Hindawi
Scientific Programming
Volume 2021, Article ID 5527713, 12 pages
https://doi.org/10.1155/2021/5527713

Research Article

Fast CU Partition Decision Based on Texture for H.266/VVC

Qiuwen Zhang, Tengyao Cui, and Rijian Su

College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China

Correspondence should be addressed to Rijian Su; rijiansu@126.com

Received 5 January 2021; Revised 7 April 2021; Accepted 17 May 2021; Published 25 May 2021

Academic Editor: Antonio J. Peña

With the development of multimedia equipment and the increasing demand for high-quality video applications, the traditional video coding standard, H.265/High Efficiency Video Coding (HEVC), can no longer effectively satisfy the requirements. To promote the development of high-quality video, a new generation video coding standard, H.266/Versatile Video Coding (H.266/VVC), is established, and it is the inheritance and development of H.265/HEVC. It not only retains many mature technologies and methods in HEVC but also adds some new coding tools, such as wide-angle prediction and Multitype Tree (MTT) partition structure. The MTT partition structure brings a more flexible partition method of Coding Unit (CU), but the accompanying increase in computational complexity is unacceptable. In order to ensure an effective balance between coding efficiency and coding quality, a fast CU partition algorithm based on texture is proposed in this paper. First, the texture complexity of the neighboring CU is used as a threshold for evaluating the complexity of the current CU, so as to skip the unpromising depth. Then, the gradient features are extracted to determine whether the Quad-Tree (QT) partition is executed. Finally, the improved Canny operator is used to extract edge features, and the partition mode in the horizontal or vertical direction is excluded. The algorithm was embedded in VTM7.0, and the video sequences with different resolutions were tested under general experimental configuration. Simulation experiment results show that the average time saving of this method reached 50.56% compared with the anchor algorithm. At the same time, the average BDBR is increased by 1.31%.

1. Introduction

The development of video hardware equipment and network transmission technology provides the basic conditions for the wide application of ultradefinition video. However, the complexity of encoding and a huge amount of data are obstacles restricting its development. Therefore, a new generation of coding standards needs to be established. Under the joint work of the Video Coding Experts Group (VCEG) and the Moving Picture Experts Group (MPEG) [1], the development of VVC is proceeding steadily, and phased results have been achieved. The hybrid encoding framework from HEVC was adopted by VVC. On this basis, some innovative encoding techniques are introduced and a lot of optimizations are implemented [2]. For example, VVC increased the angle modes from 35 to 67, which greatly improved the accuracy of prediction [3]. VVC eliminates the concept of Prediction Unit (PU) and Transform Unit (TU). The iconic difference between VVC and HEVC is that HEVC uses Quad-Tree (QT) partition structure, while VVC uses MTT partition structure including QT partition structure. Compared with HEVC, the average bitrate of VVC is saved by 50% when the subjective quality remains almost unchanged. As the decreases of bitrate, VVC also brings huge computational complexity.

In HEVC, each frame is divided into several Coding Tree Units (CTU). A CTU contains a luma block and two chroma blocks with corresponding sizes. The maximum allowed size of the luma block is 128×128, and the maximum allowed chroma block is 64×64 [4]. In addition, only the QT partition type is allowed in HEVC, which limits the shape of sub-CUs to square. But, VVC introduces a MTT partition structure, including five partition structures: Horizontal Binary Tree partition (BT_H), Vertical Binary Tree partition (BT_V), Horizontal Ternary Tree partition (TT_H), Vertical Ternary Tree partition (TT_V), and QT. Among them, the size ratio of the three parts in the TT partition structure is 1 : 2 : 1. MTT partition structure allows CU to be asymmetrically partitioned [5]. In theory, the width and height of the CU can be combined from 4 to 128, arbitrarily. However, according to configuration restrictions, a 128×128 pixel CTU will be divided into 4 sub-CUs by QT, directly. When the sub-CU is further divided, the QT or MTT partition structure is considered according to the texture characteristics. In practice, the range of width and height of CU is controlled within 4 to 64.

In addition, there is a partition rule that cannot be ignored; without considering the depth levels, any of the five partition types can be executed under a QT node, but only the TT and BT partition modes can be executed under MTT node [6]. The CU partition mode is shown in Figure 1. Among them, the black dot represents the CTU and the blue dot represents the root node. The minimum allowed QT leaf node size is 4×4, the maximum allowed TT root node size is 32×32, the maximum TT depth is 3 levels, and the minimum allowed BT leaf node size is 4×4.

The introduction of more flexible CU partition modes leads to some redundant partition problems. To solve the problem, on the one hand, VVC strictly limits the partition of current CU based on the depth level, size, and neighboring CUs. On the other hand, these partition rules are adopted to ensure that the CU is divided reasonably. Two common situations are as follows:

(a) If the current CU is divided by TT_V, the middle area of the three parts after the partition can no longer be BT_V divided. Otherwise, it is equivalent to two BT_V partitions. The same applies to the horizontal partition.

(b) If the current CU is divided by BT_V and its left part is divided by BT_H, its right part cannot be divided by BT_V anymore. Otherwise, it is equivalent to a QT partition [7]. The same applies to the horizontal partition. Figure 2 shows the disallowed partition method.

The introduction of the MTT partition structure improves the coding accuracy, but more Rate Distortion Optimization (RDO) processes are required to determine the optimal partitioning method. The time required for RDO accounts for a large proportion of the overall coding time, which results in a significant increase in coding time and complexity. Therefore, if the number of candidates that need to perform RDO can be reduced, the time cost of RDO will inevitably be reduced. To achieve the above purpose, a fast CU partition decision algorithm based on texture is proposed in this paper.

2. Related Work

In recent years, research on VVC has gradually been enriched. As a new generation video coding standard under development, the research results of VVC are still limited. In fact, VVC is developed on HEVC; the research on the intra prediction and CU partition decision algorithm for HEVC has an important reference value for VVC.

