Gradient-based Early Termination of CU Partition in VVC Intra Coding

Jing Cui*, Tao Zhang‡, Chenchen Gu‡, Xinfeng Zhang†, and Siwei Ma*

*Institute of Digital Media
Peking University
Beijing, 100871, China
jingcui106,swma@pku.edu.cn

† School of CS and Technolgy
UCAS
Beijing, 100049, China
xfzhang@ucas.ac.cn

‡ Wechat Business Group
Tencent
Shenzhen, China
toneyzhang,cicelygu@tencent.com

Abstract

The quaternary tree with nested binary and ternary tree structure is an efficient coding unit (CU) partitioning method adopted in Versatile Video Coding (VVC). Compared with quaternary tree only in HEVC, its flexible block sizes improve the coding performance significantly at the cost of computation load increase due to the recursive and nested searching for the best CU structure. In this paper, an early termination algorithm is proposed to skip unnecessary searches in CU size decision. Based on directional gradients, pre-determine the likelihood of binary partition or ternary partition in horizontal or vertical direction for current block, thus the impertinent partition pattern can be skipped. The experimental results show that the proposed method is able to save the encoding time up to 51% with about 1.2% BD-rate degradation on average compared with VVC software reference VTM5.0.

Introduction

Along with the growing demands on the high resolution videos, exploring next generation video coding standard with higher performance is becoming very imperative. VVC [1] is the newest standard developed by Joint Video Experts Team (JVET) which is a partnership of ITU-T Study Group known as VCEG and ISO/IEC JTC 1/SC 29/WG 11 known as MPEG. VVC officially starts standardization from April 2018, but its pre-research began earlier with platform called JVET test model (JEM) which aims to evaluate the potential of the explored coding tools. Although VVC is still under development, its software reference VVC test model (VTM) [2] has already achieved up to over 30% BD-rate saving compared with HEVC [3]. The higher coding performance benefits from more elaborate algorithm designs in each module, such as CU partitioning, intra modes, motion compensation, transform as well as dependent quantization [1]. However, VVC acquires the superior BD-rate performance at the expense of time cost. The encoding time increases almost 10 times compared with HEVC and there are almost 2 times decoding time rise. Reducing complexity is essential step to promote new standard and optimizing CU size decision algorithm is one of promising orientations to minimizing the computation loads. Different from HEVC, non-square blocks are allowed in block partition, since VVC adopts not only quaternary tree but also binary tree and ternary tree structure. Therefore, the CU partition is more complicated on account of additional searches in mode decision loop which increases the encoding time extraordinarily.

Plenty of works to early terminate and prune CU partition were proposed to optimize the CU size decision in intra coding. Cho *et.al.*'s work [4] proposed a fast

2375-0359/20/$31.00 ©2020 IEEE
DOI 10.1109/DCC47342.2020.00018

CU partitioning and pruning algorithm based on the Bayes decision rules for HEVC intra coding. An improved fast CU size decision algorithm using support vector machines (SVM) was presented with on-line and off-line statistical learning in work [5]. In addition, there are a lot of literatures aiming at reducing the complexity of CU splitting through well-trained classifiers [6–8] feeded by mass of data. However, all these algorithms were proposed for HEVC, where only quaternary tree structure is applied to split CU. The quaternary tree with nested multi-type tree coding block structure in VVC makes the single two class classifier impossible to decide the various partition patterns. Currently, few works to optimize the CU size decision in VVC are released. Yang *et.al* [9] proposed a statistical learning based algorithm to decide the partition tree structure. Although the encoding time saving was up to 52%, the RD performance was degraded over 1.5% due to the limitation of off-line learning in decision tree. For above learning-based classification schemes, the coding performance depends heavily on the accuracy of classifier. In the standardization stage, implementation-friendly and effective partitioning strategies are more valuable to practice. With the consideration that gradient feature calculation always occurs in the pre-analysis stage in codec development, it can be conducted in the separated thread without bringing extra computation load in frame encoding pipeline. Therefore, gradient has an inherent advantage in guiding block partition. In this work, we proposed an directional gradients based early termination algorithm for quaternary tree and nested binary and ternary tree partition in VVC intra coding.

