IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Fast CU Partition Decision Algorithm For AVS3 Intra Coding

TONG WU[1,2], SHIYI LIU[1,2], FENG WANG[1], ZHENYU WANG[1,2], RONGJIE WANG[3], AND RONGGANG WANG[1,2]

[1]School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University, Shenzhen 518055, China
[2]Peng Cheng Laboratory, Shenzhen 518055, China
[3]Department of Computer Science, City University of Hong Kong, 999077, China

Corresponding author: Rongjie Wang (e-mail: rjwang.hit@gmail.com).

This work was supported in part by the National Natural Science Foundation of China under Grant 61672063 and Grant 62072013, and in part by the Shenzhen Research Projects of JCYJ20180503182128089 and 201806080921419290.

ABSTRACT The third generation of Audio Video coding Standard (AVS3) is an emerging video coding standard that surpasses High Efficiency Video Coding (HEVC). AVS3 allows flexible block subdivision by applying Quad-Tree (QT), Binary-Tree (BT) plus Extend Quad-Tree (EQT) partition structure, while the increased flexibility comes at the cost of enormous coding complexity. In this paper, an ingenious early termination mechanism is proposed to skip unnecessary exhaustive searches of the whole tree branches. We carry out a series of restrictive measures based on Coding Unit (CU) size and iteration status to make the sophisticated EQT split mode concentrate more on small CUs with complex texture structure. Historical QTBT partition information is also adopted as an important factor to skip EQT split mode in advance. Meanwhile, a fast CU partition algorithm based on the gradient is proposed to skip horizontal or vertical BT/EQT partition of CUs with prominent texture structure in another direction, in which early termination can be directly conducted in homogenous areas at the same time. Extensive experiments demonstrate that the proposed method can save 43% encoding time with only 0.53% BDBR increase on average under All Intra(AI) configuration, which outperforms the preexisting fast algorithms.

INDEX TERMS AVS3, fast CU partition decision, intra coding, extend quad-tree, gradient, texture complexity analysis.

I. INTRODUCTION

Audio Video coding Standard (AVS) work group was founded in 2002 with the purpose of producing standards of high quality for compression, decompression, processing and representation of digital audio and video [1]. In order to meet the dramatic growing demand for 4K/8K and Virtual Reality (VR) videos in the upcoming age of 5G communication, the baseline profile of the third generation of Audio Video coding Standard (AVS3) was finalized in March 2019.

In the previous video coding standards, e.g. High Efficiency Video Coding (HEVC), the quad-tree structure is the only way to reach the adaptive Coding Unit (CU) size. Prediction Unit (PU) and Transform Unit (TU) are defined to optimize prediction and transformation procedures at the same time. However, the simple quad-split structure is less effective for the compression of 4K/8K videos. Therefore, great breakthrough in block subdivision must be made to meet the growing demand for high resolution videos.

In order to break the square-only block size limit and adapt to complex video texture in different scenes, AVS3 employs a sophisticated multi-tree portioning mechanism that contains Quad-Tree(QT), Binary-Tree (BT) and Extended Quad-Tree (EQT) [2]. Both BT and EQT partition have two directions, horizontal and vertical, to deal with different image texture structure. Diverse partition types provide more flexible block shapes. In addition, Intra Derived Tree (IntraDT), Intra Prediction Filters (IPF) and Two-Step Cross component Prediction Model (TSCPM) are adopted to optimize intra coding.

As a consequence, AVS3 baseline profile achieves a significant improvement in coding efficiency in contrast with the previous video coding standard. Experimental result shows that AVS3 can achieve 26.88% Bjøntegaard Delta Bit Rate (BDBR) saving compared with HEVC [3]. However, due to the application of the sophisticated coding tree structure and other time-consuming coding tools, the coding complexity of AVS3 is 7 times as much as HEVC [4].

The increased flexibility in block partition comes at the cost of enormous coding complexity, which significantly

reduces the competitiveness of AVS3 against state-of-the-art codecs. Since AVS3 is a newly-developing video coding standard, there are precious few fast CU partition algorithms proposed to reduce the coding complexity of the nested BT/EQT partition structure to the best of our knowledge. In this paper, we propose a novel early termination method aiming to speed up the block partition decision procedure of AVS3 by skipping the exhaustive searches of the whole tree branches in the well-established top-down encoding structure. The proposed fast intra CU partition mechanism is mainly based on CU size, iteration status, historical partition information and gradient. The main contributions of our work are described as below:

1. We carry out a series of restrictive measures based on CU size and iteration status to make EQT split mode focus more on small CUs with complex texture. To be precise, EQT split mode is forbidden when the maximum CU side length equals 64 and EQT split mode can not iterate successively if CU size is greater than 16×16.

2. Historical QTBT partition information is adopted as an important factor to skip EQT split mode in advance, which means EQT split mode will be skipped when the best QTBT split cost is a little bit bigger than the non-split cost.

3. Since CU texture structure has a strong relationship with the best split mode chosen by Rate-Distortion Optimization (RDO) procedure, we use the summation of horizontal and vertical gradient of the current CU to present its texture complexity in two directions. Horizontal or vertical BT/EQT partition will be skipped in advance if the current CU has a prominent texture structure in another direction.

The proposed algorithm is adopted by uAVS3e [5], which is an open source AVS3 encoder and available at https://github.com/uavs3/uavs3e. Experimental results show that the proposed fast intra CU partition method can save 43% encoding time with only 0.53% BDBR increase on average. At another trade-off point, encoding time saving is increased to 57% with BDBR increase below 0.93%. However, the method proposed in [20] only saves 13% encoding time at the cost of approximately 0.5% BDBR increase.

