applied sciences

MDPI

Article

A Fast Algorithm for VVC Intra Coding Based on the Most Probable Partition Pattern List

Haiwu Zhao, Shuai Zhao, Xiwu Shang * and Guozhong Wang

School of Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; zhao.hw@avsgm.com (H.Z.)
* Correspondence: dxsxw@126.com; Tel.: +86-021-6779-1084

Abstract: Compared with High-Efficiency Video Coding (HEVC), Versatile Video Coding (VVC) has more flexible division and higher compression efficiency, but it also has higher computational complexity. In order to reduce the coding complexity, a fast algorithm based on the most probable partition pattern list (MPPPL)and pixel content similarity is proposed. Firstly, the MPPPL is constructed by using the average texture complexity difference of the sub-coding unit under different partition modes. Then, the sub-block pixel mean difference is used to decide the best partition mode or shorten the MPPPL. Finally, the selection rules of the reference lines in the intra prediction process are counted and the unnecessary reference lines are skipped by using the pixel content similarity. The experimental results show that compared with VTM-13.0, the proposed algorithm can save 52.26% of the encoding time, and the BDBR (Bjontegarrd delta bit rate) only increases by 1.23%.

Keywords: versatile video coding; fast algorithm; coding unit partition; intra prediction

Citation: Zhao, H.; Zhao, S.; Shang, X.; Wang, G. A Fast Algorithm for VVC Intra Coding Based on the Most Probable Partition Pattern List. *Appl. Sci.* **2023**, *13*, 10381. https://doi.org/10.3390/app131810381

Academic Editor: Christos Bouras

Received: 26 June 2023
Revised: 16 August 2023
Accepted: 6 September 2023
Published: 17 September 2023

1. Introduction

Rapid advancements in information technology have led to the rapid development of multimedia technologies such as 4K ultra-high-definition videos, 360-degree immersive multimedia, and high dynamic range videos. Consequently, there has been a significant surge in data volume, placing immense pressure on data storage and transmission. Existing coding standards such as High-Efficiency Video Coding [1] have difficulty in meeting this requirement.

To address these challenges, the Joint Video Coding Expert Group embarked on the development of the next-generation video standard and introduced a novel video coding standard known as Versatile Video Coding (VVC) [2]. The primary objective of VVC is to achieve a compression efficiency improvement of over 50% while maintaining comparable video quality to HEVC. VVC incorporates a new segmentation technique called Quadtree with Nested Multi-Type Tree (QTMT) [3]. In addition to the QT partition structure, VVC encompasses four other multi-type tree (MTT) partition structures: vertical binary tree (VBT), horizontal binary tree (HBT), vertical ternary tree (VTT), and horizontal ternary tree (HTT). By introducing QTMT, a coding unit's (CU) shape becomes more flexible and diverse, thereby increasing the number of CUs that necessitate recursive traversal during the rate-distortion optimization (RDO) process. Figure 1 illustrates an example of the partition structure obtained after recursive traversal, along with the corresponding tree structure. Flexible partition structures, while enhancing compression efficiency, also increase computational complexity. Under the condition of an all-intra configuration, the average complexity of VVC is 18 times higher than that of HEVC [4]. This has hindered the widespread adoption of VVC (Versatile Video Coding) and its application in real-time scenarios.

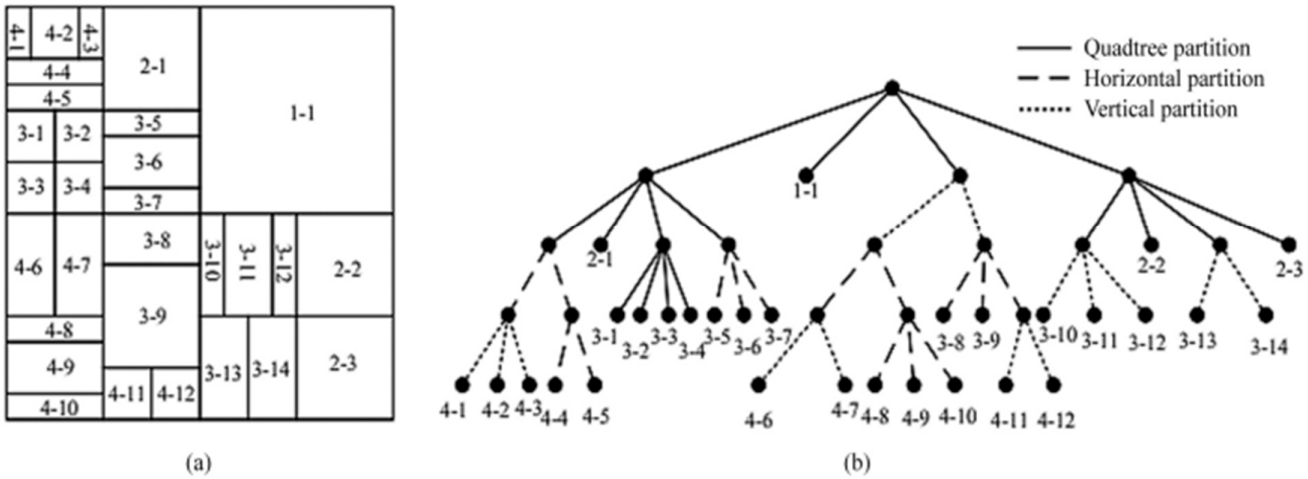


Figure 1. Example of the CU partition structure in VVC:(**a**) CU division results, ad (**b**) corresponding tree structure.

To solve the problem of the high computational complexity in the intra coding of VVC, a fast partition method is proposed. The main contributions of the proposed method are as follows:

(1) The concept of an MPPPL is proposed, and the method for constructing the MPPPL is provided. In the decision-making process of the partition mode, the partition mode in the MPPPL is sequentially tested, and the optimal partition mode is decided in advance according to the calculation results.
(2) According to the influence of the partition mode distribution in the MPPPL on the partition mode selection, the pixel mean value difference is used to shorten the MPPPL or to decide the optimal partition mode in advance.
(3) Using the pixel content similarity, the calculation of reference line three is skipped in the intra prediction process, which reduces the computational complexity of the intra prediction.

2. Related Work

Currently, studies on fast algorithms for VVC can be divided into two parts: artificial intelligence-based methods and coding content-based methods.

For artificial intelligence-based methods, in [5], a fast algorithm based on machine learning was proposed, and it used texture complexity to determine the division direction and a lightweight neural network to determine the division mode. Fu et al. [6] proposed a fast decision binary partition algorithm based on Bayesian rules by utilizing the correlation between the current CU and the sub-CU after horizontal binary partition, and it used the correlation between the current CU and the sub-CU after horizontal binary segmentation to terminate the vertical binary segmentation. In [7], a deep learning method was proposed to predict the CU partition, and a multi-stage exit CNN model was also proposed for fast CU partition in multiple stages while using an early exit mechanism to skip redundant CUs. Zhang et al. [8] divided a CTU into four parts to train the neural network and to calculate the probability of various partition modes in the CTU. Once the probability of the partition mode exceeded the threshold, it was skipped to achieve complexity reduction. In [9], the proposed algorithm was formed by five binary Light Gradient Boosting Machine (LightGBM) classifiers. In [10], a random forest classifier was used to make an early decision about the partition mode by utilizing the texture features of the image and the intra-frame prediction process was optimized based on the image content. Yang et al. [11] transformed the QTMT partition process into multiple binary classification problems for each decision layer, which were processed by a decision tree classifier. Although this method saved coding time, the BDBR (Bjøntegaard Delta Bit Rate) losses were high. In [12], a hierarchical

