Fast Quadtree Structure Decision for HEVC Intra Coding Using Histogram Statistics

Yuchen Li[1], Yitong Liu[1], Hongwen Yang[1], Dacheng Yang[1]
[1]Wireless Theories and Technologies Lab (WT&T@BUPT)
Beijing University of Posts and Telecommunication
Beijing, 100876, P.R. China
[e-mail: liyuchen@bupt.edu.cn, liuyin212@gmail.com,
yanghong@bupt.edu.cn, yangdc@bupt.edu.cn]

*Received January 8, 2014; revised March 10, 2015; accepted April 19, 2015;
published May 31, 2015*

Abstract

The final draft of the latest video coding standard, High Efficiency Video Coding (HEVC), was approved in January 2013. The coding efficiency of HEVC surpasses its predecessor, H.264/MPEG-4 Advanced Video Coding (AVC), by using only half of the bitrate to encode the same sequence with similar quality. However, the complexity of HEVC is sharply increased compared to H.264/AVC. In this paper, a method is proposed to decrease the complexity of intra coding in HEVC. Early pruning and an early splitting strategy are applied to the quadtree structure of coding tree units (CTU) and residual quadtree (RQT). According to our experiment, when our method is applied to sequences from Class A to Class E, the coding time is decreased by 44% at the cost of a 1.08% Bjontegaard delta rate (BD-rate) increase on average.

Keywords: HEVC, Fast Intra Coding Algorithm, Histogram Statistics, Quadtree Structure

A preliminary version of this paper appeared in IEEE ICC 2009, June 14-18, Dresden, Germany. This version includes a concrete analysis and supporting implementation results on MICAz sensor nodes. This research was supported by a research grant from the IT R&D program of MKE/IITA, the Korean government [2005-Y-001-04, Development of Next Generation Security Technology]. We express our thanks to Dr. Richard Berke who checked our manuscript.

1. Introduction

The state-of-the-art video coding standard, High Efficiency Video Coding (HEVC), is the current joint video coding standardization project of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. HEVC is a typical hybrid video coder, which means it uses intra and inter frames to reduce spatial and temporal redundancy. However, almost all parts of the encoder have been improved compared to H.264/MPEG-4 Advanced Video Coding (H.264/AVC), including a new picture segment method, a finer intra searching scheme using 35 directional modes, a new sample adaptive offset (SAO) filter combined with the deblocking filter, and a temporal motion vector prediction scheme (TMVP). With these new features, HEVC doubles the compression ratio at the same level of video quality compared to H.264/AVC. According to the experimental results in [1], the high compression ratio of HEVC mainly benefits from the larger coding tree unit (CTU) size and transform unit (TU) size.

The CTU is the basic unit for performing further splitting and is similar to micro blocks (MB) in H.264/AVC. The size of the CTU is 64x64, 32x32, or 16x16 [2] according to the content of the video, where 64x64 is a typical setting for most of the testing sequences [2]. To extend the adaptation of the partition, a CTU is allowed to split into four equal pieces recursively until the smallest size of partition is reached, which is usually set to 8x8. The partitioning process leads to a quadtree CTU structure, and the leaf node of the quadtree is called the coding unit (CU). A CU is the basic unit for further splitting into prediction units (PU) and transform units (TU). To decide whether a CU needs to split into four sub-CUs, Rate-distortion Optimization (RDO) will be conducted for the current CU and its four sub-CUs, from which two costs will be obtained. One represents the condition of splitting the current CU. The other stands for no-split. HEVC will choose the way that leads to a smaller cost for the current CU.

For the residual coding, HEVC divides the CUs recursively into TUs, which is the basic unit for transform coding. The partitioning process is similar to what is done in CTU. The structure of the recursive partition in deciding TU size is called the residual quadtree (RQT). When the CTU structure is determined, the RQT will set each CU as the root node and search for the best TU partition at each depth. Typically, the max depth of RQT in intra coding is set to 3, and the TU size is set to 32x32, 16x16, 8x8, or 4x4 [2].

Although larger sizes of CTUs and TUs offer a great benefit in improving the coding efficiency, a heavy computation burden is presented by searching for the best quadtree pattern at each depth. An experiment in [1] shows that if the maximum size of a CU is limited to 16x16, then the encoding time will reduce to 58% of the origin time, and if the CTU size is 64x64 and MaxDepth is 3, the number of all possible CTU partitions is calculated by the following equation.

$$1 + 1 + P_4^4 + P_4^4 \times P_4^4 = 602 \tag{1}$$

The analysis above presents it is the quadtree structure in HEVC brings huge complexity. Therefore, it is attractive to eliminate some partitions that are unlikely to be chosen as the best partition pattern. In this paper, we focus on accelerating the search for the best quadtree structure for CTU and RQT in HEVC intra coding. A unified method is proposed, which we call the fast quadtree structure deciding method using histogram statistics. Using our method, the coding complexity of HEVC can be reduced, with a slight loss in video quality.

The rest of this paper is organized as follows. Section 2 summarizes and reviews the related works. The quadtree structure of the picture partition in HEVC is introduced in Section 3. Our proposed method is described in Section 4. The performance is demonstrated in Section 5, while Section 6 presents the conclusions of this paper.

2. Related Work

2.1 Fast algorithm for deciding the structure of CTU

There are some existing methods focused on fast algorithms in HEVC intra coding. A basic assumption is that a strong correlation exists between neighboring blocks in the space and time domains. Most fast algorithms make use of this prior information in different ways.

Jiang Wei et al. [3] proposed a gradient-based fast mode decision algorithm for HEVC intra coding. In this method, the gradient directions and histogram are used for CU size selection. The method achieves 20% complexity reduction with a 0.74% Bjontegaard delta rate (BD-rate) increase.

A group-based fast mode decision algorithm for intra coding [4] was proposed by Shunqing Yan, et al. The number of intra prediction modes for RDO was reduced by grouping the rough mode decision (RMD) modes into different rough directions. The experiment shows that coding time was reduced by 23.52% with a 1.3% BD-rate increase.