The remainder of this paper is organized as follows: In Section II, the CU partition and intra coding in VVC are reviewed. The proposed early termination algorithm is presented in Section III followed by the experimental results in Section VI, which evaluate the performance of proposed algorithm. Finally, Section V concludes this paper.

Overview of CU Partition and Intra Coding in VVC

In HEVC, quaternary tree is the only structure and the largest CU size is set as 64×64 with maximum split depth being 4, thus the allowed CU sizes include 64×64, 32×32, 16×16 and 8×8. Compared with HEVC, VVC introduced more flexible partition structure to improve the coding performance, where quaternary tree with nested multi-type tree coding structure is used. The coding unit tree (CTU) with size 128×128 is firstly divided following quaternary tree structure and then the leaf nodes in quaternary tree is further split by binary tree and ternary tree partition. In Fig.1, an example of block partition structure is illustrated with final leaf node indexes, where different line indicates different partition pattern. QT, BT_H, BT_V, TT_H and TT_V are the abbreviations of quaternary tree partition, binary tree horizontal partition, binary tree vertical partition, ternary tree horizontal partition and ternary tree horizontal partition, respectively. $Depth$ is the quaternary tree partition depth, while the multi-type tree partition depth is called $MTDepth$. The allowed minimum quaternary tree block size is 16×16 and the maximum allowed multi-type tree partition depth $mxMTD$ is set as 4. The leaf nodes generated by multi-type tree partition are directly used for intra prediction and transform unless the leaf node block size

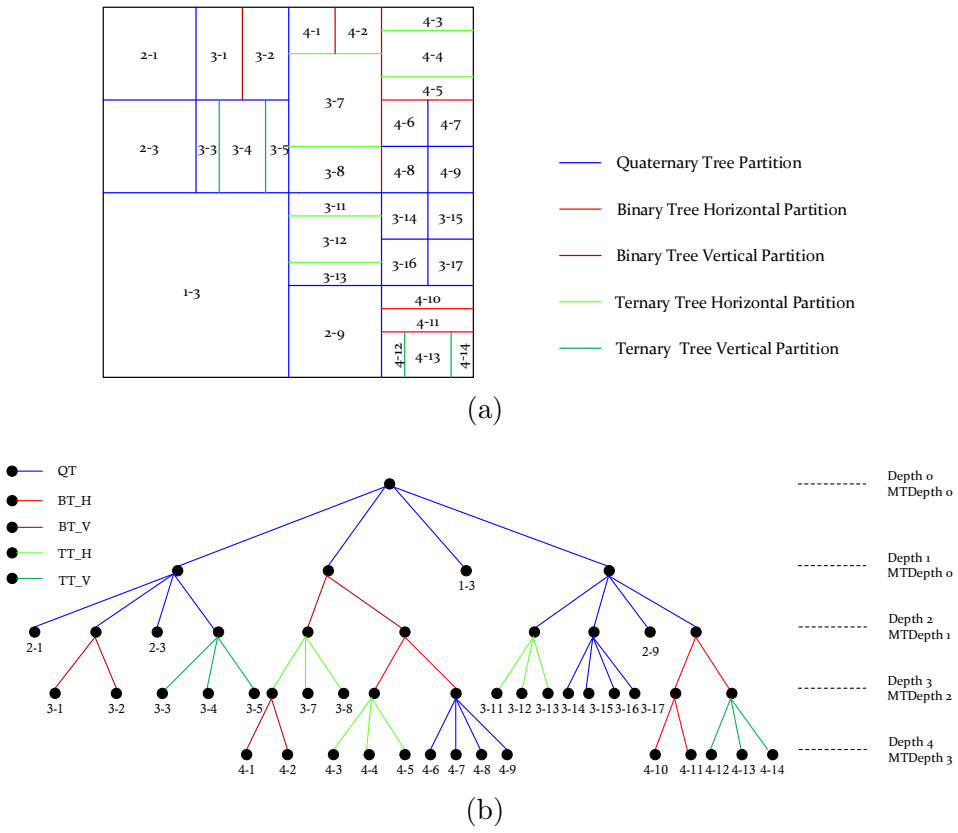

Figure 1: An example of quaternary tree with nested multi-type tree coding block structure.

exceeds the maximum transform length. In addition, for I frame, the separated partition loop is utilized for Luma and Chroma component, which means that one CU may consist one coding block of luma component and two coding blocks of chroma component. In implementation, all the possible partition patterns including QT, BT_H, BT_V, TT_H and TT_V are stored in a candidate list and then traversed one by one with RDO for current block.