The rest of this paper is organized as follows. Section II introduces the current proposed fast CU partition decision methods. Section III presents the block partition structure of AVS3 and the details of the proposed algorithm. The experimental results and conclusion are shown in Section IV and Section V respectively.

II. RELATED WORK

Numerous researches have been conducted to reduce the coding complexity of CU partition structure in the previous video coding standards. The recursive quad-tree structure is adopted to provide adaptive block size in HEVC [6]. In HEVC, a frame will be divided into multiple 64×64 LCUs at first [7]. One LCU can be recursively split into four equal-sized smaller CUs until the smallest CU size limit is reached. However, the lack of non-square CU size limits the flexibility of block partition in HEVC.

There have been many research works [8]-[10] proposed to reduce the coding complexity of QT partition in HEVC. A split cost prediction method was proposed in [8] to discard any further splitting after a specific cost threshold was reached, and the threshold was obtained by weighting the split signaling rate and current non-split cost. Shang et al. [9] exploited the depth information of neighboring CUs to make early CU split or pruning decisions. In [10], an early termination decision based on gradient was applied to terminate the partition of homogenous CUs. Non-split cost and gradient can be utilized to conduct early termination strategy with an ignorable BDBR increase as shown in [8]-[10].

In AVS3, the largest block size for coding tree unit is extended from 64×64 to 128×128, which is the same as AV1 and Versatile Video Coding (VVC). AV1 expanded the partition-tree to a 10-way structure that includes 4:1 and 1:4 rectangular partitions, and none of the rectangular partitions can be further subdivided [11]. Chiang et al. [12] proposed a two-pass block partition search mechanism that built a pre-partition map by only applying QT at first and then conducted an extensive partition search. Gu et al. [13] started encoding from the middle depth, and then utilized the partition information to guide the bidirectional pruning process. According to these researches, unnecessary rectangular partitions should be avoided and parent split information can be utilized to reduce the complexity of the CU partition structure.

VVC introduces a quad-tree with nested multi-type tree (QTMT) structure by adding binary and ternary tree which achieves great coding performance with a dramatic increase of coding complexity. Numerous fast algorithms have been proposed to reduce the coding complexity of VVC. In [14], variance and gradient were utilized to distinguish the large smooth area and complex texture region, and fast partition decision could conduct directly according to these messages. Tang et al. [15] used edge features extracted by canny edge detector to skip vertical or horizontal split mode to carry out early termination. [16] presented an early termination strategy that skipped the evaluation of QT split mode if both binary splits had been evaluated but did not reduce the coding cost. A fast partition method based on Bayesian decision rules was designed by Fu et al. [17], which jointly utilized split types and intra modes of sub-CUs to skip certain partitions in advance. Convolutional neural networks and machine learning are widely used in video coding nowadays. A size adaptive CU partition decision algorithm prepared for VVC was proposed in [18] by using a flexible CNN pooling layer to optimize CU partition. Zhang et al. [19] put forward a fast CU partition method for VVC based on random forest classifier model, which can distinguish smooth and complex area to conduct early termination. The complex QTMT structure can be optimized by taking advantage of gradient obtained by edge detector and historical partition information, which implies that the block partition structure of AVS3 may also be optimized by jointly using these information.

Few research works are concentrating on the fast CU partition decision for AVS3 intra coding. Some of the re-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3049148, IEEE Access

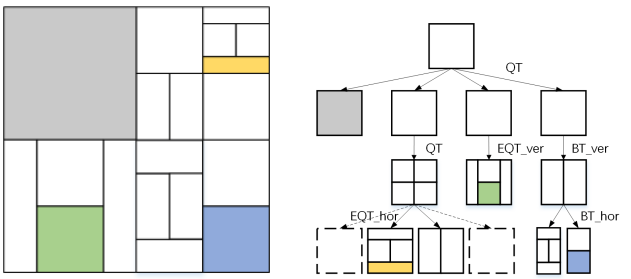
search works can achieve an acceptable trade-off between coding complexity reduction and BDBR increase. Wang et al. [20] took full advantage of the temporary partition results generated by the BT partition to terminate the EQT and QT partition when the predicted split depth satisfied the default threshold. However, with the speedy development of AVS3, the traversal of block partition now strictly follows the order of non-split, QT, BT and EQT split mode, the method mentioned above can not be used directly.

Previous research works have shown that historical partition information and gradient can be utilized to conduct early termination strategy, and the formation of narrow blocks should be avoided to reduce the complexity of the CU partition structure.

III. PROPOSED ALGORITHM

A. MOTIVATION AND OVERVIEW OF THE PROPOSED ALGORITHM

In AVS3, The largest block size for the coding tree unit is extended to 128×128. AVS3 employs a sophisticated multi-tree structure that contains Quad-Tree, Binary-Tree and Extended Quad-Tree. Three tree types are illustrated in Fig. 1. BT split mode fits the most block size while EQT split mode is only available from 8×16 or 16×8 to 64×64. Diverse partition types provide more flexible block shapes at the cost of huge coding complexity.



(a) Partition structure (b) Coding tree structure

FIGURE 1: **CU partition structure in AVS3. (a) presents an example of CTU partition in AVS3 and (b) illustrates the path to reach CUs in different depths.**

The exhaustive searches of the whole coding tree branches and the consequent RDO procedure occupy more than 90% of the encoding time in AVS3. In other words, the superposition of the well-established top-down encoding structure and the application of sophisticated multi-tree portioning mechanism causes the geometrical increase in coding complexity. Reasonable pruning operations can lead to a satisfying balance between encoding speed promotion and coding performance decline. We will analyze the feasibility of conducting the pruning algorithm based on the characteristics of AVS3 encoder in the following paragraphs.

