

國立交通大學

電子工程學系 電子研究所

碩士論文

用於HEVC編碼單元之快速決策演算法 與
結合移動向量與DCT之H. 264編碼器優化

Fast HEVC Coding Unit Decision Algorithm and
Combined MV and DCT Optimization for
H. 264/AVC Codec

研究生：許維哲

指導教授：杭學鳴 博士

中華民國一〇一年七月

用於HEVC編碼單元之快速決策演算法 與
結合移動向量與DCT之H. 264編碼器優化

Fast HEVC Coding Unit Decision Algorithm and
Combined MV and DCT Optimization for
H. 264/AVC Codec

研究生：許維哲

Student : Wei-Jhe Hsu

指導教授：杭學鳴 博士

Advisor : Dr. Hsueh-Ming Hang

國立交通大學
電子工程學系 電子研究所
碩士論文

A Thesis
Submitted to Department of Electronics Engineering and
Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Electronics Engineering

July 2012
Hsinchu, Taiwan, Republic of China

中華民國一〇一年七月

用於 HEVC 編碼單元之快速決策演算法 與 結合移動向量與 DCT 之 H.264 編碼器優化

研究生：許維哲

指導教授：杭學鳴 博士

國立交通大學 電子工程學系 電子研究所碩士班

摘要

由於高解析度影像應用的需求，視訊編碼在 3C 產品中是不可或缺的技術，例如行動電話、高畫質電視、藍光光碟機。進階視訊編碼(Advanced Video Coding, AVC/H.264)是目前商業產品中，廣泛採用的壓縮標準格式。為了達到更高的編碼效率，國際組織 JCT-VC 正在進行下一代標準的制定，即高效率視訊編碼(High Efficiency Video Coding, HEVC)。相較於進階視訊編碼，雖然高效率視訊編碼的複雜度提升許多，但是在相似的影像品質下，可以增加近一倍的壓縮效率。

此論文包含兩個研究主題：第一個主題是改善進階視訊編碼中，整數精確度的移動估測以增進編碼效能；第二個主題是關於高效率視訊編碼的編碼單元(Coding Unit, CU)大小的快速決策，以達到降低編碼器複雜度的目標。在進階視訊編碼中，整數移動估測的失真項，是以區塊之絕對誤差總和(The Sum of the Absolute Distortion, SAD)來計算，但是此方法並不能完全反應最後結果的失真。為了在相似的畫面品質下，進一步節省位元率，我們提出迭代的位元率-失真(Rate-Distortion, R-D)計算方式，以選擇較佳的移動向量。我們將此演算法實現於 JM18.0，用許多組 MPEG 測試影像來檢驗此方法的效能，並將執行結果和原始 JM 做法的結果進行比較。雖然 JM18.0 是發展已久的優化編碼器，我們仍可從中節省 1.1%至 4.2%的位元率，但代價是增加 45%的運算複雜度。

另一方面，高效率視訊編碼在傳統的編碼流程中，增加了編碼單元四元分割樹的構造。彈性的編碼單元設計提升了編碼效率，但相較於進階視訊編碼傳統的巨區塊 (Macroblock, MB) 結構而言，編碼複雜度提升不少。因此我們設計快速演算法以有效率地建造出編碼單元四元分割樹，其中演算法包括分裂決策、終止決策。這些快速編碼單元大小決策參考週遭相關的編碼單元之切割資訊以進行判斷。此外，我們設計額外的工具以增進我們提出的演算法效能，其中包含畫面層級加速控制和跳過決策後的快速預測單元判斷。最後，我們分析提出的快速演算法，並和 HM5.0 中的兩種快速演算法進行比較，以找出有效率的結合方法。相較於 HM5.0 的原始設定，我們提出的快速演算法，經過多組高解析度的影像測試，可以節省高達 49% 的整體編碼時間，但平均損失 0.06dB 的峰值信噪比 (PSNR)。



Fast HEVC Coding Unit Decision Algorithm and Combined MV and DCT Optimization for H.264/AVC Codec

Student : Wei-Jhe Hsu

Advisor : Dr. Hsueh-Ming Hang

Department of Electronic Engineering & Institute of Electronics

National Chiao Tung University

Abstract

With the growing demand for high resolution video applications, video coding is an indispensable element in many 3C products, such as mobile phone, DTV, and BD player. Today, Advanced Video coding (AVC/H.264) is one of the most popular video formats in commercial applications. Aiming at higher compression efficiency, the international JCT-VC is currently developing the next generation standard, High Efficiency Video Coding (HEVC). With a much higher encoder complexity, HEVC is able to achieve a 50% bitrate reduction compared to H.264/AVC.

This thesis has two topics, one is the enhanced motion estimation (ME) for AVC/H.264 and the other is the fast coding unit (CU) decision for HEVC. In H.264, the sum of the absolute difference (SAD) is used as the distortion term in ME, but it does not reflect the final coding distortion. To achieve further bitrate reduction, we propose an enhanced motion vector selection method based on the iterative R-D calculation. We compare the proposed method with the original H.264/AVC JM18.0 reference software on several MPEG test sequences. Although

JM18.0 is a highly optimized scheme, we can still obtain a BD-rate improvement from 1.1% to 4.2% but with additional 45% complexity increase.

In HEVC, the CU quadtree structure is added to the traditional fixed size macroblock. With flexible CU size selection, the coding efficiency increases but the complexity of HEVC becomes much higher than that of AVC/H.264 fixed macroblock (MB) structure. To reduce computational complexity, we propose a fast algorithm, which includes the splitting decision and the termination decision, in building the CU quadtree. The fast CU size decision of the current CU makes use of the size information of its neighboring CUs. Furthermore, we design the additional tools to enhance the performance of the proposed algorithm. The additional tools include the frame level acceleration and the fast PU size decision after the splitting decision. At the end, we compare it with the existing fast algorithms in HM5.0 and find an efficient way to blend them together. In comparison to the original HM5.0, our method saves the overall encoding time up to 49% with 0.06 dB average PSNR drop.

誌謝

鳳凰花開，又到了畢業的季節。而我也終於拿到了我的碩士學位。當初，剛開始進入交大電子研究所就讀時，因為是跨領域（從固態換系統），系統組的老師實在難找，領域也選擇了許久。幸運地，杭學鳴教授願意指導我。老師開啟了我的學術之路，提供了充足的研究環境和有趣的研究題目，使我心中名為研究生的種子慢慢地發芽、成長；老師豐富的學術知識和謙遜的人生態度更是我學習的目標，在此誠摯地感謝我的指導老師杭學鳴教授。

在我碩士兩年的求學之旅中，最有回憶的地方就是 Commlab。在這邊，我接觸到許多強者學長：感謝朝雄學長辛苦地管理實驗室，並且常和我分享研究的甘苦；謝謝峻利學長和宸銜學長平時跟我討論研究和課業；感謝書緯學長教我如何看 CODE、改 CODE，在學長畢業後還一直熱心地提供我 HEVC 方面的技術支援；謝謝崇豪學長，讓我了解做研究該有的熱情；謝謝鴻志學長，您留下的 MATLAB 教材非常的實用；感謝家揚學長在我做 AVC 計劃時，教我如何下手改 CODE；謝謝彰哲學長和柏森學長在我找工作時，給予我人生道路和工作態度的建議；學長們無論在研究、課業、人生態度都給我很大的啟示和幫助。另外，我也在這邊碰到了許多有趣的同學：感謝讀修從我準備考研究所開始，就不斷地幫助我解決數學上的問題；謝謝義文平時約我去健身，鍛鍊身體兼紓發壓力；感謝士傑與我一起準備七月的口試，讓我得以順利地通過口試。Commlab 的人、事、物

在這兩年來給予我很大的幫助，我在此由衷地感謝。這邊也特別感謝 MAPL 的俊吉學長和彥宇，在我口試之前，幫我確認基本的想法，消除我緊張的心情。

最後我想感謝的是交大和我的家人。我在交大就讀的六年之中，我從交大得到許多重要的東西和回憶，在此特別感謝母校。感謝我的家人，總是在背後支持著我，一路走來有他們的親情和支持，我才能有現在的成就。將此篇論文獻給所有關心我的人，因為有你們，我會盡力讓自己變得更好。

維哲 7/20 於 Commlab 筆

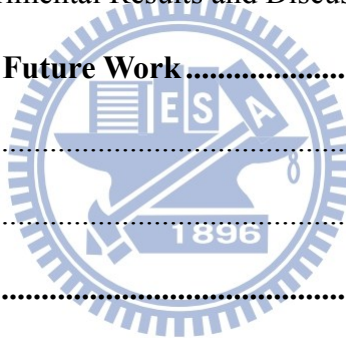


CONTENTS

摘要.....	i
Abstraction.....	iii
誌謝.....	v
CONTENTS.....	vii
LIST OF FIGURES.....	x
LIST OF TABLES.....	xii
Chapter 1 Introduction.....	1
1.1 Research Contributions.....	2
1.2 Thesis Organization.....	3
Chapter 2 Overview of H.264/AVC and HEVC.....	4
2.1 Advanced Video Coding.....	4
2.1.1 H.264 Architecture.....	5
2.1.2 Basic Coding Tools.....	5
2.1.2.1 Intra prediction.....	6
2.1.2.2 Inter Prediction.....	6
2.1.2.3 Transform and Quantization.....	7
2.1.2.4 Deblocking Filter.....	7
2.1.2.5 Entropy Coding.....	8
2.1.3 Encoder Control.....	8
2.1.3.1 Searching for Optimal Motion Vector.....	10
2.1.3.2 Selection for the Best Mode.....	10
2.2 High Efficiency Video Coding.....	11
2.2.1 Coding Unit Definition.....	12
2.2.1.1 Coding Unit.....	13

2.2.1.2	Prediction Unit	13
2.2.1.3	Transform Unit	14
2.2.2	Enhanced Coding Tools	15
2.2.2.1	Intra prediction	17
2.2.2.2	Inter Prediction	17
2.2.2.3	Transform and Quantization	18
2.2.2.4	Loop Filter	18
2.2.2.5	Entropy Coding	19
2.3	Experiment Conditions	19
Chapter 3	Nested Quadtree Coding Unit	22
3.1	Overview of Coding Unit Quadtree Structure	22
3.1.1	Partition Decision Flow of Nested Quadtree CU	23
3.1.2	Existing fast algorithms for Partition Decision Flow in HM5.0	25
3.2	Analysis of Nested CU Quadtree Structure	29
Chapter 4	Fast CU Size Decision Algorithm Design	32
4.1	Problem Formulation and Design Goal	32
4.2	Related Work	33
4.3	Core Ideas of Fast CU Size Decision	34
4.3.1	Splitting Decision	36
4.3.2	Termination Decision	37
4.3.3	Basic Fast CU Size Decision Scheme	38
4.4	Additional Tools	39
4.4.1	Frame Level Parameter Control	40
4.4.2	LCU Level Parameter Control with Error-Bound	45
4.4.3	2N×N/N×2N Decision after Splitting Decision	52

4.5	Overview of the Overall Proposed Algorithm	55
Chapter 5	HEVC Experiments and Discussions	57
5.1	Performance Measure	57
5.2	Experimental Results and Discussions	59
5.2.1	Fast CU Size Decision	59
5.2.2	Comparison with ECU/CFM	70
5.2.3	Combined Fast CU Size Decision with ECU/CFM.....	72
Chapter 6	Combined MV and DCT Optimization for H.264/AVC Codec.....	76
6.1	MV Refinement with DCT result.....	76
6.2	Modified MV Selection Scheme.....	78
6.3	AVC/H.264 Experimental Results and Discussions	81
Chapter 7	Conclusions and Future Work.....	88
7.1	Conclusions.....	88
7.2	Future Work	89
	REFERENCES	91



LIST OF FIGURES

Fig. 1 An H.264/AVC encoder.....	5
Fig. 2 R-D optimization for selecting MV and mode.....	9
Fig. 3 An HEVC encoder.....	12
Fig. 4 An Example of a nested quadtree structure [8]	15
Fig. 5 Possible PUs in low complexity setting	15
Fig. 6 An example of nested CU quadtree structure (Vidyo1, Frame 2, QP=32).....	23
Fig. 7 A G-BFOS example.....	25
Fig. 8 An example of ECU [18].....	28
Fig. 9 Program flow of CFM.....	29
Fig. 10 PU execution order in CU in the low-complexity setting	29
Fig. 11 Data representation of splitting information	34
Fig. 12 Reference CUs and the current CU.....	35
Fig. 13 An example of splitting decision.....	36
Fig. 14 An example of termination decision.....	38
Fig. 15 Flowchart of basic fast CU size decision algorithm.....	39
Fig. 16 R-D curve of Basketball.....	41
Fig. 17 Example of $N_c=3$	42
Fig. 18 Experiment for choosing N_c	44
Fig. 19 R-D curve of Basketball with N_c control	45
Fig. 20 Error bound (3%) for SAD.....	47
Fig. 21 Second order curve fitting for error bound (3%)	47
Fig. 22 Probability density distribution of SAD of “Kimono”	51
Fig. 23 Probability density distribution of SAD of “Party”	51
Fig. 24 An example of $2N \times N / N \times 2N$ Decision in depth 1	54

Fig. 25 An example of $2N \times N / N \times 2N$ Decision in depth 2	54
Fig. 26 Flowchart of overall proposed algorithm for processing an LCU	56
Fig. 27 CU distribution of the 9 th frame of BQsquare (QP=32).....	64
Fig. 28 CU distribution of the 9 th frame of Vidyo1 (QP=32)	65
Fig. 29 CU distribution of the 9 th frame of BQTerrence (QP=22)	65
Fig. 30 Pie chart of depth amount ratio of BQsquare (QP=32).....	66
Fig. 31 Pie chart of depth amount ratio of Vidyo1 (QP=32)	67
Fig. 32 Average depth of Vidyo1 (QP=37).....	68
Fig. 33 Average depth of BQTerrence (QP=22)	68
Fig. 34 R-D curve of Basketball in Table 26	73
Fig. 35 Difference between the JM-encoded and our proposed method	79
Fig. 36 Spatial domain: The residual MBs of Inter-16x16 mode on the second frame...79	
Fig. 37 Frequency domain: The transformed and quantized residual MBs of Fig.4.	80
Fig. 38 Flowchart of the combined ME and DCT algorithm	81
Fig. 39 R-D curve of Foreman for P slice	84

LIST OF TABLES

Table 1 Structure of Tools in HM 5.0 Configures [9].....	16
Table 2 Experiment Conditions	20
Table 3 Test Sequences	21
Table 4 Time percentage of “xCompressCU.cpp” in HM5.0	30
Table 5 Comparison of 64/4 CU structure and 16/2 CU structure	31
Table 6 Performance of the basic fast CU decision algorithm	41
Table 7 Specified QP versus N_c	43
Table 8 BD-performance and time reduction ratio of limited N_c	43
Table 9 PSNR and bits measurements at QP=32.....	44
Table 10 Specified QP versus error bound (3%)	48
Table 11 BD-performance and time reduction ratio with 3% error bound	48
Table 12 Simulation result with 3% error bound with 64 frames per sequence	48
Table 13 Simulation result without error bound with 64 frames per sequence	49
Table 14 Comparison of different ratios of error bound.....	50
Table 15 Performance for schemes with and without $2N \times N / N \times 2N$ decision.....	55
Table 16 Performance of the overall proposed algorithm (64 frames/sequence)	60
Table 17 Performance of the overall proposed algorithm (100 frames/sequence)	60
Table 18 Depth percentage (QP is 32)	62
Table 19 Depth percentage of the 10 th frame in low resolution sequences (QP=37)	62
Table 20 Depth percentage of the 10 th frame in HD sequences (QP=37)	62
Table 21 Time reduction ratio analysis of Vidyol and BQsquare	63
Table 22 Depth percentage of BQTerrence (QP=22)	64
Table 23 Time reduction ratio analysis BQTerrence (QP=22)	64
Table 24 Simulation results of ECU and CFM with low delay_P loco setting	70

Table 25 Simulation results of ECU and CFM with our low delay_P loco setting	71
Table 26 Simulation result of ECU, CFM, and our proposed algorithm.....	72
Table 27 Results of the adaptively combined fast algorithm with ECU and CFM	74
Table 28 R-D performance of our proposed algorithm (QP= 22)	75
Table 29 R-D Comparison for FOREMAN in P slices.....	83
Table 30 Modes and Motion Info Bits/Frame	83
Table 31 BD Rate Improvement in P Slices of all Sequences	85
Table 32 Final MV Choice from Candidate MVs (Percentages).....	86
Table 33 Partitioned Sub-Blocks and the Area Ratio Using the Changed MVs.....	86



Chapter 1 Introduction

Video coding plays an important role in the commercial products, and its techniques have been developed during the past 20 years. The matured video compression technique is adopted by many applications, such as television, digital camera, mobile communication, and video recording devices, to store and transmit a large amount of video data. For the better visual quality and the bitrate reduction, the international standard committee is still specifying new standards, and many researchers are still looking for better algorithms. The main stream of video coding in recently years is AVC/H.264. HEVC is the next generation standard that is still in progress.

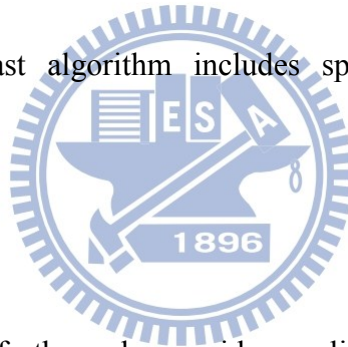


In this thesis, we study both AVC/H.264 and HEVC. In AVC/H.264, we study the transform effect on the motion vector search and design an iterative scheme to improve the overall coding performance. In HEVC, the coding unit (CU) has flexible sizes. In general, the HEVC encoder uses large CU in the stationary or smooth areas particularly at low bitrates. It uses small CUs in the texture areas at the high bitrates. Although HEVC has a better coding performance, it takes a large amount of the complexity to decide the best CU size. Therefore, we want to design a fast algorithm in deciding CU size to reduce calculations.

1.1 Research Contributions

The main contributions of the HEVC part are the development and the analysis of the fast CU decision. Our proposed algorithm achieves up to 49% encoding time reduction, or equivalently, about 2x speed up. On the other hand, the contribution of the AVC part is designing a method to improve compression efficiency by modifying the motion selection process. Our proposed iterative scheme saves up to 4.2% bitrate usage and it retains the video quality. The major contributions in this thesis are listed as below.

1. Develop a fast CU size algorithm for HEVC based on the size information of the neighboring CUs. The fast algorithm includes splitting decision and termination decision.



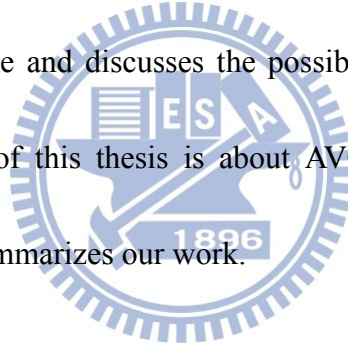
2. Propose additional tools to further enhance video quality or to reduce complexity.
3. Compare and combine our proposed method to the existing fast algorithms in HM5.0.

We investigate their advantages and disadvantages, and find an efficient way to combine them together.

4. Propose a 2-pass ME scheme to identify the best MV for the AVC/H.264 encoder.

1.2 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 gives a brief overview of the state-of-the-art encoders, AVC/H.264 and HEVC. We describe their work flows, their basic operations, and the HEVC advanced coding features. The thesis has two parts: the HEVC part is from Chapter 3 to Chapter 5, and Chapter 6 is the AVC part. In Chapter 3, we describe the CU quadtree structure in HEVC, and introduce the fast algorithms in HM5.0. In Chapter 4, we describe the proposed fast CU size decision algorithm in detail. Then, we design several compensated schemes to improve the original fast algorithm. Chapter 5 presents the simulation results of our scheme and discusses the possible combinations with the existing fast scheme. The second part of this thesis is about AVC/H.264 motion vector search in Chapter 6. Finally, Chapter 7 summarizes our work.



Chapter 2 Overview of H.264/AVC and HEVC

In 1993, the ITU-T Video Coding Experts Group (VCEG) started a long-term project (H.26L). After about ten years of development, the project led to the well-known H.264 standard [1]. The final stage of developing the H.264/MPEG Advanced Video Coding (AVC) standard was carried out by the ITU and ISO/MPEG Joint Video Team (JVT) in 2003. In the past a couple of years, MPEG and VCEG collaborate again to form the Joint Collaborative Team on Video Coding (JCT-VC). With the demand of high-resolution video applications, JCT-VC is currently specifying the next generation video standard, High Efficiency Video Coding (HEVC), which aims to achieve about 50% bit-rate reduction compared to H.264/AVC. And HEVC is expected to be finalized in 2012. For more information about the progress of AVC and HEVC, please refer to [2].

2.1 Advanced Video Coding

Basically, the H.264/AVC standard has a video coding structure similar to that of the prior video coding standards, which is known as the “hybrid coding scheme” [3]. It uses transform coding to code the motion compensated prediction errors. The basic processing unit is macroblock (MB), corresponding to a 16×16 -pixel square region of a frame. In this section, we will introduce the fundamental concept of H.264/AVC. For more details, please

refer to [1], [4].

2.1.1 H.264 Architecture

Fig. 1 shows a typical H.264/AVC encoder. The encoder includes two data paths, an encoding path (left to right) and a reconstruction path (right to left). An input video frame F_n is processed in the unit of MB. A coded MB may belong to an I-MB (intra-coded), P-MB (predictive-coded), and B-MB (bi-directional predictive-coded).

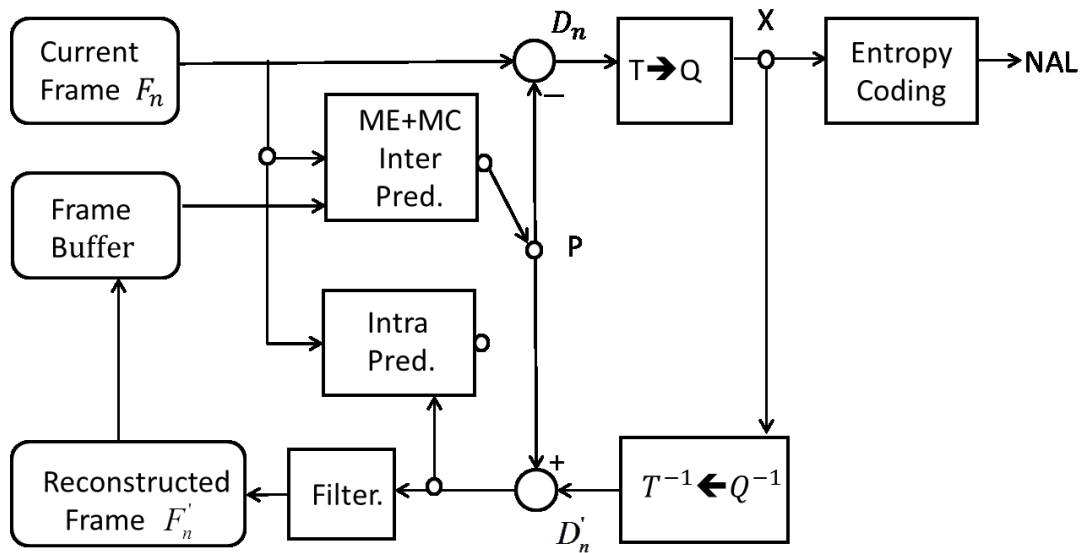


Fig. 1 An H.264/AVC encoder

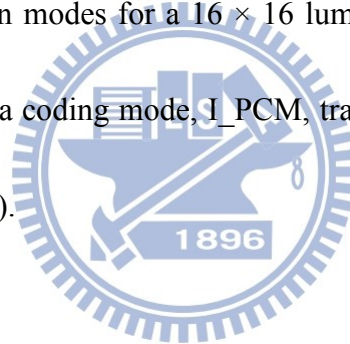
2.1.2 Basic Coding Tools

In Fig. 1, a prediction block P is formed by intra-prediction or inter-prediction. A residual block D_n is produced by subtracting the prediction block P from the current block. The residual block D_n is transformed (separable integer Discrete Cosine Transformation),

and it is quantized to X . The quantized transform coefficients are reordered, and then are entropy-coded. The above coding tools are explained in detail in the following subsections.

2.1.2.1 Intra prediction

Because the correlation between the neighboring blocks within a video frame is extremely high, the encoder, which uses the intra-prediction, can reduce the spatial redundancy. In the intra modes, a prediction block P is generated based on the neighboring blocks (top-left, top, top-right, and left.), which have been encoded and reconstructed. There are four optional intra-prediction modes for a 16×16 luma block, and nine modes for each 4×4 luma block. A special intra coding mode, I_PCM, transmits the image samples directly (without prediction or transform).