Two types of methods are frequently used for CU partition. The first type of methods is based on feature correlation, such as texture, depth, and neighboring blocks, to accelerate the CU partition process. The authors of [8–21] judge texture features by variance or gradient to make skip or early termination. In [8], a gradient method based on texture complexity and gradient-based preprocessing steps are proposed to reduce the number of candidate modes and make early decisions for CU splitting. A fast mode decision algorithm based on gradient is proposed in [9], which calculates the gradient direction and eliminates unnecessary modes based on histogram distribution to reduce the candidate modes before performing the Rough Mode Decision (RMD). The correlation of encoded information among different CU depth levels and the spatial correlation in neighboring CUs are analyzed in [10], where these correlations are used to guide an early skip mode. The study in [11] proposed a fast texture-based CU partition algorithm, in which the edge complexity of the CU in the vertical, horizontal, and 45° and 135° diagonal directions is used as an important texture feature to determine the CU partition. In [12], the time domain and spatial domain information is used as the conditions for determining the depth range of the CU. CU depths that are rarely used will be skipped or terminated early. In [13], a direction with the smallest directional variance is chosen as the dominant direction, so the number of mode candidates that require RDO processing is reduced. In order to reduce the candidate number of CU size, an early determination of CU size decision with adaptive thresholds based on texture characteristics is proposed in [14] to speed up coding. In [15], the intraprediction mode of neighboring CUs is used to adaptively select a small group of candidates into the RDO process. The relationship between the impossible mode of the current CU and the distortion distribution of the sub-CUs is studied in [16] to skip or terminate unnecessary partition modes. A fast algorithm based on texture complexity and directional energy distribution is proposed in [17] to provide a basis for the early termination strategy of CU partition. An intelligent classification model based on CU attributes is designed in [18] to delete nonpromising prediction modes, in addition, where unnecessary predictions at the remaining CU level are based on the different termination probabilities caused by different optimal modes. A fast content-based coding algorithm is proposed in [19], and the characteristics of the video content are analyzed by the pixel variance, gradient, and mean value of the CU. These characteristics are used in combination with the depth level of current CU and the neighboring CUs prediction modes in time and space to achieve fast CU partition decision. The study in [20] introduces the Otsu method to measure the texture complexity of each LCU, skipping some CU depth levels according to the texture complexity. At the same time, an improved Sobel mask is applied to measure the gradient direction of the PU for reducing the number of candidates in intraprediction. In [21], according to the direction gradient, the MTT partition mode in the horizontal or vertical direction is determined early to skip unnecessary partition modes.

The second type of mode is another research hotspot in recent years. Convolutional Neural Network (CNN) and Machine Learning (ML) methods have been studied in

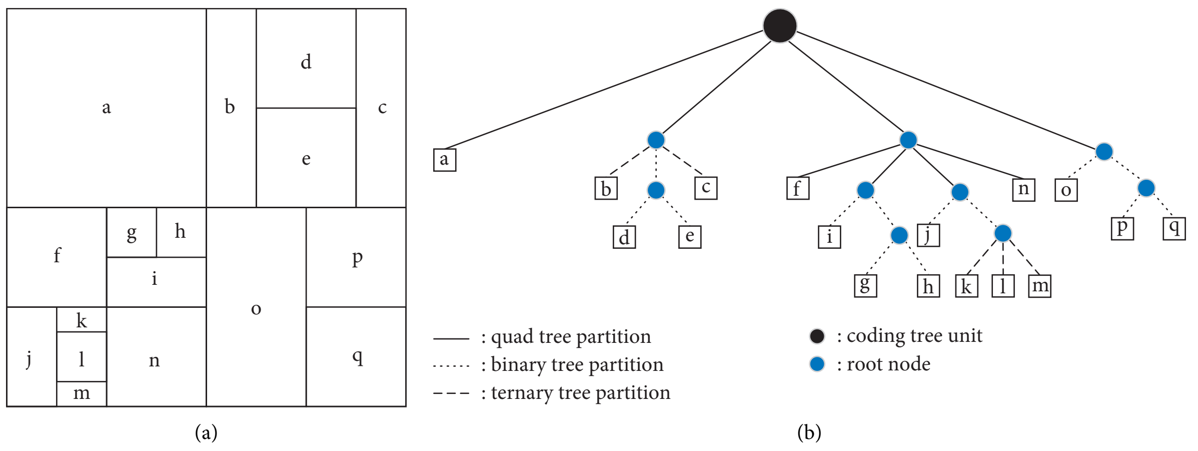


Figure 1: The schematic diagram of the CU partition mode. (a) The CU partition mode. (b) Tree structure corresponding to CU partition mode.

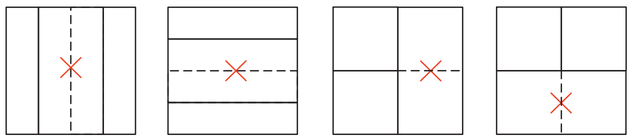


Figure 2: The disallowed partition method.

depth. Some intelligent classifiers are designed to eliminate redundant partition modes to speed up the encoding process [22–34]. The CNN algorithm builds a multilevel classifier by different levels [22–26]. The significant characteristic of ML and CNN is extracted features, the smallest error function is found, and the best threshold is trained [27–34]. In [22], QTBT structure is analyzed by statistical methods and the CNN architecture is designed. The study in [23] proposes a fast CU partition method based on CNN, which can reduce the coding complexity by the heterogeneity of texture features. For different CU shapes, [24] proposed an adaptive CU partition method based on mixed variable CNN. The algorithm realizes shape adaptation by the variable size of the convergence layer, and the original information is retained by the convergence layer of CNN. Paper [25] analyzes the CU texture characteristic of the source image by CNN, and quantization parameters are introduced into the system to reduce the number of RDO and improve prediction accuracy. In [26], CNN was used to predict CTU depth, and a CNN database was built to improve prediction accuracy by using CU attributes. In [27], a fast CU partition is proposed, which terminates the CU partition process by online learning method based on Pegasos. In [28], an early termination scheme for CU partition based on Bayesian algorithm is proposed. In [29], a progressive Bayesian classifier composed of cascaded online classifiers is designed, which uses Bayesian risk to balance efficiency and complexity. In [30], the texture features are extracted, and the early decision is designed as a skip or no skip classification problem, where feature extraction and training are crucial for ML-based methods. In [31, 32], CUs with sizes 64×64 and 32×32 are predicted by training random forest. The CU partition decision is defined as a three-level hierarchical binary decision problem in [33]. Based on the optimal weighting factor of the support vector machine (SVM), a classifier is designed to help control the error prediction, and the complexity is allocated reasonably according to the coding depth. In [34], binary and multivariate SVMs are used for CU partitioning, and a reviewer system is added to optimize sample selection.

3. Proposed Algorithm

Macroscopically, the texture of CU is distinguished into two categories: the homogeneous area and the complex texture area. Considering the accuracy and efficiency of encoding, the homogeneous regions are divided into low depths to improve encoding efficiency, and the regions with complex textures are divided into higher depth to ensure encoding accuracy. The maximum CU size allowed in VVC is 128×128. A 128×128 CU is forcibly divided into 64×64 CUs by QT when performing the partition process. In addition, the maximum size allowed for BT and TT partition in VTM is 32×32, and the maximum leaf node size allowed for QT is 16×16. In other words, there are three methods to divide 32×32 CU, including QT, MTT, and no partition. Therefore, compared with other sizes, the processing of 32×32 CU is the most complex. According to the previous research, we found that the smaller the size of CU, the less the time required to execute the partition process. For CU with a small size, considering the complexity to solve the computational complexity of RD cost by fast algorithm, there is no foreseeable benefit of dealing with it on the basis of existing research. In order to get better optimization, the proposed algorithm takes the 64×64 CU as the starting point and focuses on accelerating the 32×32 CU partition process.

