J Inf Process Syst, Vol.16, No.4, pp.959~974, August 2020
https://doi.org/10.3745/JIPS.04.0186

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Fast CU Size Decision Optimal Algorithm Based on Neighborhood Prediction for HEVC

Jianhua Wang[*,**,***], Haozhan Wang[*], Fujian Xu[*], Jun Liu[****], and Lianglun Cheng[*****]

Abstract

High efficiency video coding (HEVC) employs quadtree coding tree unit (CTU) structure to improve its coding efficiency, but at the same time, it also requires a very high computational complexity due to its exhaustive search processes for an optimal coding unit (CU) partition. With the aim of solving the problem, a fast CU size decision optimal algorithm based on neighborhood prediction is presented for HEVC in this paper. The contribution of this paper lies in the fact that we successfully use the partition information of neighborhood CUs in different depth to quickly determine the optimal partition mode for the current CU by neighborhood prediction technology, which can save much computational complexity for HEVC with negligible RD-rate (rate-distortion rate) performance loss. Specifically, in our scheme, we use the partition information of left, up, and left-up CUs to quickly predict the optimal partition mode for the current CU by neighborhood prediction technology, as a result, our proposed algorithm can effectively solve the problem above by reducing many unnecessary prediction and partition operations for HEVC. The simulation results show that our proposed fast CU size decision algorithm based on neighborhood prediction in this paper can reduce about 19.0% coding time, and only increase 0.102% BD-rate (Bjontegaard delta rate) compared with the standard reference software of HM16.1, thus improving the coding performance of HEVC.

Keywords

CU Decision, HEVC, Neighborhood Prediction, Optimal Algorithm

1. Introduction

High efficiency video coding (HEVC) [1] as one of the latest international video compression standard, have been jointly developed by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) [2,3]. It mainly users the quadtree structure based on coding tree unit (CTU) to achieve its high coding efficiency compared to previous video coding standards. It is reported that the compression efficiency of HEVC is two times better compared to previous H.264/AVC [4,5].

HEVC improves coding efficiency by using new structures, but at the same time, it also greatly adds its computational complexity because it needs to take lots of rate-distortion (RD) optimization process to obtain the optimal coding unit (CU) partition from all CU combinations [6]. However, the high compu-

Manuscript received April 26, 2018; first revision December 11, 2018; second revision December 26, 2019; accepted February 11, 2020.

Corresponding Author: Jianhua Wang (123chihua@163.com)

* College of Electronic Engineering, South China Agricultural University, Guangzhou, China (123chihua@163.com, 3508209360@qq.com, 1280137559@qq.com)

** National Center for International Collaboration Research on Precision Agricultural Aviation Pesticides Spraying Technology, Guangzhou, China (123chihua@163.com)

*** School of Computer Science and Engineering, Nanyang Technological University, Singapore (123chihua@163.com)

**** College of Automation, Guangdong Polytechnic Norma University, Guangzhou, China (liujun7700@163.com)

***** College of Automation, Guangdong University of Technology, Guangzhou, China (llcheng@gdut.edu.cn)

tational complexity is a very important problem in some embedded devices especially in some power constrained or real-time application devices. Therefore, it is very necessary to reduce the computational complexity by developing some efficient CU partition algorithms for HEVC with negligible degradation [7].

Recently, some new methods have been taken to reduce the computational complexity in CU size decision for HEVC. For example, a fast CU size decision algorithm based on early homogenous CUs termination [8], a fast CU size decision algorithm based on texture [9], a fast CU size determination algorithm based on adaptive discretization total variation threshold [10], a fast CU size partition algorithm based on data mining [11], and so on, have been proposed to reduce the high computational complexity for HEVC. Although these schemes above are well designed, but there is still a need to further develop more efficient schemes to greatly reduce the high computational complexity for HEVC.

