Quadtree Degeneration for HEVC

Yuan Gao, Pengyu Liu, Yueying Wu, and Kebin Jia

***Abstract*—The quadtree is one of the most advanced techniques contributing to the excellent compression performance of high efficiency video coding (HEVC). However, the computational complexity increases because the quadtree examines all coding unit (CU) sizes to obtain the optimal CU partitioning. This paper focuses on quadtree degeneration based on a proposed quadtree probability mechanism. Two techniques, a quadtree probability model (QPM) procedure and a quadtree probability update (QPU) procedure, are proposed. The QPM process estimates a CU distribution model based on a quantization parameter (QP) and a group of pictures (GOP). Based on the model, a new quadtree is constructed by skipping low probability tree nodes. The QPU process is performed to update the new quadtree based on scene content change. Update addresses model distortion and ensures the accuracy of the new quadtree. Experimental results demonstrate that the proposed quadtree probability mechanism for quadtree degeneration considerably reduces average encoding time (27.55%) for the low delay condition. Applied to lossless coding, the proposed mechanism achieves a significant 43.10% encoding time reduction. The experiments also show that the proposed quadtree probability mechanism improves HEVC coding efficiency for a variety of applications and sequence characteristics.**

***Index Terms*—Group of pictures (GOP), high efficiency video coding (HEVC), quadtree, quantization parameter (QP), scene content.**

I. Introduction

SINCE the development of H.264/AVC, video coding standards have been continuously improved for network services and mobile communication. HEVC [1] has now been finalized by the Joint Collaborative Team on Video Coding (JCT-VC) after a decade of preparations. Due to the rapid growth of multimedia services, realizing high-efficiency video coding while simultaneously providing high-quality images with minimal transmission delay has remained a challenge [2]. HEVC is designed to offer better coding efficiency and greater flexibility than H.264/AVC [3], [4]. In addition, HEVC is expected to have more impact on high-resolution video (4K and 8K) and high fidelity video (for high-resolution displays such as high definition TV (HD-TV) and ultra-high definition TV (UHD-TV) in the future [5], [6]. HEVC not only inherits crucial elements of H.264/AVC but also adopts numerous new techniques to achieve superior performance. In H.264/AVC, encoding a macroblock (MB) is a complicated technique derived from mode decision [7]. HEVC uses a more complex, quadtree technique to improve compression performance. However, the quadtree greatly increases the computational complexity of the encoder. Computational complexity reduction with negligible performance loss is a major research concern [8], [9]. At present, improving HEVC efficiency by reducing quadtree complexity is a particular research concern, and various research studies have been carried out to achieve this goal.

A considerable amount of effort has been focused on quadtree complexity reduction. Studies on fast coding unit (CU) encoding improvement attempt to represent the redundant information that is often present in quadtree traversal computing. This state of the art research applies machine learning theory to low complexity CU encoding. Zhang [10] proposes a flexible complexity allocation that converts the CU depth decision problem into a classification problem. Ye [11] makes use of Bayesian decision theory in adaptive CU depth decision, which achieves a notable encoding time reduction. Kim [12] combines scene change detection with a minimum error Bayesian decision rule. However, these machine-learning based approaches have large computation and memory costs. This is primarily why machine learning theory has not been widely applied to video coding, even though it could achieve significant performance improvements. Hence, the back-to-basics approach based on the encoder parameters is designed to improve low complexity video coding efficiency. Traditionally, dynamic information redundancy in videos occurs in the spatial and temporal contexts. Thus, alternative mechanisms for intra mode decision and inter mode decision would be interesting. Many works focus on the development of fast quadtree computation based on early CU size decision for HEVC intra encoders [13]–[15] and inter encoders [16]–[19]. Shen [13] skips low probability intra prediction modes in the parent CUs of the upper depth levels or spatially nearby CUs. Ahn [16] simplifies the rate-distortion (RD) competition processes by selectively conducting a mode decision process based on inter predicting unit (split-type, square-type, or non-square-type) modes. Xiong [18] uses a nearest neighbor method to determine CU splitting. These approaches effectively reduce encoding time by using fast mode decision at the prediction unit (PU) and transform unit (TU) levels. However, these approaches need to traverse all CU sizes to obtain the optimal CUs, causing performance limitations. In HEVC, whether to split the CU is determined by PU and TU partitions.

Manuscript received September 23, 2015; revised December 22, 2015, March 1, 2016, and May 31, 2016; accepted July 22, 2016. Date of publication August 8, 2016; date of current version November 15, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61672064, in part by the Project for the Key Project of Beijing Municipal Education Commission under Grant KZ201610005007, in part by the Beijing Postdoctoral Research Foundation under Grant 2015ZZ-23, in part by the China Postdoctoral Research Foundation under Grant 2015M580029 and Grant 2016T90022, and in part by the Computational Intelligence and Intelligent System of Beijing Key Laboratory Research Foundation under Grant 002000546615004. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhu Li.

The authors are with the Beijing Advanced Innovation Center for Future Internet Technology, Beijing University of Technology, Beijing 100124, China, the Beijing Laboratory of Advanced Information Networks, Beijing 100124, China, and also with the College of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China (e-mail: yuangaoyg001@emails.bjut.edu.cn; liupengyu@bjut.edu.cn; wuyueying@emails.bjut.edu.cn; kebinj@bjut.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2016.2598481

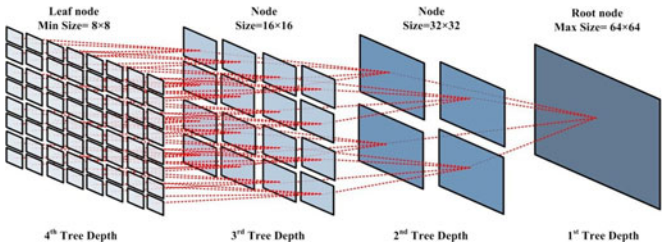


Fig. 1. Complete quadtree in HEVC.

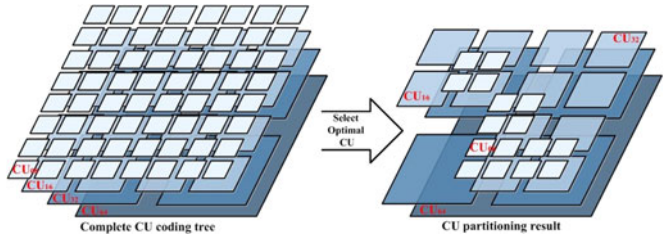


Fig. 2. Traverse a complete quadtree for CU partitioning result. CU64, CU32, CU16, and CU08 represent the optimal CU size ranging from a depth = 1 to a depth = 4.

