Scientia Iranica B (2015) 22(6), 2209–2222

SCIENTIA IRANICA

Sharif University of Technology
Scientia Iranica
Transactions B: Mechanical Engineering
www.scientiairanica.com

Temporally correlated quadtree partition algorithm for fast intra coding in high efficiency video coding

Y.-C. Lin* **and J.-C. Lai**

Department of Computer Science and Information Engineering, National Formosa University, Yunlin, 63201, Taiwan.

Received 22 May 2014; received in revised form 9 May 2015; accepted 12 October 2015

KEYWORDS
High efficiency video coding;
Fast coding-unit algorithm;
Fast encoding algorithm;
Quadtree partition.

Abstract. HEVC is a new video coding standard, which has been developed by the JCT-VC (Joint Collaborative Team on Video Coding) for applications with ultra high definition video coding. In HEVC, there are three types of encoding block: the Coding Unit (CU), the Predict Unit (PU) and the Transform Unit (TU). If every block of video pixels goes through the complete mode decision process, a large amount of encoding time is required, limiting its applicability in real-time applications. This paper proposes a fast algorithm that can reduce significantly the computation time of intra coding for each Coding Unit (CU) block. In the proposed algorithm, depth information of the Coding Block Tree (CBT) from the previously encoded frame is used to help determine the current CU split depth, which can skip many unnecessary mode computations for some cases of block decomposition. Experimental results by HM10.1 show that the proposed method can provide about 20% time-saving and maintain a negligible degradation of coding efficiency.

1. Introduction

The High Efficiency Video Coding (HEVC) proposal was established by two international organizations, namely ITU-T VCEG and ISO/IEC MPEG, developed in April 2010. The first formal edition of the HEVC was released in January 2013, becoming the newest generation video coding standard [1], with a major goal towards high definition video content compression. Different from H.264/AVC, the HEVC adopts a quad-tree based coding structure, which is a recursive block-splitting for seeking optimal coding efficiency. In HEVC, there are three types of block: CU, PU and TU, respectively. Among them, CU is the basic coding unit, whose block size varies from the 64×64 Largest Coding Unit (LCU) to 8×8, which can then be formed into PU and TU. PU is the prediction unit for deciding the prediction modes, whose sizes vary from 64×64 to 4×4 for prediction modes ranging from intra coding to various inter-frame coding modes [2-3]. TU represents the transform unit that is used to define the transform shapes and quantization parameters for the prediction residue by the PUs in a CU, whose block size can range from 32×32 to 4×4. If the encoder needs to calculate all the combinations of PUs and TUs for each incoming CU to get the best coding efficiency, it makes the encoder more computationally complex. Thus, it limits commercial applications for the HEVC encoder. Therefore, computational optimization of the encoder is also a necessary topic in the design of practical HEVC encoders. Recently, many fast algorithms have been proposed for the HEVC encoder. Refs. [4-9] proposed fast algorithms to reduce the number of CU quadtree partitions, which can skip computation for some unnecessary predictions for an incoming CU.

*. *Corresponding author. Tel.: +886-5-6315576; Fax: +886-56330456*
E-mail address: lyc@nfu.edu.tw (Y.-C. Lin)

Refs. [6,7,10-12] mention many approaches to reduce the computation complexity of angular intra prediction modes. Their methods focus on finding the best prediction mode to determine which mode is the most probable mode. Methods in [13-15] propose fast inter-frame prediction mode decision algorithms to terminate the search procedure early and skip unnecessary prediction modes.

The aforementioned methods can also be classified as reference-neighbor and non-reference-neighbor methods. The reference-neighbor methods make use of adjacent block coding information for exploiting the high correlation in continuity videos. The continuity video does not have drastic scene-changes or fast object-moving phenomena. Therefore, it has a better prediction and time-saving for static objects or slow-moving videos. On the other hand, the non-reference-neighbor methods use the variation of frame content complexity to predict the partition of the current block and mode prediction. This kind of method relies on a set of decision thresholds each of which plays an important role in the proposed fast method. Due to referring to no other encoded blocks, they are not affected by the encoded neighboring blocks, and, thus, have the advantages of the independent encoding model. Hence, the coding efficiency would not be affected significantly by drastic scene-change and fast object-moving effects.

In this paper, the proposed method belongs to the class of reference-neighbor methods. It is an extension of the previous work [16] in exploiting the information from the partition of the co-located CBT in the previous frame to make a good estimate of the possible range of partition depth-level on the current CBT block. Correlation of the temporally co-located quadtree of partition CUs or CBTs is formulated and analyzed thoroughly with a large set of benchmark videos. In addition, a detailed description of the proposed fast HEVC intra coding algorithm is given with an explicit procedural pseudo code. Based on our experimental observations, there is a higher correlation of partition-tree depth between those co-located blocks in most benchmark videos. It is anticipated with high confidence that a simple and effective fast intra coding process would be developed when making use of this kind of correlation.

This paper is organized as follows: Section 2 explains some key features of the HEVC encoder and illustrates the procedure of the CBT coding process for HEVC intra coding. Section 3 explains how the depth correlation is measured and justifies the feasibility of using correlation information. Section 4 details the operational flow of the proposed method. Section 5 demonstrates the feasibility of computation by experiments. Concluding remarks are given in Section 6.

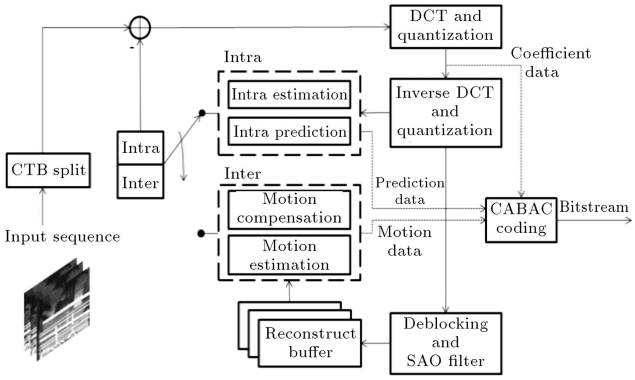


Figure 1. Block diagram of HEVC encoder.

2. HEVC coding process and fast algorithms

This section first describes the HEVC encoder process, including the introduction of HEVC intra coding, the procedure of Rate Distortion Optimization (RDO), and the quadtree partition of CUs. Then, two classes of fast HECV encoding algorithms are reviewed briefly.

2.1. HEVC encoder

Figure 1 shows the coding process of a HEVC video encoder. Firstly, the incoming frame should be divided into CBTs as the basic coding elements. In general, each CBT is processed for considering both intra and inter prediction coding. Then, the prediction residues are generated for the subsequent coding modules, including transformation, quantization and entropy coding. The family of Discrete Cosine Transforms (DCT) is used for the prediction residues, which outputs the quantized transform coefficients for the following CABAC entropy coding. In the reconstruction loop of the encoder, the inverse DCT, inverse quantization, and post-processing filters are employed to generate the corresponding reconstructed frame to the original incoming one. The purpose of designing the HEVC encoder is to obtain the comparable reconstruction quality of the previous standard, H.264/AVC, while only a half bit-rate is required by the HEVC standard.

2.1.1. Coding units

In HEVC encoding, the role of CTB is similar to the macroblock in H.264, that is, a basic coding block. Each CTB should be further split into a Coding Unit (CU), a Prediction Unit (PU) and a Transform Unit (TU).