Different from these methods above, in this paper, a fast CU size decision optimal algorithm based on neighborhood prediction is proposed to reduce its total computational complexity for HEVC by reducing lots of unnecessary prediction and partition operations. The contribution of this paper rests in the fact that we successfully make use of the partition information of neighborhood CUs in different depth to quickly determine its optimal partition mode for the current CU, which can reduce much computational complexity for HEVC. Specifically, in our scheme, we use the partition information of left, up, and left-up CUs to quickly determine the optimal partition mode for the current CU by using neighborhood prediction technology, as a result, our proposed algorithm can reduce lots of unnecessary prediction and partition operations, thus saving much computational complexity for HEVC. The simulation results show that our proposed fast CU size decision algorithm in this paper can reduce about 19.0% coding time, and only increase 0.102% BD-rate (Bjontegaard delta rate) compared with the standard reference software of HM16.1, thus saving much computational complexity for HEVC.

The remainder of the paper are organized as follows. In Section 2, the complexity analysis and related studies for CU size decision schemes are introduced. Our proposed method is presented in Section 3. The simulation results and analysis using our proposed scheme compared with some of the existing methods are presented in Section 4. In Section 5, we provide our conclusion.

2. The Complexity Analysis and Related Studies

2.1 Complexity Analysis

As we all know that HEVC uses recursive quadtree to divide a large coding unit (LCU) with 64×64 size, and it can eventually 85 small CUs after many partitioning and predicting operations. Fig. 1 is the partitioning process for a LCU with of 64×64 size. From the Fig. 1, we can see clearly that a LCU with a size of 64×64 can be partitioned into 4×0+4×1+4×2+4×3=85 small CUs by using its recursive quadtree structure. Since a PU is the basic prediction unit in HEVC, the partitioned number of PU is about $4^0+4^1+4^2+4^3+4^4=341$ in a LCU with size of 64×64. In HEVC, since the 85 small CUs need to independently execute 341×35=11935 times prediction and partition operations to find the optimal partition mode for the current CU with minimum rate-distortion cost (RDCost), which leads to high computational complexity for HEVC. So how to save the computational complexity by reducing many unnecessary prediction and partition operations become an important issue for us in HEVC in this paper.

2.2 Related Studies

In recent, many optimal decision algorithms of CU size have been proposed to reduce the computational or encoding complexity for HEVC.

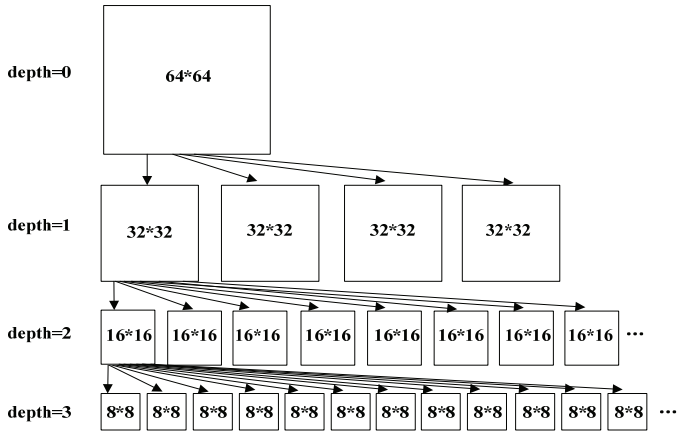


Fig. 1. Partitioning process for a LCU with 64×64 size.

Zhang et al. [8] proposed a fast CU size decision based on early homogenous CUs termination, which can reduce much computational complexity for HEVC. Ha et al. [9] presented a fast CU size decision algorithm based on texture, which can reduce lots of unnecessary computational complexity for HEVC. Zhou et al. [10] suggested an adaptive and perceptual CU size decision method based on visual saliency detection, which can reduce much encoding complexity for HEVC. Song et al. [11] proposed a fast CU size determination algorithm based on adaptive discretization total variation threshold, which can reduce much encoding complexity for HEVC. Ruiz et al. [12] presented a fast CU partition algorithm based on data mining, which can reduce much encoding complexity for HEVC. Cen et al. [13] proposed a fast CU depth decision mechanism based on adaptive CU depth range determination and comparison, which can reduce much encoding complexity for HEVC. Zhang et al. [14] proposed a fast and efficient CU size decision algorithm based on temporal and spatial correlation, which can reduce lots of encoding complexity for HEVC. In the paper [15], a fast CU size decision algorithm based on depth information of neighboring CUs is proposed to reduce its encoding complexity for HEVC. In [16], a fast CU size decision algorithm based on Bayesian theorem detection is presented to reduce the encoding complexity for HEVC. In the paper [17], a novel fast CU encoding scheme based on spatiotemporal encoding parameters is suggested to reduce its encoding complexity for HEVC. In [18], a fast CU size decision algorithm based on SKIP mode decision (SMD), CU skip estimation (CUSE), and early CU termination (ECUT) is developed to reduce its encoder complexity for HEVC. In the paper [19], a fast CU size selection based on the Bayesian decision rule is presented to reduce its whole encoder complexity for HEVC. Although these schemes above are well designed, but there is still a need to further develop more efficient schemes to reduce its high computational complexity for HEVC.