There are many existing split constraints in AVS3. EQT split mode is not allowed in picture boundary, and the sub-block of BT and EQT split mode cannot utilize QT split mode in the subsequent partition process. If all the split modes are available in current CU, the non-split mode will be conducted firstly, then QT, BT and EQT split mode will be tried in order. Thus, historical QTBT partition information may provide some useful guidance to skip EQT split mode in advance.

An example of CTU partition in AVS3 and the corresponding path to reach CUs in different depths are shown in Fig. 1. A LCU chooses QT split mode at first, then one child node stops dividing and becomes a leaf node while the rest nodes are further split by QT, vertical EQT and vertical BT split mode, respectively. As shown in Fig. 1, BT and EQT split mode can be utilized in a nested structure while QT split mode is forbidden after BT and EQT partition. The partition structure of EQT is more elaborate than QTBT. Therefore, the successive iterations of EQT split mode may result in a complex block partition structure as we can imagine. The intermittent EQT iterations might be a preferable solution to decrease coding complexity.

In order to get an intuitive understanding of the proposed method, the overall flow diagram and the pipeline of the proposed fast CU partition decision algorithm are presented in Fig. 2. It is worth mentioning that if the final judgment does not correspond to any of the branches in the flow diagram, the original block partition decision process will be applied instead.

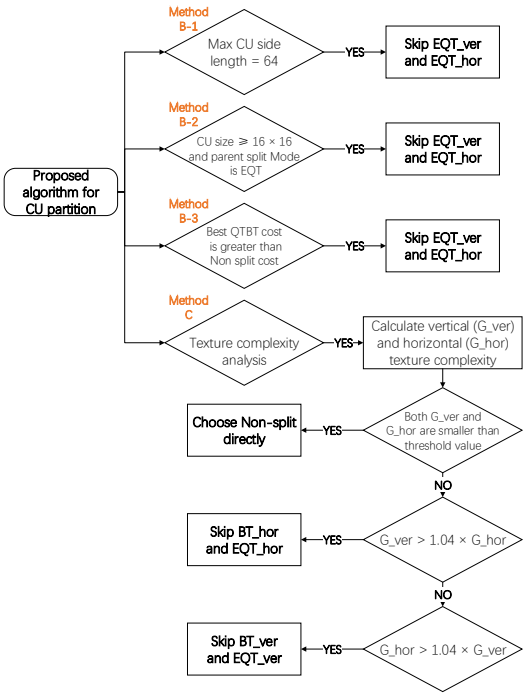


FIGURE 2: **The flow diagram of the proposed algorithm.**

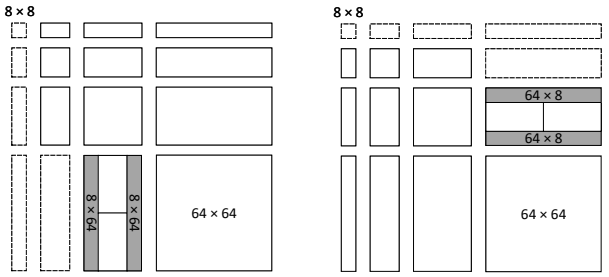
The application of the complex QT, BT plus EQT partition structure causes a geometric increase in coding complexity. Thus, based on what has been mentioned above, we will present a set of reasonable and practical limits to use the EQT split mode legitimately in the next section.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3049148, IEEE Access

B. PROPOSED EARLY TERMINATION ALGORITHM FOR EQT PARTITION

1) PROPOSED ALGORITHM BASED ON BLOCK SIZE

Appropriate aspect ratio constraint has been adopted in AVS3 baseline profile to establish a reasonable CU partition structure. If the CU width is 4 times longer than CU height, horizontal EQT split mode will not be available in current CU. Vertical EQT split mode will be abandoned if CU height is 4 times longer than CU width symmetrically. Thus, CU with size from 8×16 or 16×8 to 64×64 is available to conduct EQT partition in AVS3. Detailed block size constraints for the application of EQT split mode are illustrated in Fig. 3.



(a) EQT_ver (b) EQT_hor

FIGURE 3: **Available CU size (solid line) and unavailable CU size (dashed line) for EQT split mode in AVS3.**

However, the effect of the restrictions mentioned above is limited. The application of EQT split mode will produce two elongated rectangular blocks at the boundary which are not friendly to further partition and prediction. Even worse, as Fig. 3 shows that if we conduct vertical EQT split mode in a 32×64 block, two 8×64 narrow blocks will appear at the boundary. Therefore, effective measures should be taken to enhance the existing restrictions in AVS3. A series of experiments confirms that EQT split mode has limited performance on large blocks, and thus EQT split mode should be forbidden when the maximum CU side length equals 64. The details of the experiments are shown below.

Tool-off test generates the experimental results by turning off a single technique with other tools turned on. In order to evidence the additional contribution of EQT split mode to the overall performance, we conducted the tool-off test under All Intra(AI) and Random Access (RA) configuration respectively. The turning off of EQT split mode causes 1.55% BDBR increase and 56% encoding time saving under AI configuration, while it leads to 3.84% BDBR increase and 43% encoding time reduction under RA configuration. The test result indicates that the application of EQT split mode will increase the coding complexity greatly while the improvement of coding efficiency is limited under AI configuration.

Experimental results shown in Table 1 indicate that the performance of EQT split mode differs in CU size. Test 1-7 perform a series of tool-off experiments that forbid vertical and horizontal EQT split mode in different CU sizes to evaluate the coding performance of EQT split mode.