The intra coding in VVC extends the intra mode from 35 into 67, which includes 65 directional modes, planar and DC mode. There are 6 most probable modes (MPMs) constructed by the default intra modes, neighbouring intra modes and derived intra modes according to the intra modes of left and above neighbouring blocks. The intra modes in candidate list consist of modes generated by rough modes decision (RMD) process and MPMs, which compete with each other in sense of full RDO. To further improve the coding performance of intra prediction, several novel coding tools are proposed in VVC, such as wide-angle intra prediction for non-square blocks, cross-component linear model prediction (CCLM), position dependent intra prediction combination (PDPC), multiple reference line intra prediction (MRL), intra sub-partition (ISP) and matrix-weighted intra prediction (MIP). These coding tools bring impressive performance improvement and more details can also be seen in reference [1].

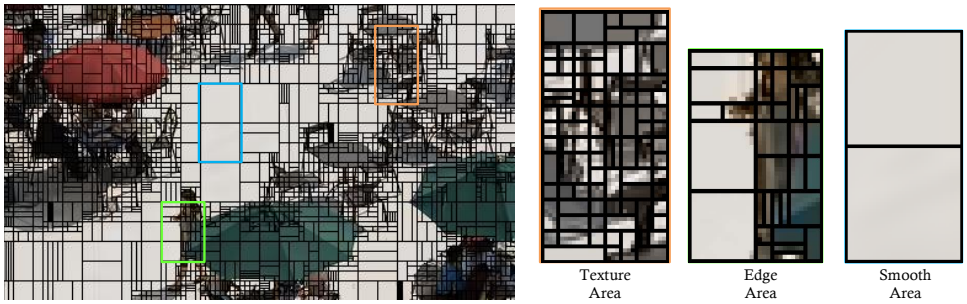

Figure 2: An example of quadtree with nested multi-type tree coding block structure.

The Proposed Early Termination Algorithm in VVC

In this section, the proposed method is motivated by the observations through analyzing the statistical characteristics of encoded videos. There are three checkpoints in proposed method, including splitting or not splitting, horizontal partition or vertical partition, and binary partition or ternary partition. Each checkpoint is only triggered for blocks with specific block size.

Statistical Analysis

Generally, texture information is one of crucial evidences of block partition. In the smooth area, the partition tends to be simple and the blocks are mostly large, while more feasible blocks occur in the complex texture area. To verify the correlation between texture characteristics and CU partition structure, the final CU partition after RDO is printed with blacking the block boundaries in the reconstructed video frame. As shown in Fig.2 for *BQSquare*, we can see that the block partition structure is strongly correlated with image contents. Not only in smooth area but in texture and edge area, CU segmentations are mostly consistent with texture. Therefore, the image contents are able to provide hints to guide CU size decision. In addition, different from HEVC, VVC adopts the directional partition including horizontal and vertical direction. In Fig.2, we can also find that the content coherence is relative to the partition direction. Taking edge area as an example, the texture is extended into vertical direction along with the woman's body, thus the partitions tend to mostly be vertical as well.

Generally, there are four directions are commonly used including horizontal, vertical, left-bottom to right-above (45°) and left-above to right-bottom (135°). In this work, four directional gradients $Grad_h$, $Grad_v$, $Grad_{45}$ and $Grad_{135}$ are generated by the following equation:

$$\begin{aligned} Grad_h &= \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} abs(org(i+1,j) - org(i,j)) \\ Grad_v &= \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} abs(org(i,j+1) - org(i,j)) \\ Grad_{45} &= \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} abs(org(i+1,j) - org(i,j+1)) \\ Grad_{135} &= \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} abs(org(i+1,j+1) - org(i,j)) \end{aligned} \tag{1}$$

