國立交通大學

資訊科學與工程研究所

碩 士 論 文

針對 MPEG-2 到 H.264/AVC 的轉換編碼的快速演算法

A Fast Algorithm for MPEG-2 to H.264/AVC Transcoding

研 究 生：陳信良

指導教授：蔡文錦　教授

中 華 民 國 九 十 七 年 七 月

針對 MPEG-2 到 H.264/AVC 的轉換編碼之快速演算法

A Fast Algorithm for MPEG-2 to H.264 Transcoding

研 究 生：陳信良　　　　Student：Shin-Liang Chen

指導教授：蔡文錦　　　　Advisor：Wen-Jiin Tsai

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

摘要

H.264/AVC 是一個可以達到比現在最被廣泛使用的 MPEG-2 視訊壓縮標準更高效率的視訊編碼，在這篇論文，我們提出了一個快速的演算法，以加快 MPEG-2 到 H.264/AVC 的轉換編碼的速度，這演算法包括了一個針對 H.264 提供的不同區塊大小的有效選擇方法以及快速決定這些區塊的移動向量的方法，實驗結果顯示所提出的演算法在與完整的 MPEG-2 到 H.264 的轉換標碼比較時，可以降低大量的計算時間且保持相同的影像品質。

關鍵字： MPEG-2、H.264/AVC、轉換編碼、型態決定、移動向量

Abstract

The H.264/AVC standard can achieve much higher coding efficiency than the widely available MPEG-2 video standard. In this thesis, we proposed a fast algorithm in order to speed up MPEG-2 to H.264/AVC transcoding time. The proposed algorithm includes a fast mode decision method for the variable-sizes blocks in H.264 and a fast motion vector decision method for these blocks. The Experiment results show the proposed algorithm can reduce much computational cost with keeping similar video quality while compared with full MPEG-2 to H.264 transcoder.

Keywords: MPEG-2, H.264/AVC, Transcoding, Mode decision, Motion vector

Contents

Chapter 1 1

Introduction 1

Chapter 2 6

Related Works 6

2.1 Fast Macroblock Mode Decision 6

2.2 Fast Motion Vector Decision 8

Chapter 3 11

Motivation 11

Chapter 4 12

Proposed Method 12

4.1 Proposed Mode Decision Method 12

4.1.1 Type check 14

4.1.2 Early Decision 14

4.1.3 Fast Mode Decision(MD) 17

4.2 Proposed MV Decision Method 19

4.2.1 Mode and Energy Check 20

4.2.2 Motion Vector Prediction 21

4.2.3 MV Mapping or Integer-Pixel Refinement 23

4.2.4 Sub-Pixel Refinement 24

Chapter 5 25

Experimental Results 25

5.1 Threshold Determination 26

5.2 Experiment at Fixed Bitrate 29

5.3 Experiment at Various Bitrate 33

5.4 Frame by Frame at Fixed QP = 30 39

Chapter 6 42

Conclusion and Future Work 42

References 43

List of Figures

Figure 1.1 Transform domain transcoder 3

Figure 1.2 Cascaded pixel domain transcoder 4

Figure 2.1 Early Termination 7

Figure 2.2 motion vector derivation 9

Figure 2.3 Dynamic search range 9

Figure 4.1 Proposed transcoding architecture 13

Figure 4.2 The Flowchart of Proposed Mode Decision 13

Figure 4.3 Decide 16X16 or 8X8 16

Figure 4.4 Determine Block Mode (a) 16X8, (b)8X16, (c)8X8 16

Figure 4.5 Compared with upper, left and current macroblock 19

Figure 4.6 The flowchart of proposed MV decision 20

Figure 4.7 Reuse of MPEG-2 MV for Eng = 0 21

Figure 4.8 Select Candidate MV 22

Figure 4.9 Set the search range 23

Figure 4.10 Eng(A) < High_Th and High_ThX2 > Eng(B) > High_Th 24

Figure 5.1 the experiment with different low thresholds and

a fixed high threshold = 200 27

Figure 5.2 the experiment with different high thresholds and

a fixed low threshold = 75 28

Figure 5.3 Experiment results at various bitrates

Figure 5.3 (a) Foreman 33

Figure 5.3 (b) News 34

Figure 5.3 (c) Bus 35

Figure 5.3 (d) Coastguard 36

Figure 5.3 (e) Mobile 37

Figure 5.3 (f) Stefan 38

Figure 5.4 Comparison in frame by frame 40

List of Tables

Table 1.1 Comparison between MPEG-2 and H.264 **2**

Table 4.1 Probability and PSNR loss of intra-to-intra **14**

Table 5.1 Experiment in bitrate = 1.2Mbps **30**

Table 5.2 Experiment at fixed QP = 30 **40**

Chapter 1

Introduction

Video transcoding is one of the video adaptation methods to convert the characteristics of video stream into the other characteristics for solving the incompatible problem for universal multimedia access [1-2]. The characteristics include bitrate, framerate, video standard, image resolution, etc. The format transcoding method purposes to convert the input stream of one video standard to the output stream of another standard and the key issue in it is to minimize the complexity while keeping the quality [3].

Currently, most of the video used in the multimedia applications such as DTV, DVD, and HDTV uses the MPEG-2 video coding standard. However, H.264/AVC developed by Joint Video Team of ISO/IEC MPEG and ITU-T VCEG is the latest video coding standard and achieves high coding efficiency. Compared with MPEG-2 video, H.264 can reduce more than half of the bitrate with same video quality. Therefore, the MPEG-2 to H.264 transcoder is necessary and that is discussed in this thesis.

H.264 employs some techniques which are different form MPEG-2 as shown in Table 1.1. The major differences are: 1. H.264 uses different intra prediction modes and variable block sizes for inter prediction; 2. 4X4 integer transform (HT) in H.264 is different from 8X8 DCT in MPEG-2; and 3. H.264 uses multiple reference frames for inter prediction. Besides, the motion vector accuracy and the entropy coding of H.264 are also different from MPEG-2.

Format	**MPEG-2**	**H.264**
Intra prediction	**DC prediction**	**9 modes(4X4)** **4 modes(16X16)**
Block size for inter prediction	**16X16**	**4 MB mode** **4 sub-block mode**
ME/MC accuracy	½ **pixel**	¼ **pixel**
Transform	**8X8 DCT**	**4X4 HT**
Ref. frame #	**1**	**5**
Entropy coding	**VLC**	**CABAC, CAVLC**

Table 1.1 Comparison between MPEG-2 and H.264

Generally, transcoders can be classified into two types, one called "Transform Domain Transcoder(TDT)" operates in the transform domain and the other called "Cascaded Pixel Domain Transcoder(CPDT)" operates in the pixel domain.

A transform domain transcoder can reduce much computational complexity of IDCT and HT. In [4], an efficient method has been proposed to convert DCT coefficients to HT coefficients entirely in the transform domain. It shows that the conversion is essentially a 2D transform, S-Transform, as shown in Figure 1.1. However, TDT has the problem of drift error due to the mismatch of motion compensation. [5] analyzed two major kinds of drift error: interpolation error and quantization error and proposed a transcoding scheme based on quantization and interpolation drift error compensation. In [6], their transcoder converts 8x8 DCT in MPEG-2 into 4x4 integer transform in H.264 and reduces the image resolution to half of the original size in both vertical and horizontal directions.

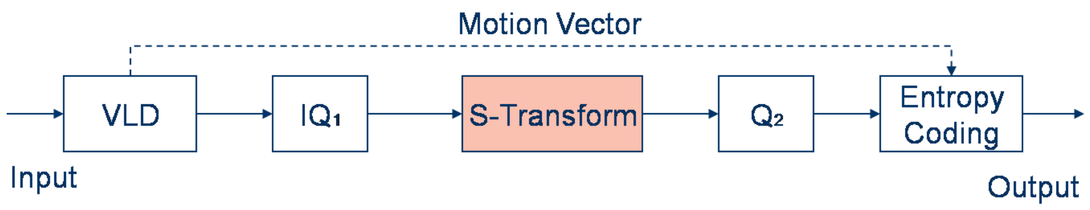


Figure 1.1 Transform domain transcoder

On the other hand, the cascaded pixel domain transcoder which involves full decoder and encoder can be accelerated by reusing the coding information as shown in Figure1.2. [7] proposed an efficient CPDT by using MPEG-2 coding mode and using motion vectors of MPEG-2 to be the prediction motion vectors in H.264 stream. The intra mode decision for MPEG-2 to H.264 transcoding has been proposed in [8]. It computes the edge angle in a block from the MPEG-2 DCT coefficients and according to the edge angle it limits the intra prediction modes to be performed in the transcoding. Since the mode decision and motion estimation hold the very great proportion of computational time in H.264 encoder, most researches have focused on reducing macroblock mode decision time [9-12] or motion estimation time [13-14] in CPDT.

In [9], the proposed algorithm is used to determine which one of the 16x16, 16x8, 8x16, and 8x8 block size modes should be used for each macroblock. [10] proposed a 2-D Sobel filter based motion vector method and a DCT domain method to measure macroblock complexity and realize efficient H.264 candidate mode decision. A fast trnascoding algorithm based on CPDT architecture has been proposed in [11]. It used the coded macroblock type and the coded block pattern included in the MPEG-2 bitstream to reduce the complexity against the full-searching mode decision. Moreover, [12] proposed a scheme which determines suitable intra or inter modes in H.264 encoder according to DCT coefficients, motion vectors, and neighboring macroblock modes of the MPEG-2 bitstream and improves quality compared with the algorithm presented in [11]. On the other hand, [13] proposed a novel motion mapping algorithm aimed for low-complexity MPEG-2 to H.264 transcoding by using MPEG-2 motion vectors efficiently. Compared to motion mapping, [14] presented reduced complexity motion estimation by reducing the search range dynamically.

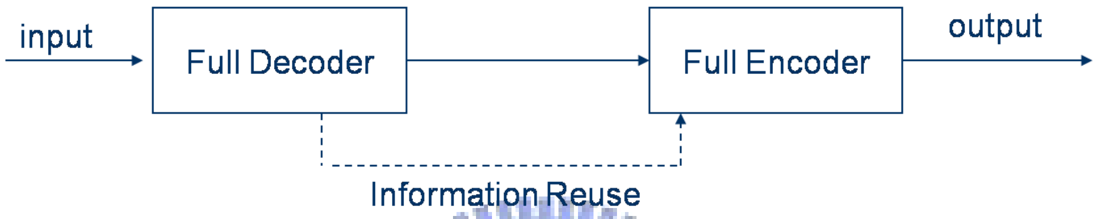


Figure 1.2 Cascaded pixel domain transcoder

Compared with the cascaded pixel domain transcoder, the complexity of the transform domain trnascoder can be reduced further because the processes of the 8x8 inverse DCT and the 4x4 integer transform are skipped. However, the TDT architecture has several disadvantages compared to CPDT. First, TDT can not use in-loop filtering to eliminate blocking artifacts but CPDT can use. This disadvantage leads to the degradation of video quality. Second, the motion estimation using the variable block size in H.264 restricts the use of MC-DCT and has some drift error in TDT architecture. Moreover, the complexities of the 8x8 IDCT and the 4x4 integer transform are much lower than those of the macroblock mode decision and motion estimation in H.264. According to these reasons, our proposed transcoder for MPEG-2 to H.264 transcoding is implemented in CPDT architecture.

In this thesis, a new MPEG-2 to H.264 transcoding is proposed, which provides a fast macroblock mode decision and a fast motion vector decision in pixel domain. The experimental results show that our transcoder speeds up much computational time and keeps the video quality. The rest of this thesis is organized as follows.

Chapter 2 gives the introduction to related works of the fast mode decision and MV decision in CPDT, and chapter 3 gives our motivation. Our proposed trnascoder architecture is discussed in chapter 4, and the experimental results are shown in chapter 5. The conclusion is given in the last chapter.

Chapter 2

Related Works

In chapter 1, we have introduced two architectures of MPEG-2 to H.264 transcoding: TDT and CPDT. The quality of video in CPDT is higher than that in TDT but it needs more computational time, thus we proposed a fast algorithm for MPEG-2 to H.264 transcoding in pixel domain. In this chapter, we introduce previous works in CPDT including fast macroblock mode decision method and fast motion estimation scheme.

2.1 Fast Macroblock Mode Decision

There are four inter macroblock modes and four subblock modes in H.264/AVC video coding standard. A good fast mode decision scheme is very important in MPEG-2 to H.264 transcoding since it can achieve very good rate-distorition performance with low complexity. [9] analyzed the energy of MPEG-2 residual macroblock and used it to select macroblock size in H.264. The energy of a residual macroblock is measured as the sum of the absolute value of the dequantized DCT coefficients of the motion compensated prediction MPEG-2 residual macroblock.