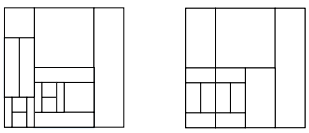
TABLE 1: **Performance of EQT split mode in different CU sizes(tool-off test).**

Test	Forbid EQT split mode		BDBR(%)			Time Saving
	Vertical	Horizontal	Y	U	V	(%)
1	64×64	64×64	0.05	0.02	0.12	11
2	64×32	32×64	0.06	-0.03	0.08	5
3	32×64	64×32	0.04	0.01	0.09	6
4	64×16	16×64	0.04	-0.08	0.09	5
5	64×8	8×64	0.02	0.03	0.10	3
6	32×32	32×32	0.14	0.21	0.28	8
7	<32×32	<32×32	0.76	1.09	1.30	22

As shown in Table 1, EQT split mode has limited performance on large CUs, especially when the maximum value of CU width and CU height is 64. The sophisticated EQT partition structure is more likely to match the texture in small areas than large areas. If EQT split mode is forbidden when the maximum CU side length equals 64, 33% encoding time will be saved with only 0.37% BDBR increase under All Intra configuration, the probability of the occurrence of narrow blocks will be greatly reduced at the same time.

2) PROPOSED ALGORITHM BASED ON ITERATE CONSTRAINT

The successive iterations of EQT split mode could result in a complex block partition structure that increases the complexity of the coding tree structure geometrically. In all of the tree branches, the probability of applying EQT split mode continuously over three times in one branch is less than 5% according to our survey. Inspired by [11], the intermittent EQT iterations will be a preferable solution to decrease coding complexity. Since BT and EQT split mode can be utilized in a nested structure, the successive iterations (EQT-EQT-EQT) can be replaced by the intermittent iterations (EQT-BT-EQT). Fig. 4 depicts the successive iteration (a) and the intermittent iteration (b) structure of EQT split mode.



(a) (b)

FIGURE 4: **Successive iterations (a) and intermittent iterations (b) of EQT split mode.**

The successive iteration structure could rarely match actual video texture even in the complex edge region while the intermittent iteration structure is more flexible and versatile in large blocks. If the CU size is larger than 16×16, successive iterations of EQT split mode will be broken and intermittent iterations will be available instead. The proposed method can save 13% encoding time at the cost of 0.17% BDBR increase under All Intra configuration.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3049148, IEEE Access

3) PROPOSED ALGORITHM BASED ON HISTORICAL PARTITION INFORMATION

Reference [16] presented an early termination strategy that QT split mode will be skipped if both binary splits were evaluated but performed worse than non-split. Since the traversal of block partition follows the order of non-split, QT, BT and EQT split mode, the QTBT Rate-Distortion (RD) cost could be a useful clue to avoid inefficient EQT partition attempts. Table 2 presents the statistical hit rate (P) of skipping EQT split mode in different CU sizes (W×H) when the best QTBT split cost is greater than the non-split cost. The statistical hit rate is averaged from all of the test sequences that are running on four Quantization Parameter (QP) points.

TABLE 2: **Hit rate of skipping EQT partition in different CU sizes when historical optimal split mode is non-split.**

	EQT_ver				EQT_hor			
P W / H	8	16	32	64	8	16	32	64
8	×	0.86	0.71	0.61	×	×	×	×
16	×	0.79	0.63	0.69	0.85	0.78	0.62	×
32	×	0.64	0.71	0.69	0.70	0.64	0.71	0.69
64	×	×	0.81	0.99	0.62	0.69	0.81	1.00

According to plentiful experimental results, if the minimum RD cost of QTBT split modes exceeds a certain proportion (α) of non-split cost, EQT split mode will almost never be selected as the best split mode for the current block. To achieve a pretty trade-off between BDBR increase and coding complexity decrease, EQT split mode will be skipped when the minimum QTBT RD cost is α times more than non-split cost. In order to obtain the optimal value of α, we calculated the mean value of α under the circumstances that EQT split cost is greater than the non-split cost. The statistical values of α in different CU sizes are shown in Table 3.

TABLE 3: **The statistical average value of α in different CU sizes.**

	EQT_ver				EQT_hor			
α W / H	8	16	32	64	8	16	32	64
8	×	1.12	1.10	1.06	×	×	×	×
16	×	1.19	1.08	1.05	1.12	1.19	1.08	×
32	×	1.07	1.08	1.03	1.09	1.07	1.08	1.03
64	×	×	1.02	1.05	1.05	1.04	1.02	1.04

5% encoding time will be saved with only 0.07% BDBR increase when α equals 1.02.

The proposed early termination algorithms for EQT partition in section B can be summarized as follows:

1. EQT split mode is forbidden when the maximum CU side length equals 64.
2. EQT split mode should iterate discontinuously if CU size is greater than 16×16.
3. EQT split mode will be skipped when the minimum QTBT RD cost is α times more than the non-split cost.

C. PROPOSED ALGORITHM BASED ON GRADIENT

Numerous research works indicate that CU texture has a strong relationship with the best split mode chosen by the RDO procedure. CUs with outstanding vertical texture will almost never choose the horizontal split mode as the final partition decision. In the digital image processing field, gradient is an excellent indicator to express the feature of image texture. Reference [10], [14] and [15] have shown that gradient is an effective reference in terminating the partition of homogenous CUs and choosing QT split mode directly to deal with CUs with complex texture. Vertical or horizontal split mode can be skipped if the gradient in the other direction is more outstanding.

Sobel operator is a famous edge detector by obtaining the first-order gradient from original image. As is known to us all, the luminance component has the highest texture similarity with the original image, thus only luminance component pixel value participates in the following detect procedure.