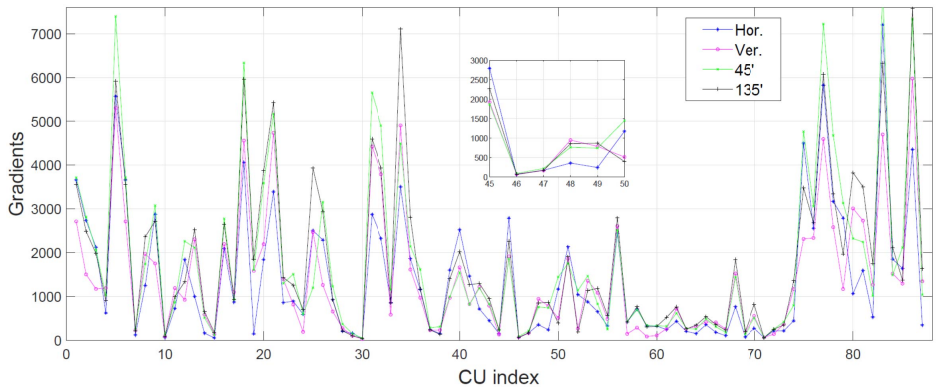

Figure 3: The four directional gradients of neighbouring coding blocks.

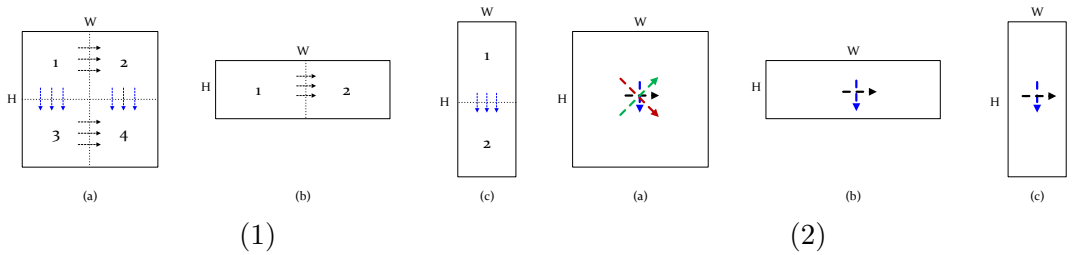

Figure 4: The directional gradients of current block and its sub-blocks in various block type.

where H and W denote the height and weight of current block, respectively. (i, j) are the position index and org represents the original pixel. Four directional gradients can reveal the texture characteristic by comparing the gradient value of each direction. As shown in Fig.3, four directional gradients of CU in coding order are displayed. Obviously, if the gradient of one direction is much smaller than that of other three, it means the content tends to follow its direction. The gradients of four directions sharing the similar gradient near zero imply that current CU is smooth and likely to keep not splitting. In addition, if the difference between neighbouring blocks is large enough, they are likely to belong to different CU. Therefore, current block can be split or skipped by comparing the gradients of its sub-blocks.

The Proposed Early Termination algorithm

The flexible block partition can improve coding performance by providing content-adaptive coding structure, but increase the load of RDO mode decision. Removing unnecessary RDO checks is one of effective method to reduce the time cost. According to the texture gradients of current block, some inapposite partition patterns can be skipped.

Firstly, for CU with size larger or equal to 16×16, let $split\_flag$ be the splitting flag of current block. If $split\_flag$ equals to 1 and current block will directly jump to its sub-blocks without intra prediction. The partition will be terminated if $split\_flag$ is 0. Otherwise, remain the same as that in VVC. To decide the $split\_flag$ of current block,

Figure 5: The examples of binary partition and ternary partition blocks.

the gradients of four sub-blocks are calculated in horizontal and vertical direction. The sub-blocks with index $(1, 2, 3, 4)$ are shown in Fig.4(1)(a), where H and W meet the condition that the ratio of H and W is smaller than 5. For the other blocks as shown in Fig.4(1)(b) and (c), we divide it into two parts. The black dotted arrow and blue dotted arrow represent the horizontal and vertical gradient, respectively. For the blocks in first case, its $split\_flag$ is calculated by:

$$split\_flag = \begin{cases} 1, & if \left(\begin{array}{l} BR(Grad_h1, Grad_h2) \,||\, BR(Grad_h3, Grad_h4) \\ ||\, BR(Grad_v1, Grad_v3) \;\; ||BR(Grad_v2, Grad_v4) \end{array} \right) \\ 0, & if \left(\begin{array}{l} SR(Grad_h1, Grad_h2) \;\&\; SR(Grad_h3, Grad_h4) \\ \&\; SR(Grad_v1, Grad_v3) \;\&\; SR(Grad_v2, Grad_v4) \end{array} \right) \\ -1, & otherwise. \end{cases} \tag{2}$$