Thus, the encoder can save significant CU encoding time by directly skipping some of the quadtree nodes rather than PUs or TUs. Accordingly, mechanisms based on low complexity CU encoding that combine early determination with other coding parameters have been successfully applied. A few works have focused on speeding up quadtree computing through early termination mechanisms [20]–[26]. Cen [22] proposes a fast CU depth decision mechanism that uses spatial correlations to achieve a CU depth range determination. Song [25] presents an early merge mode decision method to avoid exhaustive mode checks for CUs derived from recursive quadtree partitioning. Despite these achievements, some prominent studies [27], [28] into low complexity schemes focus solely on quadtree prediction. For instance, Guo [28] proposes a fast CU size selection algorithm based on hierarchical quadtree correlations and determines the size of the current CU based on the subtree distributions of adjacent CUs. In general, current researchers concentrate on reducing quadtree complexity by selecting a specific tree depth to search, instead of traversing all depths. In other words, current approaches only focus on reducing the leaf nodes of the quadtree by lessening depth, which means that the quadtree continues to be computed from its maximum tree root node. However, with different scene contents and different coding parameters, it is possible to distinguish a best quadtree for each frame using a different tree root node size and different tree depth. Indeed, using a befitting quadtree is a key method for reducing HEVC encoder computational complexity. The problem is choosing the appropriate tree root node and tree depth to degenerate a quadtree. Our work presents a quadtree probability mechanism to achieve quadtree degeneration. We propose two important concepts, the quadtree probability model (QPM) and quadtree probability update (QPU), for quadtree degeneration.

In the rest of this paper, Section II reviews quadtree structure and the motivation for quadtree degeneration. Section III describes the discovery and analysis of the factors affecting CU distribution probability. Section IV provides the details of the QPM and QPU. Extensive experimental results are documented and discussed in Section V to evaluate the performance of the proposed quadtree probability mechanism for quadtree degeneration. Finally, Section VI presents our conclusions.

II. Overview and Motivation

A. Overview of Quadtree Structure in HEVC

HEVC hierarchical coding structure is implemented by quadtree, as shown in Fig. 1. A quadtree is defined as a tree in which each node has exactly zero or four children. In other words, every node is either a leaf or has four children. The number of nodes must meet the maximum, and all the leaves must be in the same layer. Based on this definition, HEVC usually uses root node and depth for describing a quadtree. The root node of the quadtree is a square region called a coding tree unit (CTU) with a size of 64 × 64 and a leaf node of a quadtree partitioning is a CU. The quadtree allows recursive splitting of each unit into four equally sized nodes, beginning with the CTU and terminating when maximum tree depth is reached. Thus, a complete quadtree has a CTU of size 64 × 64, a maximum tree depth of four, and each CU may be as large as 64 × 64 or as small as 8 × 8.

The encoder selects the optimal CU partitioning for the current CTU by calculating the coding costs J of various CU sizes. The cost function J is specified by

$$J = SSE + \lambda \cdot B \tag{1}$$

where B specifies the bit cost, SSE represents the difference between two blocks with the same block size, and λ is the cost computation Lambda value. For each CTU, the decision to split the current CU is based on the minimum between the J value of the current CU size and the summed J values of the resulting four smaller CU sizes. A CU partitioning result is calculated by a complete quadtree as shown in Fig. 2.

The flexible quadtree structure of HEVC provides a significant improvement in coding gain. However, it causes a dramatic increase in encoding complexity because the encoding process needs to explore every single quadtree node size from 64 × 64 to 8 × 8. Each quadtree node tries all possible PU and TU sizes to determine the best PU and TU partition. This exhaustive tree node check results in an enormous increase in computational complexity.

B. Motivations for Quadtree Degeneration

In HEVC the encoder calculates one CTU by traversing a complete quadtree with root node size 64 × 64 and depth 4. Increasing video resolution increases the number of CTUs rapidly, which leads to increased encoding complexity. The process of quadtree degeneration reduces CU encoding time by omitting some tree nodes.

As stated above, a quadtree is described by root node size and tree depth. Thus a rough way to determine the relationship between quadtree complexity and encoding time is to downsize

TABLE I
ENCODING TIME SAVING (%) UNDER DIFFERENT QUADTREE STRUCTURES COMPARED WITH HEVC QUADTREE WITH A DEPTH OF FOUR AND A CTU SIZE OF 64 × 64

Sequence	Depth = 3		Depth = 2			Depth = 1		
	CTU	CTU	CTU	CTU	CTU	CTU	CTU	CTU
	64 × 64	32 × 32	64 × 64	32 × 32	16 × 16	64 × 64	32 × 32	16 × 16
Traffic	19.62	20.72	51.39	39.59	45.81	73.48	71.27	64.96
Cactus	16.98	18.34	31.88	27.18	43.31	69.34	67.49	62.63
Party Scene	20.17	23.01	35.25	37.91	39.55	74.62	71.99	68.24
Race Horses	23.59	29.74	32.26	38.14	44.45	76.90	70.84	66.98
Johnny	17.95	16.40	49.27	48.42	46.36	70.73	68.49	62.41
Slide Show	22.24	18.86	40.39	41.38	42.27	70.02	66.17	64.28
Average TS(%)	20.09	21.18	40.07	38.77	43.63	72.52	69.38	64.92

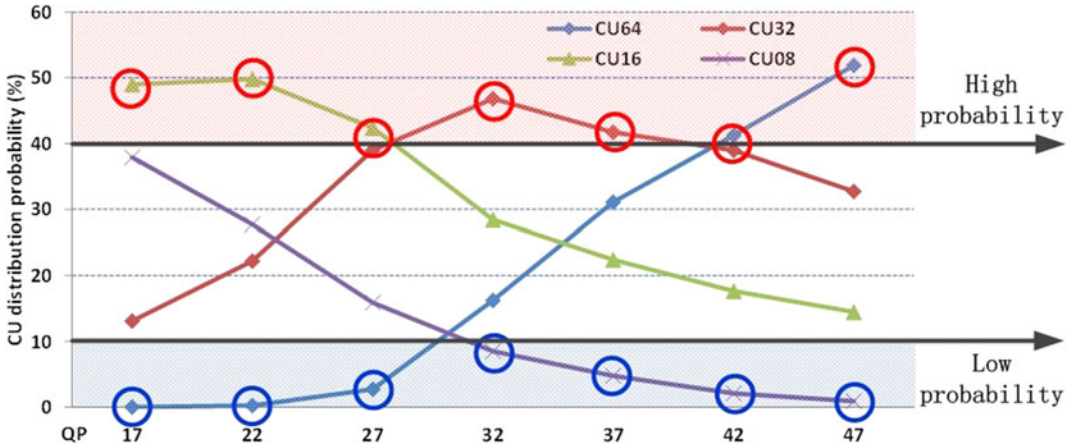


Fig. 3. CU distribution probability based on QP.

the root node and depth of a complete quadtree. Our experiment compares various degenerated quadtree structures to the HEVC complete quadtree structure. As illustrated in Table I, quadtree complexity drops proportionally with decreasing depth. In addition to depth, CTU size is also related to quadtree complexity. Table I shows not only that a simpler quadtree structure can contribute to reduced encoding complexity but also that the encoder can improve time complexity by choosing a larger CTU and shallower depth.

