Local-Constrained Quadtree Plus Binary Tree Block Partition Structure for Enhanced Video Coding

Zhao Wang[1], Shiqi Wang[1], Jian Zhang[1], Siwei Ma[1,2]

[1]*Institute of Digital Media & Cooperative Medianet Innovation Center，Peking University, Beijing, China*

[2]*Peking University Shenzhen Graduate School, Shenzhen, China*

{zhaowang, jian.zhang, swma}@pku.edu.cn; sqwang1986@gmail.com

***Abstract*— In the latest Joint Video Exploration Team (JVET) development, a quadtree plus binary tree (QTBT) block partition structure is proposed for enhanced video coding. Compared to the quadtree block partition in HEVC, QTBT can achieve much better compression performance at the expense of largely increased encoding computational complexity. In this paper, a fast QTBT block partition algorithm based on the local constraint is proposed. Given the decoded information of the previous frame, the partition parameters can be dynamically derived for each coding tree unit (CTU) without any overhead. Furthermore, a constrained binary tree partition algorithm is proposed to reduce the redundancy between the quadtree partition and the binary tree partition. Experimental results have shown that the proposed scheme can speed up QTBT block partition structure by 30% on average without obvious performance decrease, which enables its practical implementations in real application scenarios.**

***Index Terms*— Block partition, quadtree, binary tree, fast partition, local constraint**

I. INTRODUCTION

The High Efficiency Video Coding (HEVC) standard can achieve around 38% bit rate savings by maintaining the same level of objective quality when compared to its predecessor H.264/AVC. Among the newly introduced techniques, the flexible block partition structure has been recognized as the most prominent one that leads to the performance improvement, but has also been charged as the most computationally demanding inclusion.

In HEVC, each frame is divided into equal-sized *Coding Tree Units* (CTU), which are composed of one luminance *Coding Tree Blocks* (CTB) and two chrominance CTBs, and same partition structure decisions are applied to the luma and chroma CTBs [1]. Specifically, each CTU can be divided into smaller blocks called *Coding Units* (CU), following a recursive quadtree partition structure to adapt the local properties of input sequences. The default settings of *Largest Coding Unit* (LCU) size and the *Smallest Coding Unit* (SCU) size are 64x64 and 8x8, respectively. As such, up to four *Coding Tree* depths are possible. In particular, the decision whether to code a picture area using inter- or intra-prediction is made at the CU level. Each CU can be further split into one, two or four *Prediction Units* (PUs) according to the PU splitting type. The same prediction process is applied to all samples inside one PU and the side information is transmitted to the decoder on a PU basis. When the prediction residuals are transformed, each CU is assumed to be the root of another quadtree-based structure called *Residual Quadtree* (RQT). To this end, each CU is recursively partitioned into *Transform Units* (TU), which are the basic units to which transform and quantization process are applied.

The current block partition structure of HEVC has largely outperformed the previous video coding standards [2]. However, potential improvements are possible based on the following observations:

- The CU, which is the granularity for switching intra- or inter-prediction, can only be square and has to follow the quadtree structure, which may lead to the lack of flexibility.
- The PU splitting has only a few fixed types, which may restrict the potentials of the prediction ability.
- The same block partitioning is applied to luma and chroma components simultaneously. However, the properties of luma component are often different from the properties of chroma component.

To address the above issues, a quadtree plus binary tree (QTBT) block partition structure is proposed during the latest Joint Video Exploration Team (JVET) development [3]. In QTBT, the CTU is firstly partitioned using a quadtree structure, and then further partitioned using a binary tree structure. Additionally, a luma-chroma-separated block partition structure is implemented for I slice. Though the QTBT structure provides more flexibilities to better match the local characteristics of video content, the encoding complexity has also been dramatically increased due to more partition iterations are performed.