2.1.2.2 Inter Prediction

For video sequences at high frame rate, the nearby frames are generally similar. By using the inter-prediction technique to transmit the difference between successive frames, the temporal redundancy could be reduced. The P and B MBs may be coded in one of motion-compensation (MC) modes. Motion compensated prediction based on one or more reference pictures produces the prediction P . An inter-mode MB can be partitioned into various sizes corresponding to the SKIP mode, INTER- 16×16 , INTER- 8×16 , INTER- 16×8 , and INTER- 8×8 modes, and an 8×8 sub-MB mode can be further divided into smaller partitions

with block sizes of 8×4 , 4×8 , 4×4 blocks. Motion estimation (ME) is a key step in inter-prediction. The partitioned block inside an inter-mode MB is predicted from the same size region in the reference pictures. The vector from the current frame block pointing to the best matching region in the referenced frame is the so-called motion vector (MV).

2.1.2.3 Transform and Quantization

Due to the inter-pixel redundancy in the residual block, the encoder transforms the spatial domain pixels to the frequency domain coefficients to compress its original redundant information. The discrete cosine transformation (DCT) is a general tool in the state-of-the-art video encoder. In AVC/H.264, there are two variable size transforms: 4×4 and 8×8 . To increase their computational speeds, they are implemented in the butterfly structure that uses addition, bit-shift, and a few multiplication operations. The DCT coefficients of a residual block should be processed by reordering (zig-zag scanning), scaling, and rounding (quantization). The Quantization parameter (QP) ranges from 0 to 51. With an increment of 6 in QP, the quantization step becomes double.

2.1.2.4 Deblocking Filter

The deblocking filter is designed for eliminating the blocking artifacts on the boundaries, which are caused by the block-based transform with a coarse quantization and by the MC prediction in which the interpolated data are derived from different regions of multiple

reference frames. The filter is applied to each decoded MB to reduce blocking distortion, and the encoder stores the filtered MB in the reconstruction frame to be used as the reference frame in the future. The deblocking filter is an important coding tool for inter-prediction.

2.1.2.5 Entropy Coding

At the slice layer level and below, the syntax elements are encoded either by the variable length coding tool (VLC) or by the context-adaptive arithmetic coding tool (CABAC). In VLC, a quantized DCT block is coded by using the context-adaptive variable length coding (CAVLC) scheme, and the other data units are coded by using Exp-Golomb codes. The tables of CAVLC are designed to match the corresponding conditional probability. The context adaptive feature of CABAC can be more efficient because it is adaptive to the statistics of previously encoded data. Generally, CAVLC has low complexity, and CABAC has better efficiency.

2.1.3 Encoder Control

The H.264/AVC standard provides only the syntax of bit-stream and the decoder structure. Therefore, we need to design and to control the encoding process in our preferred way. How to decide the coding parameters is a key to achieve video compression efficiency. The H.264 coding parameters include MVs, quantization levels, and MB modes. The same encoder structure with different coding parameters will affect the R-D efficiency of the produced bit-stream.

The general R-D cost function for video coding is presented by (1). In (1), symbol D denotes distortion, which is often the absolute difference between the processed image block and the original block. Symbol R means rate, which is the bits needed to send the processed information. According to the information theory, we can fix R first and then minimize D . We can combine D and R together to form the total cost J . Mathematically, we can convert this constrained optimization problem to a non-constrained form, the so-called Lagrange cost function in (1). How to select the optimal Lagrange multiplier λ is a difficult problem in practice, and for more details, please refer to [5], [6].

$$J = D + \lambda R \tag{1}$$

A traditional H.264/AVC encoder splits the optimization of the cost function for the inter modes into two parts as illustrated in Fig. 2. The first part is finding the optimal MV, and second part is choosing the best mode, block size etc.

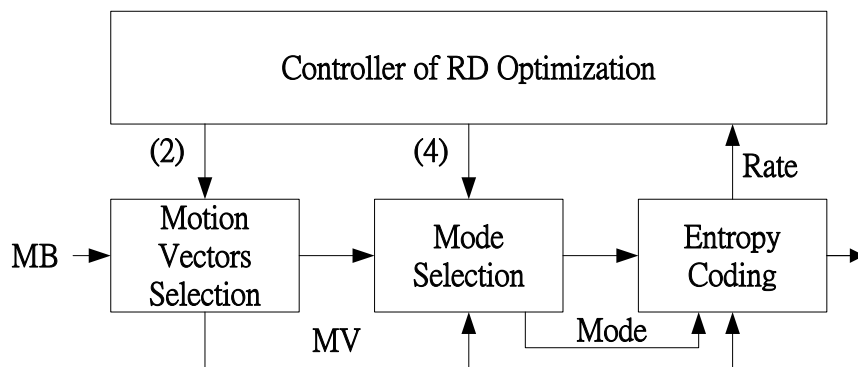


Fig. 2 R-D optimization for selecting MV and mode

2.1.3.1 Searching for Optimal Motion Vector

A traditional H.264/AVC encoder splits the optimization of the cost function for the inter modes into two parts. In the first part, the encoder finds the MVs with the optimum residual distortion and the MV coding bits. Based on the motion R-D cost function (2), [3], the motion estimation step finds the vector with the smallest cost for various block sizes. Given the current and the reference frames and the Lagrange multiplier λ_{motion} , the ME operation for a partition block s_i is to minimize (2) to find the best MV.

$$J_{motion} = D_{motion}(s_i, m) + \lambda_{motion} R_{motion}(s_i, m), \quad (2)$$

where m is the set consists of all possible vectors (m_x, m_y, m_t) , in which m_x is the MV horizontal component, and m_y is the vertical component, m_t is time difference. R_{motion} is the number of bits for transmitting MV, and D_{motion} is the distortion term given by

$$D_{motion}(s_i, m) = \sum_{(x,y) \in s_i} |pixel(x, y, t) - pixel(x - m_x, y - m_y, t - m_t)|^p \quad (3)$$

To speed-up the ME process, we usually choose $p=1$, and (3) becomes the sum of the absolute difference (SAD). The symbols, x and y , are the pixel location in a block. It should be noted that the state-of-the-art encoder often uses hadamard measure for fractional ME for coding efficiency, and the detail is describe in section 6.1.

2.1.3.2 Selection for the Best Mode

In the second part of the inter-coding process, the encoder applies integer DCT to the

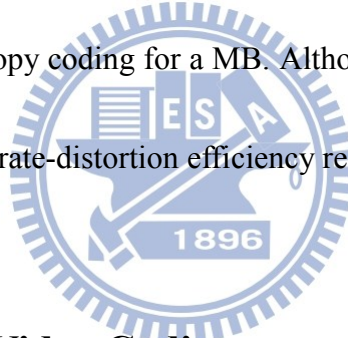
motion-compensated residual error signals, and then we choose the best MB coding mode.

With the given Lagrange parameter λ_{mode} and the quantized parameter Q , the coding mode of MB (S) is decided by minimizing the following R-D cost function [3],

$$J_{mode}((S, I_k) | Q, \lambda_{mode}) = D_{REC}(S, I_k | Q) + \lambda_{mode} R_{REC}(S, I_k | Q), \quad (4)$$

where I_k represents a legitimate mode. For example, k possible modes for P-slice in H.264/AVC are Intra-16×16, Intra 4×4, SKIP mode, INTER-16×16, INTER-8×16, INTER-16×8, INTER- 8×8 modes. D_{REC} is the distortion between the reconstructed MB and the original one, and it is usually measured in the sum of the squared difference (SSD), $p=2$ in (3).

R_{REC} denotes the rate after entropy coding for a MB. Although the calculated cost function is an approximation, it reflects the rate-distortion efficiency reliably.



2.2 High Efficiency Video Coding

A joint call-for-Proposal (CfP) for HEVC was issued by JCT-VC in January 2010, and 27 proposals in response of the CfP were submitted with their test material. The promising results were reported in [7], and the proposed scheme [8] from Heinrich Hertz Institute (HHI) was ranked among the five best performing proposals. For its wonderful performance, most of its design elements were selected to specify a first model of the initialed HEVC standardization project. The project is still in progress, and HEVC is expected to achieve excellent coding performance on high resolution video with low delay and low complexity.

Fig. 3 shows the HEVC encoder structure. Although HEVC has a similar structure to the H.264/AVC architecture, there are some significant innovations in HEVC. The innovations of re-definition of coding units and the enhancement on coding tools offer remarkable compression efficiency.

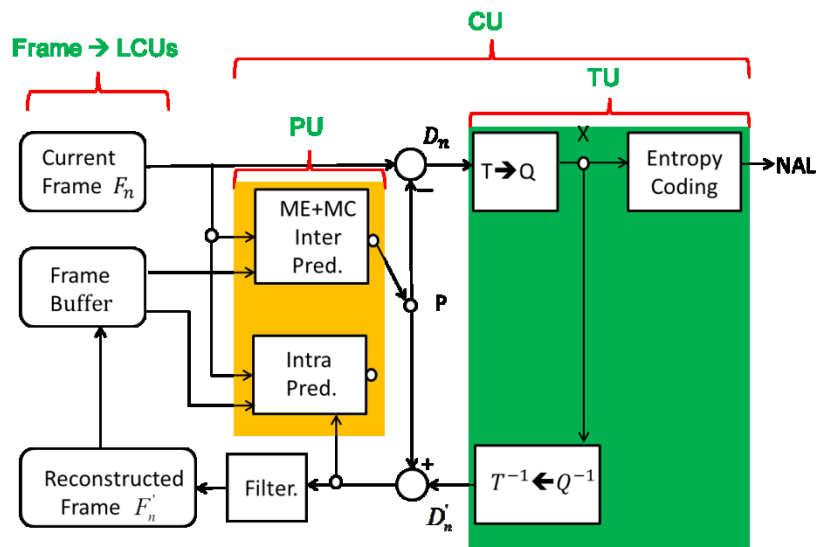


Fig. 3 An HEVC encoder

2.2.1 Coding Unit Definition

In H.264/AVC, the basic processing unit is called MB, which is expanded to what we called a coding tree block (CTB). For flexibility and efficiency, the basic coding units in HEVC have variable sizes with various resolutions. They are CU (Coding Units), PU (Prediction Units), and TU (Transform Units). A CTB in HEVC which covers $2N_{\max} \times 2N_{\max}$ luma samples, and its associated quadtree structure indicates how the CTB are further

subdivided for CUs, corresponding PUs and TUs. The concept of decomposing MB into three different units allows each to be optimized independently, which brings high adaption to enhance the performance of each coding tools. The definition and details of three units in the HEVC encoder [9] are given in the following sub sections.

2.2.1.1 Coding Unit

A basic unit of HEVC, referred as CU, is a square region of a picture, and it may contain several PUs and TUs. An input processing frame is divided into slices, and each slice is composed of CTBs, which are also called largest coding units (LCUs). Dividing a picture into LCUs and further recursively subdividing each CUs into 4 smaller CUs with half width and half height is the so called nested quadtree structure as shown in Fig. 4 (with solid lines). Both the block sizes and the block coding parameters such as maximum allowed depth will be specified in the sequence parameter set (SPS) or the slice header.

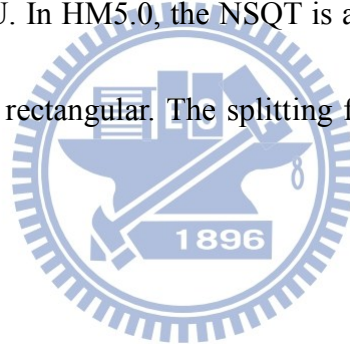
2.2.1.2 Prediction Unit

PU is defined only for the leaf node of CU in each depth level, and PUs have various partitions for prediction. They are confined within its CU node with a shape of square or rectangular, and for some cases the prediction units are asymmetric in CU as list in Table 1. The prediction ways are similar to the prediction methods of H.264/AVC, which can be the skip, the intra, or the inter modes. In Fig. 5, we can see all the possible PUs for each

prediction mode in low complexity setting. The information related prediction such as the PU splitting types, the prediction modes, the intra prediction direction, the motion vector difference (MVD), and the corresponding referenced frame indices are transmitted in PU level.

2.2.1.3 Transform Unit

TU is a basic unit of residual coding, including transform and quantization. The TUs are aligned within their corresponding CU, and the size of TUs is variable which is not constrained by boundaries of PU. In HM5.0, the NSQT is added, that is, the shape of TU has not to be square, and it may be rectangular. The splitting flag and transform coefficients are specified in TU level.



The tree structure of CU or TU splits from top to down, but the optimal structure is decided by G-BFOS algorithm [10], [11]. The algorithm makes pruning decision from bottom to up, which reduces much computational complexity, and we will describe the detail part in the next chapter. The coding tree blocks for TU are illustrated by Fig. 4 (with dashed line). More details of the encoder controller for HEVC are described in chapter 3. An Example of a nested quadtree structure (right part) for dividing a given coding tree block (left part) in Fig. 4. The order of parsing the coding blocks follows their labeling in alphabetical order.

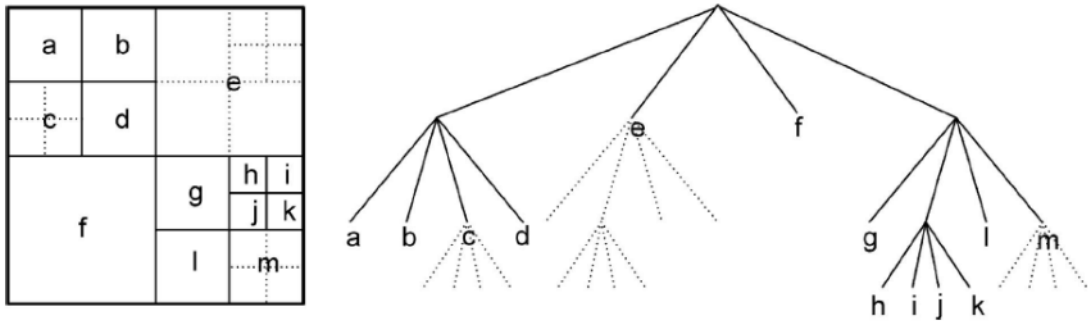


Fig. 4 An Example of a nested quadtree structure [8]

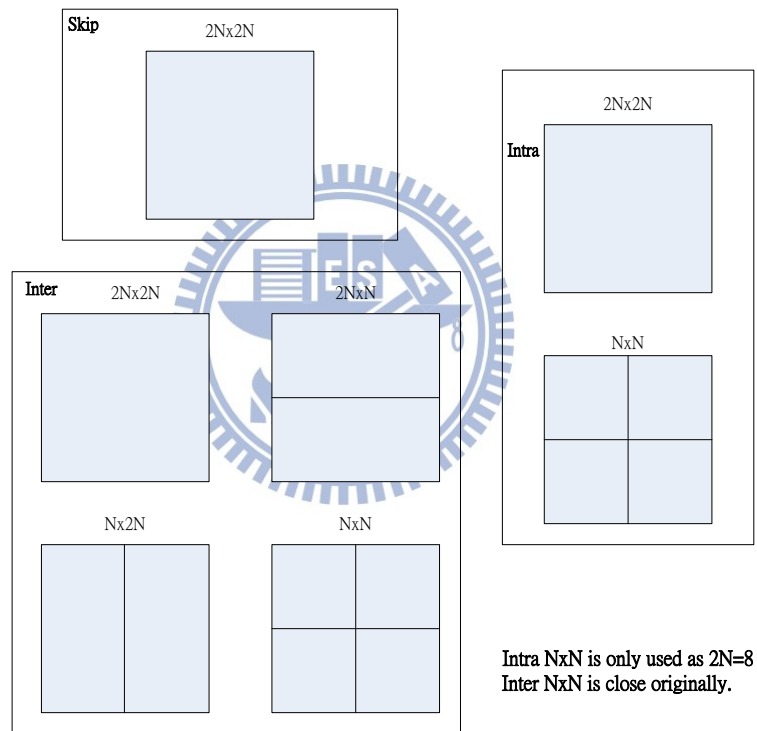


Fig. 5 Possible PUs in low complexity setting

2.2.2 Enhanced Coding Tools

After H.264/AVC standard was defined, people tried to propose algorithm to improve it. As time goes by, people notice that some modifications on the existing tools and many newly proposed tools provide a certain amount of improvement. Therefore, many adaptive and novel

tools are adopted in the current HEVC model compared to H.264/AVC. With the development of HEVC standardization project, JCT-VC adds useful tools, refines the existing tools, and removes inferior tools in the model [12]. A summary list of the tools that are included in HM5.0 is provided in Table 1 below.

Table 1 Structure of Tools in HM 5.0 Configures [9]

High Efficiency Configuration	Low Complexity Configuration
<i>Coding units, Prediction units, and Transform units:</i>	
Coding unit quadtree structure (square coding unit block sizes $2N \times 2N$, for $N=4, 8, 16, 32$; i.e., up to 64×64 luma samples in size)	
Prediction units (for coding unit size $2N \times 2N$: (1) for Inter, $2N \times 2N$, $2N \times N$, $N \times 2N$, and, for $N > 4$, also $2N \times (N/2 + 3N/2)$ & $(N/2 + 3N/2) \times 2N$; (2) for Intra, only $2N \times 2N$ and, for $N=4$, also $N \times N$)	Prediction units (for coding unit size $2N \times 2N$: (1) for Inter, $2N \times 2N$, $2N \times N$, $N \times 2N$; (2) for Intra, only $2N \times 2N$ and, for $N=4$, also $N \times N$)
Transform unit tree structure within coding unit (maximum of 3 levels)	
Transform block size of 4×4 to 32×32 samples (always square for Intra; also non-square 4×16 , 16×4 , 8×32 , 32×8 for Inter)	Transform block size of 4×4 to 32×32 samples (always square)
<i>Spatial Signal Transformation and PCM Representation:</i>	
DCT-like integer block transform; for Intra also a DST-based integer block transform (selected based on the intra prediction mode)	
Transforms can cross prediction unit boundaries for Inter; not for Intra	
PCM coding with worst-case bit usage limit	
<i>Intra-picture Prediction:</i>	
Angular intra prediction (17 directions for 4×4 , 3 directions for 64×64 , 34 directions for others)	
Planar intra prediction	

Chroma intra prediction separate from or using luma samples	
<i>Inter-picture Prediction:</i>	
Luma motion compensation interpolation: 1/4 sample precision, 8x8 separable with 6 bit tap values	
Chroma motion compensation interpolation: 1/8 sample precision, 4x4 separable with 6 bit tap values	
Advanced motion vector prediction with motion vector “competition” and “merging”	
<i>Entropy Coding:</i>	
Context adaptive binary arithmetic entropy coding	
RDOQ on	RDOQ off
<i>Picture Storage and Output Precision:</i>	
8 bit-per-sample storage and output	
<i>In-Loop Filtering:</i>	
Deblocking filter	
Sample-adaptive offset filter	-
Adaptive loop filter	-

2.2.2.1 Intra prediction

Comparing to H.264/AVC, the unified intra prediction coding tool provides extensive prediction modes up to 35 directional prediction modes including DC and Planar modes for luma component of each PU. The total number of available prediction modes depends on the size of the corresponding PU.

2.2.2.2 Inter Prediction

Each inter coded PU have a set of motion parameters consisting of motion vector, reference picture index, etc. Choosing the optimal motion parameters is crucial to the

performance of inter mode. The Advanced motion vector prediction (AMVP) is an adaptive prediction technique for motion merging. AMVP constructs the motion vector candidate list from the co-related PUs, which exploits spatial and temporal correlation. Then, remove duplicated and redundant the candidates. At the last, the encoder selects the best inferred motion parameters from multiple candidates formed by spatial neighboring PUs and temporally neighboring PUs, and it transmits the corresponding chosen candidate index. Also, merging mode plays an important role in inter prediction because it can reduce the transmitted motion information. Thanks to AMVP and merge mode, the compressed motion data often consist of a small amount of side information.

2.2.2.3 Transform and Quantization

HEVC provides larger size transforms compared to H.264/AVC, and the size of transform covers from 4×4 to 32×32 . With larger sizes transformation, the encoder is more flexible and the compression efficiency is higher in the smooth texture region especially. The scaling matrices of the quantization process are added for the additional transform sizes, which do not included in H.264/AVC.


2.2.2.4 Loop Filter

Loop filter consists of deblocking filter, sample adaptive offset (SAO), and adaptive loop filter (ALF). The goal of these filters is improving the quality of the reconstruction frames. A

deblocking filter is performed for the block boundaries. Then, SAO is applied to the reconstruction signal after the deblocking filter by using the offset values given. In the final stage of filtering, an ALF is applied to the reconstruction signal after the SAO process and deblocking filter process by using the filter coefficients also signaled in the slice header. It should be noted that ALF scheme and its control method change a lot in the later version HM.

2.2.2.5 Entropy Coding

In HM 5.0, the syntax elements are encoded by variable length coding (VLC), and the residual coefficients are encoded by CABAC. Because the complexity of CABAC is very high, it results in low data throughput when handling high resolution videos. This problem has been improved by the parallel entropy coder design. For pursuing high efficiency, the HEVC specifications retain CABAC, but remove CAVLC.



2.3 Experiment Conditions

Our experimental platforms and their configuration settings are introduced in this section. The referenced software of H.264/AVC is JM 18.0 [13], and it has four configures, which are baseline, main, extended, and high profile. We utilize the baseline configure setting to simulate our experiments with the widely used MPEG sequences [14]. Our platform for HEVC experiments is the referenced software HM5.0 [15], in which 4 configures are defined. They are all intra, low delay, low delay P, and random access. These configurations can be set

as the high efficiency or low complexity coding modes. We choose the low delay P, low complexity configuration as our experimental conditions. The experimental sequences are the testing materials of HEVC standard. To compare performance between the proposed algorithm and the original codec, we exploit the BD-rate [16] definition to measure the compression efficiency. Table 2 shows our parameters setting through this thesis, and Table 3 lists the information about size and frame rate of all video sequences in this thesis.

Table 2 Experiment Conditions

QP	22,27,32,37
AVC Encoder Configuration : baseline	Sequence Type : IPPP Motion Search : Fast full search Motion Search range : ± 32 pixels Multiple Referenced frame : Disable RDO : High complexity Fractional ME : Hadamard measure Transform Size: 4×4 Intra period : 16 Number of encoded frames : 32
HEVC Encoder Configuration : low delay P, low complexity	Sequence Type : IPPP. Motion Search range : ± 64 pixels Multiple Referenced frame : Disable GOP : 1 Intra period : Only first Max CU size : 64 Max CU partition Depth : 4 Max TU size : 32×32 Min TU size : 4×4 Inter Max RQT depth : 3 Intra Max RQT depth : 3 RDOQ : Disable

	DisableInter4x4 : On FEN: On Number of encoded frames : 16,32,64,100
--	--

Table 3 Test Sequences

HEVC sequences			
Sequence	Information	Sequence	Information
Kimono	1920x1080 24Hz	BallDrill	832x420 50Hz
Park	1920x1080 24Hz	BQMall	832x420 60Hz
Cactus	1920x1080 50Hz	Party	832x420 50Hz
Basketball	1920x1080 50Hz	HorseC	832x420 30Hz
BQTerrace	1920x1080 60Hz	BallPass	416x240 50Hz
Vidyo1	1280x720 60Hz	Bubbles	416x240 60Hz
Vidyo3	1280x720 60Hz	BQsquare	416x240 50Hz
Vidyo4	1280x720 60Hz	Horses	416x240 30Hz
H.264/AVC sequences			
Foreman	352x288 30Hz	Silent	352x288 30Hz
Bus	352x288 30Hz	Ice	352x288 30Hz
Football	352x288 30Hz	City	704x576 30Hz
Mobile	352x288 30Hz	Crew	704x576 30Hz
News	352x288 30Hz	Harbour	704x576 30Hz
Paris	352x288 30Hz	Soccer	704x576 30Hz
Mother_daughter	352x288 30Hz	Download link [14]	

Chapter 3 Nested Quadtree Coding Unit

In this chapter, we introduce the principle and decision flow of quadtree Coding Unit (CU) decision in HM5.0. This coding unit structure differs from the macroblock coding architecture in H.264 for flexible and compression efficiency. However, the CU quadtree structure with possible node sizes from 64×64 to 8×8 in 4 admissible depths also brings high computation complexity. Although HM 5.0 has some fast algorithms to accelerate the encoding procedure, we still want to reduce more complexity under the tolerable coding loss.

3.1 Overview of Coding Unit Quadtree Structure

CU is a $2N \times 2N$ square and $2N$ can be 64, 32, 16, or 8. The encoder processes LCUs in a frame in the sequential order from the left to the right, and then from top to down (raster scan). Fig. 6 illustrates a real example of the partitioned nested CU quadtree structure.

Larger CU provides less bits usage in the smooth residual texture and the static motion area in an encoded frame compared to the maximum 16×16 macroblock coding structure in H.264. The HEVC encoder can also has the same small size CU as that in H.264 to handle the areas with fast motion and complex residual texture. Targeting at high spatial resolution picture for HEVC, the CU quadtree structure is especially designed for 720P and 1080P video.