However, encoding accuracy drops severely when a quadtree is too simple. Thus, for the encoder to save CU encoding time while maintaining encoding accuracy, the quadtree degeneration must be built with the provision that a low probability tree node can be omitted, but others must be retained. However, both internal and externSal encoder factors affect the final CU partitioning probability. Thus it is necessary to identify the factors affecting CU distribution and then use these factors to ascertain how to predict CU distribution to obtain the optimal CU partitioning probability.

III. FACTORS AFFECTING CU DISTRIBUTION

A. CU Distribution Based on QP

First we test different types of videos to investigate general CU distribution. The statistics are obtained in a common video coding condition [29] recommended by JCT-VC. This configuration leads to a non-uniform J by defining the cost computation λ in (1) as

$$\lambda = \alpha \cdot W_k \cdot 2^{\frac{(QP-12)}{3}} \tag{2}$$

where α and W_k are weighting factors related to different configurations. Thus for any specific configuration, λ is related to the quantization parameter (QP). Indeed, the QP affects the optimal CU partitioning by changing J. For example, Fig. 3 shows the CU distribution probability of the encoded BQSquare sequence and demonstrates that there is a high correlation between the QP and the optimal CU partitioning. CU partitioning distribution in a particular range occupies a greater percentage of the CU allocation in the same sequence. In general, smaller QPs are highly likely to be encoded by CUs of small sizes. In contrast, larger CUs are much more likely to encode larger QPs. For a small QP, a CU size of 64 × 64 has quite a low probability, while a small CU size has a high probability. For example, CU sizes of 16 × 16 and 8 × 8 occupy 88.70% of the CU distribution when QP = 17. In the range QP = 22 to QP = 42, the CU distribution changes dramatically. The probabilities of CUs of size 64 × 64 and 32 × 32 rapidly increase, while the probability of a CU size of 16 × 16 or 8 × 8 rapidly declines. The total number of the CUs of sizes 64 × 64, 32 × 32 and 16 × 16 comprise 98.90% of the CU distribution when QP = 42. Therefore, it can be inferred that QP is an important parameter in determining CU distribution. We can also estimate the CU distribution

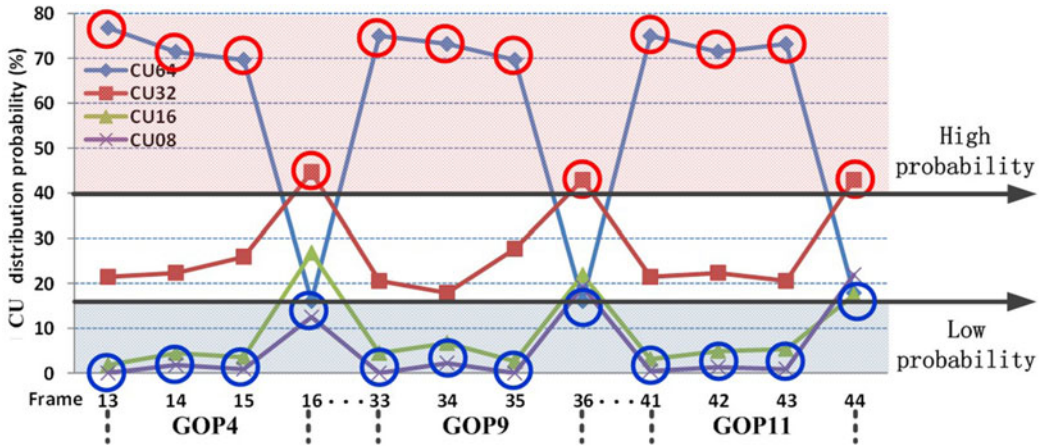


Fig. 4. CU distribution probability based on GOP.

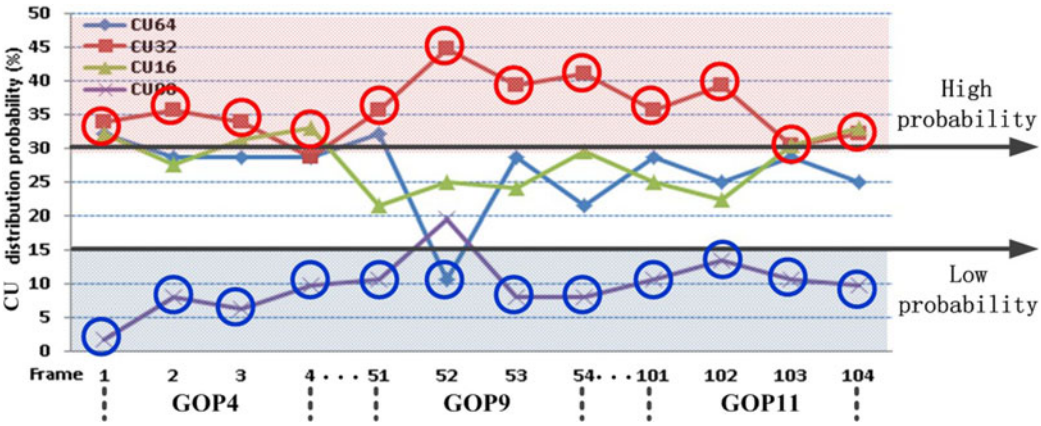


Fig. 5. CU distribution probability based on scene content.

percentage for a specific QP. Thus, it is unnecessary to calculate all quadtree nodes from CU64 to CU08 when determining the optimal CU partitioning. Based on QP, encoders need to traverse only the tree nodes with the greatest CU distribution probabilities, and can skip tree nodes with low CU distribution probabilities, as these low probability quadtree nodes are rarely used and contribute little to promoting coding efficiency.