In this paper, a local constrained QTBT block partition structure is proposed to speed up the encoding decisions. Firstly, a dynamic partition parameters deriving algorithm is proposed to limit the size and depth of the partition trees, especially for the smooth contents. Furthermore, a constrained binary tree partition algorithm is adopted to reduce the redundancy between the attempts of quadtree partition and the binary tree partition. The proposed algorithms are jointly integrated to provide an improved QTBT block partition structure for enhanced video coding.

The remaining of this paper is organized as follows. Section II presents an overview of the QTBT block partition structure. In Section III, the local-constrained QTBT block partition structure is proposed. Simulation results and comparisons are shown in Section IV and Section V concludes this paper.

978-1-5090-5316-2/16/$31.00 ©2016 IEEE

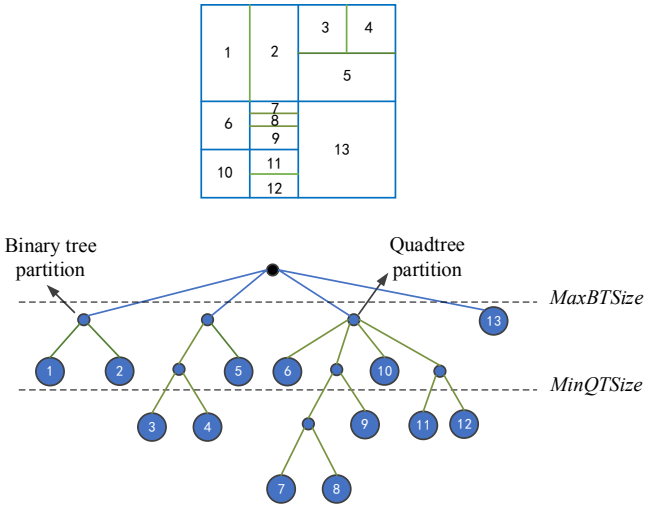

Fig. 1 Illustration of the QTBT block partition structure

II. QTBT Block Partition Structure

In QTBT block partition structure, a CTU, which is the root node of a partition tree, is firstly partitioned by the quadtree, where the quadtree splitting can be iterated until the node reaches the minimal allowed quadtree leaf node size (*MinQTSize*). If the quadtree leaf node size is not larger than the maximal allowed binary tree root node size (*MaxBTSize*), it can be further partitioned by a binary tree. The binary tree splitting also can be iterated until the node reaches the minimal allowed binary tree leaf node size (*MinBTSize*) or the maximal allowed binary tree depth (*MaxBTDepth*). There are two splitting types in the binary tree splitting, symmetric horizontal splitting and symmetric vertical splitting, and the used type is identified at the decoder by a signalled flag. The binary tree leaf node is namely CU which will be used for prediction and transform without any further partition.

Fig. 1 illustrates an example of the QTBT block partition structure, in which the blue lines indicate the quadtree splitting and the green lines indicate the binary tree splitting. The quadtree splitting is firstly applied to CTU to generate quadtree leaves nodes. If the current block size is between *MinQTSize* and *MaxBTSize*, it can be partitioned by quadtree splitting, horizontal splitting or vertical splitting, and the optimal mode minimizing the rate-distortion cost is finally selected. However, if the block has been split by the binary tree, it can not be further split by quadtree, such that only binary tree splitting or termination is allowed. Moreover, a luma-chroma-separated blocking partition structure is proposed for I slice. The luma CTB is partitioned by a QTBT structure into luma CBs, and the two chroma CTBs are partitioned by another QTBT structure into chroma CBs. For P and B slice, the block partition structure for luma and chroma is still shared.

With respect to the split signalling, a quadtree split flag is firstly signalled to represent whether the block is partitioned by quadtree splitting or not. If the quadtree split flag is false or the current block size reaches to *MinQTSize*, a binary tree split flag is further signalled, where 0, 1 and 2 represent not splitting, horizontal splitting and vertical splitting, respectively.