Shen Liquan, et al. [5] proposed a fast CU size decision algorithm for intra coding. Before codec searching for the best depth of a CU, a depth prediction is calculated using the depth of nearby CUs (Left, Left-Up, Up, Right-Up) with a series of weighted factors. Then, the algorithm searches specific depths based on the predicted depth. According to the experimental result, the proposed method approaches 21.1% time savings with a 1.74% BD-rate increase on average, compared to [6].

Shen Xiaolin et al. [7] proposed a CU splitting early termination method based on a weighted support vector machine (SVM), which is different from the above papers. The method chooses several features that are highly correlative to whether or not to split a CU, and then, the selected features are used to build a good predictor of SVM, which is applied before performing RDO on four sub-CUs.

Another paper with similar thoughts on classification is [8]. The authors assume the RDO cost of a CU of the same size to follow a Gaussian distribution whose parameters (μ,σ) are estimated in an online learning process. The paper reports 50.2% time savings with a 0.6% BD-rate increase compared to HM6.0. However, the robustness of this method is not good, which will be analyzed in Part 4.

Some early works on fast CU size decision algorithms such as [9], [3] have been implemented in HM4.0 and HM6.0, leading to 37% and 19% time savings, respectively, with little loss of coding efficiency.

Furthermore, a few conventional fast algorithms have been proposed based on the H.264/AVC encoder. The work in [10] makes use of a non-normalized Haar transform for edge detection. In [11], the sum of approximate squares difference is utilized to accelerate the H.264 intra mode decision. The work in [12] notes and utilizes the motion homogeneity of video to skip some unlikely modes.

2.2 Fast algorithm of deciding the structure of RQT

Fewer fast algorithms focus on the residual quadtree than on CTU. Su-Wei Teng et al. proposed a fast mode decision for RQT [13]. In this method, the order of searching different

TU sizes is rearranged. A TU size of 16x16 has the highest priority to be selected, followed by 8x8 and then by 32x32. The authors set two early termination conditions to accelerate the process of searching for the best partition of RQT. As a result, the algorithm achieves up to 55% encoding time reduction with negligible coding loss.

The author of [14] makes observations on the impact of maximum inter RQT depth for different CU sizes and sets different fixed depths for sizes of 64x64 and 8x8 TU. For 32x32 and 16x16 TUs, two discriminant scores will be calculated to decide the depth of the current TU. The algorithm gains a 7.2% speedup in a random access high efficiency (RAHE) configuration and a 21.1% speedup in a low-delay configuration with only 0.03~0.035 dB PSNR degradation.

Another early TU decision method was proposed by Kiho Choi and Euee S. Jang in [15]. The relationship between the determined TU size and the nonzero discrete cosine transform coefficients (NDCs) was exploited. If the NDC is less than the threshold, which is set according to the experiment, the TU will halt further RDO cost evaluation. The method was implemented on HM 3.0 and gained 61% complexity reduction with 0.58% bitrate increasing and 0.02 PSNR loss.

In the previous draft of HEVC, RQT structure was grouped into two categories: square residual quadtree and nonsquare residual quadtree. In [16], the nonsquare residual quadtree structure was finally excluded. However, the results of [13, 14], and [15] are based on the early version of the HM. The effects of these algorithms have not been verified in the most recent test model.

3. Quadtree Structure of Block Partition in HEVC

Fig. 1 shows the CU and TU partitioning with quadtree structure. As seen in **Fig. 1**, the partition is started by a Largest CU (LCU) whose size is typically set to 64x64. During the partitioning process, if the size of the current CU is larger than the smallest CU size, which is

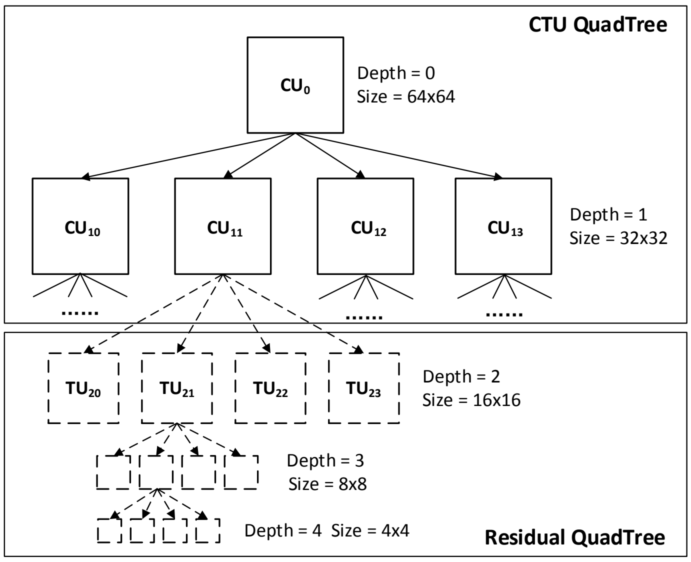

Fig. 1. CU and TU partition in HEVC with quadtree structure. The solid lines represents the CU partition, and the dotted lines represent the TU partition.

usually set to 8x8, the RDO cost of the current CU and four sub-CUs is calculated and compared to decide whether to split into sub-CUs. The RDO cost is calculated using the following Lagrangian functional:

$$J_{MODE} = D_{rec} + \lambda_{MODE} R_{rec} \tag{2}$$

where, for instance, D_{rec} is the sum of squared differences (SSD) between the original block and its reconstruction. R_{rec} is the number of bits used by compressing the block. If a CU is split into four sub-CUs like the CU_0 in **Fig. 1**, this CU will not be a leaf node of the final quadtree. In contrast, if a CU stops splitting at its position like CU_{11}, it will become a leaf node of the CTU quadtree.