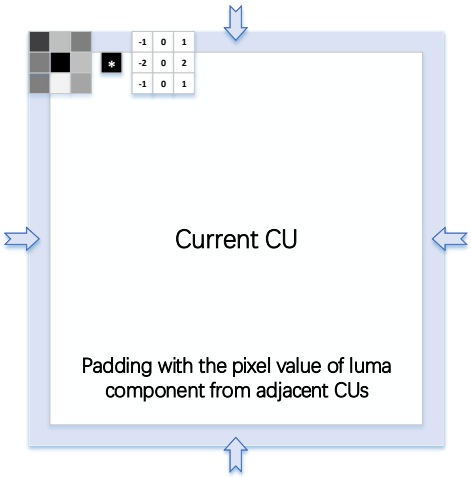


FIGURE 5: **An example of vertical texture detection by using the sobel operator.**

Fig. 5 presents the detection procedure by using the sobel operator. As we can see in Fig. 5, the current CU is padded with the pixel value from adjacent CUs at first. In this way, gradient information will be more precise than zero padding and duplicate padding.

Horizontal and vertical sobel operators can extract horizontal and vertical image texture respectively. By calculating the horizontal and vertical gradient of each point in the current CU, the texture of the whole CU can be represented by the summation. Notice that it is necessary to take the minimum value between the calculated gradient of each point and 255 (for 8-bit video) before summation to avoid noise influence. The process procedure is expressed below.

g_ver presents the gradient of the current luminance pixel (i, j) in the horizontal direction, and its value also indicates

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3049148, IEEE Access

the texture significance of the current position in vertical direction. Similarly, g_hor presents the texture significance in horizontal direction.

$$g\_ver(i,j)=img\_Y(i,j)*\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \quad (1)$$

$$g\_hor(i,j)=img\_Y(i,j)*\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \quad (2)$$

Intuitively, G_ver and G_hor can represent the complexity of CU texture in two directions.

$$G_{\_ver}=\sum_{i=1}^{width}\sum_{j=1}^{height} min(|g\_ver(i,j)|,255) \quad (3)$$

$$G_{\_hor}=\sum_{i=1}^{width}\sum_{j=1}^{height} min(|g\_hor(i,j)|,255) \quad (4)$$

The gradient based fast intra CU partition algorithm is described below.

If G_ver is β times more than G_hor, horizontal BT and EQT split mode will be skipped in current CU, and if G_hor is β times greater than G_ver, vertical BT and EQT split mode will be skipped symmetrically. 13% encoding time will be saved with only 0.07% BDBR increase when the sobel detect size is larger than 64×64 and β equals 1.04.

Meanwhile, homogenous areas could also be labeled if both G_ver and G_hor are smaller than a certain parameter. Early termination decision could be applied reasonably to terminate the partition of homogenous CU. The value of G_ver and G_hor are not normalized according to CU size, and the block partition structure will be more and more rough with the increase of QP, which means more radical strategies should be adopted when QP increases. Therefore, the parameter is expressed as $\gamma \times$ QP $\times$ CU_Size.

IV. EXPERIMENTAL RESULTS

A. TEST CONDITIONS

The proposed algorithm has been integrated into uAVS3e which is an open source AVS3 encoder. For the hardware environment, all of the experiments are conducted on the Intel Xeon CPU E5-2670 v2 at 2.50GHz with 32 GB RAM. All of the 8-bit HEVC common test sequences in Class A (UHD), B (1080P), C (480P), D (240P) and E (720P) are tested for 10 seconds under AI configuration to verify performance. The coding performance is measured by BDBR, Peak Signal-to-Noise Rate (PSNR) and encoding Time-Saving (TS). TS is calculated as follows:

$$TS=\frac{T_{anchor}-T_{proposed}}{T_{anchor}}\times 100\% \quad (5)$$

TS for each sequence is defined as the average time saving of four different QPs. Two sets of parameters presented in

TABLE 4: **The experimental results in uAVS3e. Original uAVS3e (anchor) VS Modified uAVS3e (apply the parameter settings in Test 1).**

Class	Resolution	Sequences	BDBR(%)				PSNR(dB)	Time Saving(%)
			Y	U	V	Average(4:1:1)		
A	2560 × 1600	Traffic	0.28	0.94	1.07	0.46	-0.0020	42.93
		PeopleOnStreet	0.28	1.46	1.73	0.61	-0.0080	42.64
		UHD Average	**0.28**	**1.20**	**1.40**	**0.54**	**-0.0050**	**42.79**
B	1920 × 1080	Kimono	0.51	0.92	0.70	0.61	-0.0090	38.03
		ParkScene	0.07	1.16	0.88	0.39	-0.0010	42.90
		BasketballDrive	0.62	1.41	1.35	0.87	-0.0090	43.46
		Cactus	0.26	0.91	0.98	0.49	-0.0020	42.46
		BQTerrace	0.19	0.87	1.08	0.45	-0.0040	43.00
		1080P Average	**0.33**	**1.05**	**1.00**	**0.56**	**-0.0050**	**41.97**
C	832 × 480	BasketballDrill	0.32	0.73	1.03	0.51	-0.0050	43.35
		BQMall	0.30	1.13	1.33	0.61	-0.0080	43.61
		PartyScene	0.10	0.40	0.40	0.20	0.0000	44.52
		RaceHorsesC	0.11	0.37	0.26	0.17	0.0030	42.54
		480P Average	**0.21**	**0.66**	**0.76**	**0.37**	**-0.0025**	**43.51**
D	416 × 240	BasketballPass	0.60	1.82	1.69	0.98	-0.0230	40.65
		BQSquare	0.24	0.35	0.68	0.33	-0.0080	42.03
		BlowingBubbles	0.21	0.58	0.36	0.29	0.0000	41.09
		RaceHorses	0.04	0.25	0.30	0.12	0.0020	40.40
		240P Average	**0.27**	**0.75**	**0.76**	**0.43**	**-0.0073**	**41.04**
E	1280 × 720	FourPeople	0.44	1.22	1.14	0.69	-0.0080	45.73
		Johnny	0.62	1.50	1.69	0.95	-0.0110	43.67
		KristenAndSara	0.46	1.13	1.38	0.73	-0.0080	44.84
		720P Average	**0.51**	**1.28**	**1.40**	**0.79**	**-0.0090**	**44.75**
Average			**0.31**	**0.95**	**1.00**	**0.53**	**-0.0056**	**43.66**

Table 5 are defined as Test 1 and Test 2 to implement the proposed algorithms at different trade-off point.