Based on the above considerations, we proposed a fast CU partition algorithm. The algorithm terminates the nonpromising partitioning mode early based on texture features, and the number of candidates that execute RD-cost is reduced, thereby accelerating the CU partition process. The specific steps of the proposed algorithm are as follows:

Step 1: early termination of CU partition: for 64×64 CU and 32×32 CU, the average complexity value of neighboring CUs is used as a reference threshold to exclude the higher or lower depth early. Specifically, the 64×64 CU determined whether to early perform QT partition. The 32×32 CU will judge whether it is necessary to be further divided. If 32×32 CU is divided, it will continue to Step 2. Otherwise, the 32×32 CU will be terminated for further partition.

Step 2: the Sobel operator is used to extract the gradient features and judge whether the QT partition is selected. If the partition condition is satisfied, the current CU will skip the MTT mode and execute the QT mode. Otherwise, Step 3 will be executed.

Step 3: decision of QT partition: two candidate partitions are selected based on Canny operator. The Canny operator is used to extract the edge features, and the MTT_H or MTT_V mode will be eliminated. The remaining two modes are selected as candidates for RDO calculation; the one with the smallest RD-cost is determined as the optimal mode.

3.1. Early Termination of CU Partition.

The early termination method of CU partition based on gradient is often used. However, there are limitations in judging the homogeneity of CU based on gradient characteristics; that is, smaller gradient value only appears in areas with flat textures, but areas with flat textures do not necessarily have smaller gradient value. If the region is smooth, but the variable textures exist in microscopic regions, the total gradient may become larger. The reason is that the increased absolute gradient value indirectly leads to an increase in the total gradient value. Considering the disadvantages of the above methods, in the paper, the texture complexity based on the dissimilarity of luminance is adopted to determine whether to further divide the CU, so as to terminate the nonpromising partition mode early.

Several studies have described that the complexity of each block is different at the same depth level in the same frame. The problem is difficult to solve fundamentally, that is, to find a threshold that defines the complexity for all videos by statistical analysis and artificial setting. In most instances, the neighboring CU has similar texture, which makes the complexity values close to each other. If the complexity of neighboring CUs is used as the threshold for evaluating texture features of the current CU, the number of RD-cost calculation processes will be reduced significantly. The left CU and the top CU are used as references, and the average of their complexity values is used as a threshold to compare with the complexity of the current CU. The current CU judges whether to be further divided according to the comparison result.

For the image, there is a significant dissimilarity in intensity between the rich textures region and the homogeneous region, which is used to reflect the complexity of texture. The dissimilarity of the current luma CU is calculated as

$$L_d = \sum_{i=1}^{W}\sum_{j=1}^{H}\left[p(i,j) - \frac{1}{W \times H} \times \left(\sum_{i=1}^{W}\sum_{j=1}^{H} p(i,j)\right)\right]^2, \quad (1)$$

where W and H are the width and height of the current CU, respectively, (i, j) is the pixel coordinates, and $p(i, j)$ is the value of luminance. According to the dissimilarity of the luminance L_d, the luminance complexity of the current CU is calculated as

$$\mathrm{CP} = \log^{(L_d)}. \quad (2)$$

In VVC, the QT split mode is allowed to be performed in 64×64 CU, while both QT and MTT split modes are allowed to be performed in 32×32 CU. If the texture complexity of the current CU is higher than the threshold, the 64×64 CU will be divided by QT. However, the 32×32 CU will enter the next stage, and the QT or MTT mode will be judged. The algorithm steps are as follows:

(1) The texture complexity of current CU is calculated by (1) and (2), denoted as CP_C.

(2) Get the texture complexity of the neighboring CUs; the complexity of the CU on the top and on the left is denoted as CP_U and CP_L, respectively. The average of CP_U and CP_L is used as the reference threshold CP_T. If the neighboring CU is found on the left or on the top, we take the complexity of this neighboring CU as the reference threshold CP_T.

(3) If $\mathrm{CP}_C < \mathrm{CP}_T$, it means that the texture of the current CU is simpler than that of the reference CU. If the depth level of the reference CU is low, the current CU will not be divided to the high depth level; if $\mathrm{CP}_C > \mathrm{CP}_T$, it means that the texture of the current CU is more complicated than the reference CU. If the reference CU has been divided into small CUs, the low level partition is excluded from further processing.

3.2. Decision of QT Partition.

If the 32×32 CU has not been terminated, this step will be executed to terminate the MTT. The Sobel operator will be used to extract the gradient index for supporting the further decision for CU partition. The absolute gradient F_x in horizontal and the absolute gradient F_y in vertical direction are extracted, which are obtained, respectively, by

$$F_x(i, j) = A_{i,j} \times \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad (3)$$

$$F_y(i, j) = A_{i,j} \times \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad (4)$$

where (i, j) means the coordinates of the current pixel. $A_{i,j}$ represents an initial pixel matrix constructed with the current coordinate as the center.

We use (5) to calculate the total gradient F_X in the horizontal direction. At the same time, (6) is used to calculate the total gradient F_Y in the vertical direction.

$$F_X = \sum_{i=1}^{W} \sum_{j=1}^{H} \text{abs}\left(F_x(i, j)\right), \quad (5)$$

$$F_Y = \sum_{i=1}^{W} \sum_{j=1}^{H} \text{abs}\left(F_y(i, j)\right). \quad (6)$$

For the boundary pixels of the CU, including the top and bottom row and the left and right column of the current CU, we average the pixel values for inside and outside of the boundary to fill the boundary. Compared with single-sided filling, this method improves the accuracy of filling.

Here, we set a threshold T_h 1. The quotient of F_X and F_Y means the gradient tendency of the current CU. For F_X and F_Y, the bigger one is used as the denominator and the smaller one is used as the numerator. As the ratio decreases, the gradients in the horizontal and vertical directions tend to be similar, and the monotonicity of the current CU becomes more pronounced. In most cases, the ratio of F_X and F_Y fluctuates within a certain range. We hope to find a turning point to qualitatively determine the texture trend. This turning point is the threshold T_h 1. If the calculation result is less than T_h 1, the current CU is considered monotonic. In addition, it is necessary to add a condition, the threshold T_h 2, to ensure that the number of QT is controlled within a reasonable range.

It should be noted that the monotonic texture does not mean that the CU is homogeneous. If the area is composed of symmetrical or repeated textures, the partition process cannot be skipped at this time, because these special textures are exactly what we want to accurately reflect. The most prominent feature of monotonic texture is the vertical and horizontal symmetrical structure, like the cross grid. The QT is a kind of partition method with symmetrical structure. After partition, four sub-CUs are produced with the same size and shape, which conforms to the symmetry characteristics of the monotonic texture. Therefore, the QT can play the best performance when processing horizontal and vertical symmetrical graphics.