A coding unit is a coding block with a square partition in each CTB, which is partitioned from 64×64 to 8×8 by a recursive process. Each block size corresponds to a depth level of 0 to 3. An example of a CU partition is shown in Figure 2. The depth level of 0 is an initial CU partition in CTB whose block size is 64×64. Then, it should be divided into four equal-sized CU blocks of 32×32 pixels by a recursive process, and each 32×32-block is further divided into 16×16-

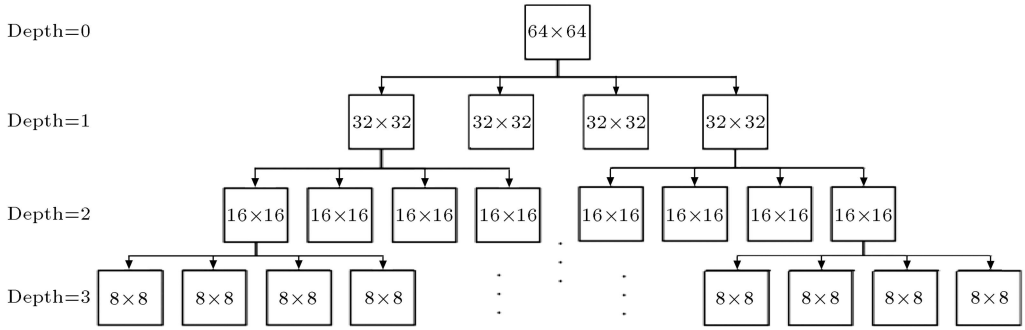


Figure 2. Quadtree structure of partitioning an incoming LCU.

block, and so on, until the maximum depth level of 3 is reached. The CU block is applied to intra and inter coding as basic elements for the subsequent Prediction Unit (PU) and Transform Unit (TU) partition.

2.1.2. Prediction units

A Prediction Unit (PU) is the basic unit of block prediction where one or more PU partition blocks are composed of a single CU. Each PU adopts the block prediction from neighboring reconstructed pixels by the intra or inter prediction modes, and the possible block sizes are between 64×64 and 4×4. Intra prediction uses only square PU blocks, which means that the partition is of a $2N \times 2N$- or $N \times N$-type. Note that the PU-$N \times N$ block is only used for the intra coding of the CU 8×8 block. In inter prediction, in addition to the squared block types, the symmetric and asymmetric partition blocks of type $2N \times N$, $N \times 2N$, $2N \times nU$, $2N \times nD$, $nL \times 2N$ and $nR \times 2N$ are also included. Figure 3 depicts the types of PU block for SKIP, intra, and inter coding modes.

2.1.3. Transform units

A Transform Unit (TU) is used in the transform and quantization process, which are performed on the residual of the PU block prediction. In the HEVC standard, the TU blocks are only partitioned to square blocks, with sizes ranging from 32×32 to 4×4 partitions, e.g. 32×32, 16×16, 8×8 or 4×4. In general, to reduce the complexity of the encoder, the maximum partition depth of TU is only three depth levels. For example, if the CU is of size 32×32, the TU partition of the CU only involves three depth levels, e.g. 32×32, 16×16 and 8×8 partition blocks of TU.

2.2. HEVC intra coding

For each incoming LCU, its intra coding process involves a series of operations, including a quadtree structure decomposition, intra-frame prediction, and residual quadtree transform and quantization. In the quadtree decomposition, the incoming LCU is recursively partitioned from the size of 64×64 pixels into the size of 4×4 pixels. Each node in the partition quadtree, which corresponds to a sub-block of the LCU under consideration, is treated as a PU, and a predicted block is generated for it using a set of available prediction directions. There are 35 prediction modes defined in the HEVC standard. Figure 4(a) depicts the directions of these prediction modes. To get better coding efficiency, the rate-distortion cost of encoding any node in the quad-tree structure is calculated by Eqs. (1) and (2), and the prediction direction with the minimum cost, J, is selected as the best intra-prediction mode for generating the predicted block. In Eqs. (1) and (2), D represents the prediction residual signal of the current block; λ is called the Lagrange multiplier whose value depends on the values of QP used in the encoder; and R is the number of bits

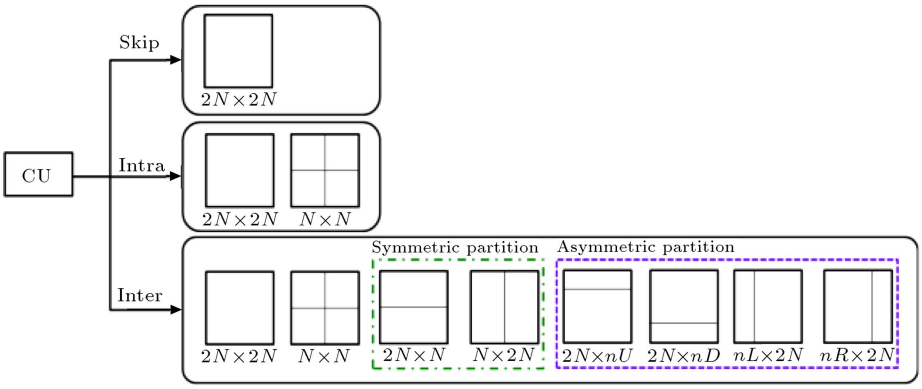


Figure 3. Types of PU partition block for SKIP, intra, and inter coding modes.

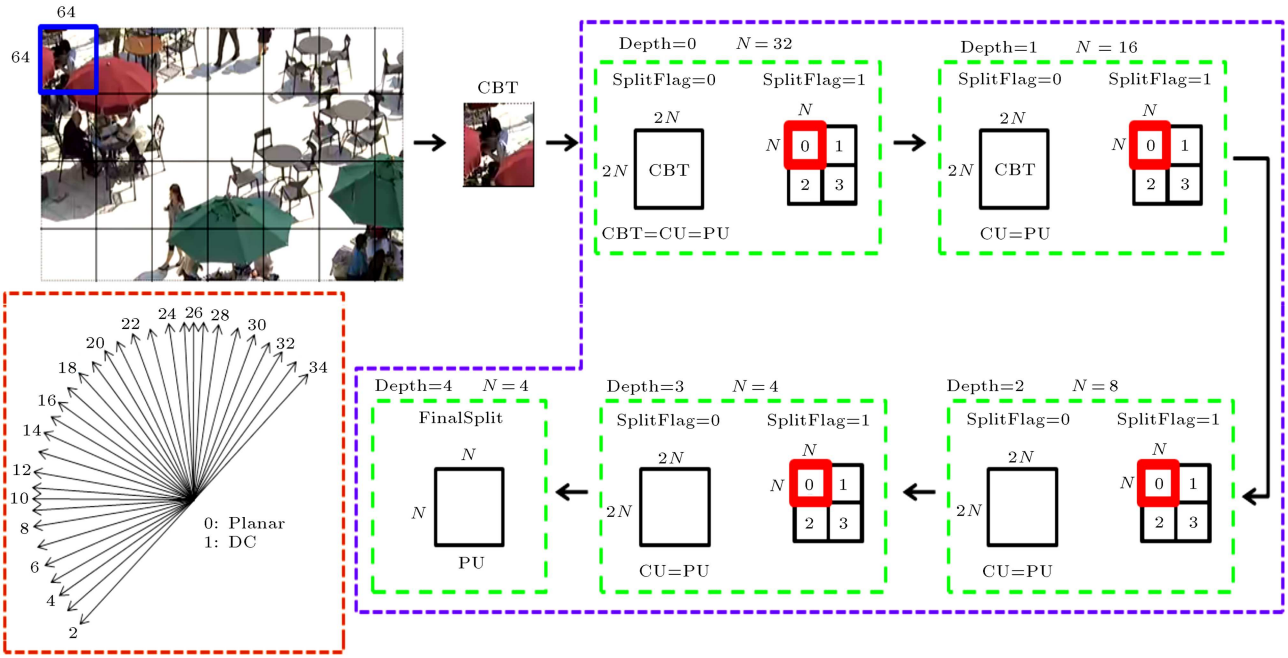


Figure 4. Illustration of CBT coding structure for HEVC intra coding: (a) 35 prediction modes for intra prediction; and (b) recursive partition process for each CBT.

used to represent the current CU block.