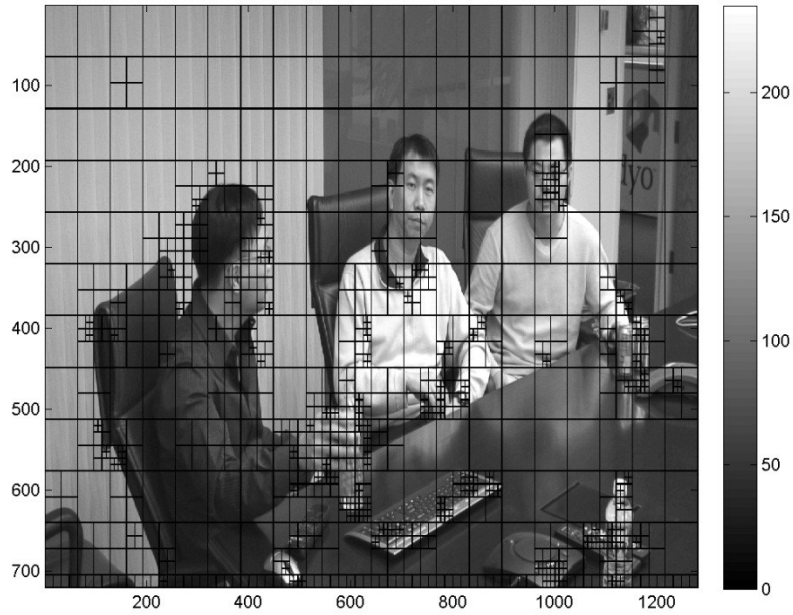


Fig. 6 An example of nested CU quadtree structure (Vidyo1, Frame 2, QP=32)

3.1.1 Partition Decision Flow of Nested Quadtree CU

In HEVC, a slice is composed of many LCUs, and a large CU can be divided into four smaller CUs. Each partitioned CU can be recursively split until the smallest size CU is reached, in which 4 depths are allowed in HM 5.0. As one $2N \times 2N$ (not 8×8) CU is processed in each depth, the encoder will analyze the R-D cost of all possible prediction modes. First, the skip mode is used for compression, and then try Inter $2N \times 2N$, $N \times 2N$, $2N \times N$ (If in the high efficiency setting, the encoder will try additional asymmetric PUs.). Last, Intra $2N \times 2N$ is tried for prediction. It should be noted that I_PCM is turned off in HM5.0 in every profile. The smallest CU (8×8) is additionally tested with $N \times N$ PUs for intra mode, but the asymmetric is not included in this depth. When the best prediction of each

mode produces the residual signal, the encoder processes it in the units of TU. The size of TU is limited to that of the CU to which the TU belongs. TU in the CU with size $2N \times 2N$ can be split into $N \times N$ and $N/2 \times N/2$ in a similar way to the CU recursively partition. However, as already stated in Table 1, the maximum TU size cannot exceed 32×32 , and the NSQT is used in some cases for inter residual signals.

At the same depth of CU, after analyzing each mode, its RD cost is compared with that of the other previously processed modes to determine the best mode for the CU in this depth. However, we still need an efficient method to compare the R-D cost of the best partitioned modes at different depths. For example, allowing three admissible depths in the CU quadtree has sizes varying from 64×64 to 16×16 . The number of the possible tree structures is 17. The exhaustive comparison is not practical if the depth becomes larger.

To reduce the redundant comparisons, G-BFOS algorithm follows the well-known “divide and conquer” concept. At the beginning, a full tree grows from the root to all possible nodes until reaching the maximum admissible depth in the way of depth first and in the Z-order ($C \rightarrow D \rightarrow E \rightarrow F$) of the same depth as shown in Fig. 7. When all nodes in one branch are constructed, a pruning decision process compares the cost of the parent and that of its children nodes to decide that the splitting process is needed or not. If (5) is satisfied, the children nodes would be pruned. Otherwise, the sum of costs of all children nodes is assigned to the parents’ node for the following comparison.

$$J(\text{parent node}) \leq \sum_{i=1}^4 J(\text{children node}) \quad (5)$$

When all the compared nodes are built up, the decision process is executed until the root node is reached. Using G-BFOS algorithm ensures that we can get the local minimum cost in each partition region, and then combine them to find the best nested CU tree structure for a LCU with the global minimum cost. Through this efficient decision algorithm, we only need 5 comparisons to decide the best CU partitioned structure in the example of Fig. 7.

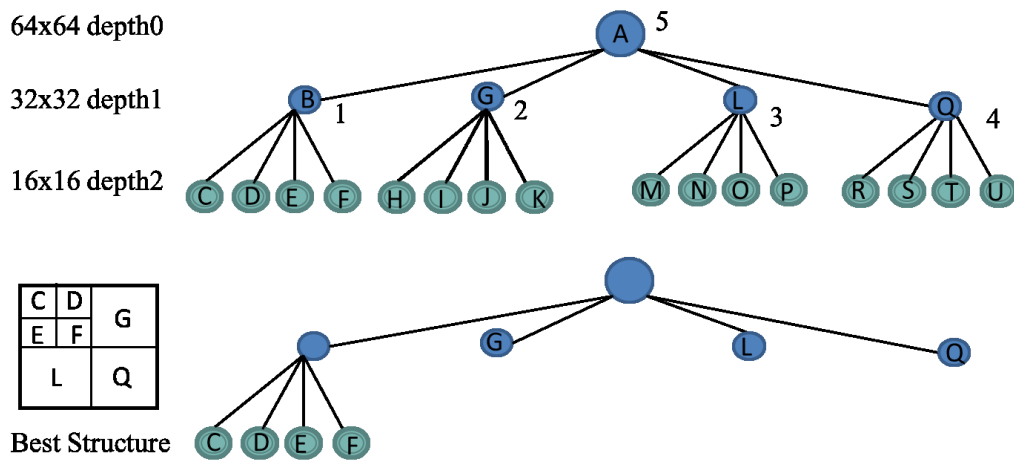


Fig. 7 A G-BFOS example.

The alpha-order is the CU processing order (depth first and Z order at the same depth), and the numerical-order is the pruning decision order.

3.1.2 Existing fast algorithms for Partition Decision Flow in HM5.0

Because of the huge complexity associated with the quadtree structure, many researchers like to reduce its complexity. G-BFOS is the good solution for quadtree structure decision. Thus, the targets of researchers are often chosen to be the efficient methods to build nodes.

There are 3 existing schemes in the literature, namely, fast encoder setting (FEN) [9] [17], early CU termination (ECU) [18], and cbf fast mode decision (CFM) [19].

There are 3 parts in FEN [9]. The first part is the CU early skip method, the second is the sub-sampled SAD calculation, and the third is the simplified bi-prediction. We describe first part in detail because it relates to the CU tree structure. The CU early skip method in FEN is based on the average rate-distortion cost statistic in each slice. That is, when the R-D cost of the current CU with skip mode in the current depth is smaller than the average cost of previously encoded CUs with skip mode which is chosen as the optimal mode in the same depth, the rest of PU modes in this depth are skipped. For a more aggressive decision, the average R-D cost is multiplied by a fix-weighting factor of 1.5, and some research people reports that an adaptive weighting factor can improve the performance of FEN [17]. The performance of FEN is about 2.0% luma BD bit-rate loss and 48% overall encoding time saving in the setting of high efficiency random access in HM3.2. Because FEN has multiple considerations for speeding up HEVC, all configurations of HM5.0 turn on FEN in the original settings.

ECU is a fast CU decision method using early termination based on the optimal PU mode which was proposed by Choi et al [18], and the algorithm is also designed for skip

modes for CU quadtree pruning. From their analysis of condition probability of the CU depth selection, they observe that if the current CU selects the skip mode as the best prediction in the current depth, 95% of this type CU will finally be encoded with the skip modes at this depth. Exploiting this property, the CU depth check is skipped for all the next sub-CUs when the R-D cost of the skip mode is minimum in the current CU. ECU algorithm has been adopted in HM4.0, and it yields approximately 42% time reduction in encoding time with negligible loss on the luma BD-rate in HM3.1 (i.e., $< 0.6\%$).

Except for the acceleration of FEN, every PU is processed to measure its R-D cost in one CU regardless of the performance of the previous PUs. The R-D costs for all allowed PUs in each depth are examined to ensure the optimal prediction, but the exhaustive method wastes a lot of time. The coded block flag (cbf) is a good indicator to estimate the benefit of using prediction. After the prediction operation of a PU, its corresponding CU becomes a residual quadtree (RQT) block, which is to be processed as the TU. After the RQT is transformed and quantized with a suitable tree structure, if all coefficients in this residual block are zero, the cbf is set to 0, which means the prediction is sufficient (no residual coefficients coding). Otherwise, cbf is 1. Gweon et al [19] proposed a CFM algorithm that uses this cbf property, and the computational complexity is reduced to about 58.8% with the luma BD-rate loss 0.85% in HM3.2. The core idea of CFM is checking that three cbf values (1 luma and 2

chromas) for every PU partition. If all of them are zero, then the processing of the PU options of the current depth are terminated. It should be noted that the encoder simply skips the analysis of PUs at this depth when the termination condition of CFM is satisfied, but it still has to process PUs of all the sub-CUs in deeper depths.

ECU and CFM are powerful tools for reducing complexity, but they are closed in the original settings of all configurations in HM5.0. An example of ECU is illustrated in Fig. 8, and the program flow of CFM with the execution order of PUs in the low-complexity profile is shown as Fig. 9 and Fig. 10 respectively.

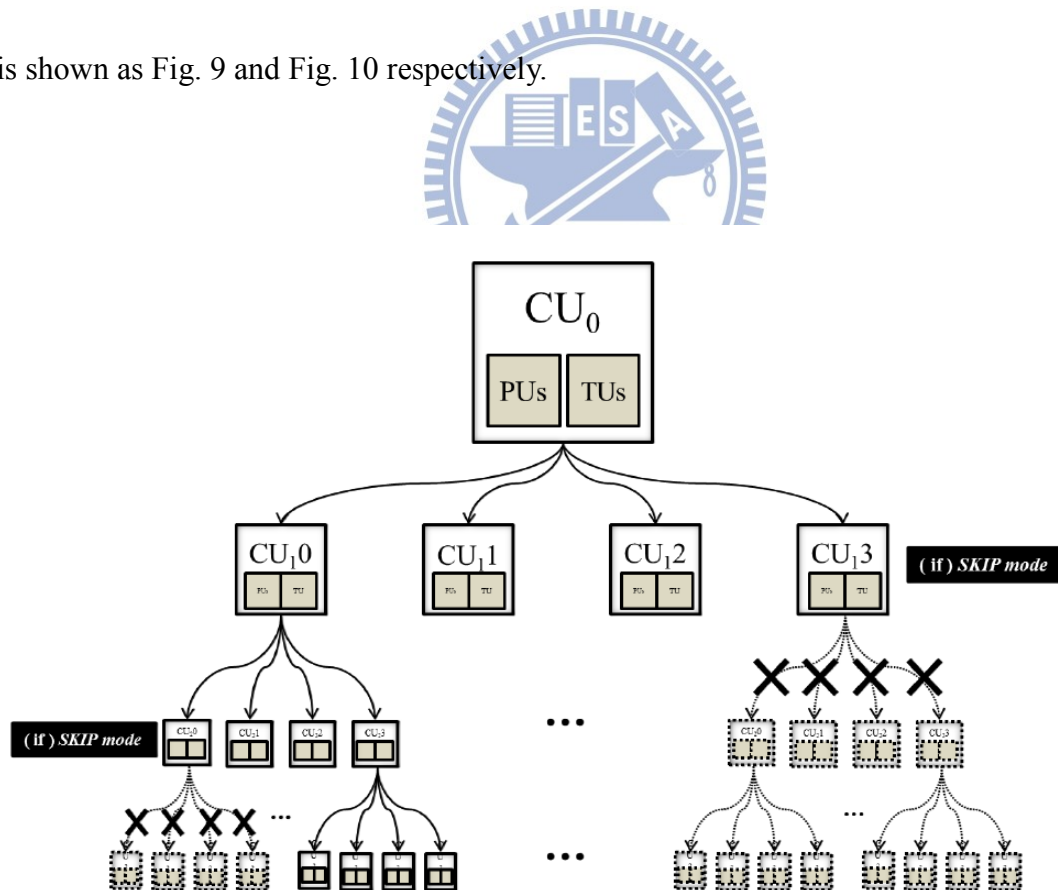


Fig. 8 An example of ECU [18]

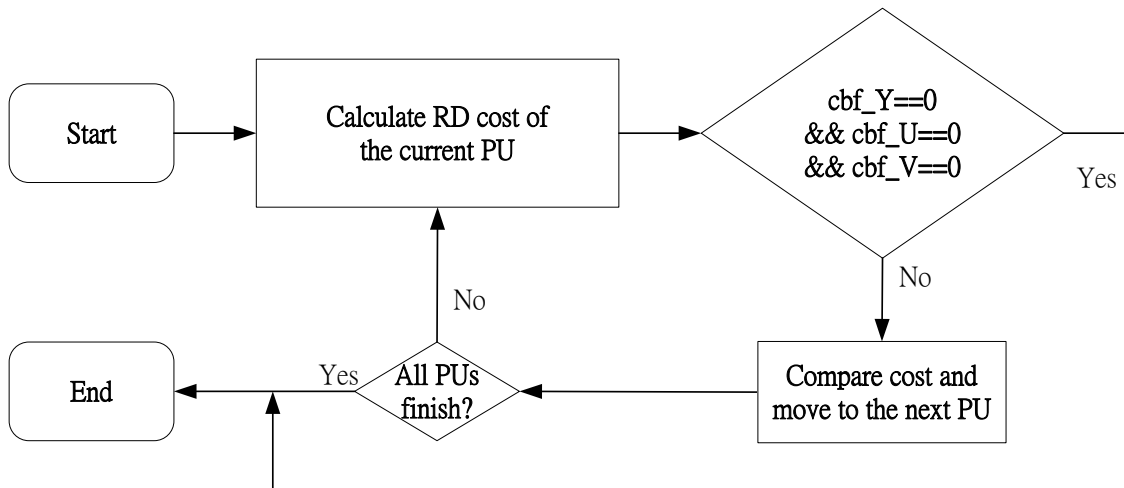


Fig. 9 Program flow of CFM

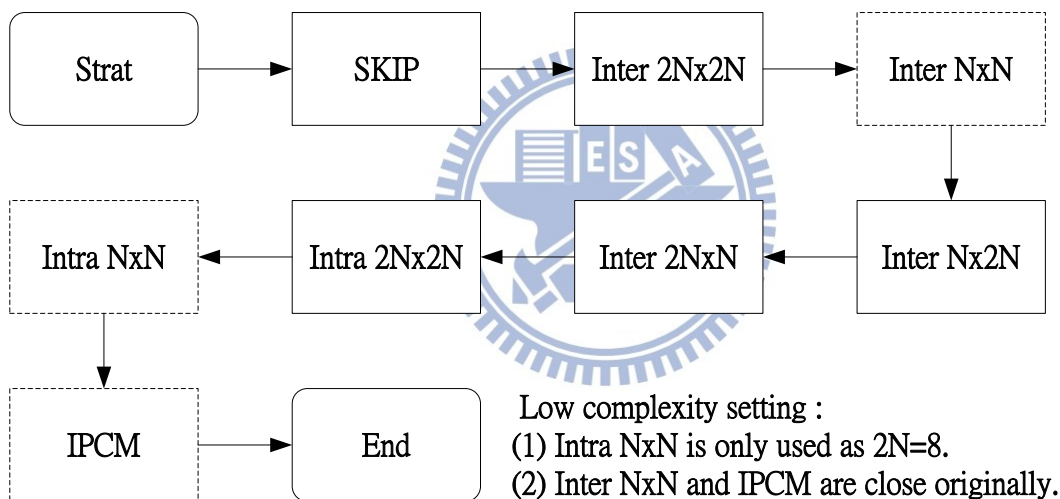


Fig. 10 PU execution order in CU in the low-complexity setting

3.2 Analysis of Nested CU Quadtree Structure

The nested CU quadtree Structure decision process in HM5.0, which pursues the optimal structure selection, is described earlier in section 3.1.1. Although there exist FEN, ECU, CFM, and G-BFOS algorithms to reduce the encoding complexity, we like to further speed up the CU quadtree processing. Thus, we need to examine that which part in HM 5.0 takes most time

and find out what factors producing the complexity.

The HEVC encoder computes the R-D cost to select the best CU size, PU partition, and TU depth. The encoder spends a huge amount of computations on PUs and RQT in a CU quadtree to identify the lowest R-D cost. We measure the computing time of the function named “xCompressCU.cpp”, which is used for CU decision in HM5.0. In Table 4, we collect the execution time ratio of “xCompressCU.cpp” regarding the overall encoding time in 8 high-resolution test sequences for 16 frames, and the average time ratio is taken over 4 selected QP cases.

Table 4 Time percentage of “xCompressCU.cpp” in HM5.0

Test Sequence	Time Percentage	Test Sequence	Time Percentage
Kimono(1080P)	99.6%	Vidyo1(720P)	99.4%
Park(1080P)	99.5%	Vidyo3(720P)	99.5%
Cactus(1080P)	99.5%	Vidyo4(720P)	99.5%
BasketballDrive(1080P)	99.5%	BQTerrace(1080P)	99.5%
AVG	99.5%		

Table 4 shows a surprising result that CU decision takes more than 99% time in the low delay P with low complexity configuration. The computation associated with CU decision includes inter prediction, intra prediction, RQT, and calculate R-D cost, and we know that the computing time grows up rapidly with the increment of maximum admissible depth. Different maximum admission depth results in different compression efficiency and computational

complexity. In Table 5, we try the original block size setting of H.264/AVC with the maximum CU size equals 16 and the maximum admissible depth is 2 compared to the original setting in HEVC; that is, the encoder only uses 16×16 and 8×8 CUs to compress the video sequences with same testing condition as Table 4.

Table 5 Comparison of 64/4 CU structure and 16/2 CU structure
(Maximum CU size / Maximum admissible depth)

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving (QP22)	-40.88%	-43.75%	-43.67%	-42.95%	-43.91%	-45.39%	-44.92%	-44.40%	-43.73%
Time-Saving (QP27)	-42.30%	-46.01%	-45.00%	-44.56%	-42.99%	-45.57%	-43.79%	-45.22%	-44.43%
Time-Saving (QP32)	-44.10%	-45.33%	-44.39%	-45.46%	-43.38%	-44.47%	-44.04%	-44.40%	-44.45%
Time-Saving (QP37)	-44.92%	-45.08%	-44.67%	-46.15%	-43.77%	-43.99%	-45.34%	-43.14%	-44.63%
AVG. Time-Saving	-43.05%	-45.04%	-44.43%	-44.78%	-43.51%	-44.86%	-44.52%	-44.29%	-44.31%
Y BD-rate (%)	5.544	2.576	3.283	9.909	3.037	7.359	7.263	8.257	5.904
Y BD-PSNR (dB)	-0.150	-0.078	-0.071	-0.167	-0.094	-0.207	-0.197	-0.193	-0.145

As depicted in Table 5, the 16/2 CU structure saves about 44% overall encoding time, but causes 5.9% luma BD-rate loss in average. In general, it is a trade-off issue between computational complexity and coding performance in designing a fast algorithm. Nevertheless, such a large loss from 16/2 CU structure is generally not considered cost-effective, so we are looking for other methods to accelerate the process of CU decision.

Chapter 4 Fast CU Size Decision Algorithm Design

In the following section, we first describe the problem and the target we want to achieve, and then we survey some ideas about fast CU quadtree decision algorithms, [20] and [21], published recently but not have been accepted in HM as the coding tools in our testing platform. After implementing the original platform, we measure and analyze its performance with many standard sequences, and propose some ideas referred from [21] to compensate the weakness of the testing platform.

4.1 Problem Formulation and Design Goal

Because HM5.0 has FEN, ECU, and CFM for CU fast algorithm, we try to design additional fast algorithms from different perspectives. The principle of our new tool should be different with those three existing tools, and the added tool should not reduce the performance of the existing and also be compatible with the CU quadtree structure in HM5.0.

For the above reasons and the simulation results in section 3.2, skipping the analysis of coding units in unnecessary depth is a possible way to accelerate encoding procedure, especially for the high resolution video. Typical fast algorithm performance or experimental results are examined by the ratio of time reduction, the bitrate and PSNR with the specified QP and R-D curve [20], [21]. Therefore, we set up a reasonable target of our final proposed

algorithm that reduces about 50% complexity and minimize the coding loss. Moreover, the collaborative effect between our proposed algorithm and the existent fast algorithms is also an important issue.

4.2 Related Work

Even though the original encoding procedure returns the best possible tree structure, its complexity is very high. Heuristics scene characteristics estimation is necessary to predict the optimal depth for the next encoded CU. In [20], the main idea is to accelerate the encoding procedure of HEVC by utilizing the correlation of related CUs. The encoder uses the size information of neighboring coding units and the processed depth-ratio in the previous frame to limit possible processed depth. In [21], a complexity-control method is proposed, which performs the time analysis and adjusts the number of fast encoding frames of each picture group. Recording the deepest depth used in the unit of LCU in the previous frame, the encoder finds the best possible tree structure until the recorded depth in each LCU in the current frame.

However, the methods, [20] and [21], are implemented in the earlier version HM, so we need to convert their ideas to fit our experimental platform HM5.0. Due to the above reasons and performance consideration, we remove the frame level algorithm in [20], and the time analysis in [21] is not suitable for our research because different computers would execute the

same program with different time, so we use QP value as the indicator to adjust our algorithm.

The details will be described in the following sections.

4.3 Core Ideas of Fast CU Size Decision

The CU-level fast decision is based on the fact that the in the temporal and spatial neighborhoods, the motion and texture characteristics of a picture patch are similar. Therefore, we can predict the candidate CU depth by checking the size of its neighbor CUs (spatial) and co-located CU (temporal).

The data structure for HEVC is that each LCU includes 21 bits for representing the splitting information as illustrated in Fig. 11. The accuracy of the data structure extends to depth 2 which is sufficient for our fast decision. For example, during the encoding procedure, G-BFOS tells us that splitting the LCU into 4 sub-CUs is better due to its lower R-D cost. Then, the encoder will record the bit of index 0 in Fig.11 as 1 to indicate the splitting. Otherwise, the bit is set to 0.

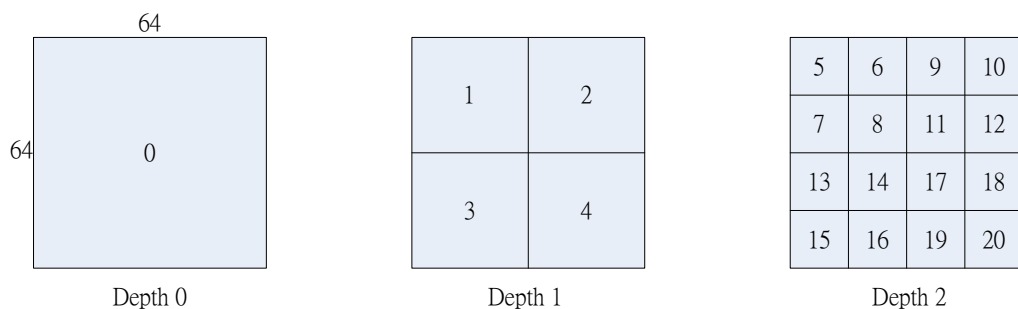


Fig. 11 Data representation of splitting information

The other important factor in our algorithm is the location of corresponding CUs. Fig. 12 depicts the relation between the referred neighboring CUs and the current encoded CU. The co-located CU means that the previous frame CU has the same position as the current encoded CU. It should be mentioned that our algorithm executes recursively in depth 0, depth 1, and depth 2 with the corresponding CU size of $2N \times 2N$ and CU index show in Fig. 11.

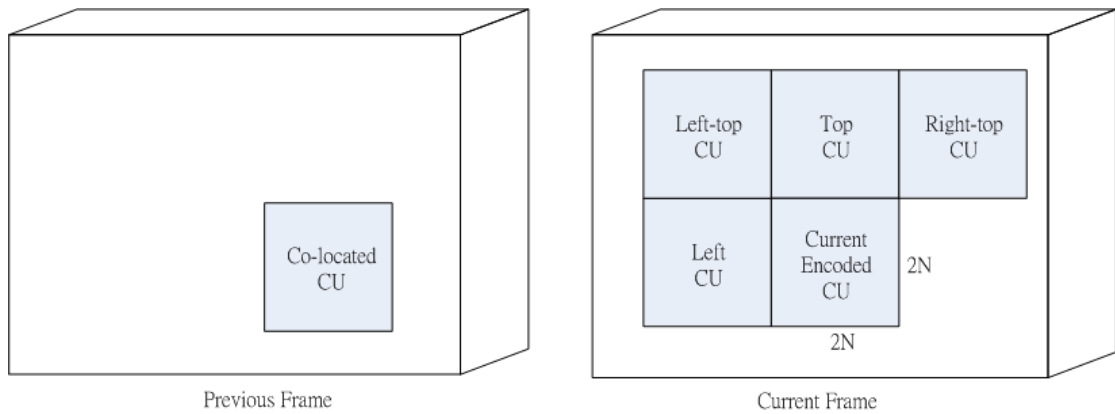


Fig. 12 Reference CUs and the current CU

As already stated in Chapter 3, some exceptions of losing reference CUs exist in Fig. 12 due to the encoding order or the picture boundary. When we want to encode a CU with index 4 in Fig. 11, the right-top referenced CU has not been processed, so the encoder can't find any information about the right-top CU as shown in Fig. 12. For this case, we ignore the right-top CU but still follow the decision rule that to be described in the next two sections. On the other hand, if the encoded CU is so close the boundary of picture that it loses more than one referred CU, it will find the best CU quadtree structure without our proposed fast decision.