To meet the following two conditions, the current CU will perform the QT partition, and the MTT partition will be terminated early:

(1) $F_X/F_Y < T_h1$ or $F_Y/F_X < T_h1$

(2) $T_h2 < F_X$ and $T_h2 < F_Y$

To avoid interference, we chose videos that were different from the test sequence to find the threshold. First, we set T_h 2 as a constant, and then, running the algorithm by different values, Time Saving (TS) and Bjøntegaard Delta Bitrate (BDBR) are considered synthetically to choose a compromise point as T_h 1. Table 1 shows the encoding performance with different T_h 1. As we can see, with the increase of T_h 1, TS presents a trend of continuous decline. After reaching 2.9, TS decreases slowly and tends to be stable because more CU will be QT and more computing resources will be occupied with the increase of T_h 1. However, when the threshold reaches the critical point, the number of QT tends to be saturated. Similar to TS, after BDBR reaching 2.9, the downward trend tends to stabilize. Taking the execution space of the next algorithm into account, the current CU cannot be partitioned too much by QT; otherwise the overall algorithm accuracy will be reduced. We choose the point, which is 3.3, as the value of T_h 1.

After determining T_h 1, we show the TS and BDBR obtained by different values in Table 2. We can find the obvious difference in Table 1 from Table 2. As shown in Table 2, the TS under different values is stable within a small range, which reflects the limited impact of T_h 2 on TS. However, BDBR fluctuates greatly as the value increases, so we took BDBR as a priority factor and selected the smallest BDBR. When the value of T_h 2 is 33000, we achieved a lowest BDBR, so we choose this point as the value of T_h 2.

3.3. Two Candidate Partitions Are Selected Based on Canny Operator. In the image processing process, the edge features are extracted by Roberts, Sobel, and Canny operator in most cases, and their extraction effects on the same image under the same conditions are shown in Figure 3. Figure 3 reflects the conclusion that the effect of edge extraction by Canny operator is significantly better than the other two methods. Therefore, the Canny operator is selected as the core algorithm for this step. First, the edge features of the current CU are extracted by the Canny operator. Then, we rely on the extracted edge feature values to make judgments and select a more representative partition direction. Finally, take Horizontal Multitype Tree (MTT_H) or Vertical Multitype Tree (MTT_V) as a candidate mode to enter the RDO process.

In the traditional Pixel-based Canny algorithm, the setting of Double-Threshold is very important, which directly affects the effect of the algorithm. The work of the two thresholds in the edge detection algorithm of the Canny operator is clear and different. The high threshold is used to distinguish the contour of the target object from the background, and the low threshold is used to smooth the contour of the edge. However, the high and low thresholds of this algorithm need to be obtained by calculating gradient values of each pixel, which brings huge computational complexity.

The processing of the traditional Canny operator is shown as follows:

Table 1: Encoding performance with different T_h *1*.

T_h 1	T_h 2	TS	BDBR
1.5	25000	50.5	0.87
1.7	25000	50.3	0.86
1.9	25000	50.1	0.81
2.1	25000	50.2	0.82
2.3	25000	49.7	0.77
2.5	25000	50.1	0.74
2.7	25000	49.3	0.67
2.9	25000	48.8	0.69
3.1	25000	48.6	0.63
3.3	25000	48.3	0.65
3.5	25000	48.7	0.62
3.7	25000	48.4	0.63
3.9	25000	48.7	0.61
4.1	25000	48.5	0.63
4.3	25000	48.7	0.61
4.5	25000	48.3	0.59
4.7	25000	48.4	0.61

Table 2: Encoding performance with different T_h *2*.

T_h 1	T_h 2	TS	BDBR
3.3	15000	48.4	0.71
3.3	17000	48.6	0.69
3.3	19000	48.5	0.73
3.3	21000	48.7	0.67
3.3	23000	48.5	0.69
3.3	25000	48.6	0.66
3.3	27000	48.5	0.74
3.3	29000	48.6	0.69
3.3	31000	48.4	0.76
3.3	33000	48.1	0.65
3.3	35000	48.2	0.68
3.3	37000	48.5	0.74
3.3	39000	48.2	0.71
3.3	41000	48.4	0.67
3.3	43000	48.5	0.75
3.3	45000	48.4	0.66
3.3	47000	48.3	0.74

(1) Gaussian filter is used to remove noise and smooth the input frame
(2) Calculate the gradient value and direction of all pixels
(3) Use nonmaxima suppression process
(4) Calculate the high and low thresholds based on the histogram of the gradient magnitude
(5) Edge detection is accomplished by suppressing weak edges isolated

If the threshold is set in advance, the adaptability of the algorithm will be reduced, which means different degrees of distortion will be caused. As shown in Figure 4, the ranges of the Double-threshold in Figure 4(a) are 0.06–0.08 and 0.06–0.1, respectively. The image with a larger value of high threshold has fewer tiny and scattered edges in Figure 4(a). This means that while the low threshold is unchanged, the high threshold with a larger value can distinguish the target object from the background more accurately, which helps to reduce unnecessary disturbance. The ranges of Double-Threshold in Figure 4(b) are 0.06–0.2 and 0.16–0.2, respectively. Clearly, under the same contrast, a smaller value of low threshold has a better smoothing effect on the broken edge, which means that the edges of the target object are more continuous. In addition, the range between the low and high thresholds is also an important factor affecting the quality of the extracted edges. One false step will make a great difference, it is impossible to find an optimal threshold and the range of optimal threshold by human intervention for each of encoding objects. More than this, the non-maximum suppression process not only requires a lot of calculations, but also causes some contours to break or lose. These drawbacks do not contribute to the coding system. Therefore, how to reduce the computational complexity of the Canny operator while maintaining the edge quality is an urgent problem to be solved.

The block-based Canny operator is used to solve this problem. In this algorithm, the nonmaximum suppression process in the traditional algorithm is replaced by the isolation suppression step. At the same time, the process of setting low and high thresholds is also eliminated. The algorithm steps are as follows:

(1) Gaussian filter is used to remove noise and smooth the input frame
(2) Calculate the gradient value and direction of all pixels
(3) Each contour pixel whose gray scale is greater more 20 in current area is set to 0
(4) Each contour pixel without neighbor contours in current area is set to 0

This method reduces the noise disturbance caused by the nonmaximum suppression process. More importantly, as a benefit from the elimination of the nonmaximum suppression process and the process of calculating Double-Threshold, the computational complexity is reduced by 20% compared with the traditional algorithm, which is essential for reducing coding complexity and coding saving time.