grid fully convolutional network (HG-FCN) framework was proposed to predict a specific level of partition structure, achieving a certain degree of complexity reduction. Saldanha et al. [13] used a classifier to make early decisions about the DC mode and the Planar mode during the prediction process, and they used the pixel variances of the sub-blocks to decide whether to use intra-frame sub-block partition technology. In [14], a convolutional neural network was trained to predict partition modes by training a probability vector. In [15], a neural network model was trained using cross-entropy, and it was used to terminate the partitioning process early. In [16], a fast CU-partitioning decision algorithm based on texture complexity and convolutional neural networks (CNN) was used, and this algorithm used symmetric convolution kernels to extract features and redesign the loss function. In [17], a novel feature based on Statistical Oriented Gradient (SOG) was proposed to extract the feature information of a coding block, and it used SOG to speed up the intra prediction mode decision process. In [18], a fast QTMT partition algorithm based on a CNN-binary tree horizontal (CNN-BTH) network was developed to predict the BTH mode decision at 32 × 32 coding units (CUs). The BTV decision tree algorithm was also predicted at this level by a CNN-binary tree vertical (CNN-BTV).

For coding content-based methods, in [19], a latitude-based preprocessing was introduced for the early termination of the coding unit (CU) partition in the polar region. In [20], the factor of an average background luminance for just-noticeable-distortion was applied to identify the visually distinguishable (VD) pixels within a coding unit (CU). Zhang et al. [21] proposed a fast partition scheme based on adjacent sub-regions, and it skipped unnecessary partition modes in advance based on the similarity of the adjacent sub-regions in the horizontal and vertical directions. In [22], a CU partition was determined early by the use of texture information, and the residual coefficient distribution of the CUs were used to skip unnecessary partition modes. In [23], Li et al. used the gradient of the Scharr operator to describe texture information, and they used the edge differences of the sub-blocks to describe structural information. On this basis, they proposed a fast algorithm based on texture features. In [24], Shang et al. created rapid decisions in the CU partition process by utilizing the partition mode and the size distribution of adjacent CUs, and they optimized the decision-making process of inter-frame prediction modes. In [25], Zhang et al. used corner features and average color brightness differences to classify screen content. Then, for different screen contents, they exploited different strategies to predict the coding modes. In [26], Fan et al. used the Sobel operator to calculate the gradient of the CU and terminate the QT partition based on the gradient. Then, texture information was used to measure the differences between the partition structures based on which partition patterns were determined. Shang et al. [27] predicted the quadtree division depth of a current CU in advance by analyzing the correlations between adjacent CUs. In addition, image texture features were utilized to make early decisions about the MTT division process. Zhang et al. [21] proposed a fast partition scheme based on adjacent sub-regions, and it skipped unnecessary partition modes in advance based on the similarity between adjacent sub-regions in the horizontal and vertical directions. In [5], Zhang et al. determined whether to split decisions using CU texture information, and they skipped unnecessary partition modes according to the distribution of the residual coefficients.

Although the above methods have achieved some complexity reductions, they have not achieved a good balance between the complexity reduction and the compression performance loss. Complexity reductions that maintain good compression performance are limited, and the improvement in the application of VVC in real-time scenarios is not significant. A compression performance with a higher complexity reduction has a greater loss, and this cannot meet coding requirements.

3. Proposed Method

3.1. Principle

Figure 2 shows schematic diagrams of the CU partition results of the QTMT structure of the coding sequence in VVC. Upon examining Figure 2, it becomes evident that as the

partition depth increases, the CU progressively becomes smoother, resulting in a gradual reduction in texture complexity. Additionally, the partition mode decision tends to favor partition modes that minimize the texture complexity of the sub-CUs.

(**a**) (**b**)

Figure 2. The partition results under QP = 32: (**a**) BQSquare, and (**b**) BlowingBubbles.

We selected five video sequences with different resolutions and texture complexities under JVET general test conditions [24], including FoodMarket4, Kimono1, BasketballPass, BQMall, and BQSquare. The relationship between the decision of the CU partition mode in the coding process and the texture complexity of the sub-CUs after different coding sequences was counted. The experimental results are shown in Figure 3. The partition mode with the largest average texture complexity of the sub-CU was recorded as Modemax, and the partition mode with the smallest average texture complexity of the sub-CU was recorded as Modemin. Texture complexity was used to describe the local texture features of the image, and the more complex the texture of the local image, the higher the texture complexity.

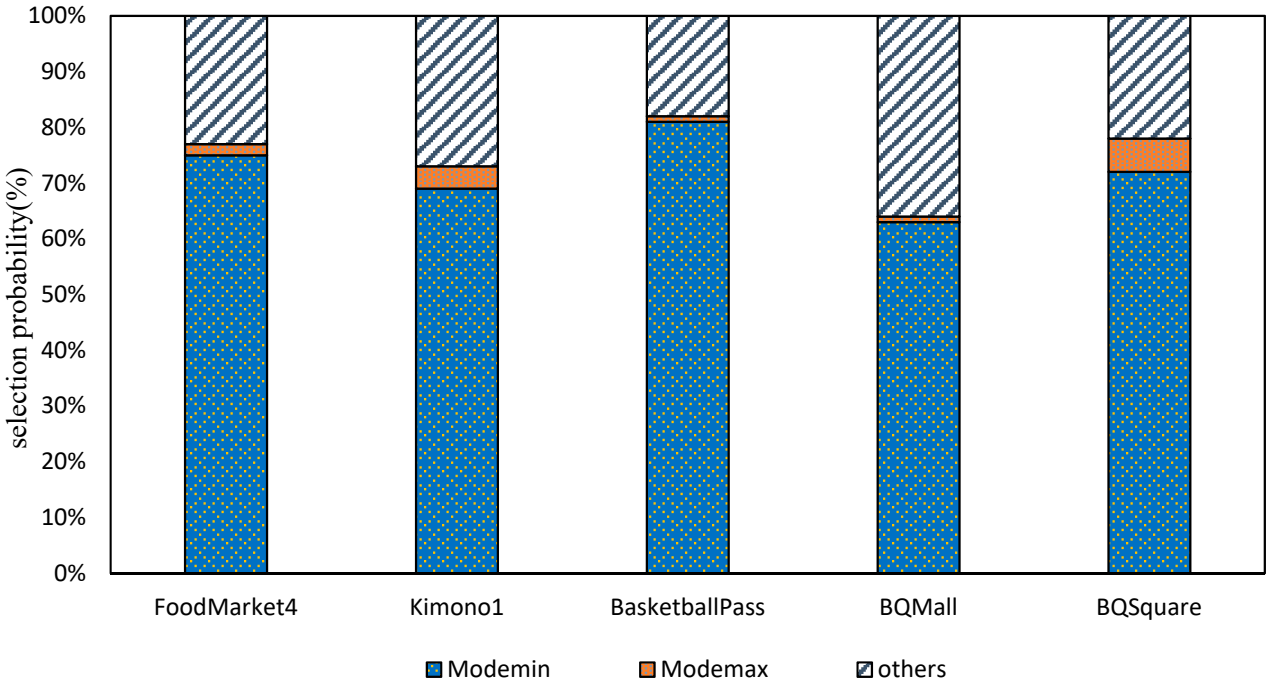


Figure 3. The relationship between the optimal partition mode of VVC and the average value of the sub-CU texture complexity.

