Fast CU Spliting Algorithms for Virtual Reality Video Based on KNN

Zhi Liu, Peiran Song, and Mengmeng Zhang

***Abstract*—The coding framework for virtual reality video at present is first projecting 3D data to 2D format, then encoding it by traditional coding tools, which has much high computational complexity. In order to reduce the coding complexity based on the quality evaluation standard of virtual reality video, this paper presents a fast algorithm to speed up the Coding Unit (CU) partitioning by predicting the maximum depth of LCU with KNN classifier. Experimental results show that the proposed fast algorithm provides an average time reduction rate of 37.9% compared to the reference HM-16.16+360lib4.0, with only 1.31% BD-rate increase.**

***Index Terms*—Fast algorithm, LCU, KNN, virtual reality video.**

I. Introduction

Virtual reality video is a special kind of video representing the whole scene of the environment in 360 degree. It is captured by multiple professional cameras, and spliced using software and can be played by special device. It also provides the viewer with various functions to manipulate the video, such as zoom in and out and moving in all directions, so as to simulate and reproduce the real environment [1].

At present, the coding and transmission of virtual reality video mainly relies on projection every frame of the 3D style data into 2D one, and then using traditional coding framework such as HEVC, H.264 to fulfill encoding. The commonly used projection formats are ERP, EAP, and CMP and so on. In addition, different from traditional video, virtual reality video has its own quality evaluation metric. In this paper, we will study the virtual reality video in ERP projection format.

HEVC is one of the coding framework used in virtual reality video coding. HEVC adopts the coding structure of coding tree unit (CTU), which is the basic processing unit of HEVC. A CTU consists of 1 brightness CTB, 2 chromaticity CTB and corresponding syntax elements. Figure 1 shows a frame divided into CUs in CTU. A CTU may contain only one encoding unit (CU), and HEVC can also use quadtree structure to recursively divide CU into many different sizes of CU [2].

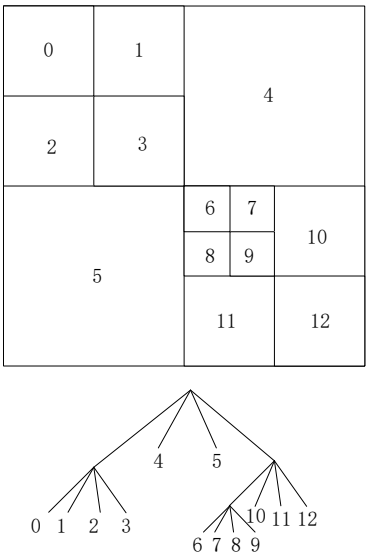


Fig. 1. Example of code tree unit.

There are four kinds of CU in HEVC: 64×64, 32×32, 16×16, and 8×8. For a LCU with size 64×64, encoder first treat it as a CU, calculate the best prediction mode of it, and record the best prediction data in the current partitioning mode. Then encoder divide the current LCU into four 32×32 CUs. Encoder calculate the best prediction mode of the 32x32 CUs, and record the best prediction data. Similarly, the encoder divides each 32×32 CU into four 16×16 CUs, and calculate the best prediction model of each 16×16 CU and record it, and divides each 16×16 CU into four 8×8 CUs. Then Encoder calculate the best prediction model corresponding to 8×8 and record the prediction data. Since 8x8 is the smallest CU, here encoder just loop through the best prediction model corresponding to each 8×8 and record its data. When 8x8 CUs' calculating is complete, encoder compare the sum of four 8×8 CUs' RD-Costs to the RD-Cost corresponding to the 16×16 CU's to decide whether to choose the four 8×8 CUs or the a 16×16 CU. After the first 16×16 CU is completed, encoder repeat the previous steps to determine whether the second 16×16 CU is divided into four 8×8 CUs or 16×16 CU, and then the third 16×16 CU and the fourth 16×16 CU. After the 16×16 CU is completed, encoder compare the sum RD-Cost of the four 16×16 CUs with the RD-Cost of a 32×32 CU to determine whether to select 32×32 CU or 16×16CUs. When the first 32×32 CU is completed, encoder repeat the previous steps to determine the partitioning pattern for the second, third, and fourth 32×32 CU. When all four 32×32 computations are complete,

Manuscript received July 20, 2019; revised June 5, 2020. This work is supported by the National Natural Science Foundation of China (No.61370111), Beijing Municipal Natural Science Foundation (No.4172020), Great Wall Scholar Project of Beijing Municipal Education Commission (CIT&TCD20180304), Beijing Youth Talent Project (CIT&TCD 201504001), and Beijing Municipal Education Commission General Program (KM201610009003).