TABLE I
Performance and Encoding Computational Complexity Comparisons for Different Partition Parameters

CFG	CFG1	CFG2	CFG3	CFG4	CFG5	CFG6	CFG7	CFG8
MinQTSize	64	32	16	8	16	16	16	16
MaxBTSize	64	64	64	64	32	16	64	64
MaxBTDepth	4	4	4	4	4	4	5	6
BD-rate	19%	-0.2%	-4.6%	-4.5%	-4.5%	-3.7%	-1.2%	-0.2%
Enc Time	78%	137%	158%	181%	129%	104%	213%	237%

III. Local-Constrained Quadtree Plus Binary Tree Block Partition Structure

Though QTBT can provide more block partition possibilities to improve the flexibility, the encoding computational complexity has also been dramatically increased. In this section, a local-constrained QTBT block partition structure is proposed to speed up the encoding process while maintaining the coding performance.

A. Dynamic Partition Parameters Deriving

In the QTBT structure, the partition parameters, mainly including *MinQTSize*, *MaxBTSize* and *MaxBTDepth*, are playing a crucial role in balancing the coding performance and the encoding computational complexity. Table I presents the simulation results for different partition configurations compared to the HEVC reference software. For CFG1, where only up to 4 depths binary tree partition is allowed, it is observed that approximately 19% Bjontegaard delta rate (BD-Rate) loss is induced, though 22% encoding time saving can be achieved. For CFG2 and CFG3, where *MinQTSize* is reduced while *MaxBTSize* and *MaxBTDepth* keep unchanged, it is observed that the coding performance keeps increasing, as well as the encoding computational complexity. However, if we continue to reduce *MinQTSize*, the performance may remain unchanged, and even little loss arises. On one hand, CFG3 can provide enough flexible block partition shapes. On the other hand, more overhead information is required to signal the splitting for smaller *MinQTSize*. Comparisons among CFG3, CFG5 and CFG6 show that small *MaxBTSize* will decrease the coding performance because the binary tree partition can only be used for the small blocks. When compare CFG3, CFG7 and CFG8, it is found that the coding performance also decreases for larger *MaxBTDepth*, which is also because the increased overhead of splitting signal.

From above simulations and analysis, it is observed that the partition parameters play an important role in the QTBT block partition structure. To reduce the unnecessary partition attempts while maintaining the partition flexibility, a dynamic partition parameters deriving algorithm is proposed in this work. As shown in Fig.2, we plot the block boundaries when the frame is compressed by QTBT structure. It is clearly seen that for the area with little texture, the block partition is simple and leads to large blocks. Therefore, large *MinQTSize*, *MaxBTSize* and small *MaxBTDepth* should be set to constrain the partition trees and hence the computational complexity can be reduced. On the contrary, for the area rich with texture, the block partition is complex and the binary tree splitting is frequently used, and hence small *MinQTSize* and large *MaxBTDepth* are demanded

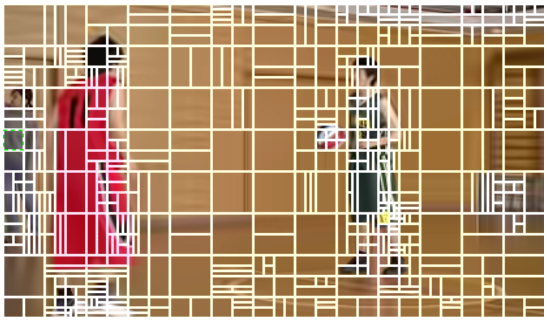

Fig. 2 Illustration of the block boundaries based on the QTBT structure

to suffice elaborate partitions. In this manner, flexible block partitions for the areas rich with texture can be maintained while the complex partition trees are constrained for the smooth areas.

The *MinQTSize* and *MaxBTSize* limit the minimal size of quadtree leaf node and the maximum size of binary tree root node, respectively. Considering the quadtree leaf node is actually the root node of binary tree, it is inferred that *MinQTSize* and *MaxBTSize* should be determined by the size of quadtree blocks. For example, if all the quadtree leaves nodes are 32x32 or 16x16 in a CTU, we can set *MinQTSize* and *MaxBTSize* as 16 and 32, respectively, to suffice the partition demand and avoid redundant partition attempts. With respect to *MaxBTDepth*, it determines how small a block can be split compared to the corresponding binary tree root node. Hence, *MaxBTDepth* can be set according to the size difference between the quadtree leaf node and the binary leaf node.

