國立交通大學

資訊科學系

碩士論文

架構於H.264/AVC之有效率區塊大小模式決策演算法

An Efficient Algorithm of Variable Block-Size Mode Decision for H.264/AVC

研究生：邵育睿

指導教授：陳玲慧　教授

中華民國九十四年六月

國立交通大學

資訊科學系

碩士論文

架構於H.264/AVC之有效率區塊大小模式決策演算法

An Efficient Algorithm of Variable Block-Size Mode Decision for H.264/AVC

研 究 生：邵育睿

指導教授：陳玲慧　教授

中華民國 九十四 年六月

架構於H.264/AVC之有效率區塊大小模式決策演算法

An Efficient Algorithm of Variable Block-Size Mode Decision for H.264/AVC

研究生：邵育睿　　　Student：Yu-Ruei Shao

指導教授：陳玲慧　　　Advisor：Ling-Hwei Chen

國立交通大學

資訊科學系

碩士論文

A Thesis
Submitted to Institute of Computer and Information Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master

in

Computer and Information Science

June 2005
Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

An Efficient Algorithm of Variable Block-Size Mode Decision for H.264/AVC

Yu-Ruei Shao and Ling-Hwei Chen

Department of Computer and Information Science, National Chiao Tung University

1001 Ta Hsueh Rd., Hsinchu, Taiwan 30050, R.O.C.

Abstract

The new video coding standard, H.264/MPEG-4 Part10, provides some new features to get average bit rate reduction of 50% compared to any other standard under given fixed fidelity. One of these features is to allow using variable block-size motion compensation with small block sizes. In the reference software JM8.2, the exhaustive search for variable block-size mode selection is adopted, this is very time consuming. Here, we will propose a fast method to find a suitable block size for each macro-block in inter prediction. The proposed method uses some information, such as still area, residue image after motion search, and the relation of motion vectors, to reduce the number of possible block-size modes in the variable block-size mode decision. The experimental result shows that this new scheme could reduce about 50% total encoding time while keeping the same quality and a mere 1.51% bit-rate increase compared to the exhaustive search scheme. Besides, we also compare the efficiency among our method and some exiting methods. The experimental result shows that our algorithm is faster than others.

架構於H.264/AVC之有效率區塊大小模式決策演算法

研究生：邵育睿　　　　　　　　　　指導教授：陳玲慧　博士

國立交通大學資訊科學研究所

摘要

由於近幾年數位電視系統的興起與普及，使得對於視訊影片的品質要求愈來愈高。加上網路的方便性與頻寬限制性，促使了新的視訊壓縮標準H.264/AVC的制定。在新的標準H.264中，雖仍延用預測、轉換、量化和entropy編碼四大部份，但H.264是在各個部份中都採取許多新技術與觀念，以達到高壓縮效能。相較於過去的標準，H.264節省了50%的空間儲存，但其運算複雜度卻是太高，尤其是在做區塊大小模式決策時。因此，在這篇論文中，我們提出一演算法來改善H.264編碼器的效能。在提出的演算法中，利用了幾項可用資訊，包括了靜態區域、殘餘資料以及運動向量關係此三種，有效的利用此三項資訊可以有效率加快區塊大小模式之決策，以致於達到改進編碼器效能的目的，以降低整個壓縮系統的複雜度。由實驗結果顯示，與暴力法來決定區塊模式的方法比較，本論文提出之演算法可達到增進50%的整體編碼時間，且可不影響視訊影片品質，只約增

加了 1.5%的位元率。除此之外，本論文亦與最近幾年裡已發表的區塊模式決策方法相互比較，並於實驗結果中，可明顯看出本論文提出之方法可更有效的改進模式決策之效能。

誌　謝

首先要感謝我的指導老師 陳玲慧教授在這二年間對我的指導,使我能得到許多的專業知識與技能，還有學習如何待人處事。

同時，還要感謝實驗室的民全學長和萱聖學長在遇到困難時給我的建議與幫助；感謝一起畢業的同學致生、合吉、崇荏、朝君及佳峯，在我需要幫忙時給予我鼓勵及支援；還要謝謝學妹及大學好友的精神上的支持，沒有這些人的幫忙我就無法順利完成這篇論文。此外，感謝口試委員張隆紋教授、鍾國亮教授及尹邦嚴教授給予的指導與建議，讓我的論文能更加完善。

最後，要感謝的是我的父母、家人及女友，由於他們在我背後所給予的支持，使得我能專心致力於我的論文，他們的鼓勵更是我動力的來源，繼續努力的原因，將這篇論文獻給我最親愛的家人。

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In 2001, the ISO Motion Picture Experts Group (MPEG) recognized the potential benefits of H.26L, and the Joint Video Team (JVT) was formed, including experts from MPEG and VCEG (Video Coding Experts Group). In March 2003, the JVT finalized the new video compression standard H.264, also known as MPEG-4 Part 10 AVC (Advanced Video Coding) [1]. H.264 achieves the target that doubling the coding efficiency, reducing about 50% of bit-rate saving, related to previous standards, such as MPEG-1, MPEG-2, H.261, H.263 etc. In fact, H.264 still adopted the hybrid-coding concept that consists of prediction, transformation, quantization and entropy coding part. And the significant achievement of H.264 efficiency is ascribed to new techniques adopted in every part of codec. For examples, directional prediction modes for intra coding, variable block-size motion estimation and compensation (ME/MC) with multi-reference frames in prediction part, in-the-loop deblocking filter to remove the blocking artifacts in post-process, a variety of DCT transformation (Integer DCT), context-based adaptive binary arithmetic coding (CABAC) in entropy coding part and so on. But coming with these improvements is the computational complexity that is higher than before, especially the ME/MC part is the most critical factor that influences the coding performance. And one of the important factors causes the heavy computation in this part is the permission of using variable block-size (VBS) motion estimation [2].

The VBS ME allows different macroblocks to use different block-size (also be called partition) according to different motion conditions and object shapes. The block-sizes are from 4x4 to 16x16. For a larger block-size mode, it needs less number of bits to represent the motion vectors, reference frames and the types of block-size mode; however, there will be higher residual energy need to be represented. With the smaller block-size mode, it contains lower residual energy after ME/MC, but it needs more bits to represent the motion vectors and reference frames that indicate the motion of each block. Therefore, the choices of the block-size modes influenced the encoding efficiency.