Different from these methods above, a fast CU size decision optimal algorithm based on neighborhood prediction is proposed to quickly determine its optimal CU partition mode for the current CU in this paper, which can reduce many unnecessary prediction and partition operations, thus saving much computational complexity for HEVC.

3. Proposed Scheme

3.1 Method Motivation

As we all know that there is a strong correlation between adjacent coding blocks of video images due to their successive texture characteristics. For example, for a coding block with size of 32×32, it can be divided into four coding blocks with 16×16 size in HEVC. If one or two in four coding blocks with 16×16 size above can be divided into multiple coding blocks with 8×8 size, the rest of coding blocks with 16×16 size are also possible to be divided into multiple coding blocks with 8×8 size. Table 1 is the average probability of statistical partition for a current CU when its one or more of its adjacent coding blocks are partitioned in different depth from 12 different HEVC official test sequences. The 12 different HEVC official test sequences include 6 class different resolutions: Class A (2560×1600), Class B (1920×1080), Class C (1280×720), Class D (1024×768), Class E (832×480) and Class F (416×240), and their frame rate and frame number are set to 25 and 100 in our experiment, respectively.

Table 1. Average statistical probability of partition for a current CU

Picture size	Sequences	Partition rate in difference depth		
		depth=1	depth=2	depth=3
Class A	PeopleOnstreet	0.7824	0.5136	0.3823
	Traffic	0.8216	0.4854	0.4058
Class B	BasketballDrive	0.7568	0.4627	0.3245
	Cactus	0.7456	0.5283	0.3163
Class C	RaceHorses	0.7624	0.4724	0.3824
	BQMall	0.7881	0.5162	0.3952
Class D	BasketballPass	0.8424	0.5309	0.3768
	BlowingBubbles	0.8257	0.4657	0.3676
Class E	Fourpeople	0.7972	0.4483	0.4057
	Johnny	0.8183	0.4206	0.3845
Class F	BasketballDrillText	0.7643	0.5587	0.3713
	Slideshow	0.7890	0.5491	0.3635

From Table 1, we can see that there is a strong partition correlation between adjacent CUs due to their successive texture characteristics of video images. Once the adjacent coding sub-blocks are segmented: in depth 1, the partition probability for the current CU is up to 79.1%; in depth 2, the partition probability is reduced to 49.5%; however, in depth 3, the partition probability is dropped to only 37.3%.

The motivation of our proposed scheme in this paper mainly comes from the statistics partition probability above in different depth on the basis of careful observation and analysis of statistics partition probability and standard partition algorithm of CU size in HEVC, and proposed a fast CU size decision optimal algorithm based on neighborhood prediction to quickly determine its optimal CU partition mode for the current CU by using relevant partition information between adjacent CUs in the same depth, which can reduce lots of unnecessary partition and prediction operations for the current CU, therefore saving much computational complexity for HEVC.

3.2 Realizing Principle

Since the standard intra prediction algorithm in HEVC does not take the relevant partition information

between adjacent CUs into account, and firstly it only independently executes prediction and partition operations for 85 CUs in a LCU, then it calculates their RDCost values, finally it saves the best way of partition mode, which needs to take lots of unnecessary CU partition and prediction operations in the realizing process, thereby leading to high computational complexity for HEVC.

