

國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

H. 264/MPEG-4 AVC 移動估測的快速演算法與架構設計

Fast Algorithms and Architecture Designs for H.264/MPEG-4

AVC Motion Estimation

研究生：王裕仁

指導教授：張添烜

中華民國 九十五年 六月

H. 264/MPEG-4 AVC 移動估測的快速演算法與架構設計

**Fast Algorithms and Architecture Designs for H.264/MPEG-4 AVC
Motion Estimation**

研究生：王裕仁
指導教授：張添烜 博士

Student: Yu-Jen Wang
Advisor: Tian-Sheuan Chang

國立交通大學
電子工程學系 電子研究所碩士班



A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics
College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of Requirements

for the Degree of

Master of Science

In

Electrical Engineering

June 2006

Hsinchu, Taiwan, Republic of China

中華民國 九十五年 六月

誌 謝

首先，要感謝我的指導教授—張添烜博士，這兩年來給我的支持和鼓勵，讓我在研究上能自由發揮，每當遇到問題和疑問時總能適時的給予最適當的建議，當意見相左時也往往能夠給予最大的尊重，永遠以鼓勵的態度支持我的想法。張老師的溫文儒雅作風也讓我莽莽撞撞的個性有潛移默化的改變，感激之情，非數十字可以言表。

也要謝謝我的口試委員們，交大電子李鎮宜系主任和清華大學陳永昌教授，感謝你們百忙中抽空來指導我，因為你們寶貴的意見讓我的論文更加完備。

接著要感謝 VSP 實驗室的好伙伴們。謝謝引我入門的鄭朝鐘學長，給予不少建議的張彥中學長、林佑坤學長，你們傳給我的經驗，將讓我受用不盡。感謝古君偉同學，參加 IC 競賽的過程，一起加油打氣，培養出短時間內迅速確實完成設計的能力。感謝蔡旻奇、吳錦木同學，在面對冷澀堅硬的研究之餘，一起興致高昂的出遊踏青，醞釀出下次更具爆發力的創意。感謝余國巨同學、郭子筠學弟，在團體競爭的刺激下，領悟出團體中個人角色如何定位，為踏出社會做先一步的準備。感謝史彥芪學長、林嘉俊、吳私璟、廖英澤、李得瑋學弟們，有你們的陪伴，我的碩士班生涯充滿了歡笑。所有的一切，都是我在交大寶貴的回憶。

最後要感謝默默支持我的家人們，我的爸媽、哥哥、姊姊，你們的溫暖是我努力最大的支柱。

在此，把本論文獻給所有愛我與所有我愛的人。

H. 264/MPEG-4 AVC 移動估測的快速演算法與架構設計

研究生：王裕仁

指導教授：張添烜 博士

國立交通大學電子研究所碩士班

摘 要

隨著高解析數位電視時代的來臨，為了兼顧大且精緻的畫面，高壓縮率規格(H. 264)是我們現行的解決方案。它不僅可有效節省儲存媒體所需的空間，同時也可在現行的通訊環境下允許傳輸更高解析的畫面。伴隨著種種好處而來的就是極之龐大的運算量，而大量的快速演算法也因此應運而生。如何兼顧畫質和運算速度成了當前最重要的課題，而這也是本篇論文的主旨。

根據已出版的文獻，位移估測是整個壓縮過程中最為費時的。更進一步去了解這個部份，我們可以把牠大致上分為整數位移估測和分數位移估測。在原始演算法的條件下，由於搜尋範圍較大整數位移估測佔去了絕大部分的時間。因此我們非常直覺的認為，若能大幅減少搜尋範圍又能使畫質維持差不多水準將可以有效節省壓縮時程，我們提出的快速演算法能夠針對不同解析畫面達到 88% (352 x 288) 和 75% (720 x 480) 的節省。分數位移估測在原始演算法的架構下，由於搜尋點數遠少於整數位移估測所以在整個壓縮的過程中並沒有決定性的影響。但隨著整數位移估測快速演算法的發展，分數位移估測搜尋點數所佔的比例慢慢升高，分數位移估測快速演算法也愈來愈有存在的必要性。在單一樣式錯誤表面的假設下，我們利用特定點的錯誤數值去預測整個搜尋視窗的錯誤表面。除此之外，我們也引進了提前終止的技術。此分數快速位移估測部分可以減少超過 50% 的運算量。在整數和分數位移估測同時使用快速演算法的情形下，以 1280 x 720 為測試解析度，我們可以加速總壓縮時間達 20 倍之鉅。另外一種常見的解決方式是利用硬體平行化同時處理多筆資料以達到加速的目的。在分數位移估測方面，拜快速演算法之賜，我們的架構可以減少將近 40% 面積和加速 14%。

Fast Algorithms and Architecture Designs for H.264/MPEG-4 AVC Motion Estimation

Student: Yu-Jen Wang

Advisor: Tian-Sheuan Chang

Institute of Electronics
National Chiao Tung University

Abstract

With modern day advances in computer processing and multimedia applications, improvements in the area of image processing and video compression are analogous. Video compression allows the reduction of high-resolution video into a more compact memory space to thereby reduce storage and video processing resources during playback.

According to the literature published before, we can find that the motion estimation process is the most time consumed part. To further realize this process, we can mainly divide it into two parts: integer motion estimation and fractional motion estimation. Integer motion estimation cost most part of time under the original algorithm unchanged. The main reason is that the search window is too large. So we have a very simple idea that we want to decrease the search window. We can reduce 88% (input sequence as CIF size) and 75% (input sequence as D1 size) search points respectively. Fractional motion estimation will not affect obviously under the original condition. But when the fast algorithm is applied for integer motion estimation, the portion of encoding time due to fractional motion estimation is getting larger. Based on the assumption of uni-modal error surface, we want to use the results of half pixel step to predict the slope of error surface. We also apply early termination technique. We can get 50% search points reduction in this part. By applying both fast algorithms, we get 20 times speed up with the input sequence size as 1280 x 720. Making use of hardware parallelism to speed up is also a common method in H.264 research field. By the benefit of applying fast fractional motion estimation algorithm, we decrease 40% area and speed up by 14% in our fast fractional motion estimation architecture.

Contents

Chapter 1 Introduction.....	1
1.1 THE SCENE	1
1.2 VIDEO COMPRESSION	2
1.3 MPEG-4 AND H.264.....	3
1.4 INTRODUCTION	4
1.5 MOTIVATION.....	6
1.6 THESIS ORGANIZATION	7
Chapter 2 Overview of H.264/AVC standard	8
2.1 OVERVIEW	8
2.1.1 Variable block-size motion compensation with multiple references	8
2.1.2 Directional spatial intra coding.....	8
2.1.3 In-loop deblocking filter.....	8
2.1.4 Context adaptive entropy coding.....	8
2.1.5 Computational profile.....	10
2.2 INTRA PREDICTION.....	11
2.2.1 Overview	11
2.2.2 Fast algorithms	11
2.3 INTER PREDICTION	14
2.3.1 Overview	14
2.3.2 Fast algorithms	15
2.4 FAST MODE DECISION.....	20
2.4.1 Overview	20
2.4.2 FAST ALGORITHMS.....	20
Chapter 3 Dynamic search range prediction for integer motion estimation	22
3.1 DESCRIPTION OF PRIOR ART	22

3.2 ANALYSIS OF INTEGER MOTION VECTOR.....	22
3.3 PROPOSED ALGORITHM	26
3.4 COMPARISON.....	28
3.5 SIMULATION RESULT	29
Chapter 4 Adaptive search pattern prediction for fractional motion estimation.....	33
4.1 ANALYSIS OF FRACTIONAL PEL MOTION VECTOR.....	33
4.2 ORIGINAL SEARCH ALGORITHM	35
4.3 PROPOSED ALGORITHM	37
4.4 COMPLEXITY AND ACCURACY COMPARISON	42
4.5 EARLY TERMINATION.....	44
4.6 SIMULATION RESULT	45
4.7 COMPARISON.....	46
Chapter 5 Integration	47
Chapter 6 Architecture design for fast sub-pel inter coding in H.264.....	48
6.1 HARDWARE CONSIDERATION	48
6.2 ALGORITHM FOR HARDWARE MODIFICATION	50
6.3 ARCHITECTURE	51
6.4 PERFORMANCE ANALYSIS	55
Chapter 7 Conclusion	57
7.1 SUMMARY	57
7.1.1 Fast integer motion estimation	57
7.1.2 Fast fractional motion estimation	57
7.1.3 Architecture design of fractional motion estimation.....	57
7.2 PERFORMANCE ANALYSIS	58
7.2.1 Fast integer motion estimation	58
7.2.2 Fast fractional motion estimation	58

7.2.3 Architecture design of fractional motion estimation..... 58

7.3 FUTURE WORK..... 58

BIBLIOGRAPHY 59

APPENDIX..... 62



List of Figures

Fig 1 Block diagram of H.264 encoder	9
Fig 2 Computational profile of H.264 video encoding	10
Fig 3 Intra prediction modes for (a)Intra_4x4 and (b) Intra_16x16.....	11
Fig 4 (a) S. Zhu and K.-K. Ma, “A new diamond search algorithm for fast block matching motion estimation”,[19]. (b) C. Zhu, X. Lin,and L.-P. Chau, “Hexagon-based search pattern for fast block motion estimation,[21]....	15
Fig 5 Temporal Neighboring Ref-frame Prediction.....	16
Fig 6 Spatial Up-Layer Prediction	16
Fig 7 (a)H.-M. Wong, O. C Au, and A. Chang, “Fast sub-pixel inter-prediction–based on the texture direction analysis”, [35](b) C.-C. Cheng, Y.-J. Wang, and T.-S. Chang, “A fast fractional pel motion estimation algorithm for H.264/AVC”,[36].	18
Fig 8 Comparison of bit stream portion with different fast algorithm.	19
Fig 9 Correlation between critical search range and matching error.....	24
Fig 10 rate distortion curve with CIF size and search range =16.....	32
Fig 11 rate distortion curve with D1 size and search range =64.....	32
Fig 12 (a) Error surface of integer pel ME (search range: 32); (b) Error surface of fractional pel ME (1/8-pel case)	33
Fig 13 Distribution of the fractional ME.....	34
Fig 14 Search algorithm in reference software.....	35
Fig 15 Proposed algorithm for half-pel.	37
Fig 16 Proposed algorithm for quarter-pel (case 1).....	39
Fig 17 Proposed algorithm for quarter-pel (case 2).....	40
Fig 18 Proposed algorithm for quarter-pel (case 3).....	41
Fig 19 Proposed algorithm for quarter-pel (case 4).....	42
Fig 20 Relationship between best SAD in integer part & best SATD in fractional part	44
Fig 21 Adaptive threshold prediction curve	45

Fig 22 Mode decision flow in H.264.....	49
Fig 23 Block diagram of fast FME hardware.....	51
Fig 24 (a) 4X4 block PU (b) 6-tap 1-D FIR filter.....	52
Fig 25 Interpolation unit.....	53
Fig 26 Bilinear filters of interpolation unit.....	54
Fig 27 Pattern 1.....	63
Fig 28 Pattern 2.....	64
Fig 29 Pattern 3.....	65
Fig 30 Pattern 4.....	66
Fig 31 Pattern 5.....	67
Fig 32 Pattern 6.....	68
Fig 33 Pattern 7.....	69
Fig 34 Pattern 8.....	70
Fig 35 Pattern 9.....	71



List of Tables

Table 1 Increasing percentage of motion vector predictor with different search range.....	23
Table 2 Increasing percentage of motion vector difference with different search range.....	23
Table 3 The correlation between search range and the factors including matching error and motion information. Critical search range means the smallest search range with similar RD performance.....	24
Table 4 saving statistic with input search range = 16 and input sequence size as CIF size.....	28
Table 5 saving statistic with input search range = 32 and input sequence size as CIF size.....	28
Table 6 saving statistic with input search range = 32 and input sequence size as D1 size.....	29
Table 7 saving statistic with input search range = 64 and input sequence size as D1 size.....	29
Table 8 rate distortion result with input search range = 16 and input sequence size as CIF size.....	30
Table 9 rate distortion result with input search range = 32 and input sequence size as CIF size.....	30
Table 10 rate distortion result with input search range = 32 and input sequence size as D1 size.....	31
Table 11 rate distortion result with input search range = 64 and input sequence size as D1 size.....	31
Table 12 performance comparison.....	32
Table 13 Search point comparisons for different algorithms.....	43
Table 14 Algorithms prediction correctness compare to full search algorithm.....	43
Table 15 Simulation result when QP = 28, speed up is only the performance in fractional ME part. RDO is off, reference frame number = 1, CIF.....	46
Table 16 Comparison between different fast algorithms for fractional ME.....	46
Table 17 rate distortion result with input search range = 64 and input sequence size as 720p size.....	47
Table 18 Simulation result when QP = 28, point stop means the early termination applied in every search point, step stop means the early termination just applied in half step.....	50
Table 19 Performance analysis after algorithm modification.....	55
Table 20 Implementation result of proposed architecture.....	55
Table 21 Comparison between the proposed architecture and architecture in [51].....	56

Chapter 1 Introduction

1.1 THE SCENE

Pervasive, seamless, high quality digital video has been the goal of companies, researchers and standards bodies over the last two decades. In some areas (for example broadcast television and consumer video storage), digital video has clearly captured the market (such as videoconferencing, video email, mobile video), market success is perhaps still too early to judge. However, there is no doubt that digital video is a globally important industry which will continue to pervade businesses, networks and homes. The continuous evolution of the digital video industry is being driven by commercial and technical forces. The commercial drive comes from the huge revenue potential of persuading consumers and businesses:

1. Replace analogue technology and older digital technology with new, efficient, high quality digital video products.
2. Adopt new communication and entertainment products those have been made possibly by the move to digital video.

The technical drive comes from continuing improvements in processing performance, the availability of higher capacity storage and transmission mechanisms and research and development of video and image processing technology.

Getting digital video from its source (a camera or a stored clip) to its destination (a display) involves a chain of components or processes. Keys to this chain are the processes of compression (encoding) and decompression (decoding), in which bandwidth-intensive 'raw' digital video is reduced to a manageable size for transmission or storage, then reconstructed for display. Getting the compression and decompression processes 'right' can give a significant technical and commercial edge to a product, by providing better image quality, greater reliability and more flexibility than competing solutions. There is therefore a keen interest in the continuing development and improvement of video compression and decompression methods and systems. The interested parties include entertainment, communication and broadcasting companies, software and hardware developers, researchers and holders of potentially lucrative patents on new compression algorithms.

The early successes in the digital video industry (notably broadcast digital television and DVD-video) were underpinned by international standard ISO/IEC 13818 [1], popularly known as ‘MPEG-2’ (after the working group that developed the standard, the Moving Picture Experts Group). Anticipation of a need for better compression tools has led to the development of two further standards for video compression, known as ISO/IEC 14496 Part 2 (MPEG-4 Visual) [2] and ITU-T Recommendation H.264/ISO/IEC14496 Part 10 (H.264) [3]. MPEG-4 Visual and H.264 share the same ancestry and some common features (they both draw on well-proven techniques from earlier standards) but have notably different visions, seeking to improve upon the older standards in different ways. The vision of MPEG-4 Visual is to move away from a restrictive reliance on rectangular video images and to provide an open, flexible framework for visual communications that uses the best features of efficient video compression and object-oriented processing. In contrast, H.264 has a more pragmatic vision, aiming to do what previous standards did (provide a mechanism for the compression of rectangular video images) but to do it in a more efficient, robust and practical way, supporting the types of applications that are becoming widespread in the marketplace (such as broadcast, storage and streaming).

1.2 VIDEO COMPRESSION

Network bit rates continue to increase (dramatically in the local area and somewhat less so in the wider area), high bit rate connections to the home are commonplace and the storage capacity of hard disks, flash memories and optical media is greater than ever before. With the price per transmitted or stored bit continually falling, it is perhaps not immediately obvious why video compression is necessary (and why there is such a significant effort to make it better). Video compression has two important benefits. First, it makes it possible to use digital video in transmission and storage environments that would not support uncompressed raw video. For example, current internet throughput rates are insufficient to handle uncompressed video in real time (even at low frame rates or small frame size). A Digital Versatile Disk (DVD) can only store a few seconds of raw video at television quality resolution and frame rate, so DVD video storage would not be practical without video and audio compression. Second, video compression enables more efficient use of transmission and storage resources. If a high bit rate transmission channel is available, then it is more attractive proposition to send high resolution compressed video or multiple compressed video channels than to send a single, low resolution, uncompressed stream. Even with constant advances in storage and

transmission capacity, compression is likely to be an essential component of multimedia services for many years to come.

An information carrying signal may be compressed by removing redundancy from the signal. In a lossless compression system statistical redundancy is removed so that the original signal can be perfectly reconstructed at the receiver. Unfortunately, at the present time lossless methods can only achieve a modest amount of compression of image and video signals. Most practical video compression techniques are based on lossy compression, in which greater compression is achieved with the penalty that the decoded signal is not identical to the original. The goal of a video compression algorithm is to achieve efficient compression whilst minimizing the distortion introduced by the compression process.

Video compression algorithms operate by removing redundancy in the temporal, spatial frequency domain. The human eye and brain (Human Visual System) are more sensitive to lower frequencies. By removing different types of redundancy (spatial and temporal) it is possible to compress the data significantly at the expense of a certain amount of information loss (distortion). Further compression can be achieved by encoding the processed data using an entropy coding scheme such as Huffman coding or Arithmetic coding.

Image and video compression has been a very active field of research and development for over twenty years and many different systems and algorithms for compression and decompression have been proposed and developed. In order to encourage inter-working, competition and increased choice, it has been necessary to define standard methods of compression encoding and decoding to allow products from different manufacturers to communicate effectively. This has led to the development of a number of key International Standards for image and video compression, including the JPEG, MPEG and H.26X series of standards.

1.3 MPEG-4 AND H.264

MPEG-4 Visual and H.264 (also known as Advanced Video Coding) are standards for the coded representation of visual information. Each standard is a document that primarily defines two things, a coded representation (or syntax) that describes visual data in a compressed form and a method of decoding the syntax to reconstruct visual information. Each standard aims to ensure that compliant encoders and decoders can successfully inter-work with each other, whilst allowing

manufacturers the freedom to develop competitive and innovative products. The standards specially do not define an encoder; rather, they define the output that an encoder should produce. A decoding method is defined in each standard but manufacturers are free to develop alternative decoders as long as they achieve the same result as the method in the standard.

MPEG-4 Visual and H.264 have related but significantly different visions. Both are concerned with compression of visual data but MPEG-4 Visual emphasizes flexibility whilst H.264's emphasis is on efficiency and reliability. MPEG-4 Visual provides a highly flexible toolkit of coding techniques and resources, making it possible to deal with a wide range of types of visual data including rectangular frames (traditional video material), video objects (arbitrary-shaped regions of a visual scene), still images and hybrids of natural (real-world) and synthetic (computer-generated) visual information. MPEG-4 Visual provides its functionality through a set of coding tools, organized into 'profiles', recommended groupings of tools suitable for certain applications. Classes of profile include 'simple' profiles (coding of rectangular video frames), object-based profiles (coding of arbitrary-shaped visual objects), still texture profiles (coding of still images or texture), scalable profiles (coding at multiple resolutions or quality levels) and studio profiles (coding for high quality studio applications).

In contrast with the highly flexible approach of MPEG-4 Visual, H.264 concentrates specifically on efficient compression of video frames. Key features of the standard include compression efficiency (providing significantly better compression than any previous standard), transmission efficiency (with a number of built-in features to support reliable, robust transmission over a range of channels and networks) and a focus on popular applications of video compression. Only three profiles are currently supported (in contrast to nearly 20 in MPEG-4 Visual), each targeted at a class of popular video compression applications. The Baseline profile may be particularly useful for 'conversational' applications such as video conferencing, the extended profile adds extra tools that are likely to be useful for video streaming across networks and the Main profile includes tools that may be suitable for consumer applications such as video broadcast and storage.

1.4 INTRODUCTION

With modern day advances in computer processing and multimedia applications, improvements in the area of image processing and video compression are analogous.

Video compression allows the reduction of high-resolution video into a more compact memory space to thereby reduce storage and video processing resources during playback. Reduced memory requirements for video footage can aid in lengthy video segments being stored onto portable media to and improve the mobility and transferability of large files. Bandwidth is also increased when performing file transfers, as quicker download and upload times are achieved through Internet and other transfer protocols.

Videos are produced through a series of different frames (or images) played in sequence. Therefore, the area of video compression reduces down to specialized forms of image compression with specific consideration for video playback. The art of video compression tends to fall into one of two categories: lossless compression and lossy compression. Lossy compression entails the reduction of certain finer image details that are sacrificed for the sake of saving a little more bandwidth or storage space. Lossless compression, on the other hand, involves compressing data such that it will be an exact replica of the original data upon decompression. For many types of binary data, such as documents and various programs, lossless compression is required as the integrity of the original data needs to be preserved. Many types of multimedia, on the other hand, need not be reproduced exactly as before. An approximation of the original image is usually sufficient for most purposes, as long as the error between the original and the compressed image is tolerable.