The authors are with the North China University of Technology Beijing, China (e-mail: lzliu@ncut.edu.cn, 18336343993@163.com, muchmeng@126.com).

doi: 10.18178/ijmlc.2020.10.6.1005

we compare the sum RD-Costs of the 32×32CUs with the RD-Cost of a 64×64CU, and then decide whether to choose 64×64 CU or four 32×32 CUs and its descending partition.

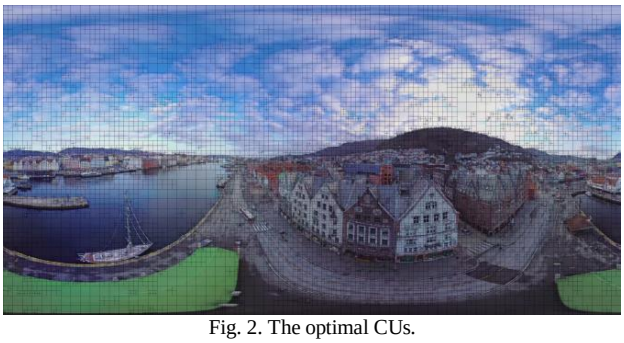


Fig. 2. The optimal CUs.

In HEVC, to determine whether a block in quadtree coding structure needs to be further partitioned, it is necessary to compare the coding cost of all blocks with that of the block after traversal of all the blocks mentioned above. If the RD-cost of CU is larger than the sum of RD-cost of its sub-CU, the smaller CUs are needed. If the RD-cost of the current CU is larger than the RD-cost of the parent CU, there is no need to divide it, and the current CU as a whole. Obviously, these comparisons occur after the end of all CU traversals of different sizes, which means much high computation burden. In fact, in many cases, the size of CU varies in the optimal partition, if we can predict the maximum depth in a LCU, we can terminate the partitioning process in advance, and there is no need to traverse all the possibilities.

In this paper, we use KNN to predict the maximum depth of LCU, and to reduce the redundancy of LCU partitioning operation.

II. Related Works

In [3], K. Choi proposed a coding tree termination method for the CU SKIP mode. Deyuan Liu [4] proposed a fast CU size decision algorithm based on Support Vector Machines (SVM). The [5] uses weighted SVM to predict CU premature termination to optimize computation complexity. In [6], a Bayesian decision rule based early termination method was reported, in which on-line learning and off-line learning were jointly applied to generate model parameters of classifiers.

In [7], a fast CU partitioning algorithm is proposed for HEVC encoder, which early on terminates the CU partitioning process based on the Bayesian decision rule using joint online and offline learning. In [8], author proposed a fast and efficient mode decision algorithm based on the Newman-Pearson rule, which consists of early SKIP mode decision and fast CU size decision. In [9], author proposed a machine learning-based fast coding unit (CU) depth decision method for High Efficiency Video Coding (HEVC), which optimizes the complexity allocation at CU level with given rate-distortion (RD) cost constraints.[10] proposes a fast CU splitting algorithm which can narrow CU depth range and early terminate the CU splitting based on the Sobel edge detection operator.[11] proposed a method to reduce the high encoding time by pruning the coding quad-trees using prediction residuals statistics.

Until now, a number of methods have been proposed to reduce the encoding complexity of HEVC on early CU decision. However, the experiment of this part of algorithm is only used for HEVC. Although the virtual reality video is encoded by HEVC after projection, its quality evaluation standard is different from HEVC. Moreover, in the above-mentioned articles, some of the algorithms are based on the threshold of video statistics for fast partitioning. But the statistical threshold does not always apply to all videos.

III. LCU Depth Prediction Based on KNN

LCU may contain four sizes of CU. If we predict the size of the smallest CU in LCU before encoding LCU, and skip calculating the smaller CU after encoding this CU, we can improve the encoding efficiency of the encoder when dividing CU.

In this paper, the minimum CU size in LCU is predicted to skip the calculation of CU partitioning method in LCU in advance. This problem can also be regarded as a classification problem. Because of the computational efficiency and the complexity of model training, KNN classifier is adopted in this paper. This part mainly includes LCU feature selection, classification method, classification accuracy analysis, KNN parameter K and prediction set proportion judgment.