where $Grad_h1$, $Grad_h2$, $Grad_h3$ and $Grad_h4$ denote the horizontal gradient of four sub-blocks, respectively. Similarly, $Grad_v1$, $Grad_v2$, $Grad_v3$ and $Grad_v4$ are the vertical gradient of four sub-blocks. $BR(\cdot)$ and $SR(\cdot)$ are the functions used to compare the gradients of two regions, which are defined as:

$$BR(g1, g2) = \begin{cases} true, & if(g1/g2 \;\geq\; ThrB \,||\, g2/g1 \;\geq\; ThrB) \\ false, & otherwise. \end{cases} \tag{3}$$

$$SR(g1, g2) = \begin{cases} true, & if(g1/g2 \;\leq\; ThrS \,\&\, g2/g1 \;\leq\; ThrS) \\ false, & otherwise. \end{cases} \tag{4}$$

where $ThrB$ and $ThrS$ are pre-set thresholds initialized by the statistical values. For the block in Fig.4(1)(b) and (c), if $BR(Grad_h1, Grad_h2)$ or $BR(Grad_v1, Grad_v2)$ is true, the block is splitting. Otherwise, keep the original block partition process.

Secondly, for the leaf node blocks of quaternary tree, there are four split patterns including binary tree horizonal partition, binary tree vertical partition, ternary tree horizonal partition and ternary tree vertical partition. Firstly, we compare the gradients of four directions and then label the block with vertical or horizontal partition, thus the other two patterns can be avoided. In particular, for the blocks in Fig.4(2)(a), its splitting direction dir_idx can be judged by:

$$dir\_idx = \begin{cases} 0, & if \left(\begin{array}{l} Grad_v \geq ThrV \cdot Grad_h \;\&Grad_v \geq ThrV \cdot \dfrac{H}{\sqrt{H^2+W^2}} \cdot Grad_{45} \\ \&Grad_v \geq ThrV \cdot \dfrac{H}{\sqrt{H^2+W^2}} \cdot Grad_{135} \end{array} \right) \\ 1, & if \left(\begin{array}{l} Grad_h \geq ThrH \cdot Grad_h \;\&Grad_h \geq ThrH \cdot \dfrac{W}{\sqrt{H^2+W^2}} \cdot Grad_{45} \\ \&Grad_h \geq ThrH \cdot \dfrac{W}{\sqrt{H^2+W^2}} \cdot Grad_{135} \end{array} \right) \\ -1, & otherwise \end{cases} \tag{5}$$

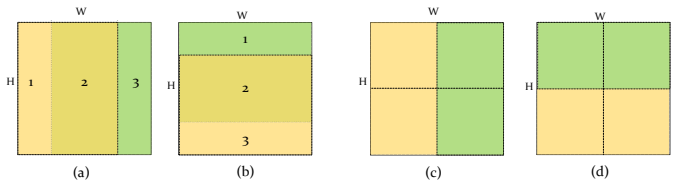

Figure 6: The region of sub-blocks in ternary tree and binary tree partition.

where $ThrV$ and $ThrH$ are two thresholds valued by empirical analysis. when dir_idx is 0, current block tends to be horizontally zoned, the heuristic searches of binary tree vertical partition and ternary tree vertical partition are skipped. Similarly, the binary tree horizontal partition and ternary tree horizontal partition are eliminated when dir_idx equals to 1. Otherwise, none of above four partition patterns should be omitted. For block in Fig.4(2)(b), if its horizontal gradient $Grad_h$ is smaller than $ThrG$ and it also meets $Grad_v \geq ThrV \cdot Grad_h$, the block is split in horizontal. Similarly, if the vertical gradient $Grad_v$ of the block in Fig.4(2)(c) satisfies the same condition, it is split in vertical direction.