In performing lossy compression, a common technique is to remove redundant information between adjacent frames to reduce memory constraints and increase bandwidth. This technique is referred to as motion estimation (ME), of which H.264 and MPEG-4 are the current known standards. These standards exploit and remove temporal redundancies between successive frames, or more simply, select a reference frame and predict subsequent frames based on the reference frame. Motion estimation makes the assumption that the objects in the scene solely possess translational motion. This assumption holds as long as there is no pan, zoom, changes in luminance, or rotational motion. Motion estimation is an intensive process which generally consumes 60-90% of the computational time of a related encoder or micro-controller.

The ME process begins first by dividing the current frame into macroblocks. The size of a macroblock is typically 16x16 pixels, but can vary for each ME technique according to the desired tradeoff between resolution and computational cost. Each macroblock of a current frame is compared to a macroblock of a reference frame by calculating a cost value for selected search points of the macroblocks. A current

macroblock that is sufficiently similar reference macroblock is then selected and paired together. Vectors denoting a displacement between each matching reference macroblock and each matching current macroblock are then determined. These vectors are known as motion vectors, and serve as a representation of the displacement between matching macroblocks from the reference frame to the current frame for use in the prediction process.

Using the reference frame and motion vectors, one can now reconstruct an approximation of the current frame (now the reconstructed frame) by copying the matching reference macroblock of the reference frame to the location noted by the corresponding motion vectors. This form of image reconstruction is also known as motion compensation. In this manner, subsequent frames can be continually predicted, without having to store redundant macroblocks from a current frame into memory. Certain macroblocks from the reconstructed frame are simply produced from a matching macroblock from a reference frame according to a motion vector. This process therefore compresses video sizes by omitting the storage of redundantly used macroblocks. The level of compression varies with the number of macroblocks replaced from frame to frame, and the desired image resolution.

The matching process in ME entails comparing selected pixels from a current macroblock with the same pixels from a reference macroblock using a cost function. A search algorithm provides the selection of search points indicating which pixels are to be used for comparison in the matching process. The cost function provides a value indicating the degree of similarity between the compared search points. One of the more common cost functions to determine the similarity between two input images includes the sum of absolute differences (SAD). The greater the similarity between the two inputs, the smaller the SAD value will result. The matching process in ME therefore uses a cost function to compare search points of a current macroblock to search points of a reference macroblock to determine the degree of similarity between the two macroblocks. If the cost values between the two macroblocks are sufficiently low, then the reference macroblock is suitable to replace the current macroblock in motion estimation.

1.5 MOTIVATION

According to the literature published before, we can find that the motion estimation process is the most time consumed part. To further realize this process, we can mainly divide it into two parts: integer motion estimation and fractional motion

estimation. Integer motion estimation cost most part of time under the original algorithm unchanged. The main reason is that the search window is too large. So we have a very simple idea that we want to decrease the search window. Reducing search range is the most effective way to decrease search window and memory accesses can be saved significantly. This is the main reason why we choose the way but other methods such as search pattern rearrangement. Fractional motion estimation will not affect obviously under the original condition. But when the fast algorithm is applied for integer motion estimation, the portion of encoding time due to fractional motion estimation is getting larger. Based on the assumption of uni-modal error surface, we want to use the results of half pixel step to predict the slope of error surface. We also apply early termination technique. Due to the unchanged system order, we use the information from integer part to predict the threshold of fractional part. Making use of hardware parallelism to speed up is also a common method in H.264 research field. To trade off between speed and area, we use certainly parallelism and decompose variable block size into 4X4. In the topic of speed up, we reach the goal by applying early termination technique.

1.6 THESIS ORGANIZATION

In the thesis, we will introduce the H.264 standard and some published algorithms in chapter2. In integer motion estimation part, we develop fast algorithm as dynamic search range prediction. We will detail it in chapter3. In fractional motion estimation part, fast algorithm named as adaptive search pattern prediction is described in chapter4. The co-simulation result by applying both fast algorithms mentioned in chapter3 and chapter4 is shown in chapter5. Then, we will show the hardware architecture and result comparisons in chapter6. Finally, a conclusion is given in chapter7.

Chapter 2 Overview of H.264/AVC standard

2.1 OVERVIEW

H.264 consists of a number of tools. Its basic structure is the so-called motion-compensated transform coder. Compared to the prior video coding standards, many important and new techniques are employed in H.264 and they together bring significant improvement on coding performance. Some of these techniques are highlighted here [5]. We may want to add that the concepts of some of these tools have existed for some time but they are nicely tuned and integrated together to form a good compression scheme in H.264.

2.1.1 Variable block-size motion compensation with multiple references

The basic unit in H.264 motion estimation is the 16x16 macroblock. It can be further split into a tree structure, with a minimum motion compensation block size as small as 4x4. Also, up to five reference frames may be used for motion compensation.

2.1.2 Directional spatial intra coding

To reduce the correlation inside a block, H.264 adopts the intra-prediction technique, which estimates the current block pixel values based on the known pixels of its neighbor blocks. The prediction results implicitly follow the edge direction, and often bring significant improvements.

2.1.3 In-loop deblocking filter

Block-based video coding produces artifacts known as blocking artifacts at low bit rates. This in-loop deblocking filter adjusts its filter strength adaptively according to the image local characteristics, and thus it provides better quality pictures at the decode end.

2.1.4 Context adaptive entropy coding

Two entropy coding methods, Context-based Adaptive Binary Arithmetic Coding (CABAC) and Context-based Adaptive Variable Length Coding (CAVLC), are provided in H.264. Both methods use context-base adaptivity to improve the entropy

coding performance and the results show this approach is quite successful.

A simplified encoding flow of H.264 is shown in Fig 1. A video frame is first partitioned into a number of 16x16 macroblocks. Then, each macroblock goes through the intra-prediction or the inter-prediction unit. The intra prediction unit uses the neighboring block data to predict the current block. The inter-prediction uses reference frames to predict the current frame. Each predictor has a number of modes. A good design should pick up the best mode with the lowest rate and distortion. The prediction residuals are then transformed, quantized and further entropy-coded into the output bitstream. In order to continue operating on the next incoming frame, the quantized current frame is reconstructed and stored. The decoder data flow is the reverse of the encoder flow.

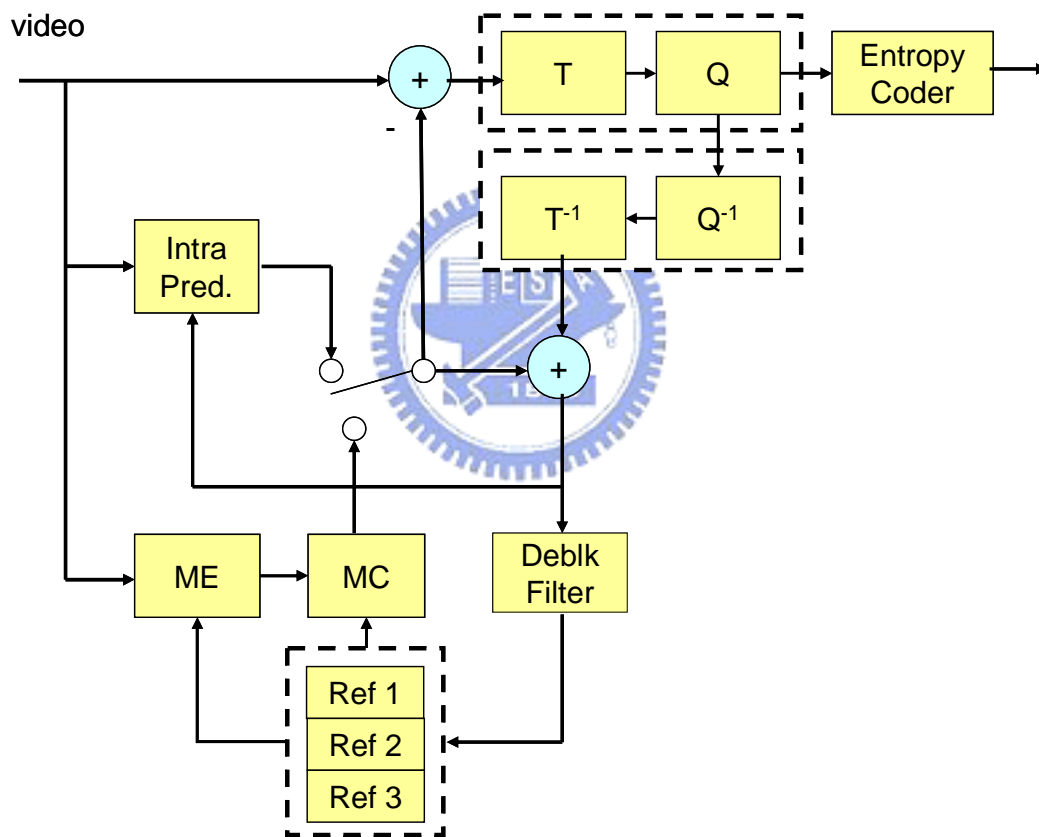


Fig 1 Block diagram of H.264 encoder

2.1.5 Computational profile

The H.264 encoder reference software provided by the ITU/MPEG standard committee is known for its high computational complexity. A typical computational profile of the H.264 encoder (ITU/MPEG reference software) running on Intel PC, is shown in Fig 2. It shows that the tools of (a) motion estimation, (b) entropy coding, (c) transform and quantization, (d) interpolation, and (e) mode decision and intra-prediction are the most time-consuming modules. Although the other results of profiling would have somewhat different, by and large, the trend is pretty much the same. As for the decoder, the tools of (a) motion compensation (including interpolation), (b) entropy decoding, and (c) intra-prediction have the CPU load.

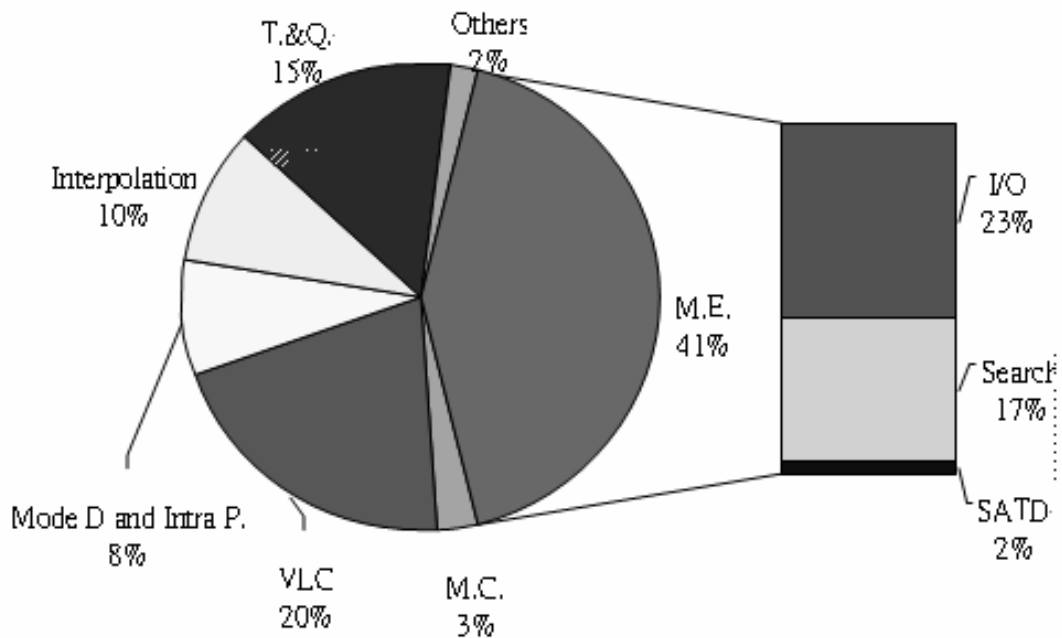


Fig 2 Computational profile of H.264 video encoding.

2.2 INTRA PREDICTION

2.2.1 Overview

Intra-prediction uses the high correlation property of neighboring samples in spatial domain to predict the current encoded samples. For the luma samples, each prediction block may be formed for each 4x4 block (denoted as I4MB) or for an entire MB (denoted as I16MB). When utilizing Intra_4x4 prediction, each 4x4 block chooses one of the nine prediction modes, which include one DC mode plus eight directional prediction modes, as shown in Fig 3 (a), as the best one. In the luma component of an MB, the Intra_16x16 prediction is typically chosen for smooth image areas, and thus, only four prediction modes are specified as shown in Fig 3 (b) except for the DC mode. The chroma samples of an MB are predicted using a similar prediction pattern, Intra_8x8, which is similar to the luma Intra_16x16 prediction.

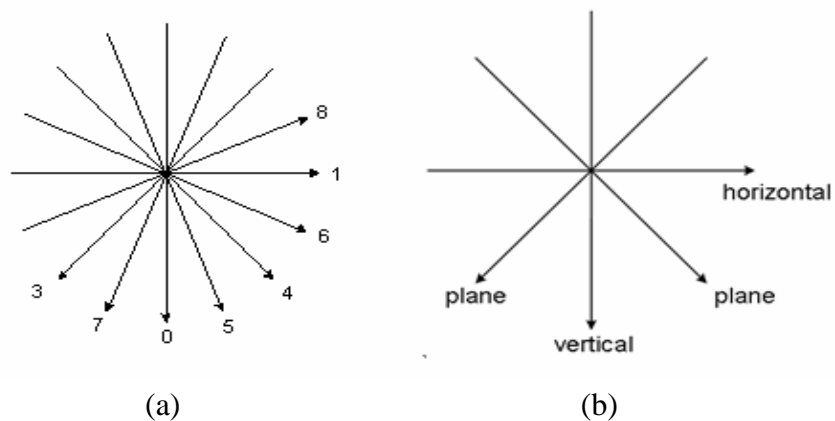


Fig 3 Intra prediction modes for (a) Intra_4x4 and (b) Intra_16x16.

2.2.2 Fast algorithms

The fast algorithms of intra prediction can be classified into several types. The first approach is “early termination”, which ends the search operation when the calculated distortion is smaller than a pre-chosen threshold. The selection of a proper measure for deciding termination is critical to the performance. It may be derived based on the macroblock smoothness [6][7] or the most probable mode [8]. The early termination based on the macroblock smoothness calculates a smoothness measure of a macroblock to determine the block type. For example, the large block type such as Intra_16x16 is chosen often for the flat image areas [6][7]. “Smooth” means that all the pixel values in a MB are similar; that is, their variance is small. The variance computation shall be simple to save computation. Therefore, the Mean Absolute Difference (MAD) operation [6] or the AC/DC ratio [7] is often used. If the variable is

smaller than a pre-selected threshold value, the Intra_16x16 mode is chosen and thus the costly Intra_4x4 can be skipped.

Another kind of early termination proposal examines the most probable mode first. For example, in searching for the best Intra_4x4 mode, if its residual is smaller than a threshold, then the other eight Intra_4x4 modes are skipped (not chosen). Otherwise, all nine modes have to be tested. Then, we set another threshold to decide whether to keep on checking the Intra_16x16 prediction or not. It was reported that in one case, this method together with the 2:1 downsampling and rate-distortion optimization (RDO) can reduce 68.8% of total computation time with only 1.35% of bit rate increase comparing to the reference software [8]. The major issue in this type of algorithms is how to determine the threshold. The threshold value can be adjusted according to the quantization parameters for instance. To construct a more efficient scheme, we propose a mixed fast intra prediction algorithm. It first examines both the most probable mode and the DC mode to determine if it meets the early termination criterion. The threshold value is decided by the average of SATD (sum of absolute transformed difference) of all the previous Intra_4x4 blocks in this frame. Once the 16 Intra_4x4 blocks are done, their total cost will be used as the threshold for deciding Intra_16x16 mode. These threshold values seem to be able to match the video local characteristics and provide good results. Even when RDO is turned off, we can achieve around 30% computational savings for the intra prediction module.

The second approach uses the edge analysis to quickly identify the edge direction since the intra prediction is basically a directional prediction [9][10]. Often the Sobel operators or the first order derivative are used as the edge analysis tool to find the most probable edge, which will be used as one of the final edge candidates. The final mode candidate list includes the one selected by the edge detector together with the other highly probable modes. In the case Intra_4x4, this would mean two modes of the neighboring blocks and the DC mode; and in the Intra_16x16 and Intra_8x8 cases, only the DC mode is considered highly probable. Therefore, only four candidate modes (for Intra_4x4) or two candidate modes (other types) are needed to be examined. The result shows that 60% of intra_only computation time reduction is observed with RDO and the bit rate increase is around 2~3% [9]. The bit rate increase may be owing to the irregular edges within a block. On the other side, the extra computation needed for edge analysis can be a computation burden and reduce the overall saving significantly.

The third approach uses the so-called three step approach [11]. It first tests the

horizontal and vertical directions, it then tests the neighboring 22.5 degree modes close to the better one from the previous step, and finally the best mode up-to-now is checked against the DC mode for the final winner. This approach has the advantage of a fixed number of modes are examined for all cases. However, computation time reduction is around 33% with about 1% bit rate increase.

The last approach makes use of the correlation in the temporal domain [12] since the best prediction mode in the current macroblock is likely similar to that in the reference macroblock in the previously coded frame(s). Thus, the primary intra prediction mode is selected from the mode of the most overlapped block in motion estimation. The computational overhead is nearly zero since all information is obtained during the inter-prediction operation. It is reported that the coding performance is nearly unchanged while the computational savings is about 50% assuming the intra-frame period is 10 [12].

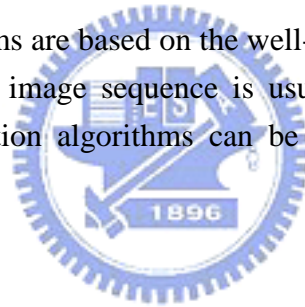
In summarizing various fast intra-prediction algorithms, although we cite the experimental results from the proposed documents, a fair comparison among all methods is difficult because their simulation environments are quite different. One important element affecting computation is the option of RDO in the reference software. This is particularly true for the early termination method with thresholds. The algorithms described in the above can be combined together to achieve further speed-up. For example, the first step could be the decision on Intra_4x4 or Intra_16x16. The second step could be the early termination for the chosen intra type. Finally, the rest of mode tests could be a fast algorithm to select one from the nine or four candidate modes.

2.3 INTER PREDICTION

2.3.1 Overview

Block matching based motion estimation and compensation is a fundamental process in the current international video compression standards. It can efficiently remove interframe redundancy. A direct implementation is the full search algorithm that examines exhaustively every candidate motion vector in the search window to find the globally best matched block in the reference frame. However, its computationally intensive nature prevents it from practical implementation on a processor for real-time applications. The computation burden is increased drastically for the H.264 encoder because there are a number of combinations of partitioning a macroblock into sub-block(s) ranging from 4x4 to 16x16. Potentially each sub-block can have its own motion vector. This feature significantly increases the computational complexity in motion estimation. Thus, many fast motion estimation algorithms have been proposed to alleviate the computational load.

Most of the fast algorithms are based on the well-known a priori knowledge, “the motion field of a real world image sequence is usually gentle, smooth and varies slowly”. Fast motion estimation algorithms can be categorized into roughly three families as described below.



2.3.2 Fast algorithms

2.3.2.1 Reduce possible candidate points

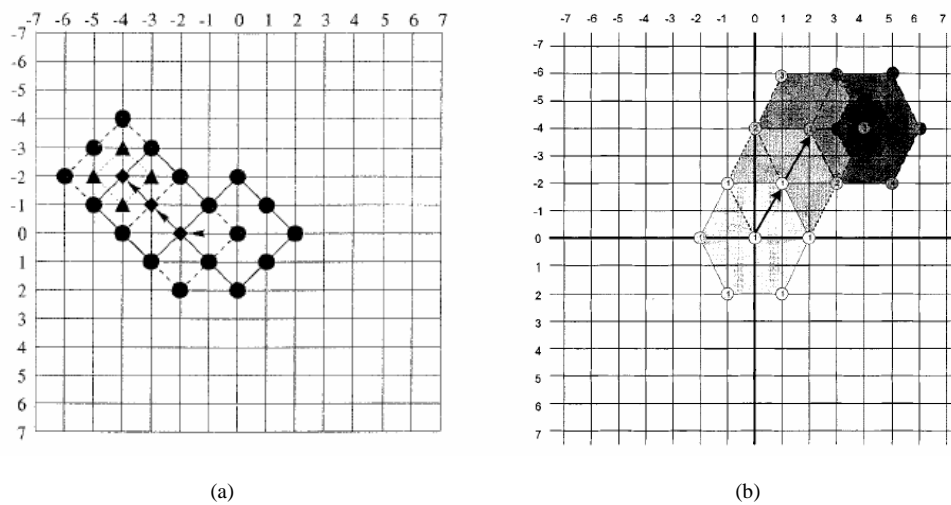


Fig 4 (a) S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation",[19]. (b) C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation,[21]

Based on the assumption of convexity of the uni-modal error surface, i.e., block matching distortion increases monotonically away from the global minimum point, many gradient-based search methods with carefully designed search patterns have been developed to limit search points to a small subset of all possible candidates. This category includes the well-known three-step search (3SS) [13], the new three-step search (N3SS) [14], the cross search (CS) [15], the one-dimensional gradient descent search (1DGDS) [16], the block-based gradient descent search (BBGDS) [17], the four-step search (4SS) [18], the diamond search (DS) [19], the cross-diamond search (CDS)[20] and the hexagon-based search (HEXBS) [21]. Although this category of algorithms may be trapped into a local minimum point and hence the efficiency of the motion compensation may drop, they can considerably reduce the number of block matching computations.