4.3.1 Splitting Decision

The splitting decision is utilized for preventing the unnecessary prediction, RQT, and R-D calculation in a larger size CU. When the CU analysis begins at the current depth and all the following conditions are satisfied, the PU mode search in the current depth will be skipped except for the $2N \times N / N \times 2N$ inter modes, and then it jumps into the next depth directly. An example of splitting decision is illustrated in Fig. 13, where the current encoded CU in depth 0 chooses the splitting decision.

- The co-located CU has smaller CUs.
- All neighboring CUs have smaller CUs.
- The current encoding frame is not I frame.

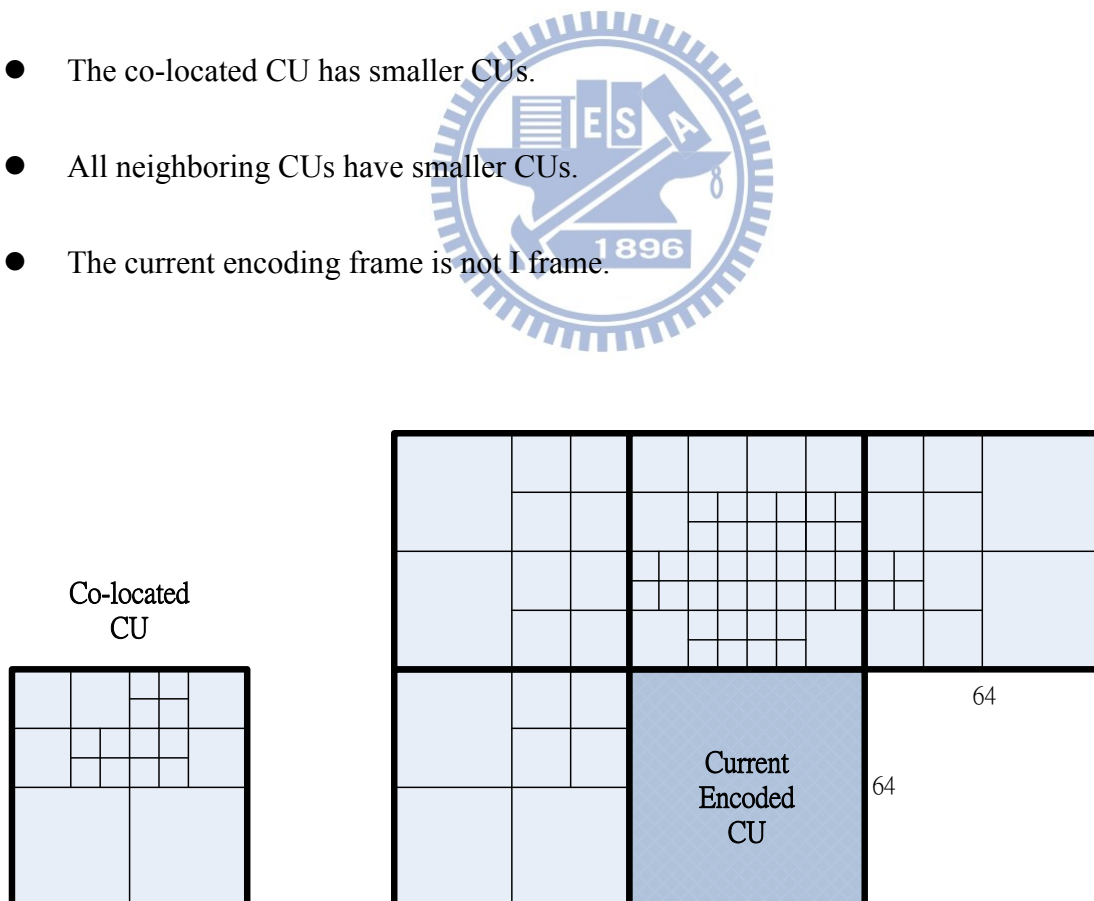


Fig. 13 An example of splitting decision

If all reference CUs prefer the splitting mode for lowering the R-D cost, which often implies that the region has complex residual texture, and the encoded block has a large probability in using the deeper depth to compress this CU. Nevertheless, when the depth of CU becomes smaller and smaller, we retain the inter modes, $2N \times N / N \times 2N$, with two MVs in the skipping data depth.

4.3.2 Termination Decision

The termination decision prevents the encoder from building a larger tree with a lot of computational complexity owing to the very small CUs. If the encoder has already finished the CU mode decision in the current depth, the termination decision is determined by the following conditions. The mode decision whose depth is greater than the current depth will not be conducted when all the conditions are satisfied. Fig. 14 shows an example of termination decision, and the current encoded CU will not build any nodes with the depth larger than 0 in the CU quadtree. The termination decision often occurs in the smooth residual texture region or the static background.

- The co-located CU does not have any smaller CU.
- 3 or more neighboring CUs do not have any smaller CU.
- The current encoding frame is not I frame.

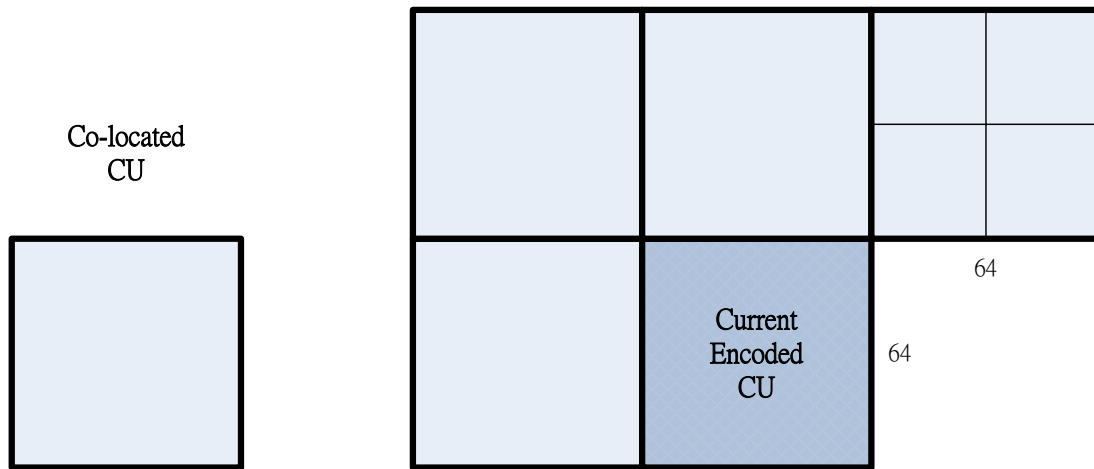


Fig. 14 An example of termination decision

4.3.3 Basic Fast CU Size Decision Scheme

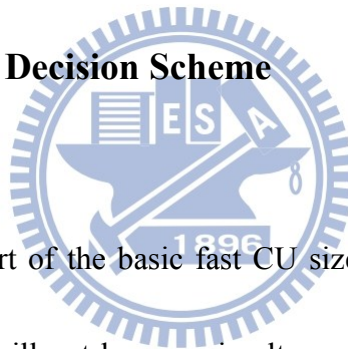


Fig. 15 shows the flowchart of the basic fast CU size decision algorithm. It should be noted that the 2 fast decisions will not happen simultaneously in each depth of the encoded CU. From the above sections, we know that splitting decision and termination decision will not happen in I frame because a mismatched CU size in intra frame will result in a great PSNR drop or bit rate increase. Moreover, for the co-located CU and the consistence of reference CU size, we set up the experimental conditions for low delay P having only one reference frame (only one co-located CU) and the GOP size is equal to one to avoid the automatic increase in QP.

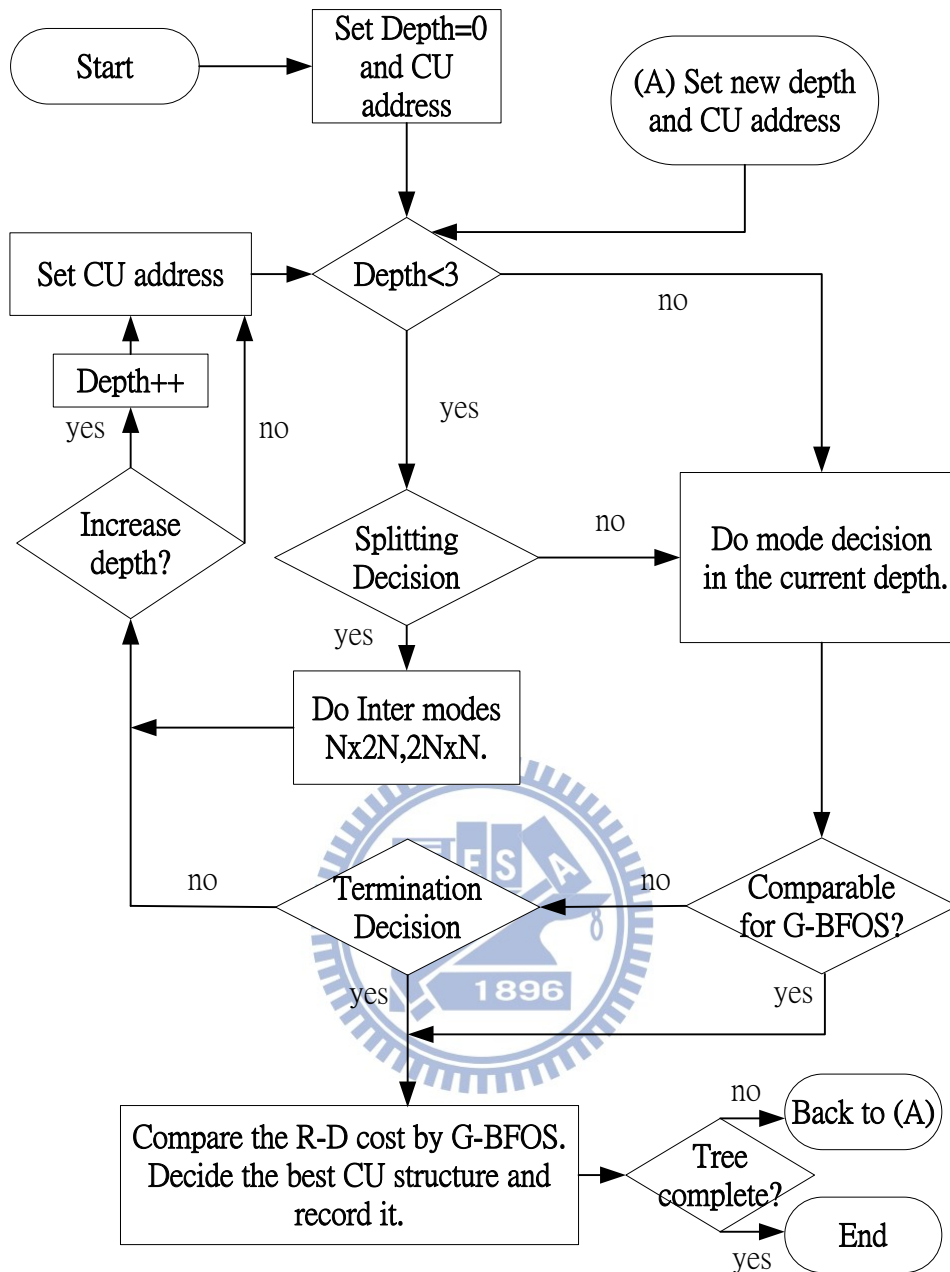


Fig. 15 Flowchart of basic fast CU size decision algorithm

4.4 Additional Tools

In this section, we try three methods to improve the performance of fast CU size decision.

There is no BD-rate measurement in [20] and [21], so we check our luma BD-rate, BD-PSNR, and R-D curve to find the weakness of the basic algorithm and enhance it for better efficiency.

First, we observe that the coding loss increases as QP gets larger, such as 32 and 37. Nevertheless, the high QP setting is important for real-time application, and we should solve this problem. Secondly, the time-saving is small in the lower QP cases. We want to solve this problem because the encoder usually spends a lot of time compressing the videos at lower QP. In the following sub-sections, we analyze the data from the result of the proposed basic algorithm and design the modifications.

4.4.1 Frame Level Parameter Control

We collect the result of eight high-resolution video sequences with 32 frames per sequence, and find that the performance is better than that of 16/2 CU structure which is defined in section 3.2, but the coding loss is too high. Table 6 lists the BD-performance and time reduction ratio, and Fig.16 shows the R-D curve of sequence “Basketball”.

The reference curve is the original HM, and the test curve is our proposed method. We can find that two curves separate far in the higher QP cases, and we also notice that the time reduction ratio is very high, which may drop some necessary mode calculations. In [21], the depth-consideration fast algorithm sets the target of complexity from 40% to 100%, and there is a large amount of R-D performance drop between 40% and 60%. Therefore, we like to modify the method to maintain an appropriate complexity and to improve its BD-performance. The improved method in [21] defines two types of frames: the unconstrained frames (F_u) and

the constrained ones (F_c). F_c represents that the CU in the frame is encoded with the fast algorithm. On the contrary, the CU in F_u is processed in the original way to find the best CU quadtree structure. Each F_u is followed by a number of N_c constrained frames F_c as illustrated by Fig. 17.

Table 6 Performance of the basic fast CU decision algorithm

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-50.58%	-37.73%	-33.35%	-35.83%	-38.29%	-41.56%	-39.06%	-40.78%	-39.65%
Time-Saving(QP27)	-57.94%	-41.80%	-44.40%	-51.38%	-36.90%	-51.74%	-50.37%	-53.91%	-48.56%
Time-Saving(QP32)	-56.62%	-46.10%	-50.93%	-55.60%	-45.34%	-60.38%	-56.03%	-60.28%	-53.91%
Time-Saving(QP37)	-55.81%	-53.26%	-56.15%	-61.44%	-54.38%	-65.85%	-61.48%	-67.31%	-59.46%
AVG. Time-Saving	-55.24%	-44.72%	-46.21%	-51.06%	-43.73%	-54.88%	-51.74%	-55.57%	-50.39%
Y BD-rate (%)	5.311	5.347	3.906	7.559	1.650	7.661	4.323	8.203	5.495
Y BD-PSNR (dB)	-0.147	-0.160	-0.086	-0.127	-0.049	-0.200	-0.132	-0.181	-0.135

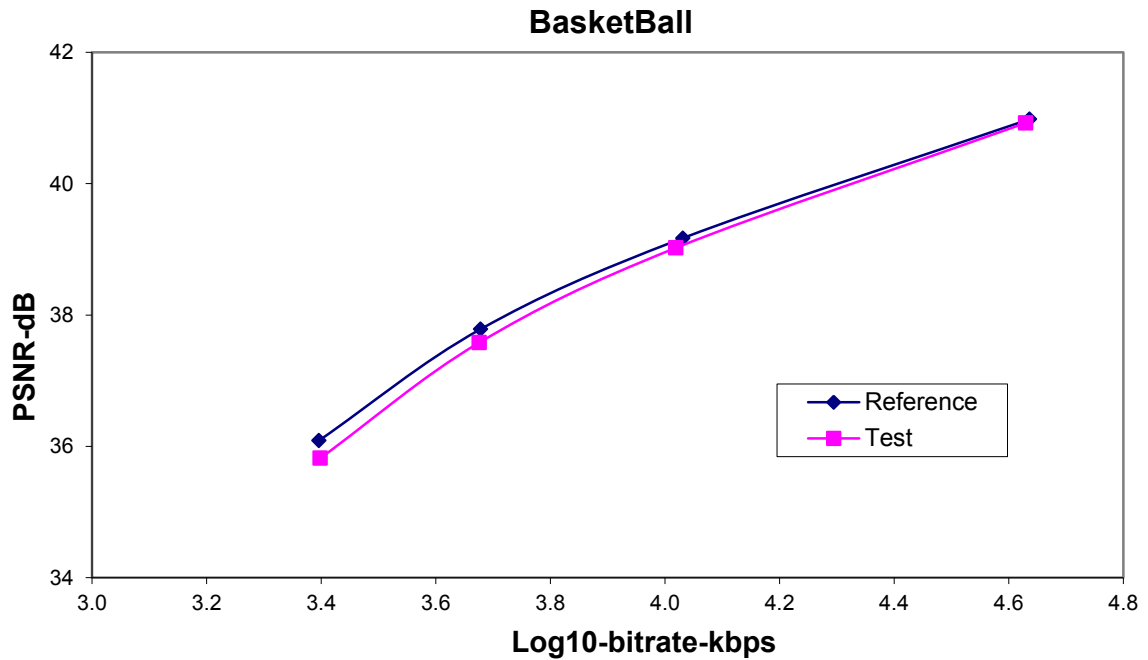


Fig. 16 R-D curve of Basketball

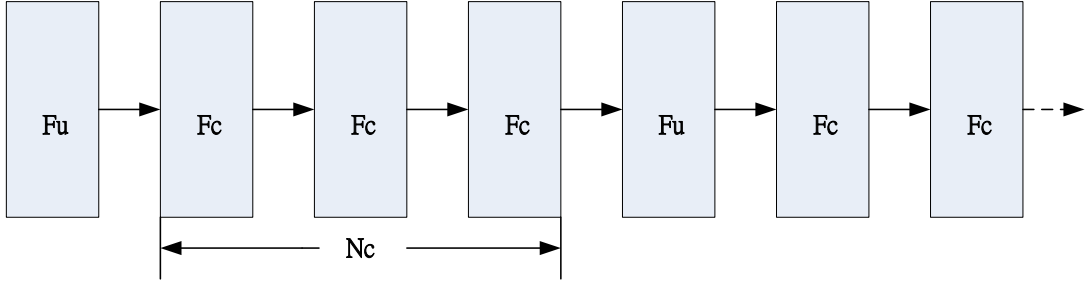


Fig. 17 Example of $N_c=3$

In Table 6, the BD performance drops due to the unlimited N_c . Our original proposed algorithm sometimes takes the reconstructed frame with lower PSNR as the reference frame which results in inaccurate prediction. Therefore, we should pay attention to the PSNR loss with fixed N_c and set the tolerable bound for the PSNR decrease. The experiments set N_c equal to 3, 6, 9, 12, and 15. Fig. 18 shows the suitable N_c as the intersection of two lines for QP=22, 27, 32, and 37, where over 75% sequences limit their drops of PSNR under 0.1dB compared to the result of the original HM. The testing sequences and the frame number are the same as the stated in the beginning of this section.

We use the results from Fig. 18 to select the proper integral N_c for the corresponding QP. Then, we estimate the relationship between N_c and QP. The minimum square error method is adopted for finding the approximated linear equation, which is

$$N_c = \text{round}(-0.32 \times QP + 14.94), \quad QP < 46 \quad (6)$$

N_c must be a positive integer, so we add the round operation outside the linear equation, and thus N_c is 0, when QP is larger than 46. The four QP values are taken into (6) iteratively to

produce stable N_c . Table 7 lists the finally selected N_c to the corresponding QP. Thus, we limit the value of N_c in our proposed fast algorithm. The BD-performance in Table 8 is much better than that in Table 6, and we also control the average time complexity at about 60%. The resulting R-D curves in Fig. 19 are closer to each other than those in Fig. 16, especially in the high QP region. The same improvement of R-D curve trend is also found in other sequences.

N_c is equal to 5 as QP=32. Therefore, we know the 7th frame and the 13th frame are encoded originally, and the other frames in Table 9 are processed with fast CU size decision.

In Table 9, the usage bits per frame alter less than 28% between the consecutive frames, and the maximum changed PSNR value is smaller than 0.14 dB. Although the reconstruction videos seem continuous as the original way, we should consider the stable bits usage and the video quality for the general applications.

Table 7 Specified QP versus N_c

QP	22	27	32	37
N_c	8	6	5	3

Table 8 BD-performance and time reduction ratio of limited N_c

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-41.54%	-33.41%	-31.47%	-31.57%	-35.91%	-34.27%	-33.46%	-32.43%	-34.26%
Time-Saving(QP27)	-45.73%	-33.05%	-36.50%	-40.78%	-32.31%	-40.75%	-39.23%	-41.21%	-38.70%
Time-Saving(QP32)	-44.32%	-35.11%	-39.86%	-42.78%	-33.90%	-46.06%	-43.99%	-44.93%	-41.37%
Time-Saving(QP37)	-40.10%	-37.53%	-40.19%	-42.76%	-37.67%	-46.42%	-44.29%	-46.64%	-41.95%
AVG. Time-Saving	-42.92%	-34.78%	-37.01%	-39.47%	-34.95%	-41.88%	-40.24%	-41.30%	-39.07%

Y BD-rate (%)	2.907	1.888	1.530	2.992	0.567	1.234	1.579	1.755	1.807
Y BD-PSNR (dB)	-0.080	-0.057	-0.032	-0.049	-0.018	-0.029	-0.049	-0.040	-0.044

Table 9 PSNR and bits measurements at QP=32

Vidyo1	PSNR (dB)	bits	BQTerrace	PSNR (dB)	bits
Frame7	39.5052	13016	Frame7	34.1248	193544
Frame8	39.4256	11408	Frame8	34.1320	201480
Frame9	39.4629	10016	Frame9	34.1214	201984
Frame10	39.4524	9432	Frame10	34.1148	200944
Frame11	39.3932	9656	Frame11	34.1008	200216
Frame12	39.3379	10848	Frame12	34.1081	197904
Frame13	39.4740	13784	Frame13	34.1911	211504

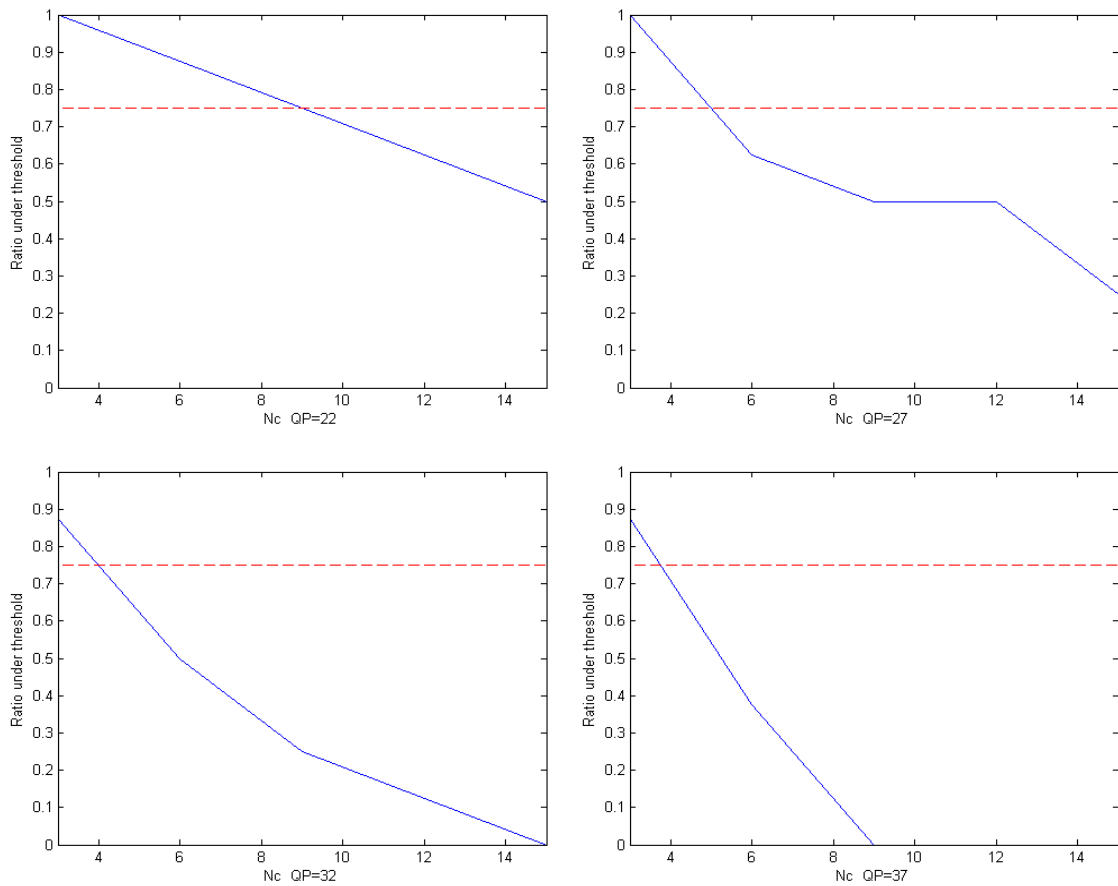


Fig. 18 Experiment for choosing N_c

The solid line means the ratio under the tolerable bound, and the dashed horizontal line represents the expected ratio which equals 75%.