Due to the benefit coming with the adoption of VBS, H.264 supports 7 types of block-size mode and it can be known as a hierarchical tree structure [3]. They are 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4, and can be classified into two levels, macroblock-level (MB-level) and sub-macroblock-level (subMB-level) [3]. As shown in Fig. 1, MB-level contains 16x16, 16x8, 8x16, and 8x8 modes, and subMB-level includes 8x8, 8x4, 4x8 and 4x4 modes. Note that the number in each sub-block stands for the encoding order in the whole block. The tree-structure means that only when MB is divided into four 8x8 blocks, the modes of subMB-level could be considered. That is, when a 16x16 block is divided into four 8x8 blocks, each 8x8 block can be further divided into 8x4, 4x8 or 4x4. However, when one MB is divided into two 16x8 (or 8x16) blocks, each 16x8 block could not be further divided into 8x8, 8x4, 4x8 or 4x4.

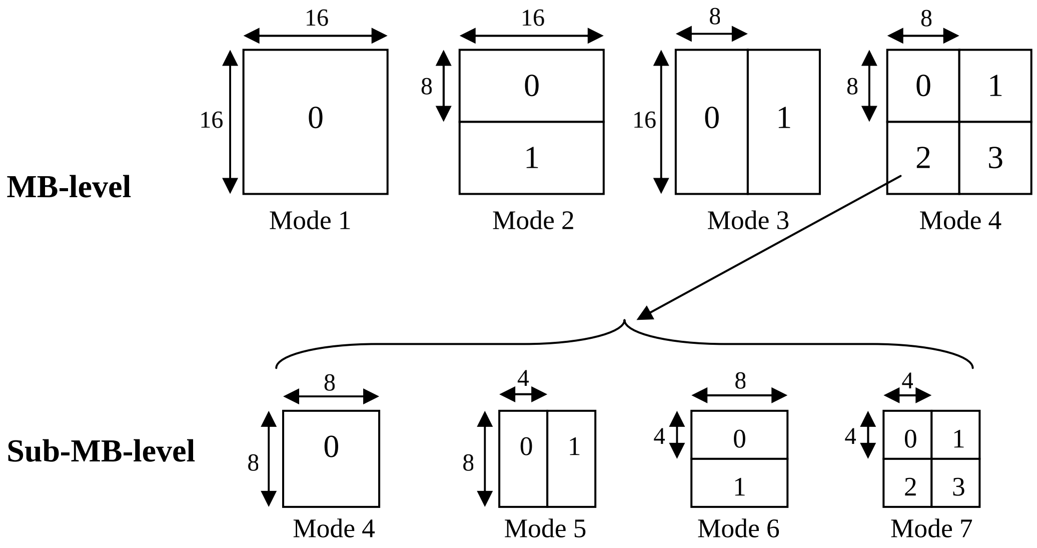


Fig. 1 Block-size modes of MB for ME/MC. Top: MB-level block-size modes. Bottom: subMB-level block-size modes.

In order to get the advantages from adopting VBS in H.264, for an MB, the encoder in reference software JM exhaustively search all available modes to find the best one. This takes a lot of time. Thus, developing a method to reduce the searching time is necessary.

Up to now, several methods have been proposed to speed the searching time. In 2004, Yu [4] proposed an algorithm which uses the DCT coefficients to determine if a block is detailed or not for reducing the computational overhead for inter-frame coding. Yu's strategy classifies all MBs into three groups. If a MB is detected to be homogeneous, then it belongs to Group 1, and only 16x16 block-size mode is allowed for motion estimation. Otherwise, if more than two 8x8 blocks in MB are homogeneous, then the MB belongs to Group 2, and 16x16, 16x8 and 8x16 modes are its possible modes. All other MBs belong to Group 3, and all seven modes are possible modes. Yu's algorithm saves about 27.19% computational time, but has about 3.15% bit-rate increasing in average.

In May, 2004, Wu et al. provided a new decision scheme [5], which detects still or homogeneous areas, to reduce the number of candidate modes. The algorithm uses the sum of absolute differences (SAD) with a self-defined threshold to find out those MBs satisfying the "stillness" conditions, then set the best mode to 16x16 and skips other block-size modes. On the other hand, they use the edge-map constructed by "Sobel" operator to determine if an MB a block is homogenous or not. If an MB belongs to a smooth area, only 16x16, 16x8 and 8x16 modes used to do ME, then, the mode with minimum cost is selected as the best mode. Otherwise, if an MB is non-homogeneous, further test is conducted, that is, check if it is homogeneous for each 8x8 sub-block. If a 8x8 block satisfies the homogeneous criterion, set the 8x8 block's mode to be 8x8, otherwise, examines all possible modes for this 8x8 block. Finally, comparing the costs of all estimated modes and finding out the best mode. Wu's method saves about 29.78% computational time with 0.65% in average of dropping ratio of bit-rate.

In August 2004, Jing [6] uses the mean of absolute differences (MAD) of frames and

MBs to find homogeneous area to improve coding efficiency. If an MB locates in a homogeneous area, then encoder do ME only using 16x16, 16x8 and 8x16 modes. Otherwise, encoder estimates all block-size modes, and set the best mode to the mode with minimum cost. The method saves about 33.8% computational time.

All of the above-mentioned methods only focus on still and homogeneous area. However, if the video without these two features, the efficiency of existing methods will be worse and constricted. For examples, for those videos with camera-motion, most MBs on background area are moving and the exhaustive search is conducted to get their proper block modes. However, for each of them, we can usually find a very similar MB in the previous frame.

In this thesis, we will propose a block-size mode decision algorithm, which will not only improve the performance of H.264 encoder, but also keep the similar bit-rate and quality. The algorithm is based on some facts to reduce the number of candidate modes for a MB encoding. The first fact is that an MB in a “still” area can usually find an almost same MB on the reference frame, then, we can directly use 16x16 or larger block-size modes to do ME. To find this kind of MBs, the difference between the current MB and the co-located MB in the reference frame is evaluated to decide if this MB is still or not. The second fact is that some video sequences have simple camera translation motion or contain rigid objects with steady motion, and for those MBs located in these kinds of areas, we can find almost the same MB in the reference frames. Thus we can directly apply 16x16 block-size to do ME. The last fact is that for a block, when its sub-blocks have similar motion vectors, this stands for smaller sub-blocks can not get better estimation. Thus, the block will not be further divided into sub-blocks. Based on the above-mentioned facts, the proposed algorithm can speed up the variable block-size mode selection and is suitable for majority of video types. To show the effect of the proposed methods, we test several different types of video sequences using the reference software provided by JVT and our method results show that we could speed-up about 50% of total encoded time comparing to the exhaustive search for block-size modes used in the reference software, and kept similar quality of encoded sequence with a little bit-rate increasing. Besides, we also take a comparison between our proposed method and some exiting methods.