Binary partition and ternary tree are introduced to remedy the shortage of quaternary tree partition. If the block is evenly full of complex texture, quaternary tree seems to be a better choice dividing it into four parts instead of two or three in theory. The benefit of choosing binary or ternary tree comes from that these blocks separated into two or three parts are able to be predicted well enough, for example, the contents in over half region share the similar details. By observing the contents within binary partition and ternary partition blocks, some typical phenomenons can be perceived. As shown in Fig.5, if the block is binary partition, the contents between left and right or bottom and above are obvious different. For the ternary partition, three sub-blocks show the totally diverse details or the fringe sub-block exerts disparate texture compared with other two sub-blocks. According to these observations, we calculate the flag bt_skip to skip binary partition as following:

$$bt\_skip = \begin{cases} 1, & if \left(\begin{array}{l} dir\_idx! = 0 \ \& \ (SR(Grad_v1,\ Grad_v2) \ || \ SR(Grad_v2,\ Grad_v3)) \\ || \ dir\_idx! = 1 \ \& \ (SR(Grad_h1,\ Grad_h2) \ || \ SR(Grad_h2,\ Grad_h3)) \end{array} \right) \\ 0, & otherwise \end{cases} \tag{6}$$

where the index $(1, 2, 3)$ presents the ternary partition sub-block index as shown in Fig.6(a) and (b). In Fig.6(c) and (d), each part can be constituted by the sub-blocks in Fig.4(1)(a), thus the flag tt_skip to skip ternary partition is calculated by reusing the gradients in Eq.(2) as:

$$tt\_skip = \begin{cases} 1, & if \left(\begin{array}{l} dir\_idx! = 0 \& (SR(Grad_v1,\ Grad_v3) || SR(Grad_v2,\ Grad_v4)) \\ \&(BR(Grad_h1,\ Grad_h2) || BR(Grad_h3,\ Grad_h4)) \\ || \ dir\_idx! = 1 \& (SR(Grad_h1,\ Grad_h2) || SR(Grad_h3,\ Grad_h4)) \\ \&(BR(Grad_v1,\ Grad_v3) || BR(Grad_v2,\ Grad_v4)) \end{array} \right) \\ 0, & otherwise \end{cases} \tag{7}$$

Finally, the overall early termination algorithm for each block can be summarized by the flowchart in Fig.7. By using the gradients of current block and its sub-blocks, the $split\_flag$, dir_idx, bt_skip and tt_skip can be deduced. Based on these variables, some heuristic searches are skipped or terminated early.

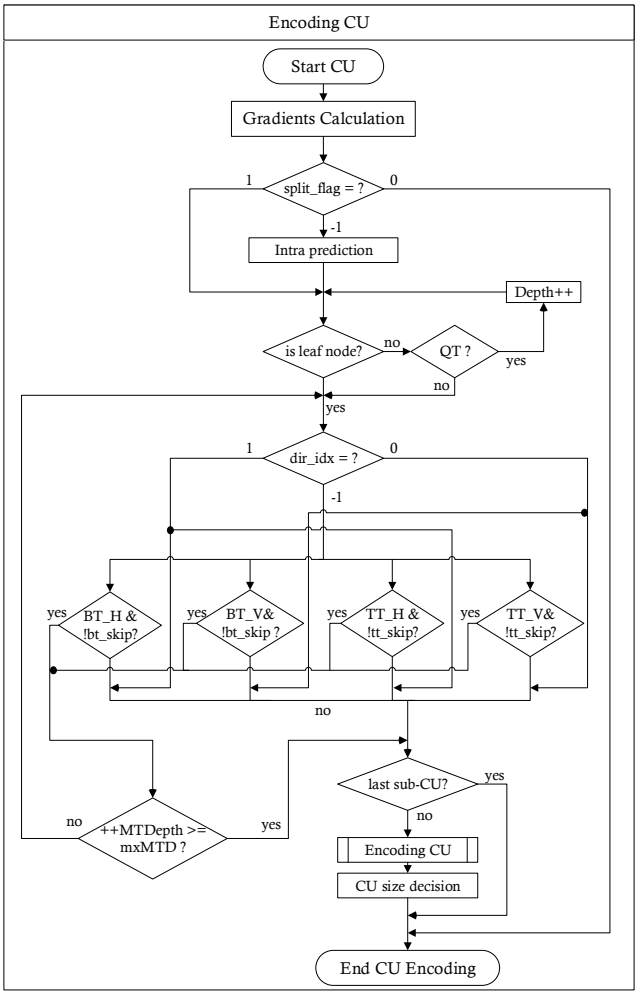

Figure 7: The flowchart of proposed early termination method.