When the energy is very low, it is a strong probability that the optimal block size mode will be 16X16. While the energy is big, the probability that the optimal size is not 16X16 is more than 90%. Therefore, [9] set a *low threshold* for 16X16 block type and a *high threshold* for 8X8 block type as shown in Figure 2.1. If the energy of a residual macroblock is lower than the low threshold, the transcoding will choose the 16X16 block mode and turn off the process of evaluating other block modes since the performance of 16X16 mode is good enough. On the other hand, if the energy is larger than the *high threshold*, the transcoding will spilt the macroblock into four 8X8 blocks to achieve better performance. If the energy of a residual macroblock is between the *low threshold* and *high threshold*, the trnascoding will determine the final mode according to the distribution of energy of the four 8X8 blocks in that macroblock.

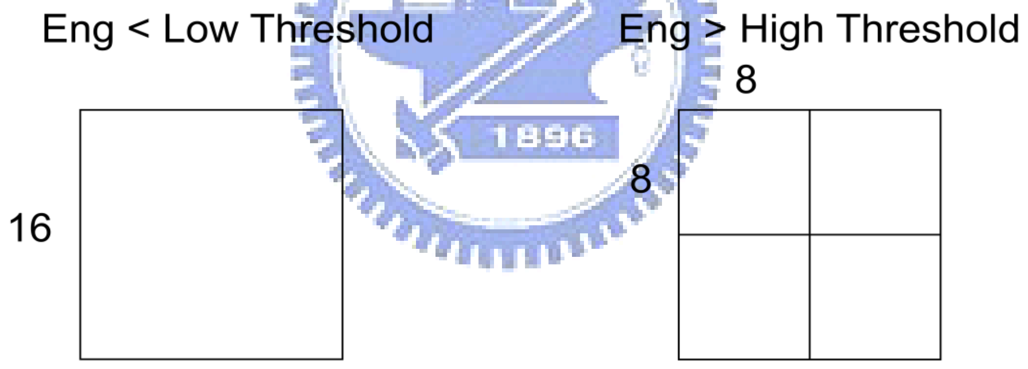


Figure 2.1 Early Termination

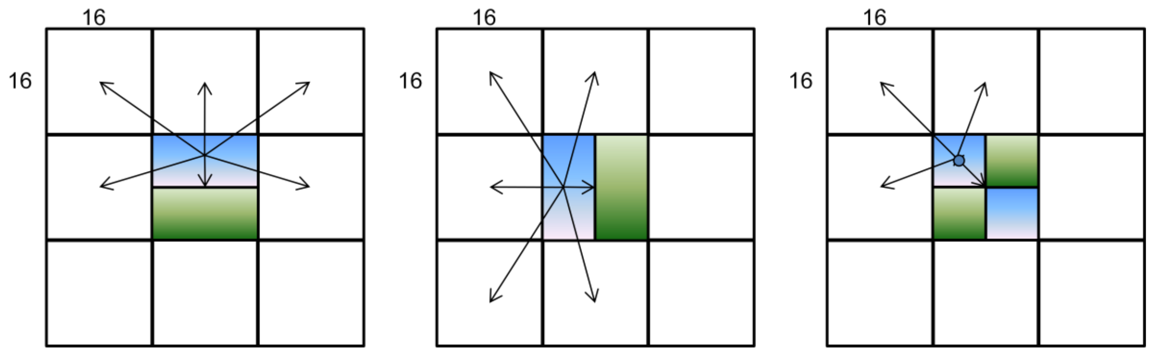
Kim, et al. proposed another fast mode decision algorithm by using the coded macroblock type (CMT) and the coded block pattern (CBP) in MPEG-2 [11]. In MPEG-2, the CMT means whether a macroblock uses the temporal prediction from reference frames or spatial perdiction from the same frame and the CBP indicates which blocks in the macroblock are coded. The transcoder of [11] uses the CMT to determine a macroblock to be intra or inter coded. If the macroblock is intra coded type in MPEG-2, the transcoder would choose intra coding for it in H.264. Similarly, when the macroblcok is inter coded type, the transcoder would use inter coding for it and select the macroblock size adaptively according to CBP, as follows.

- The number of not coded blocks >= 2 : SKIP, 16X16, 16X8, and 8X16 are enabled.
- The number of not coded blocks = 1 : SKIP, 16X16, 16X8, 8X16, and 8X8 are enabled
- If all blocks are coded : all inter macroblock modes are enable

The fast transcoder proposed in [11] reduces the computational time by reducing the number of macroblock modes for RDO evaluation in H.264 encoder. Compared to [11], the fast transcoder proposed in [9] reduces more transcoding complexity since it decides exact one macroblock mode for H.264 encoding and no RDO evaluation is needed. However, the video quality produced by the transcoder in [9] might not be good if the decided block mode is not the best one.

2.2 Fast Motion Vector Decision

In MPEG-2 to H.264 transcoding, the motion estimation needs much computational cost since there are seven macroblock modes in H.264. [11] proposed a simple motion mapping mechanism for deciding the motion vector for inter 16X16 mode. In [13], an efficient motion mapping method to decide motion vectors for inter 16X16, 16X8, 8X16, and 8X8 modes has been proposed. The output motion vector is derived as a weighted average of the MPEG-2 motion vectors of candidate macroblocks which include the macroblock containing the target block mode and those macroblocks adjacent to the target block, as shown in .Figure 2.2, where the target block modes in Figure 2.2(a), (b), and (c) for motion vector derivation is 16X8, 8X16, and 8X8 respectively. The weight of a candidate motion vector is inversely proportional to the distance between its macroblock’s geometric center to the target block’s geometric center.



(a) 16X8 (b) 8X16 (c) 8X8

Figure 2.2 motion vector derivation [13]

[14] propsoed a fast motion estimation algorithm by reducing the search range. The search range reduction depends on the MPEG-2 coding mode and MPEG-2 motion vecotr, as shown in Figure 2.3. If the macroblock mode is skip mode and inter with zero motion vector in MPEG-2, the search range is limited to 1. On the other hand, if the macroblock is intra mode in MPEG-2, the search range will be set to the maximum as specified in the H.264 encoder configuration (e.g., 16 in Figure 2.3). In the case of MPEG-2 inter mode that has motion vector, the search range for the macroblock in H.264 is set to the maximum of the x-coordinate and y-coordinate value of the MPEG-2 motion vector.

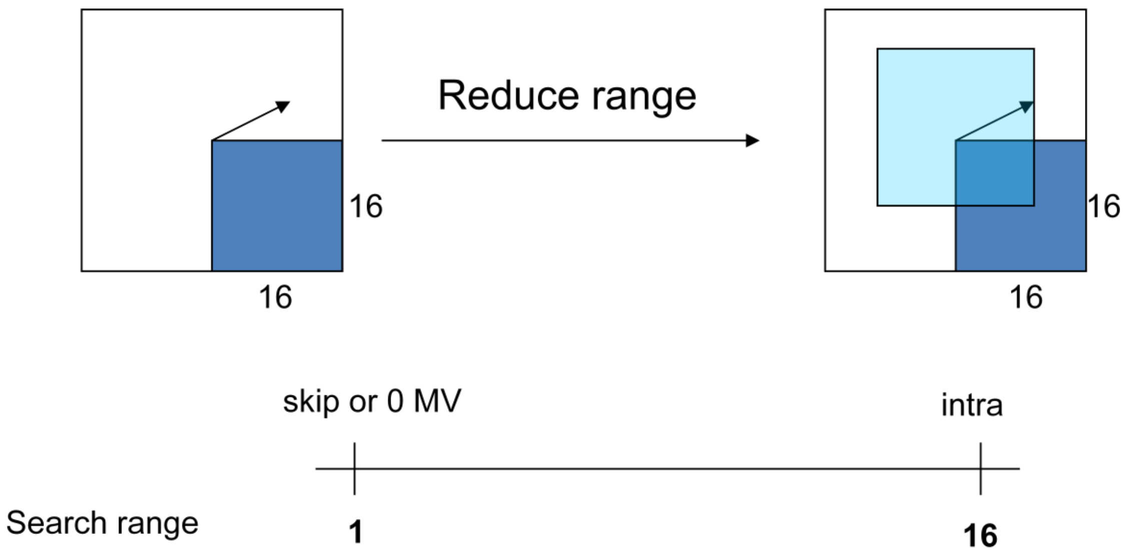


Figure 2.3 Dynamic search range [14]

Compared to [14], the motion mapping method proposed in [13] does not need full-pixel motion estimation because it derives exact one motion vector for each block mode. Therefore, the motion mapping method reduces much more computational cost. However, The video quality produced by the motion mapping method might not be good enough if the derived motion vector is not optimal.

Chapter 3

Motivation

In CPDT, a good mode decision or motion estimation can decreases transcoding time efficiently with high video quality. [9] proposes a fast mode decision method and [13] proposes a motion mapping scheme to reduce much computational time. In this thesis, we want to provide a fast mode decision and a fast motion vector decision with adapted refinement to determine macroblock size and MV efficiently. Besides, we hope that our fast mode decision and motion vector decision methods can be combined efficiently while keeping the quality nearly the same as the CPDT transcoder to achieve a high quality and low complexity MPEG-2 to H.264 transcoding. Moreover, since the inter 16X16, 16X8, 8X16, 8X8 block modes are usually used more than sub-block modes like 4X8, 8X4 and 4X4, we only focus on improving efficiency of these mode decision and motion estimation in this thesis.

Chapter 4

Proposed Method

In this chapter, we describe the proposed fast transcoding algorithm in detail. Figure 4.1 shows our MPEG-2 to H.264 trnascoding architecture including a full MPEG-2 decoder and a full H.264 encoder. In this architecture, when the MPEG-2 video stream is decoded, the information including motion vectors, residual coefficients, and macroblock types in MPEG-2 will be stored. And then, in the H.264 encoding part, we propose a fast mode decision and a fast motion vector decision methods. The proposed mode decision uses the residual coefficients and macoblock types obtained from MPEG-2 video stream to efficiently select macroblock mode to be used in the H.264 encoder. The proposed MV decision uses the MPEG-2 motion vectors and residual coefficients to speed up the motion estimation process of the H.264 encoder.

4.1 Proposed Mode Decision Method

Since H.264 employs four block modes and four subblock modes within the inter coding mode, the block mode selection has much computational cost in MPEG-2 to H.264 transcoding. Therefore, a fast block mode decision to select block mode efficiently is important. Figure 4.2 shows the flowchart of the proposed fast mode decision algorithm for an MPEG-2 to H.264 transcoder, and we only focus on the four block modes: inter 16X16, 16X8, 8X16, and 8X8. In the following subsections, we describe the algorithm for four inter block mode decision.

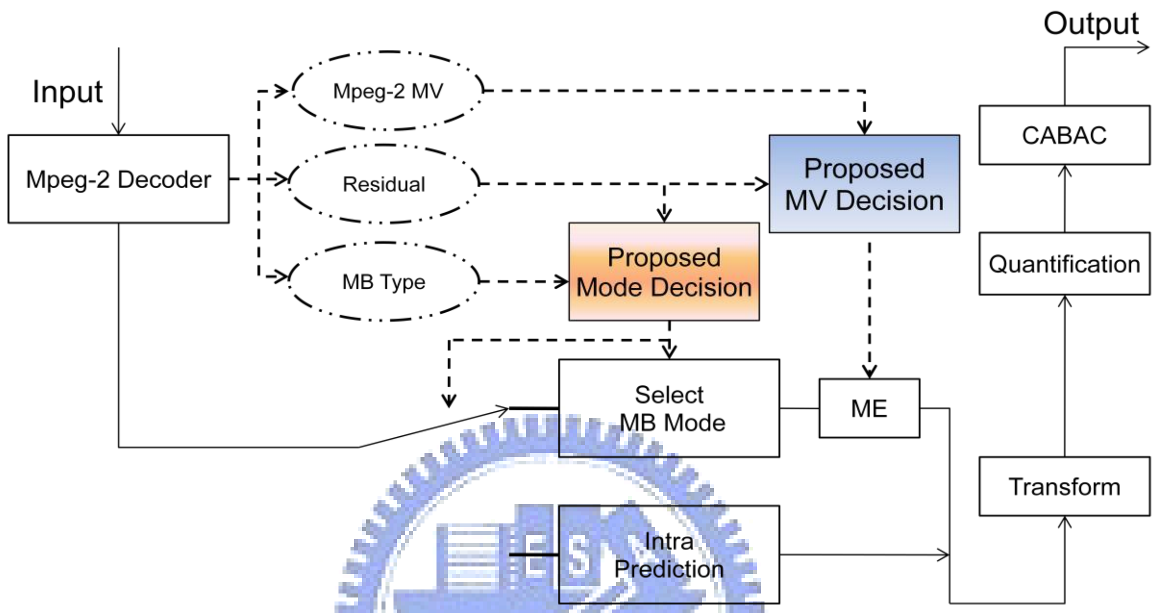


Figure 4.1 Proposed transcoding architecture

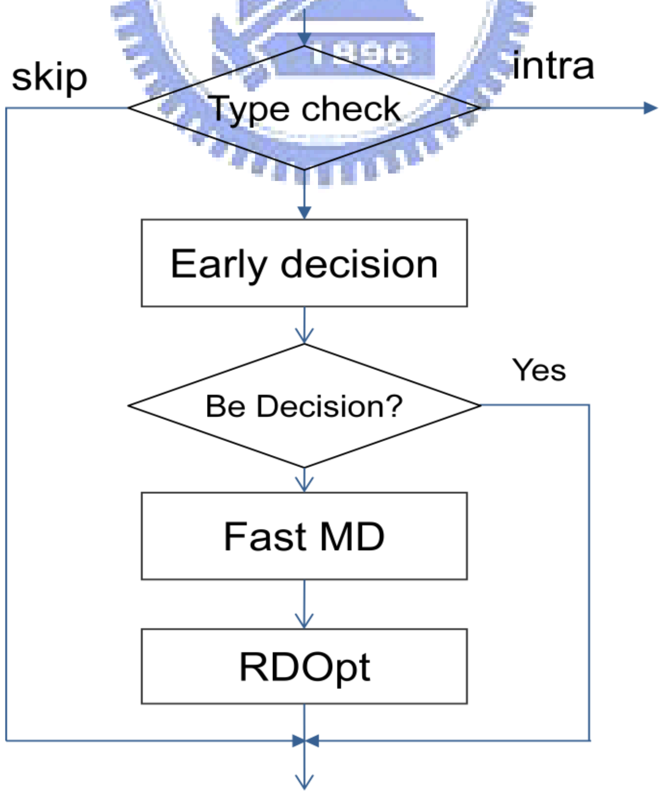


Figure 4.2 The Flowchart of Proposed Mode Decision

4.1.1 Type check

The coded macroblock in the P frame of an MPEG-2 video stream include four types, as shown in follow :