In Chapter 2, we will present the proposed method. In Chapter 3, some experimental results will be given, the comparison between our method and other exiting methods introduced in this chapter will also be provided to show the effectiveness of the proposed method. The last chapter will make a final conclusion.

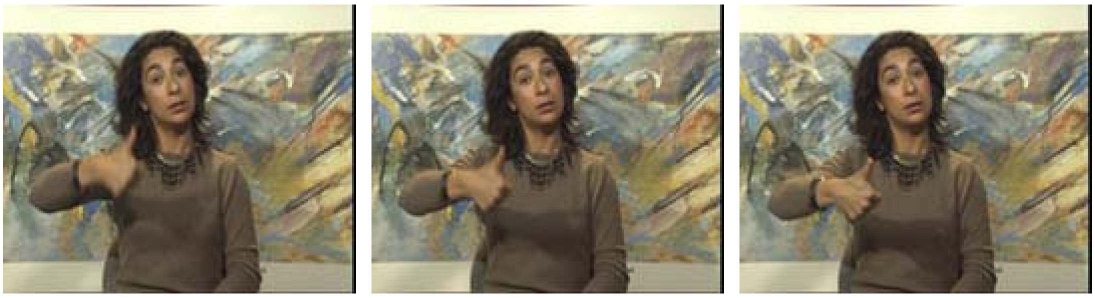
CHAPTER 2

THE PROPOSED METHOD

Three processes are used in the proposed algorithm: still area detection, residual image judgment and motion-vectors relation analysis. In the following, we will describe these processes.

2.1 Still area detection

Fig. 2 shows three consecutive frames (55th – 57th) of a video sequence called “Silent”. From this figure, we can see that these three frames have a common still background area. Thus, each MB located in the background area can be well estimated by the MB located in the same place of the previous frame. Based on this fact, we first use the mean of absolute differences (MAD) between two MBs to extract those MBs in the still area and set their block-size mode to {16x16} or {16x16, 16x8, 8x16} block-size according to difference conditions.

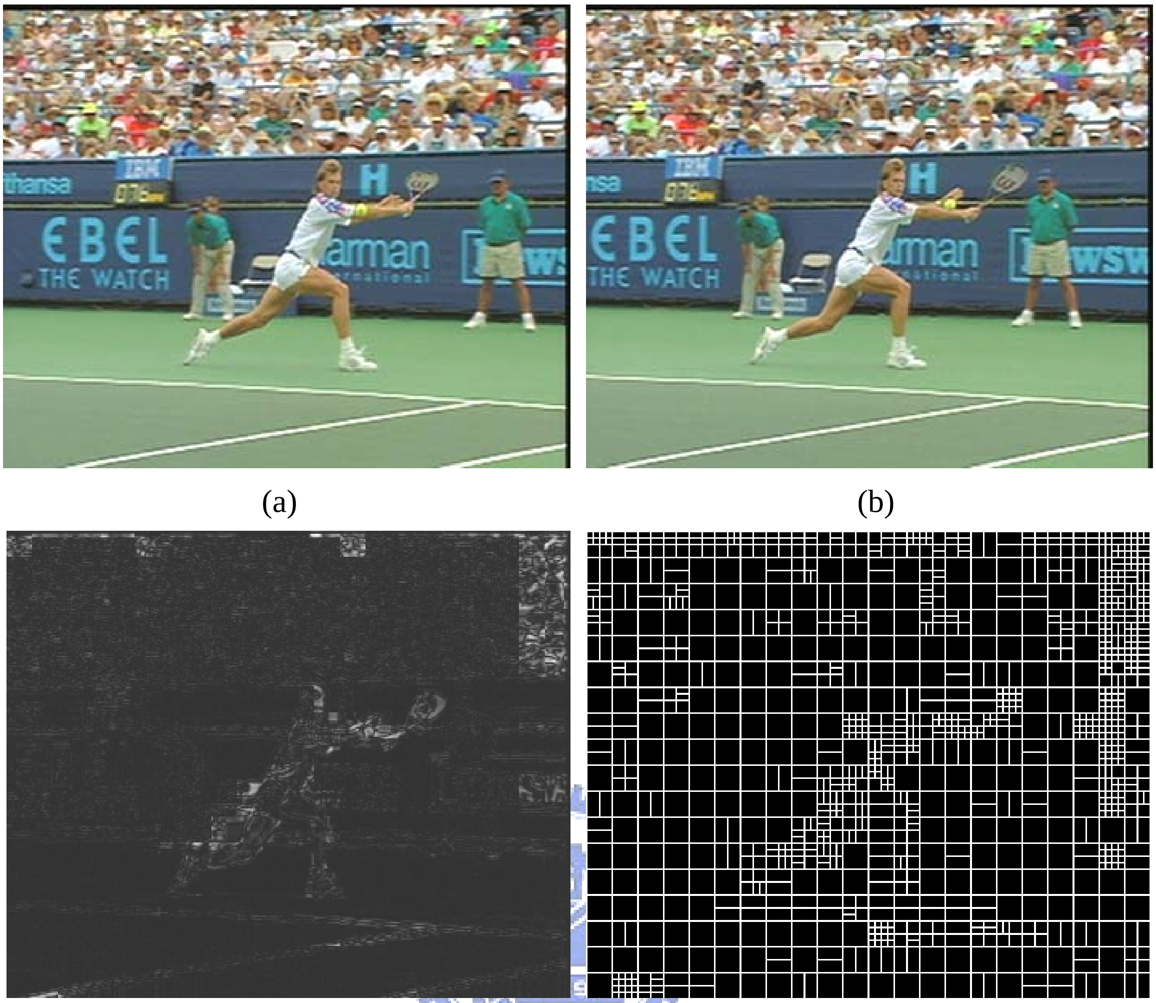


(a) (b) (c)

Fig. 2 “Silent” sequence (QCIF format). (a) The 55th frame. (b) The 56th frame. (c) The 57th frame.