Experimental Results

To verify the coding performance, the proposed method is implemented in the VVC software reference VTM5.0 [2] and tested by the JVET common test condition [10] under All Intra (AI) configuration. 100 frames of each sequence from A1 class to E class are encoded. The coding performance evaluation includes RD performance and encoding time saving (TS), and the latter is calculated by $TS = ((T_{anchor} - T_{proposed})/T_{anchor}) \times 100\%$. Table 1 shows the coding performance of the proposed method, where both luma and chroma are listed with Y, U and V component. There are two sets of experiments in term of setting 1 and setting 2 based on the thresholds $ThrB$, $ThrS$, $ThrH$, $ThrV$ and $ThrG$. In setting 1, five thresholds are set as 5.0, 1.0, 1.1, 1.1 and 54, respectively, while they are assigned as 2.0, 1.2, 1.0, 1.0 and 108 in setting 2 respectively. In addition, $ThrB$, $ThrS$ and $ThrG$ can be further adjusted by the block size and QP. From Table 1 we can see that the averaged time saving is around 38% with 0.76% Y BD-rate loss on average in setting 1. In setting 2, the averaged BD-rates of Y , U and V are 1.23%, 1.57% and 1.73%, respectively with about 51%

Table 1: The coding performance of the proposed method

Class/Sequences		Setting 1				Setting 2			
		Y	U	V	TS	Y	U	V	TS
A1	Tango2	0.71%	1.27%	1.60%	36.62%	1.74%	2.30%	2.28%	62.64%
	FoodMarket4	0.41%	0.44%	0.03%	24.90%	4.68%	4.06%	4.18%	43.01%
	Campfire	0.61%	0.15%	1.02%	43.85%	1.23%	1.67%	2.02%	58.63%
A2	CatRobot1	1.02%	1.79%	1.25%	41.63%	1.15%	1.74%	1.62%	54.62%
	DaylightRoad2	0.91%	2.28%	1.75%	43.77%	0.51%	1.93%	1.36%	66.17%
	ParkRunning3	0.23%	0.65%	0.73%	29.89%	1.17%	0.77%	0.75%	39.07%
B	MarketPlace	0.52%	1.22%	0.94%	39.68%	0.83%	1.11%	1.71%	62.62%
	RitualDance	0.62%	0.44%	0.78%	30.39%	2.18%	2.18%	3.01%	48.55%
	Cactus	0.83%	1.27%	1.71%	42.26%	0.63%	1.03%	0.98%	55.44%
	BasketballDrive	0.67%	1.29%	1.88%	40.48%	0.93%	2.02%	1.94%	65.35%
	BQTerrace	0.81%	1.57%	1.49%	40.07%	1.57%	1.40%	1.65%	47.91%
C	BasketballDrill	1.59%	2.15%	2.75%	41.00%	1.17%	1.26%	1.03%	54.39%
	BQMall	1.18%	2.25%	2.00%	42.18%	0.79%	1.47%	1.75%	50.89%
	PartyScene	0.57%	1.03%	1.20%	39.75%	0.81%	0.95%	0.59%	46.42%
	RaceHorses	0.53%	0.37%	0.65%	38.72%	1.15%	1.31%	1.84%	45.51%
D	BasketballPass	0.98%	1.87%	1.51%	35.31%	0.84%	0.47%	0.91%	49.30%
	BQSquare	0.75%	2.21%	2.41%	39.73%	0.68%	0.36%	1.52%	44.44%
	BlowingBubbles	0.63%	0.64%	0.53%	38.31%	0.80%	1.79%	0.92%	43.48%
	RaceHorses	0.58%	0.77%	0.74%	35.45%	0.62%	0.93%	1.40%	39.45%
E	FourPeople	1.01%	1.33%	1.63%	34.71%	0.81%	2.18%	2.40%	47.53%
	Johnny	0.90%	0.94%	0.97%	35.47%	1.51%	1.93%	2.65%	50.34%
	KristenAndSara	0.68%	1.53%	2.22%	34.57%	1.20%	1.76%	1.50%	46.52%
Overall		0.76%	1.25%	1.35%	37.67%	1.23%	1.57%	1.73%	51.01%

time saving. For some sequences, such as *DaylightRoad*2 and *BasketballDrive*, the time saving is up to 65%. Compared with anchor, the CU partition in proposed method follows the frame texture commendably as shown in Fig.8, especially in the red box area where the texture is horizontal but it is split in vertical in anchor.