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

$$\lambda = 0.85 * 2^{(\mathrm{QP}-12)/3}. \tag{2}$$

However, to reduce the computation load of choosing the optimal rate-distortion prediction mode, a light-weight cost criterion is derived to determine a subset of the prediction modes. This is called the Rough Mode Decision (RMD) set, and the actual Rate-Distortion Optimized (RDO) intra-prediction mode is decided from the RMD set. The size of the RMD set varies with the size of the block. Table 1 lists the valid sizes of the RMD set defined in the HEVC specifications.

The generation of the quadtree structure for the LCU (also called CTB) involves recursive decomposition, starting from the tree root (corresponding to the incoming LCU) down to the minimum allowed block size, which is the 4 × 4 block in HEVC intra coding.

Table 1. Number of prediction mode for PU block.

PU size	Number of prediction mode (N)
64 × 64	3
32 × 32	3
16 × 16	3
8 × 8	8
4 × 4	8

Figure 4(b) illustrates an example of decomposing a 64 × 64 block from a video frame as a quadtree block structure until the fourth level of partition depth. HEVC defines a binary syntax element, **SplitFlag**, to indicate if the current block is partitioned or not.

2.3. Quadtree partition process

As can be seen in Figure 2, in the HEVC intra coding mode, the input LCU should be partitioned into a quadtree structure with variable-sized CU blocks, while each CU in the quadtree is encoded with the RDO process during the quadtree partition. Each node in the quadtree shown in Figure 2 corresponds to one image block in the incoming LCU. Figure 5 illustrates the numbering scheme for the partitioned blocks with the pair (i, j) of integers associated with the jth node at the ith depth level of the quadtree structure shown in Figure 2. For example, image block (1, 3), which is located at the bottom-right quadrant of LCU, stands for the 4th node at the 1st depth level of the quadtree shown in Figure 2.

The HEVC quadtree partition process for each LCU consists of two operations:

Splitting operation: Suppose the size of a LCU is set to 64 × 64 pixels. An incoming 64 × 64 Coding Block (CB) is first dealt with by intra-prediction coding, and the RDcost and the related prediction information for this CB are recorded as the root node of the quadtree structure. Afterwards, the incoming CB is partitioned into four 32 × 32-CBs, each of which is treated as a child node of the root node and

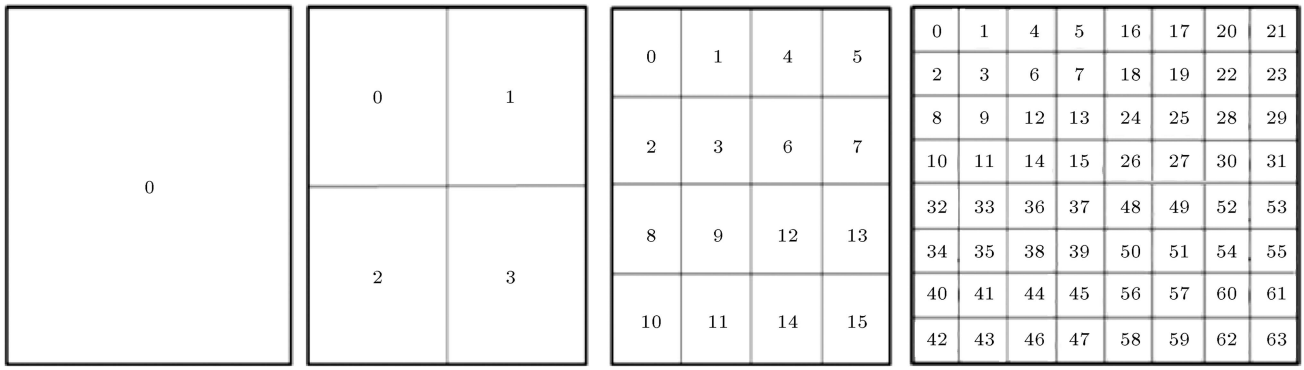


Figure 5. Block numbering scheme on the image blocks at different depth levels of partition.

further undergoes the same intra-prediction process to store the corresponding prediction information and RDcost. Then, each of the nodes for the 32 × 32-CBs also spawns its own four child nodes for the corresponding 16 × 16-CBs using the same intra-prediction and RDO process. This tree-growing process is repeated until the lowest 4 × 4-CB is reached. At last, a complete quadtree structure is available with information for all possible variable block-size predictions and RDO data.

Pruning operation: To get the best partition tree, starting from the 8×8-CB nodes, each node is checked to see if pruning their four child nodes enables a better RDcost. If the sum of RDcosts for all four child nodes is smaller than that of their parent node, the child nodes are kept by setting its **SplitFlag** to 1, and the new RDcost of the parent node is set to the sum of the children RDcosts; otherwise, the four child nodes are removed and the parent node becomes a leaf node by setting its **SplitFlag** to 0. This process is applied to each non-leaf node in the complete quadtree and is executed in a bottom-to-up manner. Having once examined the root node, the complete quadtree would be pruned to become a RDO intra-coding quadtree structure.

Referring to the quadtree structure in Figure 2, the encoder needs to deal with 85 CUs, 341 PUs, and 3 or 8 TUs for each CU to finish the intra coding of each LCU, and thus requires a huge amount of arithmetic computation. In the subsequent sections, a fast HEVC intra coding process is developed with the exploitation of temporal correlation of the quadtree structure between successive video frames.

2.4. HEVC fast encoding methods

In most of the current fast HEVC encoding methods, the speeding up strategies can be mainly divided into block-based and mode-based decision categories. The block-based method is used to reduce the block partitions. Lee et al. [4] proposed to skip the partition of the current CU in terms of block content complexity from those previously encoded blocks. Therefore, partition of the current block is determined from the previously encoded frames. This method could achieve time-saving and maintain good coding efficiency. In order to achieve an accurate prediction of splitting, it tries to remove the furthest depth level in a CBT. Hence, this method has a limited capability of time-saving. Cho and Kim [5] employed machine learning to decide whether CU splitting was terminated or not. It should create actively the best time-saving and video compression. But this method takes further time to train the prediction model. Lee et al. [4] and Cho and Kim [5] aimed at the determination of current block partitioning based on previously encoded frames. If the frame contains a scene-change or fast object movement, an inaccurate prediction for the current frame might probably occur. Shen et al. [6] presented references to neighboring blocks to predict the possible current split of a block. They proposed a classification algorithm to decide whether CU needs to be computed for rate-distortion cost or not. Cheng et al. [7] introduced a reference to the neighboring RDcost method to early terminate the CU partition, which is able to provide a simple partition process for CUs at the slowly varying region in the frame. The methods in [6,7] are not limited by previously encoded frames. Hence, they could have better prediction for change scene and fast moving sequences. However, it requires information from the neighboring reference blocks to predict the partition structure.

In addition to the CU block-partition decision methods, there are also some PU block decision methods proposed in the literature. Cheng et al. [8] used a bilinear interpolation for approximating the intensities of the PU block in order to predict the block complexity of intensity and determine the type of PU block division. Yoo and Suh [9] proposed a fast method on intra- and inter-frame coding structures that use the RDcost to derive some early termination thresholds. With the thresholds, if current RDcost is lower than the thresholds, then, the encoder can avoid further calculations on the block partition and coding process.