- MC type : motion compensation with one motion vector
- No-MC type : motion compensation with zero motion vector
- Skip type : no residual coefficient and motion vector
- Intra type : intra coded mode

In our algorithm, the current marcoblock type is first checked. If the MPEG-2 coded macroblock type is skip type, only the skip mode and inter 16X16 mode as the macroblock mode in H.264 encoding process since it means the prediction error is very low using this type and therefore the RD cost of the corresponding macroblock is supposed to be very low in the H.264 encoder. In our experiments, if the MPEG-2 macroblock is intra coded, the probability of this macroblock is intra type in H.264 will be about 50%. Moreover, if these marcoblocks are all decided intra type in H.264, the PSNR loses less than 0.09dB, as shown in Table 4.1. Therefore, when the MPEG-2 macroblock is intra coded without motion estimation, two intra modes, intra 4X4 and intra 16X16, are considered as the macroblock mode in H.264 encoder.

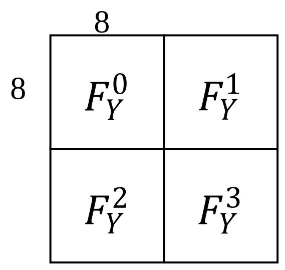
Sequence	Probability	PSNR loss	Sequence	Probability	PSNR loss
Foreman(120f)	65%	0.01	News(120f)	46%	0.00
Stefan(90f)	19%	0.09	Tempete(120f)	62%	0.00

Table 4.1 Probability and PSNR loss of intra-to-intra

4.1.2 Early Decision

In Chen, et al. algorithm [9], they have shown that the energy of the MPEG-2 residual coefficients can be used to determine the block sizes: 16X16, 16X8, 8X16, 8X8 accurately. In our algorithm, the energy of every MPEG-2 8X8 residual block in the current macroblock is calculated using the formula shown as follows :

$$Eng8[i] = \sum_{v=0}^{7}\sum_{u=0}^{7} | F_{Y}^{i}(u,v) |$$

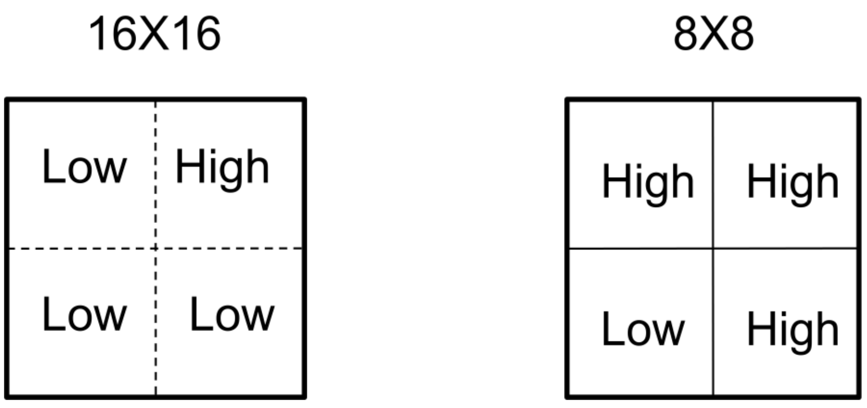


,where F_Y^i is the DCT residual coefficient of the 8X8 block *i* luminance component

Based on the energy of four 8X8 residual blocks, a *low- threshold* and a *high-threshold* are used to classify each 8X8 residual block into one of the following energy types. Low-block, high-block, and undecided-block :

- Low energy : 8X8 block energy < low-threshold
- High energy : 8X8 block energy > high-threshold
- Undecided : 8X8 block energy ≥ low-threshold and ≤ high-threshold

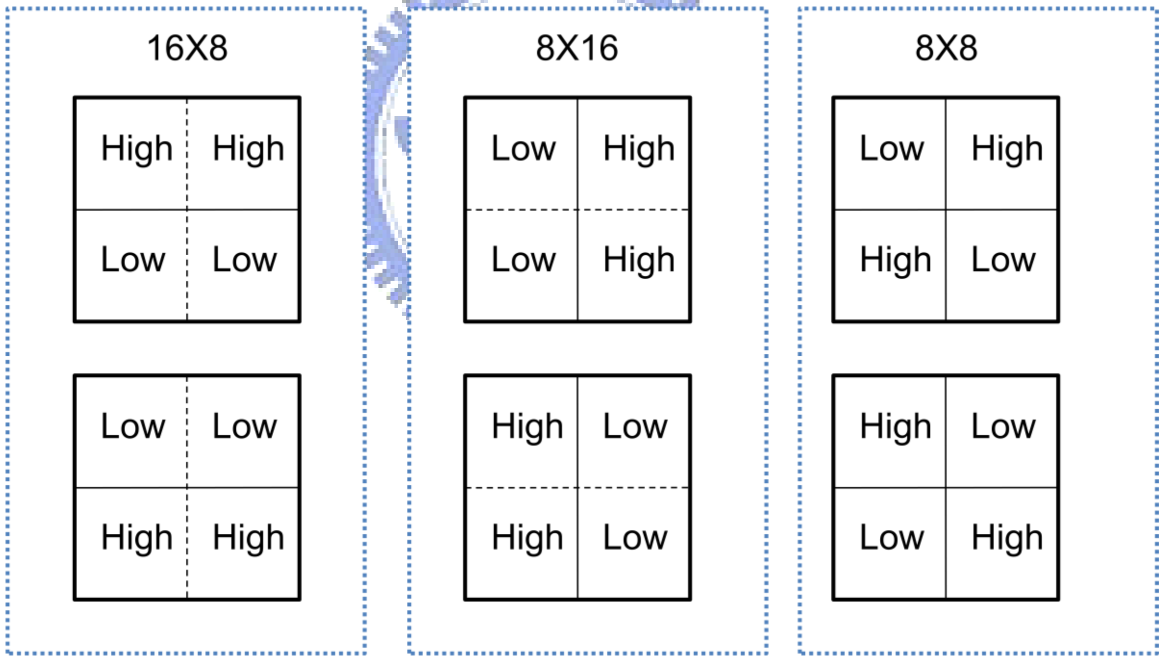
If there is no undecided-block in current macroblock, the block mode of this macroblock are be early determined as follows. If there are three or more 8X8 blocks set as *low-block* in the current macroblock, the macroblock size is determined as 16X16 (e.g., in Figure 4.3a) since the prediction error is very low. On the other hand, if there are three or more 8X8 block set as *high-block*, it means the content of current macroblock is very high complexity. Therefore, the block size of 8X8 mode is considered in this case (e.g., in Figure 4.3b).



(a) (b)

Figure 4.3 Decide 16X16 or 8X8

Moreover, If there are two low-blocks and two high-blocks, the distribution of the energy types is used to determine the block mode, as shown in Figure 4.4.



(a) (b) (c)

Figure 4.4 Determine Block Mode (a) 16X8, (b)8X16, (c)8X8

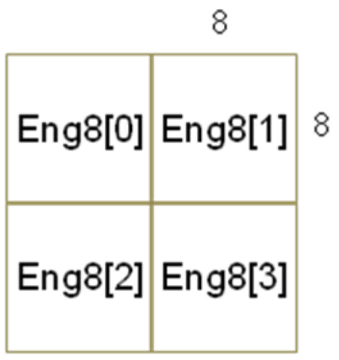
If the two upper 8X8 blocks are high-block and two bottom 8X8 blocks are low-block, or oppositely, the two upper blocks are low-block and the two bottom blocks both are high, as shown in Figure 4.4(a), then 16X8 block mode which divides the macroblock into upper and bottom parts is chosen in H.264 to make lower RD cost. Similarly, if the two left 8X8 blocks are low-blocks and the two right 8X8 blocks are high-blocks or oppositely, as shown in Figure 4.4(b), this marcoblcok is determined as 8X16 mode. Moreover, if the upper-left block and the bottom-right block are low-block and the upper-right block and the bottom-left block are high-block or oppositely, as shown in Figure 4.4(c), it means the larger block size is not good for current macroblock since the residual is very dispersive. Therefore, we choose the 8X8 block size in this case. According to experimental results shown in later, it is observed that more than half of macroblock modes in the most video sequence can be decided using this proposed early decision method which explores energy distribution patterns in a macroblock.

4.1.3 Fast Mode Decision(MD)

In the previous subchapter, we describe an early decision method to determine block mode if there is no undecided-block in current macroblock. However, when one or more among the four 8X8 blocks of the macroblock is undecided-block, it is hard to determine one block mode for the macroblock since it is hard to predict which mode will have best RD cost. On the other hand, if we consider all inter modes in this case, that is, perform exhaustively full search to determine the best mode, the conputational cost could be quite high. Therefore, we propose a fast mode decision method to choose some candidate block sizes efficiently to speed up transcoding process while still keeping with high video quality.

The 16X8 mode and 8X16 mode in H.264 are very different since the 16X8 mode divides macroblock into upper and bottom partition and the 8X16 mode divides macroblock into left and right partition. Therefore, only one of 16X8 and 8X16 modes is considered to be the candidate mode for the undecided macroblock. On the other hand, the 16X16 mode has much difference from the 8X8 mode since the 16X16 mode is large size which is good for low complex macroblock and the 8X8 mode is smaller size that is good for high complex macroblock. Similarly, only one of the 16X16 and 8X8 is considered to be the candidate mode. Due to reduce candidate modes compared with H.264, the computational cost for the undecided macroblocks is also reduced. The proposed fast mode decision algorithm is as follows :

- First, we define several variables as follows :
  - TopEng : Eng8[0] + Eng8[1]
  - BottomEng : eng8[2] + eng8[3]
  - LeftEng : eng8[0] + eng8[2]
  - RightEng : eng8[1] + eng8[3]
  - MinEng, MaxEng : the minimum and maximum energy of four 8X8 block
  - AvgEng : the average energy of four 8X8 blocks
  - K : an empiric constant value, In our experiment, K = 1.5



- Choose 16X8 or 8X16 mode :

```
if(abs(TopEng - BottomEng) > abs(LeftEng - RightEng))
    choose the 16X8 mode as the candidate mode
else
    choose the 8X16 mode as the candidate mode
```

- Choose 16X16 or 8X8 mode

```
if((MaxEng - MinEng) < AvgEng * K)
    choose the 16X16 mode as the candidate mode
else
    choose the 8X8 mode as the candidate mode
```

Besides the above algorithm, in our experiments, if the left macroblock of the current macroblock mode is equal to upper macroblock in H.264 (as Figure 4.5), the probability of current macroblock is equal to them will be more than 60% in the macroblocks which is not early decided. Therefore, while the left macroblock mode and the upper macroblcok are equal and this mode is not one of the candidate modes, we also add this mode into the candidate set. In other words, in the fast mode decision stage, only the candidate modes (three at most) will be evaluated with RDO process to determine the best mode for those undecided macroblocks.