The realizing principle for our proposed scheme is that we use the partition information of neighborhood CUs to quickly determine partition mode for the current CU by using neighborhood prediction technology. It can reduce many unnecessary prediction and partition operations, and then reduce lots of computational complexity for HEVC. Specifically, in our scheme, we mainly use the partition information of left, up, and left-up CUs to quickly determine whether the current CU in different depth needs to be further divided by its partition threshold value due to its strong correlation between adjacent CUs, which can reduce lots of unnecessary prediction and partition operations for the current CU, therefore saving much computational complexity for HEVC. Fig. 2 is the realizing principle of our proposed scheme.

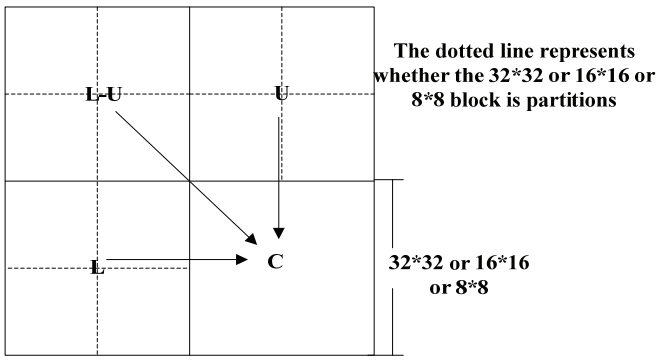


Fig. 2. Realizing principle of our proposed scheme.

Fig. 3 is the convergence figure for our proposed scheme. From Fig. 3, we can clearly see that, in our proposed method, we firstly read adjacent CUs for a current CU, then calculate their prediction, at last we determine to execute or terminate the operation of standard partition algorithm according to the prediction threshold value above.

Where Read its adjacent CUs mainly executes read operations for its adjacent CUs of the current CU. Calculate prediction threshold mainly executes the calculation operations for its adjacent CUs by proposed formula. Execute the standard partition algorithm mainly executes the operation of standard partition algorithm according to prediction threshold value. Terminate the standard partition algorithm mainly terminates the operation of standard partition algorithm according to its prediction threshold value.

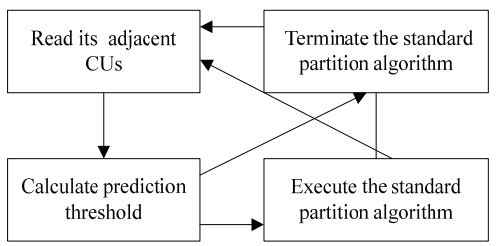


Fig. 3. Convergence figure for our proposed scheme.

3.3 Realizing Process

3.3.1 Define prediction threshold

From table1above, we can see that the subdivision correlation in depth 1 is stronger than in depth 2, and the subdivision correlation in depth 2 is stronger than in depth 3. And with the increase of depth, the partition correlation in adjacent CUs become worse and worse.

In order to realize to the principle above, in this paper, we firstly need to define a threshold value of prediction partition to determine whether the current CU needs to be partitioned according to calculating its correlation degree of partition from its adjacent CUs in different depth, which can be shown in the following formula:

$$T_{depth}=0.2\times b_{L\text{-}U}+0.4\times b_{L}+0.4\times b_{U} \tag{1}$$

where T_{depth} represents threshold value of prediction partition. b_{L-U} represents the left-up CUs of the current prediction CU. It is set to 1 when it is partitioned, otherwise it is set as 0. b_L represents the left CU of the current prediction CU. It is set to 1 when it is divided, otherwise it is set as 0. b_U represents the up CU of the current prediction CU. It is set to 1 when it is partitioned, otherwise it is set to 0. In this paper, the segmentation weigh coefficient of b_{L-U}, b_L, b_U for the current CU are obtained according to their traversal priority and importance degree with the current CU. Based on the principles above and combination with test results of kinds of different test sequences, the segmentation weigh coefficient for b_{L-U}, b_L, b_U are finally set to 0.2, 0.4, and 0.4, respectively, which can get better experimental results compared with other parameters in our experiment.

3.3.2 Design segmentation strategy

After defining the partition threshold, in our scheme, we need to design different segmentation strategy for different depth based on partition threshold to determine whether the current CU needs to be further partitioned. The specific segmentation strategies are shown as follows.