Considering the fact that the block splitting information can only be obtained after compression, the splitting information of the co-located CTU in the previous decoded frame is utilized to avoid multi-pass encoding and explicitly signalling. After the average block size of quadtree leaves nodes (denoted as QT_{ave}) and binary tree leaves nodes (denoted as BT_{ave}) in the co-located CTU are computed, *MinQTSize* and *MaxBTSize* are set to the values little less and larger than QT_{ave}, respectively, and *MaxBTDepth* is set to the ratio between QT_{ave} and BT_{ave}. Based on the proposed scheme, the partition parameters can be dynamically derived on both encoder and decoder, while the flexible partitions can be maintained for the areas rich with texture and complex partition trees are constrained for the smooth areas.

B. Constrained Binary Tree Partition

In the QTBT block partition structure, the quadtree splitting of one node (denoted as $N \times N$ shape) can be obtained by quadtree splitting straightforwardly or combinations of binary tree splitting indirectly (firstly horizontal splitting and then vertical splitting, or firstly vertical splitting and then horizontal splitting). Therefore, we propose a constrained binary tree partition algorithm to reduce the redundancy between the quadtree splitting and the binary tree splitting for the $N \times N$ shape.

Though the splitting shapes are identical for the above three splitting methods, the compression results may differ from each other because of different processing orders. As shown in Fig.

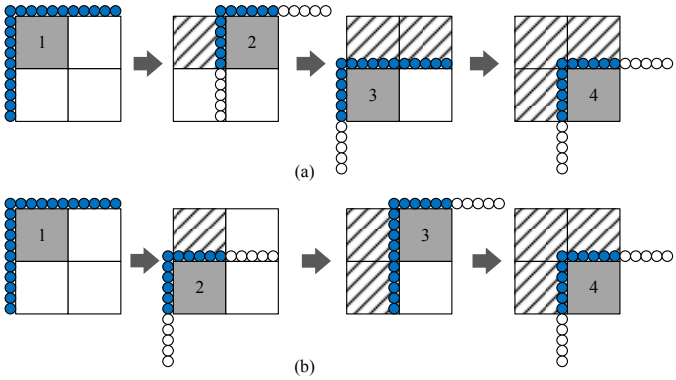

Fig. 3 Illustration of the different processing orders for $N \times N$ shapes

TABLE II
PERFORMANCE COMPARISONS AMONG DIFFERENT PROCESSING ORDERS FOR $N \times N$ SHAPE

Sequences	$QT+BT_{hor}$ RA	$QT+BT_{ver}$ RA	$QT+BT_{hor}$ All Intra	$QT+BT_{ver}$ All Intra
BasketballPass	0%	0%	0.5%	0.4%
BQSquare	-0.20%	0%	0.7%	0.6%
RaceHorses	0%	0.1%	0.4%	0.4%
BasketballDrill	0%	0%	0.5%	0.3%
BQMall	-0.1%	-0.1%	0.3%	0.4%
PartyScene	0%	0%	0.5%	0.6%

3-(a), both quadtree splitting (denoted as *QT*) and the horizontal splitting firstly and then vertical splitting (denoted as BT_{hor}) lead to the "Z" processing order. Otherwise, vertical splitting firstly and then horizontal splitting (denoted as BT_{ver}) lead to other processing order, which follows top-left, bottom-left, top-right and bottom-right order, as shown in Fig. 3-(b). Therefore, it is obviously inferred that the BT_{hor} splitting method can be removed when the quadtree splitting is valid, because both the splitting shapes and the processing orders are identical for *QT* and BT_{hor} spitting methods.