To reduce the complexity of RDO in intra coding, Yinji Piao in [17] proposed the Rough Mode Search (RMS). In this method, several suboptimal candidates are selected by sorting $J_{RMS,}$ which is calculated by (3).

$$J_{RMS} = D_{HSAD} + \lambda_{MODE} R_{rec} \tag{3}$$

In (3), D_{HSAD} is the minimum absolute sum of the Hadamard Transformed coefficients of the residual signal (HSAD), and R_{rec} is the mode bit. As a result, the smaller J_{RMS} is, the more likely it is that the mode will be the optimal result. After several suboptimal candidates are identified, RDO searching is performed among the candidates selected. In the following content, J_{MODE} and J_{RMS} are called COST together.

For the residual quadtree, the process of deciding the structure of RQT is similar to the one for CTU. Typically, the TU size is from 32x32 to 4x4. When the CU size is determined, it will be further split into TUs, as illustrated in **Fig. 1**. The criterion of whether to split is the value of J_{MODE}, which means the structure with smallest J_{MODE} will be chosen as the best partition of RQT.

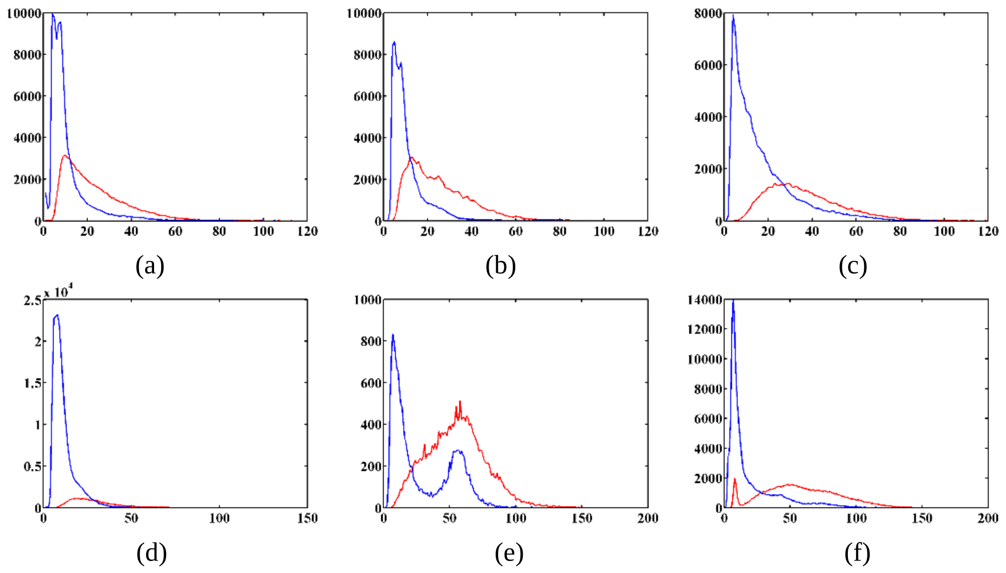

Fig. 2. Statistics on COST and distribution of choice of splitting or not: (a), (b) come from the J_{RMS} results and (c),(d),(e),(f) from the J_{mode} results. All of the data come from CU splitting, and the CU size is 16x16.

4. Fast Pattern Decision Based on Histogram Statistics

4.1 Motivation

The motivation of our method comes from [8], whose model is built on the assumption that COST follows a Gaussian distribution. We repeated the work of [8] and performed statistical analysis of the data on the relationship between COST and the choice of splitting. Some of the results are shown in **Fig. 2**. The blue curve represents non-split CU cases, and the red curve represents split CU cases. The X-axis shows the value of the COST in a particular interval length, while the Y-axis shows the count of the different cases. The data in (a), (b) come from the J_{RMS} results, and those in (c), (d), (e), (f) come from the J_{MODE} results. All of the data are collected in different videos using the HEVC common test condition. [18] The statistical length of the interval of J_{RMS} is 200, and the length of the interval for J_{MODE} is 50. Close inspection of the above image will reveal that (a), (b), (c), (d) are likely to follow the Gaussian distribution, but (e) and (f) are not a perfect approximation. To solve this problem, we build another more precise model to predict the split probability at any COST value.

4.2 Fast Pattern Decision Based on Histogram Statistics

To overcome the weakness of the method in [8], our method divides the value of COST into several intervals. The probability of splitting a node, whether it is a CU or TU, into four sub-nodes in the quadtree is calculated in each interval. Thus, the following CU/TU can make use of the results learned by previous statistics. To demonstrate our method, an experiment is performed, and the result is shown in **Fig. 3**. In this experiment, the first 30 frames of *BQTerrace* are encoded with the original HEVC intra coding algorithm. The RDO cost of every CU at depth level 2 is collected. There are two kinds of action for every CU after RDO, splitting or not splitting. We separate the RDO costs collected into two parts by these two situations. In **Fig. 3**, the blue bar shows the probability density of not splitting a CU into sub-CUs in each statistical interval. The red bar shows the probability density of splitting a CU. The dotted green line shows the splitting probability of a CU whose COST falls into a specific