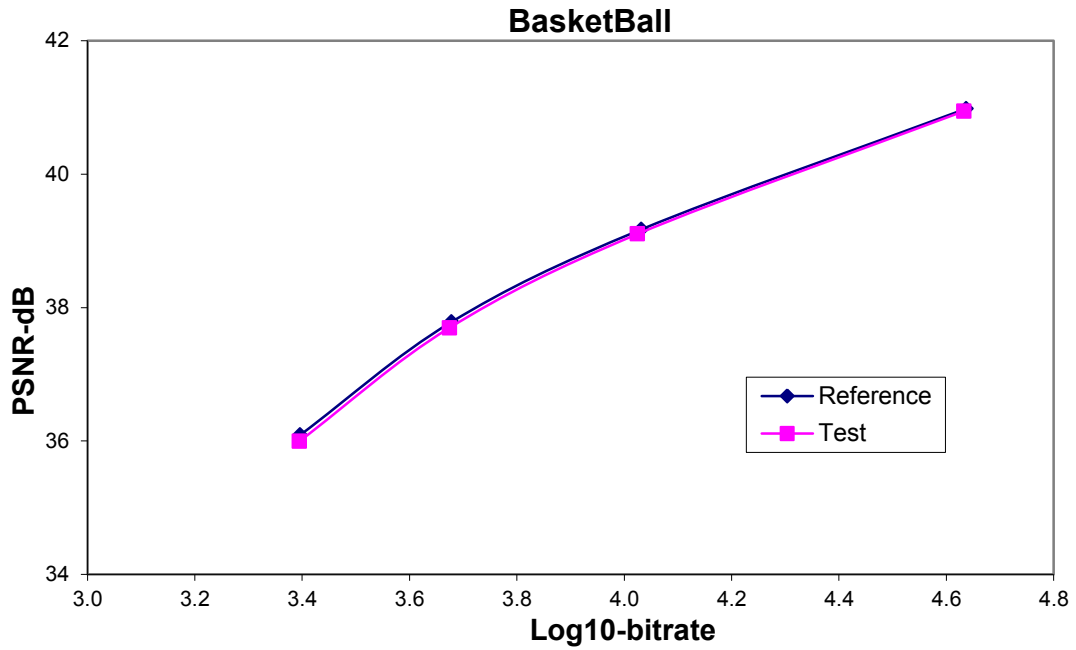


Fig. 19 R-D curve of Basketball with N_c control

4.4.2 LCU Level Parameter Control with Error-Bound

In this section, we focus on analyzing the distortion statistics between the original video and the reconstructed video at LCU level. By limiting our algorithm working in the high distortion region of the previous frame, the HEVC inter-prediction scheme can produce better matching block from the reconstructed frame. It should be noted that the data in this section is based on CU size fast decision with splitting information, and the used N_c values are different from those in other sections in this thesis.

At the beginning, we test our proposed algorithm including the N_c value control on eight high resolution sequences with 32 frames per sequence. Fig. 20 shows the probability density distribution of the sum of the absolute distortion (SAD) in each LCU at various QP.

Then, we also apply the minimum square error method to estimate the relationship between QP and the 3% bound of SAD as

$$\text{error bound (3\%)} = 1332.708 \text{ QP} - 22694.336 \quad (7)$$

where 3% means the top 3% error in our collected SAD data. In Fig. 20, we divide all collected data within the corresponding QP into 100 groups to calculate its probability density distribution, and the red dash lines in Fig. 20 indicates the position of the top 3% error for each QP.

When we insert 4 QP values into (7), we find that the estimated error bound values do not match our assumption. This is particularly true for the case of QP 22 and the corresponding error bound is 6625.24. The bound is on the left of the peak (near 7000) and thus excludes over 10% LCU for fast algorithm which decreases the time reduction. For accurate error bound, we try the second order approximation equation, and the result is

$$\text{error bound (3\%)} = 33.222 \text{ QP}^2 - 627.39 \text{ QP} + 5178.922 \quad (8)$$

Although the computation of second order equation is high, it gives us a better fitting curve to the original data. Fig. 21 shows the curve fitting, and Table 10 lists the 3% error bound of the specified QP. We add error bound threshold for LCU skipping into our proposed fast algorithm with limited N_c , and the simulation results of 1080P sequences with 32 frames per sequence are shown in Table 11.

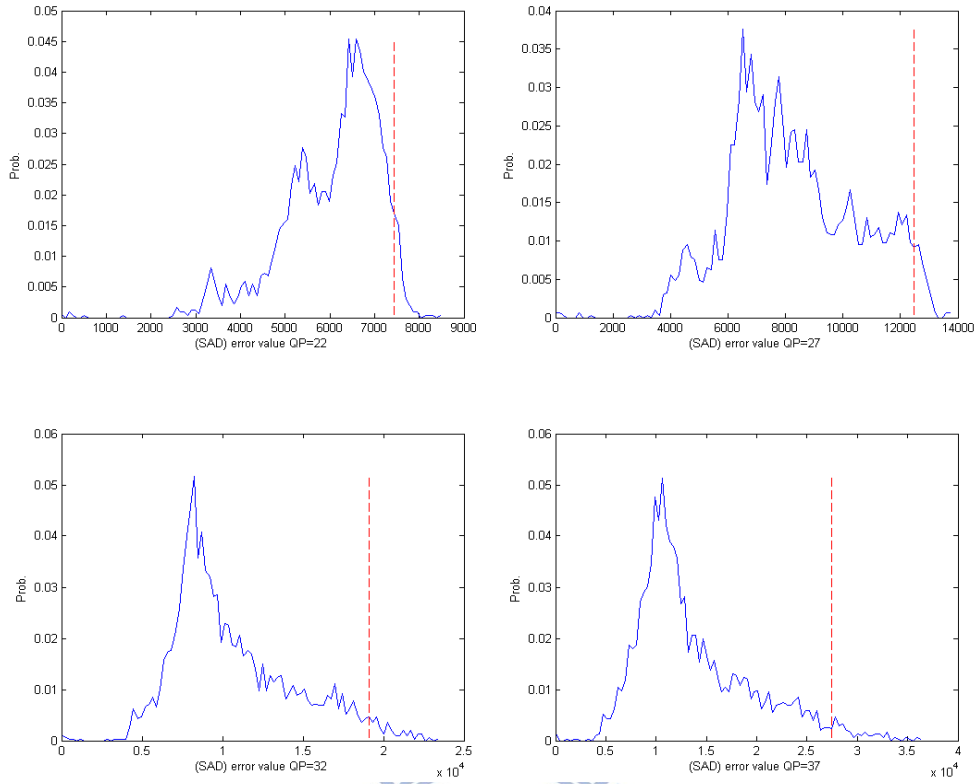


Fig. 20 Error bound (3%) for SAD

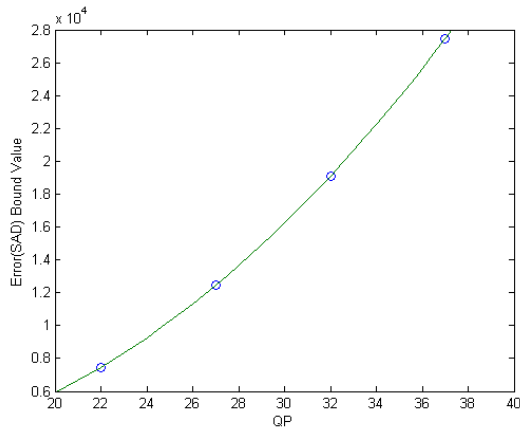


Fig. 21 Second order curve fitting for error bound (3%)

Table 10 Specified QP versus error bound (3%)

QP	22	27	32	37
ErrorBound (3%)	7455.79	12458.23	19121.77	27446.41

Table 11 BD-performance and time reduction ratio with 3% error bound

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace
Time-Saving(QP22)	-41.67%	-20.22%	-3.22%	-21.77%	-11.68%
Time-Saving(QP27)	-47.97%	-32.46%	-29.37%	-39.94%	-15.55%
Time-Saving(QP32)	-45.67%	-35.55%	-38.54%	-43.27%	-24.08%
Time-Saving(QP37)	-41.65%	-37.09%	-39.70%	-43.82%	-32.57%
AVG. Time-Saving	-44.24%	-31.33%	-27.71%	-37.20%	-20.97%
Y BD-rate (%)	3.062	2.083	1.296	2.972	0.455
Y BD-PSNR (dB)	-0.085	-0.063	-0.027	-0.051	-0.013

We only try 1080P sequences and discontinue trying other ones because the time reduction significantly decreases in the low QP setting. Nevertheless, the BD-performance improves a little bit. Thus, we can still use this method but this is a concern on time reduction. Therefore, the final scheme sets the threshold bound only on the QP equals 32 and 37. The simulation result with 64 frames per sequence is shown in Table 12. To test its robustness, we add eight lower resolution sequences to check the effect of error bounds. Table 13 shows the simulation result without error bound for comparison.

Table 12 Simulation result with 3% error bound with 64 frames per sequence

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-38.77%	-33.45%	-32.53%	-30.92%	-35.96%	-37.67%	-37.67%	-34.24%	-35.15%

Time-Saving(QP27)	-45.38%	-33.71%	-37.57%	-40.72%	-31.92%	-44.68%	-44.98%	-43.37%	-40.29%
Time-Saving(QP32)	-44.14%	-35.07%	-39.06%	-42.48%	-23.23%	-50.37%	-47.76%	-47.78%	-41.24%
Time-Saving(QP37)	-41.11%	-38.36%	-41.36%	-43.14%	-33.13%	-51.69%	-48.88%	-48.83%	-43.31%
AVG. Time-Saving	-42.35%	-35.15%	-37.63%	-39.32%	-31.06%	-46.10%	-44.82%	-43.56%	-40.00%
Y BD-rate (%)	3.364	2.395	2.063	3.229	0.696	1.702	2.530	2.430	2.301
Y BD-PSNR (dB)	-0.092	-0.074	-0.044	-0.061	-0.018	-0.043	-0.073	-0.055	-0.058

Test Sequence	BallDrill	BQMall	Party	HorsesC	BallPass	Bubbles	BQSquare	Horses	AVG.
Time-Saving(QP22)	-32.15%	-31.69%	-30.35%	-28.53%	-21.46%	-21.88%	-23.67%	-23.15%	-26.61%
Time-Saving(QP27)	-32.35%	-29.82%	-28.96%	-28.21%	-22.15%	-20.29%	-19.96%	-20.83%	-25.32%
Time-Saving(QP32)	-31.76%	-26.74%	-11.68%	-22.43%	-22.22%	-18.09%	-9.48%	-18.37%	-20.10%
Time-Saving(QP37)	-32.19%	-27.48%	-13.56%	-22.96%	-23.58%	-20.63%	-11.65%	-19.76%	-21.48%
AVG. Time-Saving	-32.11%	-28.93%	-21.14%	-25.53%	-22.35%	-20.22%	-16.19%	-20.53%	-23.38%
Y BD-rate (%)	4.413	2.309	0.510	1.762	1.611	0.662	0.161	0.820	1.531
Y BD-PSNR (dB)	-0.174	-0.094	-0.025	-0.075	-0.08	-0.028	-0.008	-0.04	-0.066

Table 13 Simulation result without error bound with 64 frames per sequence

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-39.19%	-33.60%	-32.56%	-32.48%	-37.12%	-38.12%	-37.62%	-34.90%	-35.70%
Time-Saving(QP27)	-45.27%	-34.25%	-37.86%	-42.01%	-32.66%	-45.15%	-45.06%	-44.35%	-40.83%
Time-Saving(QP32)	-44.11%	-36.19%	-41.76%	-43.72%	-35.01%	-51.07%	-48.21%	-48.55%	-43.58%
Time-Saving(QP37)	-41.12%	-39.17%	-43.51%	-44.55%	-41.48%	-52.18%	-49.53%	-49.77%	-45.16%
AVG. Time-Saving	-42.42%	-35.80%	-38.92%	-40.69%	-36.57%	-46.63%	-45.11%	-44.39%	-41.32%
Y BD-rate (%)	3.364	2.383	2.237	3.217	0.818	1.702	2.529	2.430	2.335
Y BD-PSNR (dB)	-0.092	-0.073	-0.047	-0.062	-0.024	-0.043	-0.073	-0.055	-0.059

Test Sequence	BallDrill	BQMall	Party	HorsesC	BallPass	Bubbles	BQSquare	Horses	AVG.
Time-Saving(QP22)	-31.47%	-30.51%	-29.38%	-27.90%	-23.54%	-23.17%	-24.38%	-23.22%	-26.70%
Time-Saving(QP27)	-30.98%	-28.75%	-27.94%	-28.07%	-22.88%	-20.61%	-21.19%	-21.51%	-25.24%
Time-Saving(QP32)	-31.90%	-29.03%	-25.93%	-27.43%	-24.06%	-21.17%	-19.56%	-20.44%	-24.94%
Time-Saving(QP37)	-32.61%	-29.80%	-25.83%	-26.56%	-25.64%	-24.25%	-19.34%	-21.35%	-25.67%
AVG. Time-Saving	-31.74%	-29.52%	-27.27%	-27.49%	-24.03%	-22.30%	-21.12%	-21.63%	-25.64%
Y BD-rate (%)	4.509	2.540	0.723	2.092	1.566	0.903	0.216	1.029	1.697
Y BD-PSNR (dB)	-0.178	-0.102	-0.037	-0.09	-0.078	-0.039	-0.011	-0.052	-0.073

We find that the BD-performance improves a little but not much, especially in some worst cases. Although the error bound scheme reduces the efficiency of time saving, we still increase the bound up to 5% to compare their performance and evaluate the necessity of the error bound. The comparison is in Table 14.

Table 14 Comparison of different ratios of error bound

High resolution sequences(1080P,720P)				Low resolution sequences			
Error Bound	BD-rate (%)	BD-PSNR(dB)	Time saving	Error Bound	BD-rate (%)	BD-PSNR(dB)	Time saving
none	2.335	-0.059	-41.32%	none	1.697	-0.073	-25.64%
3%	2.301	-0.058	-40.00%	3%	1.531	-0.066	-23.38%
5%	2.297	-0.057	-39.86%	5%	1.517	-0.065	-22.33%

From Table 14, we know that increasing the ratio of the error bound is not useful for coding gain, and there are 2 phenomena we notice in comparing Table 12 to Table 13. Firstly, the performance of “Kimono” seems no different with the error bound. Secondly, the time reduction becomes about half in the sequence of “Party”. For investigate these cases, we analyze their SAD distribution separately, and find that the measurement of error bound scheme is too rough for representing the individual sequences. The SAD distribution of “Kimono” is shown in Fig. 22, and the 3% threshold decided by all sequences does not work on the “Kimono” because its PSNR is higher than the average PSNR of the high resolution sequences. On the other hand, the threshold limits about 50% case for fast algorithm in “Party” because the threshold is located near the center of its SAD probability density distribution

which is shown in Fig. 23. In conclusion, we remove this tool from our algorithm due to the large variation of probability distribution of individual sequences and additional operations of calculation for SAD values. An effective threshold scheme should consider both PSNR and bitrate in setting up the adaptive threshold for different sequences.

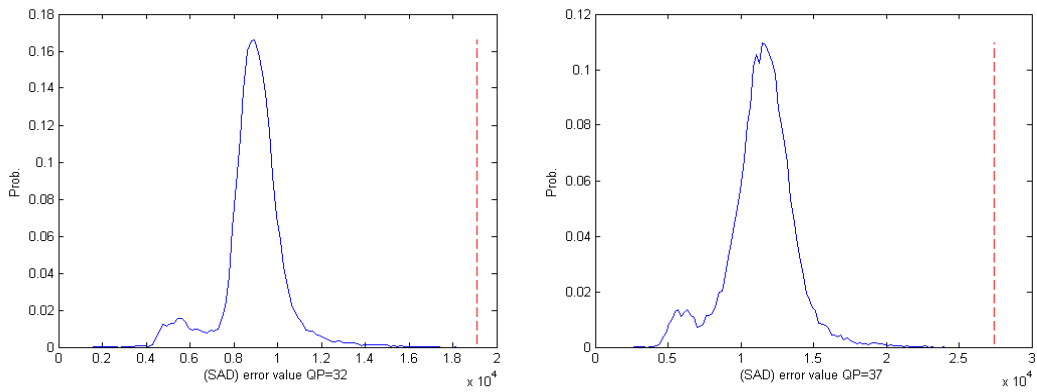


Fig. 22 Probability density distribution of SAD of “Kimono”

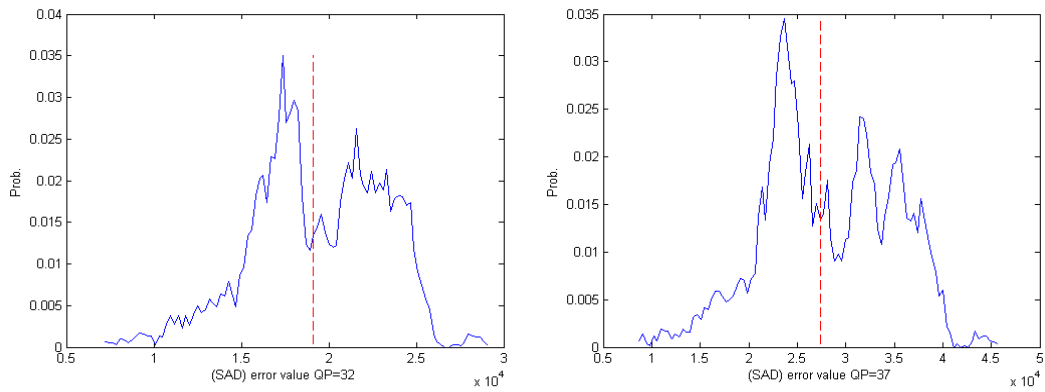


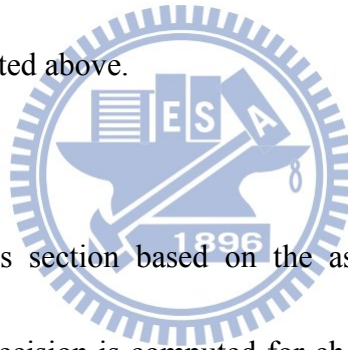
Fig. 23 Probability density distribution of SAD of “Party”

4.4.3 $2N \times N / N \times 2N$ Decision after Splitting Decision

In the last two sections, we notice that the time reduction of low QP is lower than that of high QP. We also know that the small sized CUs are often used in lower QP case, so the splitting decision occurs easily in the region of many small sized CUs. However, the time saving of the splitting decision is less than that of the termination decision, so we are expected to use only one inter prediction after the splitting decision to reduce the complexity further. There are 2 possible inter modes examined originally after the splitting decision, $2N \times N / N \times 2N$. We assume that the shape is highly dependent on the size of neighboring CUs. If the number of small CU in the horizontal direction is larger than that in the vertical direction, the encoder will compute the R-D cost of $2N \times N$ in the current depth. Otherwise, we will only use $N \times 2N$ prediction instead. The positions of reference CUs for depth 0 and 1 are shown in Fig. 24, and those for depth 2 are shown in Fig. 25.

In the example of Fig. 24, the current encoded CU of depth 1 refers CU_1 and CU_2 as the horizontal referenced CUs, and takes CU_3 and CU_4 as the vertical referenced CUs. That is, as the depth of current encoded CU is smaller than 2, the referenced CUs are the sub-CUs at the top and the left, and we decide the suitable mode by the splitting bits in those referenced CUs. However, we only save the splitting information of CU up to depth 2, so we should adjust the

decision rule for the encoded CU in the depth 2. In Fig. 25, CU₅ and CU₆ are horizontal referenced CUs for the current encoded CU of depth 2, and CU₉ and CU₁₀ are its vertical referenced CUs. As the encoded CU shifts in range of 3×3 in the current depth, the referenced CUs shift in the same way. It should be noted that CU₁₃ is located at the bottom row of the LCU, then the encoder takes CU₇ and CU₈ as the referred CUs due to that the CUs under CU₈ are not encoded. For symmetry, the CU₁₄ located the right column of LCU refers CU₁₁ and CU₁₂ to decide the shape of prediction in the current depth. The rest CUs after the splitting decision in the right edge and the bottom edge of green LCU refers the corresponding CUs with the similar way we stated above.



The design scheme in this section based on the assumption that after the splitting decision, the $2N \times N / N \times 2N$ decision is computed for choosing the only proper inter mode in the current depth. Table 15 shows the improvement of time reduction by the $2N \times N / N \times 2N$ decision, the number of testing frame is 64 per sequence. Obviously, the time reduction increases 3% in average with negligible coding loss, especially in lower QP, where higher percentage of splitting decisions happening.

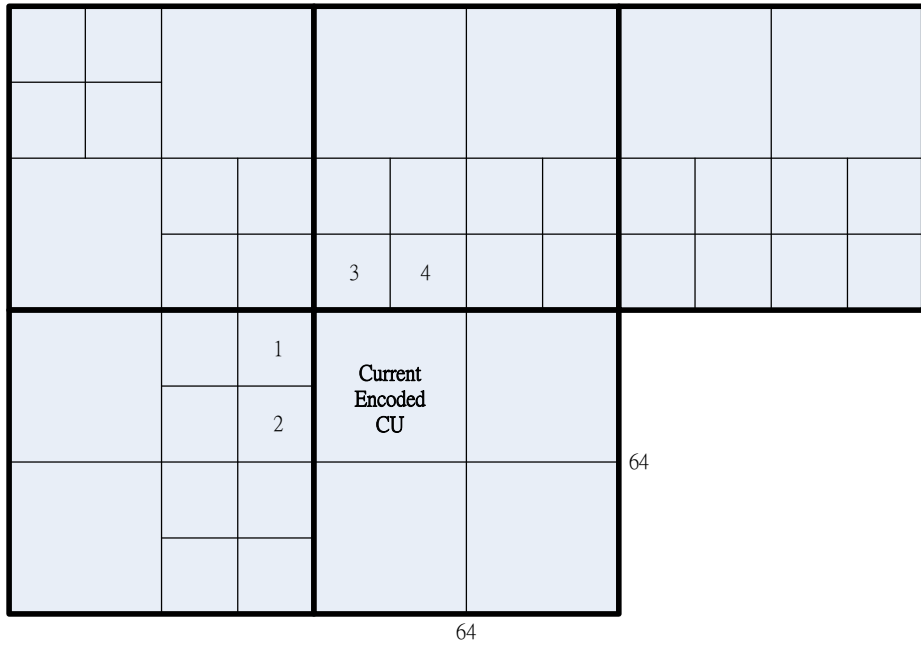


Fig. 24 An example of 2NxN/Nx2N Decision in depth 1

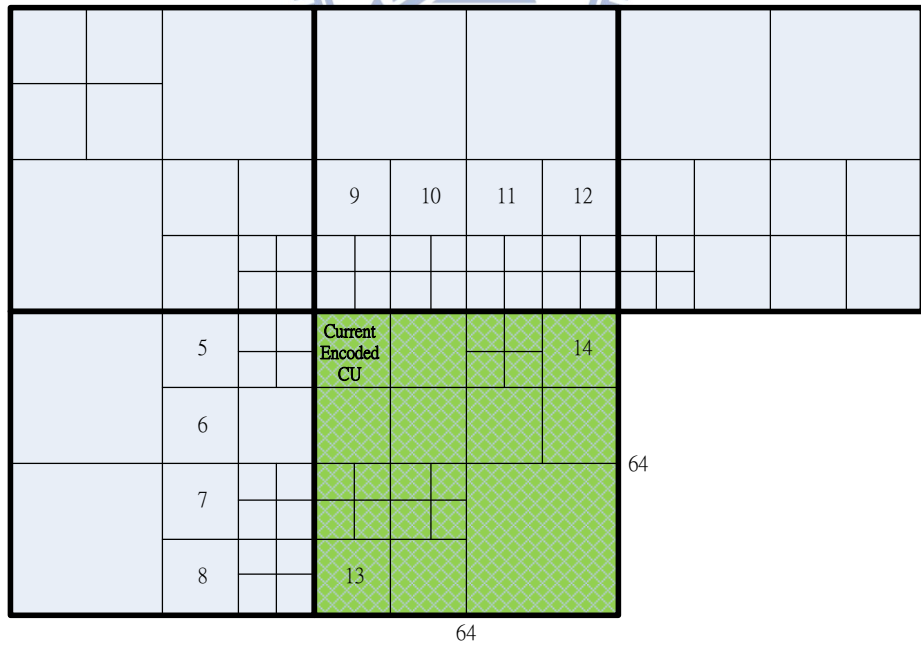


Fig. 25 An example of 2NxN/Nx2N Decision in depth 2

Table 15 Performance for schemes with and without $2N \times N / N \times 2N$ decision

With $2N \times N / N \times 2N$ Decision?	Average Time reduction QP=22	Average Time reduction QP=27	Average Time reduction QP=32	Average Time reduction QP=37	Average Time reduction	BD-rate (%)	BD-PSNR (dB)
No(1080P,720P)	-34.28%	-38.75%	-42.01%	-41.53%	-39.14%	2.089	-0.052
No (Other)	-25.01%	-23.31%	-23.72%	-23.39%	-23.86%	1.417	-0.062
Yes(1080P,720P)	-40.18%	-41.95%	-44.15%	-42.75%	-42.26%	2.050	-0.051
Yes (Other)	-30.81%	-28.16%	-27.07%	-25.55%	-27.90%	1.406	-0.060

4.5 Overview of the Overall Proposed Algorithm

In this chapter, we firstly propose the basic algorithm for fast CU size decision, and then we design two useful additional tools to enhance its coding performance and to increase time reduction, respectively. The detailed experiments and discussions are in the next chapter. The final flowchart of our algorithm is depicted in Fig. 26. The main additional parts are N_c control block and $N \times 2N$ decision block. N_c decision block executes before the splitting decision to reduce the coding loss by long-term N_c . $N \times 2N$ decision block places after the splitting decision to reduce the calculation from the unnecessary PU in the current depth. Due to these tools, our proposed scheme increases its efficiency.