The mode-based methods concern the derivation of decision rules to improve the speed of calculating the prediction modes from PU blocks. They could be divided into two types of decision rules for the intra and inter mode decisions, respectively. In the intra mode decision method, the goal is primarily towards the improvement of performing the 35 prediction modes for the full RDcost. Shen et al. [6] and Cheng et al. [7] proposed to use the neighboring coding information for complexity reduction of intra prediction for the current PU block. Cheng et al. [7] employed the parent PU prediction mode to speed up calculations for the best PU prediction mode of the current PU block. Shen et al. [6] improved the performance of the method developed in [7] by adding a Rough Mode Decision (RMD) rule and the Most Probability Mode (MPM) to enhance the accuracy of PU prediction. Jiang et al. [10] and Silva et al. [11] utilized the gradient variation to find a more accurate prediction mode. Differing from methods proposed in [6,7,10,11], which endeavor to reduce the number of available prediction modes for Rate-Distortion Optimization (RDO), Motra et al. [12] introduced the reduced RMD prediction mode to replace the full mode prediction for RMD. For inter coding rules, Yang et al. [13] employed the SKIP mode from the inter $2N \times 2N$ PU to skip Advanced Motion Vector Prediction (AMVP) in the inter prediction coding process. For speeding up motion estimation, Pumachand et al. [14] proposed a new search method to improve the accuracy for inter coding, and added detection thresholds to early terminate the search procedure. Similar to [14], Pan et al. [15] proposed an improved TZSearch algorithm to reduce the motion estimation execution time for predicting the current PU block.

3. Correlation analysis of CBT partition

3.1. CBT partition depth

The structure of the quadtree with the best RDO coding for the input LCU block depends highly on the content complexity of the LCU block and its spatial neighboring pixels in a frame. Further, the content complexity of one frame is also very similar to its immediately neighboring frames in a video sequence. Therefore, it is intuitive to infer that the quadtree structure for a LCU block in a frame should be very similar or even identical to the co-located ones in the temporally neighboring frames. In this section, the correlation of quadtree partition-depth between consecutive frames is analyzed for its usefulness in improving the speed of decomposing a LCU block. Figure 6 demonstrates an example for this correlation of quad-tree partition-leveling between the current frame (Figure 6(b)) and its previous frame (Figure 6(a)). The quadtree partition, as indicated by a red-dotted square in Figure 6(b), is found to be exactly identical to its

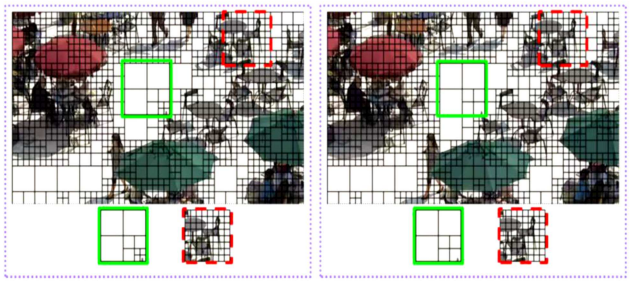


Figure 6. Examples of CBT correlation between video frames: (a) Previously encoded frame; and (b) currently encoded frame.

counterpart co-located in the previous frame indicated in Figure 6(a). In addition, there is another case that shows a similar quadtree partition structure to those indicated by the green-solid square in Figure 6.

3.2. Correlation of CBT depth between frames

Once one frame has been coded by the intra-coding process, the coded frame can be viewed as a union of the non-overlapped, variable-sized CBs that correspond to the nodes in the quadtree structures generated by the intra-coding process. Since the size of each CB is specified in the power of two, say $2^{6-j} \times 2^{6-j}$, $j = 0, 1, 2$ or 3, each CB can be segmented into a set of non-overlapped 4×4-blocks. Each 4×4-block in a CB of size $2^{6-j} \times 2^{6-j}$ can be given a depth value, j, to carry the quadtree depth information about this 4×4-block. When combining this depth information of 4×4-blocks from a coded frame, a two-dimensional array, $d(m, n) = j$, is obtained for the 4×4-blocks in a coded frame. Figure 7(a) illustrates the sample quadtree partition results for a frame of size 416×240 pixels, and highlights a quadtree partition with a green box at position (41, 17) (in units of 4×4-block) in the frame. Figure 7(b) shows the value of $d(m, n)$ for the highlighted quad-tree partition in Figure 7(a).

Suppose $d_c(m, n)$ and $d_p(m, n)$ denote the quadtree depth information arrays for the current coded frame and its immediately previous frame, respectively. The frame difference of quadtree depths can be calculated by Eq. (3):

$$\Delta D(m, n) = |d_c(m, n) - d_p(m, n)|. \tag{3}$$

In Eq. (3), (m, n) denotes the 4×4-block coordinate in a frame. To measure the similarity of depth information between consecutive frames, the ratio of frame depth difference is defined in Eq. (4), which computes the similarity information based on the frame difference, $\Delta d(m, n)$, being less than one level:

$$\text{FDR} = \frac{\sum_{m=0}^{\frac{M}{4}-1} \sum_{n=0}^{\frac{N}{4}-1} f(m, n)}{M \times N} \times 16 \times 100\%, \tag{4}$$

where:

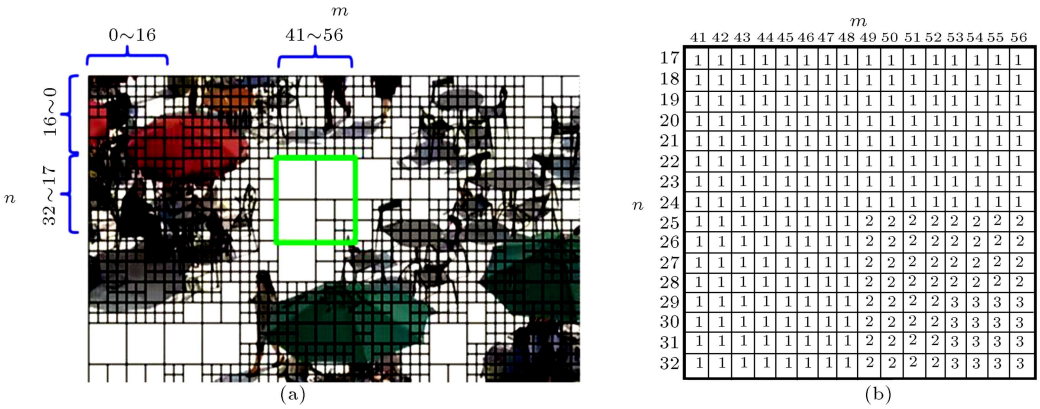


Figure 7. Representation of the partition depth of each CU.

$$f(m,n) = \begin{cases} 1 & \text{if } \Delta d(m,n) \le 1 \\ 0 & \text{otherwise} \end{cases}$$

In Eq. (4), FDR denotes the ratio of frame depth difference for a single frame in a video sequence. Table 2 illustrates the average values of quadtree correlation computed for a large set of test sequences. Each value shown in the table is computed using Eq. (4) and averaged over all the frames in the sequence encoded by HM using QP values ranging from 22 to 37. The video sequences that were used include those test sequences specified in the HEVC call for proposal document [17], with different classes for the test constraints. The resolution of frames for videos in Classes A, B, C, and D decrease from '2560 × 1600', '1920 × 1080', '832 × 480', and '416 × 240', respectively. Videos in Class E are recorded from videoconference applications, having a resolution of '1280 × 720' for each of them. The 'ChinaSpeed' sequence is a non-camera video, which is captured from a computer game scene, with display resolution of '1024 × 768'. Based on the computed results, it can be found that for most videos, the quadtree partition depth is very similar to the temporal neighboring frames, with over 96% area in the coded frame having a deviation of quadtree depth within one level. In the next section, a fast quadtree partition algorithm based on this observation will be designed.