The edge map is extracted according to the block-based Canny operator. H_E and V_E are calculated as follows:

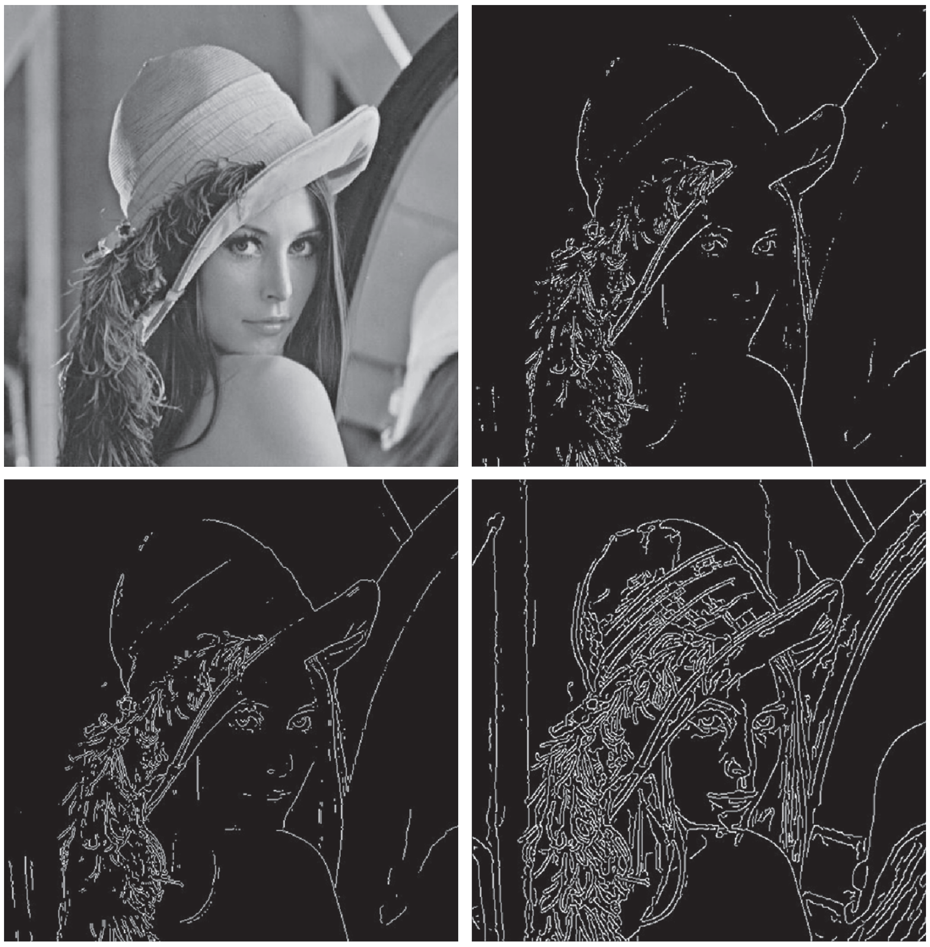


Figure 3: The edge images are extracted by different operators.

$$\begin{cases} ve_i = \sum_x \text{can}(x,y)|_{y=i}, & (x = 0,1,2,3,\ldots,h-1), \\ V_E = \max(ve_i) - \min(ve_j), & (i,j = 0,1,2,3,\ldots,w-1), \end{cases} \tag{7}$$

$$\begin{cases} he_i = \sum_x \text{can}(x,y)|_{x=i}, & (y = 0,1,2,3,\ldots,w-1), \\ H_E = \max(he_i) - \min(he_j), & (i,j = 0,1,2,3,\ldots,h-1), \end{cases} \tag{8}$$

where $\text{can}(x,y)$ means the pixel value of the current coordinate in edge map. If the pixel value of the current coordinate is equal to 0, then $\text{can}(x,y)$ is recorded as 0; otherwise, $\text{can}(x,y)$ is recorded as 1. We divide H_E by V_E; if H_E/V_E is greater than 1, MTT_H will be used as the candidate of partition mode; otherwise, MTT_V will be determined as the candidate.

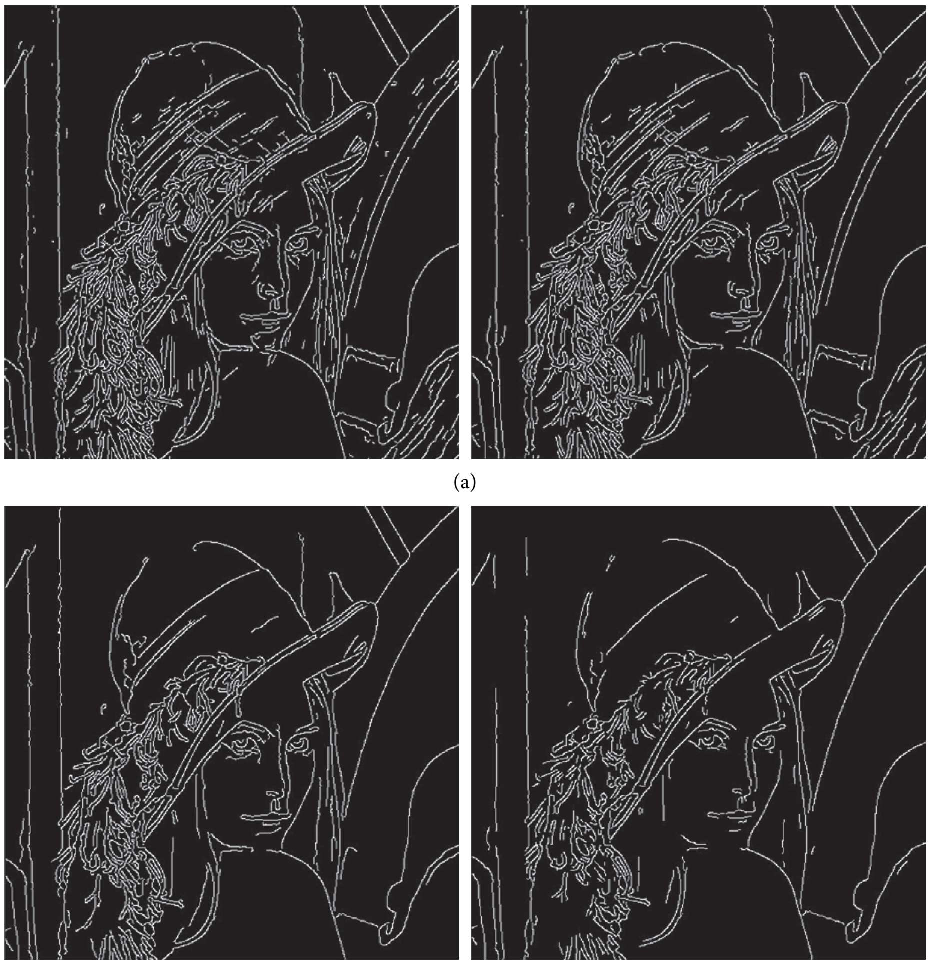
4. Experimental Results

In this paper, the video sequences B, B1 C, D, E, and E1 are used as test sequences. The proposed fast algorithm is tested under the all-intra-conditions in the official reference software VTM7.0 [35]. In VVC, Average Time Saving (ATS) and BDBR are used as indicators for evaluating the performance of the proposed algorithm. ATS reflects the level of coding complexity. If ATS increase is greater, the more time is saved. BDBR is used to reflect the overall quality of encoding. If the BDBR increases less, it means that the coding quality is better. ATS is defined as follows:

$$\text{ATS}(\%) = \frac{\text{Time}_{\text{VTM7.0}} - \text{Time}_{\text{Proposed}}}{\text{Time}_{\text{VTM7.0}}}, \tag{9}$$

where $\text{Time}_{\text{VTM7.0}}$ means the encoding time under the anchor method in VTM7.0 and $\text{Time}_{\text{Proposed}}$ means the encoding time obtained by our algorithm.