A. LCU Complexity Feature Analysis Based on Sobel Filtering

Generally, the simpler area can get better coding effect under the larger CU, while in the more complex area, it needs to be divided into smaller CU for prediction. According to this idea, many people have proposed some related algorithms, which can achieve the goal of fast partitioning CU to a certain extent. In this paper, the Sobel operator is used to filter the content of the encoding LCU to calculate the complexity of the encoding LCU.

In this paper, we use horizontal and vertical Sobel filtering for CU content to be coded. The gradients Gx, Gy in horizontal and vertical directions are obtained by calculation. A in the equation (1) represents the content of LCU.

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \tag{1}$$

$$Gy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{2}$$

After obtaining Gx and Gy, we calculate edge points and average gray values by equations (3) and (4). i and j represent the coordinates of each pixel, and n represent the width of the LCU .The more the number of edge points in an

LCU is, the more complex the block is, and the greater the depth level of division is.

$$\text{Mean}_{Gx} = \frac{\sum_{i}^{m}\sum_{j}^{n}|Gx(i,j)|}{n*n} \quad (3)$$

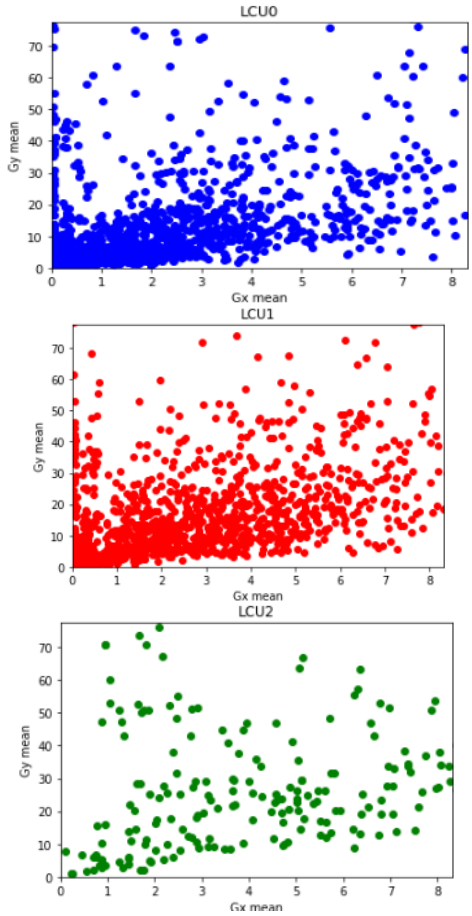
$$\text{Mean}_{Gy} = \frac{\sum_{i}^{m}\sum_{j}^{n}|Gy(i,j)|}{n*n} \quad (4)$$

B. Prediction of CU Depth Range Based on the Number of Edge Means

After encoding, the LCU with the optimal partitioning result may have 64×64 CU, 32×32 Cu, 16×16 CU and 8×8 minimum CU blocks. These four different sizes of CU blocks also represent the four different depths of CU. We define the minimum CU size in the LCU as the depth of the LCU. According to the depth of LCU, LCU can be divided into four categories: $LCU_0, LCU_1, LCU_2, LCU_3$. It means that the minimum CU sizes in a LCU are 64×64, 32×32, 16 ×16, and 8×8.

If the size of the smallest CU block is predicted before LCU partitioning, further partitioning can be terminated after encoding the corresponding size CU, thus improving the encoding efficiency.

Through experiments, we find that the horizontal and vertical edge mean values of LCU are closely related to its depth.



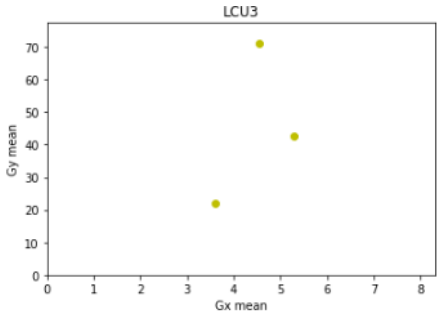


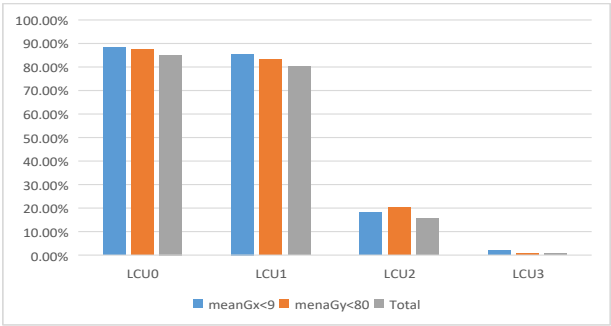
Fig. 3. Distribution of LCU at different depths in threshold range.