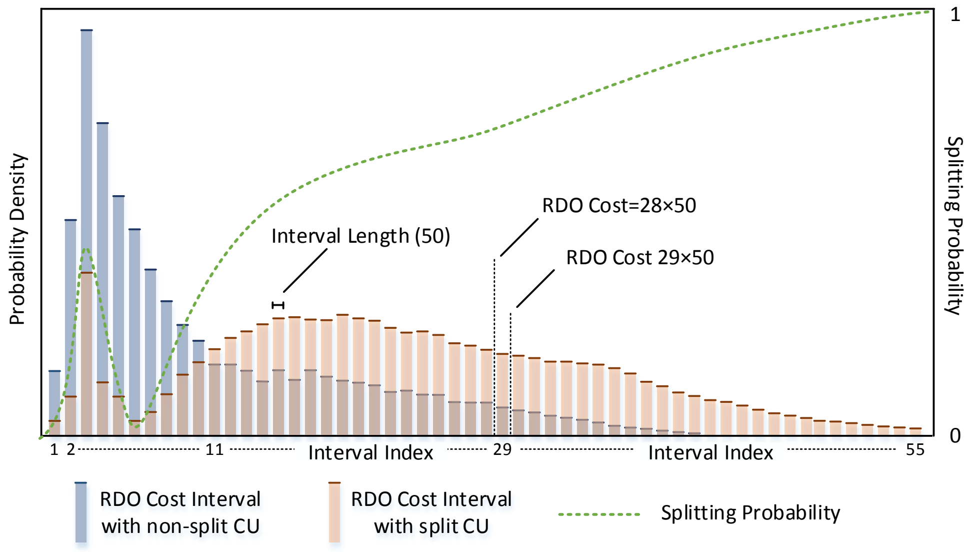

Fig. 3. An example of isometric interval splitting. The length of intervals is 200. The dotted green line is the splitting probability of CU in each interval.

interval. The probability is calculated by equation (4)

$$p_{split}(i) = \frac{C_{split}(i)}{C_{split}(i)+N_{no-split}(i)} \tag{4}$$

where $C_{split}(i)$ is the count of split CUs in interval i, and $C_{no-split}(i)$ is the count of non-split CUs. With this method, the green curve precisely reflects the change in probability with the growth of J_{MODE}. Notice that "probability of splitting" means the probability of splitting one node into four sub-nodes. The node may be a CU or TU according to the structure on which the method is applied.

For implementation, there are two defects in the algorithm above. Because all intervals are equally spaced, the total number of the intervals will not be determined at the start of the encoding process, which will lead to low efficiency in coding performance. Second, the estimation of probability for CUs with large COST is not accurate because the number of samples with large values is small. It takes too long to collect enough samples to make predictions for future CUs.

In our modified method, COST is separated into three segments, as illustrated in **Fig. 4**. The segment partition is defined by (5)

$$COST \in \begin{cases} Seg.I & 0 < COST \le Th1 \\ Seg.II & Th1 < COST \le Th2 \\ Seg.III & Th2 < COST \end{cases} \tag{5}$$

Table 1. Values of *Th1*, *Th2*, *Length1*, *Length2* for Acceleration Algorithm in deciding the CTU structure

	CU size	Th1	Th2	Length1	Length2
J_{RMS}	**16x16**	8000	16000	200	500
	32x32	40000	60000	2000	5000
	64x64	70000	150000	3500	20000
J_{MODE}	**16x16**	16000	48000	400	800
	32x32	60000	180000	1500	5000
	64x64	120000	360000	3000	10000

Table 2. Values of *Th1*, *Th2*, *Length1*, *Length2* for Acceleration Algorithm in deciding the RQT structure

	TU size	Th1	Th2	Length1	Length2
J_{MODE}	**8x8**	5000	15000	200	1000
	16x16	20000	60000	1000	2000
	32x32	80000	240000	4000	8000

In Segments I and II, the length of the intervals is set differently. In most cases, samples in Segment I vary more drastically than samples in Segment II, and fewer samples belong to Segment II than to Segment I. According to the fact observed above, we set *Length1* smaller than *Length2*, which ensures a more accurate probability estimation. In Segment III, all of the samples are counted in one interval because the number of samples in this segment is much smaller than in the other two segments. Another reason for using only one interval is that the trend of the rate between splitting and non-splitting varies slowly in this section. Almost all of the CUs/TUs in this interval will choose splitting as the best strategy. The values of *Th1*, *Th2*, *Length1*, and *Length2* for different CU and TU sizes are listed in **Table 1** and **Table 2**, which are our empirical values.

To predict whether a node in the quadtree needs to be split into four sub-nodes, the probability of splitting must be known. In our proposed method, the algorithm is divided into two parts, the probability-estimating stage and the predicting stage. During the overall encoding process, an estimating-predicting flag (EPF) is set for every statistical interval. When EPF is E, the correlative interval is marked as the estimating stage; otherwise, when EPF is P, the interval is marked as the predicting stage. At the beginning of the encoding, all of the EPFs are set to E.

When the estimating stage of an interval begins, the current node, which is probably CU or TU, undergoes RDO, and the same process operates for its four sub-nodes if the CU/TU does not reach its MaxDepth. After the partition is finally determined, the depth and the COST of each level will be stored in a container that is set at the beginning of the estimating stage. The size of container S_{learn} is a crucial parameter. When a small size is set, the probability estimate will be inaccurate with a small number of samples. However, if the size is too large, because video content is changing constantly, a large statistical bias may exist between the learning samples and the predicting samples. Furthermore, if too many samples are learned, encoding efficiency will be lost. In our experiment, S_{learn} is set to 50 empirically. Once the number of learned samples in an interval reaches the size of the container, the probability of splitting for this interval will be calculated using equation (4). The probability will be stored in a probability container. Then, the stage of this interval is set to P.

When the predicting stage for an interval begins, the algorithm will perform the accelerating algorithm to decide the CTU and RQT structure according to the estimated probability of splitting. In this stage, another parameter m is used to control the number of CUs/TUs, which is under the predicting stage in the interval. In one predicting stage, the upper limit of predicted CUs/TUs, $S_{predict}$, is defined as follows.

$$S_{predict} = S_{learn} \times m \tag{6}$$

If the count reaches the upper limit, the stage of the interval is set to E, and the probability of this interval is reset to wait for the next update.

