), ISBN: 978-953-7619-87-9, INTECH, pp. 257-280.

Articles in Journals

3. Mohammed Golam Sarwer, and Q.M. Jonathan Wu “Enhanced SATD based Cost Function for Mode Selection of H.264/AVC Intra Coding”. Submitted to *Springer Journal of Signal, Image and Video Processing*, Date of submission: November 10, 2010, manuscript number: 1556, 16 pages
4. Mohammed Golam Sarwer, and Q.M. Jonathan Wu “Performance Improvement of Intra Coding in H.264/AVC Advanced Video Coding Standard”. Submitted to *Journal of Visual Communication and Image Representation*, Date of submission: May 20, 2010, manuscript number: JVCI-10-80, 29 pages
5. Mohammed Golam Sarwer, and Q.M. Jonathan Wu “Efficient Two Step Edge based Partial Distortion Search for Fast Block Motion Estimation” *IEEE Transactions on Consumer Electronics*, Vol. 55, No. 4, Nov. 2009, pp. 2154-2162.
6. Mohammed Golam Sarwer, and Q.M. Jonathan Wu, " An Efficient search range Decision Algorithm for Motion Estimation of H.264/AVC, " *International Journal of Circuits, Systems and Signal Processing*, vol. 3, issue 4, pp. 173-180, 2009.

7. Mohammed Golam Sarwer, and Q.M. Jonathan Wu "Adaptive Variable Block-Size Early motion estimation termination algorithm for H.264/AVC Video Coding Standard." *IEEE Transaction on Circuit and System for Video Technology*, volume 19, number 8, August 2009, pp 1196-1201.
8. Mohammed Golam Sarwer, Lai Man Po, Kai Guo and Q.M. Jonathan Wu, "Transform-Domain Rate-Distortion Optimization Accelerator for H.264/AVC Video Coding Standard," International Journal of Signal Processing., vol. 5, no. 3, 2009, pp. 238-248.
9. Mohammed Golam Sarwer, Lai Man Po and Jonathan Wu, "Fast Sum of Absolute Transformed Difference based 4x4 Intra Mode Decision of H.264/AVC Video Coding Standard," Elsevier Journal of Signal Processing: Image Communications, Volume 23, Issue 8, September 2008, Pages 571-580.
10. Mohammed Golam Sarwer, and Lai Man Po "Fast bit rate estimation for mode decision of H.264/AVC" IEEE Transaction of Circuit and System for Video Technology, volume 17, number 10, October 2007, pp 1402-1407.
11. Manoj Datta, Md. Abdur Rafiq, Md. Golam Sarwer and B.C. Ghosh, "An improved induction motor rotor flux estimator based on real time recurrent learning algorithm," Journal of Electrical Engineering, IEB, Bangladesh, vol. EE 33, No 1&II, December 2006, pp-9-14.
12. Md. Abdur Rafiq, Mohammed Golam Sarwer, and B. C. Ghosh. "Fast Speed Response Field-Orientation Control of Induction Motor Drive with Adaptive Neural Integrator." Istanbul University Journal of Electrical and Electronics Engineering, vol. 6, no. 2, pp 229-235, 2006.
13. Mohammed Golam Sarwer, Md. Abdur Rafiq, and B.C. Ghosh. "Chattering Free Neuro Sliding Mode Controller of DC Drives " Journal of Electrical Engineering, The Institution of Engineers, Bangladesh, Vol. EE 32, No. I & II, pp 32-37, December 2005.
14. Md. Abdur Rafiq, Mohammed Golam Sarwer, R. Ahshan, & B.C. Ghosh, "Model reference control of speed sensorless Induction Motor Drive." Journal of Electrical Engineering, Bangladesh, The Institution of Engineers, Vol. EE 32, No. I & II, pp 56-61, December 2005.