B. CU Distribution Based on GOP

The QP has a serious impact on CU distribution. However, when encoding a video sequence, an HEVC encoder makes use of a complex QP offset rather than a constant QP value. The QP offset causes the encoder to use different QPs based on the temporal layer. For different coding configurations, the QP of each inter encoded picture is derived by adding an offset to the QP of the intra encoded picture (QPI), depending on the temporal layer. For example, if a base QP is set to the QPI for the first layer, the QP of the second layer QPL2 = QPI + 2, the QP of the third layer QPL3 = QPI + 3, and so on. In HEVC, the QP offset hierarchy level is configured within the group of pictures (GOP). This means the encoder sets a series of QP values for each frame in a GOP, and each GOP has the same QP configuration. It can be inferred that frames in the same order in different GOPs may exhibit a similar CU distribution. As an example, Fig. 4 demonstrates the CU distribution probability of the encoded FourPeople sequence with QPI = 32 and QP offset = 3, 2, 3, 1. We observe that there is a relationship between GOP structure and optimal CU partitioning. A CU size of 64 × 64 has the greatest probability in the first three frames of a GOP. However, in the fourth frame, the probability of a CU of size 64 × 64 drops severely and other CU sizes' probabilities simultaneously increase markedly. It is noteworthy that almost all of the GOPs in this sequence follow the above pattern. Because different frames have the same, fixed QP offset in different GOPs, each GOP fixes the picture order count (POC), and the same POC has the same QP offset, there is a similar CU distribution in adjacent GOPs. Therefore, we conclude that GOP structure affects CU distribution when a QP offset is present. Moreover, we find that the N^{th} frame and the $(N + GOPSize)^{\text{th}}$ frame have a similar CU distribution. We use this rule to predict CU distribution probability and then omit low probability quadtree nodes.

C. CU Distribution Based on Scene Content

Another factor affecting CU distribution is scene content change. A variety of factors may cause scene content change including shot change, lens zoom, lens rotation, and motion occlusion. A current frame may be different from the previous frame due to the introduction of new objects. In other words, the steady CU distribution probability related to QP and GOP is broken. For example, Fig. 5 shows the CU distribution probability for the encoded BasketballPass sequence. To eliminate the

influence of a varying QP, we set the QP offset equal to zero. All the frames are coded with QP = 32. If there were no scene content change, the CU distribution probability would not change during the sequence. However, probability of a CU of size 64 × 64 drops below 10% and the probability of a CU of size 8 × 8 increases to 20% in the 9th GOP. A typical scene content change occurs in the 9th GOP and obviously affects the CU distribution. In addition, Fig. 5 shows that the CU distribution changes begin at the 51st frame; the previous CU distribution probabilities show little change, as scene content does not change much for 50 frames. However, the frame rate of this sequence is equal to 50 fps. We believe that the interval of scene content change is usually longer than one second. Therefore, excluding the effect of the QP and GOP, CU distribution is not invariable and varies due to scene content change. Updating CU distribution is important in maintaining an accurate CU distribution probability. When updating the CU distribution probability, the frame rate parameter must be considered for updating CU distribution probability and determining when to compute a new quadtree.

IV. Proposed Quadtree Probability Mechanism

Based on the factors affecting CU distribution probability presented above, we propose a quadtree probability mechanism for quadtree degeneration. Fig. 6 shows the flowchart for the implementation of the quadtree probability mechanism in the HEVC encoder. The encoder encodes the first GOP, then uses its CU partitioning result to direct the encoding of later GOPs. Frames in later GOPs degenerate the quadtree based on previous GOPs. The quadtree probability mechanism consists of two key procedures. The first concerns the quadtree probability model, which determines how to establish the probability model and how to predict a new quadtree using the model. The second is the quadtree probability update, which determines how to recompute the probability model and how often to perform an update. The goal of our method is to achieve HEVC encoding complexity reduction with negligible additional computation.

A. Quadtree Probability Model

The quadtree probability model (QPM) uses an optimal CU partitioning to establish a probability model. The key technique proposes to use the N^{th} optimal CU partitioning to predict the $(N + GOPSize)^{\text{th}}$ quadtree. QPMs are designed to construct a new quadtree for each frame. The QPM process uses two steps to obtain the root node size and depth for constructing a new quadtree.

Step 1. QPM establishing: First, a probability model is established by computing the CU distribution probability for an encoded GOP (this may be the first GOP). The QPM uses the optimal CU partitioning acquired via a complete quadtree to establish the function F_{esta}. $F_{esta}(CU, Frame|GOP_{coded})$ represents the probability of each CU size for each frame in the encoded GOP. F_{esta} is defined as

$$F_{esta}\left(CU, Frame\,|GOP_{coded}\right) = P_{ij} = \begin{bmatrix} P\left(CU_{64}, Frame_1\right) \cdots P\left(CU_{64}, Frame_n\right) \\ \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ P\left(CU_{08}, Frame_1\right) \cdots P\left(CU_{08}, Frame_n\right) \end{bmatrix} \tag{3}$$

where P_{ij} is a two-dimensional matrix in which each element is calculated as

$$P(CU, Frame) = P(CU|Frame) = CU \Big/ CU_{64} + CU_{32} + CU_{16} + CU_{08} \tag{4}$$

where is P an empirical frequency used to approximate the corresponding conditional probability of a certain CU size in a frame. Each CU size is determined based on a 4 × 4 pixel block.

Step 2. QPM Predicting: Second, QPM predicts a new quadtree for the current, unencoded GOP. Another important function, δ_{ij}, is used to determine whether the nodes of the new quadtree exist. δ_{ij} contains only $\{0, 1\}$ which means the state of the node is {nonexistent, existent}

$$\delta_{ij} = \begin{cases} 0, P_{ij} < \sigma \\ 1, P_{ij} \geq \sigma \end{cases} \tag{5}$$

where σ is an empirical value.

A new function, F_{pred}, similar in form to F_{esta}, predicts the new quadtree of each frame in the current GOP:

$$F_{pred}\left(CU, Frame\,|GOP_{current}\right) = \delta_{ij} = \begin{bmatrix} \delta\left(CU_{64}, Frame_1\right) \cdots \delta\left(CU_{64}, Frame_n\right) \\ \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ \delta\left(CU_{08}, Frame_1\right) \cdots \delta\left(CU_{08}, Frame_n\right) \end{bmatrix} \tag{6}$$

A new quadtree is calculated for each frame when $\delta(CU, Frame) = 1$.

1) **Root node** is the maximum size of $\delta(CU, Frame)$
   leaf node is the minimum size of $\delta(CU, Frame)$
2) **Depth** is $\log_2(Root\ node) - \log_2(leaf\ node) + 1$, **node** $= \{64,\ 32,\ 16,\ 8\}$

As shown in Fig. 6, the proposed method uses a complete quadtree,δ_{ij}, to encode the first GOP. After the quadtree probability model process runs, the encoder obtains a degenerated quadtree $\overline{\delta}_{ij}$. $\overline{\delta}_{ij}$ is used to encode later GOPs. The root node or depth of the new quadtree is less than or equal to that of the complete quadtree, which means the encoder can obtain an approximate optimal CU partitioning result without searching all the CU sizes.