2.2 Residual image judgment

For a video sequence, background may be moving due to camera motion, and some objects in the video sequence may move steadily. This kind of area appearing in a frame will usually appear in a contiguous area of its neighboring frames. Thus, if a MB is located in this kind of area, we can find a very similar MB in the reference frame. To find this kind of MBs, a residual image is created. For each MB in the current frame, the exhaustive search using 16x16 block-size is applied to find the best matching MB in the reference frame, the difference between the current MB and its best matching MB forms a residual MB. All these residual MBs form a residual image. For example, Fig. 3 (a) and Fig. 3 (b) show the 183th and 184th frame in the "Stefan" sequence, and Fig. 3 (c) shows the residual image that is produced by the ME using 16x16 block-size. And for the purpose of presentation, the residual image is produced with normalization to make the pixels with significant value stand out. The gray lines shown in Fig. 3 (d) divide the current frame into blocks of variable size according to the most suitable block-size mode by exhaustive search for VBS modes. We can see that most areas with high difference value contain object boundary or non-steady motion objects, they need to be encoded by smaller block-size modes. Therefore, we can use the information in the residual image to decide which mode is suitable for current coding block coding.



(a) (b)

(c) (d)

Fig. 3 “Stefan” sequence (CIF format) (a) the 183th frame, (b) the 184th frame, (c) the residual image between (a) and (b). (d) The final result of VBS selection for each MB using exhaustive search (One frame has 396 (22x18) MBs totally.)

2.3 Motion-vectors relation analysis

As the previous description, using the ME information may provide a good way to reduce the computational complexity. Therefore, besides the residual image, we will also employ the relation among motion vectors from the ME’s result of smaller block-size modes to improve the encoding time efficiency. We will import the “divide and conquer” concept into our proposed algorithm. First, a 16x16 (8x8) block is split into four 8x8 (4x4) blocks, and after ME, we could get four motion vectors that represent the motion of each 8x8 (4x4) blocks. If certain blocks have the same or similar motion vectors, they will have high probability to merge into larger block-size. For example, Fig. 4 shows that an MB is divided into four 8x8 blocks, the number in each block stands for the block index. If blocks 0 and 1 have similar motion vectors, blocks 2 and 3 also have similar motion vectors but different from those of blocks 0 and 1, they have a high probability to use 16x8 block-size mode to do ME.

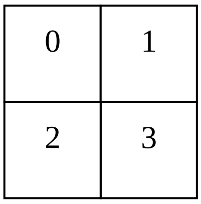


Fig. 4 An MB divided into four 8x8 blocks.

We will use the statistic of some experimental results to support the above concept. First, we apply the exhaustive search to all MBs in “Silent” video sequence to get the best mode. Then, all MBs are divided into 4 classes according to their modes. Class A contains all MBs using 16x16 mode, Class B contains all MBs using 16x8 mode. Class C contains all MBs using 8x16 mode, others are in Class D.

Finally, for each MB *k* we split it into four 8x8 blocks and do ME for each 8x8 block to get four motion vectors *MV(i)* , where *i* stands for block *i*, then, calculate 4 distances as follows:

$$MVD_k(0,1) = | MV_k(0) - MV_k(1) | \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$MVD_k(2,3) = | MV_k(2) - MV_k(3) | \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$MVD_k(0,2) = | MV_k(0) - MV_k(2) | \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$MVD_k(1,3) = | MV_k(1) - MV_k(3) | \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Based on these distance obtained, we compute four average distances for each class as follows:

$$Dist_S(i,j) = \underset{k \in S}{Average}\{MVD_k(i,j)\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

where $S \in \{A, B, C, D\}$; $(i, j) \in \{(0,1), (2,3), (0,2), (1,3)\}$. In Table. 1, all these distances are shown. From Table. 1, we can see that if the best mode is 16x16, the motion vector distance between each 8x8 block motion is smaller. If the best one is 16x8, the motion vector distance between blocks (block 0 and 1) and lower blocks (block 2 and 3) are lower than those of the block pair (0, 2) and (1, 3). This information supports 8x16 block-size mode is selected.

Table. 1 The distance statistic for "Silent" sequence with exhaustive search.

Sequence	Silent (QCIF format)			
Class \ Dist.	Dist(0,1)	Dist(2,3)	Dist(0,2)	Dist(1,3)
A: 16x16 mode	**0.861561**	**0.995636**	**0.954331**	**0.88899**
B: 16x8 mode	**3.712982**	**3.330146**	5.654145	5.089886
C: 8x16 mode	5.151063	5.52542	**4.051184**	**3.506896**
D: Others	10.22414	11.28439	9.308493	10.7181

According to the outlook from intuitive observation and statistic result, if we analyze the motion vectors that come from smaller block-size and make sure they can be merged into larger block-size, then, the rest of block-size mode can be skip. Hence, we can reduce the number of candidate block-size modes and make the speed-up purpose accomplish. Here, we adopt this idea into our proposed algorithm.

2.4 The proposed algorithm

Here, we will give a detailed description for these four steps. The proposed algorithm contains four steps. The first step will detect the still area and decide the block-size mode for each MB on the area. The second step will detect the moving background and the rigid objects with steady motion, and decide the block-size mode for each MB on these areas. The third step will use the relation among motion vectors of smaller blocks to decide the block-size mode. The remaining undecided MBs will be further processing in the final step.

2.4.1 The first step

As mentioned previously, detecting the still area can reduce the number of candidate block-size modes. In this step, we evaluate the mean of the absolute differences (MAD) between the current encoding MB and the co-located MB in the nearest reference frame as

$$TMAD = \frac{1}{MN}\sum_{i=0,j=0}^{M,N} | S_{i,j}^{n} - S_{i,j}^{n-1} | \text{................................}(6)$$

where *(i, j)* means the location of pixels in the current MB, *M* and *N* denote the block size in X-axis and Y-axis, $S_{i,j}^{n}$ and $S_{i,j}^{n-1}$ denote the intensity value of pixel *(i, j)* of original and the previous frame, respectively. And we also evaluate the MAD between each 8x8 block *k* of the current encoding MB and the corresponding block *k* of the co-located MB in the nearest reference frame and denoted as *SMAD(k)*.

Our detection rule is that if the *TMAD* and all *SMAD(k)* are lower than a preset threshold T_{still}, the 16x16 block-size mode is considered as the best mode for this MB. *TMAD* is less than T_{still}, ad at least one *SMAD(k)* is larger than T_{still}, this means that some sub-blocks in the MB may not in still area. Therefore, we need split the MB into smaller blocks, here we only consider 16x16, 16x8 and 8x16 modes as candidate modes, and from these modes find the mode with the minimum cost is selected. All remaining MBs will be processed by the second step.