2.3.2.2 Motion vector prediction

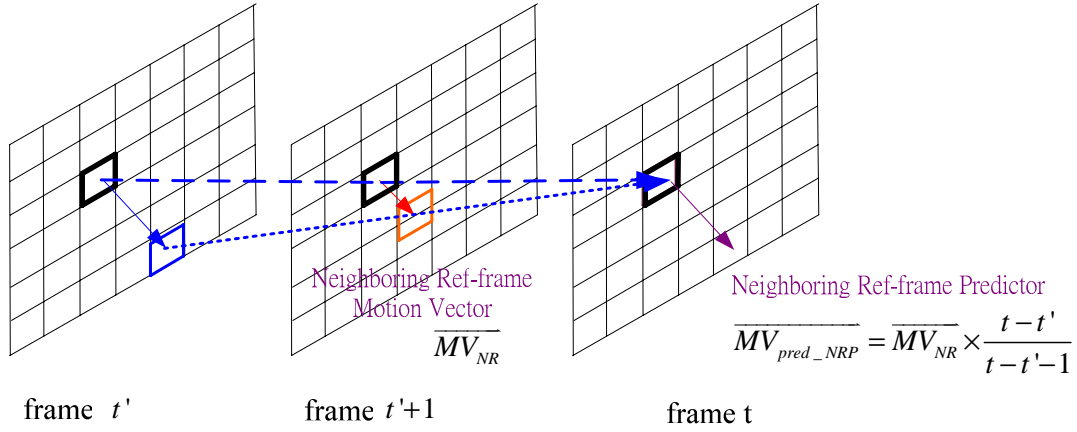


Fig 5 Temporal Neighboring Ref-frame Prediction

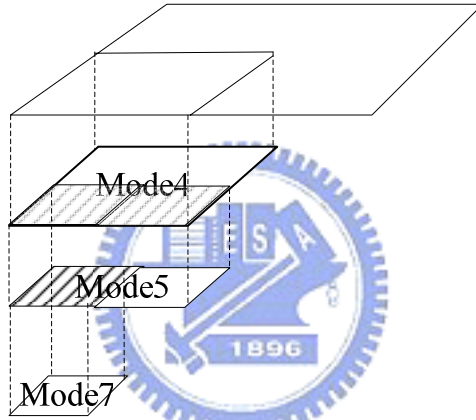


Fig 6 Spatial Up-Layer Prediction

Motion in most natural image sequences involves a few blocks and lasts for a few frames. Therefore, spatially or temporally adjacent blocks often have similar motion vectors. Taking the advantage of the correlation among neighboring motion vectors, the search window can be constrained to a small clique surrounding the “predicted vector”, a candidate position predicated based on the known neighboring motion vectors. Many prediction algorithms have been developed with different complexities. The prediction search algorithm (PSA) [22] simply predicts the current block motion vector as the mean value of its neighboring blocks’ motion vectors. Fuzzy search [23] applies fuzzy logic to predict the motion vector. In [24], motion vectors are predicted by integral projections. In [25], a spatial-temporal AR model of motion vectors is constructed and an adaptive Kalman filter is employed. The multi-resolution search [26] down-samples a picture to obtain raw motion vectors at different resolution levels, then it estimates finer motion vectors from the coarser ones. The multi-resolution-spatiotemporal (MRST) scheme [26] modifies the normal

raster scan order so that some blocks can reference more motion information by increasing their neighboring blocks along more directions. It then combines a multi-resolution scheme and spatiotemporal correlation to predict motion vectors. For burst motions and blocks at the top-left corner, which has little correlation information, the performance of this category of algorithms may deteriorate because the refinement of prediction is restricted to a small search region. Moreover, the prediction overhead may reduce the speed gain.

2.3.2.3 Low complexity block matching criteria

The majority of the computations in motion estimation originate from computations of block matching distortion. In general, block matching metrics, such as the mean absolute difference (MAD) and the mean square error (MSE), involve pixel-wise operations, which are highly computationally intensive. Some methods try to simplify distortion computation by substituting the distortion defined on a subset of pixels for the whole block distortion. For instance, the MAD of 128 pixels is used as the matching distortion for a 16x16 macroblock in [26]; the computations can be reduced by one half with little performance loss. However, this method is not suitable for small blocks such as 4x4 blocks. Partial distortion elimination (PDE) in [27] compares every line's distortion in a block to avoid computing the distortion of the entire block. In [28], hypothesis testing is used to estimate the MAD from the partial mean absolute difference (PMAD), and the estimated MAD value is used to judge the matching result.

When fast algorithms in the above three categories are put together, the motion estimation accuracy may degrade. Additional calculations such as the initial motion vector prediction could lead to a considerable amount of computational overhead. An approach proposed without quality degradation is the successive elimination algorithm (SEA) suggested by Li and Salari [29], which pre-excludes some impossible candidate points before completing the matching distortion calculation. SEA is a fast full search algorithm having a performance identical to FS while it speeds up the search process approximately by 10 times for 16x16 macroblock based motion estimation. Some further improvements have been made in subsequent research [27][30]-[33].

2.3.2.4 Fast fractional motion estimation

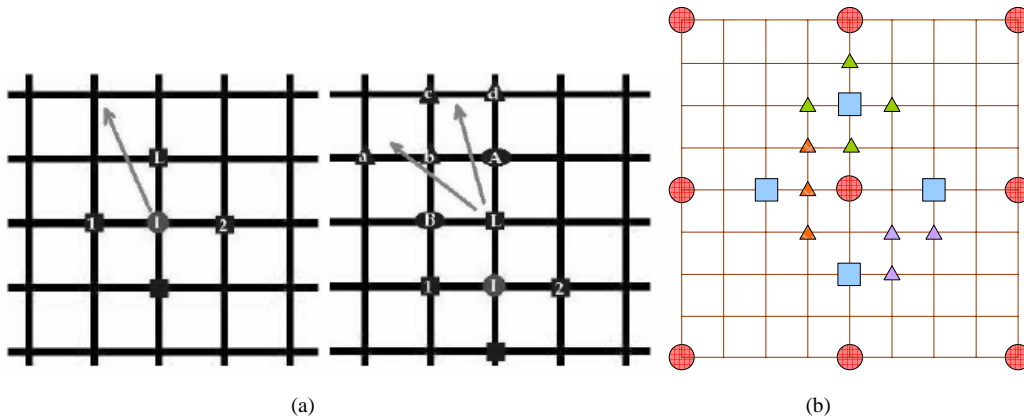


Fig 7 (a)H.-M. Wong, O. C Au, and A. Chang, “Fast sub-pixel inter-prediction–based on the texture direction analysis”, [35](b) C.-C. Cheng, Y.-J. Wang, and T.-S. Chang, “A fast fractional pel motion estimation algorithm for H.264/AVC”,[36].

In the H.264 video coding scheme [4], the inter prediction (motion vectors) precision has been increased to quarter pixel. Typically, people perform the integer pixel motion estimation (IME) first. Then, the sub-pixel motion estimation or fractional motion estimation (FME) is applied to achieve refinement. As compared to the integer-value search, FME has a somewhat different statistical character. This may be due to the facts that the search window of FME refinement is much smaller than that of IME and that the referenced sub-pixels are interpolated from the integer-coordinate pixels. Consequently, the error surface of FME is much closer to a uni-modal one, which favors fast algorithms.

Therefore, traditional fast algorithms in IME can also be used and can be more effective. The scheme adopted by the H.264 reference software is a three-step-like fast algorithm. It first checks the nine candidates surrounding the best match of IME, and then checks further the nine candidates surrounding the best match from the previous step. However, to take even more advantage of the uni-modal surface property and the highly centralized distribution of sub-pixel motion vectors, several fast FME algorithms with additional features are proposed. In [35], a gradient based search algorithm is brought up. The search direction is determined first and looks for the best motion vector along that direction. In [36], an adaptive search-pattern algorithm is proposed. The search-pattern is determined by outcome of the previous step and it is biased towards the search center. This method saves half of the computations when compared to the reference software.

2.3.2.5 Some recent approaches

The recent trend to further reduce the motion estimation calculations is to combine the techniques mentioned before. The idea is each technique, a fast algorithm, is placed its most suitable target area. Thus, how to find a specific combination that achieves the optimal solution for a specific application becomes the most important issue. In [37], a fast algorithm with better coding efficiency on residuals is proposed, which leads to a lower bit rate compared to the full search algorithm. The method proposed in [38] produces larger residuals (due to fewer search points) but less motion information. Overall, it has a better encoding efficiency and a rather fast coding speed. This type of solutions seems to be the target now researchers are aiming at.

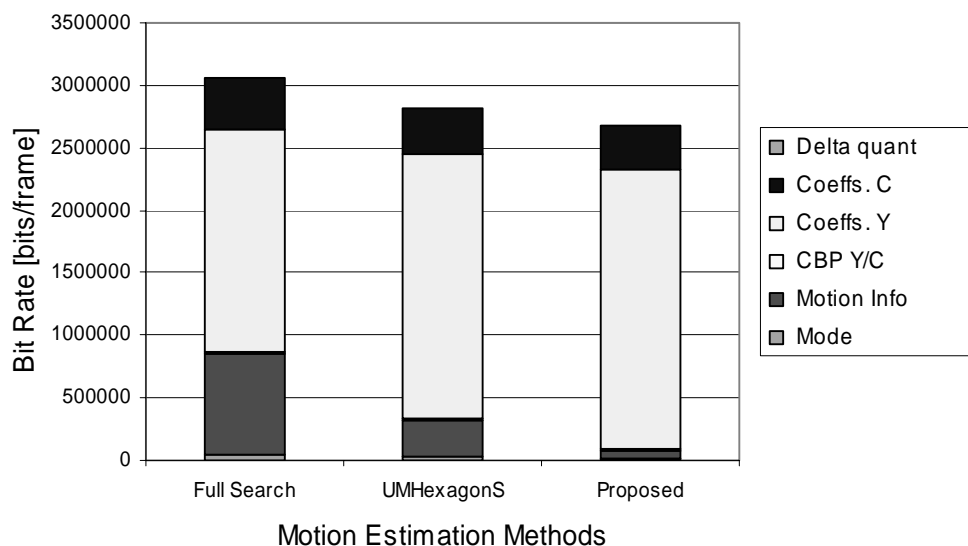


Fig 8 Comparison of bit stream portion with different fast algorithm.

2.4 FAST MODE DECISION

2.4.1 Overview

The mode decision algorithm determines the best mode of the macroblock from various combinations of inter-prediction and intra-prediction. It can be coded with seven different block sizes for motion-compensation in the inter mode, and various spatial directional prediction modes in the intra mode. To achieve the highest coding efficient as close as possible, the reference software calculates the rate distortion costs of all possible modes and then it chooses the best one that has the minimum cost. This is a very time-consuming process. To reduce the computation load, a fast mode decision algorithm is necessary, which can do a quick screening to drop most poor modes and then it examines the reminders and identifies the (nearly) best one.

2.4.2 FAST ALGORITHMS

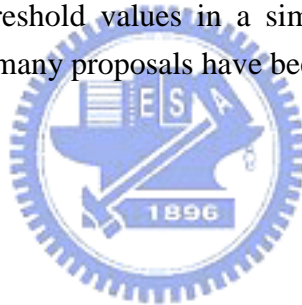
The fast mode decision algorithm can be divided into two types. The first type uses an early termination threshold to terminate the lengthy mode decision process. The early termination step can be placed between the intra and inter prediction processes [43][44] or inside the inter prediction process [45].

The scheme proposed in [43][44] uses the fact that intra mode needs more bits for coding and thus has a lower priority than the inter mode. Thus, if the best inter mode cost is smaller than a threshold, the intra prediction mode is skipped. The threshold can be the average of rate distortion cost of a number of previously coded intra blocks [43] or a ratio between the average boundary error (ABE) and average rate (AR) [44], where AR is the average bits for encoding the motion-compensated residuals and ABE is the average pixel error between the pixels at boundary of the current and its adjacent blocks in the best inter mode. The simulation results show that it can achieve about 20% reduction of computational time with a slight bitrate increase.

In [45], it observes the fact that the 16x16 block usually is the best block size for large areas of background with still or uniform motion since it has less motion vector overhead. Thus, it first checks the cost of 16x16 block size. If it is smaller than a threshold, say, an average value of previous 16x16 blocks, the inter prediction process is terminated. Otherwise, a similar procedure is applied to the 8x8 block size. The second type of the mode decision algorithms is to reduce the number of candidate modes. Intuitively, if the cost of a larger block-size mode is higher than the cost of the

current block-size mode, the even larger block-size modes can be excluded. Similarly, if the cost of a smaller block-size is higher than that of the current block-size mode, the even smaller block-size modes can be excluded. Following this argument, we give different priority to each mode. If the mode with higher priority can provide sufficient image quality, we can skip the other lower priority modes. A specific case is the SKIP mode. The SKIP mode refers to the 16x16 mode of which no motion and residual information is coded. Thus, no motion search is required and it has the lowest complexity. Therefore, many algorithms assign the highest priority to the SKIP mode and thus a large percentage of macroblocks would get the SKIP mode based on spatial-temporal neighborhood information [46]-[48]. This approach can save a significant proportion of the encoding time with a slightly bit rate increase and quality drop.

In summary, the fast mode decision algorithms can be combined with the other fast intra and inter prediction algorithms to achieve further speedup. In all these algorithms, the SKIP mode first approach can save significant computational time. How to determine proper threshold values in a simple and automatic way is one critical issue for research and many proposals have been suggested.



Chapter 3 Dynamic search range prediction for integer motion estimation

3.1 DESCRIPTION OF PRIOR ART

Motion estimation is a well known technique in video coding to achieve high coding efficiency by reducing the temporal redundancy between successive frames. Motion estimation plays an important role in such an inter-frame predictive coding system.

The full search block matching algorithm for motion estimation is the simplest but computationally very intensive, especially when the search range is large. It provides an optimal solution by exhaustively evaluating all the possible candidates within the search range in the reference frame. Many fast algorithms, such as the new three-step search [14], the block-based gradient descent search [17], the three-step search [39], the dynamic search window scheme [40], and one-at-a-time search [41] have been proposed to reduce the computational complexity by limiting the number of check points within the constant search range. The basic idea behind these fast algorithms is the assumption of the monotonically increasing block distortion function. Limited points are tested in the first stage; search is then continued in the vicinity of the point whose distortion is the smallest in previous stage. In [40], the window size in subsequent stage is determined based on the superiority of the best matched point to others in the present stage. It is clear that all these algorithms start with a constant search range and the computational complexity reduction is done at the expense of estimation accuracy due to its limited number of check points in the first stage. Different approaches of fast algorithms have also been proposed. In [42], the sub-sampled motion-field estimation scheme is proposed. It starts with sub-sampled motion-field estimation and then selectively replicates it to produce all the motion vectors. However, it performs poorly when two or more objects within the same block are moving in different directions or different velocities [42].

3.2 ANALYSIS OF INTEGER MOTION VECTOR

It is well known that the larger search range fed into motion estimation, the better rate distortion performance is obtained. We can intuitively know that the

performance increasing rate will saturate until certain degree video quality has been achieved. In order to make the compress process more efficient, we may need to know the saturate boundary of the input search range. The factor straight affected by changing input search range is motion information. Motion vector can be decomposed of motion vector predictor and motion vector difference. In Table 1, we can find the increasing rates of motion vector predictor are really small compare to the increments of input search range. For the reason that comparisons from 32 to 16 are very similar as that from 24 to 16, we can easily conclude that the increment of input search range over the saturate boundary will not get better coding efficiency.

	Compare 24 to 16				Compare 32 to 16			
	MVP_x	MVP_y	Δ PSNR	Bitrate(%)	MVP_x	MVP_y	Δ PSNR	Bitrate(%)
Stefan	4.77	4.58	-0.01	-0.003	7.73	12.15	-0.01	0.001
Foreman	0.51	1.17	-0.01	-0.001	1.06	2.64	-0.02	-0.001
Mobile	11.01	17.59	-0.01	-0.005	12.99	23.48	-0.01	-0.002
Coastguard	0.45	3.04	-0.01	-0.001	1.24	7.04	-0.01	-0.001
News	0.43	0.89	-0.01	-0.001	0.66	1.10	0.00	-0.004

Table 1 Increasing percentage of motion vector predictor with different search range.

	Compare 24 to 16				Compare 32 to 16			
	MVD_x	MVD_y	Δ PSNR	Bitrate(%)	MVD_x	MVD_y	Δ PSNR	Bitrate(%)
Stefan	20.44	15.18	-0.01	-0.003	36.30	38.28	-0.01	0.001
Foreman	6.23	4.50	-0.01	-0.001	11.55	8.87	-0.02	-0.001
Mobile	54.23	45.73	-0.01	-0.005	67.86	59.97	-0.01	-0.002
Coastguard	3.32	21.12	-0.01	-0.001	10.48	50.06	-0.01	-0.001
News	13.12	12.92	-0.01	-0.001	19.00	10.60	0.00	-0.004

Table 2 Increasing percentage of motion vector difference with different search range.

In Table 2, the same conclusion can be epitomized. Motion vector difference shows larger increasing rate with comparison to motion vector predictor, but it still not efficient enough when input search range is too large. To determine whether the input search range is too large or not, we experimented the input sequence size as CIF size to find the saturate boundary of input search range for every input sequence respectively.

As shown in Table 3, critical search range means the smallest search range with similar rate distortion performance. We listed all possible factors that will announce the search range needed. The factors that we considered can mainly be divided into

two families: matching error group and motion information group. In the former group, we record not only sum of absolute difference (SAD) but also sum of absolute transformed difference (SATD); in the later group, we record motion movement information. It is generally believed that temporal and spatial correlations of motion vector exist. As the result, it gives us spaces to apply fast algorithm.

	Critical SR	SAD	SATD	MVP_x	MVP_y	MVD_x	MVD_y
Stefan	8	303.34	381.88	23.86	4.22	1.37	0.40
Foreman	4	176.64	267.79	17.51	6.51	0.66	0.55
Mobile	4	408.33	500.96	17.15	3.85	0.96	0.43
Coastguard	2	276.61	436.79	19.21	2.33	0.70	0.08
News	2	122.55	197.70	13.57	4.12	0.12	0.11

Table 3 The correlation between search range and the factors including matching error and motion information. Critical search range means the smallest search range with similar RD performance.

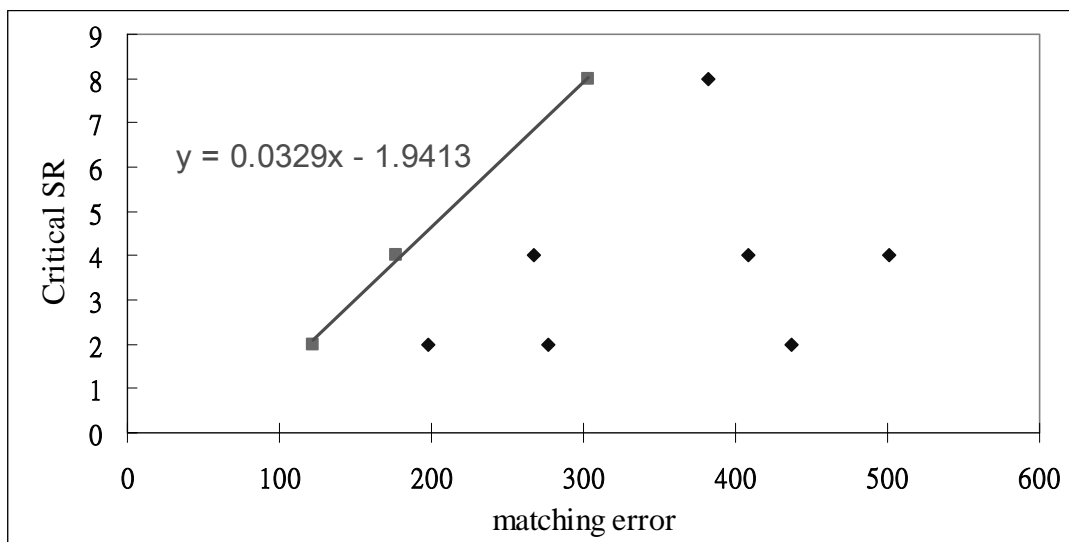


Fig 9 Correlation between critical search range and matching error.