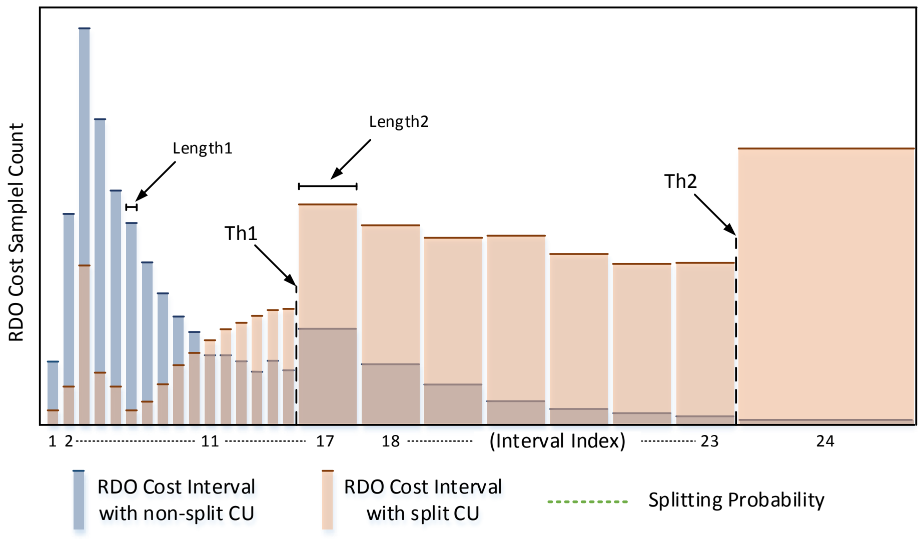

Fig. 4. Splitting ProbabilityStatistics using histogram.

4.3 Early splitting and early pruning in deciding the CTU and RQT structure in intra coding

In this part, the above method will be used to accelerate searching for the best CTU structure in intra coding. As mentioned in Part 3, whether a CU needs to be split into four sub-CUs is determined by J_{MODE} of the current CU and sub-CUs. The choice that leads to a smaller J_{MODE} will be chosen as the strategy for the current CU. Thus, an assumption is made that CUs with small values of J_{MODE} tend not to be split, while CUs with large values of J_{MODE} tend to be split. Based on this assumption, we design early pruning (EP) and early splitting (ES) algorithms to decide the structure of CTU for intra coding.

EP is the process that skips calculating the J_{MODE} of four sub-CUs and makes the current CU the leaf node of the CTU quadtree when the splitting probability of the current CU is below the threshold α. Coding complexity will be reduced because RDO will not be performed on part of the partition.

ES is slightly more complicated than EP. In our proposed method, the statistical analysis is applied on both J_{MODE} and J_{RMS}. The splitting probability is estimated on these two kinds of COST. When the stage of the interval is P, two splitting probabilities exist. After calculating the J_{RMS} of the CU, the proposed method will look for the splitting probability derived from the statistics for J_{RMS}. If the probability is larger than the threshold β, the algorithm will skip computing J_{MODE} for the current CU and set J_{MODE} Max directly. When the sum of the four J_{MODE} values of sub-CUs is computed, the codec will definitely choose the split as the best choice for the current CU.

In the predicting stage, if the current CU does not perform either ES or EP, the full RDO process will be performed. When the encoding process of the current CU goes into the deeper level, the ES and EP test will be continued until the size of the current CU is 8x8.

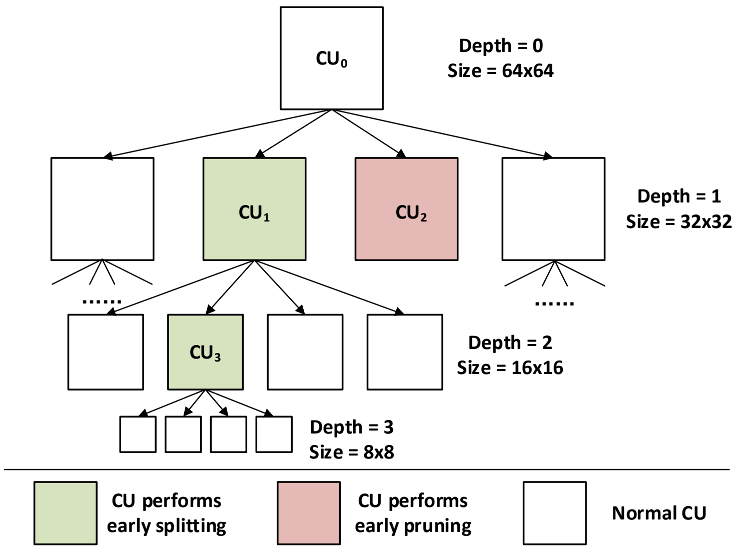

Fig. 5. An example of CU partitioning. LCU size: 64x64 MaxDepth: 3

However, there exists an unsolved problem in the ES algorithm. Consider the situation illustrated in **Fig. 5**. In this figure, CU$_1$ is selected to perform early splitting when the smallest J_{RMS} is calculated. The calculation for J_{MODE} of CU$_1$ is skipped and set to Max to perform ES. CU$_0$ is a normal CU for which no acceleration algorithm is loaded. It is unreasonable to compare the J_{MODE} of CU_0 and the sum of the J_{MODE} values of four sub-CUs because the J_{MODE} of CU_1 is Max. The same situation recurs in CU_1 and CU_3. To solve this problem, we defined two kinds of J_{MODE} for a CU: $J_{MODE\text{-}split}$ and $J_{MODE\text{-}pruning}$. When a CU is determined to perform ES, $J_{MODE\text{-}split}$ is set to Max, and $J_{MODE\text{-}pruning}$ is set to the sum of the $J_{MODE\text{-}pruning}$ values of the four sub-CUs. In a normal CU or a CU that performs early pruning, $J_{MODE\text{-}split}$ and $J_{MODE\text{-}pruning}$ are set to the original J_{MODE}. Because all of the CUs in depth 3 are normal CUs, for which $J_{MODE\text{-}pruning}$ is the original J_{MODE}, every $J_{MODE\text{-}pruning}$ of CU in depth 2 and depth 1 can be calculated recursively. When a CU that has performed ES is used as one of the four sub-CUs to calculate the sum of the J_{MODE} of the four, $J_{MODE\text{-}pruning}$ is used instead of $J_{MODE\text{-}split}$ to calculate the sum.