4. Proposed method

4.1. The concept of how to speed up the quad-tree partition process

In the proposed method, simplification of quadtree partition is achieved by exploiting the quadtree depth information from the temporally co-located CBT in the same sequence to make a rough partition range for the current LCU. For example, Figure 8(a) shows the quadtree partition of the co-located CBT quadtree with depth 0. With the observation in Table 2, the possible depth range of the corresponding quadtree

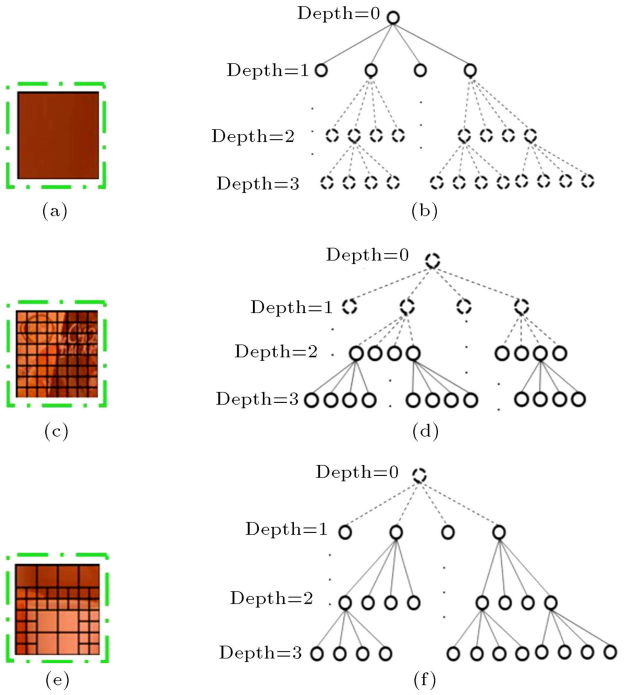


Figure 8. Examples partition of CTB by referring the partition depth of co-located CTB.

of LCU in the next frame is highly likely to be increased by one, as the partition tree with solid lines shows in Figure 8(b). Therefore, it can skip the RDO computations for the CU at levels 2 and 3 (e.g. Figure 8(b), dotted line). Figure 8(c) shows another sample partition for the co-located CBT with highly complicated content, such that the maximum partition depth grows up to 3. The possible range of partition depth for the current LCU can only be set to between 2 and 3, in this case (see Figure 8(d), solid line). Figure 8(e) illustrates a case that does not give much reduction of computation to the proposed method, where the partition tree spans from depth 1 to depth 3 (see Figure 8(f), solid line), since the proposed method can only skip the computation of depth 0 partitioning.

Table 2. Quadtree depth correlation between temporally neighboring coded video frames.

Class	Sequences	Temporal depth correlation (%)			
		QP 22	QP 27	QP 32	QP 37
A	PeopleOnStreet	95.9	95.8	96.1	96.2
	Traffic	96.1	96.2	95.7	95.1
B	BasketballDrive	93.8	94.3	94.8	95.6
	BQTerrace	95.4	96.7	96.7	96.2
	Cactus	93.6	96.2	96.0	96.0
	Kimono1	94.5	96.0	95.8	95.7
	ParkScene	93.9	94.8	94.7	94.2
	Tennis	92.6	93.2	93.7	94.6
C	BasketballDrill	98.5	95.5	95.5	96.0
	BQMall	97.8	97.5	97.3	96.8
	Flowervase	97.8	98.5	98.5	98.6
	Keiba	91.7	91.5	91.6	92.3
	Mobisode2	95.6	96.6	97.4	97.9
	PartyScene	99.6	99.4	99.2	98.6
	RaceHorses	94.5	94.9	95.4	95.5
D	BasketballPass	97.6	97.9	98.1	97.8
	BlowingBubbles	99.8	99.7	99.4	98.8
	BQSquare	98.1	98.5	99.5	99.5
	Flowervase	98.9	98.8	98.5	98.3
	Keiba	93.6	94.1	94.1	93.8
	Mobisode2	97.6	97.9	97.6	98.0
	RaceHorses	98.4	98.1	97.6	97.5
E	FourPeople	97.5	97.7	97.8	97.4
	Johnny	96.0	98.3	98.6	98.6
	KristenAndSara	96.0	97.6	98.2	98.4
	SlideEditing	99.5	99.4	99.5	99.4
	SlideShow	97.8	98.0	98.1	98.2
	Vidyo1	96.4	96.8	96.8	97.6
	Vidyo3	97.1	97.3	97.2	97.3
	Vidyo4	96.1	97.3	97.3	97.5
Other	ChinaSpeed	97.5	97.5	97.3	97.5
Average		96.8			

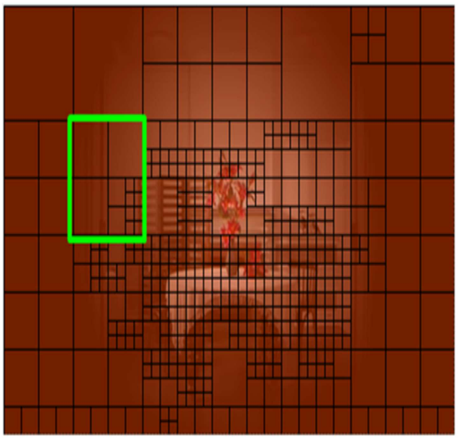
4.2. Estimation of quadtree depth

In the proposed method, the first frame of each sequence is coded with the HM10.1 encoder. The purpose is to generate accurate reference information for the succeeding frames in the sequence that is designated to be coded by the proposed method. When encoding the succeeding frames, the proposed fast algorithm first extracts the depth partition information, $d_p(m,n)$, of the co-located CBT, each of which records the depth value of a 4×4-block at location (m,n) in the co-located CBT. Figure 9(b) shows an example of $d_p(m,n)$, storing the depth values of the co-located CBT partition, highlighted with a green line box in Figure 9(a).

Among the values of $d_p(m,n)$, the maximum depth (*MaxDepth*) and minimum depth (*MinDepth*) are first obtained. Then, the depth range for the current LCU, the upper ($D_{\max}$) and lower ($D_{\min}$) limits, are determined by Eq. (5). The rationale of Eq. (5) reflects the facts observed in Section 3, that is, the possible depth range of the current LCU is most probably equal to the range expanded by the +1/-1 depth from the range of the co-located LCU:

$$D_{\max} = \text{Min}(\text{MaxDepth} + 1, 3),$$
$$D_{\min} = \text{Max}(0, \text{MinDepth} - 1). \quad (5)$$

On applying the upper and lower limits for speeding



(a)

$d_p(m,n)$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	2
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3
1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3

(b)

Figure 9. Example of co-located CBT depth array: (a) Previously encoded frame; and (b) co-located CBT depth $d_p(m,n)$.

up the quadtree partition of CUs in intra coding, a fast quadtree partition function is described in the following subsection. It can be used by invoking the function call, **Fast_Quadtree_Partition**$(0, D_{\min}, D_{\max}, 0)$, with arguments, including starting level 0, minimum depth value $D_{\min}$, maximum depth value $D_{\max}$, and block number 0.

4.3. Quadtree partition procedure of the proposed algorithm

In general, the CU partition and coding is executed recursively by splitting a lot of sub-CU blocks. We propose a method that utilizes a depth correlation property to decrease encoding time complexity. Figure 10 shows the flow chart of the proposed method when encoding the current CBT.

The proposed method is divided into two parts: First, it generates the estimated depth range for a partitioning current CBT block, based on information from the co-located CBT block. In the second part, the quadtree partition process is carried out with maximum and minimum levels of partition depth. Unlike the HEVC CB partition process, the proposed algorithm limits the number of CU block to be processed for intra prediction of intra coding.