Based on the statistical data in Figure 3, it is not difficult to find that the VVC partition mode decision has a strong correlation with the average texture complexity of the partitioned sub-CUs. The smaller the average texture complexity of the divided sub-CUs, the greater the possibility of the partition mode being selected as the optimal partition mode, accounting for an average of 75%. The average texture complexity of the divided sub-CUs reflects the probability of the current partition mode being selected as the optimal partitioning mode, but it cannot be used as the basis for the final partition mode decision because there are still some partition modes that are selected as the optimal partition mode, even though their sub-CU average texture complexity is not the lowest.

3.2. Construct the Most Probable Partition Pattern List

As shown in Figure 4, there are five partition modes in the VVC intra-frame coding partition process, namely, Quad-tree (QT), vertical binary tree (VBT), horizontal binary tree (HBT), vertical ternary tree (VTT), and horizontal ternary tree (HTT), where HBT and VBT are collectively referred to as binary tree (BT) and VTT and HTT are collectively referred to as ternary tree (TT).

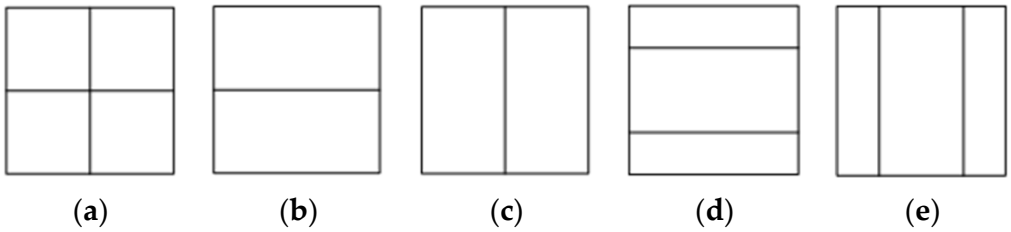


Figure 4. VVC partition structures: (**a**) QT, (**b**) HBT, (**c**) VBT, (**d**) HTT, and (**e**) VTT.

The most probable partition pattern list (MPPPL) initially included these five partition modes. We calculated the average texture complexity of sub-CUs under these five partitioning modes. The formula used for calculating the texture complexity is as follows:

$$fT\ (p,q,m,n) = \sqrt{\frac{1}{m \times n}\sum_{i=p}^{p+m}\sum_{j=q}^{q+n}[pix(i,j) - p_{ave}(p,q,m,n)]^2}, \quad (1)$$

where p and q represent the starting coordinates of the CU, m and n represent the end coordinates of the CU, and $p_{ave}(p, q, m, n)$ represents the CU whose starting coordinates are p and q, the end coordinates are represented by m, and n is the inner pixel average. $p_{ave}(p, q, m, n)$ is calculated as follows:

$$p_{ave}(p,q,m,n) = \frac{1}{m \times n}\sum_{i=p}^{p+m}\sum_{j=q}^{q+n} pix(i,j). \quad (2)$$

The average texture complexity of the sub-CUs after horizontal binary partition is denoted as T_{hbt}, and its calculation formula is shown below, where x and y are the starting coordinates of the current CU and w and h are the width and height of the current CU.

$$T_{hbt} = \frac{1}{2}[f_T(x,y,x+w,y+\frac{h}{2}) + f_T(x,y+\frac{h}{2},x+w,y+h)]. \quad (3)$$

The average value of the sub-CU texture complexity after horizontal ternary partition is denoted as T_{htt}, and its calculation formula is as follows:

$$T_{htt} = \frac{1}{3}[f_T(x,y,x+w,y+\frac{h}{4}) + f_T(x,y+\frac{h}{4},x+w,y+\frac{3h}{2}) + f_T(x,y+\frac{3h}{4},x+w,y+h)]. \quad (4)$$

The average value of the sub-CU texture complexity after vertical binary partition is recorded as T_{vbt}, and its calculation formula is as follows:

$$T_{vbt} = \frac{1}{2}[f_T(x, y, x + \frac{w}{2}, y + h) + f_T(x + \frac{w}{2}, y, x + w, y + h)]. \quad (5)$$

The average value of the sub-CU texture complexity after vertical ternary partition is recorded as T_{vht}, and its calculation formula is as follows:

$$T_{vtt} = \frac{1}{3}[f_T(x, y, x + \frac{w}{4}, y + h) + f_T(x + \frac{w}{4}, y + h, x + \frac{3w}{4}, y + h) + f_T(x + \frac{3w}{4}, y + h, x + w, y + h)]. \quad (6)$$

The average value of the sub-CU texture complexity after QT partition is recorded as T_{qt}, and its calculation formula is as follows:

$$T_{qt} = \frac{1}{4}[f_T(x, y, x + \frac{w}{2}, y + \frac{h}{2}) + f_T(x + \frac{w}{2}, y, x + w, y + \frac{h}{2}) + f_T(x, y + \frac{h}{2}, x + \frac{w}{2}, y + h) + f_T(x + \frac{w}{2}, y + \frac{h}{2}, x + w, y + h)]. \quad (7)$$

After obtaining the average texture complexity of the sub-CUs under the five partition modes, we used this as a reference to update the MPPPL. The larger the average texture complexity of the sub-CUs, the lower the probability of that partition mode being selected as the optimal partition mode. Therefore, we used the reciprocal of the average texture complexity of the sub-CUs after division to calculate the probability that the partition mode was the optimal partition mode. The formula used is as follows:

$$\mathrm{P}mode = \frac{\frac{1}{Tmode}}{\frac{1}{Tqt} + \frac{1}{Tvtt} + \frac{1}{Tvbt} + \frac{1}{Thtt} + \frac{1}{Thbt}}, \quad (8)$$

where *mode* represents one of the QT partitions, vertical ternary partitions, vertical binary partitions, horizontal ternary partitions, and horizontal binary partitions.

After obtaining the probability of each partition mode being selected as the optimal partition mode, the MPPPL was sorted from large to small according to the probability of the partition mode being selected as the optimal partition mode, and the MPPPL was updated. At this time, the MPPPL was expressed as follows:

$$\mathrm{MPPPL} = \{\mathrm{Mode-a}, \mathrm{Mode-b}, \mathrm{Mode-c}, \mathrm{Mode-d}, \mathrm{Mode-e}\}. \quad (9)$$

In addition to the five partition modes, VVC also has the possibility of adopting a non-partition strategy. Considering the spatial continuity of an image's content, we considered whether to include the non-partition mode in the MPPPL. First, we obtained the partition sizes of adjacent CUs. If the partition sizes of the adjacent CUs were all larger than the current CU, based on spatial correlation, the current CU had a possibility of not being partitioned. Further, we compared the difference value (d) between the average texture complexity of the sub-CUs corresponding to partition mode Mode-a (T_a) and the average texture complexity of the sub-CUs corresponding to partition mode Mode-e (T_e). The comparison formula used is as follows:

$$d = \frac{T_a}{T_e}, \quad (10)$$

where if the two values T_a and T_e are very close, then d will be close to 1. This indicates that the texture complexity of the current CU has already approached smoothness, and further partition will not effectively reduce the texture complexity of the CU. Based on practical experience, when d is less than 1.1 and greater than 0.9, we can include the non-partitioning strategy in the MPPPL and place it at the first position, and then the remaining partition modes are shifted back in order.