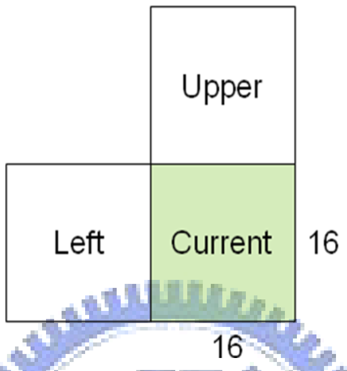


Figure 4.5 Compared with upper, left and current macroblock

4.2 Proposed MV Decision Method

In H.264 encoder, it uses quarter-pixel-accuracy prediction and various block sizes ranging from 16X16 to 4X4 for motion estimation on. The one with the best RD cost is used for motion compensation. Therefore, the computation cost of the motion estimation is very high. In order to speed up the motion estimation for MPEG-2 to H.264 transcoding and we proposed an efficient motion vector decision method. Our algorithm determines motion vector for the partition blocks of inter 16X16, 16X8, 8X16 and 8X8 only. Figure 4.6 shows the flowchart of the proposed algorithm. Detailed description is given in the following subsections.

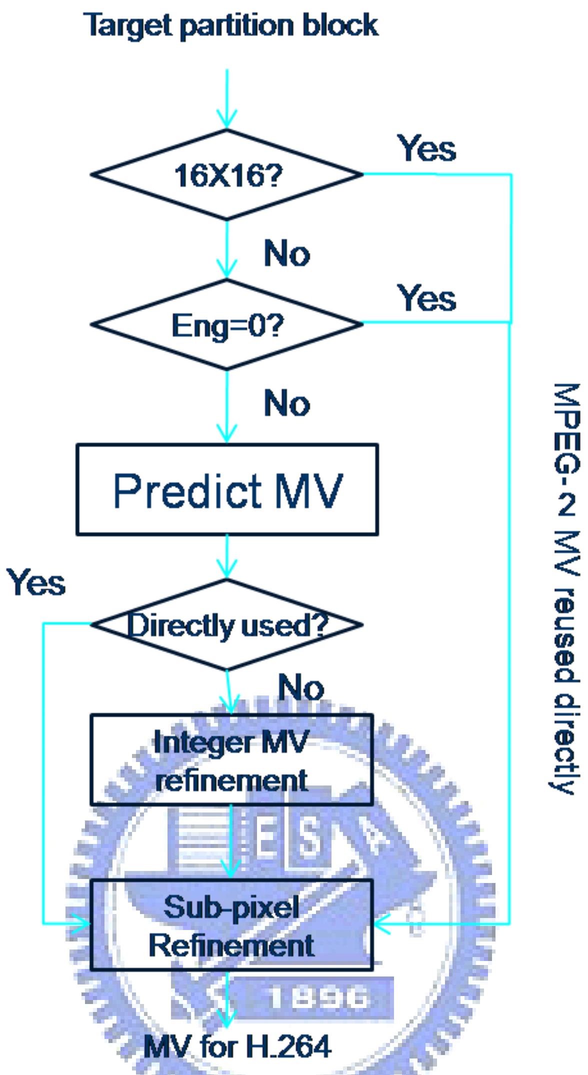


Figure 4.6 The flowchart of proposed MV decision

4.2.1 Mode and Energy Check

The motion vectors included in an MPEG-2 coded video stream are estimated for block size of 16X16. Therefore, the motion vectors of MPEG-2 seems can be reused for the inter 16X16 mode in H.264. However, both MPEG-2 and H.264 allow half-pixel accuracy and the different filters are used for interpolation at the half-pixel position. In MPEG-2, two-tap filter: *(1,1)/2* is used whereas six-tap filter: *(1,-5,20,20,-5,1)/32* is used in H.264. Therefore, we reuse the motion vectors of MPEG-2 only for integer-pixel motion vectors of 16X16 blocks and apply the sub-pixel refinement around the integer motion vectors.

On the other hand, although MPEG-2 standard does not apply the block sizes of 16X8, 8X16, and 8X8, we still consider reusing the MPEG-2 motion vectors directly for these partition blocks in H.264 if the energy of the partition block is equal to zero. For example, in Figure 4.7, the motion vector of MPEG-2 is reused directly in right 16X8 partition block since the energy value of the upper-right and bottom-right 8X8 blocks of the corresponding macroblock are both zero.

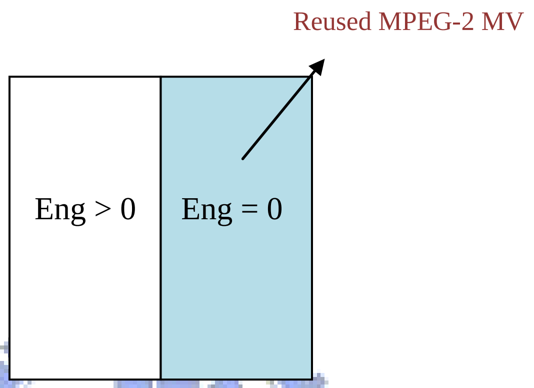
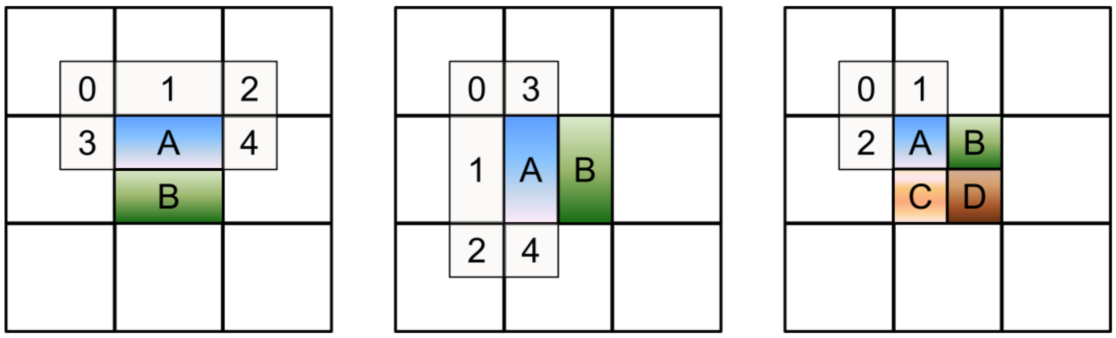


Figure 4.7 Reuse of MPEG-2 MV for Eng = 0

4.2.2 Motion Vector Prediction

In the motion mapping algorithm [13], they used the motion vectors of the current macroblock and those macroblocks adjacent to the current target block to derive the output motion vector (Figure 2.2). In our approach, we also use these motion vectors as the candidate motion vectors for H.264 motion vector prediction. However, the motion vectors of those macroblocks are not always good for the target block because some of these candidate motion vectors, called unreliable motion vectors, may have different directions from the real motion of the target block. Therefore, we present an efficient method to remove such unreliable motion vectors from the candidate motion vectors. We consider using the energy of the residual coefficients to define the unreliable motion vectors and remove them from the candidate motion vectors, as shown in Figure 4.8.



(a) 16X8 (b) 8X16 (c) 8X8

Figure 4.8 Select Candidate MV

We compare the energy of the target block with the energy of those blocks adjacent to the current target block. If the energy of the neighbor block is larger than the two times energy of the target block, this motion vector is defined as an unreliable motion vector for the target block and therefore, it will be removed from the set of candidate motion vectors. For example, for the 16X8 mode in Figure 4.8(a), the partition-A block and the block of number 1 are both in 16X8 size which has two 8X8 blocks, and the blocks of number 0, 2, 3 and 4 are all in 8X8 size block. We compare the energy of partition-A block with that of number 0, 2, 3 and 4 blocks and compare two times energy of partition-A block with that of number 1 block, as follows :

- *if(Eng(i)8X8 > Eng(A)16X8)   i = 0, 2, 3, and 4*

  remove this MV from the candidate MVs

- *if(Eng(1)16X8 > (Eng(A)16X8 * 2))*

  remove this MV from the candidate MVs

Finally, the target motion vector are computed as :

$$MV(A) = round(\sum w_i \times MV_i) \qquad (1)$$

where the weight w_i is inversely proportional to the distance between the geometric center of the candidate macroblocks which are not removed and that of target partition block A. Since the motion vectors predicted in this way may not be good enough, in the next subsection, we describe how to decide the motion vector mapping or refine it adaptively.

4.2.3 MV Mapping or Integer-Pixel Refinement

For equation (1), since the most proportion of the predicted motion vector comes from the motion vector of the macroblock with the target partition block, we use the magnitude of the energy to estimate the accuracy of the original MPEG-2 motion vector of this macroblcok and determine the search range of the refinement window. The refinement is performed with full search method of H.264 in this search window centered at the predicted motion vector.

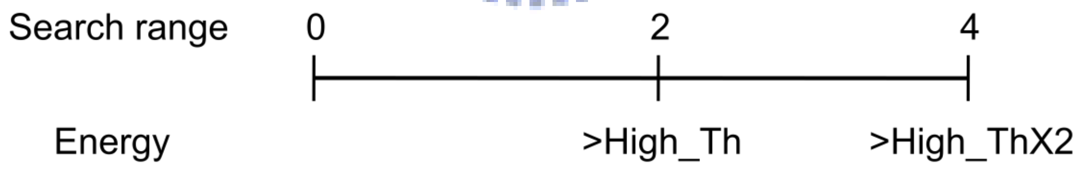
$$\text{Search range} = \begin{cases} 2 & \text{if larger than High\_Th} \\ 4 & \text{if larger than High\_ThX2} \end{cases}$$

For 16X8 or 8X16 mode

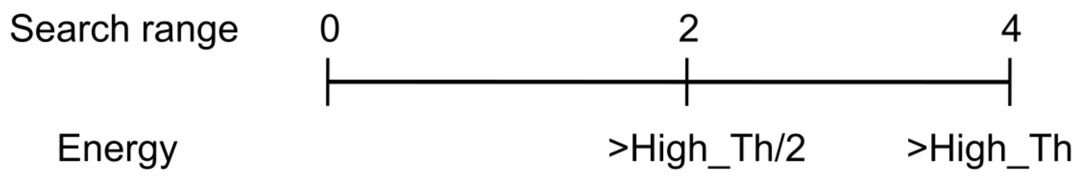
$$\text{Search range} = \begin{cases} 2 & \text{if larger than High\_Th/2} \\ 4 & \text{if larger than High\_Th} \end{cases}$$

For 8X8 mode

Formula (2)



(a) 16X8 or 8X16 mode



(b) 8X8 mode

Figure 4.9 Set the search range

The search range is defined in formula (2), where the *High_Th* is the same as the *high-threshold* defined previously in the subsection 4.1.2. In inter 16X8 mode, if the energy of the target block is lower than *High_Th*, the predicted motion vector directly is used. If the energy is larger than *High_Th* or even two times of *High_Th*, the search range of the refinement window is set be 2 or 4 pixels in order to find a better motion vector efficiently. The inter 8X16 and 8X8 mode are performed similar motion vector refinement. Figure 4.10 shows an example for inter 8X16 mode, where assume Eng(A) is less than *High_Th*, while Eng(B) is in between *High_Th*X2 and *High_Th*. In this case, the 8X16 partition A block uses the predicted motion vector directly, but the predicted motion vector of the partition B block needs to be refined.

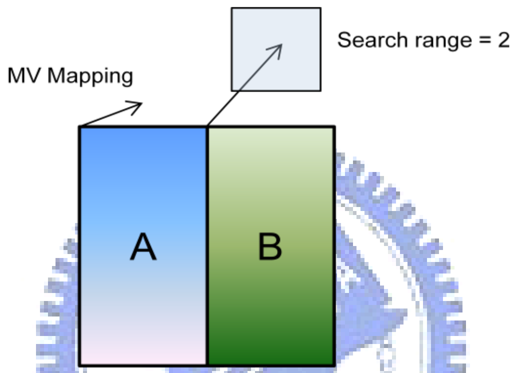


Figure 4.10 Eng(A) < High_Th and High_ThX2 > Eng(B) > High_Th

4.2.4 Sub-Pixel Refinement

After the above process is finished, the best integer motion vector for each target partition block is estimated. We propose to do a sub-pixel motion refinement in order to combat the difference from the half pixel interpolation methods used in MPEG-2 and H.264. We first perform half-pixel refinement around the best integer motion vector and finally quarter-pixel refinement around the best half-pixel motion vector.