The splitting process of each block is recursively done from top to bottom and left to right; the pruning process is run from bottom to top and left to right during quadtree computation. During the splitting process, the RDcost for the jth CU at depth level i of the quadtree partition is evaluated by the intra coding function of the HM encoder, and is stored in $Cost(i,j)$ for later access by the pruning operation. If the level of the current block partition is smaller than $D_{\min}$, computation for the intra coding is skipped and the values of RDcost for these nodes are assigned with infinity. When the partition depth is located between the minimum depth level, $D_{\min}$, and maximum depth level, $D_{\max}$, the algorithm adopts the original HEVC process to make the cost values until the splitting process reaches depth level $D_{\max}$.

```
Algorithm Fast CU quadtree partition

Fast_Quadtree_Partition(level, D_min, D_max, n)
Begin
1. Initialize the working variables:
   i=level; j = n;
2. If(i < D_min)
      Cost(i, j) = ∞;
   else
      Cost(i, j)=Intra_Coding(i, j);
   Endif
3. If(i < D_max)
      For k := 0 to 3
         Fast_Quadtree_Partition (i + 1, D_min, D_max, 4j + k);
      Endfor
   else
      SplitFlag(i, j) = 0
      Return;
   Endif
4. SubCost=0;
   If (i < D_min)
      SplitFlag(i, j) = 1;
   else
      For k := 0 to 3
         SubCost+=Cost(i + 1, 4j + k);
      Endfor
      If(Cost(i, j) >SubCost)
      SplitFlag(i, j) = 1;
         Cost(i, j)=SubCost;
      else
         SplitFlag(i, j) = 0;
      Endif
   Endif
End
```

Figure 10. The pseudo code description of the proposed fast quad-tree partition algorithm.

In the pruning process, RDcost $Cost(i,j)$ is updated with the smaller one of the current CU's RDcost or the sum of RDcosts of its four children nodes. If all the children blocks of a parent block have been evaluated with their RD costs, the pruning process is invoked to update the value of $Cost(i,j)$ for the parent block. If the sum of the children RDcosts is smaller than that of the parent block, the **SplitFlag** of the

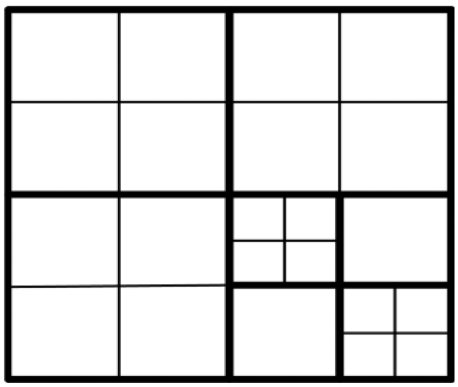


Figure 11. Co-located CBT from previously frame.

parent block is set to 1, and the parent block cost is set to the sum of the children RDcosts; otherwise, the **SplitFlag** is set to 0 and there is no update to the parent block RDcost. Afterwards, the algorithm keeps running recursively and checks whether the current process node is the root or not; if yes, the algorithm stops the encoding process on the incoming LCU, and gets the SplitFlag(i, j) array for indicating the best partition of incoming LCU.

4.4. Example of the proposed quadtree partition

This proposed method can be used to skip unnecessary block decomposition computation. For example, when considering the co-located CBT from the previous frame, as shown in Figure 11, the proposed algorithm can find out that $D_{\min}$ and $D_{\max}$ are 3 and 2 by Eq. (5), respectively. Since, in the HEVC standard, the CU block maximum depth is 3, the depth range of partitioning the current CBT during the RDO process is from 1 to 3. Therefore, the algorithm can skip the computation for depth-0 partition and the associated RDO computation to reduce computation complexity in this situation. In this case, RDO process starts from level 1 of the quadtree partition to undertake RDO computation on each 32×32 CU. Then, each 32×32 CU is handled as a root node and further split into four 16×16-CUs. Then, the algorithm repeats partition and RDO computation until 8×8 CUs are reached. In order to find the best CBT partition, each node is examined to undertake the pruning process based on their four sub-CU RDcosts. If the sum of the RDcosts from the four sub-CUs is smaller than the parent node, then the sub-CUs are kept intact and the sum of the sub-CU RDcosts is used to update the parent node. Otherwise, the sub-CUs are removed from the partition tree and the **SplitFlag** of the parent block is set to 0. When repeating this process for each encoded node until the 32×32 CU partition nodes, the algorithm can terminate the pruning process to generate the best partition of CBT.

5. Experiment results

To verify the effectiveness of the proposed algorithm, experiments are conducted on several test sequences, which are classified into five classes: Classes A to E, according to their own characteristics, are as those illustrated in Table 3. The experimental platform was implemented based on HM10.1 [18] reference software and executed on a computer with Microsoft Windows 7 Professional Service Pack 1 64-bit O/S, Intel(R) Core(TM) i7-4930K 3.4 GHz CPU and 5 GB RAM.

The program that executed the experiments is developed by modifying the HM encoder software to integrate the proposed fast CU partition algorithm. The first frame in each test video is encoded by the experimental program using the original HM intra coding configuration. The purpose of this special treatment for the first frames is to generate good temporally initial reference information for the succeeding frames in the same sequence, while each of the succeeding frames is encoded by the proposed fast partition algorithm. The measurement of time-saving by the proposed fast encoding algorithm is defined in Eq. (6):

$$\text{Time-saving}(\%) = \frac{T_O - T_F}{T_O} \times 100\%, \tag{6}$$

where T_O stands for the execution time by the HM original encoder, and T_O is the execution time by the fast encoding algorithm. The degradation of coding efficiency by the modified HEVC encoder is measured using the BD-Rate and BD-PSNR [19], with respect to the corresponding coding performance of the original HEVC HM 10.1.

Table 4 lists the results of experiments performed on the different sequences with Quantization Parameters (QP) 22, 27, 32 and 37 in an intra only and low complexity case. As indicated in Table 4, the time-saving ability of the proposed method is about 20.34%, on average, at the expense of a 0.17% increase of output BD-Rate. When compared to the method proposed by Lee in [4], the proposed algorithm has an increase of average time-saving of 2.39%, under the compromise of increased BD-Rate of about 0.12% compared with Lee's method. When looking at the individual sequence, almost all the considered sequences have better and comparable time-saving abilities than those of Lee's method, except for few cases on the 'ParkScene' (1920×1080), and 'PartyScene' (832×480) sequences. The reason for slightly less time-saving of the sequences is the drastic change of content that these sequences have, such that a less accurate prediction is able to be undertaken from the previous co-located partition. Usually, these co-located partitions contribute a lot more partition information and cause many smaller partitions in the current quadtree partition by the proposed algorithm. Quadtree partition with many

Table 3. Description of the test sequences.

Class	Sequence	Resolution	Frame count	Frame rate
A	PeopleOnStreet	2560 × 1600	150	30
	Traffic	2560 × 1600	150	30
B	BasketballDrive	1920 × 1080	500	50
	BQTerrace	1920 × 1080	600	60
	Cactus	1920 × 1080	500	50
	Kimono1	1920 × 1080	240	24
	ParkScene	1920 × 1080	240	24
	Tennis	1920 × 1080	240	24
C	BasketballDrill	832 × 480	500	50
	BQMall	832 × 480	600	60
	Flowervase	832 × 480	300	30
	Keiba	832 × 480	300	30
	Mobisode2	832 × 480	300	30
	PartyScene	832 × 480	500	50
	RaceHorses	832 × 480	300	30
D	BasketballPass	416 × 240	500	50
	BlowingBubbles	416 × 240	500	50
	BQSquare	416 × 240	600	60
	Flowervase	416 × 240	300	30
	Keiba	416 × 240	300	30
	Mobisode2	416 × 240	300	30
	RaceHorses	416 × 240	300	30
E	FourPeople	1280 × 720	600	60
	Johnny	1280 × 720	600	60
	KristenAndSara	1280 × 720	600	60
	SlideEditing	1280 × 720	300	30
	SlideShow	1280 × 720	500	20
	Vidyo1	1280 × 720	600	60
	Vidyo3	1280 × 720	600	60
	Vidyo4	1280 × 720	600	60
Other	ChinaSpeed	1024×768	500	30

smaller blocks would normally generate a higher output bitrate for the LCU. However, compared to Lee's method, the proposed algorithm can offer an additional 2.39% time-saving, on average. The reason for this is that the proposed algorithm can provide up to two levels of depth reduction, rather than the single level reduction in Lee's method.