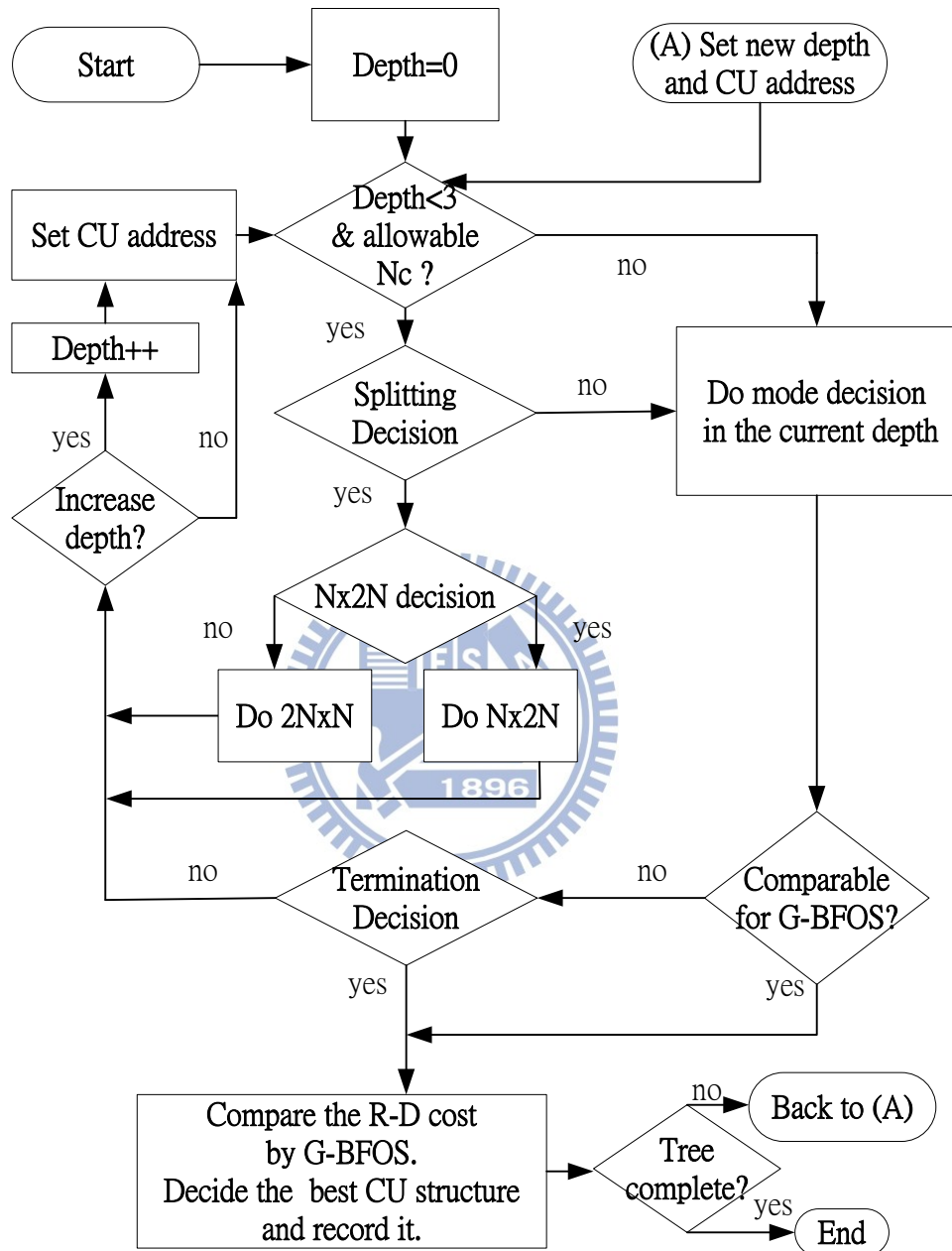


Fig. 26 Flowchart of overall proposed algorithm for processing an LCU

Chapter 5 HEVC Experiments and Discussions

In this chapter, we examine the performance of the proposed algorithm by testing 16 sequences with 100 frames per sequence. Then, we discuss the different time reduction efficiency due to the different depth combinations. The experiment conditions and the platform are already stated in section 2.3. The rest of this chapter is organized as follows. The performance measurements for all experiments in this study are listed in section 5.1, and then section 5.2 conducts several experiments and discussions for ECU, CFM, and our proposed algorithm. At the end of this chapter, we analyze the useful combination of the above algorithms.

5.1 Performance Measure

The time reduction, also called time saving (TS) in the thesis, is defined as

$$TS(QP_i) = \frac{Time_{tested}(QP_i) - Time_{referenced}(QP_i)}{Time_{referenced}(QP_i)} \times 100\% , \quad (9)$$

where $Time_{referenced}(QP_i)$ is the overall encoding time for referenced setting, such as the original HM5.0, and $Time_{tested}(QP_i)$ is the overall encoding time for the tested setting with the fast algorithm. QP_i is usually set as 22, 27, 32, or 37 (QP_1 , QP_2 , QP_3 , or QP_4) for BD-performance measurement described later. In general, we use the arithmetic average to represent the overall time saving ($TS_{average}$) as

$$TS_{average} = \frac{1}{4} \sum_{i=1}^4 TS_i \quad (10)$$

On the other hand, we also need a way to show the loss in the R-D performance as the trade-off for time reduction. The BD-rate and BD-PSNR [16] are adopted to measure the average performance in the most standard contests, so we use it to show the average difference between 2 R-D curves produced by the reference scheme and the proposed scheme. The BD-measurement [16] only needs the R-D results of 4 QP_i as mentioned previously to interpolate the overall R-D curve and further to estimate the average difference between 2 schemes.

When we want to observe the R-D performance of the specified QP, we analyze the data based on the formulas defined as (11) and (12) to represent the difference between the reference scheme and the proposed scheme.

$$\Delta PSNR = PSNR_{tested} - PSNR_{referenced} \quad (11)$$

$$\Delta BitRate(\%) = \frac{BitRate_{tested} - BitRate_{referenced}}{BitRate_{referenced}} \times 100\% \quad (12)$$

Last but not the least, the depth analysis is essential to know the strong and weak points of procedure for our fast decision algorithm. Thus, we should compare the depth changing trend due to the fast size decision to show the usefulness of each proposed tool. In (13), $AvgDepth$ means the average depth per LCU in the frame. $CUDepth_i$ is the depth of i_{th} CU, $\frac{CUArea_i}{LCUArea}$ is the area ratio of the i_{th} CU to LCU, $FrameArea$ is the area of the Frame, and n is the number of CU in a frame. This depth measurement is defined by [21].

$$AvgDepth = \frac{1}{(FrameArea / LCUArea)} \sum_{i=1}^n CUDepth_i \times \frac{CUArea_i}{LCUArea} \quad (13)$$

5.2 Experimental Results and Discussions

In this section, we show the results of our proposed fast algorithm including N_c control scheme and $2N \times N / N \times 2N$ decision described in subsection 5.2.1. Here, we also analyze the depth changing-trend in videos with different characteristics. Then, we simulate the original HM plus the ECU and CFM tools with the original low delay P and low complexity configuration in subsection 5.2.2, and compare them to the results of our schemes with GOP=1 and referenced frame=1. For the aggressive design, we add ECU and CFM into our proposed algorithm, and discuss the advantages and disadvantages caused by integrating these tools together. Hence, we have to find an efficient way to use these tools at proper QP values in subsection 5.2.3.

5.2.1 Fast CU Size Decision

The performance of our proposed fast decision in section 4.5 is listed in Table 16 (64 frames per sequence) and in Table 17 (100 frames per sequence) respectively. The reference scheme is the original HM5.0 without ECU and CFM. The simulation data in Table 16 shows that our scheme can averagely provide about 43 % overall encoding time saving in the high resolution test sequences. On the average, the luma BD-rate increment is about 2.24% and the luma BD-PSNR loss is about 0.06 dB.

When the number of frames increases, the BD-performance decreases slightly because we set the number of the reference frame is 1 to lower complexity but the inaccurate prediction in the IPPP sequence type also decreases coding performance. Hence, we should select a suitable intra period for the fast decision scheme, when the loss is not tolerated. The changing trend is about -0.2% BD-rate as adding the additional 32 encoded frames, averagely.

Table 16 Performance of the overall proposed algorithm (64 frames/sequence)

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-42.73%	-39.54%	-39.84%	-38.61%	-43.65%	-39.39%	-40.03%	-37.62%	-40.18%
Time-Saving(QP27)	-45.88%	-36.83%	-39.77%	-43.17%	-37.57%	-44.09%	-44.64%	-43.66%	-41.95%
Time-Saving(QP32)	-45.76%	-38.21%	-42.96%	-43.80%	-36.73%	-49.53%	-47.90%	-48.27%	-44.15%
Time-Saving(QP37)	-40.04%	-37.47%	-41.86%	-41.82%	-39.34%	-48.72%	-46.19%	-46.58%	-42.75%
AVG. Time-Saving	-43.60%	-38.01%	-41.11%	-41.85%	-39.32%	-45.43%	-44.69%	-44.03%	-42.26%
Y BD-rate (%)	3.094	2.173	1.909	2.723	0.789	1.059	2.460	2.189	2.050
Y BD-PSNR (dB)	-0.084	-0.067	-0.040	-0.051	-0.022	-0.024	-0.074	-0.049	-0.051

Test Sequence	BallDrill	BQMall	Party	HorsesC	BallPass	Bubbles	BQSquare	Horses	AVG.
Time-Saving(QP22)	-34.57%	-35.71%	-35.81%	-33.49%	-24.73%	-27.29%	-28.15%	-26.72%	-30.81%
Time-Saving(QP27)	-33.20%	-31.82%	-33.30%	-31.62%	-23.73%	-23.45%	-24.18%	-23.96%	-28.16%
Time-Saving(QP32)	-33.34%	-30.81%	-30.72%	-30.96%	-23.47%	-22.56%	-21.67%	-23.02%	-27.07%
Time-Saving(QP37)	-31.69%	-29.47%	-27.97%	-27.98%	-23.83%	-22.87%	-18.65%	-21.96%	-25.55%
AVG. Time-Saving	-33.20%	-31.95%	-31.95%	-31.01%	-23.94%	-24.04%	-23.16%	-23.92%	-27.90%
Y BD-rate (%)	3.583	2.231	0.613	1.773	1.255	0.788	0.187	0.819	1.406
Y BD-PSNR (dB)	-0.143	-0.090	-0.030	-0.077	-0.062	-0.033	-0.009	-0.040	-0.061

Table 17 Performance of the overall proposed algorithm (100 frames/sequence)

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-41.95%	-39.66%	-39.66%	-38.74%	-43.05%	-43.60%	-40.60%	-37.02%	-40.54%
Time-Saving(QP27)	-45.09%	-36.43%	-39.73%	-45.33%	-36.69%	-48.33%	-45.06%	-43.06%	-42.47%
Time-Saving(QP32)	-44.97%	-37.73%	-42.70%	-47.73%	-36.87%	-52.83%	-48.97%	-47.41%	-44.90%
Time-Saving(QP37)	-39.85%	-36.84%	-41.02%	-45.67%	-41.82%	-51.89%	-46.32%	-46.35%	-43.72%
AVG. Time-Saving	-42.97%	-37.67%	-40.78%	-44.37%	-39.61%	-49.16%	-45.24%	-43.46%	-42.91%

Y BD-rate (%)	3.112	2.231	2.052	2.775	0.934	1.839	2.615	2.379	2.242
Y BD-PSNR (dB)	-0.084	-0.069	-0.044	-0.056	-0.026	-0.043	-0.077	-0.053	-0.057

Test Sequence	BallDrill	BQMall	Party	HorsesC	BallPass	Bubbles	BQSquare	Horses	AVG.
Time-Saving(QP22)	-34.44%	-35.32%	-36.13%	-34.32%	-26.48%	-27.52%	-28.50%	-27.25%	-31.25%
Time-Saving(QP27)	-33.37%	-31.55%	-33.15%	-32.32%	-25.15%	-23.43%	-24.53%	-24.53%	-28.50%
Time-Saving(QP32)	-32.98%	-31.29%	-30.49%	-30.99%	-24.49%	-22.67%	-22.06%	-23.07%	-27.26%
Time-Saving(QP37)	-31.12%	-29.63%	-27.72%	-28.40%	-23.65%	-23.49%	-20.02%	-22.23%	-25.78%
AVG. Time-Saving	-32.98%	-31.95%	-31.87%	-31.51%	-24.94%	-24.28%	-23.78%	-24.27%	-28.20%
Y BD-rate (%)	3.422	2.747	0.588	1.923	1.317	0.883	0.301	1.008	1.524
Y BD-PSNR (dB)	-0.134	-0.109	-0.028	-0.084	-0.066	-0.036	-0.015	-0.051	-0.065

It should be noted that the low resolution sequences has less time saving averagely. The main reason is that the depth combination of low resolution sequences is often different from that of the high resolution sequences. In general, the encoder takes more 8×8 CUs as QP equals 22, and the large sized CU is usually used in the case of the higher QP and the static region. Our proposed algorithm consists of splitting decision and termination decision. Splitting decision can speed up the convergence of small CUs area. On the other hands, termination decision cuts off unnecessary depth in the CU quadtree construction resulting in larger CU sizes. The depth data of our experiments explains the above observation. Here, we examine 3 sequences with the specified QP in the consecutive frames, Vidyo1 (QP=32), BQsquare (QP=32), and BQTerrence (QP=22). In Table 18, the depth distribution is listed from 7^h to 13th frames for observing the complete acceleration period with QP=32. It is should be mentioned that all the depth measurements in the section include the area factor.

Table 18 Depth percentage (QP is 32)

Vidyo1	Depth0	Depth1	Depth2	Depth3	BQsquare	Depth0	Depth1	Depth2	Depth3
Frame7	44.0%	28.8%	20.0%	7.3%	Frame7	0.0%	15.4%	28.7%	55.9%
Frame8	61.8%	21.9%	14.3%	2.1%	Frame8	0.0%	19.5%	33.3%	47.2%
Frame9	59.1%	25.4%	14.6%	0.9%	Frame9	0.0%	19.5%	34.1%	46.4%
Frame10	65.3%	16.6%	17.6%	0.5%	Frame10	0.0%	14.4%	37.9%	47.7%
Frame11	60.0%	20.8%	18.9%	0.4%	Frame11	0.0%	12.3%	36.4%	51.3%
Frame12	68.4%	17.1%	14.2%	0.2%	Frame12	0.0%	16.4%	26.7%	56.9%
Frame13	51.6%	26.6%	16.0%	5.9%	Frame13	0.0%	20.5%	28.5%	51.0%

We notice that the larger CUs are seldom used in the sequence “BQsquare”. The same phenomenon usually happens in the small sized videos even when QP is 37. Table 19 shows that the depth combination of the 10th frame in all small sequences as QP equals 37. For comparison, we also list the depth combination in the high resolution videos with the same conditions in Table 20.

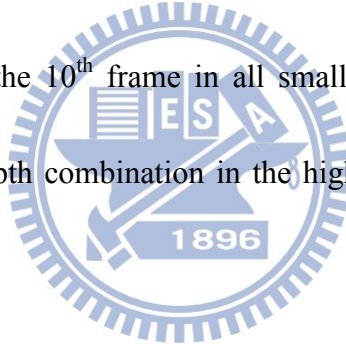


Table 19 Depth percentage of the 10th frame in low resolution sequences (QP=37)

Frame10	BallDrill	BQMall	Party	HorsesC	BallPass	Bubbles	BQSquare	Horses
Depth0	49.2%	44.1%	3.1%	1.0%	32.8%	4.1%	0.0%	0.0%
Depth1	24.1%	29.2%	25.6%	38.7%	32.8%	45.1%	31.8%	11.3%
Depth2	20.5%	18.5%	42.9%	42.2%	23.8%	37.7%	41.0%	55.9%
Depth3	6.2%	8.1%	28.4%	18.0%	10.5%	13.1%	27.2%	32.8%

Table 20 Depth percentage of the 10th frame in HD sequences (QP=37)

Frame10	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4
Depth0	25.5%	35.4%	55.7%	55.5%	39.5%	75.6%	77.3%	80.9%
Depth1	54.0%	39.8%	24.0%	31.5%	32.6%	18.0%	12.4%	15.4%
Depth2	19.4%	20.8%	16.7%	11.5%	22.6%	5.9%	9.8%	3.4%
Depth3	1.1%	4.0%	3.5%	1.5%	5.3%	0.6%	0.5%	0.3%

The time saving measure is highly depends on depth combination. For example, Vidyo1 has many large CUs in the case of QP=32, and its depth combination tends to be larger in size because the termination decision works frequently. The time reduction ratio analysis of Vidyo1 is listed in Table 21. Moreover, we also find the depth information of BQsquare in Table 18 with large amount small size CU, so the splitting decision is the major fast decision operation as shown in Table21.

Table 21 Time reduction ratio analysis of Vidyo1 and BQsquare

Vidyo1 (QP32 with encoding 64frames)			BQsquare (QP32 with encoding 64frames)		
Fast Setting	Time (sec)	TS	Fast Setting	Time (sec)	TS
None	581.968	0%	None	106.910	0%
Overall	293.730	-49.53%	Overall	83.742	-21.67%
Only Termination	306.052	-47.41%	Only Termination	102.062	-4.53%
Only Splitting	570.772	-1.92%	Only Splitting	87.600	18.06%

Although Vidyo1 has more time reduction than that of BQsquare, it does not mean that the splitting decision is useless relative to the termination decision. It depends on the depth combination and the CU distribution in a frame. For example, Sequence “BQTerrence” (QP=22) with dense small size CUs leads to that the splitting decision is the major fast decision and that the time saving is about 44%. Table 22 lists its depth combination and Table 23 shows its time reduction analysis. Further, we show the real examples of 9th frame of sequences “BQsquare (QP=32)”, “Vidyo1 (QP=32)”, and “BQTerrence (QP=22)”,

respectively in Fig. 27, Fig. 28, and Fig 29. We also show the depth convergence processes of “BQsquare (QP=32)” and “Vidyo1 (QP=32)” from 12th frame to 17th frame in the pie chart respectively in Fig. 30 and Fig. 31.

Table 22 Depth percentage of BQTerrence (QP=22)

BQTerrence	Frame12	Frame13	Frame14	Frame15	Frame16	Frame17
Depth 0	2.0%	2.4%	2.6%	2.4%	2.0%	2.2%
Depth 1	12.9%	12.0%	12.7%	10.7%	11.5%	11.6%
Depth 2	13.7%	13.6%	12.1%	15.8%	13.4%	13.5%
Depth 3	71.5%	71.9%	72.6%	71.1%	73.1%	72.7%

Table 23 Time reduction ratio analysis BQTerrence (QP=22)

Fast Setting	None	Overall	Only Splitting	Only Termination
Time(Sec)	3536.034	1992.612	2168.625	3367.291
TS	0%	-43.65%	-38.67%	-4.77%

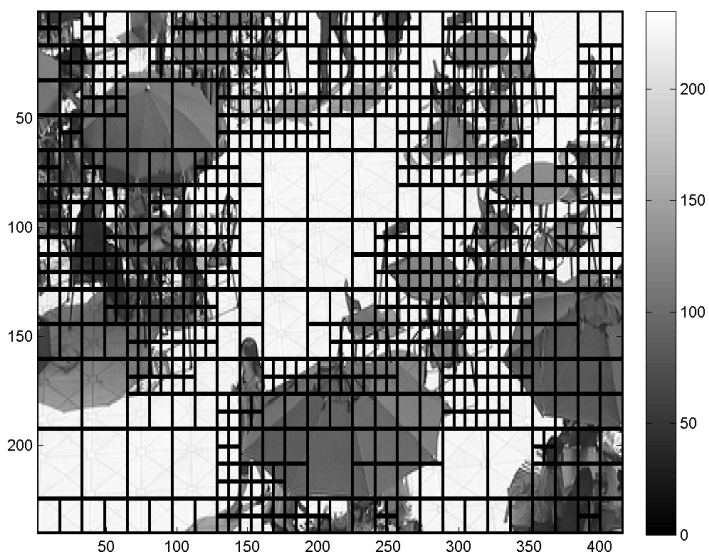


Fig. 27 CU distribution of the 9th frame of BQsquare (QP=32)



Fig. 28 CU distribution of the 9th frame of Vidyol (QP=32)

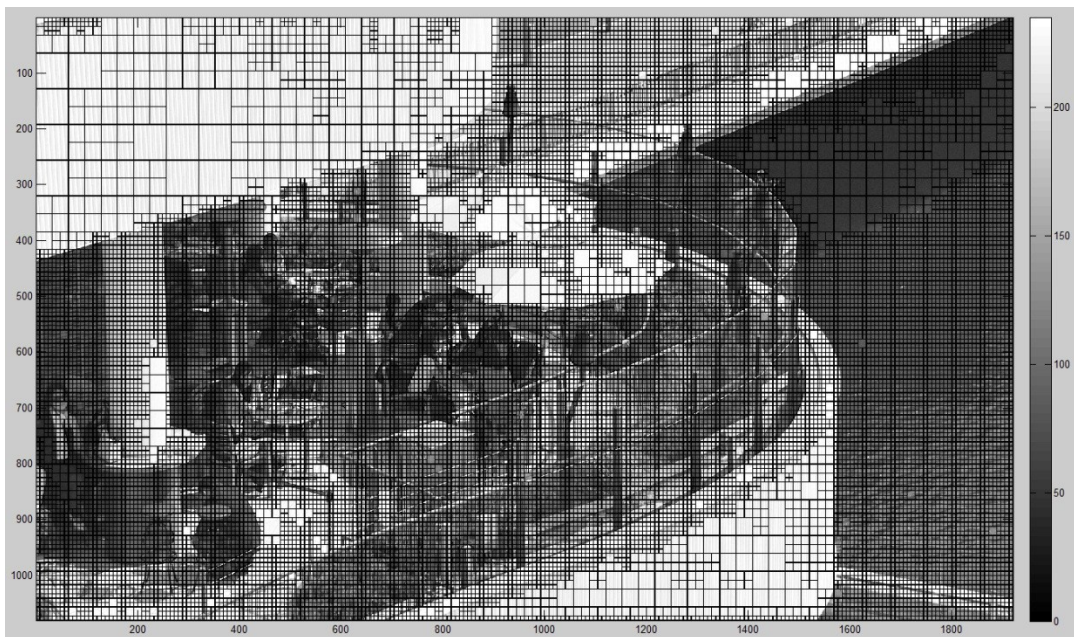


Fig. 29 CU distribution of the 9th frame of BQTerrence (QP=22)

According to the CU distribution in Fig. 27 and Fig. 29, we find that sequence “BQTerrence” has densely populated small CU in the center. Therefore, the splitting decision in sequence “BQTerrence” appears more often than that in sequence “BQsquare”.