Chapter 5

Experimental Results

In the chapter, we compare the proposed method with the "Chen's algorithm" [9] and the "Xin's algorithm" [13], which speed up the block mode decision and motion vector decision respectively. We also compare the proposed transcoder with the MPEG-2 to H.264 standard transcoder. The parameters of our experimental environment are set as follows:

CPU : Intel Pantium4 3.0 GHz

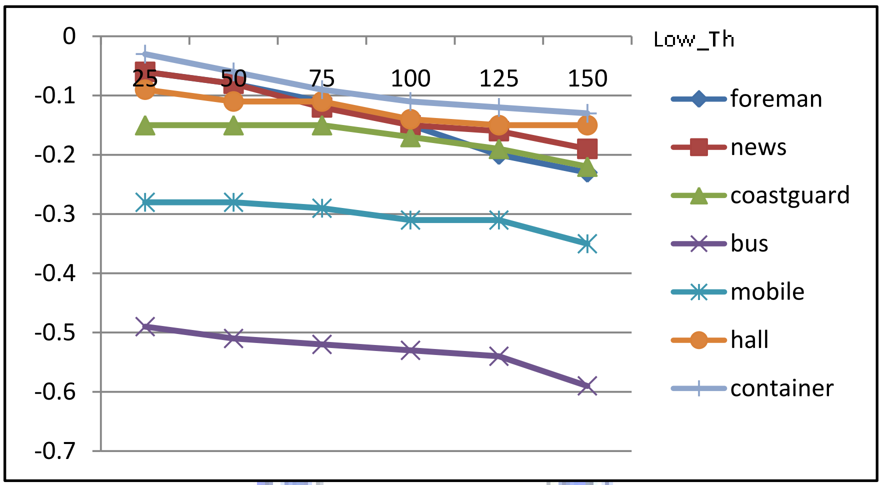
- Test sequence(frames): Foreman(120), Coastguard(120), Bus(120), News(120), Mobile(120), Stefan(90)
- Group of Picture (GOP): I P P P P ……
- GOP size: 30 frames
- Frame rate: 30 fps
- Frame format: CIF (352 x 288 pixels)
- Codec : MPEG-2(TM5), H.264(JM13.1)
- MPEG-2 bitrate : 3.2 Mbps
- RD Optimization : High complexity mode (if used)
- Motion Estimation : Search window size = 16
- Rate Control : used

- Inter Mode : Skip, 16X16, 16X8, 8X16, and 8X8 are enabled

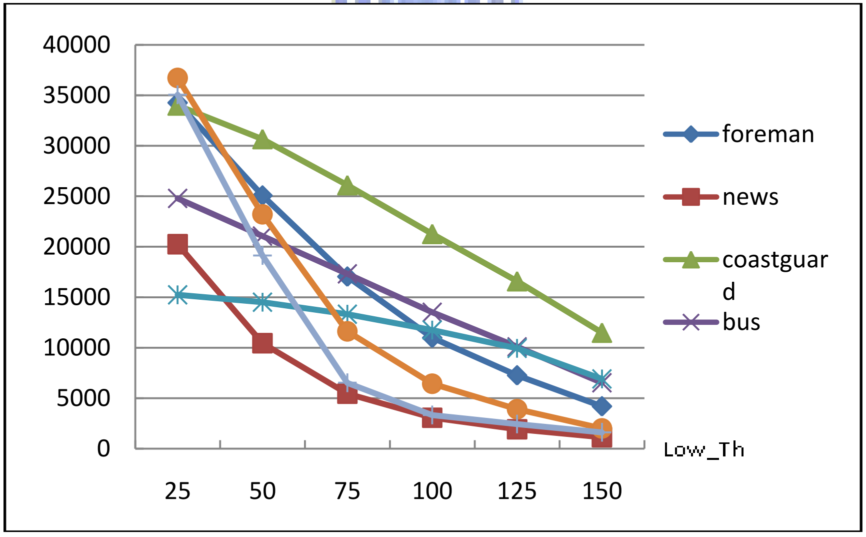
As mentioned above, in our experiment the input MPEG-2 bitstream is encoded at a bitrate of 3.2 Mbps. And the output H.264 bitstreams are encoded at various bitrates in order to compare the performance at the different bitrates and the rate-control of the H.264 standard is used. The frame structures of both MPEG-2 and H.264 are IPPP structure, and the “High complexity mode” is enabled in the H.264 encoding process if the RD-optimization is used. We also experimented the results of [9] and [13] algorithms in the same condition of their paper to verify the correctness. However, in order to make fair comparison, we set the different experimental environment from their paper in following experiment.

5.1 Threshold Determination

First, experiments are conducted for various *high thresholds* and *low thresholds* in order to find the best thresholds. From chapter 4 we know that if *high-threshold* is set too high or *low-threshold* is set too low, then the number of undecided blocks increases and thus the computation overhead also increases. On the other hand, if *high-threshold* is set too low or *low-threshold* is set too high, then the probability that the proposed block mode decision method makes wrong decision increases and thus reduces the video quality for a given bitrate. Therefore, for better choice of both *high-threshold* and *low-threshold*, we need to make a trade off between video quality and the number of undecided blocks. Figure 5.1 and 5.2 show the experiment with different *low thresholds* and *high thresholds* respectively. In Figure 5.1, the x-coordinate means the value of *low threshold*. The y-coordinate in Figure 5.1(a) and 5.1(b) mean quality loss in terms of PSNR (△PSNR) and the number of the undecided macroblocks, respectively. It is observed that, for most of the streams, when the *low threshold* was larger than 100, the video quality lost a lot but only few number of undecided macorblocks decreased. Similarly, when the *low threshold* is less than 50, although PSNR improved, the number of undecided macroblocks also increased a lot. Therefore, we propose to set the *low-threshold* in between 50 and 100. On the other hand, in Figure 5.2, while the *high threshold* is less than 200, the video quality degraded dramatically although the number of undecided blocks can be reduced a little bit. Therefore, it is better to set the *high-threshold* more than or equal to 200. In the following experiments, the *low-threshold* and the *high-threshold* are set to 75 and 200, respectively..

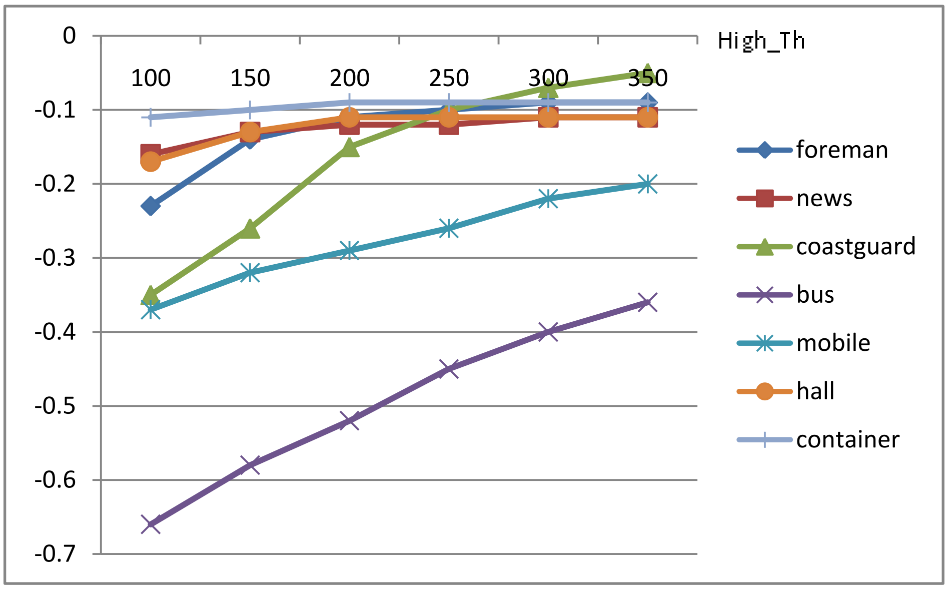


(a) △PSNR

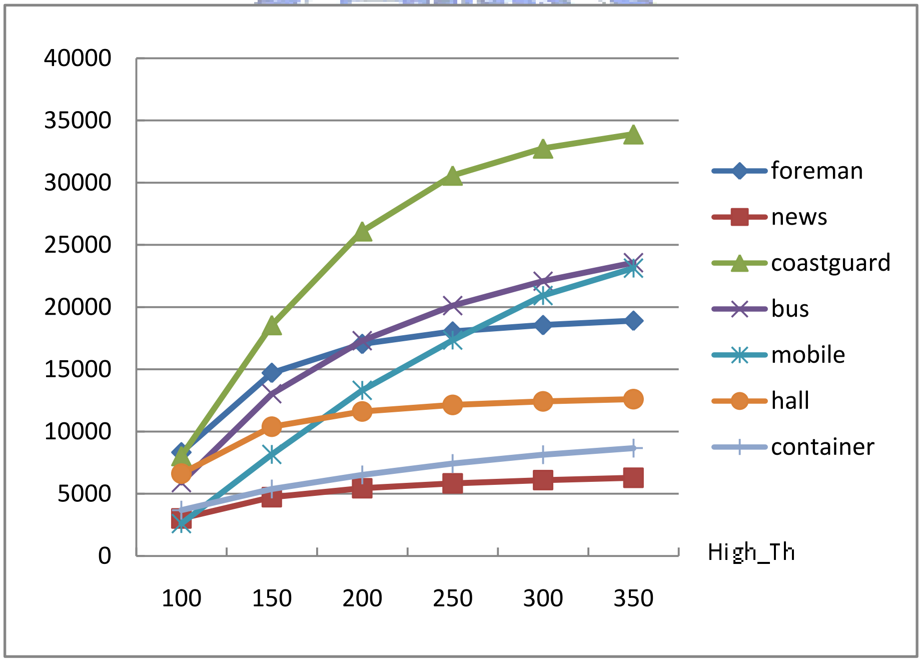


(b) Number of undecided MBs

Figure 5.1 the experiment with different low thresholds and a fixed high threshold = 200



(a) △PSNR



(b) Number of undecided MBs

Figure 5.2 the experiment with different high thresholds and a fixed low threshold = 75

5.2 Experiment at Fixed Bitrate

Foreman						
	PSNR(dB)	Total Time(s)	ME Time(s)	△PSNR	△T-Time(%)	△ME-Time(%)
re-encoder	40.09	749.354	362.389	0	0.00	0.00
re-encoder-fast	40.09	568.151	177.281	0	-24.18	-51.08
Ref [9]	39.71	161.773	101.157	-0.38	-78.41	-72.09
Proposed-MD	39.85	181.832	109.597	-0.24	-75.73	-69.76
Ref [13]	39.9	235.628	90.046	-0.19	-68.56	-75.15
Proposed-MVD	40	237.81	93.121	-0.09	-68.26	-74.30
Ref [9] + [13]	39.46	79.66	20.843	-0.63	-89.37	-94.25
Proposed algorithm	39.81	100.236	29.256	-0.28	-86.62	-91.93
News						
	PSNR(dB)	Total Time(s)	ME Time(s)	△PSNR	△T-Time(%)	△ME-Time(%)
re-encoder	44.22	773.529	351.791	0	0.00	0.00
re-encoder-fast	44.21	599.238	171.691	-0.01	-22.53	-51.20
Ref [9]	44	156.197	96.415	-0.22	-79.80	-72.60
Proposed-MD	44.03	165.626	101.22	-0.19	-78.59	-71.23
Ref [13]	44.08	230.907	86.332	-0.14	-70.15	-75.50
Proposed-MVD	44.15	232.573	87.455	-0.07	-69.93	-75.14
Ref [9] + [13]	43.86	74.236	16.702	-0.36	-90.40	-95.25
Proposed algorithm	44.02	81.44	19.203	-0.20	-89.47	-94.54
Bus						
	PSNR(dB)	Total Time(s)	ME Time(s)	△PSNR	△T-Time(%)	△ME-Time(%)
re-encoder	33.74	763.143	366.361	0	0.00	0.00
re-encoder-fast	33.74	593.49	182.552	0	-22.23	-50.17
Ref [9]	33.04	180.981	106.752	-0.7	-76.28	-70.86
Proposed-MD	33.17	198.711	115.698	-0.57	-73.96	-68.42
Ref [13]	33.01	246.396	90.692	-0.73	-67.71	-75.25
Proposed-MVD	33.44	250.894	98.553	-0.3	-67.12	-73.10
Ref [9] + [13]	31.25	96.438	25.722	-2.49	-87.36	-92.98
Proposed algorithm	33.06	117.9	37.015	-0.68	-84.55	-89.90

coastguard						
	PSNR(dB)	Total Time(s)	ME Time(s)	△PSNR	△T-Time(%)	△ME-Time(%)
re-encoder	34.69	768.36	386.824	0	0.00	0.00
re-encoder-fast	34.68	565.746	179.1	-0.01	-26.36	-53.70
Ref [9]	34.27	166.214	104.189	-0.42	-78.37	-73.07
Proposed-MD	34.44	193.26	116.782	-0.25	-74.85	-69.81
Ref [13]	34.46	235.97	92.049	-0.23	-69.29	-76.20
Proposed-MVD	34.62	243.555	101.079	-0.07	-68.30	-73.87
Ref [9] + [13]	34.04	83.874	22.8	-0.65	-89.08	-94.11
Proposed algorithm	34.42	113.077	36.748	-0.27	-85.28	-90.50
Mobile						
	PSNR(dB)	Total Time(s)	ME Time(s)	△PSNR	△T-Time(%)	△ME-Time(%)
re-encoder	29.98	755.179	365.833	0	0.00	0.00
re-encoder-fast	29.98	588.965	182.121	0	-22.01	-50.22
Ref [9]	29.57	189.846	118.112	-0.41	-74.86	-67.71
Proposed-MD	29.64	195.953	118.448	-0.34	-74.05	-67.62
Ref [13]	29.81	241.891	93.03	-0.17	-67.97	-74.57
Proposed-MVD	29.95	252.752	104.79	-0.03	-66.53	-71.36
Ref [9] + [13]	29.4	103.664	32.159	-0.58	-86.27	-91.21
Proposed algorithm	29.69	114.54	39.51	-0.29	-84.83	-89.20
Stefan						
	PSNR(dB)	Total Time(s)	ME Time(s)	△PSNR	△T-Time(%)	△ME-Time(%)
re-encoder	34.64	534.752	246.347	0	0.00	0.00
re-encoder-fast	34.64	420.128	130.687	0	-21.43	46.95
Ref [9]	33.87	127.329	79.569	-0.77	-76.19	-67.70
Proposed-MD	33.99	137.013	84.639	-0.65	-74.38	-65.64
Ref [13]	34.04	173.255	66.452	-0.6	-67.60	-73.03
Proposed-MVD	34.36	178.563	72.187	-0.28	-66.60	-70.70
Ref [9] + [13]	33.33	65.925	17.763	-1.31	-87.67	-92.79
Proposed algorithm	34.13	77.392	24.556	-0.51	-85.53	-90.03

Table 5.1 Experiment in bitrate = 1.2Mbps

In this subchapter, we examined the transcoding performance and complexity at a fixed target bitrate of 1.2 Mbps. The performance is measured in terms of the reduced PSNR of the video after transcoding, while the complexity is measured in terms of the reduced re-encoding time. In order to evaluate the respective effect of the proposed mode decision and motion vector decision algorithms, we have conducted the experiments for the following methods.