The correlation of critical search range and factors mentioned in Table 3 are very similar. Taking matching error as the example, when the x coordinates (matching error) are getting larger, the y coordinates (critical search range) are not getting larger proportionally. In Fig 9, we can obviously find that when the matching error too small or too large have smaller critical search range. To examine the reason, it is very straight forward that too small matching error has smaller critical search range. As the belief of the spatial correlation, slight motion movement (usually small matching error) in previous macroblock means probably slight motion movement in current macroblock. To cover the slight motion movement, only small critical search range is

needed. In the case of larger matching error, it usually means that we can not find a good match position in the search window such as changing scene or complex texture within the macroblock. Indeed even if we increase the input search range will have very little RD performance improvement. As the result that motion estimation does not compress the macroblock information with large matching error, we may want to reduce the search range to achieve our goal as speedup.



3.3 PROPOSED ALGORITHM

A fast algorithm for motion estimation is proposed in this chapter. In contrast to previously proposed fast algorithms which use limited number of check points in a constant search range. The proposed algorithm performs search in a dynamic search range.

Block motion fields in real world video sequences are usually smooth and varies slowly. This produces a high correlation between the motion vectors of neighboring blocks. We record the matching error of the previous macroblock. By making use of this information, we will determine the search range used in current macroblock dynamically. The proposed algorithm can mainly divided into three steps. The details are as follows:

Step 1: Predict search range.

If ($qp > 30$)

$qp_factor = 2;$

else

$qp_factor = 1;$



$sr_factor = (input->search_range) >> 4;$

$shift_factor = qp_factor + sr_factor;$

To serve different resolution video content, we should adjust the predict scheme dynamically. Two main factors result in different resolution are quantization parameter and input video size. The former one let users can define the final video quality according to their application. The later one let users can compress video content with different input size such as QCIF for network streaming and D1 for DVD player. As the different input size, the different input search range comes. In order to reduce the error generate by predicting search range, we should adjust the sr_factor dynamically.

$$Mvd_max = (|mvd_x_prev| , |mvd_y_prev|);$$
$$max_sr = Mvd_max \ll shift_factor;$$

We record the motion vector difference of the previous macroblock for the reason that correlation exists. It is generally believed that motion vector is likely

similar as the previous one. However, this is not point motion vector difference directly. We may know the entire motion vector likely is but we can not judge the refined part (motion vector difference) accurately, so Mvd_max need to be increased to generate the probable predict search range (max_sr).

Step 2: Check the upper bound.

```
if( $sad\_previous > 600$ )
```

```
     $max\_sr\_up = (search\_range \gg 2);$ 
```

```
else if( $sad\_previous > 50$ )
```

```
     $max\_sr\_up = search\_range;$ 
```

```
else
```

```
     $max\_sr\_up = (search\_range \gg 1);$ 
```

```
     $max\_sr = \min(max\_sr, max\_sr\_up);$ 
```

In this step, we want to clip some redundant search range that was over predicted in previous step. The main idea is cut off the search range when the match error is too large or too small. The correlation is shown in Fig 9 and details are mentioned above. 600 and 50 are experiment result with input sequence as CIF size. Bad match (with too large matching error) shows more spaces to reduce search range than good match (with too small matching error) does. When matching error is over 600, it means that there is no good match position in search window. In other words, even if we skip the motion estimation process, it will not result in terrible performance loss. The amount of residual data can not be saved, so spending time to refine motion vector is not efficient and can be reduced.

Step 3: Check the lower bound.

```
if( $max\_sr == 0$ )
```

```
     $max\_sr = 4;$ 
```

The last step is to avoid skipping motion refined operation. In this step, we will make sure that the max_sr is not equal to zero. The action that skipped motion refined operation will lead to significantly rate distortion performance loss.

3.4 COMPARISON

This section shows the speedup improved by proposed fast algorithm. *Saving* mentioned below means the reduced search points compared to original ones. We record every determined search range in all macroblocks and calculate the average of them. *Saving* is calculated manually. It means not the total encoding time saving but motion refinement time saving. As listed in Table 4, the input sequence size is CIF size and input search range is given by 16. We find that the proposed algorithm obviously decreases the number of search points. When the quantization parameter is smaller than 30, almost 90% saving can be achieved. It still has more than 80% saving even the quantization parameter is bigger than 30.

CIF	QP=20		QP=24		QP=28		QP=32	
	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)
Stefan	6.764	82.126	6.753	82.184	6.663	82.652	7.915	75.528
Mobile	4.573	91.828	4.544	91.931	4.464	92.213	6.453	83.732
Foreman	5.665	87.462	5.704	87.287	5.676	87.410	7.335	78.978
Coastguard	4.699	91.373	4.745	91.204	4.790	91.034	6.639	82.778
News	4.365	92.555	4.391	92.465	4.391	92.468	4.758	91.154
Average	89.069		89.014		89.156		82.434	

Table 4 saving statistic with input search range = 16 and input sequence size as CIF size.

CIF	QP=20		QP=24		QP=28		QP=32	
	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)
Stefan	11.90	86.162	11.76	86.475	11.45	87.184	13.53	82.100
Mobile	8.537	82.882	8.447	93.031	8.216	93.407	11.57	86.918
Foreman	9.801	90.618	9.906	90.416	9.816	90.588	12.68	84.288
Coastguard	7.664	94.263	7.813	94.037	7.923	93.869	11.44	87.201
News	5.430	97.120	5.501	97.044	5.473	97.074	6.015	96.466
Average	92.209		92.201		92.424		87.395	

Table 5 saving statistic with input search range = 32 and input sequence size as CIF size.

In Table 5, we see the similar result with different simulation environment. We get even better result than that shown in Table 4. As the total encoding time issue, when the search range is larger, the time spending on motion estimation occupies bigger portion of total encoding time. So we can achieve 40% ~ 60% total encoding time saving with input search range given by 16 but 60% ~ 80% total encoding time

saving with input search range given by 32.

D1 SR=32	QP=20		QP=24		QP=28		QP=32	
	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)
Crew	17.23	70.984	15.85	75.452	14.43	79.653	16.21	74.311
Harbour	12.54	84.633	11.91	86.144	11.07	88.031	13.51	82.153
Might	13.48	82.239	12.91	83.718	12.43	84.890	13.97	80.922
Sailormen	14.93	78.203	13.97	80.927	13.10	83.239	17.45	70.237
Average	79.015		81.560		83.953		76.906	

Table 6 saving statistic with input search range = 32 and input sequence size as D1 size.

D1 SR=64	QP=20		QP=24		QP=28		QP=32	
	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)	Sr_avg	Saving(%)
Crew	42.49	55.919	39.39	62.103	36.20	67.992	40.39	60.155
Harbour	32.51	74.184	30.87	76.723	28.69	79.898	35.50	69.216
Might	30.36	77.489	28.72	79.862	27.58	81.421	31.12	76.348
Sailormen	39.99	60.940	37.88	64.955	36.04	68.284	46.44	47.332
Average	67.133		70.911		74.399		63.263	

Table 7 saving statistic with input search range = 64 and input sequence size as D1 size.

In order to make the method suitable for all kinds of video content, we concerned about many factors and adjusted the prediction scheme respectively. We developed the algorithm with input sequence size as CIF size. As shown in Table 6 and Table 7, we took input sequence size in D1 size as an experiment. The results show that smaller saving comes with larger input sequence size. It means that the proposed algorithm is a little conservative for larger input sequence size. Even if the determined search range is over predicted, it still has almost 80% saving as listed in Table 6 and almost 70% saving as listed in Table 7. As the total encoding time issue, both of them are about 40% ~ 60% saving.

3.5 SIMULATION RESULT

After the comparison of the speedup, this section shows the corresponding rate distortion performance. We summarized the result into Table 8 to Table 11. We have less than 0.03 dB PSNR drop and less than 0.5 % bit rate increased in the case of input sequence size as CIF size (Table 8 and Table 9).

JM 8.2	CIF	QP = 20			QP = 24		
		Original	Proposed	$\Delta(\text{dB},\%)$	Original	Proposed	$\Delta(\text{dB},\%)$
Stefan	PSNR	41.57	41.54	-0.03	38.42	38.37	-0.05
	Bit rate	3974.9	4042.69	1.705	2493.37	2538.14	1.795
Mobile	PSNR	40.61	40.6	-0.01	37.08	37.07	-0.01
	Bit rate	4996.31	4973.41	-0.458	3199.12	3187.18	-0.373
Foreman	PSNR	41.72	41.7	-0.02	38.83	38.8	-0.03
	Bit rate	1834.93	1859.84	1.357	969.3	987.24	1.850
Coastguard	PSNR	40.83	40.82	-0.01	37.52	37.51	-0.01
	Bit rate	3289.18	3274.37	-0.450	2003.72	1996.76	-0.347
News	PSNR	43.13	43.13	0	40.67	40.64	-0.03
	Bit rate	622.02	623.01	0.159	373.98	374.66	0.181
Average	PSNR	-0.014			-0.026		
	Bit rate	0.462			0.621		

Table 8 rate distortion result with input search range = 16 and input sequence size as CIF size.

JM 8.2	CIF	QP = 20			QP = 24		
		Original	Proposed	$\Delta(\text{dB},\%)$	Original	Proposed	$\Delta(\text{dB},\%)$
Stefan	PSNR	41.59	41.55	-0.04	38.45	38.4	-0.05
	Bit rate	3907.46	3931.66	0.619	2414.45	2445.22	1.274
Mobile	PSNR	40.6	40.59	-0.01	37.08	37.06	-0.02
	Bit rate	5016.42	4983.1	-0.664	3212.55	3190.48	-0.686
Foreman	PSNR	41.72	41.7	-0.02	38.83	38.8	-0.03
	Bit rate	1837.43	1854.24	0.914	970.15	983.24	1.349
Coastguard	PSNR	40.83	40.82	-0.01	37.52	37.51	-0.01
	Bit rate	3296.59	3279.85	-0.507	2006.07	1997.74	-0.415
News	PSNR	43.14	43.12	-0.02	40.67	40.63	-0.04
	Bit rate	624.8	627.92	0.499	375.32	378.82	0.932
Average	PSNR	-0.02			-0.03		
	Bit rate	0.172			0.490		

Table 9 rate distortion result with input search range = 32 and input sequence size as CIF size.

The previous section have pointed out that the speedup of the larger input sequence size has less speedup. In other words, less speedup means better rate distortion performance. The argumentation can be proved in this section through Table 10 to Table 11. We can find that both PSNR drop and bit rate increased are obviously smaller than that in Table 8 and Table 9.

JM 8.2		D1	QP = 20			QP = 24		
SR = 32			Original	Proposed	$\Delta(\text{dB},\%)$	Original	Proposed	$\Delta(\text{dB},\%)$
Crew	PSNR	41.97	41.95	-0.02	39.34	39.33	-0.01	
	Bit rate	9727.04	9611.98	-1.182	4935.23	4896.6	-0.782	
Harbour	PSNR	41.22	41.19	-0.03	38.28	38.26	-0.02	
	Bit rate	13516.99	13176.58	-2.518	8157.34	7999.64	-1.933	
Night	PSNR	41.97	41.95	-0.02	39.1	39.08	-0.02	
	Bit rate	10574.44	10446.64	-1.208	5740.23	5712.69	-0.479	
Sailormen	PSNR	41.04	41.02	-0.02	38.01	38	-0.01	
	Bit rate	12398.86	12267.2	-1.061	5955.03	5920.67	-0.576	
Average	PSNR	-0.022			-0.015			
	Bit rate	-1.492			-0.943			

Table 10 rate distortion result with input search range = 32 and input sequence size as D1 size.

JM 8.2		D1	QP = 20			QP = 24		
SR = 64			Original	Proposed	$\Delta(\text{dB},\%)$	Original	Proposed	$\Delta(\text{dB},\%)$
Crew	PSNR	41.97	41.96	-0.01	39.35	39.33	-0.02	
	Bit rate	9806.21	9725.62	-0.821	4973.31	4941.56	-0.638	
Harbour	PSNR	41.22	41.2	-0.02	38.29	38.26	-0.03	
	Bit rate	13616.41	13328.96	-2.111	8195.73	8064.2	-1.604	
Night	PSNR	41.97	41.95	-0.02	39.1	39.08	-0.02	
	Bit rate	10663.88	10503.1	-1.507	5779.62	5727.58	-0.900	
Sailormen	PSNR	41.04	41.03	-0.01	38.02	38	-0.02	
	Bit rate	12449.29	12354.03	-0.765	5974.17	5948.22	-0.434	
Average	PSNR	-0.015			-0.022			
	Bit rate	-1.301			-0.894			

Table 11 rate distortion result with input search range = 64 and input sequence size as D1 size.

We have less than 0.022 dB PSNR drop and even lower than original bit rate performance. When the quantization parameter is getting bigger, the less coding efficiency is carried with. However, in order to get so huge a speedup, sacrificing small amount of quality loss is still worth. Rate-distortion curves are shown in Fig 10 and Fig 11. As the input sequence as CIF size, we simulated search range equal to 16 and 32; as the input sequence as D1 size, we also simulated search range equal to 32 and 64. All of them are very close to original method.

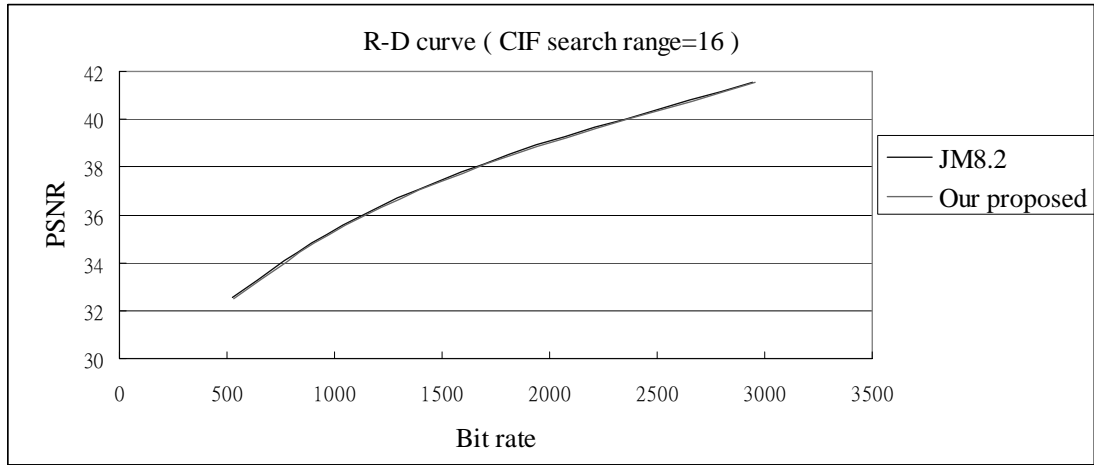


Fig 10 rate distortion curve with CIF size and search range =16.

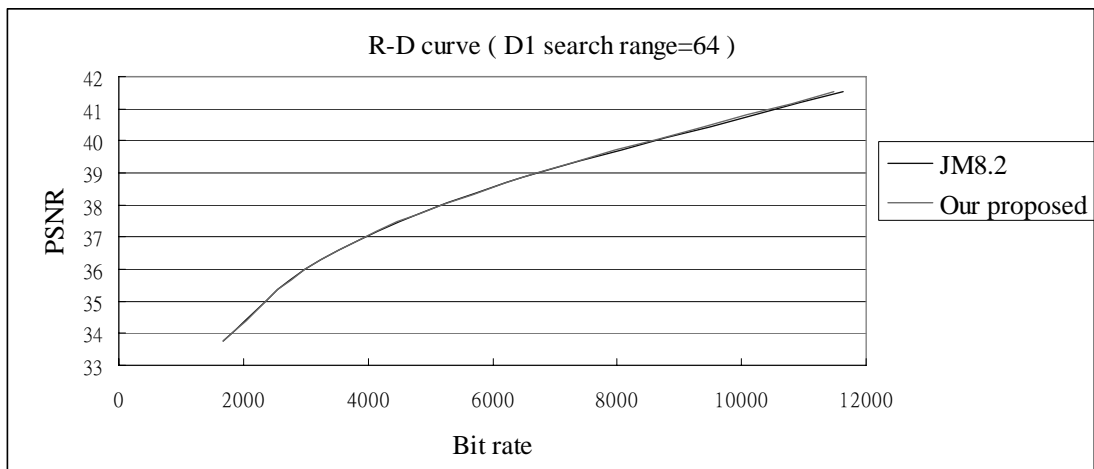


Fig 11 rate distortion curve with D1 size and search range =64.

	JVT-B022	JVT-D117	JVT-Q088	Our proposed
Worst PSNR loss (dB)	0.068	0.09	0.022	0.03
Worst Bit rate increase	1.37%	01.63%	0.42%	1.58%
ME Time saving	n/a	n/a	13%	75%
Total Time saving	49.44%	61.27%	8.3%	51%

Table 12 performance comparison

As listed in Table 12, we can find that our proposed algorithm is not the fastest one and not the most accurate one either. But it is the best solution if we have to consider speedup and video quality at the same time.

Chapter 4 Adaptive search pattern prediction for fractional motion estimation

4.1 ANALYSIS OF FRACTIONAL PEL MOTION VECTOR

It is generally believed that the fast ME algorithm works best if the error surface inside the search window is unimodal.

As shows in Fig 12, the error surface of integer pel ME is not unimodal due to the large search window and complexity of video content. So the ME search would easily be trapped into a local minimum. On the other hand, since the sub-pels are generated from the interpolation of integer pels, the correlation inside a fractional pel search window is much higher than that of the integer pel search window. Thus, the uni-modal error surface will be valid in most cases of the fractional pels. So the matching error decreases monotonically as the search point moves closer to the global minimum.

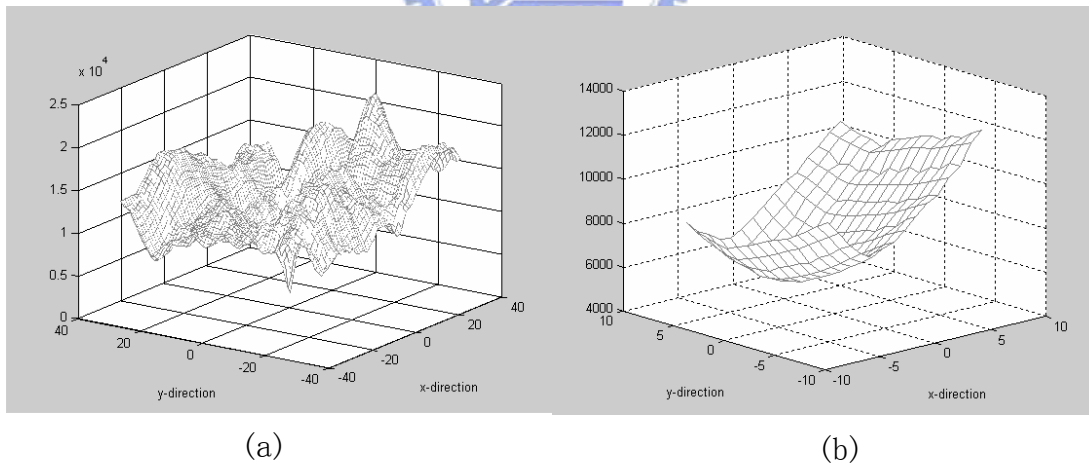
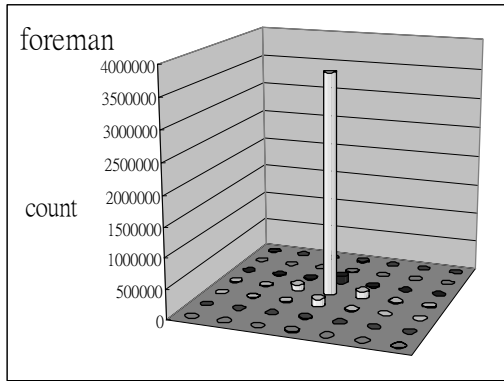


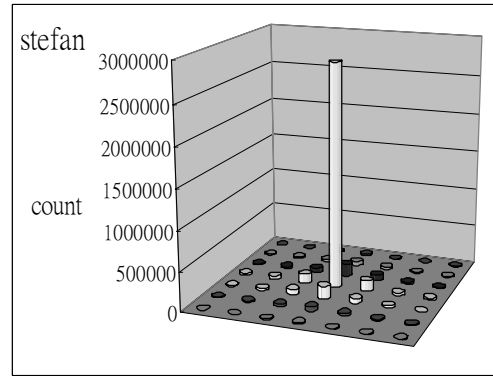
Fig 12 (a) Error surface of integer pel ME (search range: 32); (b) Error surface of fractional pel ME (1/8-pel case)

In the full search method, every fractional pels around the original integer pels are treated equal. However, with the valid unimodal error surface assumption, a fast algorithm can work well if every candidate of the sub-pel refinement has different occurring probabilities. Fig 13 shows the distribution of the fractional motion vector around the best integer motion vector. It is obvious that more than 90% of fractional

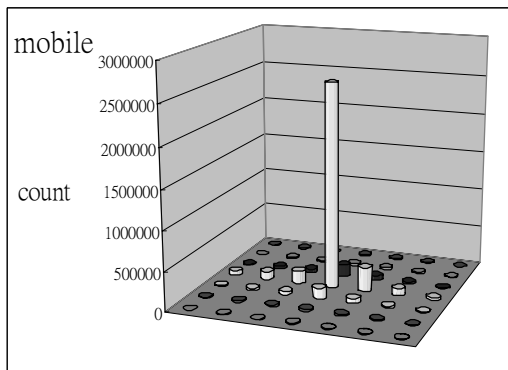
motion vector are at the search center in all kinds of video content. However, we still can not just avoid the fractional part even there are huge density diagram appear near the bias search center. The small error drift of fractional part in motion vector will lead to significantly bit rate increase.