To avoid redundant partitioning, VVC has partitioning condition limitations. After the MPPPL construction was completed, we modified the MPPPL according to the limitations

of the partitioning rules, and we deleted the partitioning modes that were not allowed in the MPPPL.

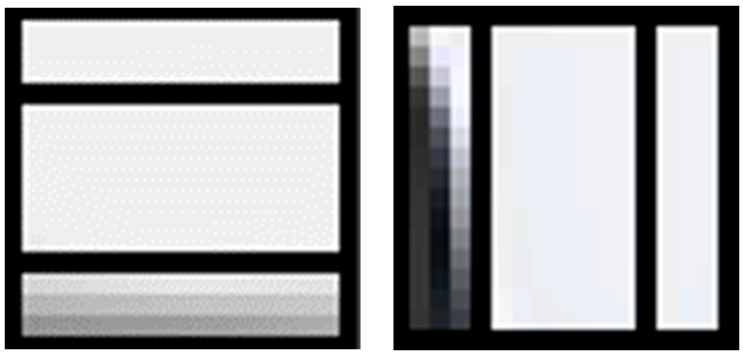
3.3. Partition Mode Decision

After the construction of the MPPPL was completed, the partition mode was decided in advance on this basis or the length of the MPPPL was reduced. We conducted a statistical analysis on the probability of a particular direction being selected as the final partition direction when the first two partition modes in the MPPPL were of the same direction, and the results are shown in Table 1. When both Mode-a and Mode-b in the MPPPL were either horizontal partition modes or vertical partition modes, it indicates that the current CU exhibits strong directional texture and is more inclined towards either horizontal or vertical partition. In such cases, the MPPPL can be modified by removing all partition modes except for Mode-a and Mode-b.

Table 1. Mode-a and Mode-b partition mode selection probability in the same direction.

Coding Sequence	Both Vertical	Both Horizontal
Campfire	85%	79%
Cactus	94%	83%
BasketballPass	83%	85%
BQSquare	86%	87%

In the partition process of VVC, the probability of a TT partition being adopted is very small (no more than 20%) [25]. Therefore, when determining the direction of a CU partition, further consideration is given as to whether to skip TT partition and decide the optimal partition mode. If Mode-a is a TT partition at this time, it indicates that in the previous calculation, the TT partition had a high probability of being the optimal partition mode, and the TT partition is not skipped at this time. If Mode-a is a BT partition, we would consider skipping the TT partition directly at this time. It is not difficult to determine from Figure 5 that when the texture structure in a CU is mainly concentrated on the left and right sides or at the upper and lower sides of a CU boundary, the TT division structure is used. At this time, there was a significant difference in the average pixel values between the sub-CUs, and so we used the pixel mean difference between the sub-CUs to decide whether to skip the TT partition.



(**a**) (**b**)

Figure 5. Schematic diagram of the TT division structure: (**a**) HTT, and (**b**) VTT.

The pixel mean difference of the horizontal binary partition (P_{HBT}) was calculated as follows:

$$P_{HBT} = p_{ave}(x, y, x + w, y + \frac{h}{2}) - p_{ave}(x, y + \frac{h}{2}, x + w, y + h). \quad (11)$$

The pixel mean difference of the horizontal ternary partition (P_{HTT}) was calculated as follows:

$$P_{HTT} = p_{ave}(x, y + \frac{h}{4}, x + w, y + \frac{3h}{4}) - min(p_{ave}(x, y, x + w, y + \frac{h}{4}), p_{ave}(x, y + \frac{3h}{4}, x + w, y + h)). \quad (12)$$

Similarly, the formulas used for calculating the pixel mean difference of the vertical binary partition (P_{VBT}) and the pixel mean difference of the vertical ternary partition (P_{VTT}) are as follows:

$$P_{VBT} = p_{ave}(x, y, x + \frac{w}{2}, y + h) - p_{ave}(x + \frac{w}{2}, y, x + w, y + h) \text{ and} \quad (13)$$

$$P_{VTT} = p_{ave}(x + \frac{w}{4}, y, x + \frac{3w}{4}, y + h) - min(p_{ave}(x, y, x + \frac{w}{4}, y), p_{ave}(x + \frac{3w}{4}, y, x + w, y + h)). \quad (14)$$

We selected different coding sequences and conducted a statistical analysis on the relationship between the selection of partition modes in the MTT partition process and the difference in the pixel mean values. The statistical results are shown in Table 2.

Table 2. Effect of pixel mean difference on the selection of the division mode.

Coding Sequence	BT (%)	TT (%)
(a) FoodMarket4		
$P_{HBT} > P_{HTT}$	94	6
$P_{VBT} > P_{VTT}$	97	3
(b) Kimono1		
$P_{HBT} > P_{HTT}$	91	9
$P_{VBT} > P_{VTT}$	95	5
(c) BasketballPass		
$P_{HBT} > P_{HTT}$	99	1
$P_{VBT} > P_{VTT}$	92	8
(d) BQMall		
$P_{HBT} > P_{HTT}$	96	4
$P_{VBT} > P_{VTT}$	89	11
(e) BQSquare		
$P_{HBT} > P_{HTT}$	90	10
$P_{VBT} > P_{VTT}$	93	7

It can be observed that when the pixel mean difference of the horizontal binary partition is greater than that of the horizontal ternary partition, there is a stronger preference for selecting the horizontal binary partition. Similarly, when the pixel mean difference of the vertical binary partition is greater than that of the vertical ternary partition, there is a stronger preference for selecting the vertical binary partition. Therefore, when both Mode-a and Mode-b in the MPPPL are either horizontal partition modes or vertical partition modes, an early decision on the optimal partition mode is considered. If Mode-a is a horizontal binary partition, then we would calculate the pixel mean difference of the horizontal binary partition (P_{HBT}) and the pixel mean difference of the horizontal ternary partition (P_{HTT}), and if P_{HBT} is greater than P_{HTT}, then this indicates that the horizontal binary partition has a high probability of being the optimal partition mode at this time. We would then skip the RDC test of the horizontal ternary tree partition and set the horizontal binary partition as the optimal partition mode for the current CU. Similarly, if the pixel mean difference of the vertical binary partition (P_{VBT}) is greater than P_{VTT} in the vertical direction, then the vertical binary partition is set as the optimal partition mode.

If the partition modes in the MPPPL do not follow any clear rules, then the RDC (rate-distortion cost) is first calculated for the Mode-a and Mode-b partition modes in the order specified by the MPPPL. In the MPPPL, Mode-a has the highest probability of being the optimal partition mode. If the RDC of Mode-b is greater than the RDC of the Mode-a, then we would stop calculating the RDCs of the other partition modes, and the optimal mode would be the partition mode corresponding to Mode-a. If the RDC of the Mode-b partition mode is smaller than the RDC of the Mode-a partition mode, then we would

calculate the RDC of the Mode-c partition mode at this time and compare it with the RDC of the Mode-b partition mode. If the RDC of the Mode-b partition mode is smaller than that of Mode-c, then Mode-b would be the optimal partition mode at this time; otherwise, this process is repeated until the optimal partition mode is found. The algorithm flow chart is shown in Figure 6.