15. Mohammed Golam Sarwer, Md. Abdur Rafiq & B.C "Sliding Mode Controller of DC Drive." Journal of Electrical Engineering,The Institution of Engineers, Bangladesh, Vol. EE 31, No. I & II, December 2004, pp 45-49.
16. Mohammed Golam Sarwer, Md. Abdur Rafiq & B.C. Ghosh "Adaptive Fuzzy Artificial Neural Network Based Speed Controller for DC motor Drive." Journal of Electrical Engineering,The Institution of Engineers, Bangladesh, Vol. EE 31, No. I & II, December 2004, pp 20-26.

Articles in Conferences

17. Mohammed Golam Sarwer, Q.M. Jonathan Wu, X. P Zhang "Efficient rate-distortion optimization of H.264/AVC intra coder". Submitted to International Conference on Image Proceeding, 2011
18. Mohammed Golam Sarwer, and Q. M. Jonathan Wu, " Improved DC Prediction for H.264/AVC intra Coding " Accepted in 2011 International Conference on Communication and Electronics Information - ICCEI 2011, Haikou, China
19. Mohammed Golam Sarwer, and Q. M. Jonathan Wu, " Enhanced Low Complex Cost Function for H.264/AVC Intra Mode Decision" Accepted in International Conference on Multimedia and Signal Processing (CMSP'11), Guilin, China.
20. Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "A Novel Bit Rate Reduction Method of H.264/AVC Intra Coding", accepted in the 3rd International Congress on Image and Signal Processing (CISP'10), 16-18 October 2010, Yantai, China.
21. Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "Enhanced Intra Coding of H.264/AVC Advanced Video Coding Standard with Adaptive Number of Modes", 2010 International Conferences on Active Media Technology (AMT 2010), Toronto, Canada, August 2010, pp. 361-372.
22. Mohammed Golam Sarwer, and Q.M. Jonathan Wu, "Adaptive Search Area Selection of Variable Block-Size Motion Estimation of H.264/AVC Video Coding Standard", IEEE International Symposium of Multimedia ISM 2009, pp. 100-105.

23. Mohammed Golam Sarwer, and Q.M. Jonathan Wu, “Fast Block Motion Estimation by edge based partial distortion search”, Proceedings of IEEE International Conference on Image Processing, 2009, (ICIP-2009), Cairo, Egypt, November 7-10, 2009, pp. 1573-1576.
24. Mohammed Golam Sarwer, and Q.M. Jonathan Wu, “Efficient Partial Distortion Search Algorithm for Block based Motion Estimation”, Proceedings of IEEE CCECE 09, pp. 890-893.
25. Mohammed Golam Sarwer, and Q.M. Jonathan Wu, “Region Based Searching for Early Terminated Motion Estimation Algorithm of H.264/AVC Video Coding Standard”, Proceedings of IEEE CCECE 09, pp. 468-471.
26. Thanh Minh Nguyen, Mohammed Golam Sarwer, and Q.M. Jonathan Wu, “A new probability neural network for image classification problem”, Proceedings of the 10th IASTED International Conference SIP 2008, HI, USA, pp. 1-6.
27. Mohammed Golam Sarwer, Thanh Minh Nguyen and Q.M. Jonathan Wu, “Fast Motion estimation of H.264/AVC by adaptive early termination”, Proceedings of the 10th IASTED International Conference SIP 2008, HI, USA, pp. 140-145.
28. Mohammed Golam Sarwer, Lai Man Po, and Jonathan Wu “Complexity Reduced Mode Selection of H.264/AVC Intra Coding.” Proceeding on International Conference on Audio, Language and Image Processing (ICALIP 2008), China, pp.1492-1496.
29. Mohammed Golam Sarwer, and Lai Man Po, “Bit Rate estimation for cost function of 4x4 intra mode decision of H.264/AVC.” Proceeding of IEEE International Conference on Multimedia and Expo (ICME 2007), pp-1579-1582.
30. Manoj Datta, Md. Abdur Rafiq, Mohammed Golam Sarwer, B. C. Ghosh. “An Improved Induction Motor Rotor Flux Estimator Based on Real Time Recurrent Learning Algorithm.” Proceeding of 8th ICCIT 2005 Islamic University of Technology (IUT), Dhaka, Bangladesh, 28-30 December 2005, pp-585-590.
31. Md. Abdur Rafiq, Mohammed Golam Sarwer, Manoj Datta, and B. C. Ghosh. “Genetic Algorithm Based Fast Speed Response Induction Motor Drive With ANN Flux Estimator.” Proceeding of IEEE ICIT 2005, December 14-17, City University of Hong Kong, pp 882-887.