Figures 12 and 13 demonstrate two examples of quadtree partition illustrating the difference of time-saving abilities between the proposed algorithm and Lee's method. For the case shown in Figure 12, when the depth of the previously co-located partition is level 0, which is in a smooth region, the maximum depth

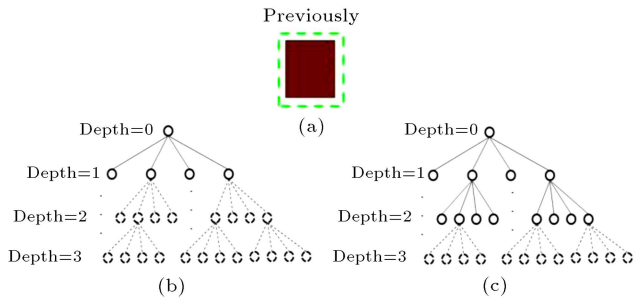


Figure 12. Smooth region time-savings between the proposed method and Lee's method: (a) Co-located CTB partition; (b) the proposed method encoding depth level; and (c) Lee's depth level.

Table 4. Experiment results.

Class	Sequence	BD-Rate (%)		BD-PSNR (dB)		Time-Saving (%)	
		The proposed method	[4]	The proposed method	[4]	The proposed method	[4]
A	PeopleOnStreet	0.02	0.0	0.0	0.0	11.2	13.3
	Traffic	0.06	0.01	0.0	0.0	11.1	12.6
B	BasketballDrive	0.18	0.04	0.0	0.0	20.4	17.0
	BQTerrace	0.02	0.0	0.0	0.0	14.0	15.6
	Cactus	0.07	0.01	0.0	0.0	18.5	16.5
	Kimono1	0.16	0.05	0.0	0.0	35.1	32.2
	ParkScene	0.06	0.02	0.0	0.0	11.9	15.0
	Tennis	0.65	0.21	-0.01	0.0	38.5	31.3
C	BasketballDrill	0.04	0.01	0.0	0.0	10.2	11.0
	BQMall	0.05	0.01	0.0	0.0	12.5	12.2
	Flowervase	0.03	0.01	0.0	0.0	20.9	17.0
	Keiba	0.84	0.22	-0.04	-0.01	23.7	18.5
	Mobisode2	0.37	0.07	-0.01	0.0	36.1	27.6
	PartyScene	0.0	0.0	0.0	0.0	20.0	25.5
	RaceHorses	0.06	0.02	0.0	0.0	18.5	19.9
D	BasketballPass	0.05	0.02	0.0	0.0	15.4	11.1
	BlowingBubbles	0.0	0.0	0.0	0.0	17.7	9.4
	BQSquare	0.0	0.0	0.0	0.0	14.1	9.2
	Flowervase	0.01	0.0	0.0	0.0	14.6	14.7
	Keiba	0.31	0.07	-0.02	0.0	17.9	12.2
	Mobisode2	0.22	0.03	-0.01	0.0	22.8	15.3
	RaceHorses	0.02	0.01	0.0	0.0	16.2	8.9
E	FourPeople	0.02	0.0	0.0	0.0	13.7	13.9
	Johnny	0.06	0.03	0.0	0.0	30.6	27.5
	KristenAndSara	0.06	0.01	0.0	0.0	25.8	25.1
	SlideEditing	0.07	0.03	-0.01	0.0	17.5	14.7
	SlideShow	1.49	0.43	-0.14	-0.04	41.6	35.7
	Vidyo1	0.09	0.0	0.0	0.0	17.2	16.7
	Vidyo3	0.25	0.12	-0.01	0.0	21.7	21.4
	Vidyo4	0.08	0.02	0.0	0.0	18.4	18.0
Other	ChinaSpeed	0.05	0.02	0.0	0.0	22.8	17.5
Average		0.17	0.05	-0.01	0.0	20.34	17.95

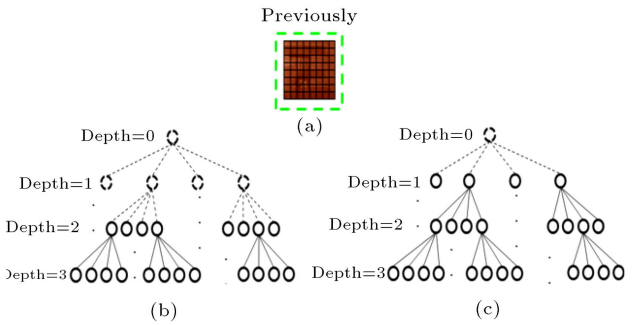


Figure 13. Time-savings of rich texture region between the proposed method and Lee's method: (a) The co-located CTB partition; (b) the proposed method encoding depth level; and (c) Depth level generated by [4].

of the quadtree structure by the proposed algorithm is level 1, whereas the quadtree partition by Lee's method requires up to level 2. There is one more level of partition and computation for Lee's method. As for the example of Figure 13, where the co-located quadtree structure has partitioned blocks with uniform size, the proposed algorithm only computes the intra coding process for the CUs at depth levels 2 and 3, while Lee's method requires computing CUs at levels 1, 2 and 3. Lee's method needs one more level of partition to complete the encoding process of LCU.

With these experiment results, although the output BD-Rate by the proposed method is relatively higher than Lee's method with an insignificant

increase, the proposed algorithm indeed offers a faster quadtree partition process than Lee's method. For example, on the sequences 'Tennis', 'Mobisode2' and 'SlideShow', the BD-Rates by the proposed algorithm are significantly increased, because these sequences have scene-changing effects or fast object motion in the video. This makes a poor prediction from the reference co-located CTB information. Although the correlation analysis is as high as up to 90%, on average, some frames in a video with abrupt motion still have less temporal correlation. Figure 14 shows an example of the 'SlideShow' video exhibiting significant scene-changing effects from the 4th to the 5th frame. Thus, it causes the lower correlation to make the poor partition for the 5th frame, as shown in Figure 14(b).

The advantage of the proposed method lies in the accurate prediction of quadtree partition and, thus, obtains a lot of time-saving for blocks in smooth regions without disrupting motion objects. Figure 15 illustrates an example of smooth region partition. Figure 15(a) shows the previously best splitting of co-located CBT with depth 0 (64×64) partition. Figure 15(c) shows the partition result by the proposed method that only calculates the partition from depth 0 to 1, to get the exact identical partition to that obtained by HEVC HM 10.1 as that shown in Figure 15(b). The proposed algorithm saves the computations for the partitions at depths 2 and 3.

Figure 16 demonstrates one example that reveals the possible disadvantage of the proposed method. In this example, the image content of the current CBT (Figure 16(c)) has a significant difference with the previous co-located CBT. Therefore, when the best

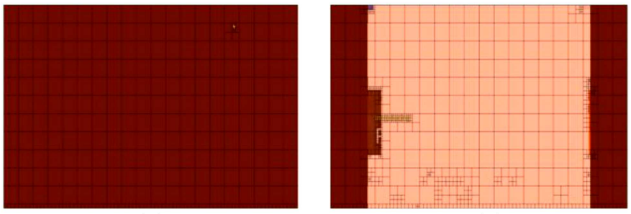


Figure 14. Example of scene changing effect in 'SlideShow' sequence.