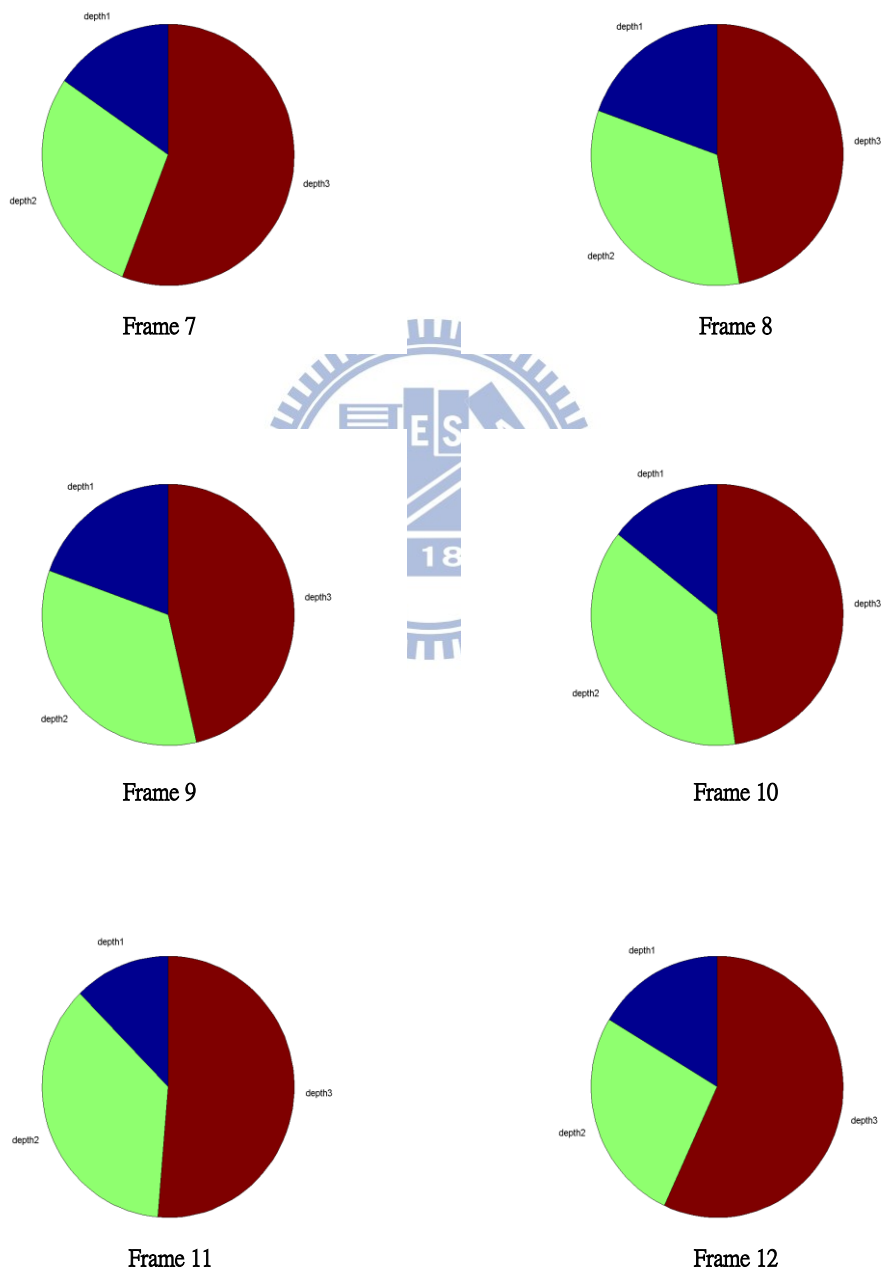


Fig. 30 Pie chart of depth amount ratio of BQsquare (QP=32)

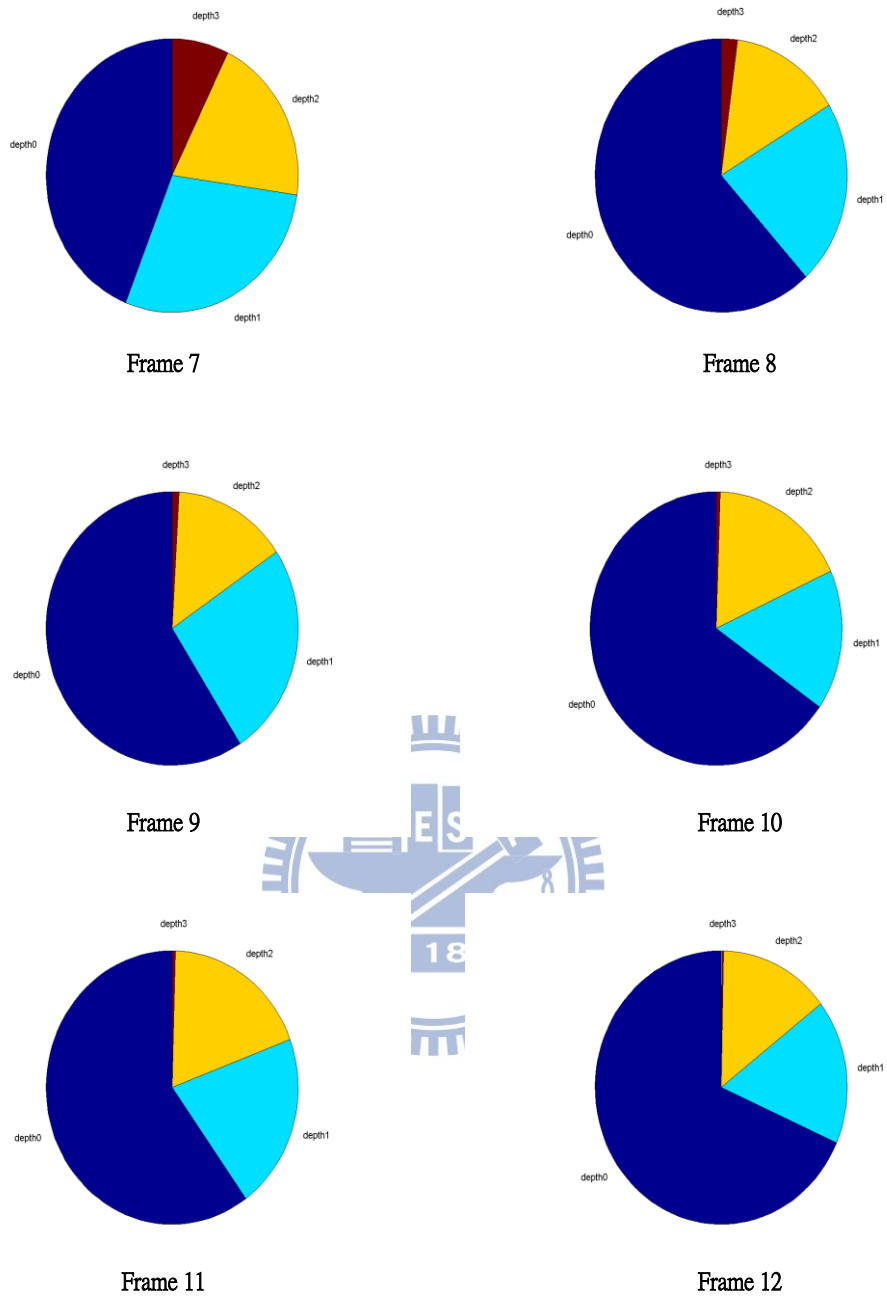


Fig. 31 Pie chart of depth amount ratio of Vidyo1 (QP=32)

From the above experiments and discussions of several frames in three sequences, the CU changing trend is dominated by the majority CU sizes. Furthermore, we like to examine the CU distribution for the entire encoding period, and we also combine the CU area factor with the amount of the specified CU sizes to represent the depth information. Hence, we

illustrate two significantly different properties of $AvgDepth$ defined in (13) for sequences

“Vidyo1 (QP is 37)” and “BQTerrence (QP is 22)” in Fig. 32 and Fig. 33, respectively.

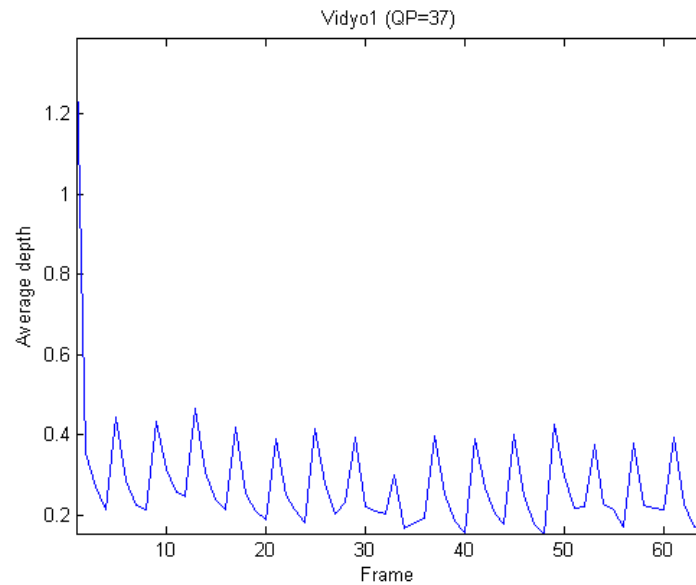


Fig. 32 Average depth of Vidyo1 (QP=37)

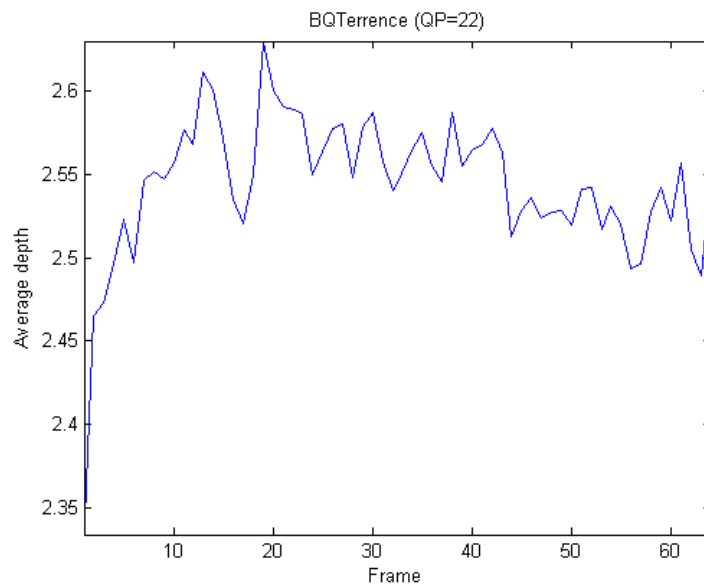


Fig. 33 Average depth of BQTerrence (QP=22)

The majority CU depths are 0 and 1 in Fig.32. When the frame is not N_c , which is encoded without our proposed fast decision, the average depth then increases slightly since the encoder uses the small size CU for coding detailed residual texture. In Fig. 33, the majority CU depths are 2 and 3 obviously. The average depth is almost the same no matter the fast decision turns on or off, so the BD loss is the minimal among high resolution sequences. However, Fig. 32 and Fig. 33 are extreme examples for explaining the phenomena of changing trend. In general, most encoding cases in the middle QP region have the uniform depth distribution. Thus, the termination decision and splitting decision both are needed for saving time.



In summary, we propose the fast CU size decision algorithm including splitting decision and termination decision with 2 additional tools, which are N_c control scheme and $2N \times N / N \times 2N$ decision. When the video is encoded mostly by small sized CUs, the encoding procedure can speed up by the splitting decision operation. On the other hand, as the encoder uses more large sized CUs for processing the video, it will benefit from the termination decision operation. The simulation results of high resolution sequences in Table 17 show that our fast decision method averagely provides about 43% overall encoding time reduction, and the BD-rate increases by about 2.24%.

5.2.2 Comparison with ECU/CFM

In this section, we enable the fast encoding tools, ECU and CFM, to accelerate HM5.0 without our proposed scheme. The simulation results with the original low delay P with low complexity setting (GOP=4 and 4 reference frames) are listed in Table 24 for eight high resolution sequences (32 frames per sequence).

Table 24 Simulation results of ECU and CFM with low delay_P loco setting

Only ECU	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-8.92%	-21.29%	-15.51%	-14.37%	-13.88%	-47.18%	-39.21%	-39.27%	-24.95%
Time-Saving(QP27)	-18.53%	-37.40%	-33.56%	-28.52%	-40.82%	-59.46%	-52.29%	-54.33%	-40.61%
Time-Saving(QP32)	-31.86%	-50.42%	-43.71%	-40.64%	-57.87%	-66.24%	-60.97%	-63.26%	-51.87%
Time-Saving(QP37)	-44.92%	-60.55%	-51.96%	-50.39%	-68.05%	-70.71%	-67.03%	-68.92%	-60.32%
AVG. Time-Saving	-26.06%	-42.42%	-36.19%	-33.48%	-45.16%	-60.90%	-54.88%	-56.45%	-44.44%
Y BD-rate	0.456	0.640	0.765	0.399	1.410	-0.159	0.916	-0.028	0.550
Y BD-PSNR	-0.015	-0.019	-0.014	-0.008	-0.022	0.009	-0.022	-0.001	-0.012

Only CFM	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-16.33%	-27.21%	-20.64%	-21.82%	-18.02%	-41.10%	-35.08%	-36.43%	-27.08%
Time-Saving(QP27)	-24.90%	-39.38%	-33.76%	-31.70%	-41.24%	-49.82%	-45.72%	-47.16%	-39.21%
Time-Saving(QP32)	-34.93%	-47.96%	-41.95%	-40.00%	-51.63%	-53.75%	-51.50%	-52.74%	-46.81%
Time-Saving(QP37)	-43.54%	-52.86%	-47.24%	-46.39%	-56.45%	-55.88%	-54.97%	-55.66%	-51.62%
AVG. Time-Saving	-29.93%	-41.85%	-35.90%	-34.98%	-41.84%	-50.14%	-46.82%	-48.00%	-41.18%
Y BD-rate	0.449	0.756	1.126	1.044	1.046	0.713	0.964	0.622	0.840
Y BD-PSNR	-0.015	-0.023	-0.024	-0.022	-0.021	-0.017	-0.028	-0.018	-0.021

Both ECU and CFM	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-20.64%	-38.06%	-29.46%	-28.39%	-24.40%	-61.66%	-52.04%	-53.29%	-38.49%
Time-Saving(QP27)	-33.31%	-56.06%	-48.52%	-43.56%	-60.85%	-74.48%	-67.75%	-70.85%	-56.92%
Time-Saving(QP32)	-48.60%	-68.92%	-59.88%	-56.35%	-76.65%	-81.01%	-76.77%	-78.75%	-68.37%
Time-Saving(QP37)	-61.81%	-77.30%	-68.48%	-66.44%	-84.44%	-84.91%	-82.32%	-83.30%	-76.13%

AVG. Time-Saving	-41.09%	-60.09%	-51.59%	-48.69%	-61.59%	-75.52%	-69.72%	-71.55%	-59.98%
Y BD-rate	0.804	2.667	3.155	1.469	3.026	0.824	2.553	0.558	1.882
Y BD-PSNR	-0.026	-0.079	-0.062	-0.027	-0.051	-0.024	-0.073	-0.01	-0.044

We notice that the time saving with low QP is less than that with high QP, and it achieves about 60% time saving with increasing BD-rate 1.88% when ECU and CFM both turn on. Another interesting observation is the side-effect of combining 2 fast algorithms together. For example, the ideal maximum time saving is 75% for perfectly combining two 2x faster algorithms. That is, the overall time saving is less than the ideal maximum time saving, but the overall loss of BD-rate is higher than the sum of their separate coding loss. Unfortunately, our proposed method has not been designed for adaptive QP case and multiple referenced frames yet, so we simulate the ECU and CFM in HM5.0 with our low delay P and low complexity setting (GOP =1 and 1 referenced frames), and the result is listed in Table 25 with eight high resolution sequences (64 frames per sequence).

Table 25 Simulation results of ECU and CFM with our low delay_P loco setting

Both ECU and CFM	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-6.34%	-5.92%	-1.90%	-4.98%	-5.57%	-37.84%	-34.89%	-27.63%	-15.63%
Time-Saving(QP27)	-15.28%	-23.51%	-22.77%	-19.34%	-15.22%	-55.57%	-49.39%	-43.12%	-30.53%
Time-Saving(QP32)	-23.29%	-38.69%	-37.20%	-31.84%	-33.24%	-69.03%	-62.68%	-59.87%	-44.48%
Time-Saving(QP37)	-33.49%	-53.56%	-47.81%	-43.49%	-57.75%	-77.06%	-71.64%	-70.62%	-56.93%
AVG. Time-Saving	-19.60%	-30.42%	-27.42%	-24.91%	-27.95%	-59.88%	-54.65%	-50.31%	-36.89%
Y BD-rate	0.513	1.230	0.965	0.818	0.659	-0.976	0.893	0.006	0.514
Y BD-PSNR	-0.014	-0.038	-0.021	-0.016	-0.019	0.026	-0.028	-0.001	-0.014

The QP in my setting is smaller than that in the GOP case (fixed n VS. n,n+3,n+2,n+3,n+1, n+3.....) , and reference frame=1 makes the rough prediction. Therefore, “cbf=0” and “skip mode is the best mode” cannot happen easily especially when QP=22. So, the time saving has room to improve. However, the R-D performance of ECU and CFM is much better than that of our proposed CU-correlation algorithm.

5.2.3 Combined Fast CU Size Decision with ECU/CFM

Due to the experiments in section 5.2.2, the performance of ECU and CFM in our setting is good for time saving with negligible coding loss. Hence, we should try to combine our algorithm with them to get more acceleration. The experiment turns on ECU, CFM, and our proposed algorithm with the same testing conditions as Table 25, and the result is shown in Table 26.

Table 26 Simulation result of ECU, CFM, and our proposed algorithm

Both ECU and CFM	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-45.66%	-41.67%	-39.92%	-40.69%	-44.70%	-54.95%	-54.25%	-49.10%	-46.37%
Time-Saving(QP27)	-50.48%	-47.67%	-48.59%	-48.07%	-43.75%	-66.03%	-63.97%	-60.38%	-53.62%
Time-Saving(QP32)	-52.62%	-54.30%	-57.05%	-53.40%	-51.91%	-74.88%	-71.82%	-68.99%	-60.62%
Time-Saving(QP37)	-52.47%	-61.91%	-61.21%	-58.17%	-65.98%	-80.45%	-76.58%	-75.57%	-66.54%
AVG. Time-Saving	-50.31%	-51.39%	-51.69%	-50.08%	-51.59%	-69.08%	-66.66%	-63.51%	-56.79%
Y BD-rate	3.495	3.619	3.198	3.833	1.637	3.001	4.164	3.215	3.270
Y BD-PSNR	-0.095	-0.111	-0.068	-0.072	-0.046	-0.072	-0.124	-0.072	-0.083

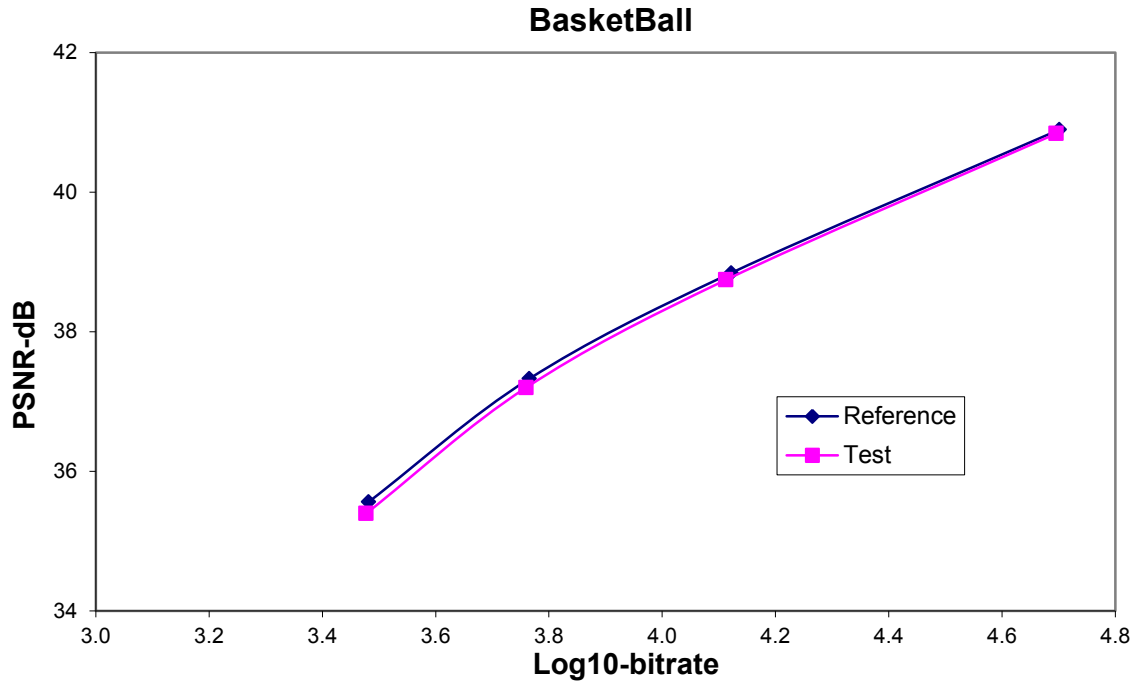


Fig. 34 R-D curve of Basketball in Table 26

The performance reduction of mixed algorithms also occurs in this case. Although the time reduction reaches about 57%, the BD rate also increases, too. We observe the R-D curve of Basketball in Fig. 34 to find a way to solve this problem. When QP becomes larger, the R-D curves separate far as illustrated in Fig. 34. The coding loss mainly comes from the low rate regions. Thus, we turn off our algorithm when QP is larger than 29. We take 16 sequences with 100 frames per sequence to test the adaptively combined algorithm, and the results are listed in Table 27. As QP is smaller than 30, the encoder adopts ECU, CFM, and our fast decision method. On the other hand, we only use ECU and CFM to accelerate encoding procedure to avoid excessive coding loss when QP is larger than 29.

Table 27 Results of the adaptively combined fast algorithm with ECU and CFM

Test Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4	AVG.
Time-Saving(QP22)	-45.59%	-42.25%	-40.14%	-40.65%	-44.52%	-58.36%	-54.96%	-48.41%	-46.86%
Time-Saving(QP27)	-50.53%	-47.33%	-48.88%	-50.61%	-43.30%	-68.43%	-64.46%	-59.14%	-54.09%
Time-Saving(QP32)	-22.61%	-36.59%	-35.74%	-35.54%	-32.02%	-71.13%	-65.00%	-59.22%	-44.73%
Time-Saving(QP37)	-32.97%	-51.83%	-45.79%	-46.35%	-57.63%	-78.84%	-73.36%	-70.50%	-57.16%
AVG. Time-Saving	-37.93%	-44.50%	-42.64%	-43.29%	-44.37%	-69.19%	-64.45%	-59.32%	-50.71%
Y BD-rate	1.791	2.197	2.291	2.432	1.332	1.582	2.671	1.861	2.020
Y BD-PSNR	-0.056	-0.070	-0.054	-0.055	-0.032	-0.050	-0.088	-0.049	-0.057

Test Sequence	BallDrill	BQMall	Party	HorsesC	BallPass	Bubbles	BQSquare	Horses	AVG.
Time-Saving(QP22)	-40.29%	-39.68%	-37.99%	-35.70%	-35.54%	-29.06%	-30.32%	-29.16%	-34.72%
Time-Saving(QP27)	-41.76%	-41.06%	-37.03%	-35.81%	-38.63%	-29.94%	-31.85%	-29.02%	-35.64%
Time-Saving(QP32)	-28.60%	-26.80%	-11.21%	-11.82%	-33.03%	-18.83%	-17.47%	-11.37%	-19.89%
Time-Saving(QP37)	-40.02%	-38.83%	-27.58%	-23.17%	-42.47%	-34.43%	-35.71%	-20.17%	-32.80%
AVG. Time-Saving	-37.67%	-36.59%	-28.45%	-26.63%	-37.42%	-28.07%	-28.84%	-22.43%	-30.76%
Y BD-rate (%)	2.589	1.909	0.498	0.947	1.688	1.513	0.431	1.028	1.325
Y BD-PSNR (dB)	-0.098	-0.080	-0.023	-0.039	-0.082	-0.061	-0.019	-0.051	-0.057

From Table 27, our proposed algorithm improves the time saving efficiency in the low QP region because the principle of our fast decision method is not using cbf and the skip mode to decide the early termination scheme. Moreover, we can combine our algorithm with ECU and CFM without implementation conflict. In Table 28, we check the coding performance of our proposed method at QP= 22 for high resolution sequences, and the results show that the combined method is valuable in improving the time saving for the high bitrate applications.

In summary, the combined algorithm not only retains the coding efficiency at the low bitrate region but it also reduces computing time at the high bitrate region. On the average, it offers about 51% time reduction with the increment of BD-rate about 2.02% for high resolution sequences. In addition, it provides 31% time saving but adds the BD-rate 1.33% for low resolution sequences.

Table 28 R-D performance of our proposed algorithm (QP= 22)

Sequence	Kimono	Park	Cactus	Basketball	BQTerrace	Vidyo1	Vidyo3	Vidyo4
$\Delta PSNR$	-0.109	-0.029	0.004	-0.012	-0.001	-0.120	-0.121	-0.074
$\Delta BitRate$	-3.53%	-0.22%	0.49%	0.41%	0.11%	-1.57%	-1.56%	-0.42%
TS	-41.95%	-39.66%	-39.66%	-38.74%	-43.05%	-43.60%	-40.60%	-37.02%

Chapter 6 Combined MV and DCT Optimization for H.264/AVC Codec

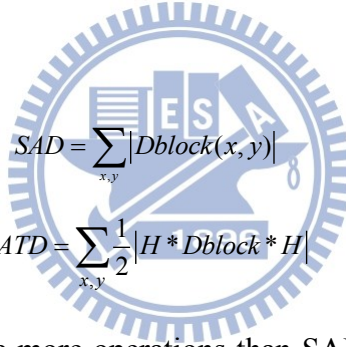
Although we introduce the process of encoding control in section 2.1.3, some details are described in this section. The experimental platform and the research topic are different from the previous chapters which are based on HEVC. Here, we use H.264/AVC encoder JM 18.0 [14] as the platform and we explore the effect of transform on ME in video coding. The chapter organization is as follows. Section 6.1 introduces the cost functions for MV searching in JM18.0 and the related work. Then, we design the algorithm to change the data flow concerning the problems mentioned in the related work in section 6.2. Finally, section 6.3 represents the experimental results and discussions.