32. Md. Abdur Rafiq, Mohammed Golam Sarwer, Manoj Datta, and B. C. Ghosh. "Fast Speed Response Field-Orientation Control Of Induction Motor Drive With Adaptive Neural Integrator." Proceeding of IEEE ICIT 2005, December 14-17, City University of Hong Kong, pp 610-614.
33. Manoj Datta, Mohammed Golam Sarwer, Md. Abdur Rafiq, B. C. Ghosh. "A High Performance Decoupling Control Scheme for Induction Motor with Modified Adaptive Neural Rotor Flux Estimator." Proceedings of the International Conference on Intelligent Systems 2005 (ICIS 2005) Kuala Lumpur, 1 – 3 December 2005.
34. Manoj Datta, Md. Abdur Rafiq, Mohammed Golam Sarwer, B. C. Ghosh "An Efficient Induction Motor Rotor Flux Estimator Based on Real Time Recurrent Learning Algorithm." Proceedings of the International Conference on Intelligent Systems 2005 (ICIS 2005) Kuala Lumpur, 1 – 3 December 2005.
35. Md. Abdur Rafiq, Manoj Datta, Mohammed Golam Sarwer, and B.C. Ghosh "A New Scheme for Field-Orientation Control Of Induction Motor Drive With Adaptive Neural Flux Estimator." Proceeding of IEEE PEDS 2005, Nov 28-Dec.1, pp 704-709.
36. Mohammed Golam Sarwer, Md. Abdur Rafiq, Manoj Datta, B.C. Ghosh, and S. Komada. "Chattering Free Neuro-Sliding Mode Control of DC Drives." Proceeding of IEEE PEDS 2005, Nov 28-Dec.1, pp 1101-1106.
37. Y. Hamada, N. Nakamura, S. Komada, J. Hirai, and Mohammed Golam Sarwer " Research on determination of distance measurement density for autonomous mobile robot using fuzzy decision-making." Proceeding of 2004 SICE Mie Conference, pp. A15-1-A15-2, December 22, 2004.
38. Mohammed Golam Sarwer, Md. Abdur Rafiq & B.C. Ghosh "Adaptive Fuzzy Artificial Neural Network Based Speed Controller for DC motor Drive." Proceedings of the ICEECE December 22-24, 2003, Dhaka, Bangladesh, pp 237-242

Patents

39. Po Lai Man, and Sarwer Mohammed Golam, "Bit Rate Estimation in Data or Video Compression" US pending patent, filing number: 12/039,904, filing date: 29 Feb, 2008.

List of Scholarships and Awards

1. Ontario Ministry of Research and Innovation postdoctoral fellowship, 2011-2012.
2. **Outstanding Graduate Research Award**, University of Windsor, 2011.
3. Ontario Graduate Scholarship (OGS), 2010-2011.
4. Fredrick Atkins Graduate Award, Department of Electrical and Computer Engineering, University of Windsor, 2010-2011.
5. University of Windsor Doctoral Tuition Scholarship from January 2008 to December 2010.
6. Fredrick Atkins Graduate Award, Department of Electrical and Computer Engineering, University of Windsor, 2009-2010.
7. University of Windsor President excellence Scholarship from summer 2008 to winter 2009.
8. University of Windsor International Excellence Scholarship from January 2008 to December 2009.
9. Awarded for studentship of monthly rate of HK$ 12,860 per month during study period of City University of Hong Kong.
10. Awarded by Prime Minister Gold Medal due to outstanding academic performance in undergraduate level.
11. Awarded from student chapter, Institute of Engineer, Bangladesh (IEB), for brilliant result in undergraduate period.
12. Institute Scholarship throughout the four years for excellent result in undergraduate level.
13. Education Board Scholarship of Comilla board for excellent result in S.S.C