The proposed method is also applied to searching for the best partition for RQT. The structure of RQT is an extension of the CTU, as shown in **Fig. 1**. The root node of the RQT is a leaf node of the CTU quadtree. When deciding whether a TU needs to be split, the J_{MODE} values of the current TU and four sub-TUs are compared, as in CU splitting. The assumption that CUs with large COST values tend to be split and vice versa can be generalized in RQT. Early pruning will be performed based on the assumption that the probability of splitting, which comes from the statistics for J_{MODE} of TU, is smaller than the threshold γ. If EP is performed, the current TU will skip partitioning in the deeper level and set the TU of the leaf node of the RQT.

4.4 The total algorithm

The flow chart of the full algorithm is shown in **Fig. 6**. The left part shows the flow in the CU level, and the right part shows the process in the TU level.

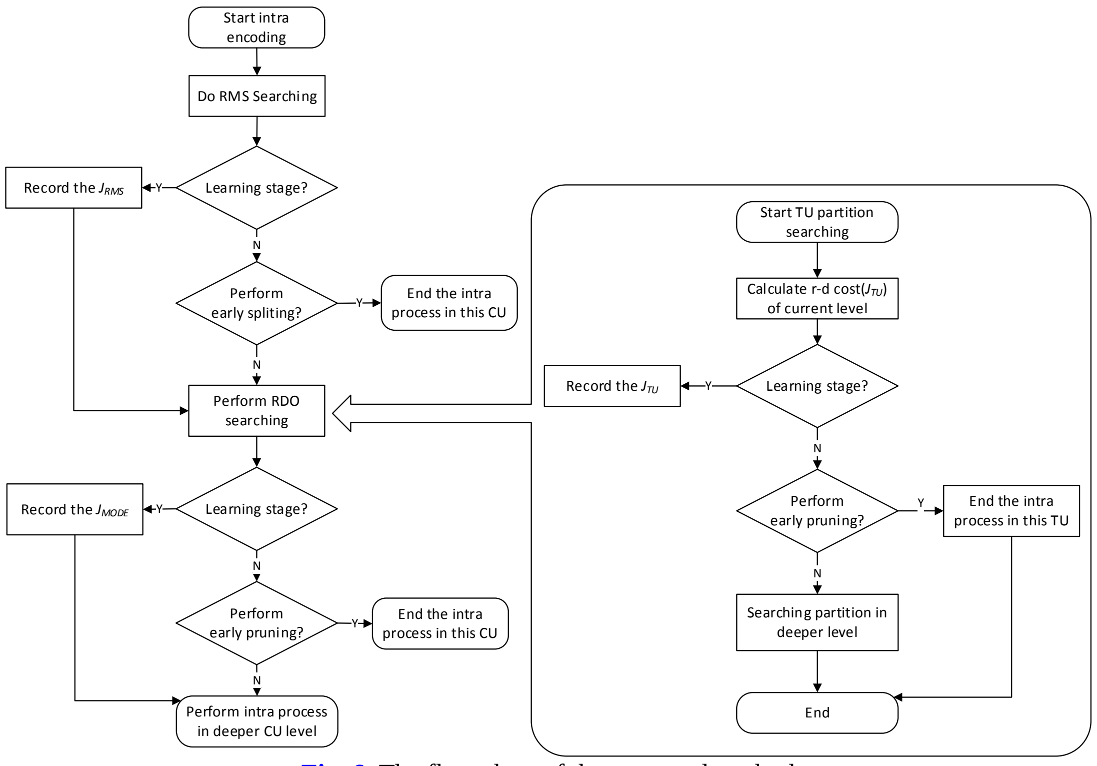
Start intra encoding
Do RMS Searching
Learning stage?
Y
Record the J_{RMS}
N
Perform early spliting?
Y
End the intra process in this CU
N
Perform RDO searching
Learning stage?
Y
Record the J_{MODE}
N
Perform early pruning?
Y
End the intra process in this CU
N
Perform intra process in deeper CU level
Start TU partition searching
Calculate r-d cost(J_{TU}) of current level
Learning stage?
Y
Record the J_{TU}
N
Perform early pruning?
Y
End the intra process in this TU
N
Searching partition in deeper level
End

Fig. 6. The flow chart of the proposed method

Table 3. The main configuration parameters

Parameters	Value
MaxCUSize	64x64
MaxCUDepth	4
MaxTUSize	32x32
MinTUSize	4x4
MaxTUDepth	3
QP	22, 27, 32, 37

5. Simulation Results and Analysis

The proposed method is implemented based on HM10. The default algorithm in HM10 and the main profile are used as a comparison. The test condition strictly follows the common test conditions defined in [18]. The main parameters of the intra configuration of the HEVC main profile are listed in **Table 3**. S_{learn}, which is mentioned in (6), is set to 50, and *m* is set the frame rate of sequence.

Fig. 7 shows the performance results of different methods of accelerating, including early pruning of CU, early splitting of CU and early pruning of TU. Three sequences (*ParkScene*, *BasketballDrill*, and *BasketballPass)* are used in this test, which come from Class B to Class D. The data in **Fig. 7** are the average results of the three sequences. The different thresholds are set to observe how the RD performance changes and at what level the most time will be saved.