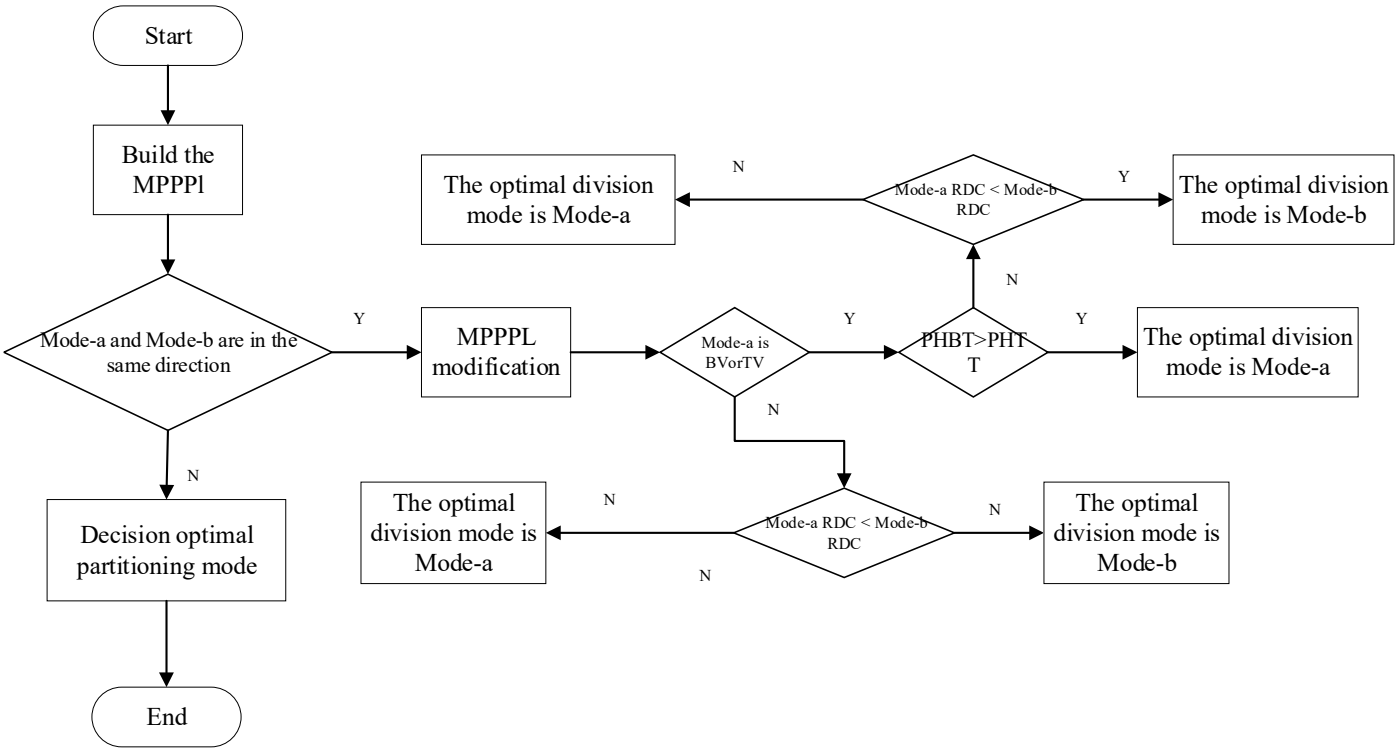


Figure 6. The flowchart of the proposed algorithm.

3.4. Reference Line for Quick Decision

VVC has increased the number of intra prediction modes to 65, and it introduced the multiple reference line (MRL). The traditional intra prediction mode only refers to the reference pixels in the adjacent left column and the upper row when completing predictions. MRL extends this to three rows and three columns. In MRL, in order to obtain the optimal intra prediction angle, MRL references line 0, line 1, and line 3. Line 0 and line 1 can provide reference information for nearby pixels and line 3 provides reference information for distant pixels. Line 2 is difficult but provides useful information, and so no reference is made. In the calculation process, the three lines are tested in turn, and then the line or column with a smaller RDC is selected as the reference line or reference column, which increases the computational complexity of the intra prediction.

We randomly selected the reference sequence in the VVC standard and counted the proportion of different reference lines or columns selected as the optimal reference lines during the intra prediction process. The statistical results are shown in Figure 7, where FO, BQ, KI, BA, and JO correspond to the coding sequences FoodMarket4, BQMall, Kimono1, BasketballPass, and Johnny, respectively. Based on the statistical data shown in Figure 7, it is not difficult to determine that line 0 was selected as the reference pixel for prediction in most cases while the probability of reference line 3 being selected as the reference pixel was very small, with an average of approximately 8%, and so reference line 3 was skipped in the vast majority of cases.

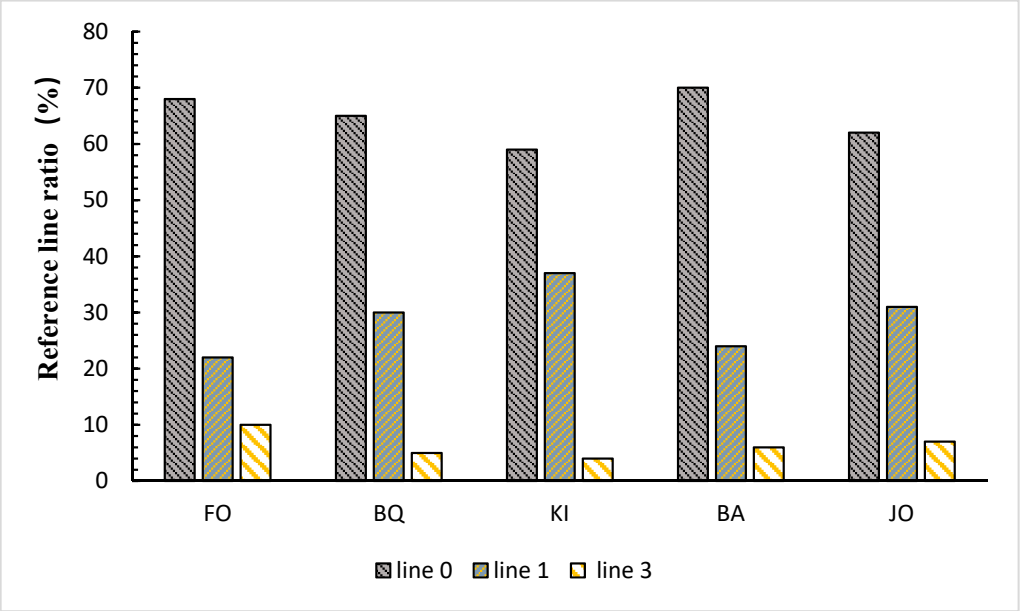


Figure 7. Reference line ratios of the different partition sequences.