The performance of our algorithm is shown in Table 3. We can see that our proposed algorithm saves encoding time about 51.23% compared with the anchor method, while


(a)

(b)

Figure 4: The effect of high or low thresholds on edge images. (a) The edge images with different high threshold. (b) The edge images with different low threshold.

Table 3: The encoding performance of the proposed method.

Class	Sequence	Proposed	
		BDBR (%)	ATS (%)
Class B 1920 × 1080	Kimono	1.77	61.74
	ParkScene	1.24	52.84
	Catus	1.63	53.31
	BQTerrace	0.87	48.42
	BasketballDrive	2.09	56.94
Class C 832 × 480	PartyScene	0.29	41.37
	RaceHorsesC	0.76	52.42
	BasketballDrill	1.67	51.05
	BQMall	1.62	51.43
Class D 416 × 240	BQSquare	0.35	35.24
	RaceHorses	0.68	45.15
	BlowingBubbles	0.49	44.92
Class E 1280 × 720	KristenAndSara	2.13	57.09
	FourPeople	2.42	56.89
	Johnny	3.16	59.62
Average		1.41	51.23

BDBR only increases by 1.41%. It should be noted that the experimental results might have some differences for different video sequences because of the difference in resolution under the same experimental conditions. But these differences are not enough to affect our judgment on the experimental results, which is acceptable. Based on the results of test, the proposed fast CU partition algorithm has achieved good encoding performance.

Figure 5 indicates the RD property of the proposed algorithm compared with anchor method for four test videos with different resolution, including "*BQTerrace*," "*KristenAndSara*," "*PartScene*," and "*BlowingBubbles*." Compared to anchor method, the proposed method has almost consistent RD property.

Table 4 shows the performance results of the proposed fast algorithm in this paper compared with other fast algorithms. In order to evaluate the performance of the proposed algorithm, we have compared with three fast CU partition algorithms, namely, CTTD [36], FCPD [7], and FPDS [8].

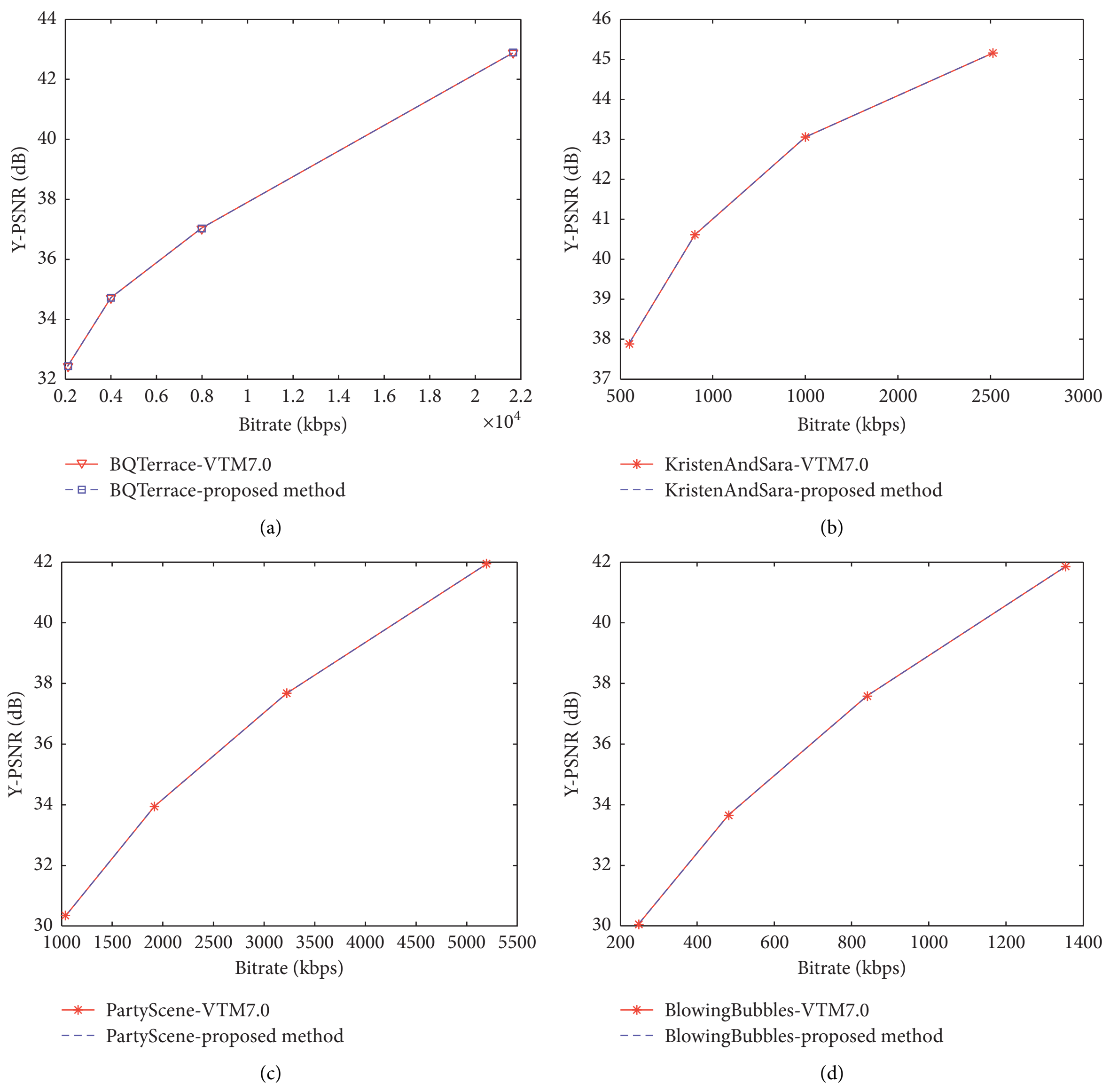


FIGURE 5: The RD property of the proposed algorithm. (a) R-D of "*BQTerrace*," (b) R-D of "*KristenAndSara*," (c) R-D of "*PartScene*," and (d) R-D of "*BlowingBubbles*."

TABLE 4: The results of the proposed method compared with other fast methods.

Class	Test sequence	Proposed (VTM7.0)		CTTD [36] (VTM4.0)		FCPD [7] (VTM4.0)		FPDS [8] (VTM7.0)	
		BDBR	*ATS*	*BDBR*	*ATS*	*BDBR*	*ATS*	*BDBR*	*ATS*
Class B1 1920 × 1080	Kimono	1.77	61.74	1.05	34.33	0.52	35.05	1.93	59.51
	ParkScene	1.24	52.84	1.11	34.56	0.63	43.29	1.26	51.84
	BQTerrace	0.87	48.42	1.00	31.07	0.81	35.29	1.08	45.30
Class C 832 × 480	PartyScene	0.29	41.37	0.76	35.93	0.34	33.15	0.26	38.62
	RaceHorsesC	0.76	52.42	0.82	36.06	0.65	30.53	0.88	49.05
	BasketballDrill	1.67	51.05	1.67	33.94	1.30	29.61	1.82	48.48
Class D 416 × 240	BQSquare	0.35	35.24	0.61	33.98	0.22	26.02	0.19	31.95
	RaceHorses	0.68	45.15	0.95	35.96	0.30	26.21	0.54	41.69
	BlowingBubbles	0.49	44.92	0.74	35.04	0.23	27.71	0.47	40.35

Table 4: Continued.