As shown in the Fig. 3, statistical data show that the majority of LCU_0 and LCU_1 are located in $mean_{Gx} < 9$ and $mean_{Gy} < 80$ regions.

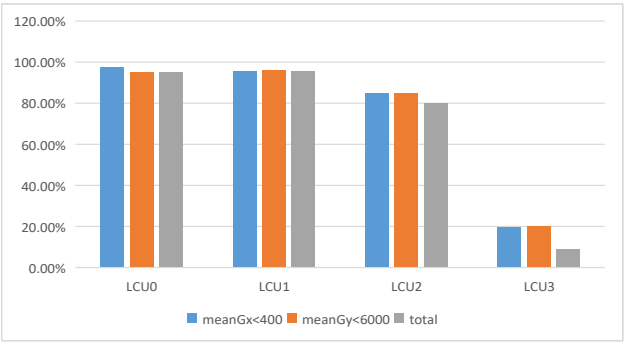
The different LCU ratios in the thresholds are shown in the following Table I:

TABLE I: PROPORTION WITHIN IN DIFFERENT THRESHOLDS

	LCU_0	LCU_1	LCU_2	LCU_3
$mean_{Gx} < 9$	88.3%	85.5%	18.2%	2.1%
$mean_{Gy} < 80$	87.6%	83.3%	20.1%	0.6%
$mean_{Gx} < 9$ & $mean_{Gy} < 80$	85.1%	80.3%	15.6%	0.15%



	LCU_0	LCU_1	LCU_2	LCU_3
$mean_{Gx} < 400$	97. 4%	95.6%	85.0%	16.9%
$mean_{Gy} < 6000$	95.4%	96.3%	85.1%	20.1%
$mean_{Gx} < 400$ & $mean_{Gy} < 6000$	95.4%	95.6%	80.3%	8.9%



It can be seen that $mean_{Gx}$ and $mean_{Gy}$ can effectively classify LCUs of different depths.

At present, the commonly used fast CU algorithm often extracts the features of CU, quantifies them and then extracts the threshold value, which is used to process the CU larger than or less than the threshold value. However, the threshold of this algorithm is usually derived from the

statistics of the test video, and it may not be able to represent all the features of the video very well.

For different videos, the optimal threshold will not be the same because the image characteristics and complexity of each video are different. The statistical thresholds are often neutralized by the characteristics of the statistical video set. The statistical thresholds are not necessarily optimal for the video to be coded.

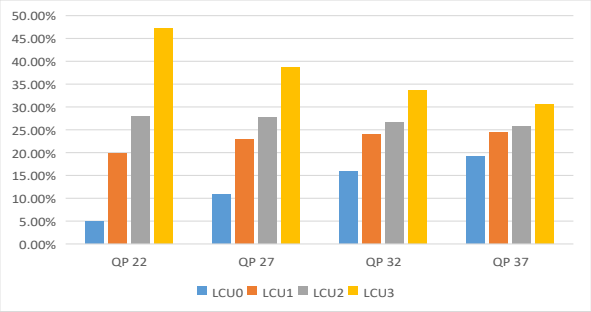
C. Adjustment of Classification Method

The experimental results show that the edge characteristics of LCU_0 and LCU_1 frames are similar, and it is difficult to distinguish them by using edge features.

We counted the proportion of each CU in the video, as shown in the Table II.

TABLE II: LCU STATISTICS

	QP 22	QP 27	QP 32	QP 37
LCU_0	5.02%	10.80%	15.91%	19.25%
LCU_1	19.77%	22.95%	23.97%	24.35%
LCU_2	28.03%	27.60%	26.55%	25.75%
LCU_3	47.17%	38.63%	33.55%	30.66%



As shown in the table, statistically, LCU_0 does not account for a large proportion of video. Because it is similar to LCU_1, we classify it as $LCU_{shallow}$.

Therefore, in this classification, we classify LCUs into three class, which are: $LCU_{shallow}$, LCU_2, LCU_3.