Reference line 3 was added to the reference line as we hoped that it could provide different reference information from lines 0 and 1. The difference in this reference information was directly reflected in the difference between the reference pixel contents. We measured this difference by comparing the pixel content similarity (PCS) of reference line 3 and reference line 2. The calculation formulas used are shown below, where PCS_H is the content similarity between reference line 3 and reference line 2, PSH_V is the content similarity between reference column 3 and reference column 2, (x, y) is the starting coordinate of the current prediction block, and w and h are its width and height.

$$PCS_H = \frac{\sum_{i=x}^{x+w} [p(i, y-4) - \frac{p(i,y-2)+p(i,y-1)}{2}]}{\sum_{i=x}^{x+w} [p(i, y-3) - \frac{p(i,y-2)+p(i,y-1)}{2}]} \text{ and} \quad (15)$$

$$PCS_V = \frac{\sum_{i=y}^{y+h} [p(x-4, i) - \frac{p(x-2,i)+p(x-1,i)}{2}]}{\sum_{i=y}^{y+h} [p(x-3, i) - \frac{p(x-2,i)+p(x-1,i)}{2}]}. \quad (16)$$

In Formulas (15) and (16), if reference line 3 is similar to reference line 2, the difference between them and other reference lines is closer. If the pixel content similarity between reference line 3 and reference line 2 is between Re and $1/Re$, this means that reference line 3 and reference line 2 have a high degree of similarity in pixel content. At this time, reference line 3 was difficult to use for providing useful reference information compared to reference line 2, and the calculation for reference line 3 was skipped in subsequent calculations. In order to determine the optimal threshold m, we counted the accuracy rate (Ac) of reference line 3 as the final reference line under the different thresholds. The calculation formula used for Ac is as follows:

$$Ac = Z_m / Z_{vvc}, \quad (17)$$

where Z_m is the probability that reference line 3 will be selected as the final reference line under the Re threshold and Z_{vvc} is the probability that reference line 3 will be selected as the final reference line in the original VVC. The experimental results are shown in Figure 8, where T, M, B, P, and F represent the coding sequences Tango2, MarkPlace, BQTerrace,

PartyScene, and FourPeople, respectively. Observing Figure 8, it is not difficult to determine that the accuracy rate gradually increased with the increase in the threshold, and it reached its highest when the threshold was 1, but at this time, the judgment condition was too strict and it was difficult to reduce the complexity. It had a better accuracy rate when the value of *Re* was 0.8, and so we set the final value of *Re* as 0.8.

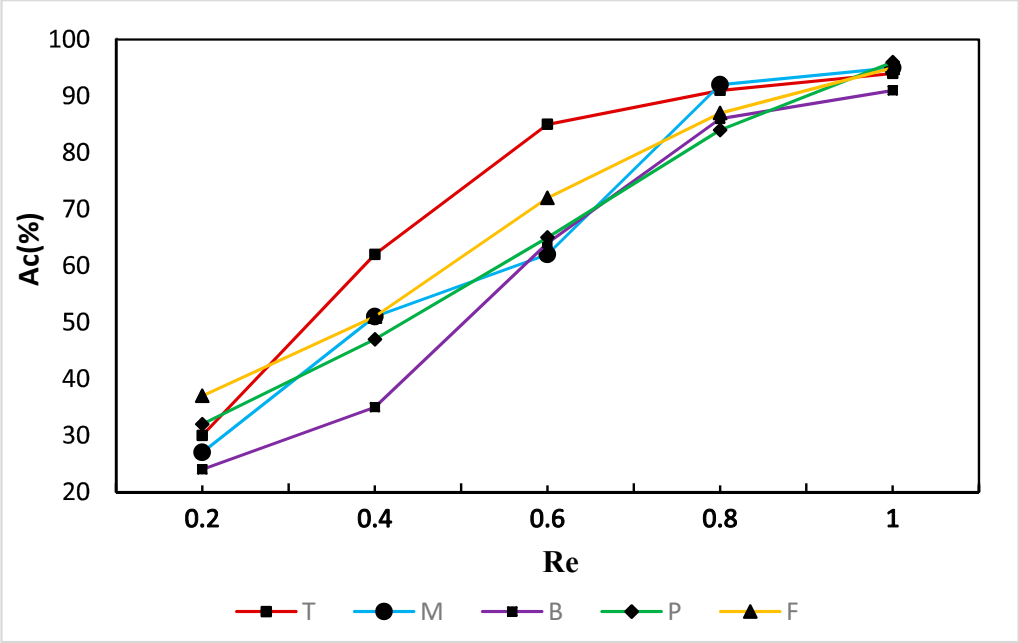


Figure 8. Reference line of *Ac* under the different thresholds.

4. Experimental Results

In order to assess the performance of our method, we conducted experiments using the reference software VVC VTM-13.0. These experiments involved testing 21 sequences which were selected from six sequences with different resolutions as recommended by the JVET common test conditions. Our proposed algorithm was evaluated using the following four different quantization parameters (QPs): 22, 27, 32, and 37. The performance of the algorithm was measured by *Ts* and BDBR, where *Ts* was calculated as follows:

$$Ts = \frac{1}{4}\sum_{Qpi\in\{22,27,32,37\}} \frac{\mathrm{To}(Qpi) - Tp(Qpi)}{\mathrm{To}(Qpi)} \times 100\% \quad (18)$$

In the context of our evaluation, the total encoding time of the original VVC standard was represented by To while the total encoding time of our proposed method was denoted as *Tp*. Additionally, BDBR was used to measure the degree of loss in the encoding performance. A smaller BDBR value indicated a lesser loss in compression performance. By conducting these experiments and utilizing the aforementioned metrics, our aim was to evaluate and analyze the effectiveness of the proposed algorithm in terms of saving encoding time and increasing compression efficiency.

We tested the coding performances of the two individual algorithms, and the results are shown in Table 3. As shown in Table 3, the proposed fast partition algorithm saved 49.77% of the encoding time, and the reference line fast decision algorithm saved 5.43% of the encoding time, with a negligible loss in encoding performance.

In Table 4, we present the experimental results, comparing our proposed method with the original VTM-13.0 platform as well as other fast algorithms. Our method demonstrated significant improvements in terms of encoding time and compression performance. Compared to VTM-13.0, our proposed method achieved an average reduction in encoding time of 52.26% compared to the BDBR loss of 1.23%. This indicated that our method significantly

improved the efficiency of the encoding process while maintaining a high compression performance.

Table 3. Experimental results of the different algorithms for the different test sequences.

Coding Sequence	Fast Partition		Skip Reference Line 3 Decision	
	BDBR (%)	Ts (%)	BDBR (%)	Ts (%)
Tango2	1.11	47.12	0.04	5.75
Campfire	0.99	50.25	0.06	6.72
CatRobot	1.35	51.11	0.08	4.11
DatLightRoat2	1.16	47.25	0.11	7.25
ParkRunning3	1.05	50.11	0.09	3.21
MarkPlace	1.23	49.32	0.09	6.42
Cactus	1.01	48.75	0.11	5.21
BasketballDrive	0.88	50.42	0.21	7.32
BQTerrace	0.71	52.31	0.33	4.11
RaceHorses	0.91	45.15	0.12	6.21
BasketballDrill	1.02	49.03	0.09	4.32
BQMall	1.22	50.62	0.04	7..21
PartyScene	1.11	49.32	0.12	4.21
RaceHorses	0.87	48.01	0.09	5.01
BasketballPass	0.96	52.01	0.14	6.02
BQSquare	0.93	51.04	0.13	6.45
BlowingBubbles	1.11	49.21	0.06	4.66
FourPeople	0.94	50.22	0.21	5.23
Johnny	0.92	47.42	0.21	6.69
KristenAndSara	1.08	50.32	0.09	4.24
average	1.02	49.77	0.12	5.43

Table 4. Comparison of the algorithm results.