Class	Test sequence	Proposed (VTM7.0)		CTTD [36] (VTM4.0)		FCPD [7] (VTM4.0)		FPDS [8] (VTM7.0)	
		BDBR	*ATS*	*BDBR*	*ATS*	*BDBR*	*ATS*	*BDBR*	*ATS*
Class E 1280 × 720	KristenAndSara	2.13	57.09	1.19	32.95	0.94	40.61	2.78	55.11
	FourPeople	2.42	56.89	1.38	35.05	1.18	42.29	2.70	57.57
	Johnny	3.16	59.62	1.44	33.02	1.29	40.72	3.22	56.88
Average		1.31	50.56	1.06	34.32	0.70	34.20	1.43	48.03

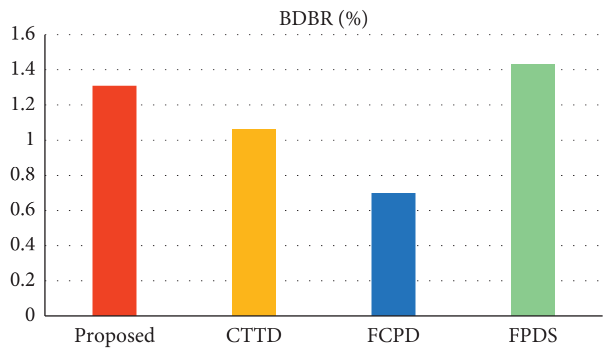


Figure 6: The BDBR increase of different methods.

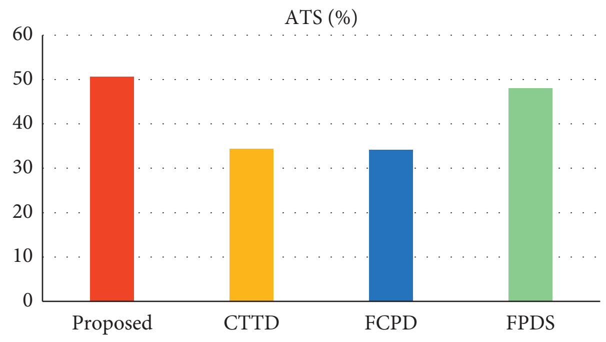


Figure 7: The average time saving of different methods.

Table 5: The results of the proposed method compared with CNN fast methods.

Class	Sequence	Proposed		CNN [24]	
		BDBR (%)	ATS (%)	BDBR (%)	ATS (%)
Class B1 1920 × 1080	Kimono	1.77	61.74	0.87	33.32
	ParkScene	1.24	52.84	0.83	35.41
	BQTerrace	0.87	48.42	0.95	34.50
Class C 832 × 480	PartyScene	0.29	41.37	0.55	31.10
	RaceHorsesC	0.76	52.42	0.37	23.63
	BasketballDrill	1.67	51.05	1.30	33.39
Class D 416 × 240	BQSquare	0.35	35.24	0.68	30.73
	RaceHorses	0.68	45.15	0.71	31.79
	BlowingBubbles	0.49	44.92	0.95	33.90
Class E1 1280 × 720	KristenAndSara	2.13	57.09	1.61	34.84
	FourPeople	2.42	56.89	1.38	38.01
Average		1.15	49.74	0.93	32.78

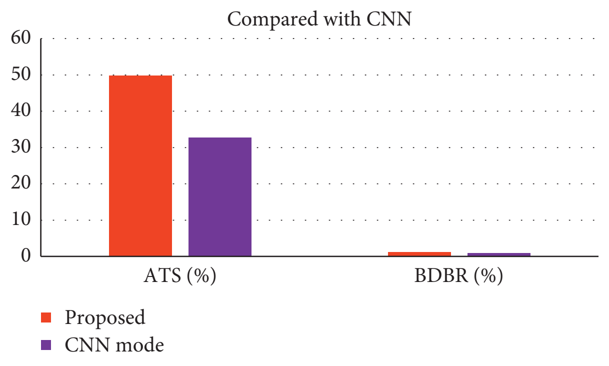


Figure 8: The average time saving and BDBR increase compared to CNN.

Relative to the anchor method, as can be seen from the table, FCPD has achieved the best encoding quality, the average BDBR of FCPD increased by 0.70%, but the ATS is the smallest compared to other methods, only 34.20%. The BDBR of the CTTD and FPDS have increased by 1.06% and 1.43%, respectively. Compared with the proposed algorithm, the BDBR has increased by −0.25% and 0.12%, respectively. It means that the encoding quality of the two methods is close to the proposed algorithm. However, compared with CTTD and FPDS, the ATS of the proposed algorithm is 16.24% and 2.53%, respectively.

The performance comparison between the proposed method and the other traditional fast methods is visually shown in Figures 6 and 7. Figure 6 shows the BDBR of the proposed method compared with CTTD, FCPD, and FPDS. It can be seen that the proposed algorithm achieves reasonable coding quality. In addition, Figure 7 shows that the proposed scheme can achieve the best ATS compared with CTTD, FCPD, and FPDS. The BDBR difference between this method and CTTD and FPDS is negligible, but it can reduce a lot of coding time. In summary, the proposed method has the best coding performance.

Table 5 shows the performance comparison between the proposed algorithm and the fast algorithm based on CNN. It can be seen that the proposed algorithm saves 16.96% of the ATS compared to the CNN algorithm, while BDBR only increases by 0.22%.

The performance comparison between the proposed algorithm and the CNN algorithm is shown in Figure 8. It can be seen that, compared to the latest CNN algorithm, the proposed algorithm achieves a BDBR close to CNN, and the difference between the two methods is negligible. The proposed method achieves ATS much higher than the CNN algorithm, which reflects better coding performance.

5. Conclusion

A fast texture-based CU partition decision algorithm is proposed in this paper. We have designed a partition decision method from large size to small size, which solves the problem of high computational complexity of VVC in the CU partition process. First, the texture complexity of the neighboring CU is used as a threshold to evaluate the complexity of the current CU, and the nonpromising depth level is skipped according to the comparison result. Then, the characteristics are extracted by Sobel operator to judge the symmetry of the texture, so as to terminate the MTT partition early. Finally, the main direction of the CU texture is extracted by the block-based Canny algorithm, which cancels the nonmaximum suppression and threshold selection process to speed up the MTT partition direction decision. The simulation results show that this method saves the coding time 50.56%, while BDBR only increases by 1.31%. The proposed algorithm achieves an effective compromise between coding efficiency and coding quality, which reflects better performance than other methods.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Nos. 61771432, 61773018 and 61302118), the Basic Research Projects of Education Department of Henan (Nos. 21zx003 and 20A880004), and the Key Research and Development Program of Henan (No. 202102210179).