2.4.2 The second step

In the second step, first, a residual image is established. Then a bi-level threshold function is applied to the residual image to get a binary image *F'*. That is,

$$F'(i,j) = \begin{cases} 0 & , \quad | R_{ij} < T_{wp} | \\ 255 & , \quad otherwise \end{cases} \text{................................}(7)$$

where R_{ij} denotes the value of pixel *(i, j)* in the residual image. Then, we count the number of white points in the current MB, *WP_NUM*. If the *WP_NUM* is less than preset threshold T_{num}, we will set the mode to be 16x16 block size. The remaining MBs will be processed in the third step.

2.4.3 The third step

Here, we will use the information of motion vectors of smaller blocks to determine if these smaller blocks could be merged into lager blocks. We can classify situations into several cases:

Case 1: deciding the best mode for the MB block to be 16x16, if

$MV(0) = MV(1) = MV(2) = MV(3)$ and

$RefFrm(0)=RefFrm(1)=RefFrm(2)=RefFrm(3)$

Case 2: deciding the best mode for the MB block to be 16x8, if

$MV(0) = MV(1)$ *and* $MV(2) = MV(3)$ and $MV(0) \neq MV(2)$ and

$RefFrm(0)=RefFrm(1)$ and $RefFrm(2)=RefFrm(3)$ and $RefFrm(0) \neq RefFrm(2)$.

Case 3: deciding the best mode for the MB block to be 8x16, if

$MV(0) = MV(2)$ *and* $MV(1) = MV(3)$ and $MV(0) \neq MV(1)$ and

$RefFrm(0)=RefFrm(2)$ and $RefFrm(1)=RefFrm(3)$ and $RefFrm(0) \neq RefFrm(1)$.

Case 4: otherwise, examining all block-size modes and selecting the best one.

where *MV(i)* means that the motion vector of sub-block *i* and *i* indicates the index of sub-blocks as shown in Fig. 5. and *RefFrm(i)* denotes the reference frame of a block.

0	1
2	3

Fig. 5 The index for each sub-block in a MB block.

2.4.4 The final step

The final step is similar to the third step. In this step, a MB is split into four 8x8 blocks, for each of them, we apply motion vectors relation analysis again to determine the most suitable mode. We also classify theses situations into following four cases:

Case 1: deciding the best mode for a 8x8 block to be 8x8 mode, if

$SMV(0) = SMV(1) = SMV(2) = SMV(3).$

Case 2: deciding the best mode for a 8x8 block to be 8x4 mode, if

$SMV(0) = SMV(1)$ *and* $SMV(2) = SMV(3)$ and $SMV(0) \neq SMV(2).$

Case 3: deciding the best mode for a 8x8 block to be 4x8 mode, if

$SMV(0) = SMV(2)$ *and* $SMV(1) = SMV(3)$ and $SMV(0) \neq SMV(1).$

Case 4: otherwise, examining all block-size modes {8x8, 8x4, 4x8 and 4x4} and selecting the best one.

where $SMV(i)$ means that the motion vector of 4x4 block i and i indicates the index of 4x4 block as shown in Fig. 6.

0	1
2	3

Fig. 6 The index for each 4x4 block in a 8x8 block.

2.4.5 The overall algorithm

Fig. 7 shows the flowchart of the overall proposed algorithm. First, we apply still area detection to decide if a MB is located in still area. If the MB satisfied the rules, then set best mode of this MB to be 16x16 or one of 16x16, 16x8 or 8x16 corresponding to different conditions, and the computation for the others modes are skipped. Otherwise, we will do ME using the 16x16 block-size mode to create the residual block. If the number of white points in the residual block is less than threshold T_{num}, we will set the best mode to 16x16 and skip the other modes. Otherwise, we first split the MB into four 8x8 blocks. Then, do ME using 8x8 block-size mode for each of them and enter the third step. For each MB, we check which case is satisfied. If one of the first three cases is satisfied, the mode is decided. Otherwise, enter to the final step. For each 8x8 blocks, we split 8x8 block into four 4x4 blocks and do ME using 4x4 mode to get motion vectors of 4x4 blocks. Then, we check four cases to see which case is satisfied. And the best mode is decided for the 8x8 block. The mode obtained through above process will be called P8x8 for convenient illustration.

After the mode decision of each 8x8 block, we sum the prediction error of the four 8x8 blocks. Then, we do ME using 16x8 and 8x16 modes, and select the mode with minimum prediction error from the 16x16, 16x8, 8x16 and p8x8 modes as the best mode for the MB.