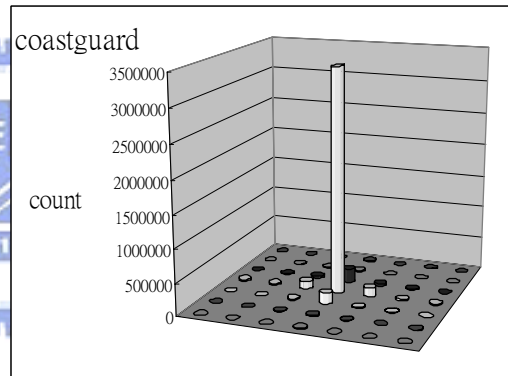
(a)foreman.



(b)Stefan.



(c) mobile and calendar.



(d)coastguard

Fig 13 Distribution of the fractional ME.

4.2 ORIGINAL SEARCH ALGORITHM

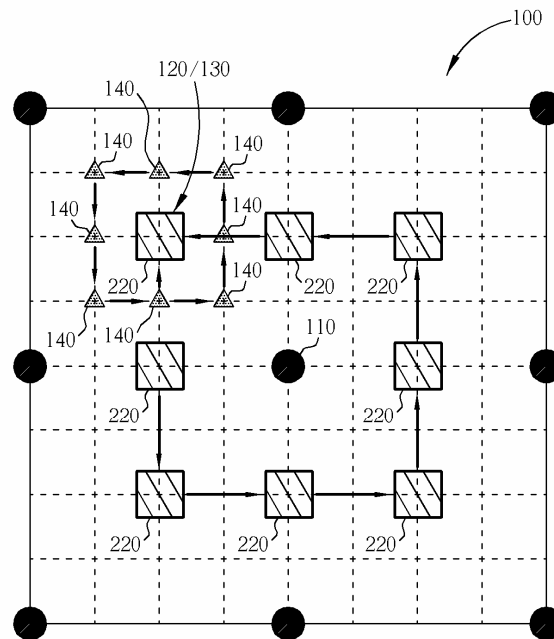


Fig 14 Search algorithm in reference software.

Fig 14 details a typical search algorithm for the ME process 100 according to the reference software. The search process in fractional motion estimation is typically divided into two parts. The first part consists of half-pel motion estimation, where specific pixels at half-pel spacing are searched for comparison. The second part consists of quarter-pel motion estimation, where pixels at quarter-pel spacing centered around a search point obtained in the first part are used for comparison.

In the first part of half-pel ME, a cost value for each of eight search points 120 in a square search pattern surrounding the integer spaced pel called search center 110 is calculated. A cost value calculation for the search center 110 is not performed. The single search point from the group of search points 120 possessing the lowest cost value is then selected as the quarter-pel motion estimation search center 130 in the next step. The fractional motion estimation step utilizes an additional eight fractional search points 140 displaced around the FME search center 130 in a smaller square pattern. A total of 17 search points (1 search point from integer pel, 8 search points from half-pel ME and 8 search points from quarter-pel motion estimation) are therefore searched and compared in a single round of the traditional ME procedure according to the reference software.

Although the typical search algorithm for the ME process 100 does manage to

sufficiently locate suitable search points for the motion vector refinement process, the excess amount of search points may result in significant delays in the encoding process. The typical search algorithm for the ME process 100 may possess too many search points to visit within one motion vector refinement process. Furthermore, the search pattern used in the ME process 100 may provide further constraints in finding optimal search points, as refinement in the fractional ME searching can cause the search area to stray away from the search center.

In order to overcome these problems, and produce a more efficient search pattern, a fast ME search algorithm is proposed. This algorithm produces a search pattern based on the high statistical probability that fractional motion is located close to the integer search center (as shown in Fig 13). In this way, fewer search points based near or on the integer search center should be visited in the proposed algorithm. This allows the complete fractional ME process to be accomplished with fewer overall search points compared to the original method, while providing a comparable accuracy. Furthermore, the overall computational resources and complexity to search a predefined search area is greatly reduced.



4.3 PROPOSED ALGORITHM

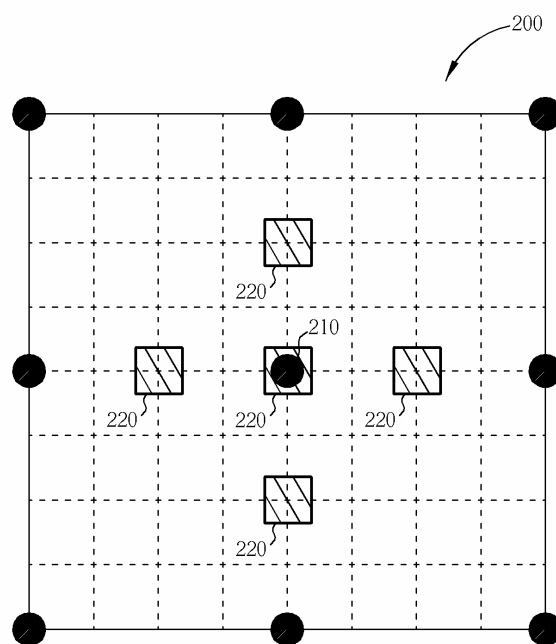


Fig 15 Proposed algorithm for half-pel.

Fig 15 illustrates a fast ME search algorithm. The algorithm is used to determine an optimized search pattern comprising a half-pel search pattern 200 and a quarter-pel search pattern (discussed later). An integer spaced pel 210, or search center is shown as a circle in the center of the macroblock in Fig 15. The first stage of the algorithm comprises half-pel ME, where the half-pel search pattern is formulated. A total of 5 search points 220 are selected to form the half-pel search pattern: four search points aligned to form a cross pattern around the search center and one search point located at the search center. Fewer search points can be used in other embodiments. Once the half-pel search pattern has been determined, the cost value for each search point 220 is calculated. Any suitable cost function can be used in this step, however, the sum of absolute transform differences (SATD) is generally used for the fractional ME. The cost function is used to determine the lowest 2 (or 3) cost values of the search points 220. Upon determination of the search points 220 producing the lowest cost values, a quarter-pel search pattern for the fractional ME process is adaptively selected.

The next stage of the fast ME algorithm process entails selecting a quarter-pel search pattern. The quarter-pel search pattern is selected according to the ranking of cost values for each specific search point, and provides search points in a certain area to approach the global minimum cost in the search window. In an effort to reduce confusion, the search points deduced in the quarter-pel ME stage will be referred to as

quarter-pel search points. However, both types of search points serve the same purpose in providing matching points for the ME process.

Once the quarter-pel search pattern is determined (further below), cost values for the quarter-pel search points of the fractional search pattern are then calculated. The cost values attained here are used in conjunction with the cost data accumulated from search points in the first stage to determine whether the current macroblock is a suitable match to the reference macroblock. The entire search pattern therefore comprises the half-pel search pattern used in the first stage and the quarter-pel search pattern used the second stage for fractional ME.

The following cases illustrate how the quarter-pel search pattern is selected in the second stage in fractional ME. The quarter-pel search pattern is based on a ranking of the cost values for each search point in the first stage for half-pel ME. The cases are as follows



Case 1: The lowest cost search point is located at the search center, and the second and third lowest search points are opposite to each other.

This case is illustrated in Fig 16. In this case, the lowest cost search point 320 is located at the search center 310, and the second lowest cost search point 330 and third lowest cost search point (not shown) are opposite each other. For this case, three quarter-pel search points 340 placed between the minimum cost search point 320 and the second lowest cost search point 330 are selected as the quarter-pel search pattern in fractional ME. The three quarter-pel search points 340 are configured such that they form a straight line perpendicular to the axis formed by the lowest cost search point 320 and the second lowest cost search point 330, and are located in between the two half-pel search points 320 and 330.

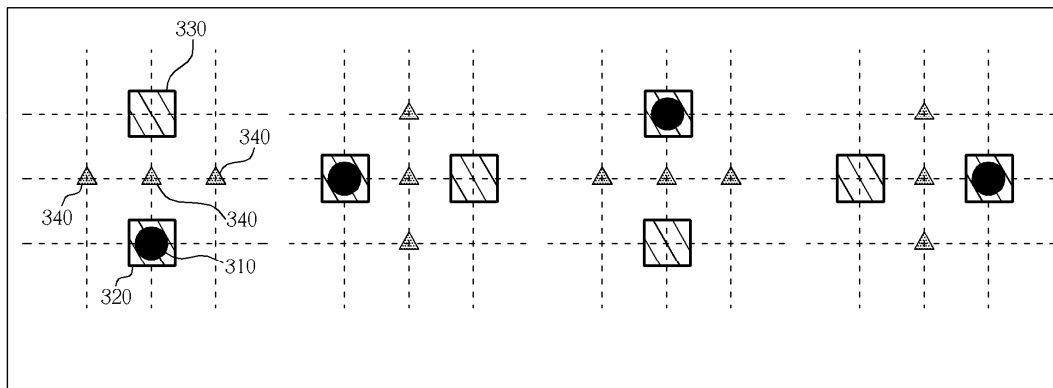


Fig 16 Proposed algorithm for quarter-pel (case 1).

Case 2: The lowest cost search point is located at the search center, and the second and third lowest cost search points are adjacent to each other.

This case is illustrated in Fig 17. The lowest cost search point 420 is located at the search center 410, and the second lowest cost search point 430 is adjacent to the third lowest cost search point 440. For this case, three quarter-pel search points 450 are used to form the quarter-pel search pattern in fractional ME. The three quarter-pel search points 450 are arranged between the three lowest cost search points such that a connection among the three quarter-pel search points 450 would form a right angle with the vertex of the right angle concave to the search center 410.

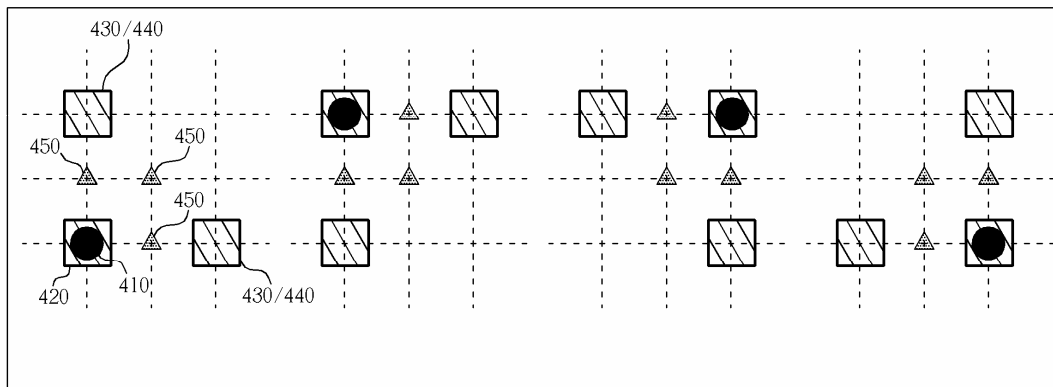


Fig 17 Proposed algorithm for quarter-pel (case 2).

Case 3: The two lowest cost search points are adjacent to each other and surround the search center.

This case is illustrated in Fig 18. The lowest cost search point 520 and the second lowest cost search point 530 are adjacent to each other and both surround the search center 510. For this case, three quarter-pel search points 550 are used to form the quarter-pel search pattern in fractional ME. The three quarter-pel search points 550 are arranged between the two lowest cost search points such that lines connecting the three quarter-pel search points 550 would form a right angle, with the vertex of the right angle convex to the search center 510.

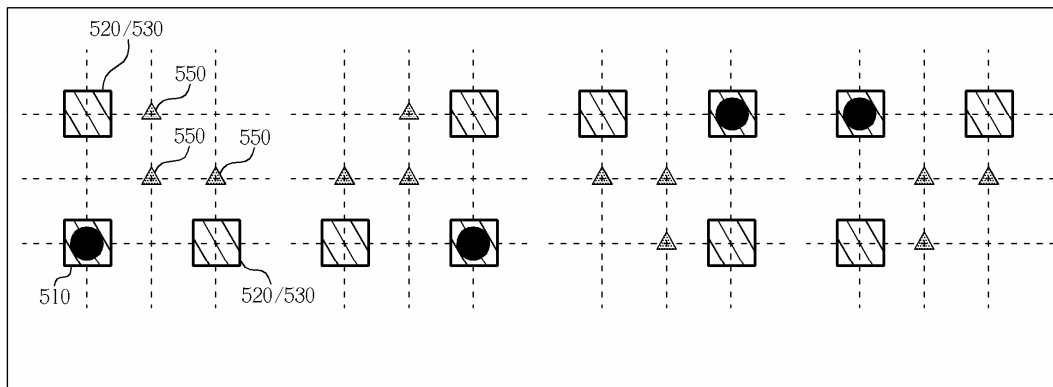


Fig 18 Proposed algorithm for quarter-pel (case 3).

Case 4: The two lowest cost search points are opposite to each other and surround the search center.

This case is illustrated in Fig 19. The lowest cost search point 620 is opposite to the second lowest cost search point 630, neither being located at the search center 610. For this case, four quarter-pel search points 650 are used to form the quarter-pel search pattern in fractional ME. The quarter-pel search pattern is arranged such that the four quarter-pel search points 650 surround the lowest cost search point 620 in a square pattern.

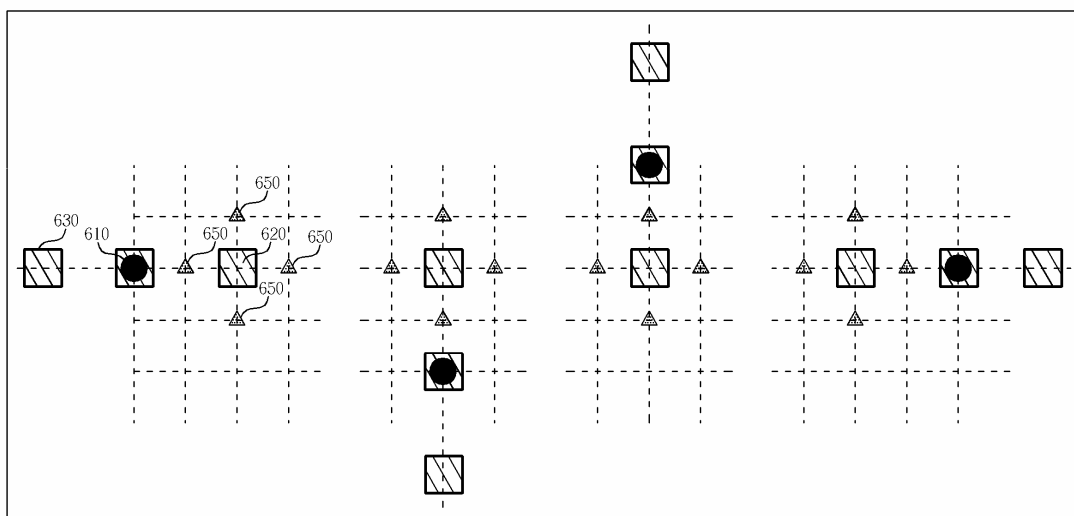


Fig 19 Proposed algorithm for quarter-pel (case 4).

Once a fractional search pattern is selected based on one of the 4 above cases, calculations for each quarter-pel search point using a specified cost function can be performed to complete the matching process. The data provided from the half-pel search points and the quarter-pel search points serve to provide a comprehensive data set in an area approaching a local minimum of the cost function. This allows for a more accurate match result, while lowering the need for calculating additional search points. The best matching macroblock that minimizes the difference between the current and reference macroblock can now be chosen.

4.4 COMPLEXITY AND ACCURACY COMPARISON

The following table provides a summary of the total search points used in the method of the present invention for each potential case, compared to alternative search algorithms for ME.

		Total Search Points
Full Search Algorithm		49
Reference software Algorithm		17
Proposed Algorithm	Case1	8
	Case2	8
	Case3	8
	Case4	9

Table 13 Search point comparisons for different algorithms

As illustrated above in Table 13, the search algorithm of the present invention for motion estimation significantly reduces the total search points in comparison with the reference software method. For cases 1-3, a 52% reduction in search points is attained, while a 47% reduction in search points is achieved in case 4. This significantly reduces the hardware processing time required by a related compression encoder or a microprocessor for use in video compression.

	Reference software		Proposed algorithm	
	MV_x hit rate	MV_y hit rate	MV_x hit rate	MV_y hit rate
Stefan	0.95086006	0.94094485	0.81168958	0.84223568
Mobile	0.93886042	0.90366406	0.79969447	0.78225377
Foreman	0.9223743	0.88868022	0.82468642	0.82102286
News	0.9759824	0.96602579	0.9272946	0.91579924
Coastguard	0.9268235	0.94608455	0.77841821	0.84063553

Table 14 Algorithms prediction correctness compare to full search algorithm

Additionally the method of the present invention manages to arrive at a comparable matching accuracy while reducing the total search points and processing time. Table 14 below details the prediction accuracy of both the proposed algorithm and the reference software algorithm. The prediction accuracies are measured as a hit rate of the fractional motion vector in the x and y axis of the respective algorithm compared to the motion vector produced through the full search algorithm applied in the fractional search window. We see that the algorithm of the reference software manages to consistently produce a hit rate of around 90% for the various video samples. The proposed algorithm produces a comparable hit rate of about 80%, while reducing the search points by roughly half.

4.5 EARLY TERMINATION

We also apply the early termination technique to every single search point in each step. The problem for early termination is how to define the threshold. The matching error considered as SATD is used in fractional motion estimation and SAD in integer motion estimation. SATD is the results after SAD go through 2D Hadamard transform. The threshold value (SATD) used in fractional ME can be estimated from the integer-pels matching error (SAD). We experiment from several test sequences and get the formula listed in the Fig 20.

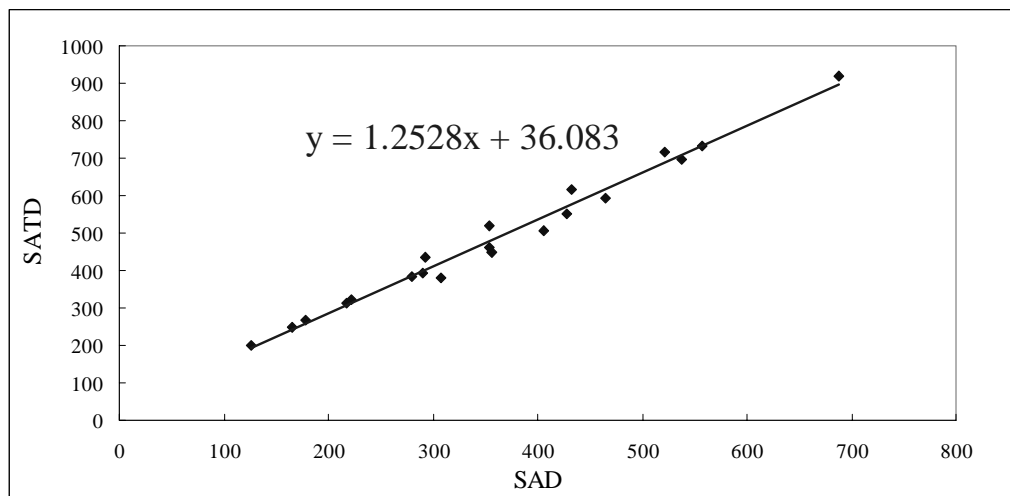


Fig 20 Relationship between best SAD in integer part & best SATD in fractional part

In most of situations, we can use above approximated formula to predict the threshold. However, direct linear prediction may lead to a too large threshold and arise too much imprecision when the SAD getting larger. To solve the problem and avoid second or high order approximation, we adopt adaptive linear prediction threshold. We have found that while the quantization parameter (QP) is getting larger, the rate distortion performance is getting better. It means that we have more spaces to save when larger QP comes. To achieve the shorter searching time without significant performance loss, we increase the threshold associating to the current QP. The final prediction formulas are listed below.

```

if (SAD > 1000)
{ threshold = SAD*0.75 + (QP-28)*16 + 375 + 36; }
else if (SAD > 500)
{ threshold = SAD*1 + (QP - 28)*16 + 125 + 36;}
else
{ threshold = SAD*1.25 + (QP - 28)*16 + 36;}

```

Every coefficient used in the formula could be calculated by add and shift, and the summation of constants could be combined easily. Constant with the value of 36 is obtained in the formula listed in Fig 20. Constants with the value of 375 and 125 are used to maintain the continuity of the adaptive prediction curve. The adaptive threshold prediction curve is shown in Fig 21.