References

[1] Y.-W. Huang, C.-W. Hsu, C.-Y. Chen et al., “A VVC proposal with quaternary tree plus binary-ternary tree coding block structure and advanced coding techniques,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 5, pp. 1311–1325, 2020.

[2] J. Lee, S. Kim, K. Lim, and S. Lee, “A fast CU size decision algorithm for HEVC,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 411–421, 2015.

[3] 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*, vol. 30, no. 6, pp. 1668–1682, 2020.

[4] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard, “Complexity reduction opportunities in the future VVC intra encoder,” in *Proceedings of the 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, Kuala Lumpur, Malaysia, September 2019.

[5] T. Fu, H. Zhang, F. Mu, and H. Chen, “Fast cu partitioning algorithm for H.266/VVC intra-frame coding,” in *Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 55–60, Shanghai, China, July 2019.

[6] S. De-Luxán-Hernández, V. George, J. Ma et al., “An intra subpartition coding mode for VVC,” in *Proceedings of the*

2019 IEEE International Conference on Image Processing (ICIP), pp. 1203–1207, Taipei, Taiwan, September 2019.

[7] N. Tang, "Fast CTU partition decision algorithm for VVC intra and inter coding," in *Proceedings of the 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 361–364, Bangkok, Thailand, November 2019.

[8] Y. Fan, J. A. Chen, H. Sun, J. Katto, and M. E. Jing, "A fast QTMT partition decision strategy for VVC intra prediction," *IEEE Access*, vol. 8, pp. 107900–107911, 2020.

[9] W. Jiang, H. Ma, and Y. Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," in *Proceedings of the 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 1836–1840, Yichang, China, April 2012.

[10] X. Dong, L. Shen, M. Yu, and H. Yang, "Fast intra mode decision algorithm for versatile video coding," *IEEE Transactions on Multimedia*, vol. 1, p. 1, 2021.

[11] B. Min and R. C. C. Cheung, "A fast CU size decision algorithm for the HEVC intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 892–896, 2015.

[12] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An effective CU size decision method for HEVC encoders," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 465–470, 2013.

[13] D. Ruiz, G. Fernández-Escribano, J. L. Martínez, and P. Cuenca, "Fast intra mode decision algorithm based on texture orientation detection in HEVC," *Signal Processing: Image Communication*, vol. 44, pp. 12–28, 2016.

[14] L. Shen, Z. Zhang, and Z. Liu, "Effective CU size decision for HEVC intracoding," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4232–4241, 2014.

[15] L.-L. Wang and W.-C. Siu, "Novel adaptive algorithm for intra prediction with compromised modes skipping and signaling processes in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 10, pp. 1686–1694, 2013.

[16] A. Lee, J. DongSan, K. Jongho et al., "An efficient inter prediction mode decision method for fast motion estimation in HEVC," in *Proceedings of the 2013 International Conference on ICT Convergence (ICTC)*, pp. 502–505, Jeju Island, South Korea, October 2013.

[17] X. Sun, X. Chen, Y. Xu, Y. Xiao, Y. Wang, and D. Yu, "Fast CU size and prediction mode decision algorithm for HEVC based on direction variance," *Journal of Real-Time Image Processing*, vol. 16, no. 5, pp. 1731–1744, 2019.

[18] L. Shen, Z. Zhang, and Z. Liu, "Adaptive inter-mode decision for HEVC jointly utilizing inter-level and spatiotemporal correlations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 10, pp. 1709–1722, 2014.

[19] Z. Yang, Q. Shao, and S. Guo, "Fast coding algorithm for HEVC based on video contents," *IET Image Process*, vol. 11, no. 6, pp. 343–351, 2017.

[20] X. Wang and Y. Xue, "Fast HEVC intra coding algorithm based on Otsu's method and gradient," in *Proceedings of the 2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–5, Nara, Japan, June 2016.

[21] J. Cui, T. Zhang, C. Gu, X. Zhang, and S. Ma, "Gradient-based early termination of CU partition in VVC intra coding," in *Proceedings of the 2020 Data Compression Conference (DCC)*, pp. 103–112, Snowbird, UT, USA, March 2020.

[22] Z. Wang, S. Wang, X. Zhang, S. Wang, and S. Ma, "Fast QTBT partitioning decision for interframe coding with convolution neural network," in *Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 2550–2554, Athens, Greece, October 2018.

[23] Y. Zhang, G. Wang, R. Tian, M. Xu, and C. C. J. Kuo, "Texture-classification accelerated CNN scheme for fast intra CU partition in HEVC," in *Proceedings of the 2019 Data Compression Conference (DCC)*, pp. 241–249, Snowbird, UT, USA, November 2019.

[24] G. Tang, M. Jing, X. Zeng, and Y. Fan, "Adaptive CU split decision with pooling-variable CNN for VVC intra encoding," in *Proceedings of the 2019 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, Sydney, Australia, December 2019.

[25] Z. Liu, X. Yu, S. Chen, and D. Wang, "CNN oriented fast HEVC intra CU mode decision," in *Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2270–2273, Montreal, Canada, May 2016.

[26] K. Kim and W. W. Ro, "Fast CU depth decision for HEVC using neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1462–1473, 2019.

[27] J. F. de Oliveira and M. S. Alencar, "Online learning early skip decision method for the HEVC Inter process using the SVM-based Pegasos algorithm," *Electron.Lett.*vol. 52, no. 14, pp. 1227–1229, 2016.

[28] H. Kim and R. Park, "Fast CU partitioning algorithm for HEVC using an online-learning-based bayesian decision rule," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 130–138, 2016.

[29] J. Chen and L. Yu, "Effective HEVC intra coding unit size decision based on online progressive Bayesian classification," in *Proceedings of the 2016 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, Seattle, WA, USA, July 2016.

[30] Q. Hu, X. Zhang, Z. Shi, and Z. Gao, "Neyman-pearson-based early mode decision for HEVC encoding," *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 379–391, 2016.

[31] B. Du, W. Siu, and X. Yang, "Fast CU partition strategy for HEVC intra-frame coding using learning approach via random forests," in *Proceedings of the 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1085–1090, Hong Kong, China, December 2015.

[32] D. Ruiz-Coll, V. Adzic, G. Fernández-Escribano, H. Kalva, J. L. Martínez, and P. Cuenca, "Fast partitioning algorithm for HEVC Intra frame coding using machine learning," in *Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP)*, pp. 4112–4116, Paris, France, January 2014.

[33] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, and L. Xu, "Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2225–2238, 2015.

[34] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and multi-class learning based low complexity optimization for HEVC encoding," *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 547–561, 2017.

[35] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/master/cfg.

[36] S. Park and J. Kang, "Context-based ternary tree decision method in versatile video coding for fast intra coding," *IEEE Access*, vol. 7, pp. 172597–172605, 2019.