- Fast mode decision (proposed-MD)
- Fast motion vector decision (proposed-MVD)
- Proposed integrated algorithm (including proposed-MD and proposed-MVD)

We made a comparison for these proposed methods with current state-of-two-art methods listed below.

- Mode decision method proposed in [9]
- Motion vector decision method proposed in [13]
- Integrated method of the above two (i.e. [9]+[13])
- Full H.264 re-encoder with full search (re-encoder)
- Fast H.264 re-encoder with fast full search (re-encoder-fast)

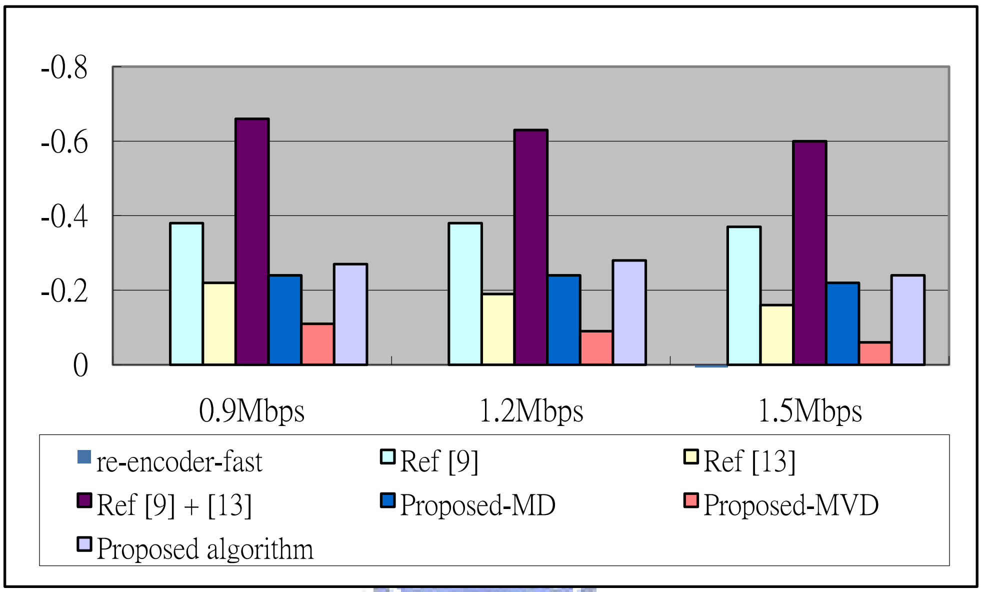
The results are shown in Table 5.1, where the △PSNR stands for the degradation of PSNR compared with full H.264 re-encoder, △T-Time and △ME-Time stand for the percentage of reduced processing time, also compared with full H.264 re-encoder, as shown in follows :

$$\Delta\text{Time}(\%) = \frac{(\text{fast method processing time}) - (\text{H.264 processing time})}{\text{H.264 processing time}} \times 100\%$$

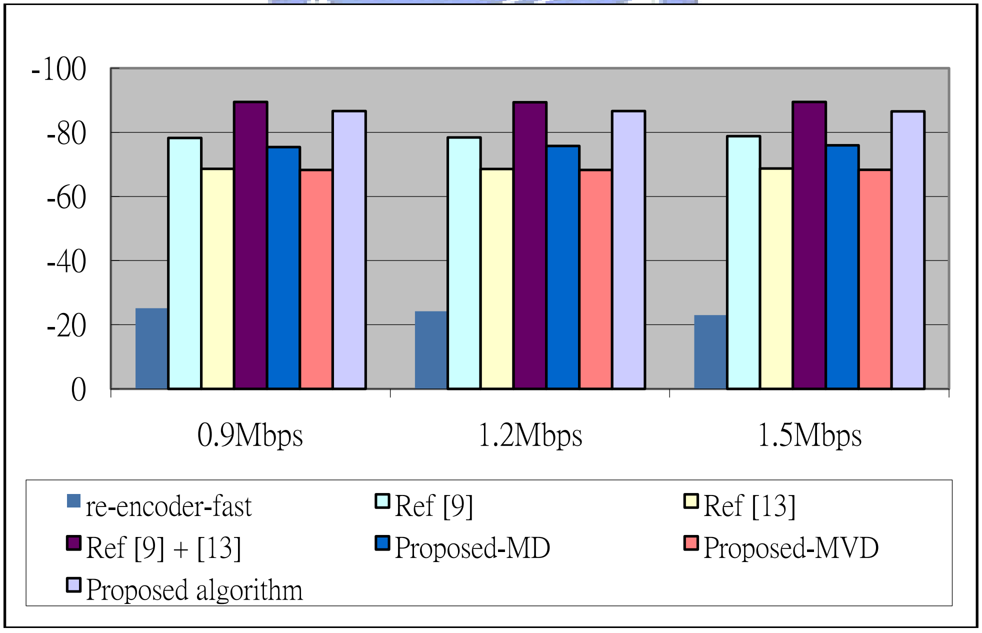
In Table 5.1, the video quality of H.264 re-encoder with fast full search is similar to the re-encoder used full search and it can reduce about 22% processing time. Compared with the full H.264 re-encoder, the algorithm in [9] and [13] can reduce total encoding time up to 75% and 65% respectively with small PSNR degradation (0.48dB and 0.34dB respectively on the average). The integrated method of [9] and [13] can reduce more processing time however, with more PSNR degradation.

Compared with [9], the proposed mode decision method can increase PSNR of 0.11dB and compared with [13], the proposed motion vector decision method can increase PSNR of 0.21dB with the increase of a little computational cost. However, the proposed integrated algorithm can reduce total re-encoding time by 84.55% to 89.47% with small PSNR degradation (0.36dB on the average), compared with full H.264 re-encoder. Especially, in the high motion video (e.g. *Bus* and *Stefan*), the quality of the video encoded by the proposed integrated algorithm performs even better and the processing time is also faster than [9] and [13]. The reasons that the proposed integrated method performs much better on high motion video are due to that the proposed mode decision method choose a better mode in the hard decision macroblock and the proposed motion vector decision method refine the predict motion vector in the partition blocks with more residual coefficient.