B. Quadtree Probability Update

Quadtree probability update (QPU) is designed to address the problem that scene content change leads to inaccuracy in the probability model. First, the key to guaranteeing coding efficiency is to determine how often to perform a QPU. Overly

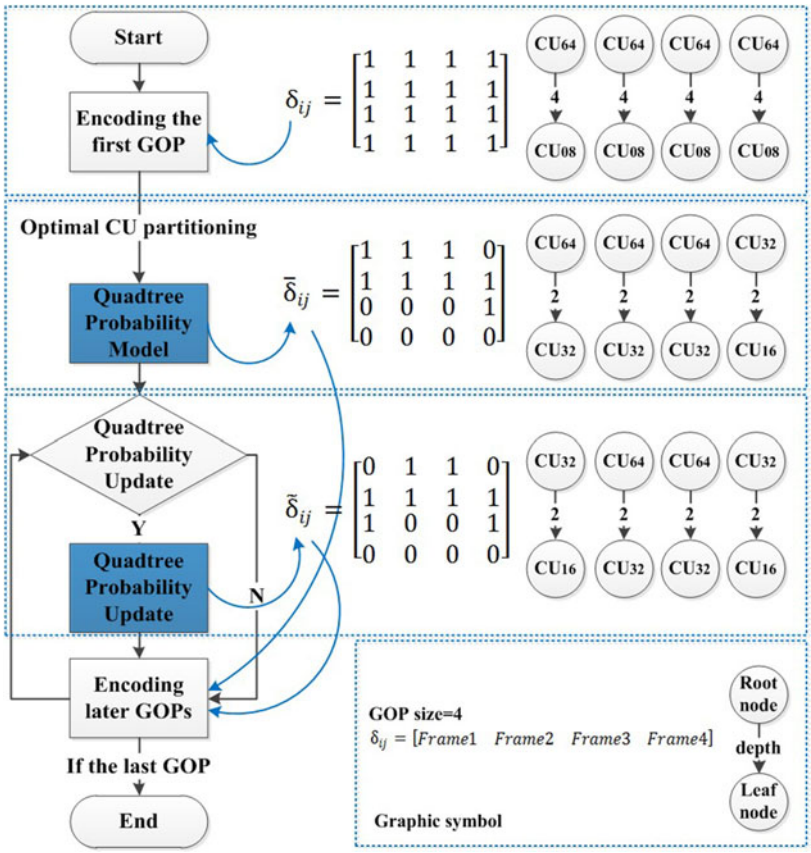


Fig. 6. Flowchart of the proposed quadtree probability mechanism.

frequent updating introduces unnecessary computational complexity, but updating too infrequently causes QPM inaccuracy. Second, the QPU process defines how to recompute F_{esta} for the current GOP. Once the encoder performs the QPM process, the new F_{esta}^{n} replaces the F_{esta}^{n-1} or constructing the quadtrees of later GOPs.

1) *Update Depends on Frame Rate:* To balance coding efficiency, QPU is performed based on the frame rate parameter of each sequence. The frame rate parameter determines the number of frames per second. The number of GOPs within one second is given by N as

$$N = GOPSize \cdot floor\left(\frac{FrameRate}{GOPSize}\right). \tag{7}$$

The QPU period T is defined as

$$T = \begin{cases} floor\left(\frac{N}{2}\right), & GOPSize < FrameRate \\ GOPSize, & GOPSize \geq FrameRate. \end{cases} \tag{8}$$

T is the frame interval between updates. When GOPSize is small, T is a multiple of GOPSize that is close to the frame rate. When GOPSize is large, T is equal to GOPSize and frame rate is immaterial. The GOPSize setting impacts the performance of QPU because the larger GOPSize is, the less frequently F_{esta} updates.

2) *Update Depends on* F_{esta}^{n-1} : Although the frame rate period determines the frame where update occurs, if the current F_{esta}^{n} and the previous one, F_{esta}^{n-1} have the same CU distribution, it is unnecessary to update F_{esta}^{n}. If F_{esta}^{n} and F_{esta}^{n-1} have different CU distribution probabilities, then QPU must make use of F_{esta}^{n-1}. Hence, F_{esta}^{n} is updated in the QPU according to

$$F_{esta}^{n} = \begin{cases} F_{esta}^{n-1}, & Z\left(F_{esta}^{n}\right) = Z\left(F_{esta}^{n-1}\right) \\ \rho \cdot F_{esta}^{n-1} + (1-\rho) \cdot F_{esta}^{n}, & Z\left(F_{esta}^{n}\right) \neq Z\left(F_{esta}^{n-1}\right) \end{cases} \tag{9}$$

where Z represents the number and positions of the "0" elements in matrix F. ρ is an equilibrium factor used for adjusting the performance of QPU and is in [0, 1]. Next, the encoder uses (5) and (6) to obtain a new F_{pred}.

As shown in Fig. 6, whether the encoder performs QPU is determined by T. After update, the encoder obtains a new degenerated quadtree δ_{ij}. This δ_{ij} is used to encode later frames. QPU is needed in video coding not only to maintain the accuracy of the probability model but also to avoid propagating errors to later GOPs.

In summary, the QPM avoids traversing a complete quadtree by constructing a new quadtree for encoding later frames. QPU maintains the accuracy of the new quadtree while limiting increase in computational complexity as much as possible.

V. Performance Assessments

A. General Experimental Configuration

To evaluate the coding effectiveness of the proposed quadtree probability mechanism, we make experiments with respect to

TABLE II
TEST SEQUENCES

Class	Size	Sequence	Frame Rate
Class A	2560 × 1600	Traffic, PeopleOnStreet	30fps, 30fps
Class B	1920 × 1080	ParkScene, Cactus	24fps, 50fps
Class C	832 × 480	RaceHorses, BQMall, Basketbal Drill	30fps, 60fps, 50fps
Class D	416 × 240	Rac Horses, BQSquare, BasketballPass	30fps, 60fps, 50fps
Class E	1280 × 720	FourPeople, Johnny	60fps, 60fps
Class F	–	ChinaSpeed, SlideShow	30fps, 20fps

TABLE III
GENERAL EXPERIMENTAL CONFIGURATION

PC Configuration	
CPU	Intel Core i7
Memory	8 GB
CU Definition	
Max CTU	64 × 64
Min CTU	16 × 16
Max Partition Depth	4
Min Partition Depth	1
Parameter Setting	
σ in (5)	0.15
ρ in (9)	0.25

HEVC test model version 15.0 and sequences of various resolutions from 2560 × 1600 to 416 × 240, as shown in Table II. The encoder configures all the sequences according to the common test conditions of HEVC standardization. Table III shows other experimental design details. For the experiments, HM15.0 sets an anchor by using a completed quadtree.