D. LCU Depth Prediction Based on KNN Classifier

Because LCU depth prediction itself can also be regarded as a classification problem, we can use classifier to encode a part of the video frame normally, record the depth and edge features of LCU, and then use these data to predict the depth of the LCU of another part of the frame.

Since the training of classifier in this algorithm is carried out in coding, it is necessary to select the algorithm with lower computational complexity for training and prediction in order to achieve the goal of improving coding efficiency.

KNN algorithm is low in training complexity and relatively simple in structure. Although the time complexity of prediction is relatively high, its computational complexity is much lower than that of traversing all depth LCUs. Therefore, this paper intends to use KNN to predict the depth of LCU.

KNN is a basic classification and regression method. Its input is the feature vector of an instance. By calculating the distance between the new data and the trained data, K (K >= 1) neighbors are selected for classification and judgment (voting) or regression. If K = 1, the new data is simply assigned to the class of its nearest neighbors.

As shown in Fig. 4, which class is the blue circle determined to be, is it a red square or a green triangle? If K = 3, the green circle will be assigned to the green triangle class because the proportion of the red square is 2/3. If K = 5, the green circle will be assigned to the blue quadrangle class because the proportion of the green triangle is 4/5. The KNN method is more suitable than other methods for the intersection or overlap of class domains.

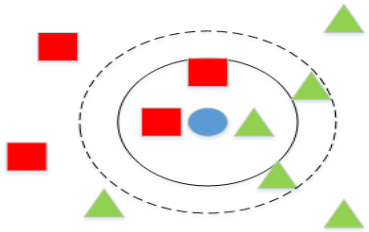


Fig. 4. KNN classification example.

The KNN algorithm itself is simple and effective. It is a lazy-learning algorithm. The classifier does not need training set, and the training time complexity is 0. The computational complexity of KNN classification is directly proportional to the number of documents in the training set, that is, if the total number of data in the training set is n, the classification time complexity of KNN is O (n). Although the KNN method also depends on the limit theorem in principle, it is only related to a small number of adjacent samples in class decision making. KNN method is more suitable than other methods for the intersected or overlapped sample sets because it mainly depends on the neighboring samples, rather than on the method of discriminating class domains. Because of the distribution characteristics of $mean_{Gx}$ and $mean_{Gy}$ in LCU, KNN is more suitable for classification.

In this algorithm, as shown as Fig. 5 we divide the encoded sequence into frames set, and each frames set has some frames. Among them, some frame is the training set, and the rest is the prediction set.

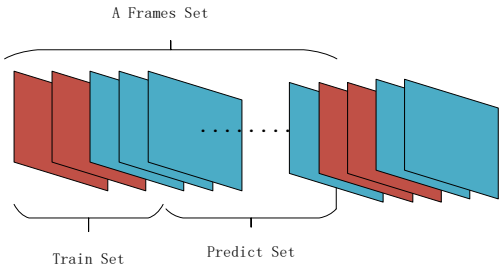


Fig. 5. Frames set.

When encoding the training set video, the original algorithm is used to partition the LCU, and the edge density attributes $mean_{Gx}$ and $mean_{Gy}$ of the LCU are recorded at the same time, and the depth after encoding is recorded.

Before encoding a LCU, the edge density features $mean_{Gx}$

and $mean_{Gy}$ are extracted. It is input into KNN classifier to classify and predict LCU depth.

If the predicted result is $LCU_{shallow}$, it will only code 64×64 CU and 32×32 CUs, and stop dividing deeper after coding 32×32 CUs.

If the predicted result is LCU_2, then the 64×64 CU and 8×8 CUs are skipped, and only 32×32 CUs and 16×16 CUs are coded, from which the optimal partition results are selected.

If the predicted result is LCU_3, then the 64×64 CU is skipped, only 32×32 CUs, 16×16 CUs and 8×8 CUs are coded, from which the optimal partition results are selected.