5.3 Experiment at Various Bitrate

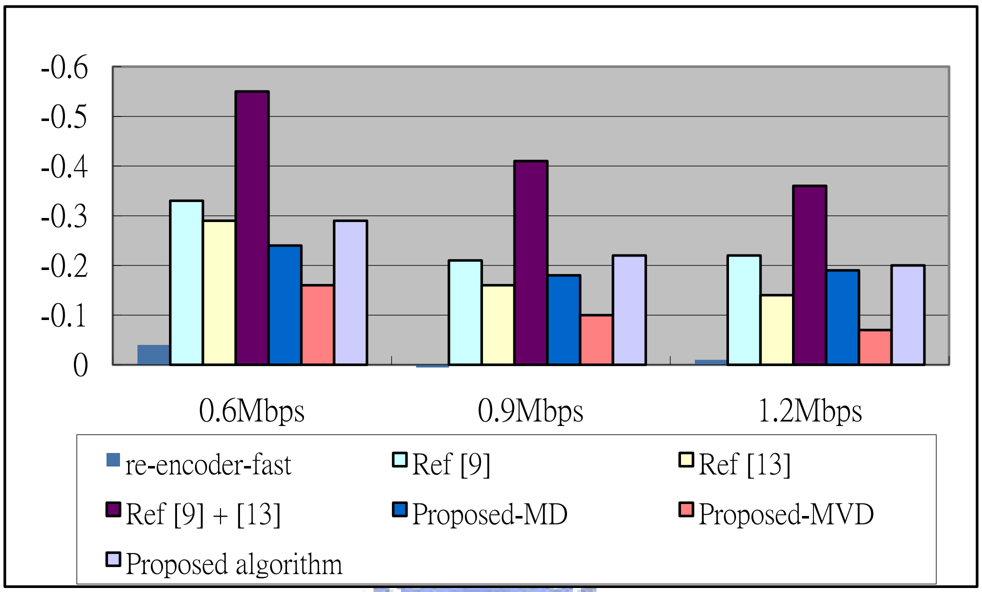


△PSNR

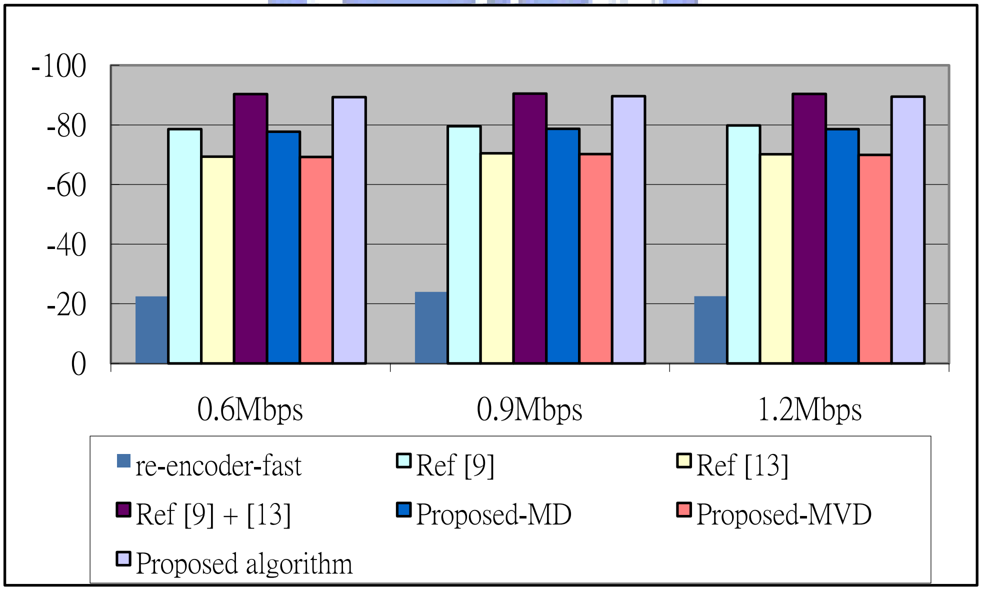


△T-Time

Figure 5.3 (a) Foreman

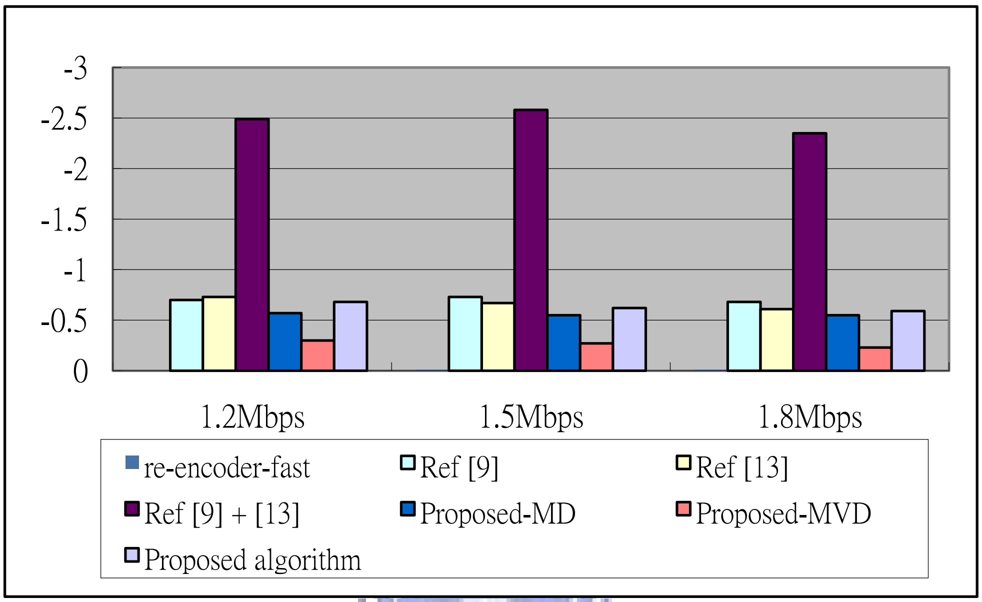


△PSNR

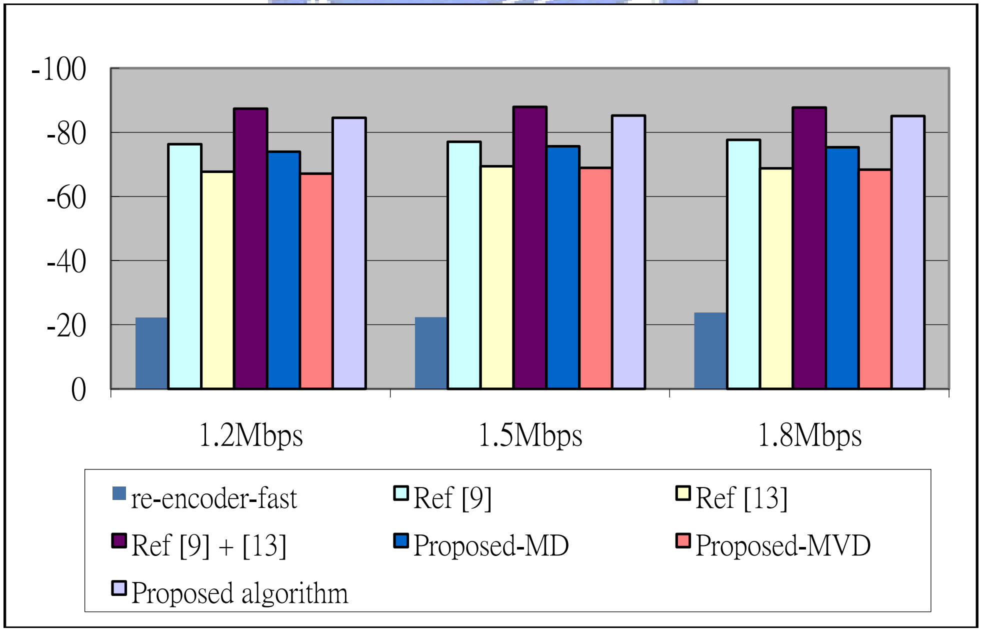


△T-Time

Figure 5.3 (b) News

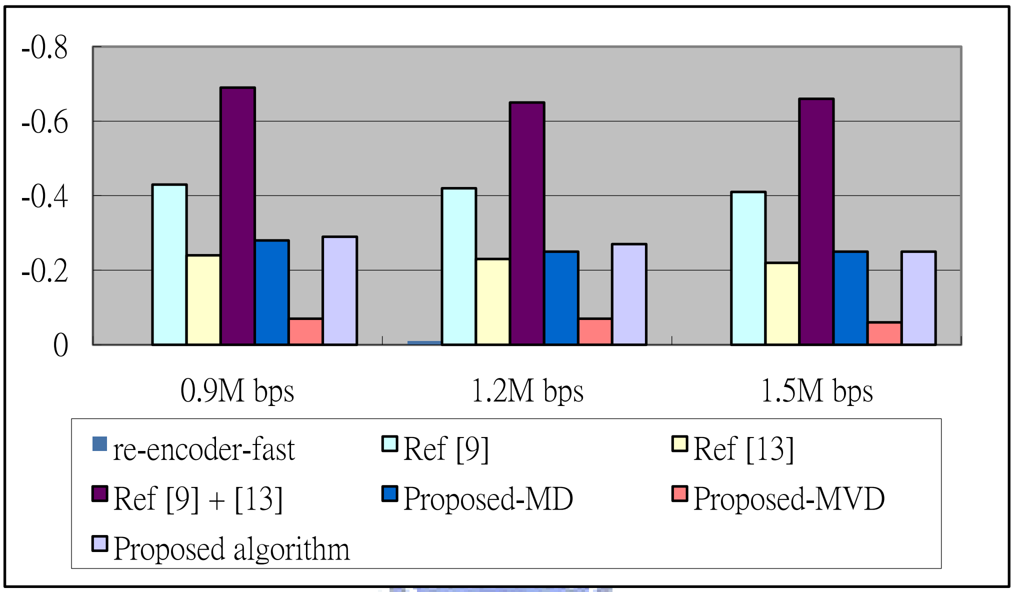


△PSNR

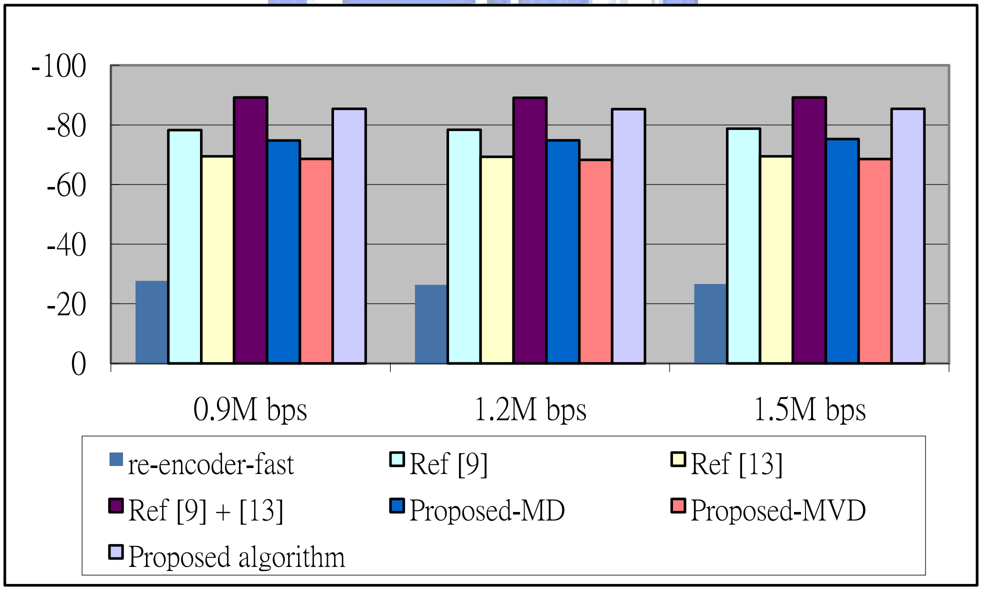


△T-Time

Figure 5.3 (c) Bus

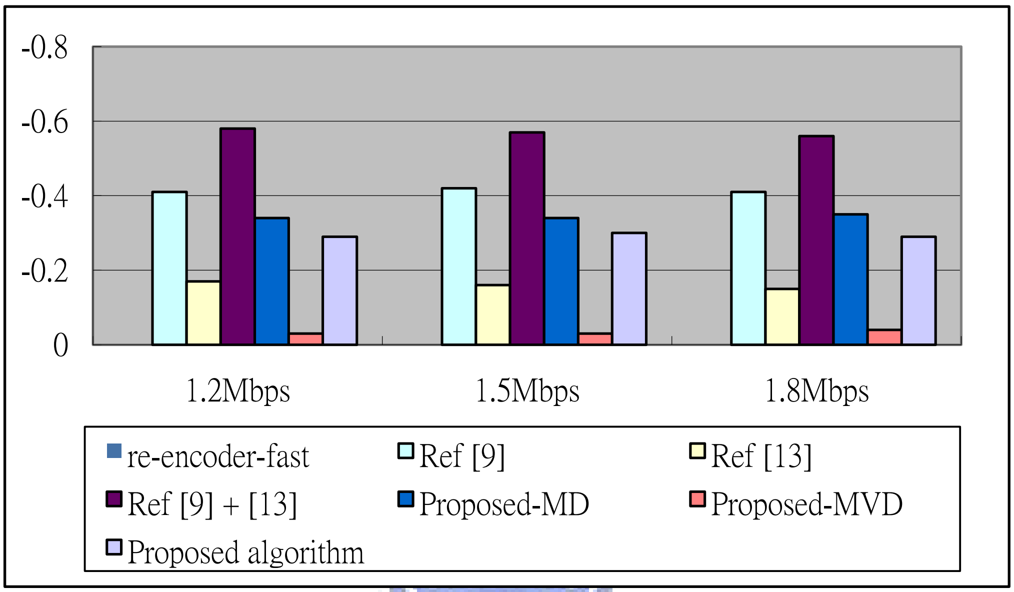


△PSNR

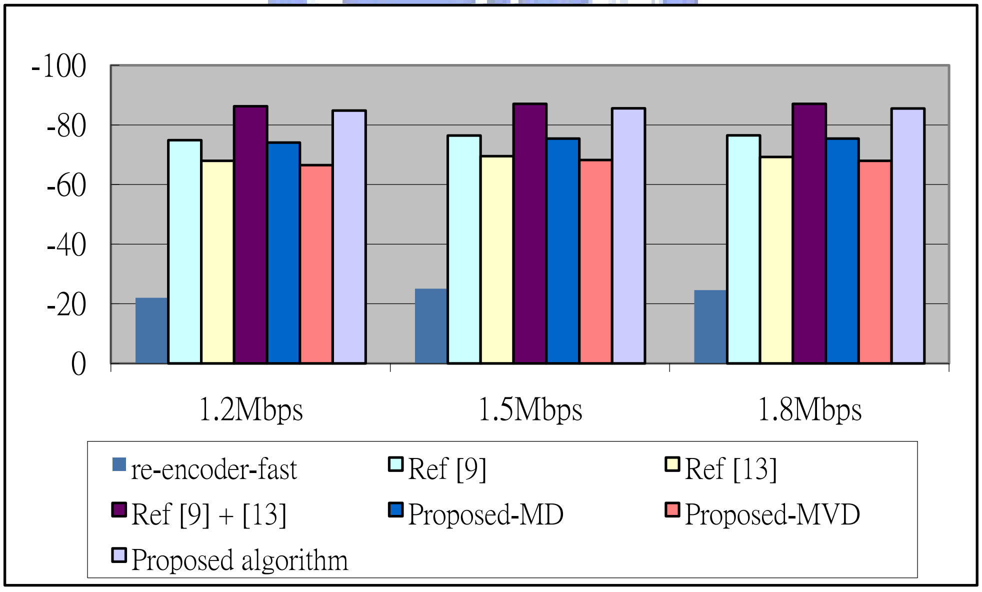


△T-Time

Figure 5.3 (d) Coastguard

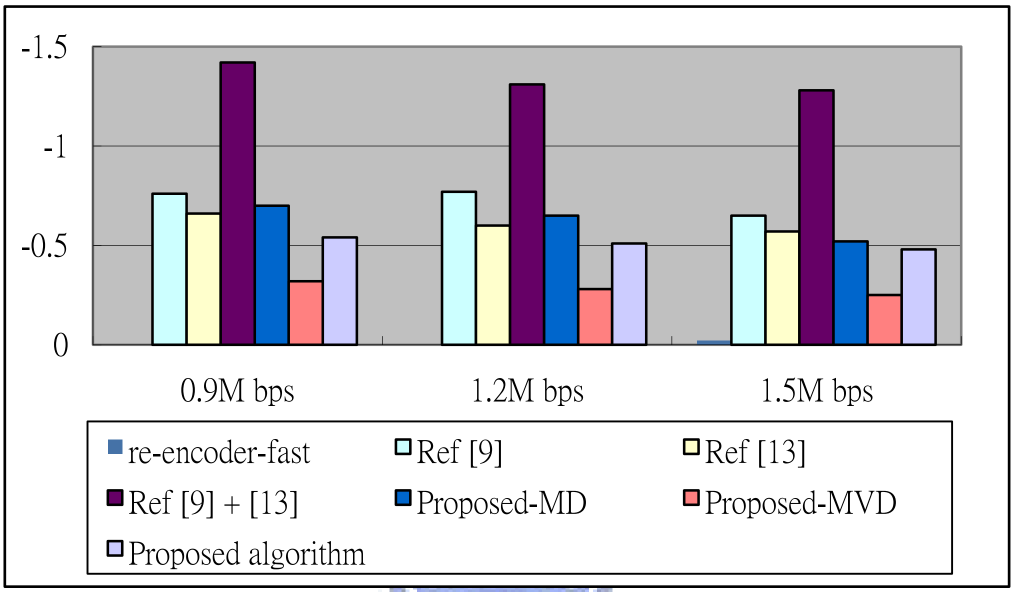


△PSNR

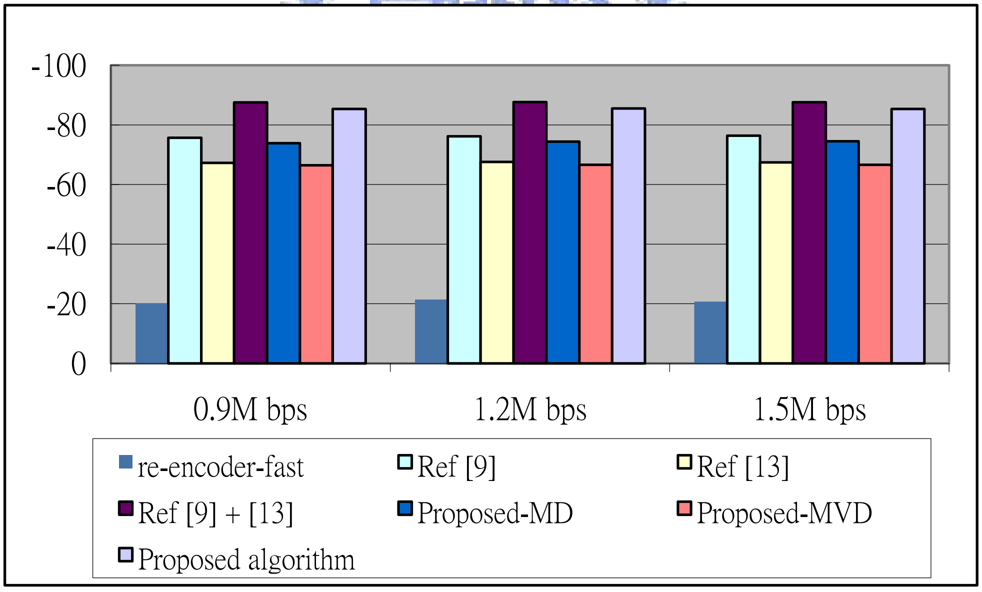


△T-Time

Figure 5.3 (e) Mobile



△PSNR



△T-Time

Figure 5.3 (f) Stefan

Figure 5.3 Experiment results at various bitrates

We also conducted experiments on the transcoding methods for six different streams at three different bitrates and the results are shown in Figure 5.3. △PSNR means the degradation of the PSNR compared with full H.264 re-encoder and

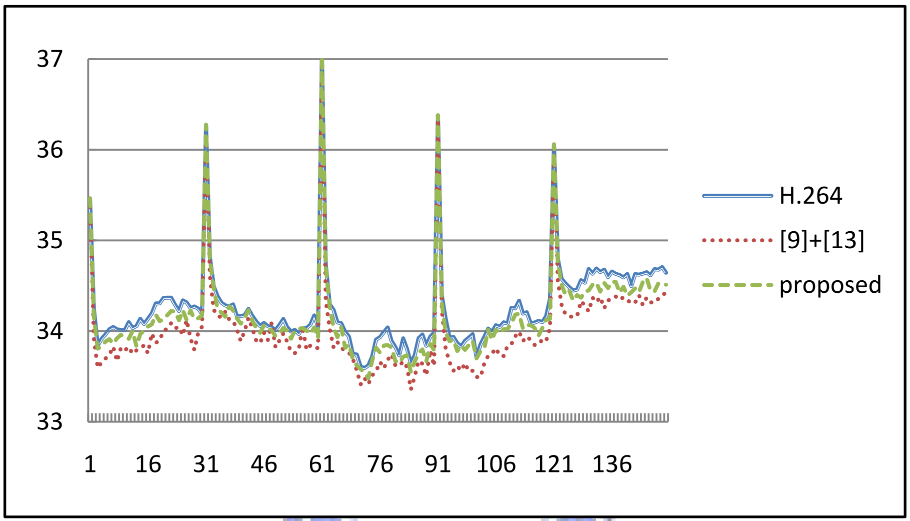
△T-Time is the percentage of reduced re-encoding time. The results for all the streams show the video quality of the proposed integrated algorithm is always better than the integrated transcoder of [9] and [13] with a little more processing time at different bitrate, especially in *Bus* and *Stefan*. Besides, while the re-encoding bitrate is larger, the degradation of PSNR is usually less with the same processing time. In the most of the streams, the degradation of PSNR at low bitrate is larger than at high bitrate since a bad block mode or a bad motion vector caused more distortion when the re-encoder used larger QP. However, since the proposed integrated algorithm chose better block modes and better motion vectors, the degradation of PSNR at low bitrate increased little. On the other hand, the PSNR of the integrated transcoder of [9] and [13] is much lower than the mode decision method of [9] and the motion vector decision method of [13] respectively, especially in Bus and Stefan sequences. The reasons are due to that bad block modes and bad motion vectors could cause more video quality loss.

5.4 Frame by Frame at Fixed QP = 30

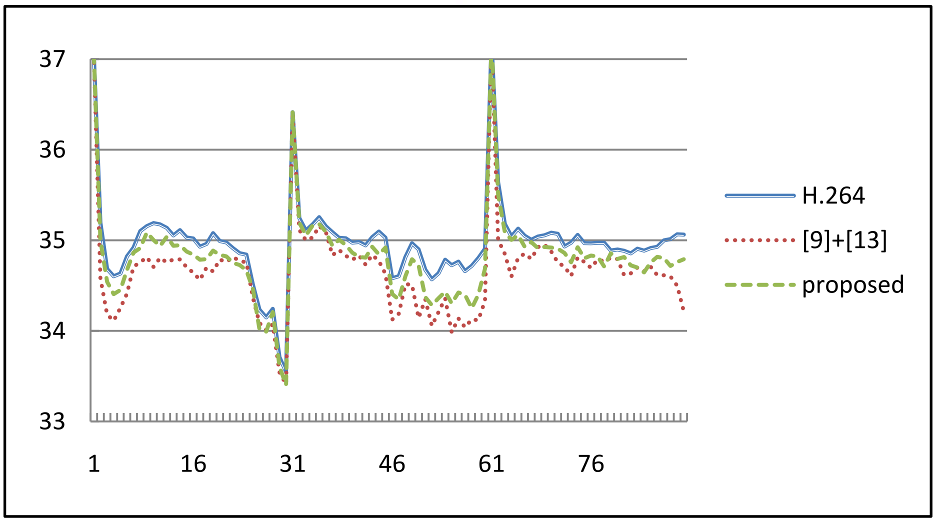
In this subchapter, we compare the proposed integrated algorithm with full H.264 encoder and the integrated method of [9] and [13] at fixed QP (30) and GOP size is 30 frames. We use *Bus* and *Stefan* video sequences which belong to high motion video. We re-encode Bus and Stefan for 150 frames and 90 frames, respectively. In *Bus* sequence, the integrated [9] and [13] method reduced computational cost up to 87% with 0.25dB PSNR degradation and increasing bitrate up to 41% and our proposed integrated algorithm reduced processing time up to 85% with 0.1dB PSNR degradation and increasing 9.63% bitrate, as shown in Table 5.2. In other words, the encoding performance of our proposed integrated algorithm is much better than the integrated [9] and [13] transcoder with slight increase in processing time.

Bus					
	PSNR(dB)	bitrate(Kbps)	T-Time(s)	△bitrate(%)	△T-Time(%)
H.264	34.25	1286.10	969.757	0	0
[9]+[13]	34(-0.25)	1820.54	121.676	41.56%	-87.45%
proposed	34.15(-0.1)	1409.98	143.194	9.63%	-85.23%
Stefan					
	PSNR(dB)	bitrate(Kbps)	T-Time(s)	△bitrate(%)	△T-Time(%)
H.264	34.97	1263.55	541.482	0	0
[9]+[13]	34.66(-0.32)	1503.6	67.139	19.00%	-87.60%
proposed	34.81(-0.17)	1340.43	77.328	6.08%	-85.72%

Table 5.2 Experiment at fixed QP = 30



(a) Bus



(b) Stefan

Figure 5.4 Comparison in frame by frame

Figure 5.3 shows the PSNR value of every frame in Bus and Stefan for the proposed integrated algorithm, the integrated [9] and [13] transcoder, and full H.264 re-encoder. As can be seen in the figure, the PSNR value of every frame of our proposed algorithm is usually better. Moreover, since the first frame of a GOP is intra frame which is coded by H.264 standard encoder, the PSNR of the first frame of a GOP in the three different re-encoders is the same. When the frames are at the end of GOP, the PSNR of the integrated [9] and [13] transcoder decreased more but the PSNR of the proposed integrated algorithm is usually stable since our algorithm can decide block modes and motion vectors correctly and efficiently, therefore, reducing coding mode mismatch between MPEG-2 and H.264.