Performance is measured by distortion and rate for a variety of conditions. The Bjøntegaard delta peak signal-to noise ratio (BDPSNR) and the Bjøntegaard delta bit rate (BDBR) [30] are used to evaluate the performance of the proposed method. All the experiments calculate encoding time-savings according to the following equation:

$$TS\,(\%) = \frac{Enc.time\,(Anchor) - Enc.time\,(Prop)}{Enc.time\,(Anchor)} \times 100. \tag{10}$$

B. Coding Performance Assessment

For evaluation of lossy coding, the experiment has three possibilities: the all intra (AI) condition, the low delay (LD) condition and the random access (RA) condition. For all of the conditions, the first frame is encoded by a complete quadtree, and QPI is set to 22, 27, 32 and 37. Specifically, for AI the GOP Size is equal to 1 and there is no QP offset. For RA, due to the Intra Period, all I frames use the complete quadtree to guarantee the quality of reconstruction. Table IV gives the specific configurations for LD and RA.

Table V and Table VI show the experimental results for the proposed quadtree probability mechanism. As shown in Table V, under the AI condition the proposed method reduces encoding time by 20.24% with a 0.06 dB BDPSNR drop and 0.63% BDBR increase. Because there is a single frame in any GOP, then F_{esta} and F_{pred} have a single column vector. The quadtree of each frame imitates its previous frame. Without a QP offset, CU distribution under the AI condition is related to the fixed QPI value and scene content.

TABLE IV
ENCODER CONFIGURATION

	LD	RA
GOP Size	4	8
POC	1, 2, 3, 4	8, 4, 2, 1, 3, 6, 5, 7
QP offset	3, 2, 3, 1	1, 2, 3, 4, 4,3,4,4

TABLE V
PERFORMANCE OF PROPOSED MECHANISM COMPARED WITH HM15.0 UNDER AI

Class	Proposed method under AI condition		
	BDPSNR (dB)	BDBR (%)	TS (%)
Class A	−0.01	0.33	14.92
Class B	−0.03	0.78	21.44
Class C	−0.07	0.84	23.85
Class D	−0.02	0.31	15.12
Class E	−0.01	0.20	11.31
Class F	−0.19	1.30	34.78
Average	−0.06	0.63	20.24

TABLE VI
PERFORMANCE OF PROPOSED MECHANISM COMPARED WITH HM15.0 UNDER LD AND RA

Class	Proposed method under LD condition			Proposed method under RA condition		
	BDPSNR (dB)	BDBR (%)	TS (%)	BDPSNR (dB)	BDBR (%)	TS (%)
Class A	–	–	–	−0.02	0.94	26.47
Class B	−0.03	1.07	27.89	−0.03	0.76	24.29
Class C	−0.08	1.44	28.61	−0.06	1.31	29.88
Class D	−0.05	1.02	23.01	−0.04	1.08	19.03
Class E	−0.01	0.75	32.99	–	–	–
Class F	−0.04	1.28	25.27	−0.03	1.10	21.45
Average	−0.04	1.11	27.55	−0.04	1.04	24.22

In most cases, the LD condition applies to real-time communication, while the RA condition applies to video playback and stream splicing. According to Table VI, when compared to HM15.0 LD conditions, the proposed method yields a 27.55% average reduction in total encoding time with an average BDBR gain of 1.11% and a 0.04 dB BDPSNR loss. Under the RA condition, the proposed method achieves a 24.22% reduction in total encoding time with a 1.04% BDBR gain and 0.04 dB BDPSNR loss. The greatest improvement in encoding time occurs for Class E under LD. For the Class E test sequences, the proposed method saves on total encoding time with negligible loss in BDPSNR. This indicates that Class E has vast background regions that exhibit little motion. Even if Class E performs the most complex quadtree, the probabilities of CUs of size 16 × 16 and 8 × 8 are quite small. Therefore, using a new quadtree without CUs of size 16 × 16 or 8 × 8 as

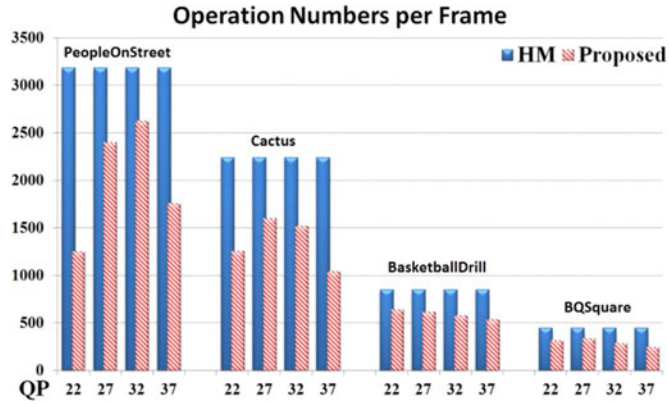


Fig. 7. Assessment for the number of operations.

the anchor for the frame can achieve high coding performance while markedly reducing encoding time. The proposed method can achieve considerable efficiency improvement and precise quadtree prediction.

The complexity can be assessed with respect to the number of operations per frame as shown in Fig. 7. Compared with HM15.0, our proposed method achieves an average reduction in number of operations of 35.57%. The proposed method achieves greater reductions in number of operations as video resolution increases. Moreover, a small or large QP (we assume a medium QP to be 32) provides a greater reduction in operations.

The efficiency of optimal CU partitioning is defined as the probability of obtaining the optimal CUs by performing a quadtree node once. For the anchor, the CU partition time is a constant determined by the number of nodes in the complete quadtree. The optimal CU must be of CU size CU64, CU32, CU16 or CU08, so the efficiency of the anchor is 0.25. For the proposed method, the number of quadtree nodes is less than or equal to those of the anchor. Therefore, the proposed method obtains the optimal CU by performing fewer quadtree nodes. In most instances, the efficiency of the proposed method is better than 0.25. Occasionally, the omission of some low probability nodes by the proposed method prevents it from obtaining the optimal CU partitioning result. This deviation is reflected by the BDPSNR drop and BDBR increase. However, the deviation affects the coding efficiency only slightly, which we consider as a fair trade for the coding complexity reduction.