E. KNN Parameter Selection

For a given input sample $\vec{x}$, if its true value is y, the output value $\hat{y}$ predicted by the classifier $\hat{y} = f(\vec{x})$ may be inconsistent with the true value y. The result of correct classification is measured by the correct rate function and recorded as $R(y, f(\vec{x}))$.

$$R(y, f(\vec{x})) = \begin{cases} 1, & y = f(\vec{x}) \\ 0, & y \neq f(\vec{x}) \end{cases} \tag{5}$$

If the total number of frames is N, then the correct rate P of the classifier is:

$$P = \frac{\sum R(y, f(\vec{x}))}{N} \tag{6}$$

In this algorithm, since the prediction result is the depth of the LCU, if LCU_2 is predicted to be LCU_3, the LCU with depth of 2 will be calculated at 8×8 CUs, but in fact it has no effect on the coding result. If the number of LCU_2 divided into LCU_3 is L_{23}, the overall accuracy of the classifier is achieved:

$$P_{Total} = \frac{\sum R(y, f(\vec{x})) + L_{23}}{N} \tag{7}$$

If the total number of frames of the coded video is N, and the number of frames using KNN to predict LCU depth is N_{PF}, the proportion of training frames R_{PF} is:

$$R_{PF} = \frac{N_{PF}}{N} \times 100\% \tag{8}$$

In this paper, there are two parameters to be determined for KNN classification. One is the value of k and the other is R_{PF}. The P_{Total} of different R_{PF} and k is as Table III.

Because the increase of k and the decrease of the proportion of predicted frames will affect the efficiency of the fast algorithm, we choose the KNN classifier with k to be 2 and R_{PF} to be 60%.

TABLE III: TOTAL ACCURACY RATE

R_{PF} \ k	2	3	4	5	6
80%	83.7%	82.1%	81.8%	82.9%	82.3%
60%	86.7%	85.2%	85.0%	84.4%	84.7%
40%	84.9%	84.3%	86.6%	86.0%	86.0%
20%	86.1%	83.9%	86.8%	86.2%	87.1%

IV. EXPERIMENTAL RESULTS

360Lib is the JVET testing platform for virtual reality video. 360Lib is maintained under JVET under Subversion code. Users need to download the Subversion client for code download and version control. 360Lib mainly completes the projection of panoramic video. 360Lib can be used as plug-in, integrated into HM and JEM, and can be run separately. The main difference is that after conversion projection, it can send compression code directly instead of intermediate YUV, saving middle time. At this stage 360 lib is only responsible for projection, video coding is still dependent on HM or JEM. This paper aims at the optimization of 360lib in HM16.16.

We use our proposed LCU depth prediction algorithm on HM16.16 + 360lib4.0 to evaluate the effectiveness of the algorithm. Time reduction is calculated by:

$$\Delta T = \frac{T_{HM16.16+360lib4.0} - T_{propoesd}}{T_{HM16.16+360lib4.0}} * 100\% \tag{9}$$

$T_{HM16.16+360lib4.0}$ is the coding time of HM16.16 + 360lib4.0, $T_{propoesd}$ is the coding time of HM16.16 + 360lib4.0 using the proposed algorithm, and ΔT is the time reduction.

WS_PSNR is an objective quality assessment standard of virtual reality video adopted by 360Lib. According to the evaluation criterion, the pixels of different latitudes have different weights when projecting a 2D image onto a spherical field of view, and virtual reality videos are evaluated by adding weights to different latitudes of the 2D images projected from virtual reality videos. Assuming that the size of the 2D image after projection is *M*x*N*, the weighted mean square error is as follows:

$$WMSE = \frac{1}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} w(i,j)} \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} [y(i,j) - y'(i,j)]^2 * w(i,j) \tag{10}$$

$y(i, j)$ and $y'(i, j)$ are reference and test pixel values. $w(i, j)$ is the weight.

$$WS\_PSNR = 10\log(\frac{I_{MAX}^2}{WMSE}) \tag{11}$$

I_{MAX} is the largest grayscale in the current image. Since *WS_PSNR* is directly calculated from the projected 2D image, its weight $w(i, j)$ is related to the projection format. The weight of the ERP format video is as follows:

$$w(i,j)_{ERP} = \cos\frac{(j + 0.5 - N/2)\pi}{N} \tag{12}$$

Due to the characteristics of the virtual reality video, WS_PSNR is used as the objective quality assessment standard. We use WS_PSNR instead of the original PSNR to calculate the BD-rate in video coding.

The actual encoding time is measured on a workstation with a 3.60-GHz processor and 8GB of RAM. The anchor is under "encoder intra main" with "encoder 360 ERP" configuration. As shown in Table IV, the proposed algorithm achieves 37.9% time reduction, 1.31% BD-rate increase.