Conclusion

In this paper, an early termination algorithm based on directional gradients are proposed to skip some unnecessary heuristic searches in CU size decision process in VVC intra coding. Via analyzing the partition characteristics and directional gradients of each block, find some rules to utilize for various block types. The proposed method is easy to implement once the relative directional gradients are ready, thus no extra computation burden is brought. Experimental results show that the proposed method is able to save encoding time more than half with about 1.2% BD-rate loss. It is worth to mention that the encoding time saving is up to 65% for some sequences.

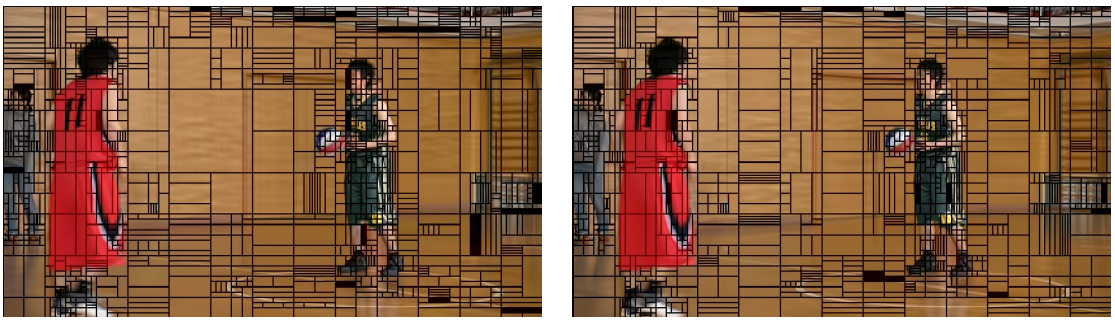

Figure 8: The CU partition comparison between anchor and proposed method.

Acknowledgment

This work was supported by National Key Research and Development Project (2019YFF0302703), MoE-China Mobile Research Fund Project (MCM20180702) and High-performance Computing Platform of Peking University, which are gratefully acknowledged.

References

[1] S. Liu B. Bross, J. Chen, "Versatile video coding (draft 6)," *15th JVET meeting: Gothenburg, document JVET-O2001*, July 2019.

[2] JVET Group, "Vtm software reference," `https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM`.

[3] G. J. Sullivan, J. R. Ohm, W. J. 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.

[4] Seunghyun Cho and Munchurl Kim, "Fast cu splitting and pruning for suboptimal cu partitioning in hevc intra coding," *IEEE Transactions on Circuits Systems for Video Technology*, vol. 23, no. 9, pp. 1555–1564, 2013.

[5] 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.

[6] Xiaolin Shen and Lu Yu, "Cu splitting early termination based on weighted svm," *Eurasip Journal on Image Video Processing*, vol. 2013, no. 1, pp. 4, 2013.

[7] Yun Zhang, Zhaoqing Pan, Na Li, Xu Wang, Gangyi Jiang, and Sam Kwong, "Effective data driven coding unit size decision approaches for hevc intra coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3208–3222, 2018.

[8] Xingang Liu, Yayong Li, Deyuan Liu, Peicheng Wang, and Laurence T. Yang, "An adaptive cu size decision algorithm for hevc intra prediction based on complexity classification using machine learning," *IEEE Transactions on Circuits Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[9] H. Yang, L. Shen, X. Dong, Q. Ding, P. An, and G. Jiang, "Low complexity ctu partition structure decision and fast intra mode decision for versatile video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2019.

[10] X. Li V. Seregin K. Shring F. Bossen, J. Boyce, "Jvet common test conditions and software reference configurations for sdr video," *14th JVET meeting: Geneva, document JVET-N1010*, March 2019.