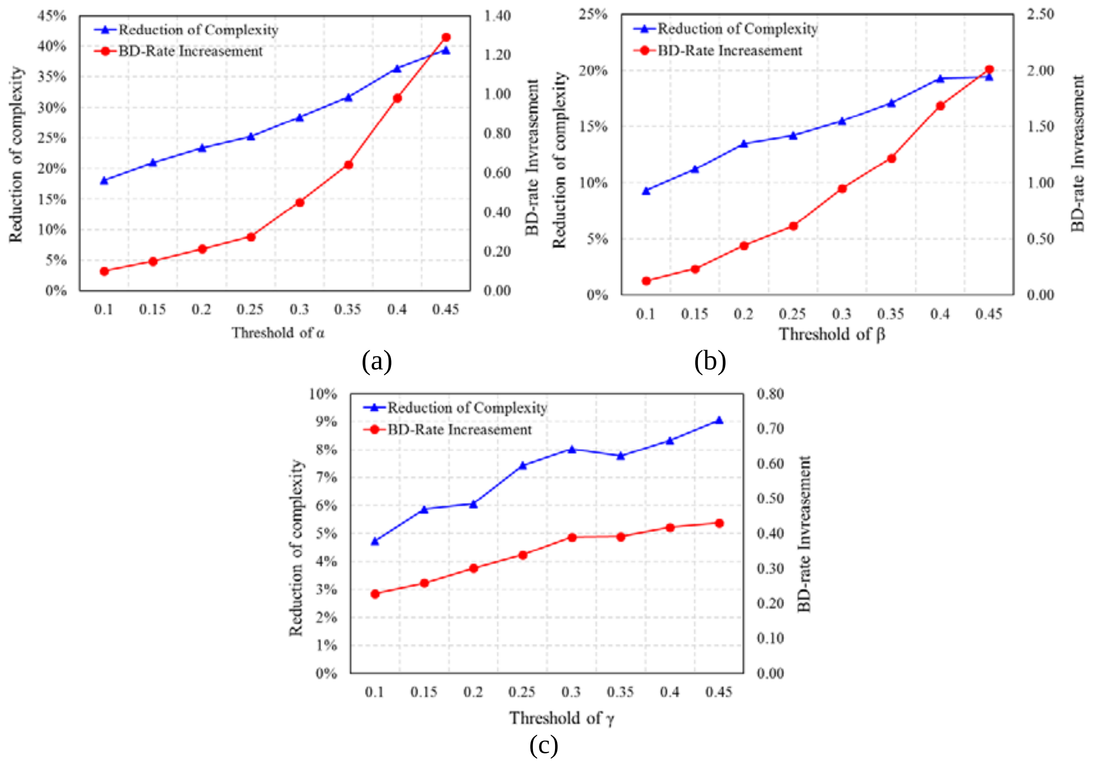

Fig. 7. Performance comparison of different threshold values.
In (a), only early CU pruning is performed. In (b), early splitting is performed, and in (c), early TU pruning is performed.

Table 4. Summary results of the proposed method for sequences in Class A to Class E: α, β, γ are set to 0.25, 0.8, and 0.2.

Class & Sequences		ΔTime	Ave.	ΔBD-rate	Ave.
A	NebutaFestival	73.8%	60.6%	0.52%	1.25%
	PeopleOnStreet	35.4%		0.97%	
	SteamLocomotiveTrain	72.6%		2.33%	
	Traffic	41.3%		1.17%	
B	BasketballDrive	56.6%	49.7%	1.60%	1.47%
	BQTerrace	38.9%		1.10%	
	Cactus	40.8%		1.10%	
	Kimono	74.7%		2.89%	
	ParkScene	37.4%		0.65%	
C	BasketballDrill	33.0%	33.0%	0.77%	0.67%
	BQMall	34.7%		0.90%	
	PartyScene	31.6%		0.33%	
	RaceHorsesC	32.7%		0.66%	
D	BasketballPass	38.4%	31.9%	0.84%	0.50%
	BlowingBubbles	30.5%		0.23%	
	BQSquare	31.0%		0.43%	
	RaceHorses	27.9%		0.52%	
E	FourPeople	40.6%	49.8%	1.21%	1.53%
	Johnny	56.2%		1.93%	
	KristenAndSara	52.6%		1.45%	
	Total		44.0%		1.08%

The reduction of complexity will be calculated by equation (7).

$$\Delta Time\% = \frac{(T_{HM10} - T_{prop.})}{T_{HM10}} \tag{7}$$

In **Fig. 7**, early pruning of CU turns out to be the best accelerating method because it obtains the greatest time saving with the smallest BD-rate increase compared to the other two methods. Early TU pruning provides consistent performance because with the increasing of the threshold, little change occurs in the reduction of complexity and BD-rate increment.

The overall results of the algorithm in intra coding, including EP of CU, ES of CU, and EP of TU, are listed in **Table 4**. The thresholds of α, β, γ are set to 0.25, 0.8, and 0.2. The reduction of complexity is represented by the increase in BD-rate. A trend can be obtained showing that the performance of the proposed method improves with an increase in the size of

Table 5. Performance comparison between other fast algorithms for Intra coding and our proposed method

	[3]		[4]		[19]		Proposed Method	
	ΔR	ΔT	ΔR	ΔT	ΔR	ΔT	ΔR	ΔT
Class A	0.6%	20%	1.0%	22%	2.8%	55%	1.2%	60%
Class B	0.7%	25%	0.8%	21%	3.1%	56%	1.4%	49%
Class C	0.7%	19%	1.5%	25%	1.6%	56%	0.6%	33%
Class D	0.9%	20%	1.7%	24%	1.4%	53%	0.5%	31%
Class E	0.9%	19%	1.3%	23%	3.5%	60%	1.5%	49%
Ave.	0.76%	20.6%	1.26%	23.0%	2.48%	56.0%	1.04%	44.4%

the sequences. When compressing sequences in Class D, the time is reduced by 31.9% while the BD-rate increase is 0.71%. However, when compressing the sequence in Class A, the proposed method achieves 60.6% time savings with a BD-rate increase of 1.75%, which is slightly higher than the previous result.

The performance of our algorithm surpassed the previous works [3, 4, 19] in different aspects. The performance comparison is shown in **Table 5**. ΔR represents the BD-rate increase, and ΔT is the reduction in complexity. Among the results, Jiang's work [3] has the lowest BD-rate increase; however, its performance is the worst, with only a 20.6% reduction in complexity. Sun Heming's work [19] has the best time performance at the cost of a high BD-rate increase. On average, the method in [3] provides 20.6%/0.76=27% encoding time reduction per 1% BD-rate increase. The corresponding values for [4, 19] and our proposed method are 18.25%, 22.58%, and 42.69%, respectively, which clearly shows that our proposed method outperforms the other methods.