Coding Sequence	Reference [11]		Reference [16]		Algorithm in This Paper	
	BDBR (%)	Ts (%)	BDBR (%)	Ts (%)	BDBR (%)	Ts (%)
Tango2	0.74	37.01	1.59	51.85	1.34	55.97
Campfire	0.66	34.05	1.61	50.11	1.43	50.96
CatRobatl	0.54	29.91	1.55	50.59	1.21	51.41
DatLightRoat2	0.71	32.12	1.77	47.92	1.16	53.25
ParkRunning3	0.68	32.11	1.39	54.33	1.41	51.07
MarkPlace	0.55	34.15	1.46	48.11	1.19	55.11
Cactus	0.61	30.73	1.31	44.95	1.32	50.07
BasketballDrive	0.74	34.48	0.94	48.33	1.26	49.61
BQTerrace	0.62	30.85	1.49	46.16	1.02	49.01
RaceHorses	0.46	27.83	0.98	51.04	1.35	54.27
BasketballDrill	0.41	26.55	1.05	51.18	1.28	53.21
BQMall	0.65	33.79	0.81	46.95	1.14	53.02
PartyScene	0.42	31.62	0.83	45.88	1.22	52.65
RaceHorses	0.55	30.17	1.24	48.33	1.19	57.21
BasketballPass	0.7	30.53	1.18	45.17	1.07	48.95
BQSquare	0.29	29.97	1.41	40.04	1.11	53.35
BlowingBubbles	0.43	29.34	0.99	43.86	1.36	51.15
FourPeople	0.78	35.63	0.89	46.68	1.29	53.77
Johnny	0.69	30.65	1.27	39.21	1.05	51.57
KristenAndSara	0.59	31.38	1.63	49.82	1.38	50.91
average	0.59	31.44	1.29	47.91	1.23	52.26

Compared to the method described in reference [11], our approach saved more encoding time. Compared with the method in reference [16], our method was superior for both the time savings and the BDBR. The method in [16] required a large amount of data to train

the model, and the performance of the method may have been affected if the training data were insufficient or not comprehensive enough.

We plotted the corresponding curve graph for the data in Table 3, as shown in Figure 9. By observing Figure 9, it is evident that the proposed algorithm in this paper significantly saved encoding time compared to the method proposed in reference [11]. Furthermore, when compared to reference [16], the curve corresponding to the saved encoding time in the proposed algorithm is smoother, indicating that the proposed algorithm was stable for various coding sequences. The method presented in reference [16] showed significant variation in its encoding results for different coding sequences, indicating lower stability compared to our proposed algorithm.

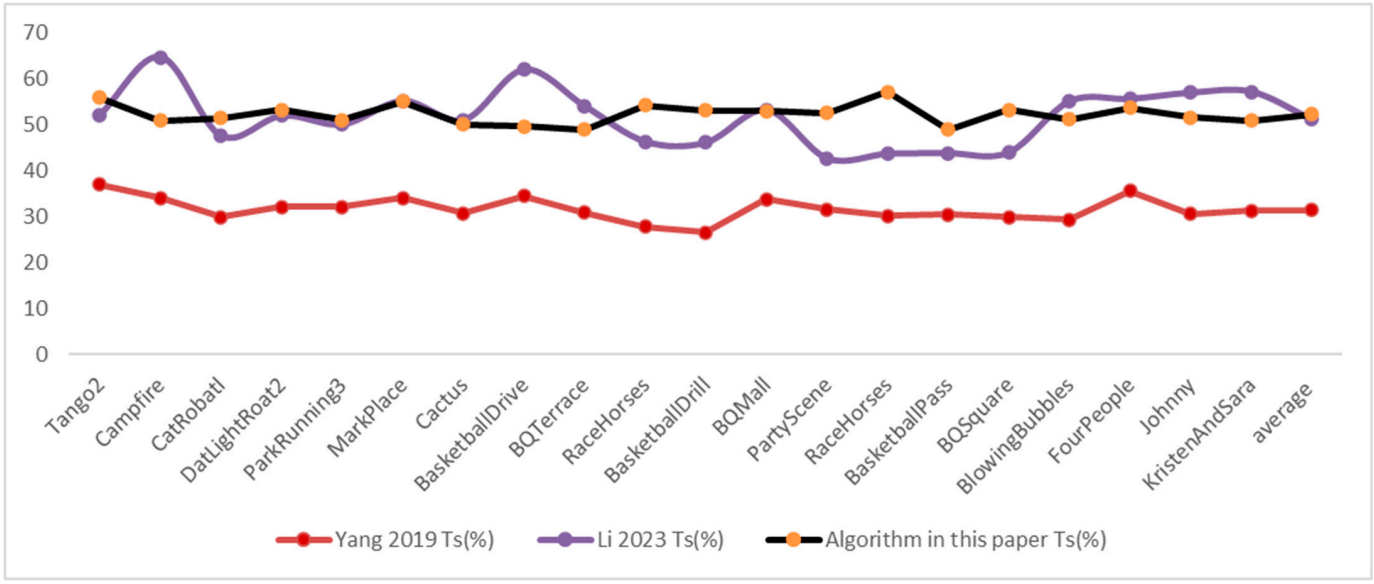


Figure 9. Encoding time savings curves for the different algorithms [11,16].

In Figure 10, we present the rate-distortion (RD) curves for the sequences BasketballPass and BlowingBubbles, comparing the performance of our proposed method (represented by the red curve) with that of the original encoder (represented by the black curve). Figure 10a,b shows the RD curves for BasketballPass and BlowingBubbles, respectively. For BlowingBubbles, the RD curve of the proposed method is slightly lower than that of the original encoder. For BasketballPass, the RD curve of our proposed method nearly coincides with that of the original encoder. Therefore, our algorithm effectively saved encoding time while maintaining a high RD performance.

Figure 11 shows a subjective quality comparison for BasketballDrill. As we can see in Figure 11, the reconstructed frame of the proposed algorithm is nearly the same as that of the original encoder. Other sequences had similar situations, which verified the effectiveness of the proposed algorithm.

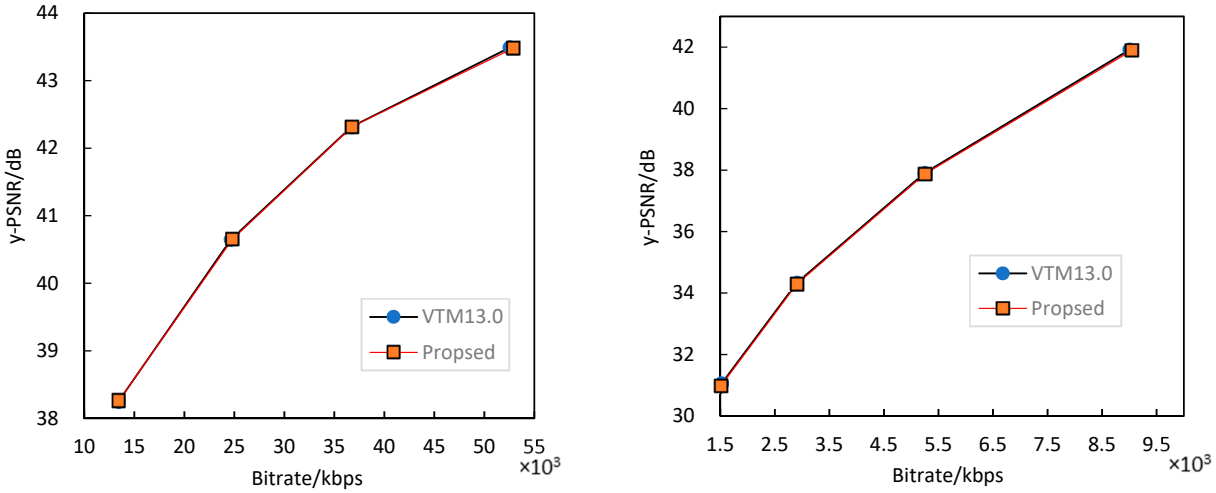


Figure 10. Comparison of the RD performances between the proposed algorithm and the original encoder: (**a**) RD performance of BasketballPass, and (**b**) RD performance of BlowingBubbles.