TABLE 5: **Parameters of proposed algorithms in Test 1 and Test 2.**

Proposed	Parameters	Test1	Test2
B-1	max CU side length(=)	64	64
B-2	Intermittent iterations	Turn on	Turn off
B-3	α	1.11	1.05
	Sobel detect size($\geq$)	64×64	32×32
C	β	1.04	1.04
	γ	Turn off	0.1

The details of Test 1 and Test 2 are shown above. Experimental results of Test 1 and Test 2 are presented in Table 4 and Table 6 with detailed BDBR (%), PSNR (dB) and TS (%). Average BDBR is obtained from Y, U and V components with weighting coefficients of 4, 1 and 1.

B. TEST RESULTS OF THE PROPOSED ALGORITHM

In Test 1, all of the EQT split mode constraints are adopted and the sobel operator shall detect CUs that are larger than 64×64. To be more specific, EQT split mode is forbidden when the maximum CU side length equals 64 and EQT split mode should iterate discontinuously if CU size is greater than 16×16. Meanwhile, EQT split mode will be skipped when the minimum QTBT RD cost is α times more than the non-split cost.

As shown in Table 4, it is quite obvious that the proposed algorithm is effective for all of the test sequences and almost all of the sequences have a time reduction of over 40%. The proposed algorithm is quite useful for some test sequences, such as RaceHorses, RaceHorsesC, PartyScene and BlowingBubbles. Almost half of the encoding time is saved with a negligible BDBR increase in the sequences mentioned above. These sequences rarely apply the sophisticated EQT split mode in large areas and the texture is more prominent in smaller areas. Thus the adjustment of the application of EQT split mode has a positive effect on these sequences in terms of encoding time reduction. Texture complexity analysis in large blocks can also provide more precise guidance to the subdivision of small areas in these sequences.

Test 2 adopts an aggressive strategy to achieve more time saving. EQT split mode will not be available when the maximum CU side length equals 64 or the historical partition information satisfies the threshold(α) constraint. The detect size of the sobel operator is extended to 32×32 at the same time. As shown in Table 6, all of the test sequences have achieved double encoding speed at the cost of acceptable BDBR increase.

C. VISUALIZED ADJUSTMENTS IN BLOCK PARTITION STRUCTURE

In order to get an intuitive understanding of the CU partition structure adjustment, example frames from “Johnny”, "KristenAndSara", "Kimono" and "ParkScene" are presented in

TABLE 6: **The experimental results in uAVS3e. Original uAVS3e (anchor) VS Modified uAVS3e (apply the parameter settings in Test 2).**

Class	Resolution	Sequences	BDBR (%)				PSNR(dB)	Time Saving(%)
			Y	U	V	Average(4:1:1)		
A	2560 × 1600	Traffic	0.66	1.46	1.52	0.87	-0.0057	56.75
		PeopleOnStreet	0.65	1.96	2.36	1.03	-0.0111	56.43
		UHD Average	**0.66**	**1.71**	**1.94**	**0.95**	**-0.0084**	**56.59**
B	1920 × 1080	Kimono	1.08	1.56	0.94	1.14	-0.0192	55.46
		ParkScene	0.26	1.87	1.04	0.66	-0.0064	56.56
		BasketballDrive	1.07	1.96	1.82	1.35	-0.0108	60.29
		Cactus	0.55	1.24	1.52	0.83	-0.0040	56.58
		BQTerrace	0.33	1.29	1.50	0.68	-0.0072	56.81
		1080P Average	**0.66**	**1.58**	**1.36**	**0.93**	**-0.0095**	**57.14**
C	832 × 480	BasketballDrill	0.67	1.47	1.93	1.01	-0.0109	57.58
		BQMall	0.88	2.19	2.50	1.37	-0.0170	58.29
		PartyScene	0.11	0.63	0.57	0.27	-0.0012	57.46
		RaceHorsesC	0.36	0.41	0.97	0.47	0.0014	55.25
		480P Average	**0.51**	**1.18**	**1.49**	**0.78**	**-0.0069**	**57.14**
D	416 × 240	BasketballPass	0.98	3.18	3.24	1.73	-0.0409	58.15
		BQSquare	0.27	0.54	0.88	0.42	-0.0122	54.83
		BlowingBubbles	0.26	0.96	0.58	0.43	0.0005	55.83
		RaceHorses	0.25	0.63	0.54	0.36	-0.0019	54.09
		240P Average	**0.44**	**1.33**	**1.31**	**0.74**	**-0.0136**	**55.72**
E	1280 × 720	FourPeople	0.91	2.22	2.11	1.33	-0.0175	59.92
		Johnny	1.09	1.79	2.76	1.48	-0.0165	58.84
		KristenAndSara	1.00	2.10	2.27	1.39	-0.0136	58.93
		720P Average	**1.00**	**2.04**	**2.38**	**1.40**	**-0.0159**	**59.23**
Average			**0.63**	**1.53**	**1.61**	**0.93**	**-0.0108**	**57.11**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3049148, IEEE Access

(a) Johnny-original (b) Johnny-modified

(c) KristenAndSara-original (d) KristenAndSara-modified

(e) Kimono-original (f) Kimono-modified

(g) ParkScene-oroginal (h) ParkScene-modified

FIGURE 6: **Comparison between the original (left) and modified (right) CU partition structure when QP equals 27. The labeled rectangular areas are processed by the pruning strategy while the circular regions maintain a sophisticated block partition structure.**