Chapter 6

Conclusion and Future Work

In this thesis, we proposed a fast algorithm focusing on 16X16, 16X8, 8X16 and 8X8 block mode decision and motion vector decision in MPEG-2 to H.264 transcoding. The proposed mode decision method can alleviate the complexity of variable size block decision in H.264/AVC and the proposed motion vector method predicts good motion vectors and finds the best motion vector with the efficient refinement to reduce the computational cost of motion estimation.

The experimental results show that our algorithm can save a lot of computational cost while the video quality is reduced a little bit compared with the full H.264 re-encoder. Besides, the computation time of the proposed algorithm is less than the fast mode decision proposed in [9] and the fast motion mapping proposed in [13] with similar video quality.

In the future, we will focus on reducing the computational time of intra type prediction and deciding the motion vectors and prediction direction efficiently in B frame type for helps in real-time implementation of MPEG-2 to H.264 transcoding.

References

[1] Shih-Fu Chang and Anthony Vetro, "Video Adaptation: Concepts, Technologies, and Open Issues ", *Proceedings of the IEEE, Volume 93, No. 1,* January 2005

[2] Ishfaq Ahmad, Xiaohui Wei, Yu Sun and Ya-Qin Zhang, " Video Transcoding: An Overview of Various Techniques and Research Issues ", *IEEE Transactions on Multimedia, Volume 7, No. 5,* October 2005

[3] Hari Kalva, " Issues in H.264/MPEG-2 Video Transcoding ", *Proceedings of the IEEE Consumer Communications and Networking Conference,* January 2004, pp. 657-659

[4] Jun Xin, Anthony Vetro and Huifang Sun, " Converting DCT Coefficients to H.264/AVC Transform Coefficients ", *Proceedings of the IEEE Pacific-Rim Conference on Multimedia (PCM)*, November 2004, pp.939-946

[5] Tuanjie Qian, Jun Sun, Dian Li, Xiaokang Yang, Jia Wang, " Transform Domain Transcoding From MPEG-2 to H.264 With Interpolation Drift-Error Compensation ", *IEEE Transactions on Circuits and Systems for Video Technology,* Volume 16, No. 4, April 2006, pp.523-534

[6] Chen Chen, Ping-Hao Wu and Homer Chen, " MPEG-2 to H.264 Transcoding ", *Picture Coding Symposium,* December 2004

[7] Zhi Zhou, Shijun Sun, Shawmin Lei and Ming-Ting Sun, " Motion Information and Coding Mode Reuse for MPEG-2 to H.264 Transcoding ", *IEEE International Symposium on Circuits and Systems,* Volume 2, May 2005 pp.1230-1233

[8] Hari Kalva and Branko Petljanski, " Exploiting the Directional Features in MPEG-2 for H.264 Intra Transcoding ", *IEEE Transactions on Consumer Electronics*, Volume 52, No. 2, May 2006

[9] Gao Chen, Yong-dong Zhang, Shou-xun Lin and Feng Dai, " Efficient Block Size Selection for MPEG-2 to H.264 Transcoding ", *Proceedings of the 12th Annual ACM International Conference on Multimedia,* October 2004, pp.300-303

[10] Shen Li, Lingfeng Li, Takeshi Ikenaga, Shunichi Ishiwata, Mastaka Matsui and Satoshi Goto, " Complexity Based Fast Coding Mode Decision for MPEG-2/H.264 Video Transcoding ", *IEEE Asia Pacific Conference on Circuits an Systems*, December 2006, pp. 574-577

[11] Donghyung Kim, Kicheol Jeon, Jongho Kim and Jechang Jeong, " A Fast MPEG2-to-H.264 Transcoding Algorithm in Spatial Domain ", *in IWAIT*, 2006, pp.695-700

[12] H. Kato, A. Yoneyama, Y. Takishima and Y. kaji, " Coding Mode Decision For High Quality MPEG-2 to H.264 Transcoding ", *IEEE International Conference on Image Processing*, October 2007

[13] Jun Xin, JianJun Li, Anthony Vetro, Huifang Sun and Shun-ichi Sekiguchi, " Motion Mapping for MPEG-2 to H.264/AVC Transcoding ", *IEEE International Symposium on Circuits and System*, May 2007, pp. 1991-1994

[14] Gerardo Fernandez-Escribano, Hari Kalva, Pedro Cuenca and Luis Orozco-Barbosa, " Reducing Motion Estimation Complexity in MPEG-2 to H.264 Transcoding ", *IEEE International Conference on Multimedia and Expo,* July 2007, pp.440-443