Figure 11. Subjective quality comparison: (**a**) the reconstructed frame of the original encoder, and (**b**) the reconstructed frame of the proposed algorithm.

5. Conclusions

We have proposed a low-complexity algorithm for VVC intra coding which exploits the effect of the sub-block texture complexity on the selection of partitioning modes to construct the MPPPL, and it decides the optimal mode on this basis. At the same time, unnecessary reference lines are skipped by using the pixel content similarity between the reference lines. The algorithm was tested on the reference software VVC VTM-13.0 and achieved an average coding time savings of 52.26%, with a BDBR increase of 1.23%.

Author Contributions: Methodology, X.S.; Software, S.Z.; Resources, H.Z.; Data curation, S.Z.; Writing – original draft, S.Z.; Funding acquisition, G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [National Natural Science Foundation of China] grant number [62001283].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yuan, H.; Guo, C.; Liu, J.; Wang, X.; Kwong, S. Motion-homogeneous-based fast transcoding method from H. 264/AVC to HEVC. *IEEE Trans. Multimed.* **2017**, *19*, 1416–1430. [CrossRef]
2. Hamidouche, W.; Biatek, T.; Abdoli, M.; Francois, E.; Pescador, F.; Radosavljevic, M.; Menard, D.; Raulet, M. Versatile video coding standard: A review from coding tools to consumers deployment. *IEEE Consum. Electron. Mag.* **2022**, *11*, 10–24. [CrossRef]

3. Tissier, A.; Mrecat, A.; Amestoy, T. Complexity reduction opportunities in the future VVC intra encoder. In Proceedings of the International Workshop on Multimedia Signal Processing, Kuala Lumpur, Malaysia, 27–29 September 2019; pp. 27–29.
4. Pakdamaf, F.; Adelimanesh, M.; Gabbouj, M. Complexity analysis of next-generation VVC encoding and decoding. In Proceedings of the International Conference on Image Processing, Abu Dhabi, United Arab Emirates, 25–28 October 2020; pp. 25–28.
5. Amna, M.; Imen, W.; Fatma, E.S. Fast multi-type tree partitioning for versatile video coding using machine learning. *Signal Image Video Process.* **2023**, *17*, 67–74. [CrossRef]
6. Fu, T.; Zhang, H.; Mu, F. Fast CU partition algorithm for H. 266/VVC intra-frame coding. In Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME), Shanghai, China, 8–12 July 2019; pp. 55–60.
7. Li, T.; Xu, M.; Tang, R.; Chen, Y.; Xing, Q. Deep QTMT: A deep learning approach for fast QTMT-based CU partition of intra-mode VVC. *IEEE Trans. Image Process.* **2021**, *30*, 5377–5390. [CrossRef] [PubMed]
8. Zhang, Q.; Guo, R.; Jiang, B. Fast CU decision-making algorithm based on DenseNet network for VVC. *IEEE Access* **2021**, *9*, 119289–119297. [CrossRef]
9. Taabane, I.; Menard, D.; Mansouri, A. Machine learning based fast QTMTT partitioning strategy for VVenC encoder in intra coding. *Electronics* **2023**, *12*, 1338. [CrossRef]
10. Zhang, Q.; Wang, Y.; Huang, L. Fast CU partition and intra mode decision method for H. 266/VVC. *IEEE Access* **2020**, *8*, 117539–117550. [CrossRef]
11. Yang, H.; Shen, L.; Dong, X.; Ding, Q.; An, P.; Jiang, G. Low-complexity CTU partition structure decision and fast intra mode decision for versatile video coding. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 1668–1682. [CrossRef]
12. Wu, S.; Shi, J.; Chen, Z. HG-FCN: Hierarchical Grid Fully Convolutional Network for Fast VVC Intra Coding. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *8*, 5638–5649. [CrossRef]
13. Saldanha, M.; Sanchez, G.; Marcon, C. Learning-based complexity reduction scheme for VVC intra-frame prediction. In Proceedings of the 2021 International Conference on Visual Communications and Image Processing (VCIP), Munich, Germany, 5–8 December 2021; pp. 1–5.
14. Tissier, A.; Hamidouche, W.; Vanne, J. CNN oriented complexity reduction of VVC intra encoder. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 25–28 October 2020; pp. 3139–3143.
15. Hoangvan, X.; NguyenQuang, S.; DinhBao, M. Fast QTMT for H. 266/VVC intra prediction using early-terminated hierarchical CNN model. In Proceedings of the 2021 International Conference on Advanced Technologies for Communications (ATC), Ho Chi Minh City, Vietnam, 14–16 October 2021; pp. 195–200.
16. Li, H.; Zhang, P.; Jin, B. Fast CU Decision Algorithm Based on Texture Complexity and CNN for VVC. *IEEE Access* **2023**, *11*, 35808–35817. [CrossRef]
17. Yao, Y.; Wang, J.; Du, C. A support vector machine based fast planar prediction mode decision algorithm for versatile video coding. *Multimed. Tools Appl.* **2022**, *81*, 17205–17222. [CrossRef]
18. Abdallah, B.; Belghith, F.; Ben Ayed, M.A. Fast QTMT decision tree for Versatile Video Coding based on deep neural network. *Multimed. Tools Appl.* **2022**, *81*, 42731–42747. [CrossRef]
19. Zheng, J.S.; Zong, J.P.; Gang, Y.J. Fast intra partition and mode prediction for equirectangular projection 360-degree video coding. *IET Image Process.* **2023**, *17*, 558–569. [CrossRef]
20. Tsai, Y.H.; Lu, C.R.; Chen, M.J. Visual Perception Based Intra Coding Algorithm for H. 266/VVC. *Electronics* **2023**, *12*, 2079. [CrossRef]
21. Zhang, S.; Zhang, R.; Jing, X. A fast Multi-Type-Tree split decision algorithm of intra coding unit in VVC. In Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Bilbao, Spain, 15–17 June 2022; pp. 1–5.
22. Zhang, D.; Li, Q. An Efficient CU Partition Algorithm for VVC Intra Coding. *J. Phys. Conf. Ser.* **2021**, *1815*, 012006. [CrossRef]
23. Li, Q.; Meng, H.; Li, Y. Texture-based fast QTMT partition algorithm in VVC intra coding. *Signal Image Video Process.* **2022**, *17*, 1581–1589. [CrossRef]
24. Shang, X.; Li, G.; Zhao, X. Low complexity inter coding scheme for Versatile Video Coding (VVC). *J. Vis. Commun. Image Represent.* **2023**, *90*, 103683. [CrossRef]
25. Zhang, Q.; Zhao, Y.; Jiang, B. Fast CU partition decision method based on texture characteristics for H. 266/VVC. *IEEE Access* **2020**, *8*, 203516–203524. [CrossRef]
26. Fan, Y.; Sun, H.; Katto, J. A fast QTMT partition decision strategy for VVC intra prediction. *IEEE Access* **2020**, *8*, 107900–107911. [CrossRef]
27. Shang, X.; Li, G.; Zhao, X. Fast CU size decision algorithm for VVC intra coding. *Multimed. Tools Appl.* **2023**, *82*, 28301–28322. [CrossRef]