TABLE IV: EXPERIMENT RESULTS OF PROPOSED ALGORITHM

Class	Sequence	△ WS_PSNR_Y(dB)	BD-rate_Y (%)	△ T(%)
A	AerialCity	-0.04	1.27%	37.3%
	PoleVault	-0.09	1.80%	36.1%
	DrivingInCountry	-0.03	1.43%	40.5%
	DrivingInCity	-0.03	1.14%	30.2%
B	Balboa	-0.04	1.59%	38.4%
	Broadway	-0.08	1.74%	42.3%
	Landing2	-0.02	1.25%	41.6%
	BranCastle2	-0.03	1.03%	42.1%
C	GasLamp	-0.01	0.97%	39.8%
	ChairliftRide	-0.03	1.21%	36.9%
	SkateboardInLot	-0.05	1.4%	32.1%
	Trolley	-0.01	0.93%	38.0%
Average		-0.04	1.31%	37.9%

V. CONCLUSION

In order to reduce the computational complexity of virtual reality video coding, this work proposes a fast algorithm to speed up the CU partition process based on KNN classifier. The classifier use edge information to predict the depth of LCU and terminate the CU partition process based on this depth in advance. Experimental results show that the proposed fast algorithm provides an average time reduction rate of 37.9% compared to the reference HM-16.16+360lib4.0, with only 1.31% BD-rate increase.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

AUTHOR CONTRIBUTIONS

All authors made substantial contributions to conception and design, analysis, and interpretation of data, and critical review of the manuscript. Zhi Liu carried out the idea and design of the study, optimized the algorithm, and wrote the manuscript. Peiran Song carried out the experiments, and performed the statistical analyses. Mengmeng Zhang got the funding for the study, and helped rewrite the manuscript substantially during the revision process.

REFERENCES

[1] S. E Alshina, J. Boyce, A. Abbas, and Y. Ye, "JVET common test conditions and evaluation procedures for 360° video," Joint Video Exploration Team of lTU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JVET-H1030 Macau, 2017.

[2] G. J. Sullivan, R. Ohm, and W. J. Han, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. 22, no. 12, pp. 1649-1668, 2012.

[3] K. Choi, S. H. Park, and E. S. Jang, "Coding tree pruning based CU early termination," *JCTVC-F092, JCT-VC of ISO/IEC and ITU-T*, Torino, Italy, July 2011.

[4] X. Shen and L. Yu, "CU splitting early termination based on weighted SVM," *EURASIP J. Image Video Process.*, vol. 2013, pp. 1-11, Jan. 2013.

[5] H. Poor, *An Introduction to Signal Detection and Estimation*, New York: Springer-Verlag, 1985, ch. 4.

[6] H. S. Kim and R. H. Park, "Fast CU partitioning algorithm for HEVC using an online-learning-based Bayesian decision rule," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 130-138, Jan. 2016.

[7] H. S. Kim and R. H. Park, "Fast CU partitioning algorithm for HEVC using an online-learning-based Bayesian decision rule," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 130-138, Jan. 2016.

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

[9] 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 Trans. Image Process.*, vol. 24, no. 7, pp. 2225-2238, Jul. 2015.

[10] K. Goswami, B. G. Kim, D. S. Jun, S. H. Jung, and J. S. Choi, "Early coding unit-splitting termination algorithm for High Efficiency Video Coding (HEVC)," *Electron. Telecommun. Res. Inst. J.*, vol. 36, no. 3, pp. 407-417, Jun. 2014.

[11] H. L. Tan, C. C. Ko, and S. Rahardja, "Fast coding quad-tree decisions using prediction residuals statistics for High Efficiency Video Coding (HEVC)," *IEEE Trans. Broadcast.*, vol. 62, no. 1, pp. 128-133, Mar. 2016.

Zhi Liu is a doctor of engineering, master instructor. He received the B.S. degree in electronic information technology and the Ph.D. in signal and information processing from Beijing Jiaotong University, China in 2001 and 2011 respectively. Currently, he is a lecturer in North China University of Technology. His major research interests include the video codec, pattern recognition, and self-organizing network.

Peiran Song is studying master of North China University of Technology. Her major research is HEVC.

Mengmeng Zhang is a doctor of engineering, professor, master instructor, master of Communication and Information Systems. His major research interests include the video codec, embedded systems, image processing, and pattern recognition. He has authored or co-authored more than 40 refereed technical papers in international journals and conferences in the field of video coding, image processing, and pattern recognition. He holds 21 national patents and 2 monographs in the areas of image/video coding and communications.