6.1 MV Refinement with DCT result

A typical H.264 video encoder (such as JM) selects the best motion vector based on the sum of absolute difference (SAD) and the sum of absolute transformed difference (SATD) in the different accuracy layers to get the matching prediction block. Then, it uses the transform coding technique to encode the motion-compensated prediction errors. In baseline profile, a residual block is transformed by the 4x4 separable integer DCT (IDCT) or the 4x4 hadamard transform (H matrix) as shown in (14) which is an approximation form of IDCT [22] for low complexity.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (14)$$

In the integer ME, the distortion term in the motion R-D cost function (2) is decided by SAD as equation (15) where x and y are the pixel locations, and $Dblock$ is the difference block between the referenced candidate block and the original block. In the sub ME (searching the MV in the half and the quarter accuracy), the distortion term is calculated with SATD in (16) to get less transmitted frequency information for better compression efficiency.



$$SAD = \sum_{x,y} |Dblock(x,y)| \quad (15)$$

$$SATD = \sum_{x,y} \frac{1}{2} |H * Dblock * H| \quad (16)$$

Although SATD needs little more operations than SAD, the number of searching points in the sub ME is only nine points in each level, so the additional encoding complexity from hadamard transform is tolerable.

In [23], the effect of SATD on ME in different layers is discussed and tested. The encoder adopts SATD for searching integer MV directly, and averagely gets the 1.85% bitrate saving with increasing 781% encoding time as the sub-pixel motion search is unable. However, the same method brings little coding loss about 0.39% BD-rate [16] when sub MV is enabled. The reason is that SATD aims to match frequencies instead of residual pixels, so the interpolated

filter for the sub-pixel accuracy would bring the negative effect. The research in [23] is interesting, but there are two problems should be noted. First, the number of searching points with SATD is $(2 \times \text{search range} + 1)^2$ which brings too much complexity. Secondly, the experiment shows that SAD seems a better way to find MV in integer level. Therefore, we proposed our algorithm with concerning about the above problems in the next section.

6.2 Modified MV Selection Scheme

In this section, we describe the principle behind the proposed combined ME and DCT algorithm and its implementation step by step. In the traditional H.264/AVC encoder, the ME procedure chooses the integer vector that minimizes (2) with SAD consideration. However, (2) does not truly reflect the final distortion and the bit rate of encoded the coded block. Therefore, we include (1) into the ME procedure further in selecting MVs to improve coding performance. That is, we combine (1) and (2) in the integer ME procedure further.

The motivation is as follows. Although a selected MV is not the best candidate in the MV decision in the integral level, its residual DCT may have fewer large coefficients and thus produces fewer bits in the entropy coding in the final stage. Figs. 35–37 show the image examples. Fig. 35 shows the ten times magnified difference between the JM-encoded frame and our encoded frame using the proposed method, and QP is 22. In Figs. 36–37, we compare the residual MBs produced by two MVs on the second frame of the FOREMAN sequence.

The comparison is done in both the spatial domain and the frequency domain. Our proposed algorithm chooses a different quarter MV in the final stage (called Motion RDcost#2 means the 2nd best MV in the integer ME step). The resultant residual block has a more clustered frequency domain distribution; that is, the large magnitude coefficients are few and are close to each other as shown in Fig. 37 (Right). Therefore, these coefficients are easier to compress.

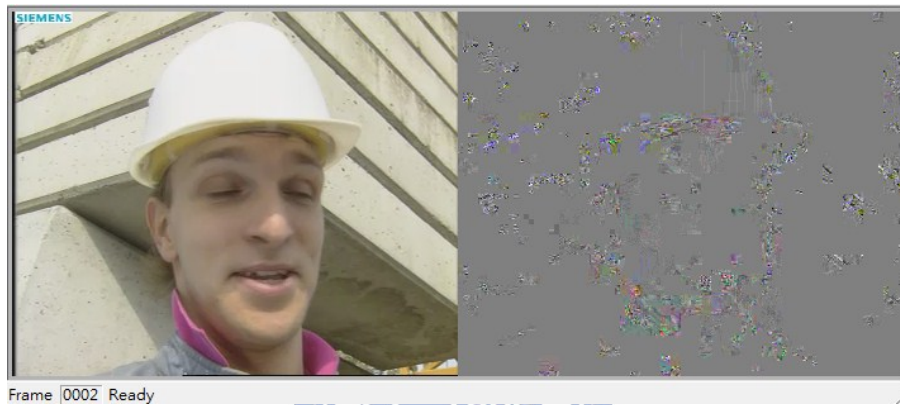


Fig. 35 Difference between the JM-encoded and our proposed method

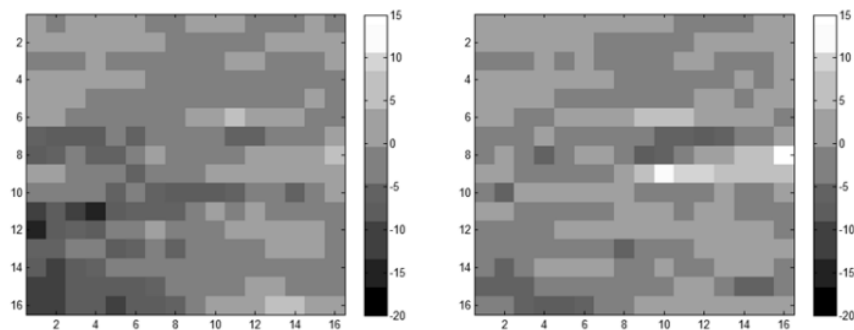


Fig. 36 Spatial domain: The residual MBs of Inter-16x16 mode on the second frame. The MB location (upper-left corner) is (80,160). Gray values are adjusted to show a range from 15 to -20 (the maximum and minimum pixel values). (Left) The residual block produced by the MV with Motion RDcost#1. (Right) The residual block produced by the MV with Motion RDcost#2.

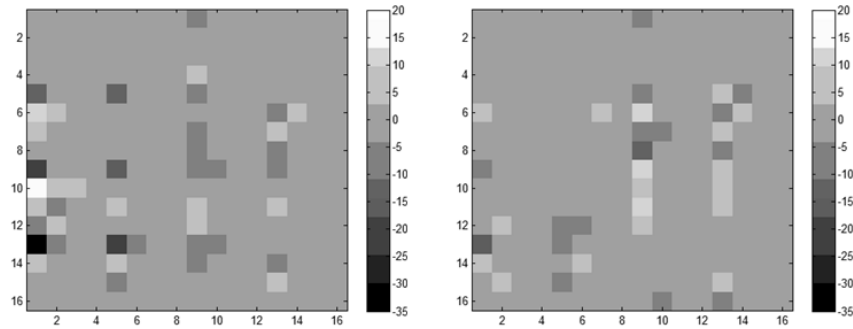


Fig. 37 Frequency domain: The transformed and quantized residual MBs of Fig.4. Coefficients are produced by 4x4 integral DCT with QP 22. Gray values are adjusted to show a range from 20 to -35. (Left) A residual transform block produced by the MV with Motion RDCost#1. (Right) A residual transform block produced by the MV with Motion RDCost#2.

The flowchart of the combined ME and DCT algorithm is to decide the best integer MV illustrated by Fig. 38. In the integral layer of ME procedure, our proposed method chooses the top five candidate MVs in the integral accuracy based on SAD, and then finds their corresponding half and quarter MVs using hadamard SAD. At the end, we use the modified function from the mode decision function to calculate the distortion based on hadamard again and estimate the bit rate. Therefore we choose the best integer MV with additional complexity from SATD about $5 \times [2 \times (\text{sub search points}) + 1]$ times for each integer MV searching. After our proposed scheme, we get the best integer vector of each partitioned block, and then take it to the following steps as the original JM, such as the sub-pixel ME and the mode decision.

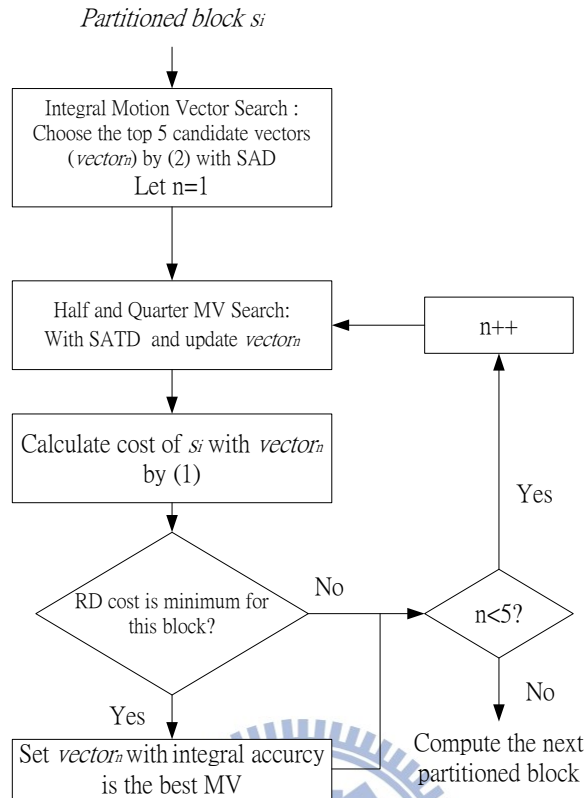


Fig. 38 Flowchart of the combined ME and DCT algorithm

6.3 AVC/H.264 Experimental Results and Discussions

To examine the effectiveness of our proposed motion estimation and DCT combined algorithm, we implement it on the software JM 18.0 [14], which is the reference software of the H.264/AVC encoder. We compare its performance with that of the original JM encoder. In the experiments, we use nine CIF sequences and four 4CIF sequences as already stated in Table 3 with a frame rate of 30 frame/sec: FOREMAN, BUS, FOOTBALL, MOBILE, NEWS, PARIS MOTHER_DAUGHTER, SILENT, ICE, CITY, SOCCER, HARBOUR, CREW [14].

In all experiments in this section, the number of encoded frames is 32 and I-frame period

is 16. We run four different QP values: 22, 27, 32, and 37. The search range is ± 32 , and the number of sub search points is nine. The previous frame is the reference frame. The Configuration is the baseline profile in JM18.0 with IPPP structure and CAVLC coding. It should be mentioned that the comparing JM setting of RDO is high complexity, and the MV search method is “fast full search”. For integer MV, motion cost in (2) is only decided by SAD and MV information. Then for half and quarter MV searching, the hadamard consideration is added to calculate the cost in (2) in JM18.0. The distortion of mode decision function is also calculated with SATD.

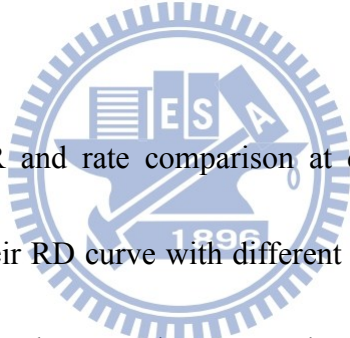


Table 29 shows the PSNR and rate comparison at different QP for the FOREMAN sequence, and Fig. 39 shows their RD curve with different QPs. We find that the curve has a larger gain in the high rate region because the 8x8 modes are used more often. In this case, because more MVs may be altered and because different MVs may result in different quantized residuals when QP is small, our coding gain becomes more obvious. This phenomenon happens also in the other sequences. Table 30 shows statistics of the chosen coding modes at different QP values. In general, a smaller QP produces fewer zero blocks, which leads to fewer skip modes. Therefore, our method would get benefits from a better MV choice.

Table 29 R-D Comparison for FOREMAN in P slices

FOREMAN	JM18		Proposed Method		BD-rate Y
	Y_PSNR (dB)	Bitrate (kbps)	Y_PSNR (dB)	Bitrate (kbps)	
QP=22	41.078	1121.89	41.115	1091.63	-3.4
QP=27	37.648	423.31	37.679	409.61	
QP=32	34.651	183.02	34.668	179.77	
QP=37	31.911	97.47	31.924	94.57	

Table 30 Modes and Motion Info Bits/Frame

FOREMAN	JM18		Proposed Method	
QP=22	Modes	MV_bits	Modes	MV_bits
16x16	2498	488.33	2205	426.93
16x8	1410	593.27	1320	557.47
8x16	1431	605.67	1423	569.27
8x8s	3449	4591.40	3965	5306.07
QP=27	Modes	MV_bits	Modes	MV_bits
16x16	2999	678.47	2958	662.47
16x8	1423	649.93	1391	631.27
8x16	1440	617.60	1488	643.67
8x8s	1760	2116.00	1880	2314.33
QP=32	Modes	MV_bits	Modes	MV_bits
16x16	3249	802.93	3307	794.40
16x8	1087	511.47	1104	498.40
8x16	1109	480.40	1157	493.60
8x8s	636	721.07	625	723.73
QP=37	Modes	MV_bits	Modes	MV_bits
16x16	2872	769.80	2901	748.27
16x8	721	309.40	713	303.60
8x16	635	252.93	637	260.33
8x8s	211	220.53	205	209.93

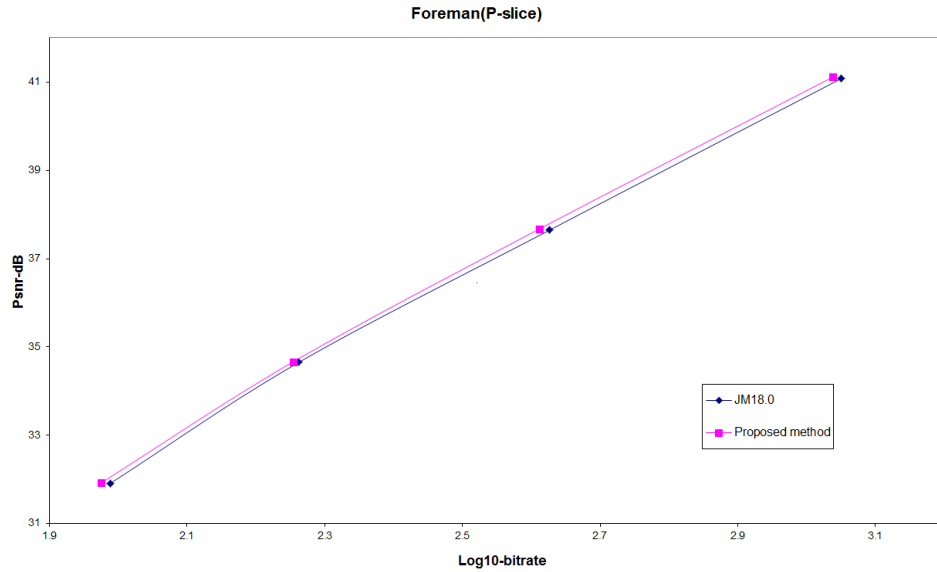


Fig. 39 R-D curve of Foreman for P slice

Table 31 shows the luma BD-rate [16] gain for all sequences. There are two sequences, MOTHER_DAUGHTER and SILENT, which have smaller gains at about 1% because these two videos have very little motion and thus the encoder frequently chooses the skip modes. Our MV selection scheme is applied only to the motion-compensated blocks, whose number is now small. Another factor affects the performance is image contents (patterns). In some sequences, such as CITY and MOBILE, our method provides more gain because they contain a number of fine edges, and thus our method has more chances to manipulate the residual distribution patterns. In summary, two factors seem to have major impact on our algorithm performance. One is the percentage of motion-compensated modes in P-slices, and the other one is the texture pattern of the residual blocks.

Table 31 BD Rate Improvement in P Slices of all Sequences

Sequence	Y BD-rate	Encoding Time	Sequence	Y BD-rate	Encoding Time
FOREMAN	-3.4	+43.2%	MOTHER_ DAUGHTER	-1.3	+41.2%
BUS	-2.6	+46.6%	SILENT	-1.1	+43.2%
FOOTBALL	-1.9	+49.6%	HARBOUR	-2.2	+47.0%
MOBILE	-2.4	+48.9%	CITY	-2.9	+45.9%
NEWS	-2.7	+43.0%	SOCCER	-1.8	+46.1%
ICE	-4.2	+39.8%	CREW	-1.7	+45.2%
PARIS	-1.6	+45.3%	Average	-2.3	+45.0%

We collect the final MV choices in our method in Table 32. It shows that the best motion R-D cost vector is chosen with higher probability when QP is large. In this case, because the number of transform coefficients is small, it thus makes little difference on the residual blocks produced by different MVs. On the average, the probability of choosing the fifth candidate MV is less than 5%. Thus, retaining more than five candidate MVs does not seem to offer much improvement. Finally, we may like to know how many “different” MVs in the integral level are chosen at the end using this approach (versus JM 18.0). We examine both the numbers of sub-blocks and their area. Table 33 shows the sub-block numbers and the area ratio of the changed MVs that are chosen by our algorithm.

Table 32 Final MV Choice from Candidate MVs (Percentages)

FOREMAN	QP=22	QP=27	QP=32	QP=37
Motion RDcost1	53.4%	56.7%	61.4%	67.3%
Motion RDcost2	21.8%	21.3%	19.9%	17.4%
Motion RDcost3	11.8%	10.5%	9.1%	7.5%
Motion RDcost4	7.6%	6.7%	5.7%	4.5%
Motion RDcost5	5.5%	4.7%	3.9%	3.3%
SILENT	QP=22	QP=27	QP=32	QP=37
Motion RDcost1	83.4%	84.0%	85.8%	89.1%
Motion RDcost2	7.8%	7.8%	7.0%	5.6%
Motion RDcost3	4.2%	4.0%	3.4%	2.5%
Motion RDcost4	2.7%	2.5%	2.2%	1.6%
Motion RDcost5	2.0%	1.7%	1.5%	1.2%

Table 33 Partitioned Sub-Blocks and the Area Ratio Using the Changed MVs

FOREMAN	Changed MV Blocks	Partitioned Blocks	Changed Area Ratio
QP=22	16223	36196	35.38%
QP=27	9739	23969	31.60%
QP=32	5780	17047	26.52%
QP=37	3559	14158	20.05%
SILENT	Changed MV Blocks	Partitioned Blocks	Changed Area Ratio
QP=22	4534	21115	9.27%
QP=27	3016	16681	8.93%
QP=32	1932	14066	7.92%
QP=37	1214	12743	6.62%

In summary, we propose a possible way to enhance R-D performance that further combines motion estimation and DCT for the H.264/AVC encoders. The algorithm considers the transform coding effect on choosing the best motion vectors from the integer to the quarter

accuracy. Based on the multiple sequences tests, we demonstrate that the proposed algorithm can achieve 2.3% bitrate saving averagely without changing the syntax of the standard AVC/H.264.

There is a trade-off between coding efficiency and time complexity. Although we reduce much SATD operations comparing to [23], the encoding time is still increased by about 45%.

To overcome the high complexity of our method, two properties can be introduced:

(a) There are still some redundant calculations in our program. For example, we should directly use the best sub MV instead of the best integer MV in Fig. 38 to the following encoding steps to save the operations from SATD. In addition, Some of 5 candidate MVs from the integral layer would have the same sub MV with repeated calculations.

(b) A parallel design should be feasible in hardware implementation because a data-independent loop exists in Fig. 38. Also, computing the cost in our proposed method for all candidate MVs can be executed in parallel.

Utilizing well the above properties, the encoding complexity of our proposed algorithm can decrease further. Acceleration of our scheme in software or hardware level is one of our future work items.

Chapter 7 Conclusions and Future Work

7.1 Conclusions

In this thesis, we design two algorithms for different goals. Thus, we conclude them in two parts.

In first part from Chapter 3 to Chapter 5, we study the computational complexity of building CU quadtree. Our fast CU size decision algorithm, which is based on the size information of the neighboring CUs and the co-located CU, speeds up the encoding procedure at about 1.75 times faster in average comparing to the original encoding process. Then, we also combine the existing ECU and CFM schemes together with our proposed algorithm in an efficient way. In the low QP cases, our algorithm provides more time reduction over ECU and CFM with acceptable coding loss. Totally, the combined fast algorithm offers averagely 51% time reduction, and the BD-rate increases at about 2.02% for high resolution videos. Our algorithm is also particularly useful in the low motion videos such as vidyo1, vidyo3, and vidyo4. This type of videos often occur in the mobile video communication, and the combined algorithm achieves up to about 69% time reduction with tolerable BD-PSNR drop about 0.05dB for the test sequences.

Chapter 6 is the second part: We study the effect of transform on motion vector selection. We propose the modified AVC/H.264 motion vector search process. First, we keep five

integer MV candidates by their SAD values, and then process the sub-pixel MV searching by using SATD. At the end, we use the modified AVC mode decision function to estimate the R-D cost to decide the best MV from five candidates. In comparing to the previous approach, our method not only reduces the time-consuming SATD calculations but also avoids the poor performance of using SATD directly in integer MV selection. In general, our proposed optimization scheme achieves 2.3% bit rate saving with an additional 45% encoding time, averagely. The method can achieve up to 4.2% BD-rate improvement in our test sequences, and the algorithm performs well especially for the sequences with strong residual texture.

7.2 Future Work

In proposing fast algorithms for HEVC, we design our algorithm under the configuration of low complexity and low delay P, but we change the encoding parameter setting in the GOP size and the number of reference frames. For real applications, we should consider the incremented QP to adjust the decision rule and the thresholds, adaptively. On the other hand, considering the MV offset and the multiple reference frames in the search of co-located CU will decrease the coding loss for our proposed fast algorithm. Last but not the least, we can include other indicators in reducing candidates. For example, cbf is an important indicator telling us whether the nearby CU partitions are reliable or not, especially for the termination decision in large size CU. Reducing the coding loss in the low bitrate case is a research

challenge.

Another topic is the R-D performance improvement for H.264/AVC. Its bottleneck is the high complexity. Therefore, we suggest some methods for speeding up the modified encoding procedure at the end of section 6.3. If we want to extend this combined ME and transform idea to HEVC, the scheme will be very complicated because HEVC has transform of different sizes. Also, because the current HEVC has very flexible ME modes and transform modes, the combined scheme may not provide much additional advantage.



REFERENCES

- [1] T. Wiegand et al., “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC),” ISO/IEC JTC/SC29/WG11 and ITU-T SG16 Q.6, JVT-Go50r1, Mar.2003.
- [2] H. M. Hang et al., “Towards the next video standard: high efficiency video coding,” *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2010.
- [3] B. Girod, “The Efficiency of Motion-compensating prediction for hybrid coding of video sequences,” *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 7, pp. 1140-1154, 1987.
- [4] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*, Wiley.
- [5] T. Wiegand, et. al., “Rate-constrained coder control and comparison of video coding standards,” *IEEE Trans. Circuit Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, Jul. 2003.
- [6] T.Wiegand and B. Girod, “Lagrange multiplier selection in hybrid video coder control,” *in Proc. Int. Conf. Image Proc.*, pp542–545, Oct. 2001.
- [7] T. Wiegand et al., “Special section on the joint call for proposals on High Efficiency Video Coding (HEVC) standardization,” *IEEE Trans. Circuits Syst. Video Technol.*, vol.

- 20, no. 12, pp. 1661–1666, 2010.
- [8] D. Marpe et al., “Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1676–1687, 2010.
- [9] JCT-VC, “High Efficiency Video Coding (HEVC) Test Model 5 (HM 5) Encoder Description”, JCT-VC document, JCTVC-G1102, January 2012.
- [10] P. A. Chou et al., “Optimal pruning with applications to tree-structured source coding and modeling,” *IEEE Trans. Inform. Theory*, vol. 35, no. 2, pp. 299–315, Mar. 1989.
- [11] G. J. Sullivan and R. L. Baker, “Efficient quadtree coding of images and video,” *IEEE Trans. Image Process.*, vol. 3, no. 3, pp. 327–331, May 1994.
- [12] JCT-VC, “WD5: Working Draft 5 of High-Efficiency Video Coding”, JCT-VC document, JCTVC-G1103, December 2012.
- [13] H.264/AVC codec [online] <http://iphome.hhi.de/suehring/tml/download/>.
- [14] Test sequences of H.264/AVC [Online] <http://media.xiph.org/video/derf/> .
- [15] HEVC codec [online] <http://hevc.kw.bbc.co.uk/trac/browser/tags/HM-5.0> .
- [16] G. Bjontegaard, “Calculation of Average PSNR Differences between RD-curves,” Document VCEG-M33, Apr. 2001.
- [17] J. Kim et al., “Adaptive coding unit early termination algorithm for HEVC,” *International Conference on Consumer Electronics*, pp. 261–262, 2012.

- [18] K. Choi et al., “Coding tree pruning based CU early termination,” JCT-VC document, JCTVC-F092, Jul. 2011.
- [19] R. H. Gweon et al., “Early termination of CU encoding to reduce HEVC complexity,” JCT-VC document, JCTVC-F045, Jul. 2011.
- [20] J. Leng et al., “Content based hierarchical fast coding unit decision algorithm for HEVC,” *International Conference on Multimedia and Signal Processing*, pp. 56–59, 2011.
- [21] G. Correa and L. Agostini, “Complexity control of high efficiency video encoders for power-constrained devices,” *IEEE Trans. on Consumer Electronics*, Vol. 57, No. 4, Nov. 2011.
- [22] H. S. Malvar et al., “Low complexity transform and quantization in H.264/AVC,” *IEEE Trans. Circuit Syst. Video Technol.*, vol. 13, no. 7, pp. 598–603, Jul. 2003.
- [23] A. Abdelazim et al., “Effect of the hadamard transform on motion estimation of different layers in video coding,” *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXVIII, Part5 Commission V Symposium, Newcastle upon Tyne, UK. 2010.