To verify the impact of processing order on the rate-distortion performance, simulations have been conducted and the results are presented in Table II, where the anchor is the QTBT structure without any constraint on binary tree splitting. According to the simulation results, it is clearly observed that the processing order can be regarded to have no influence on the inter prediction, and hence the redundant binary tree splitting can be removed for the $N \times N$ shape. However, for the intra prediction, the coding performance may become better if more processing orders are selected, which mainly results from different processing orders leading to different reference samples as well as prediction signals. Therefore, we propose the constrained binary tree partition algorithm to reduce the redundancy of binary tree splitting for the $N \times N$ shape. For the I slice, BT_{hor} splitting is removed if quadtree splitting is valid, and both BT_{hor} and BT_{ver} splitting are allowed if quadtree split is invalid. For the P and B slice, both BT_{hor} and BT_{ver} splitting are removed if quadtree splitting is valid, and only BT_{hor} splitting is allowed if quadtree splitting is invalid.

IV. EXPERIMENTAL RESULTS

The proposed dynamic partition parameters deriving algor-

TABLE III
PERFORMANCE OF THE PROPOSED LC-QTBT BLOCK PARTITION STRUCTURE COMPARED TO HM 13.0

Class	All Intra				Random Access			
	Y	U&V	Enc T	Dec T	Y	U&V	Enc T	Dec T
A	0.6%	1.7%	69%	100%	0.5%	1.6%	69%	100%
B	0.5%	2.1%	66%	99%	0.7%	1.9%	71%	100%
C	0.6%	1.9%	62%	99%	0.4%	2.0%	74%	99%
D	0.6%	2.0%	61%	100%	0.6%	1.9%	71%	100%
E	0.7%	1.8%	64%	99%	0.5%	2.1%	73%	100%
Overall	**0.6%**	**1.9%**	**65%**	**99%**	**0.5%**	**-1.9%**	**72%**	**100%**

TABLE IV
PERFORMANCE OF THE PROPOSED LC-QTBT BLOCK PARTITION STRUCTURE COMPARED TO HM 13.0

Class	Lowdelay				Lowdelay-P			
	Y	U&V	Enc T	Dec T	Y	U&V	Enc T	Dec T
A	0.5%	2.1%	72%	100%	0.6%	2.0%	79%	100%
B	0.8%	2.3%	78%	99%	0.6%	2.1%	81%	99%
C	0.7%	1.6%	75%	100%	0.7%	1.9%	77%	99%
D	0.7%	1.9%	71%	100%	0.8%	1.7%	79%	100%
E	0.6%	3.0%	73%	99%	0.5%	2.3%	76%	100%
Overall	**0.7%**	**2.2%**	**74%**	**100%**	**0.6%**	**2.0%**	**78%**	**100%**

TABLE V
PERFORMANCE OF THE PROPOSED LC-QTBT BLOCK PARTITION STRUCTURE COMPARED TO QTBT

Class	All Intra				Random Access			
	Y	U&V	Enc T	Dec T	Y	U&V	Enc T	Dec T
A	-3.6%	-14.2%	407%	104%	-7.1%	-15.2%	134%	105%
B	-3.9%	-14.3%	432%	104%	-5.8%	-15.0%	127%	105%
C	-3.1%	-11.6%	546%	111%	-4.9%	-10.3%	141%	107%
D	-2.2%	-8.7%	538%	117%	-4.7%	-8.3%	136%	106%
E	-5.9%	-20.2%	424%	103%	-5.1%	-10.7%	133%	105%
Overall	**-3.7%**	**-13.8%**	**469%**	**108%**	**-5.5%**	**-11.9%**	**134%**	**106%**