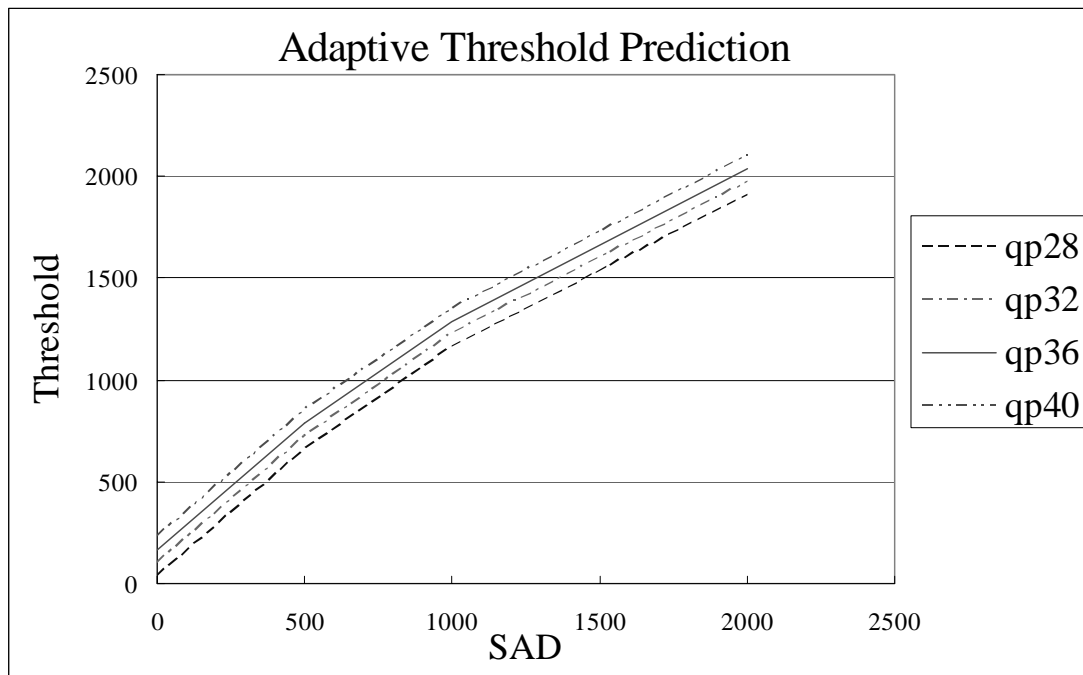


Fig 21 Adaptive threshold prediction curve

By applying early termination technique, we can improve searching speed about 8.5 % to 14%. While the QP is getting larger, we may get a bigger threshold and lead to the shorter searching time.

4.6 SIMULATION RESULT

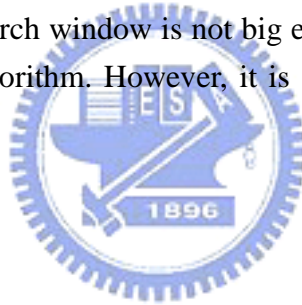
Table 15 shows the simulation results of the proposed algorithm compared with that of the reference software. We integrate our algorithm into the reference software and use the full search algorithm for integer ME for fair comparison. It can be found that our algorithm greatly reduces computational complexity but only leads to a small amount of quality loss. For the low motion sequences, our algorithm has about 0.1-0.2dB PSNR degradation at the same bit rate as reference software. For the median motion sequence, such as foreman, and coastguard, we can find that about 0.1 dB PSNR degradation at the same bit rate with respect to algorithm in reference software.

QP = 28		Stefan	Mobile	Foreman	Coastguard	News
ref. software	bit rate	1441.14	1888.69	498.62	1127.87	223.72
	PSNR	35.36	33.75	36.24	34.52	38.12
	time (sec)	491.604	471.993	496.974	488.039	450.37
proposed	bit rate	1474.06	1933.34	507.5	1139.94	228.72
	PSNR	35.27	33.64	36.17	34.48	38.06
	time (sec)	209.884	210.554	210.435	213.736	200.354
	Δ bit rate(%)	2.2843	2.36407	1.780915	1.070159	2.23494
	Δ PSNR	-0.09	-0.11	-0.07	-0.04	-0.06
	speed up	2.34227	2.24167	2.361651	2.283373	2.24787

Table 15 Simulation result when QP = 28, speed up is only the performance in fractional ME part.

RDO is off, reference frame number = 1, CIF.

In the high motion sequences, such as stefan, our algorithm has about 0.2 dB PSNR degradation at the same bit rate of reference software. The reason of the quality loss is the coverage of our search window is not big enough. Thus, some position can not be arrived by our fast algorithm. However, it is still acceptable since the loss is still small.



4.7 COMPARISON

Besides the simulation result with constant QP, we also turn on the rate control option. The comparison result are listed in Table 16, we can see that our proposed algorithm is the fastest and the most accurate one. The lower bit rate is; the better performance comes. Our algorithm is very kind for network communication.

Rate control enable		64kbps			128kbps			256kbps			512kbps		
		2SS	FSIP	Our	2SS	FSIP	Our	2SS	FSIP	Our	2SS	FSIP	Our
Foreman	Δ PSNR	0.17	0.19	0.02	0.17	0.17	0.02	0.1	0.1	0.04	0.13	0.14	0.09
	Speedup	2	2.62	4.52	2	2.55	4.52	2	2.85	3.91	2	2.7	3.52
Coastguard	Δ PSNR	0.05	0.07	0.01	0.02	0.02	0.01	0.03	0.04	0.02	0.05	0.05	0.05
	Speedup	2	3	3.53	2	2.89	3.53	2	3.01	3.18	2	2.82	2.83

Table 16 Comparison between different fast algorithms for fractional ME.

Δ PSNR: PSNR drop compare with original method used in reference software.

Speedup: speedup in fractional motion estimation.

2SS: fast algorithm proposed in [34]. FSIP: fast algorithm proposed in [35].

Chapter 5 Integration

In this chapter, we applied the fast algorithms mentioned in chapter 3 and chapter 4. We used dynamic search range prediction scheme for integer motion estimation and adaptive search pattern prediction scheme for fractional motion estimation. Fast algorithms are necessary for intolerant large computation time, such as SDTV or even HDTV applications. For the great reduction of computational complexity, we take 720p (1280 x 720) as the input sequence size. The total encoding time can be reduced to 5% compared to original one. In other words, we can achieve 20 times speed up. The details are listed in Table 17.

JM 8.2 SR = 64 720p		QP = 20			QP = 24		
		Original	Proposed	$\Delta(\text{dB},\%)$	Original	Proposed	$\Delta(\text{dB},\%)$
Mobcal	PSNR	40.78	40.74	-0.04	37.54	37.51	-0.03
	Bit rate	44878.75	45184.02	0.680211	23555.1	23794.68	1.017105
Parkrun	PSNR	40.37	40.33	-0.04	36.76	36.71	-0.05
	Bit rate	68330.01	67666.61	-0.97088	44269.89	43893.55	-0.8501
Shields	PSNR	40.86	40.82	-0.04	37.7	37.66	-0.04
	Bit rate	39812.74	39968.13	0.390302	18962.41	19028.51	0.348584
Stockholm	PSNR	40.75	40.74	-0.01	37.49	37.46	-0.03
	Bit rate	42461.87	42881.22	0.987592	20223.58	20335.03	0.551089
Average	PSNR	-0.035			-0.037		
	Bit rate	0.271			0.267		

Table 17 rate distortion result with input search range = 64 and input sequence size as 720p size.

As the 20 times speedup, the rate distortion performance is quite good enough. Furthermore, we propose a VLSI architecture design of sub-pel ME for H.264/AVC in chapter 5. By taking advantage of the correlation between motion vectors and uni-modal error surface, the proposed algorithm can significantly decrease more than 95% computational complexity and with at worst 0.04 dB PSNR degradation. The corresponding architecture can significantly decrease the total number of 4x4 block PU by reducing the candidates in the same step and speed up the search process by modified early termination technique. The resulting architecture achieves the slight video quality loss but nearly 40% area saving and 14% time saving when compared to the previous one (proposed architecture in [51]). Finally, some intermediate results of fast sub-pel ME are shown in appendix.

Chapter 6 Architecture design for fast sub-pel inter coding in H.264

6.1 HARDWARE CONSIDERATION

The encoding procedure is dominated (90%) by the inter prediction for new techniques in H.264 as variable block sizes, multiple reference frames and Lagrangian mode decision. Inter prediction can be mainly divided into two parts: integer motion estimation (IME) and fractional motion estimation (FME). Complexities of the former one and the later one are quite the same and both dominate the encoding time of inter prediction. For the speed up in system level, we may pipeline the IME and FME process. So the dedicated hardware is needed for FME only. For the speed up in the macroblock level, we can use the fast algorithm instead of the method applied in JM8.2 [52].

Lagrangian mode decision of a macroblock should be done after choosing the best cost among the 41 sub-blocks in every reference frame with quarter precision. In Fig 22, we can find there are total seven types of block sizes and may execute independently. If the critical concern is the encoding time, the parallelism as the hardware acceleration technique can be applied. But it will result in unacceptable huge chip area and power consumption. So we should make use of the common part of different block sizes. Every type of block sizes can be decomposed by 4X4 block. Only 4X4 block should be implemented necessarily, refinement of all the other types can be done with the hardware technique named as folding.

In spite of the regular algorithm of FME used in JVT, the complex Lagrangian mode decision will also arouse the difficulty of hardware implementation, so irregular search algorithm is further not prefer to implement. In [35], the gradient based search algorithm is proposed. In which, the interpolated pixels needed in next stage have higher variation probability in search window and that is often regarded as the defect of hardware implementation.

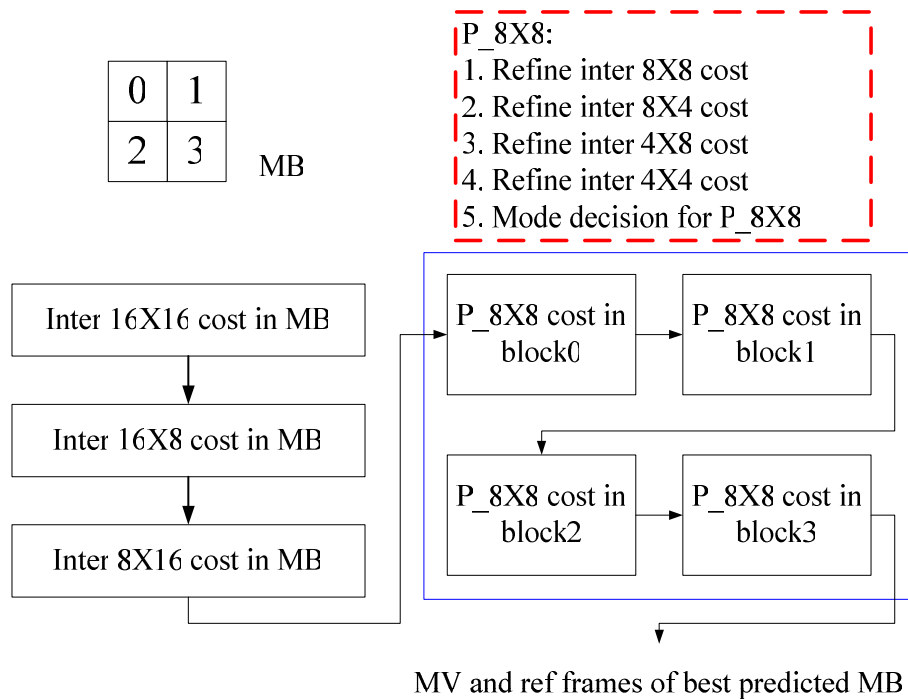


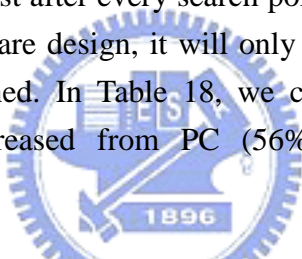
Fig 22 Mode decision flow in H.264.

4X4 block decomposition and vertical integration are proposed in [51]. All block types can be decomposed by 4X4 block, and the SATD of each element is accumulated to get the final cost. For the data reusability, vertical integration is one of the ways to reduce the encoding time. Redundant interpolating operations appear in the overlapped area of adjacent interpolation window and can be merged by scheduling technique. But the overhead as the more complex timing control circuit will be introduced.

6.2 ALGORITHM FOR HARDWARE MODIFICATION

The algorithm implemented is slightly different from the fast algorithm mentioned before. The main difference is when to apply the early termination technique. As the method mentioned before, we still use adaptive prediction SATD as criterion to terminate the refine process or not. However, early termination at each search point is only reasonable for the application running on DSP or CPU since instruction is executed sequentially. But in the case of hardware design, the available resources allow us to use parallel processing unit to speed up the whole FME process. Thus, we use five 4X4 block PU's (processing element, discuss later) to manage all of the search points in the same step, and only terminate the second step process if meeting the requirement.

As shown in Table 18, we numerate the probability when different early termination techniques activated. Point Check (PC) means the way used on DSP or CPU, it will check the final cost after every search point refinement. Step Check (SC) means the way used in hardware design, it will only check the criteria after the best cost in each step is determined. In Table 18, we can see that the count of early termination occurrences decreased from PC (56%) to SC (28%), but is still significantly.



QP=28	Point Check(PC)	PC hit rate(%)	Step Check(SC)	SC hit rate(%)	total count
Stefan	3164430	65.1846386	1761015	36.2754513	4854564
Foreman	2869524	59.1098191	1634614	33.6716953	4854564
Coastguard	2099169	43.2411438	1026942	21.1541551	4854564
News	3068064	63.1995788	1325303	27.3001448	4854564
Mobile	2577329	53.0908440	1000258	20.6044868	4854564
average	2755703.2	56.7652049	1349626.4	27.8011867	4854564

Table 18 Simulation result when QP = 28, point check means the early termination applied in every search point, step check means the early termination just applied in half step

The total encoding time of above modification can be calculated as follows. First, the encoding time is the same in each step refinement since every search point is calculated in parallel. Let us assume the total time without early termination is T , and t as the total time with step stop early termination. We can find the following relationship:

$$t = T * (1 - 0.28) + 0.5T * 0.28 = 0.86 T$$

Thus, by using step stop early termination technique, we can save 14% search time in FME module. The related quality loss will be shown latter.

6.3 ARCHITECTURE

Fig 23 shows proposed architecture for fast FME module. The core procedure of FME includes interpolation, residual generation and Hadamard transform.

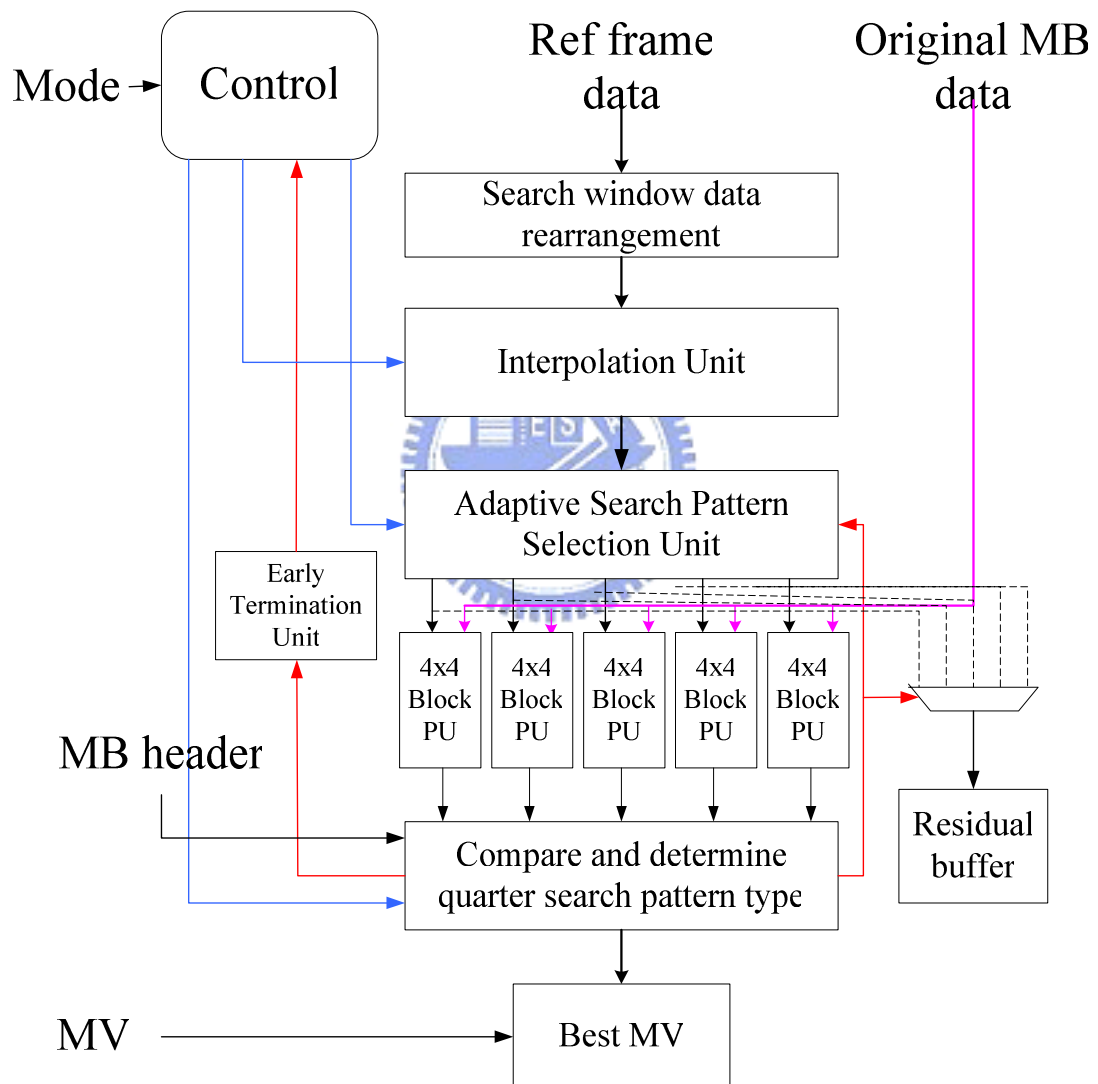


Fig 23 Block diagram of fast FME hardware.

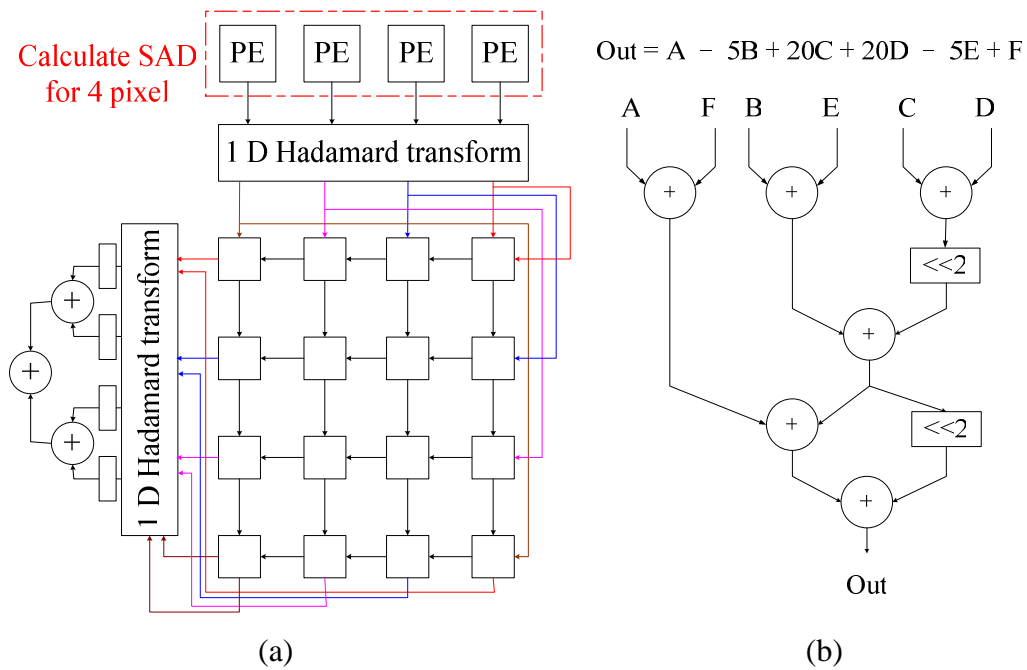


Fig 24 (a) 4X4 block PU (b) 6-tap 1-D FIR filter.