In addition, the proposed method achieves a better coding performance compared to other, similar fast quadtree algorithms. Based on (10), the anchor is set to the same low complexity quadtree used by the methods of [22] and [28]. Table VII compares the coding performance of these two methods to the proposed method's results. Compared with the reference methods, the proposed method maintains a good quality reconstructed picture, the PSNR increases at most 0.15 dB, and the bit rate drops at most 1.97%, maintaining the HEVC high compression advantage. Most importantly, the proposed method further reduces encoding time an average of 8.67% and 6.09%, compared, respectively to method [22] and method [28]. These results show that the proposed method is more efficient than simply optimizing either the tree root node or the tree depth. Generally, the

TABLE VII
PERFORMANCE OF PROPOSED MECHANISM COMPARED WITH SIMILAR METHODS UNDER LD

Class	Compared with the method in [22]			Compared with the method in [28]		
	BDPSNR (dB)	BDBR (%)	TS (%)	BDPSNR (dB)	BDBR (%)	TS (%)
Class A	0.12	−1.04	4.50	0.01	−0.03	2.41
Class B	0.02	−1.78	9.72	−0.01	0.09	9.82
Class C	−0.03	0.55	14.69	−0.02	0.18	14.09
Class D	0.01	0.02	10.02	0.01	0.06	4.18
Class E	0.15	−1.29	7.75	0.02	−0.05	3.92
Class F	0.12	−1.97	5.36	0.01	−0.04	2.11
Average	0.07	−0.92	8.67	0.00	0.04	6.09

TABLE VIII
PERFORMANCE OF PROPOSED MECHANISM COMPARED WITH HM15.0 FOR VISUALLY LOSSLESS CODING AND LOSSLESS CODING UNDER LD

Class	Proposed method for visually lossless coding			Proposed method for lossless coding	
	BDPSNR (dB)	BDBR (%)	TS (%)	Bit-rate Increase (%)	TS (%)
Class B	−0.01	0.23	48.79	0.21	48.66
Class C	−0.03	0.89	36.45	0.44	38.47
Class D	−0.02	0.80	23.00	0.37	30.31
Class E	−0.01	0.41	43.98	0.13	47.79
Class F	−0.03	1.02	47.02	0.79	50.25
Average	−0.02	0.67	39.85	0.39	43.10

proposed method achieves significant encoding time reduction with negligible BDBR loss in comparison to the HM15.0 and the state of the art low complexity quadtree methods.

Another experiment is designed to evaluate the performance of the proposed method for lossless coding and visually lossless coding. For the visually lossless condition, the QP is set to 0, 4, 8 and 12. For the lossless condition, QP is set to 0. The specific configuration is sourced from [31]. Visually lossless or lossless compression is desirable for many professional applications such as medical imaging, surveillance, and archiving. Table VIII presents the coding performance of the proposed method in comparison to HM15.0 for the visually lossless and lossless conditions. Average encoding time-savings of 39.85% and 43.10%, respectively, are achieved, while maintaining a good RD performance. Additionally, the proposed method achieves better encoding complexity reductions for visually lossless and lossless coding than for lossy coding because, with small QPs, low probability CU sizes (CU64 and CU32) occupy almost none of the optimal CU distribution, making the new quadtree simpler. Therefore, the proposed method provides a great approach to complexity reduction for high quality video coding.

This paper also investigates the CU partitioning results and object assessment of the proposed quadtree mechanism. The CU partitioning results for the Johnny sequence are shown in Fig. 8. The proposed method chooses not to perform an 8×8 CU size, due to the large number of still and flat background blocks. Hence, the proposed method may lose some details in lossy coding. However, it can be observed that, although CU

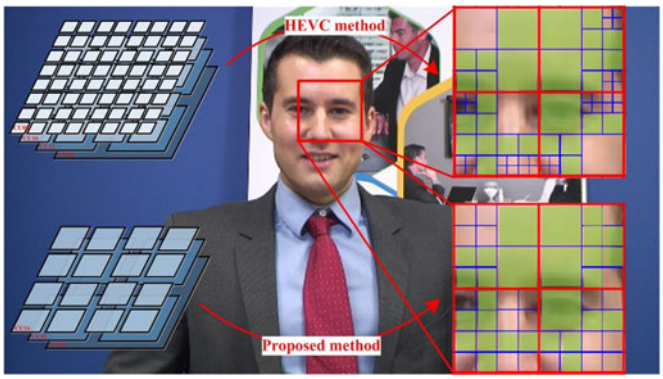


Fig. 8. Object assessment between different CU partitioning results.

partitioning results in proposed method do not contain a CU of size 8×8 in the facial area, the reconstructed picture has little object difference compared to the HEVC method. This is because the QP has a greater impact on quality degeneration than does partitioning the CU without a CU of size 8×8. With the lossy coding QP, the details are missing regardless of whether the encoder performs a CU of size 8×8. The proposed method uses a simpler quadtree to achieve a similar CU partitioning result.

Generally speaking, the quadtree probability mechanism successfully reduces CU encoding complexity by degenerating the quadtree for each frame, reducing computing complexity and memory usage and making the method appropriate for common applications.

VI. Conclusion

Factors such as QP, GOP and scene content change greatly affect CU partitioning. Our research emphasizes the importance of degenerating the quadtree by both root node and depth. To maintain compression performance, quadtree degeneration considers QP, GOP and scene content change. Successful quadtree degeneration provides an effective fast CU encoding scheme without adding computing complexity, further validating the universality of low complexity algorithms.

In this paper, a quadtree probability mechanism is proposed for quadtree degeneration. We examine CU distribution to address the high computational complexity caused by the traversal of the complete quadtree in HEVC, and propose a quadtree probability mechanism for optimizing quadtree degeneration. We show that CU distribution is related to encoder parameters and changes in scene content. Based on this discovery, the QPM and QPU are designed to create a new quadtree model and update it. The QPM and QPU of the proposed method together provide better performance compared to HM15.0 and state of the art fast CU size decision algorithms. The experimental results show that the proposed quadtree probability mechanism achieves 24% encoding time reduction on average for lossy coding and an average 42% encoding time reduction for (visually) lossless coding. Compared with HM15.0, the average number of operations is reduced 36%. Compared to other advanced fast CU size decision algorithms, the proposed method also achieves an 8% encoding time reduction. The proposed quadtree probability mechanism prunes the original quadtree, avoiding low probability CU traversal, reduces unnecessary encoding time and achieves similar compression performance to the original tree. Fundamentally, the proposed quadtree probability mechanism efficiently improves the performance of HEVC real-time encoding and can be combined with other fast video coding techniques to accelerate encoding speed. Additionally, we believe the proposed quadtree probability mechanism can be used for various applications with limited computational resources.

References

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] G. J. Sullivan, J. Gary, and J. R. Ohm "Recent developments in standardization of high efficiency video coding (HEVC)," presented at the SPIE Applications of Digital Image Processing, San Diego, CA, USA, 2010.

[3] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[4] D. Marquardt, M. Jongbloed-Pereboom, and B. Staal "Performance comparison of H.265/MPEG-HEVC, VP9, and H. 264/MPEG-AVC encoders," in *Proc. Picture Coding Symp.*, 2013, pp. 394–397.

[5] H. Li, K. N. Ngan, and Q. Liu "FaceSeg: automatic face segmentation for real-time video," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 77–88, Jan. 2009.