TABLE VI
PERFORMANCE OF THE PROPOSED LC-QTBT BLOCK PARTITION STRUCTURE COMPARED TO QTBT

Class	Lowdelay				Lowdelay-P			
	Y	U&V	Enc T	Dec T	Y	U&V	Enc T	Dec T
A	-4.2%	-9.5%	132%	104%	-4.3%	-12.1%	119%	104%
B	-6.0%	-11.6%	140%	106%	-6.4%	-14.4%	123%	105%
C	-5.6%	-9.6%	141%	108%	-5.4%	-10.5%	122%	109%
D	-4.0%	-6.2%	137%	109%	-3.7%	-6.7%	119%	110%
E	-7.8%	-21.1%	119%	103%	-8.6%	-21.7%	117%	105%
Overall	**-5.5%**	**-11.8%**	**134%**	**106%**	**-5.7%**	**-13.3%**	**120%**	**106%**

ithm can provide flexible partitions for the area rich with texture and constrain the partition trees for the smooth area. Furthermore, binary tree partition is constrained to remove the redundancy between the quadtree and binary tree partitions for the $N \times N$ shape. When the proposed algorithms are jointly integrated into the QTBT structure, we regard it as a local-constrained QTBT (LC-QTBT) block partition structure. The proposed LC-QTBT scheme has been integrated into HEVC reference software HM13.0 [4], and simulations are conducted on the common test sequences and configurations defined in HEVC standardization [5]. The experimental results, compared to the QTBT structure, are shown in Table III and Table IV, where **Y** and **U&V** represent the BD-rate performance for luma and chroma components, respectively. The relative encoding time and decoding time are denoted as **Enc T** and **Dec T**, respectively. From Table III and Table IV, it is clearly observed that approximately 35% encoding time saving can be achieved for AI configuration with only 0.6% performance loss introduced. For the RA, LD and LDP configurations, approximately 26% encoding time saving can be achieved with also slight BD-rate loss (0.6%).

For the comparisons between the proposed LC-QTBT algorithm and the partition structure of HEVC, the results are shown in Table V and Table VI. It is clearly seen that the LC-QTBT scheme can achieve 3.7% BD-rate gain for the All Intra (AI) configuration, while the encoding time increases to 469%. For other configurations, averagely 5.5%, 5.5% and 5.7% BD-rate gain can be achieved for Random Access (RA), Lowdelay (LD) and Lowdelay-P (LDP), respectively, with only 20%~30% encoding time increased. According to the simulation results, it is found that the proposed LC-QTBT algorithm can speed up the QTBT partition process while maintaining the coding performance, which enables its practical implementations in real application scenarios.

V. CONCLUSIONS

In this paper, we propose a LC-QTBT block partition structure to further optimize the QTBT partition. In the proposed scheme, two algorithms are adopted. Firstly, the partition parameters are derived dynamically for each CTU, and both encoder and decoder can perform the same process to avoid explicitly information signaling. Based on the dynamic partition parameters deriving scheme, complex partition trees are constrained for the smooth areas. Secondly, a constrained binary tree partition algorithm is proposed to remove the redundant binary tree splitting for the $N \times N$ shape. Compared to the original QTBT block partition structure, approximately 30% encoding time savings can be achieved with ignorable coding performance decrease.

ACKNOWLEDGMENT

This work was supported in part by the National Basic Research Program of China (973 Program, 2015CB351800), and National Natural Science Foundation of China (61322106, 61421062), which are gratefully acknowledged.

REFERENCES

[1] I. Kim, J. Min, T. Lee, W. Han, Block Partitioning Structure in the HEVC Standard, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 22, No. 12, pp. 1697-1706, Dec. 2012.

[2] Sze V, Budagavi M, Sullivan G J. High Efficiency Video Coding (HEVC)[M]//Integrated Circuit and Systems, Algorithms and Architectures. Springer, 2014: 1-375.

[3] J. An, H. Huang, K. Zhang, Y.-W. Huang, S. Lei. Quadtree plus binary tree structure integration with JEM tools, JVET-B0023, Joint Video Exploration Team (JVET).

[4] HEVC software repository. Available online: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware.

[5] Common test conditions and software reference configurations, JCTVC-J1100, ISO/IEC-JCT1/SC29/WG11, Stockholm, Sweden, 2012.