The 4X4 block PU has four times parallelization of horizontal adjacent pixels and is in charge of residual generation and Hadamard transform. The architecture of PU is shown in Fig 24(a), four processing elements (PE), 2-D Hadamard transform decomposed by two 1-D Hadamard transform and a transpose register array [53] can continually process four pixels in each cycle without any latency. It processes 4X4 element blocks decomposed from sub block in sequential order.

Five 4X4 block PUs around the refinement center process five candidates simultaneously. Four horizontal adjacent pixels from original MB are broadcasted to every PU at the same time and the reference sub pixels are provided by interpolation unit.

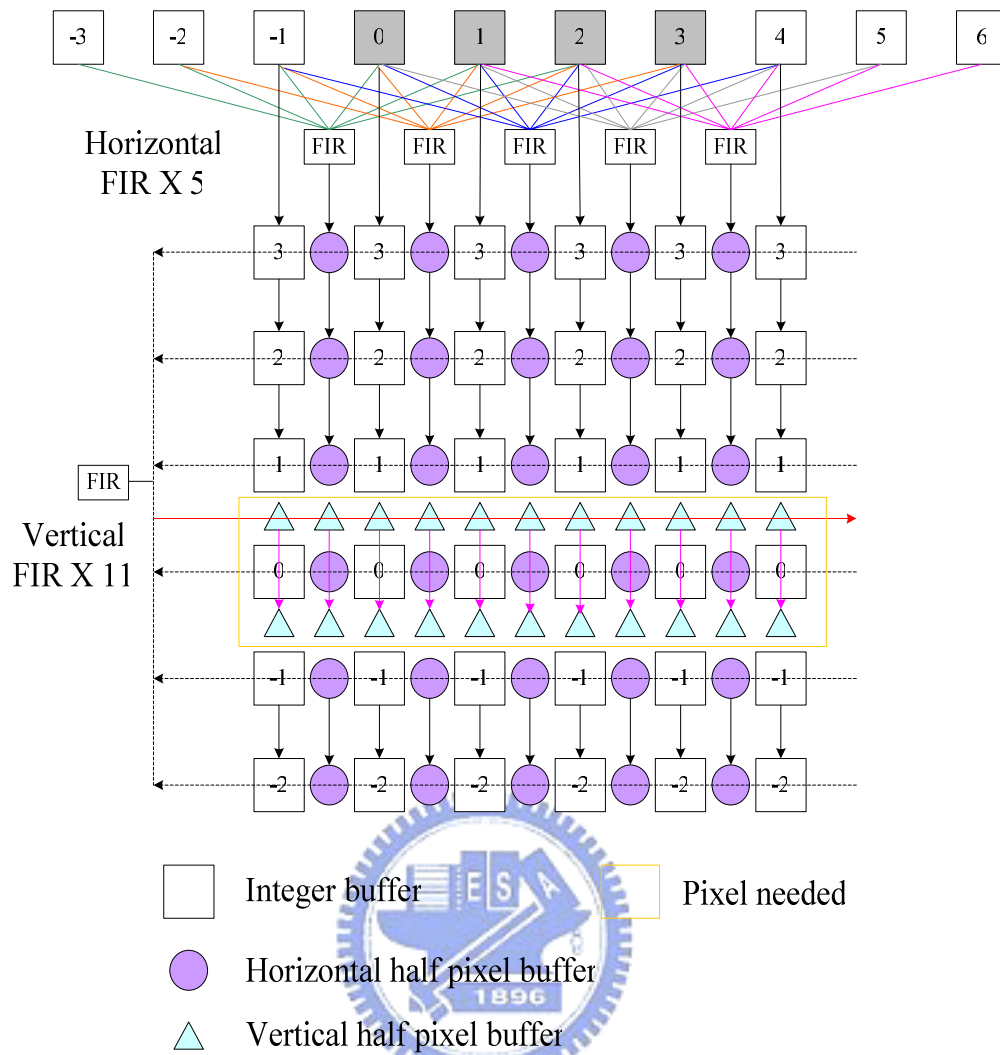


Fig 25 Interpolation unit

As shown in Fig 25, the 6-tap 2-D FIR filter is divided into two directions (horizontal and vertical) 1-D FIR filter which is shown in Fig 24(b). First, we interpolate the horizontal half pixels by five FIR filters from 10 adjacent integer pixels. These five intermediate values and six integer pixels are stored and shifted cycle by cycle in the interpolation buffer. We use the same way to interpolate the vertical half pixels with 11 FIR filter. In our algorithm, since we will not visit the entire positions in the whole refinement window, some redundant interpolations appear in certain pixels in the quarter precision.

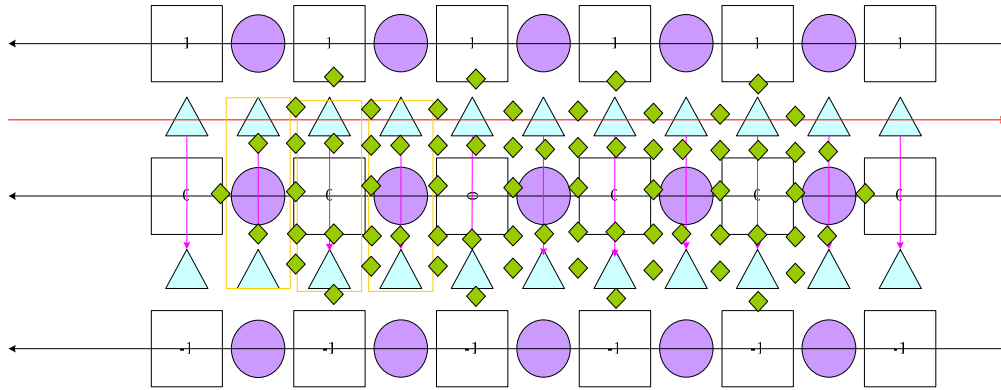


Fig 26 Bilinear filters of interpolation unit.

Only the rhombuses in the Fig 26 are the required bilinear filters. To avoid the redundant interpolate operation, we remove those redundant bilinear filters that is reduced from 106 (no positions skipped) to 68 (no positions redundant). Thus, the 36% of bilinear filters that each includes an adder and a shifter can be saved by using the proposed algorithm.

Because of the irregular search pattern used in second step, the adaptive selection should be done before the pixels sent into PU. That is one of the overhead by applying fast FME algorithm. The others are the early termination unit and compare unit. In the former one, the way to predict threshold is the same but different in check time. In the later one, we should know not only the best position but also second and third places.

Mode decision is combined with comparator shown in Fig 23. MB header related information included motion vector, reference frames and type of block sizes are sent into the compare and determination unit for the Lagrangian mode decision. The information of the first step is sent into selection unit to choose the input of the next step. At the same time, the final cost is checked by early termination unit to judge the refine process should be skipped or not.

6.4 PERFORMANCE ANALYSIS

With the little modification of fast algorithm, the quality loss and speed up of the hardware design are shown in Table 19. Due to the decreased probability for early termination, we may slower encoding speed but get smaller PSNR drop.

QP = 28		Stefan	Mobile	Foreman	Coastguard	News
ref. software	bit rate	1441.14	1888.69	498.62	1127.87	223.72
	PSNR	35.36	33.75	36.24	34.52	38.12
	time (sec)	491.604	471.993	496.974	488.039	450.37
proposed	bit rate	1475.09	1940.28	508.88	1142.9	227.35
	PSNR	35.29	33.68	36.19	34.49	38.02
	time (sec)	220.261	219.782	220.254	222.32	211.988
	Δ bit rate(%)	2.35577	2.73152	2.057679	1.3326	1.62256
	Δ PSNR	-0.07	-0.07	-0.05	-0.03	-0.1
	speed up	2.23192	2.14755	2.256368	2.19521	2.12451

Table 19 Performance analysis after algorithm modification

5.5 IMPLEMENTATION RESULT

Due to the 4X4 block decomposition and the adaptive search pattern, the control unit is the most challenge part of the whole design. We implement this part with finite state machine. The proposed FME architecture for H.264 is implemented by Verilog and synthesized in UMC 0.18u technology at 100MHz. The details of every part are listed in Table 20. The latency per MB can be calculated as follows if all 41 modes do the FME.

	Gate Count
Interpolation Unit	15436
Selection Unit	4933
PU x 5	21335
Control	349
Compare and Determine	4658
Early Termination	1354
Total	48065

Table 20 Implementation result of proposed architecture

Latency per macroblock is added by latencies of seven different block size respectively and the re-calculate stage. To avoid so large the register area, we decide to calculate the residual data after we have know the most suitable block size for current macroblock, the following equation shows the detail:

$$\text{Latency per MB} = [21 \times 16 + 21 \times 2 \times 8 + 29 \times 8 + 29 \times 2 \times 4 + 29 \times 4 \times 2 + 45 \times 2 \times 2 + 45 \times 4] + [17 \times 16] = 2000 \text{ cycles}$$

For such case, our design can process 50K MB/sec in 100MHz and is sufficient to support SDTV format in 30Hz for one reference frame. When compared with other design [51], our design has slight quality loss but 14% faster and 40% smaller.

	Architecture in [51]	Propose
△bit rate (%)	0	2.02003
△PSNR (dB)	0	-0.064
Operating clock	100MHz	100MHz
Largrangian mode decision	Support	Support
Gate count (total)	79372	48065
Time to refine MV	T	0.86T
MB/sec	49K	50K
Technology	UMC 0.18u	UMC 0.18u

Table 21 Comparison between the proposed architecture and architecture in [51]

Chapter 7 Conclusion

7.1 SUMMARY

The point proposed in the paper can be mainly summarized into three parts:

7.1.1 Fast integer motion estimation

7.1.1.1 Search range determination part

Too large matching error means can not find a good match position in the whole search window. In this case, we can reduce the search range because the magnitude of search range does not matter. Too low matching error means perfect match. In this case, we can also reduce the search range.

7.1.2 Fast fractional motion estimation

7.1.2.1 Search pattern part

Higher probability the best position will fall near the search center, so we use center bias search pattern.



7.1.2.2 Early termination part

The system order can not be changed, so we use SAD from integer motion estimation to predict the SATD threshold of fractional motion estimation.

7.1.3 Architecture design of fractional motion estimation

7.1.3.1 Area reduction part

We use parallelism technique for hardware implementation. The search points in the same step of our proposed algorithm decrease, so we do not have to calculate as many points as the origin simultaneously. We can reduce the process elements duplicate for parallelism.

7.1.3.2 Latency reduction part

We modify early termination check time from point to step. Once the matching criterion is satisfied, the second step process can be skipped and certainly result in shorter refined time.

7.2 PERFORMANCE ANALYSIS

7.2.1 Fast integer motion estimation

We can reach the biggest speed up by applying fast algorithm for integer motion estimation. From our simulation result, we can get average 9/12.5 times speed up with input search range equal to 16/32. We have less than 1% bit rate increase and degrade PSNR only 0.02 dB.

7.2.2 Fast fractional motion estimation

In this part, the portion of search point reduction is fixed and early termination does not happen very often. So we can only reach 2.25 times speed up. Besides this, we have almost 2% bit rate increase and degrade PSNR 0.1 dB.

7.2.3 Architecture design of fractional motion estimation

We propose the fractional motion estimation architecture with smaller area cost and shorter refined time than architecture proposed in [51]. We save almost 40 % area cost and achieve 1.15 times speed up. Besides this, we have average 2 % bit rate increase and degrade PSNR 0.064 dB.

7.3 FUTURE WORK

One of integer or fractional motion estimation apply fast algorithm mentioned in chapter 3 and chapter 4 and the other remain the same as original method shows tolerable performance loss. But when we applied fast algorithms for both integer and fractional motion estimation, the experiment result shows inferior R-D performance, especially in lower resolution sequence. We guess the main reason is both of our fast algorithms are not accurate enough. Improvement of R-D performance is needed if we want to use fast algorithms for both motion estimation modules.

Hardware implementation is completed only in fast fractional motion estimation part. For fast integer motion estimation, the architecture design is very straight forward. The calculation core is full search systolic array and only the control unit and scheduling timing should be redesigned. The area of the design depends on how large the systolic array is. The larger systolic array comes more data reusability and hardware utilization. The main advantage of this fast algorithm architecture is to shorten the searching time. We can flow less data when the search range becomes smaller.

BIBLIOGRAPHY

- [1] ISO/IEC 13818, Information Technology-Generic Coding of Moving Pictures and Associated Audio Information, 2000.
- [2] ISO/IEC 14496-2, Coding of Audio-Visual Objects – Part2 : *Visual*, 2001..
- [3] ISO/IEC 14496-10 and ITU-T Rec. H.264, *Advanced Video Coding*, 2003.
- [4] ITU-T Rec.H.264, ISO/IEC 14496-10 “Advanced video coding”, *Final Draft International Standard, JVT-G050r1, Geneva, Switzerland, May 2003*.
- [5] T. Wiegand, G. J. Sullivan, G. Bjontegaad, and A. Luthra, “Overview of the H.264/AVC video coding standard”, *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560-575, July 2003.
- [6] C.-L. Yang, L.-M. Po, and W.-H. Lam, “A fast H.264 intra prediction algorithm using macroblock properties,” in *Proc. ICIP*, vol. 1, pp. 461 – 464, Oct. 2004
- [7] Y.-K. Lin and T.-S. Chang, “Fast block type decision algorithm for intra prediction in H.264 FRext,” in *Proc. ICIP*, Oct. 2005
- [8] B. Meng, O.C. Au, C.-W. Wong, and H.-K. Lam, “Efficient intra-prediction algorithm in H.264,” in *Proc. ICIP*, vol. 3, pp. 837-840, Sept. 2003.
- [9] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, G. N. Feng, D. J. Wu, and S. Wu, “Fast mode decision algorithm for JVT intra prediction,” *JVT-G013, 7th JVT Meeting, Pattaya, Thailand, March 2003*.
- [10] Y.-D. Zhang, F. Dai, and S.-X. Lin, “Fast 4x4 intra-prediction mode selection for H.264,” in *Proc. ICME*, vol. 2, pp.1151 – 1154, June 2004.
- [11] C. C. Chen, T. S. Chang, “Fast three step intra prediction algorithm for 4x4 blocks in H.264,” in *Proc. ISCAS*, 2005.
- [12] M.-C. Hwang, J.-K. Cho, J.-H. Kim, and S.-J. Ko, “A fast intra prediction mode decision algorithm based on temporal correlation for H.264,” in *Proc. of 2005 Int’l Tech. Conf. on Circuits Systems, Computers and Communications*, vol. 4, pp. 1573-1574, Jeju, July 2005.
- [13] J. Jain and A. Jain, “Displacement measurement and its application in interframe image coding,” *IEEE Trans. Commun.*, Vol.29, (12), pp. 1799–1808, 1981.
- [14] R. Li, B. Zeng, and M.L. Liou, “A new three-step search algorithm for block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, (4), pp. 438–443, 1994
- [15] M. Ghanbari, “The cross-search algorithm for motion estimation,” *IEEE Trans. Commun.*, 38, (7), pp. 950–953, 1990.
- [16] O.T.-C. Chen, “Motion estimation using a one-dimensional gradient descent search,” *IEEE Trans. Circuits Syst. Video Technol.*, 10, (4), pp. 608–616, 2000
- [17] L.-K. Liu and E. Feig, “A block-based gradient descent search algorithm for block motion estimation in video coding,” *IEEE Trans. Circuits Syst. Video Technol.*, 6, (4), pp. 419–422, 1996.

- [18] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, 6, (2), pp. 313–317, 1996.
- [19] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," *IEEE Trans. Image Process.*, 9, (2), pp. 287–290, 2000
- [20] C.-H. Cheung and L.-M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, 12, (12), pp. 1168–1177, 2002
- [21] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, 12, (5), pp. 349–355, 2002
- [22] L. Luo, C. Zou, X. Gao, and Z. He, "A new prediction search algorithm for block motion estimation in video coding," *IEEE Trans. Consumer Electron.*, 43, (1), pp. 56–61, 1997.
- [23] Y.-T. Roan and P.-Y. Chen, "A fuzzy search algorithm for the estimation of motion vectors," *IEEE Trans. Broadcast.*, 46, (2), pp. 121–127, 2000
- [24] J.H. Lee and J.B. Ra, "Block motion estimation based on selective integral projections," *Int. Conf. on Image Processing*, vol. 1, pp. 689–692, Sept. 2002.
- [25] C.-M. Kuo, C.-P. Chao, and C.-H. Hsieh, "A new motion estimation algorithm for video coding using adaptive Kalman filter," *Real-Time Imaging*, 8, pp. 387–398, 2002
- [26] J. Chalidabhongse and C.-C.J. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans. Circuits Syst. Video Technol.*, 7, (3), pp. 477–488, 1997
- [27] H.-S. Wang and R.M. Mersereau, "Fast algorithms for the estimation of motion vectors," *IEEE Trans. Image Process.*, 8, (3), pp. 435–438, 1999
- [28] K. Lengwehasatit and A. Ortega, "Probabilistic partial-distance fast matching algorithms for motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, 11, (2), pp. 139–152, 2001
- [29] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Process.*, 4, (1), pp. 105–107, 1995
- [30] S.-M. Jung, S.-C. Shin, H. Baik, and M.-S. Park, "New fast successive elimination algorithm," *Proc. 43rd IEEE Midwest Symp. on Circuits and Systems*, vol. 2, pp. 616–619, Aug. 2000
- [31] X.Q. Gao, C.J. Duanmu, and C.R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Process.*, 9, (3), pp. 501–504, 2000
- [32] S.-M. Jung, S.-C. Shin, H. Baik, and M.-S. Park, "Efficient multilevel successive elimination algorithms for block matching motion estimation," *IEE Proc., Vis., Image Signal Process.*, 149, (2), pp. 73–84, 2002
- [33] M. Yang, H. Cui, and K. Tang, "Efficient tree structured motion estimation using successive elimination," *IEE Proc. Vis., Image Signal Process.*, Vol. 151, No. 5, Oct. 2004
- [34] Bo Zhou, Jian Chen, "A fast two-step search algorithm for half pixel motion estimation", *Electronics, Circuits and Systems, 2003. ICECS. Proceedings of the 10th IEEE International Conference on*, 2003 Volume: 2, Pages: 611 – 614 Vol.2.

- [35] H.-M. Wong, O. C. Au, and A. Chang, "Fast sub-pixel inter-prediction – based on the texture direction analysis," *Proc. IEEE International Symposium, Circuits and Systems*, Oct. 2005.
- [36] C.-C. Cheng, Y.-J. Wang, and T.-S. Chang, "A fast fractional pel motion estimation algorithm for H.264/AVC," in *Proc. VLSI/CAD Conf.*, 2005.
- [37] Z. Chen, P. Zhou, and Y. He, "Fast motion estimation for JVT", *JVT G-016*, 2003
- [38] X. Yi, J. Zhang, N. Ling, and W. Shang, "Improved and simplified fast motion estimation for JM," *JVT P-021*, Oct. 2005
- [39] T. Koga, K. Iinuma, A. Hirano, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing" in *National Telecommunications Conferences*, pp. G5.3.1-G5.3.5, 1981
- [40] L.W.Lee, J.F.Wang, J.F.Lee, and J.D.Shie, "Dynamic search window adjustment and interlaced search for block-matching algorithm" *IEEE Transactions on Circuits and Systems for Video Technology*, vol.3, pp. 85-87, February 1993.
- [41] R. srinivasan and K. Rao, "Predictive coding based on efficient motion estimation" *IEEE Transactions on Communications*, vol. COM-33, pp. 888-896, August 1985.
- [42] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors" *IEEE Transactions on Circuits and Systems for Video Technology*, vol.3, pp. 148-157, April 1993.
- [43] K.-H. Han and Y.-L. Lee, "Fast macroblock mode determination to reduce H.264 complexity," *IEICE Trans. Fundamentals*, Vol.E88–A, 3, pp.800-804, March 2005
- [44] J. Lee and Y. Jean, "Fast mode decision for H.264", LG Electronics Inc, Digital media research laboratory
- [45] Z. Zhou and M.-T. Sun, "Fast macroblock inter mode decision and motion estimation for H.264/MPEG-4 AVC," *IEEE International Conference on Image Processing*, Oct. 2004.
- [46] P. Yin, A. M. Towropes, and J. Boyce, "Fast mode decision and motion estimation for JVT/H.264," *Proc. ICIP*, pp.853-856, 2003
- [47] A. C. Yu and G.R. Martin, "Advanced block size selection algorithm for inter frame coding in H.264/MPEG-4 AVC," *Proc. ICIP*, pp. 95-98, 2004
- [48] C. Grecos and M.Y. Yang "Fast inter mode prediction for P Slices in the H264 video coding standard," *IEEE Trans on Broadcasting*, Vol. 51, 2, pp.256-263, June 2005.
- [49] *Texas Instruments, TMS320C6000 Programmer Guide, 2001.*
- [50] S.-W. Wang, Y.-T. Yang, C.-Y. Li, Y.-S. Tung, and J.-L. Wu, "An optimization of H.264/AVC baseline decoder on low-cost TriMedia DSP processor", *Proc. of 49th SPIE Annual Meeting*, 2004.
- [51] T. C. Chen, Y.-W. Huang, and L. G. Chen, "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC" in *Proc. of ICASSP*, vol. 4, pp.9-12, 2004.
- [52] *JM8.2, Reference Software of JVT.*
- [53] T. C. Wang, Y. W. Huang H. C. Fang, and L. G. Chen, "Parallel 4x4 2D transform and inverse transform architecture for MPEG-4 AVC/H.264," in *Proc. of ISCAS*, 2003.