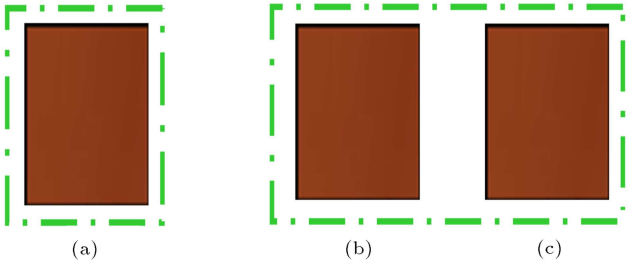


Figure 15. (a) CBT in previously encoded frame. (b) The best split of current CBT by HM10.1 method. (c) The best split of CBT in the proposed method.

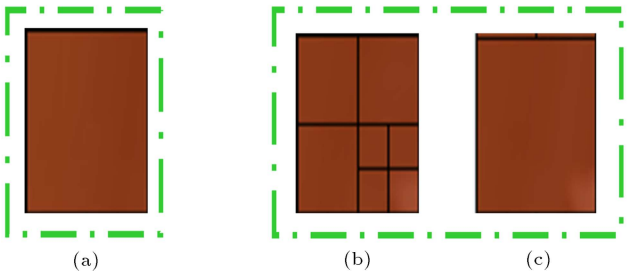


Figure 16. (a) CBT in previously encoded frame. (b) The best split of current CBT by HM10.1 method. (c) The best split of CBT by the proposed method.

partition of the co-located CBT is only at depth 0, which has smooth content, and partitioned as a single 64×64-block (as that shown in Figure 16(a)), the current CBT is partitioned up to depth 2 by the HEVC HM10.1 (see Figure 16(b)), because the content of the CBT is more complicated than that of the co-located CBT. In this case, due to the prediction from the co-located CBT, the proposed algorithm only partitions the current CBT into a single 64×64-block, causing an erroneous prediction of partition depth.

Conclusions

The highly complex computation requirement for the HEVC encoder is the concern of this paper. The temporal correlation of the HEVC quadtree structure between successive frames in a single video sequence is analyzed in this study to seek advantageous information of depth range prior to the actual quadtree partition, in order to reduce the amount of computation incurred in the HEVC encoder. This paper proposes a simple fast algorithm that exploits the co-located quadtree structure in the preceding frame to estimate the possible range of quadtree depth for the current LCU in a frame to improve the speed of HEVC intra coding.

Based on experiments, the primary results of this research can be summarized as follows:

a) The temporal correlation of the quadtree structure existing in a video sequence is really able to help speed up the HEVC encoder, being able to offer over 20% of time-saving, with negligible loss in coding efficiency;

b) The speedup factor using the proposed algorithm becomes more prominent when dealing with videos that are partitioned into blocks with uniform size, i.e. a narrower range of quadtree depth;

c) The computation overhead used to estimate the possible range of quadtree partition depth is very simple, merely requiring the location of maximum and minimum depth values from the co-located LCU in the temporally-previous frame.

References

1. Sullivan, G.J., Ohm, J., Han, W.J. and Wiegand T. "Overview of the high efficiency video coding (HEVC) standard", *IEEE Trans. Circuits and Systems for Video Technology*, **22**(12), pp. 1649-1668 (2012).
2. Lainema, J., Bossen, F., Han, W.J., Min, J.H. and Ugur, K. "Intra coding of the HEVC standard", *IEEE Trans. Circuits and Systems for Video Technology*, **22**(12), pp. 1792-1801 (2012).
3. Piao, Y., Min, J.H. and Chen, J. "Encoder improvement of unified intra prediction", *JCT-VC Document JCTVC-C207*, 3rd meeting Guangzhou, China (2010)
4. Lee, H.S., Kim, K.Y., Kim, T.R. and Park, G.H. "Fast encoding algorithm based on depth of coding-unit for high efficiency video coding", *Optical Engineering*, **51**(6), pp. 1-11 (2012).
5. Cho, S. and Kim, M. "Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding", *IEEE Trans. Circuits and Systems for Video Technology*, **23**(9), pp. 1555-1564 (2013).
6. Shen, L., Zhang, Z. and An, P. "Fast CU size decision and mode decision algorithm for HEVC intra coding", *IEEE Trans. Consumer Electronics*, **59**(1), pp. 207-213 (2013).
7. Cheng, Y., Teng, G., Shi, X. and Li, H. "A fast intra prediction algorithm for HEVC", *9th International Forum on Digital TV and Wireless Multimedia Communication*, Shanhai, China, pp. 292-298 (2012).
8. Qiu, J., Liang, F. and Luo, Y. "A fast coding unit selection algorithm for HEVC", *IEEE Int. Conf. Multimedia and Expo Workshops*, San Jose, California, USA, pp. 1-5 (2013).
9. Yoo, H. and Suh, J.W. "Fast coding unit decision algorithm based on inter and intra prediction unit termination for HEVC", *IEEE Int. Conf. Consumer Electronics*, Las Vegas, NV, USA, pp. 300-301 (2013).
10. Jiang, W., Hanjie, M. and Chen, Y. "Gradient based fast mode decision algorithm for intra prediction in HEVC", *IEEE Int. Conf. Consumer Electronics*, Las Vegas, NV, USA, pp. 1836-1840 (2012).
11. da Silva, T.L., Agostimi, L.V. and da Sliva Cruz, L.A. "Fast HEVC intra prediction mode decision based on EDGE direction information", *20th European Signal Processing Conference*, Bucharest, Romania, pp. 1214-1218 (2012).
12. Motra, A.S., Gupta, A., Shukla, M., Bansal, P. and Bansal, V. "Fast intra mode decision for HEVC video encoder", *IEEE Int. Conf. Software, Telecommunications and Computer Networks*, Split, Croatia, pp. 1-5 (2012).
13. Yang, S., Lee, H., Shim, H.J. and Jeon, B. "Fast inter mode decision process for HEVC encoder", *IEEE IVMSP Workshop*, Seoul Korea, pp. 1-4 (2013).
14. Purnachand, N., Alves, L.N. and Navarro, A. "Fast motion estimation algorithm for HEVC", *IEEE. Int. Conf. Consumer Electronics - Berlin*, Berlin, Germany, pp. 34-37 (2012).
15. Pan, Z., Zhang, Y., Kwong, S., Wang X. and Xu, L. "Early termination for TZSearch in HEVC motion estimation", *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Vancouver, Canada, pp. 1389-1393 (2013).
16. Lin, Y.C. and Lai, J.C. "A fast depth-correlation algorithm for intra coding in HEVC", *National Symposium on Telecommunications*, Tainan, Taiwan, pp. 1-5 (2013).
17. ITU-T Q6/16 and ISO/IEC JTC1/SC29/WG11 "Joint call for proposals on video compression technology", ITU-T SG16/Q6/VCEG-AM91, 39th VCEG, Kyoto (2010).
18. HM10.1 Reference Software [Online]. Available: https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-10.1
19. Bjontegaard, G. "Calculation of average PSNR differences between RD curves", *ITU-T Document VCEG-M33*, Thirteenth Meeting: Austin, Texas, USA (2001).

Biographies

Yih-Chuan Lin received a BS degree in Applied Mathematics from the Chinese Culture University, Taipei, Taiwan, in 1990, and MS and PhD degrees in Electrical Engineering from the National Cheng Kung University, Tainan, Taiwan, in 1992 and 1997, respectively. He is currently a Professor of Computer Science and Information Engineering at the National Formosa University Yunlin, Taiwan. His research interests include computer networks, wireless sensor networks, Internet technology and applications, image/video coding and processing. He is a member of IEEE.

Jain-Cheng Lai received a BS degree in Computer Information and Network Engineering from Lunghwa University, Taoyuan, Taiwan, in 2012, and an MS degree in Computer Science and Information Engineering from the National Formosa University, Yunlin, Taiwan, in 2014. He is currently fulfilling his compulsory military service in Taiwan. His research interests include image processing and video coding.