[6] S. Ma, S. Wang, and W. Gao "Low complexity adaptive view synthesis optimization in HEVC based 3D video coding," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 266–271, Jan. 2014.

[7] A. Jiménez-Moreno, E. Martínez-Enríquez, and F. Díaz-de-María. "Mode decision-based algorithm for complexity control in H.264/AVC," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1094–1109, Aug. 2013.

[8] Z. Ma, H. Hu, and Y. Wang "On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding," *IEEE Trans. Multimedia*, vol. 13, no. 6, pp. 1240–1255, Dec. 2011.

[9] K. Won and B. Jeon "Complexity-efficient rate estimation for mode decision of the HEVC encoder," *IEEE Trans. Broadcast.*, vol. 61, no. 3, pp. 425–435, Sep. 2015.

[10] Y. Zhang, S. Kwong, and X. Wang "Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding," *IEEE Trans. Image Process.*, vol. 24, no. 7, pp. 2225–2238, Jul. 2015.

[11] F. Ye "An adaptive CU mode decision mechanism based on Bayesian decision theory for H. 265/HEVC," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2014, pp. 1–6.

[12] H. S. Kim and R. H. Park "Fast CU partitioning algorithm for HEVC using an online-learning-based bayesian decision rule," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 130–138, Jan. 2016.

[13] L. Shen, Z. Zhang, and P. An "Fast CU size decision and mode decision algorithm for HEVC intra coding," *IEEE Trans. Consum. Electron.*, vol. 59, no. 1, pp. 207–213, Feb. 2013.

[14] M. Zhang, Y. Zhang, and H. Bai "Fast CU Splitting in HEVC Intra coding for screen content coding," *IEICE Trans. Inf. Syst.*, vol. 98, no. 2, pp. 467–470, 2015.

[15] S. Cho and M. Kim "Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 9, pp. 1555–1564, Sep. 2013.

[16] Y. J. Ahn and D. Sim "Square-type-first inter-CU tree search algorithm for acceleration of HEVC encoder," *J. Real-Time Image Process.*, vol. 12, pp. 419–432, 2015.

[17] J. Xiong, H. Li, F. Meng, S. Zhu, and Q. Wu "MRF-based fast HEVC inter CU decision with the variance of absolute differences," *IEEE Trans. Multimedia*, vol. 16, no. 8, pp. 2141–2153, Dec. 2014.

[18] J. Xiong, H. Li, and Q. Wu "A fast HEVC inter CU selection method based on pyramid motion divergence," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 559–564, Feb. 2014.

[19] S. Ahn, B. Lee, and M. Kim "A novel fast CU encoding scheme based on spatiotemporal encoding parameters for HEVC inter coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 3, pp. 422–435, Mar. 2015.

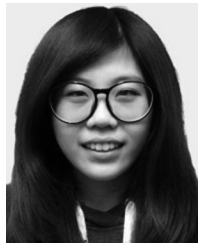
[20] J. Lee, S. Kim, and K. Lim "A fast CU size decision algorithm for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 3, pp. 411–421, Mar. 2015.
[21] L. Shen, P. An, and Z. Zhang "A 3D-HEVC fast mode decision algorithm for real-time applications," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 3, 2015, Art. no. 34.
[22] Y. F. Cen, W. L. Wang, and X. W. Yao "A fast CU depth decision mechanism for HEVC," *Inf. Process. Lett.*, vol. 115, pp. 719–724, 2015.
[23] L. Shen, Z. Liu, and X. Zhang "An effective CU size decision method for HEVC encoders," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 465–470, 2013.
[24] S. Huade, L. Fan, and C. Huanbang "A fast CU size decision algorithm based on adaptive depth selection for HEVC encoder," in *Proc. Int. Conf.* Audio, Language Image Process., 2014, pp. 143–146.
[25] Y. Song and K. Jia "Early merge mode decision for texture coding in 3D-HEVC," *J. Visual Commun. Image Representation*, vol. 33, pp. 60–68, 2015.
[26] C. S. Park, B. G. Kim, G. S. Hong, and S. K. Kim "Fast coding unit (CU) depth decision algorithm for high efficiency video coding (HEVC)," *Advances in Computer Science and Its Applications*. Berlin, Germany: Springer, 2014. pp. 293–299.
[27] G. Chi, X. Jin, and Q. Dai "A quad-tree and statistics based fast CU depth decision algorithm for 3D-HEVC," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, 2014, pp. 1–5.
[28] L. Guo, L. Zhou, and X. Tian "Adaptive coding-unit size selection based on hierarchical quad-tree correlations for high-efficiency video coding," *J. Electron. Imag.*, vol. 24, no. 2, 2015, Art. no. 023036.
[29] H. Yu "Common conditions for screen content coding tests," presented at the 18th Joint Collaborative Team Video Coding Meeting, Sapporo, Japan, Paper JCTVC-R1015, 2014.
[30] G. Bjontegaard "Calculation of average PSNR differences," Calculation of Average PSNR Differences between RD-curves," presented at the Video Coding Experts Group Meeting, Austin, TX, USA, 2001.
[31] "Using qp=0, 4, 8, 12 for visually lossless coding experiments and results from SC coding by packed pixel Pseudo-2D-matching integrated with HM12.0RE4.0" Joint Collaborative Team on Video Coding of ISO/IEC and ITU-T, Geneva, Switzerland, JCTVC-O0269, Oct. 23 2013.

Yuan Gao received the M.E. degree in circuit and system engineering from the Beijing University of Technology, Beijing, China, in 2016.

His research interest includes low complexity video coding and low bitrate video coding for AVC and HEVC.

Pengyu Liu received the Ph.D. degree in circuit and system engineering from the Beijing University of Technology, Beijing, China.

She is currently an Associate Professor with the College of Electronic and Control Engineering, Beijing University of Technology, Beijing, China. She has authored or coauthored more than 30 academic papers in the video coding field indexed by SCI/EI. Her research interest includes the development of multimedia information processing and intensive study of video coding technology.



Yueying Wu received the B.E. degree from Beijing University of Technology, Beijing, China, in 2014, where she is currently working toward the M.E. degree in information and communication engineering.

Her research interest includes video coding.

Kebin Jia received the Ph.D. degree in signal and information processing from the University of Science and Technology of China, Hefei, China.

Since 1998, he has been with the College of Electronic and Control Engineering, Beijing University of Technology, Beijing, China. He is responsible for the national 973 project, national science support plan (sub-topic), and National/Beijing Natural Science Foundation projects. He has authored or coauthored more than 150 SCI/EI papers and published two monographs. His research interests includes image/video content retrieval technology, 3D video fast coding, transcoding and processing technology, internet-based multimedia information processing technology, and biomedical image processing technology.