TABLE 7: **Experimental results of the proposed method in uAVS3e (apply the parameter settings in Test 1, Test 2 and Wang[20], respectively) and HPM4.0.**

Class	Resolution	Proposed Test1 (uAVS3e)		Proposed Test2 (uAVS3e)		Wang[20] (uAVS3e)		Proposed Test1 (HPM4.0)	
		BDBR(%)	**TS**(%)	**BDBR**(%)	**TS**(%)	**BDBR**(%)	**TS**(%)	**BDBR**(%)	**TS**(%)
A	2560 × 1600	0.54	42	0.95	56	0.55	14	0.90	37
B	1920 × 1080	0.56	41	0.93	57	0.42	13	0.88	41
C	832 × 480	0.37	43	0.78	57	0.61	14	0.88	42
D	416 × 240	0.43	41	0.74	55	0.48	13	0.76	40
E	1280 ×720	0.79	44	1.40	59	0.62	13	1.12	42
Average		**0.53**	**43**	0.93	57	0.53	13	0.90	40

Fig. 6. Frames on the left are coded by the original uAVS3e while frames on the right are all coded by the modified uAVS3e with parameter settings in Test 2.

The brighter labeled areas commendably illustrate the effectiveness of the proposed fast CU partition algorithm. It should be noted that the labeled rectangular areas are precisely processed by the pruning strategy while the circular regions maintain a sophisticated block partition structure.

As shown in Fig. 6 (a) and (b), CUs in the background region are adjusted to a simple and clear partition structure. Many smooth areas are no longer divided to achieve early termination strategy, while the areas with complicated texture, such as the tie, face and edge, maintain a sophisticated partition structure. Large smooth areas in sequence "KristenAndSara" and "ParkScene" are also adjusted to a simplified block partition structure which can effectively reduce coding complexity. In Fig. 6 (e) and (f), the flat area of the cloth is no longer divided into a sophisticated structure while the face contour maintains a fine structure. The partition structure is getting more close to the image texture, especially the ear area. The comparison between the original and modified CU partition structure indicates the effectiveness of the proposed algorithm.

D. COMPARISON WITH OTHER WORKS

Reference [20] put forward a series of intra and inter fast CU partition algorithms for AVS3. We compared the proposed algorithms with Wang[20] by applying their fast intra algorithm (proposed method B, split depth prediction) to uAVS3e. However, the traversal of block partition in AVS3 now strictly follows the order of non-split, QT, BT and EQT split mode, so we only use the temporary optimal sub-tree after QT and BT partition to early terminate EQT partition in the current CU.

It's worth noting that by the end of July, 2020, uAVS3e has completed over seventy times iterations. By applying Wavefront Parallel Processing (WPP), frame-level parallel, adaptive chroma quantization parameters adjustment, assembly instruction optimization and other algorithms, uAVS3e achieves approximately 5% overall BDBR saving and 60 times speed up compared with HPM4.0 [21], which is the standard reference software of AVS3. Therefore, it is not surprising that the pretty performance achieved in [20] declined a lot in uAVS3e. Meanwhile, the proposed algorithm in Test 1 is implemented in HPM4.0 to verify performance. However, the parameters satisfy uAVS3e best, thus the BDBR increase might be a little large in HPM4.0.

The experimental results are shown in Table 7. The proposed method with parameter settings in Test 1 achieves a satisfying trade-off between BDBR increase and encoding time reduction. Meanwhile, Test 2 adopts an aggressive strategy which saves 57% encoding time. In uAVS3e, the proposed algorithm can save approximately 43% encoding time while the method proposed in [20] only saves 13% encoding time at the cost of approximately 0.5% BDBR increase. The proposed algorithm with parameter settings in Test 1 is also implemented in HPM4.0. Approximately 40% encoding time is saved at the cost of acceptable BDBR increase. The effectiveness of our algorithm is verified by various comparisons.

V. CONCLUSION

In this paper, we present a series of fast CU partition algorithms for AVS3 intra coding. The fast intra CU partition mechanism is designed mainly based on CU size, iteration status, historical partition information and gradient. Through setting constraints according to CU size and iteration status, EQT split mode can concentrate more on small CUs with complex texture structure. Historical QTBT partition information is also adopted as an important factor to skip EQT split mode in advance. The gradient is used to express texture complexity, thus inefficient BT and EQT partitions can be skipped to reduce encoding time. Experimental results demonstrate that the proposed algorithm can achieve about 43% encoding time saving on average with only 0.53% BDBR increase under All Intra configuration. At another trade-off point, speed-up is doubled with BDBR increase below 0.93% compared with the latest uAVS3e. The proposed algorithms achieve a good trade-off between BDBR increase and complexity reduction.

REFERENCES

[1] About AVS. [Online]. Available: http://www.avs.org.cn
[2] M. Wang *et al.*, "Extended Quad-Tree Partitioning for Future Video Coding," in *2019 Data Compression Conference (DCC)*, Snowbird, UT, USA, 2019, pp. 300-309, doi: 10.1109/DCC.2019.00038.
[3] J. Zhang, C. Jia, M. Lei, S. Wang, S. Ma and W. Gao, "Recent Development of AVS Video Coding Standard: AVS3," in *2019 Pic-*

ture Coding Symposium (PCS), Ningbo, China, 2019, pp. 1-5, doi: 10.1109/PCS48520.2019.8954503.

[4] K. Fan *et al.*, "Performance and Computational Complexity Analysis of Coding Tools in AVS3," in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, London, United Kingdom, 2020, pp. 1-6, doi: 10.1109/ICMEW46912.2020.9106018.

[5] About uAVS3e. [Online]. Available: https://github.com/uavs3/uavs3e

[6] G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649-1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.

[7] J. 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)," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669-1684, Dec. 2012, doi: 10.1109/TCSVT.2012.2221192.