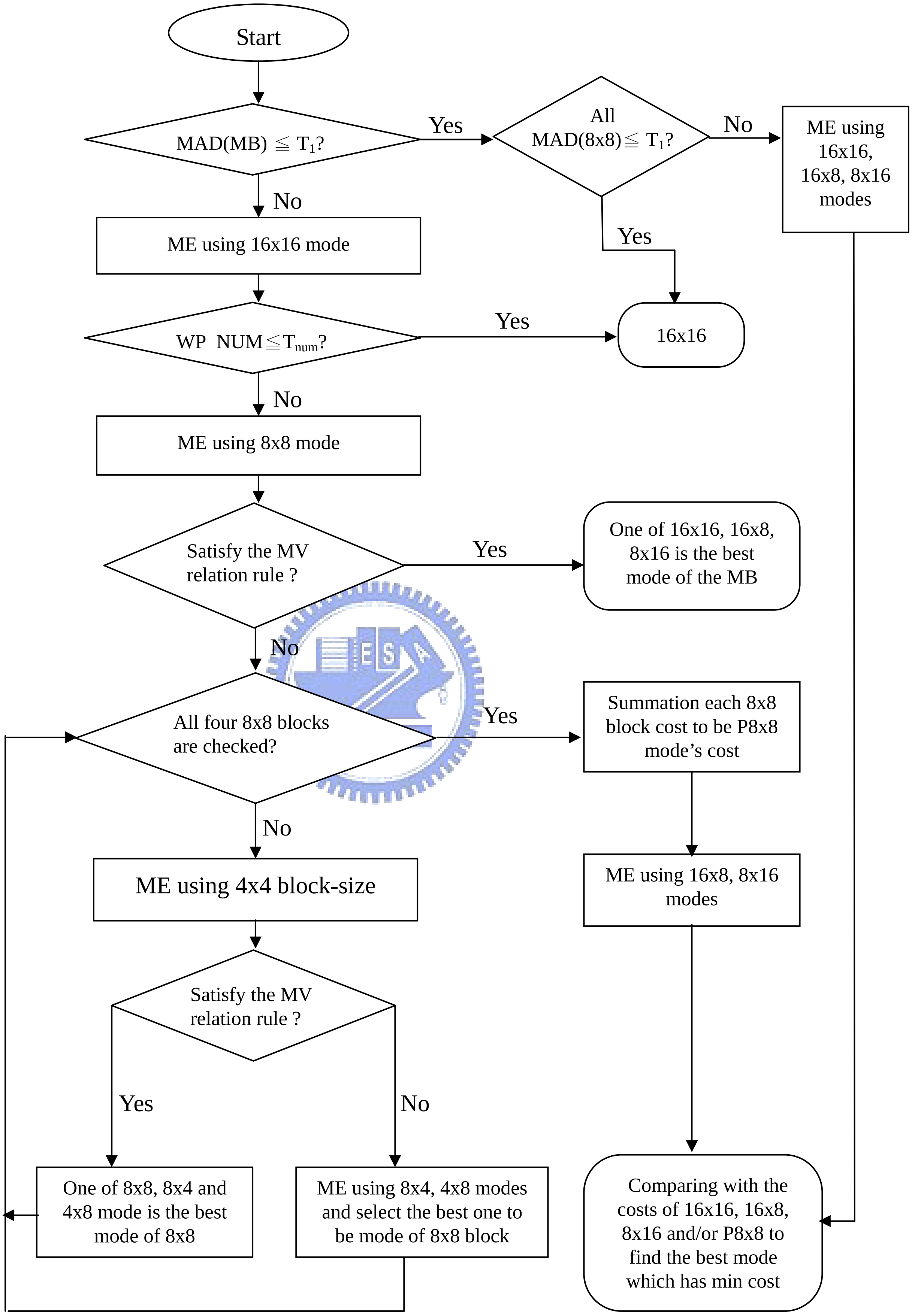

Start
MAD(MB) ≦ T1?
Yes
All MAD(8x8)≦ T1?
No
ME using 16x16, 16x8, 8x16 modes
No
ME using 16x16 mode
Yes
WP NUM≦Tnum?
Yes
16x16
No
ME using 8x8 mode
Satisfy the MV relation rule ?
Yes
One of 16x16, 16x8, 8x16 is the best mode of the MB
No
All four 8x8 blocks are checked?
Yes
Summation each 8x8 block cost to be P8x8 mode's cost
No
ME using 4x4 block-size
ME using 16x8, 8x16 modes
Satisfy the MV relation rule ?
Yes
No
One of 8x8, 8x4 and 4x8 mode is the best mode of 8x8
ME using 8x4, 4x8 modes and select the best one to be mode of 8x8 block
Comparing with the costs of 16x16, 16x8, 8x16 and/or P8x8 to find the best mode which has min cost

Fig. 7 The flowchart of overall proposed algorithm in our thesis.

CHAPTER 3

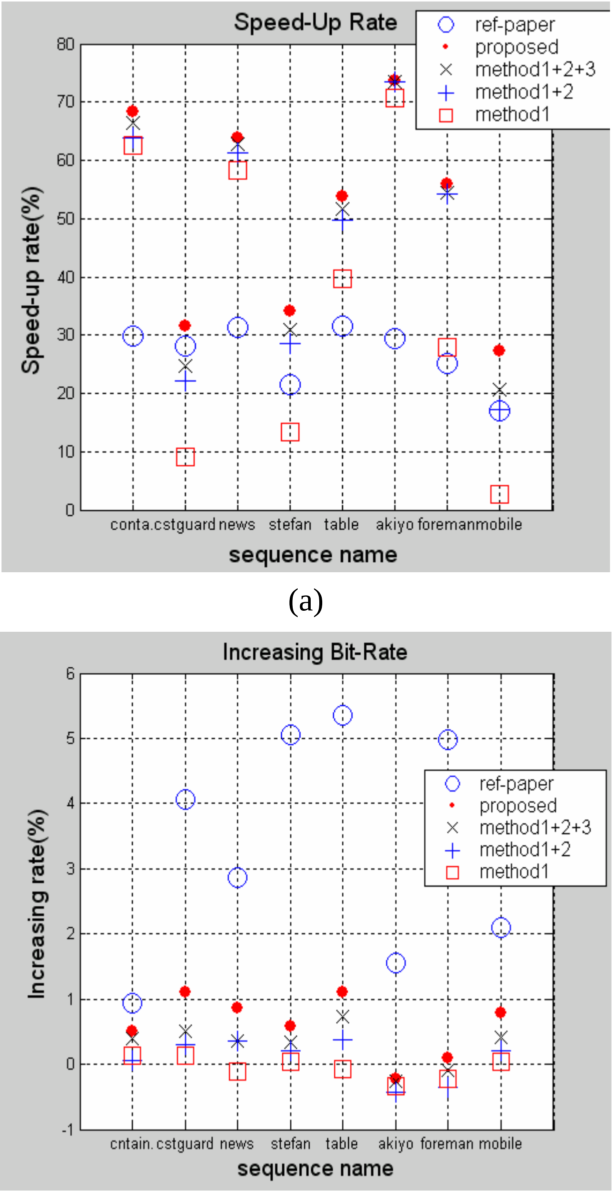
EXPERIMENTAL RESULTS

In this chapter, we will present the simulation results that are derived from implementing the proposed algorithm for inter-frame mode decision in H.264. Further, we will make comparison with methods surveyed in this thesis.

Here, we use three factors, time-saving rate, PSNR and bit-rate to do comparisons. The simulator is based on the Joint Model version 8.2 (JM8.2)〔7〕encoder that provided by JVT. In the following tables and figures, the "*Method 1*" means that we only use the first step to reduce candidate modes for those MBs, their modes are not decided in the first step, the exhaustive search is applied. "*Method 1+2*" means we use first two steps to reduce candidate modes. "*Method 1+2+3*" means that we adopt the first three steps to reduce candidate modes.

3.1 Comparison with Yu's algorithm〔4〕

Yu proposed a fast approach for inter mode decision based on the homogeneous area detection. They classified the block-size modes into three categories, {16x16}, {16x16, 16x8, 8x16} and {all possible modes} corresponding to different level of homogeneity. The speed-up-rate of encoding time lied in between 17% ~ 32% for each video sequence, the increasing rate of bit-rate is about 3.15% in average. Comparing to our proposed algorithm (see Fig. 8), we have surpassed the result of Yu's proposed method even double the encoding efficiency in time and only have a little increase of bit-rate.