APPENDIX

This section gives the experiment inter-media result. Even the methods tried in this section were not implemented in my design; they are still reported in order to offer the readers more reference material. Only the sub-pel correlated results are listed below.

PARAMETER DEFINITION:

AVG POINT:

This index means the average search points needed for sub-pel motion estimation. The result of first step will lead to different search pattern next step. To make the dispassion observation, we combined the probability concern into the index. For example, if 5 points needed in first step, the result will 90% fall on the search center (case1) and 10% fall on one of the other points (case2). Then average points needed are equal to

$$\text{AVG POINT} = (\text{points needed in first step}) + 0.9 * (\text{points needed in case1}) + 0.1 * (\text{points needed in case2})$$

By using this approximation, we may get more accurate comparison of speeding up.

PSNR:

The index is the video quality degradation. Only luminance PSNR is listed. The simulation result is the average of four different input sequences as Stefan, Foreman, Mobile and news.

Bit rate:

The index is the transmit bandwidth changing percentage.

Research Report

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel

Gray point (rectangle): the second step search position in quarter pixel.

First we visit the five circle points shown in Fig 27. When the half pixel best position falls on the search center (the initial position before refinement regard as the integer pixel position), we will find the eight points surrounding the search center. When the half pixel best position fall on one of the four end points in step one, we will use horizontal and vertical search pattern according to the best position determined in previous step.

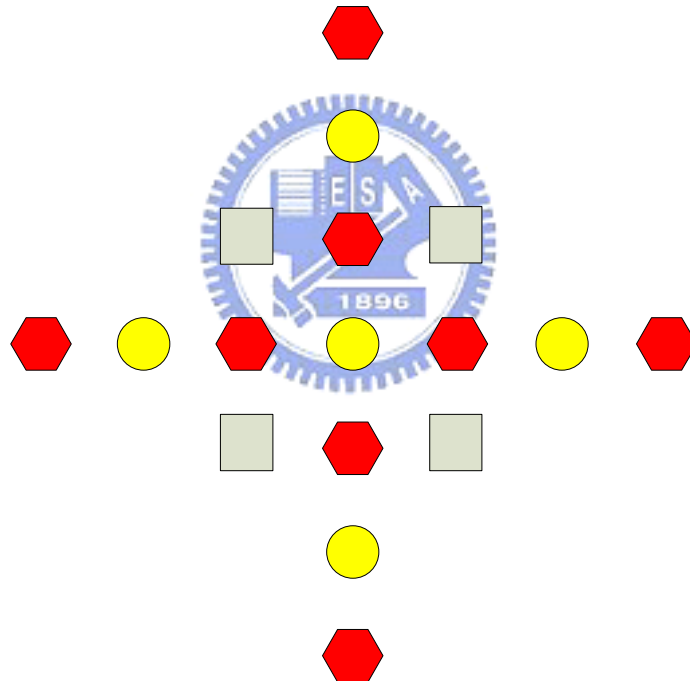


Fig 27 Pattern 1

Simulation Result:

AVG POINT	12.4
PSNR (dB)	-0.08
Bit rate (%)	3.767

Conclusion:

Too much quality loss and not significantly speeding up.

Algorithm

Yellow point (circle): the first step search position

Red point (hexagon): the second step search position

First we visit the five circle points shown in Fig 28. When the half pixel best position falls on the search center (the initial position before refinement regard as the integer pixel position), we will find the eight points surrounding the search center. When the half pixel best position falls on one of the four end points in step one, we will use four different directional triangle patterns according to the best position determined in previous step.

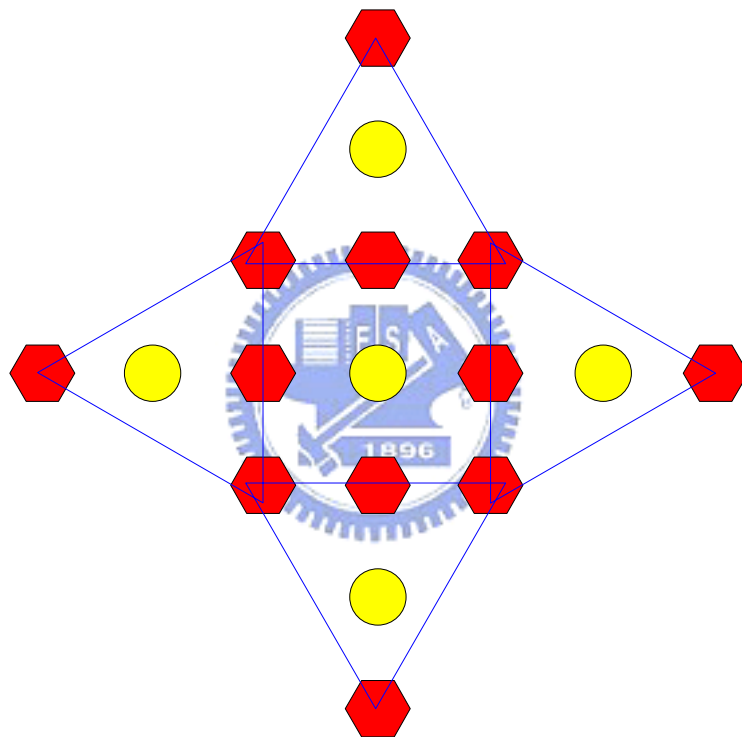


Fig 28 Pattern 2

Simulation Result:

AVG POINT	12.6
PSNR (dB)	-0.043
Bit rate (%)	2.221

Conclusion:

Quality loss is acceptable but not significantly speeding up.

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel.

First we visit the five circle points shown in Fig 29. No matter where the half pixel best position in step one fall, we still find the cross pattern in step two.

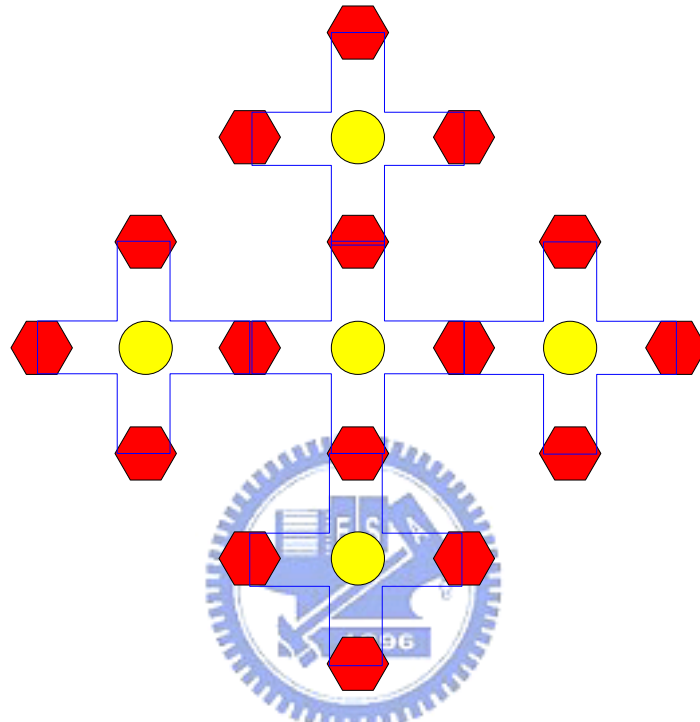


Fig 29 Pattern 3

Simulation Result:

AVG POINT	9
PSNR (dB)	-0.043
Bit rate (%)	2.388

Conclusion:

Quality loss and speeding up are acceptable up but the method is proposed before (used in original x264).

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel.

First we visit the five circle points shown in Fig 30. When the half pixel best position falls on the search center (the initial position before refinement regard as the integer pixel position), we will find cross four points surrounding the search center. When the half pixel best position falls on one of the four end points in step one, we will use four different direction triangle patterns according to the best position determined in previous step.

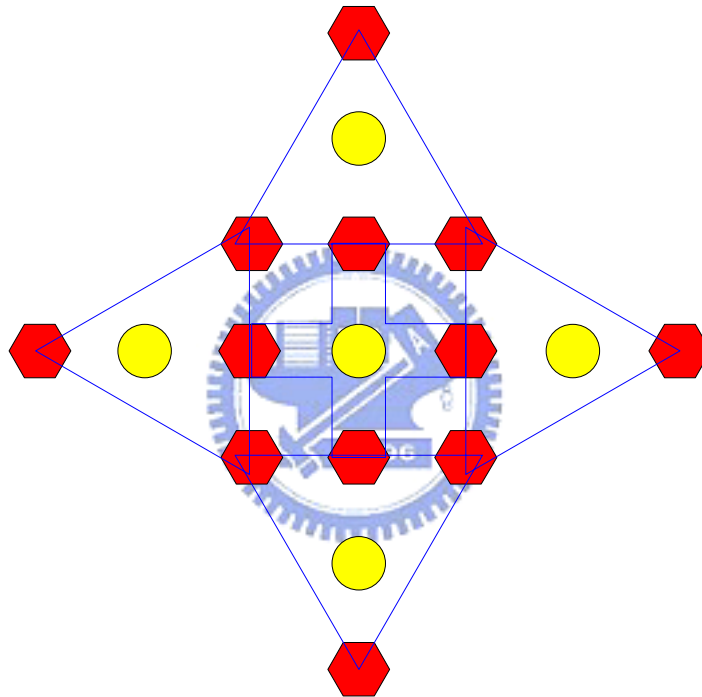


Fig 30 Pattern 4

Simulation Result:

AVG POINT	9
PSNR (dB)	-0.06
Bit rate (%)	3.330

Conclusion:

Quality loss is a little serious and significant speeding up but high overhead complexity will be introduced in this method.

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel.

First we visit the five circle points shown in Fig 31. When the half pixel best position falls on the search center (the initial position before refinement regard as the integer position), we will find eight points surrounding the search center. When the half pixel best position falls on one of the four end points in step one, we will use cross search pattern.

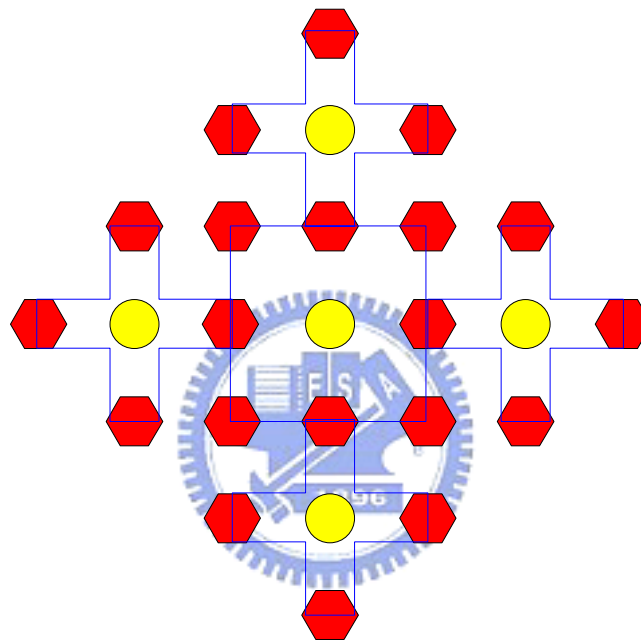


Fig 31 Pattern 5

Simulation Result:

AVG POINT	12.6
PSNR (dB)	-0.023
Bit rate (%)	1.455

Conclusion:

Little quality loss is produced but with not significantly speeding up.

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel.

First we visit the nine circle points shown in Fig 32. We make the most part of effort to find the best half pixel search position. After that, we just need to find only two search positions except two positions appears on the end points on y axis. The two exceptions will visit four cross search points around the best position refined in previous step.

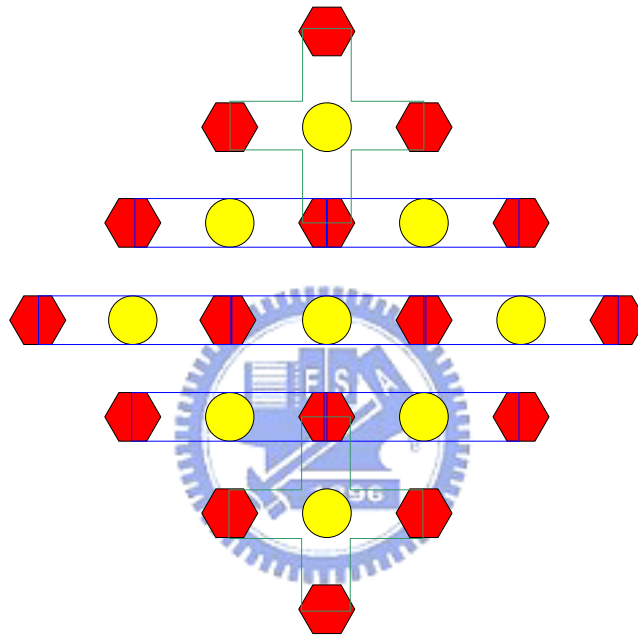


Fig 32 Pattern 6

Simulation Result:

AVG POINT	11.2
PSNR (dB)	-0.046
Bit rate (%)	1.833

Conclusion:

Little quality loss is produced but with not significantly speeding up.

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel.

First we visit the five circle points shown in Fig 33. It is a modification version of dual cross search. When the half pixel best position falls on the search center (the initial position before refinement regard as the integer pixel position), we find cross pattern with one x one y axis distance. When the half pixel best position falls on the y axis end points, we find cross pattern with two x one y axis distance. Besides, we find cross pattern with one x two y axis distance.

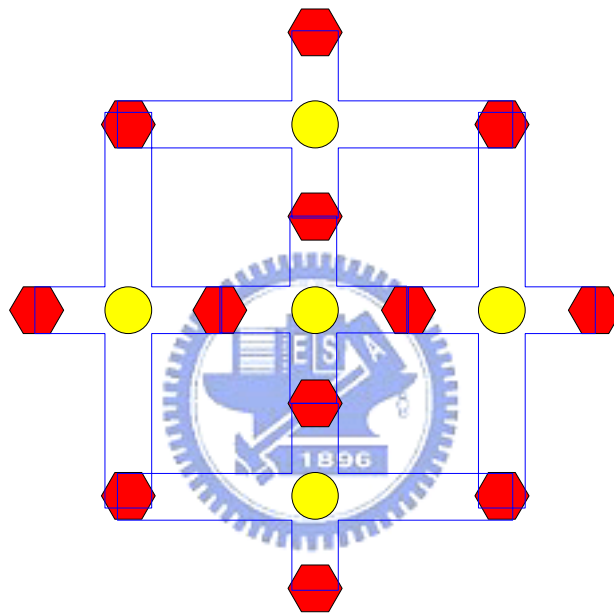


Fig 33 Pattern 7

Simulation Result:

AVG POINT	9
PSNR (dB)	-0.09
Bit rate (%)	4.092

Conclusion:

Quality loss is serious but good speeding up.

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel.

First we visit the five circle points shown in Fig 34. It is a modification version of dual cross search. When the half pixel best position falls on the search center (the initial position before refinement regard as the integer pixel position), we find eight points surrounding the search center. When the half pixel best position falls on the y axis end points, we find cross pattern with two x one y axis distance. Besides, we find cross pattern with one x two y axis distance.

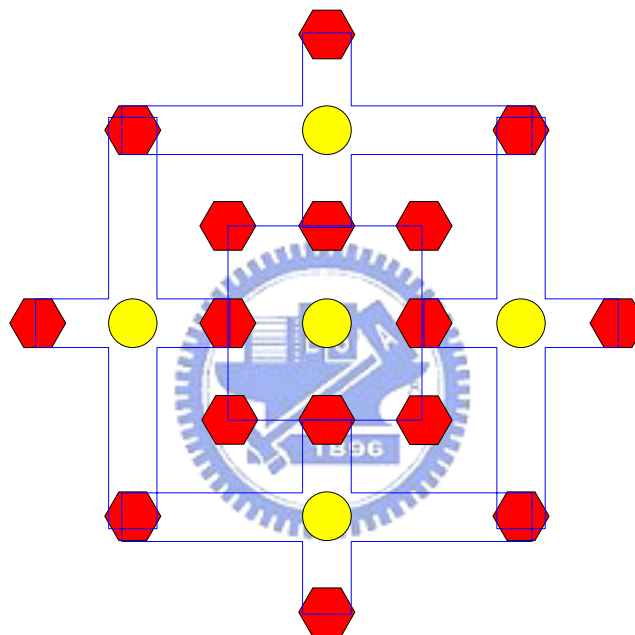


Fig 34 Pattern 8

Simulation Result:

AVG POINT	12.6
PSNR (dB)	-0.056
Bit rate (%)	2.889

Conclusion:

Quality loss is a little serious and not significantly speeding up.

Algorithm

Yellow point (circle): the first step search position in half pixel.

Red point (hexagon): the second step search position in quarter pixel.

First we visit the five circle points shown in Fig 35. When the half pixel best position falls on the search center (the initial position before refinement regard as the integer pixel position), we will find four cross points around the search center. When the half pixel best position falls on one of the four end points in step one, we will use horizontal and vertical search pattern according to the best position determined in previous step.

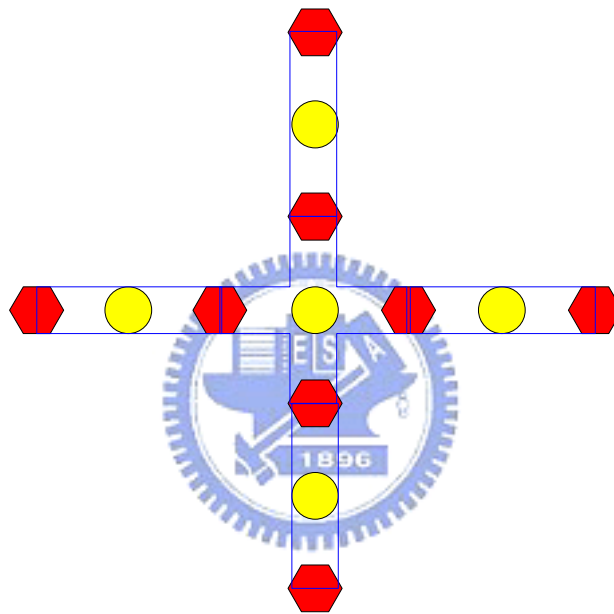


Fig 35 Pattern 9

Simulation Result:

AVG POINT	8.4
PSNR (dB)	-0.116
Bit rate (%)	5.469

Conclusion:

Quality loss is very serious but good speeding up. Extra low area can be introduced by applying this architecture; the requirement of register array in interpolation unit can significantly decrease.

作者簡歷

姓名：王裕仁

籍貫：台北市

學歷：

台北市立建國高級中學 (民國 86 年 09 月 ~ 民國 89 年 06 月)

國立交通大學電子工程學系 學士 (民國 89 年 09 月 ~ 民國 93 年 06 月)

國立交通大學電子所系統組 碩士 (民國 93 年 09 月 ~ 民國 95 年 06 月)

獲獎紀錄：

[1] 九十三學年度 大學院校積體電路設計競賽(IC Contest)

研究所/大學部 標準單元式設計組(Cell-based) 佳作

[2] 九十四學年度 大學院校積體電路設計競賽(IC Contest)

研究所/大學部 標準單元式設計組(Cell-based) 特優

著作：

[1] Chao-Chung Cheng, Yu-Jen Wang, Tian-Sheuan Chang, 'A Fast Fractional Pel Motion Estimation Algorithm for H.264/AVC' VLSI/CAD, 2005

[2] Hung-Chih Lin, Yu-Jen Wang, Kai-Ting Cheng, Shang-Yu Yeh, Wei-Nien Chen, Chia-Yang Tsai, Tian-Sheuan Chang, Hsueh-Ming Hang, 'Algorithms and DSP Implementation of H.264/AVC' ASP-DAC, 2006

[3] Yu-Jen Wang, Chao-Chung Cheng, Tian-Sheuan Chang, 'A Fast Fractional Pel Motion Estimation Algorithm for H.264/MPEG-4 AVC' ISCAS, 2006

[4] Yu-Jen Wang, Chao-Chung Cheng, Tian-Sheuan Chang, 'A Fast Fractional Pel Motion Estimation Algorithm and Hardware Implementation for H.264/AVC' CSVT, 2006 (minor revision)