[8] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe and T. Wiegand, "Recursive Partitioning Search Space Pruning Using Split Cost Prediction," in *2019 Data Compression Conference (DCC)*, Snowbird, UT, USA, 2019, pp. 260-269, doi: 10.1109/DCC.2019.00034.

[9] X. Shang, G. Wang, T. Fan and Y. Li, "Fast CU size decision and PU mode decision algorithm in HEVC intra coding," in *2015 IEEE International Conference on Image Processing (ICIP)*, Quebec City, QC, 2015, pp. 1593-1597, doi: 10.1109/ICIP.2015.7351069.

[10] T. Zhang, M. Sun, D. Zhao and W. Gao, "Fast Intra-Mode and CU Size Decision for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1714-1726, Aug. 2017, doi: 10.1109/TCSVT.2016.2556518.

[11] Y. Chen *et al.*, "An Overview of Core Coding Tools in the AV1 Video Codec," *2018 Picture Coding Symposium (PCS)*, San Francisco, CA, 2018, pp. 41-45, doi: 10.1109/PCS.2018.8456249.

[12] C. Chiang, J. Han and Y. Xu, "A Multi-Pass Coding Mode Search Framework For AV1 Encoder Optimization," in *2019 Data Compression Conference (DCC)*, Snowbird, UT, USA, 2019, pp. 458-467, doi: 10.1109/DCC.2019.00054.

[13] J. Gu and J. Wen, "Mid-depth Based Block Structure Determination for AV1," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, 2019, pp. 1617-1621, doi: 10.1109/ICASSP.2019.8682955.

[14] Y. Fan, J. Chen, H. Sun, J. Katto and M. Jing, "A Fast QTMT Partition Decision Strategy for VVC Intra Prediction," in *IEEE Access*, vol. 8, pp. 107900-107911, 2020, doi: 10.1109/ACCESS.2020.3000565.

[15] N. Tang *et al.*, "Fast CTU Partition Decision Algorithm for VVC Intra and Inter Coding," in *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Bangkok, Thailand, 2019, pp. 361-364, doi: 10.1109/APCCAS47518.2019.8953076.

[16] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe and T. Wiegand, "Fast Partitioning Decision Strategies for The Upcoming Versatile Video Coding (VVC) Standard," in *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, pp. 4130-4134, doi: 10.1109/ICIP.2019.8803533.

[17] T. Fu, H. Zhang, F. Mu and H. Chen, "Fast CU Partitioning Algorithm for H.266/VVC Intra-Frame Coding," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, Shanghai, China, 2019, pp. 55-60, doi: 10.1109/ICME.2019.00018.

[18] G. Tang, M. Jing, X. Zeng and Y. Fan, "Adaptive CU Split Decision with Pooling-variable CNN for VVC Intra Encoding," in *2019 IEEE Visual Communications and Image Processing (VCIP)*, Sydney, Australia, 2019, pp. 1-4, doi: 10.1109/VCIP47243.2019.8965679.

[19] Q. Zhang, Y. Wang, L. Huang and B. Jiang, "Fast CU Partition and Intra Mode Decision Method for H.266/VVC," *IEEE Access*, vol. 8, pp. 117539-117550, 2020, doi: 10.1109/ACCESS.2020.3004580.

[20] M. Wang *et al.*, "Fast Coding Unit Splitting Decisions for the Emergent AVS3 Standard," in *2019 Picture Coding Symposium (PCS)*, Ningbo, China, 2019, pp. 1-5, doi: 10.1109/PCS48520.2019.8954510.

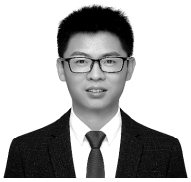
[21] About HPM. [Online]. Available: https://gitlab.com/AVS3_Software/hpm

TONG WU received the B.S. degree in electronic and information engineering from TianJin University, China, in 2018. He is currently pursuing the M.S. degree in computer science with Peking University. His research interests include video coding and digital image processing.

SHIYI LIU received the B.S. degree in Computer Science and Technology from Northwestern Polytechnical University, China. In 2019, She is currently pursuing the M.S. degree in computer engineering with Peking University. Her research interests include fast methods of hybrid video coding framework and image quality assessment.

FENG WANG received the B.S. degree in School of Electronic and Information Engineering from Beijing Jiaotong University, China, in 2019. He is currently pursuing the M.S. degree in computer science with Peking University. His research interests include video coding techniques and computer vision.

ZHENYU WANG received the Ph.D. degrees from the School of Electronics Engineering and Computer Science, Peking University, Beijing, China. As an associate research assistants, he is currently working in the research of digital video coding, Peking University Shenzhen Graduate School, Shenzhen, China.



RONGJIE WANG received the B.S. degree in mathematics and applied mathematics from the Heilongjiang University, China, in 2006, and the M.S. and Ph.D. degrees in applied mathematics and computer applied technology from Harbin Institute of Technology, China, in 2009 and 2019, respectively. His research interests include genome compression, deep learning in bioinformatics, and image/video processing.

RONGGANG WANG (M'12) was born in Hailin, Heilongjiang, China, in 1976. He received the Ph.D. degree in computer engineering from the Institute of Computing Technology, Chinese Academy of Sciences, in 2006.

From 2006 to 2010, he was a Researcher with the France Telecom R&D Laboratory. He is currently a professor in the School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School. He is the author of more than 60 papers and more than 50 inventions. His research interests include video coding, video processing, 3D video, and virtual reality. He has led the development of the ISO/IEC MPEG Internet Video Coding. He is currently the Co-Chair of the ISO/IEC MPEG Internet Video Coding Ad Hoc Group and the Chair of the IEEE 1857.9 Subgroup on Immersive Visual Content Coding.