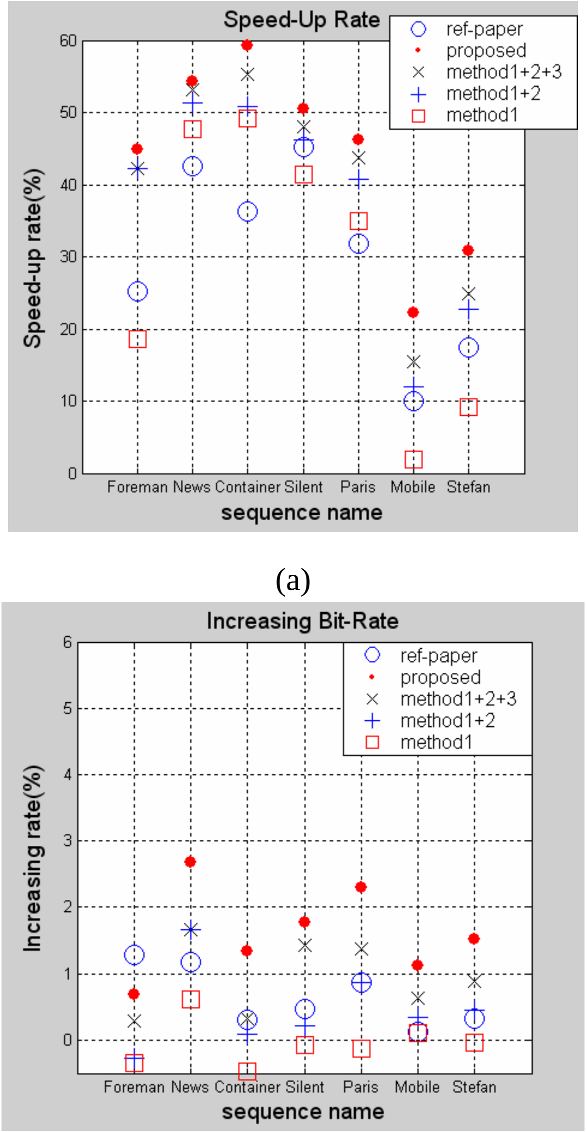
(a)

(b)

Fig. 8 Comparison between our method and Yu's. (a) Comparison in "Speed-Up Rate." (b) Comparison in "Increasing Ratio of Bit-rate."

3.2 Comparison with Wu's algorithm〔5〕

Wu's algorithm only uses the information of homogeneous and still area to do candidate mode reduction. For some video sequences, like "News", it actually can get higher time saving rate and lower bit-rate increase. However, for some video types without the homogeneous or still area, like "Mobile" and "Stefan" sequences, Wu's algorithm can not get good result. And for those video sequences with camera motion or steady-motion objects, such as “Foreman” sequence, Wu’s method does not work well. By contrast, for those video with camera-motion or steady-motion objects, we can use residual image judgment to reduce the candidate modes. Motion-vectors relation also provides a good feature to do candidate mode reduction for all video types. The comparison results are shown in Fig. 9. We can see that the result of our algorithm have overtaken 21% (Mobile sequence) at least in time-saving rate and outshined the result of Wu’s algorithm 9.97% (Mobile sequence).



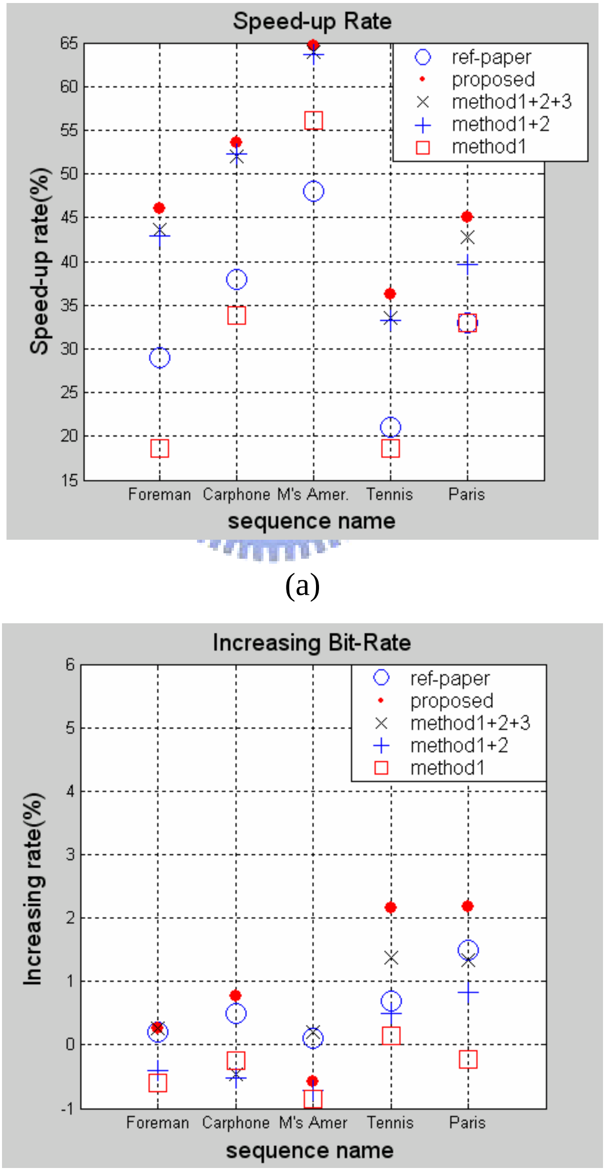
(a)

(b)

Fig. 9 Comparison between our method and Wu’s. (a) Comparison in “Speed-Up Rate.” (b) Comparison in “Increasing Ratio of Bit-rate.”

3.3 Comparison with Jing-Chau's algorithm〔6〕

The method proposed by Jing and Chau uses the MAD to determine if a MB is homogeneous or not. Similar to the previous mentioned methods[4, 5], the algorithm could not get good results for some video types. Fig. 10 shows the comparison result. We can see that the time-saving rate of our proposed scheme could outstrip 35% for all examined sequences.



(a)

(b)

Fig. 10 Comparison between our method and Jing's. (a) Comparison in "Speed-Up Rate." (b) Comparison in "Increasing Ratio of Bit-rate."