6. Conclusion

In this paper, we have proposed a fast quadtree structure deciding method for intra coding of HEVC. The overall algorithm is separated into two parts, the learning stage and the predicting stage. In the learning stage, a probability model between splitting the node and the COST of CU/TU is constructed. When performing a non-optimized intra coding algorithm, the COST of different CU/TU sizes is separated into different intervals, and the probability of splitting of the current node is estimated. In the predicting stage, thresholds are set to the splitting probability calculated in the learning stage. Then, ES and EP will perform to accelerate the process of deciding the structure of CTU and RQT. The experimental results show that the proposed method speeds up the HEVC intra coding by 44.4% at the cost of a 1.08% BD-rate increase on average.

Acknowlegement

This work is supported by innovation fund for graduate students provided by school of information and communication engineering, Beijing Univ. of Posts and Telecomm. 2015.

References

[1] J Ohm, Gary J Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand, "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC)," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, pp. 1669-1684, 2012. Article (CrossRef Link)

[2] G.J Sullivan, J.Ohm, W.Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, pp. 1649-1668, 2012. Article (CrossRef Link)

[3] W.Jiang, Hanjie Ma, and Yaowu Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," in *Proc. of Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pp. 1836-1840, 2012, Article (CrossRef Link)

[4] S. Yan, L. Hong, W. He, and Q. Wang, "Group-based fast mode decision algorithm for intra prediction in HEVC," in *Proc. of Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*, pp. 225-229, 2012, Article (CrossRef Link)

[5] L.Shen, Z. Zhang, and P. An, "Fast CU size decision and mode decision algorithm for HEVC intra coding," *Consumer Electronics, IEEE Transactions on*, vol. 59, pp. 207-213, 2013.

Article (CrossRef Link)

[6] L. Zhao, L. Zhang, S. Ma, and D. Zhao, “Fast mode decision algorithm for intra prediction in HEVC,” *Visual Communications and Image Processing (VCIP), 2011 IEEE*, pp. 1-4, 2011, Article (CrossRef Link)

[7] X. Shen and L. Yu, “CU splitting early termination based on weighted SVM,” *EURASIP Journal on Image and Video Processing*, vol. 2013, pp. 1-11, 2013. Article (CrossRef Link)

[8] S. Cho and M. Kim, “Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding,” 2013. Article (CrossRef Link)

[9] K. Choi and E. S Jang, “Fast coding unit decision method based on coding tree pruning for high efficiency video coding,” *Optical Engineering*, vol. 51, pp. 030502-1-030502-3, 2012. Article (CrossRef Link)

[10] H. Li, K. N. Ngan, and Z. Wei, “Fast and efficient method for block edge classification and its application in H. 264/AVC video coding,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, pp. 756-768, 2008. Article (CrossRef Link)

[11] Y. Chao, K. Lin, B. Liu, and J. Yang, “An approximate square criterion for H. 264/AVC intra mode decision,” in *Proc. of Multimedia and Expo, 2008 IEEE International Conference on*, pp. 333-336, 2008, Article (CrossRef Link)

[12] F. Pan, X. Lin, S. Rahardja, K. Pang Lim, ZG Li, Dajun Wu, *et al.*, “Fast mode decision algorithm for intraprediction in H. 264/AVC video coding,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, pp. 813-822, 2005. Article (CrossRef Link)

[13] S. Teng, H. Hang, and Yi-Fu Chen, “Fast mode decision algorithm for residual quadtree coding in HEVC,” in *Visual Communications and Image Processing (VCIP), 2011 IEEE*, pp. 1-4, 2011, Article (CrossRef Link)

[14] Y. Zhang and M. Zhao, “An adaptive RQT mode selection algorithm for HEVC,” in *Proc. of Image and Signal Processing (CISP), 2012 5th International Congress on*, pp. 173-177, 2012, Article (CrossRef Link)

[15] K. Choi and E. S Jang, “Early TU decision method for fast video encoding in high efficiency video coding,” *Electronics letters*, vol. 48, pp. 689-691, 2012. Article (CrossRef Link)

[16] B. Bross, W. Han, J. Ohm, Gary J Sullivan, and Thomas Wiegand, “High efficiency video coding (HEVC) text specification draft 8,” *JCTVC-J1003*, 2012.

[17] Y Piao, J Min, and J Chen, “Encoder improvement of unified intra prediction,” *JCTVC-C207*, 2010.

[18] F Bossen and HM Common, “test conditions and software reference configurations” *JCTVC-L1100, Jan*, 2013.

[19] H. Sun, D. Zhou, and S. Goto, “A low-complexity hevc intra prediction algorithm based on level and mode filtering,” in *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pp. 1085-1090, 2012. Article (CrossRef Link)

.

Yuchen Li is a Ph.D. student in the Wireless Theories and Technologies Lab, Beijing University of Posts and Telecommunications (BUPT), China. He received his B.E. degree in Communication Engineering from BUPT in 2012. His current research is focused on video coding and multimedia communication.

Yitong Liu received her M.S degree in communication engineering from BUPT, China, in 2007. She is a Ph.D. student at the School of Information and Communication Engineering in BUPT. Her research includes multimedia communication and QoE of streaming services.

Hongwen Yang received his Ph.D. degree in communication engineering from BUPT in 2005. Prof. Yang is currently the director of the Wireless Communication Center in the School of Information and Communication Engineering at BUPT. His research mainly focuses on wireless physical aspects.

Dcheng Yang received his Ph.D. degree in communication engineering from BUPT in 1988 and is currently a professor in BUPT. He is also the director of the WT&T Lab at BUPT. Since 1988, he has engaged in studies on communication systems. His recent interests lie in wireless transmission techniques and systems.