3.4 Summary and analysis

The proposed algorithm consists of three major concepts, which are used to reduce the candidate modes. By the demonstration of simulation result, we can see that the proposed method has noticeable improvement in encoding time saving. Evidently, the performance of each step is influenced by the video types. For example, for several video sequences, such as “Silent”, “News” and etc., the frames have the same background, and the artist is almost still except a little motion of face and body. The first step – still area detection will be useful at this condition and has good response on coding efficiency. Another, for the sequence with simple camera motion or steady movement of objects, like “Foreman”, ”Stefan” sequences, the second step - residual image judgment, has better response. Finally, the third and final step using motion vectors relation analysis is suitable for majority of video types.

In summary, the improved efficiency of the proposed algorithm in time-saving rate is about 50% in average with a little increase of bit-rate. However, if we will lay more stress on the increasing rate of bit-rate, we can only use the first few steps (1, or 1+2, or 1=2+3) that influence the bit-rate slightly instead of using the whole algorithm adoption. And it can also keep the improvement of coded time saving. On the contrary, if the reduction of time complexity is the main achievement, the whole proposed algorithm is the best choice to improve the encoded efficiency of H.264 encoder.

Table. 2 is the simulation environmental parameters used in the experiments, and the other parameters not mentioned here follow the setting of the main profile provided by JM8.2 encoder. The “IPPP” structure here denotes that only the first frame of total encoded sequence is Intra-frame, the remaining frames are all of Inter-frame, P-frame, we do not use B frame here. In our proposed algorithm, we set T_{still}=4, T_{wp}=9 and T_{num}=7.

Table. 2 .Environmental Parameters

Parameter Name	Value
Frame Rate	30 frame/sec
Quantization Parameter	28
Search Range	±16
Reference Frame Number	5
RD-Optimization	Disable
Entropy Coding	CABAC
MV resolution	1/4
GOP type	IPPP

Table. 3 and Table. 4, list the simulation result of the proposed methods at each step and the whole proposed algorithm with several video sequences. From these tables, we could be clear acquainted with the fact we described before.

Table. 3 The simulative result includes reduced encoding time rate, dropping value of PSNR and increasing ratio of bit-rate. Method 1 means only adopt the first step. Method 1+2 means adopt the first and second steps.

Sequence	Method 1			Method 1+2		
	Time(%)	PSNR(dB)	Bit-rate(%)	Time(%)	PSNR(dB)	Bit-rate(%)
Foreman (Qcif)	20.95	0.0	0.47	39.29	0.02	0.50
Carphone (Qcif)	31.67	0.03	0.52	46.78	0.02	0.26
News (Qcif)	49.11	0.01	0.43	52.33	0.01	0.42
Container (Qcif)	49.64	0.01	0.53	50.65	0.01	0.53
M's America (Qcif)	58.61	0.02	0.0	64.45	0.02	0.63
Paris (Cif)	40.23	0.00	0.31	43.14	0.01	0.61
Akiyo (Cif)	65.81	-0.01	0.27	68.84	0.01	1.03
Mobile (Cif)	2.74	-0.01	0.30	10.79	-0.01	0.28
Stefan (Cif)	11.57	0.01	0.38	23.64	0.01	0.35
Table (Qcif)	34.47	0.01	0.21	41.91	0.02	0.46
Average	**36.48**	**0.007**	**0.34**	**44.08**	**0.012**	**0.51**

Table. 4 The simulation result including reduced encoding time rate, dropping value of PSNR and increasing ratio of bit-rate. Method 1+2+3 means adopt the first, second and third steps.

Sequence	Method 1+2+3			Proposed Algorithm		
	Time (%)	PSNR(dB)	Bit-Rate(%)	Time (%)	PSNR(dB)	Bit-Rate(%)
Foreman (Qcif)	40.48	0.01	1.68	44.65	0.03	1.92
Carphone (Qcif)	46.49	0.06	1.08	49.35	0.04	1.15
News (Qcif)	54.34	0.02	1.33	56.72	0.02	2.30
Container (Qcif)	58.12	0.01	-0.05	61.69	0	0.58
M's America (Qcif)	64.83	-0.01	0.54	65.44	0.01	0.80
Paris (Cif)	47.32	0.01	1.32	50.42	0.02	2.25
Akiyo (Cif)	69.18	0.01	1.07	69.69	0.03	1.46
Mobile (Cif)	15.70	-0.01	0.76	25.04	0	1.24
Stefan (Cif)	27.53	0.01	0.94	32.35	0.02	1.52
Table (Qcif)	45.82	0.02	1.21	49.21	0.03	1.90
Average	**46.98**	**0.013**	**0.99**	**50.46**	**0.020**	**1.51**

CHAPTER 4

CONCLUSIONS

In this thesis, we have proposed a new algorithm to improve the searching speed for VBS selection in ME part of H.264 encoder. It consists of three major techniques, still area detection, residual-image judgment and analysis of motion vectors relation. By the use of these three techniques, the proposed algorithm can be more suitable for majority of video types. The experimental result shows that an efficient use of these techniques with simple criteria or rules in both MB-level and sub-MB-level modes can reduce number of the candidate modes and get a significant improvement in time saving. The proposed algorithm can get encoding time saving rate about 50.46% in average while keeping similar quality and bit-rate.

REFERENCES

〔1〕“Draft ITU-T recommendation and final draft international standard of joint video specification ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC,” in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, 2003.

〔2〕T. Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Trans. Circuits Syst. Video Technol., Vol. 13, pp. 560–570, July 2003.

〔3〕Iain E. G. Richardson, “H.264/MPEG-4 Part 10 white paper: Prediction of inter macroblock in P-slices,” http://www.vcodex.com , Spring, 2003.

〔4〕A. C. Yu, “Efficient block-size selection algorithm for inter-frame coding in H.264/MPEG-4 AVC,” Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04), Vol. 3, pp. iii-169-172, 17-21 May 2004.

〔5〕D. Wu, S. Wu, K. P. Lim, F. Pan, Z. G. Li, X. Lin, “Block inter mode decision for fast encoding of H.264,” Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04), Vol. 3, pp. iii-181-184, 17-21 May 2004.

〔6〕X. Jing, L.-P. Chau, “Fast approach for H.264 inter mode decision,” IEE Electronics Letters, Vol. 40, Issue 17, pp. 1050-1052, 19 Aug. 2004.

〔7〕Joint Video Team (JVT), Reference Software “Joint Model Version 8.2,” http://iphome.hhi.de/suehring/tml/download/old_jm/jm82.zip.