From the previous partition statistics in Table 1, we can see that, in depth 1, the statistical probability of partition for the current CU reaches about 80% when one or more of b_L, b_U, and b_{L-U} are partitioned. This shows that once one of the b_L, b_U, and b_{L-U} is partitioned, the partition probability for the current CU is very high. So in depth 1, we designed our segmentation strategy as followed:

The current CU will execute partition operations when one or more of b_L, b_U, and $b_{L-U}$are partitioned. Otherwise it will not execute the partition operations. So in our scheme of this paper, we can set 0 as our partition threshold value according to its segmentation requirements above, and the segmentation strategy can be expressed the following formula.

$$\begin{cases} b_c=1, when T_{depth=1}>0 \\ b_c=0, when T_{depth=1}=0 \end{cases} \tag{2}$$

where b_c represents the current prediction CU block. $b_c = 1$ represents that the current prediction CU block to be partitioned. $b_c = 0$ stands for that the current prediction CU block to not be partitioned.

From the previous partition statistics in Table 1, we can also see that, in depth 2, once one of b_L, b_U,

and b_{L-U} are partitioned, the statistical partition probability for the current CU is about 50%. The partition probability is higher than 50% when two or more of b_L, b_U, and b_{L-U} are partitioned. So in depth 1, we designed our segmentation strategy as followed:

The current CU will execute the partition operations when one or more of b_L, b_U, and b_{L-U} are partitioned; Otherwise it will not execute the partition operations. In order to realize the segmentation requirements above, we set 0.3 as our partition threshold in our scheme of this paper, and the partition strategy can be expressed as follows:

$$\begin{cases} b_c = 1, when T_{depth=2} > 0.3 \\ b_c = 0, when T_{depth=2} < 0.3 \end{cases} \tag{3}$$

where b_c represents the current prediction CU block. $b_c = 1$ stands for the current prediction CU block to be partitioned. $b_c = 0$ represents the current prediction CU block not to be partitioned.

When depth is 3, since the partition probability for the current CU with Size N×N mode is only about 38% when one of the b_L, b_U, and b_{L-U} uses the Size N×N mode as its optimal partition mode. So in this paper, the segmentation strategy is designed as followed:

The current CU will execute the partition operations by using Size N×N mode when two or more of b_L, b_U, and b_{L-U} are partitioned by using N×N mode. Otherwise it will not execute the partition operations with N×N mode. In order to meet the conditions above, 0.5 is set as our partition threshold in our scheme of this paper, and the segmentation strategy can be expressed as follows:

$$\begin{cases} b_c = 1, when T_{depth=3} > 0.5 \\ b_c = 0, when T_{depth=3} \le 0.5 \end{cases} \tag{4}$$

where b_c stands for the current prediction CU block. $b_c = 1$ represents that the current prediction CU block needs to be partitioned. $b_c = 0$ represents that the current prediction CU block does not need to be partitioned.

Note that, in depth 0, since the statistical partition probability is very high when one or more of b_L, b_U, and b_{L-U} are partitioned, almost reaching 1, we do not consider to optimize it in this paper due to its high partition probability. In depth 3, since the smallest 8×8 CU has 2 prediction modes (PART_N×N and PART_2N×2N), we only consider whether the current CU uses the prediction mode PART_N×N by judging whether its neighborhood CUs uses the prediction mode PART_N×N as their optimal prediction modes when it requires the same CU size. In our segmentation strategy, we also further defined that if only b_{L-U} are partitioned, the current CU does not execute the partition operations.

3.3.3 Design realizing process

Fig. 4 is the realizing process for our proposed scheme. From Fig. 4, we can see that our suggested scheme mainly includes the following sit contents: Obtain the coding CU, Judge the depth of CU, Read its adjacent CUs, Calculate prediction threshold, Compare prediction threshold and partition threshold, and Execute/Terminate the standard partition algorithm.

Where Obtain the coding CU mainly executes the obtain operations for the current coding CU. Judge the depth of CU is mainly to judge in which depth for the current CU locates. Read its adjacent CUs

mainly executes read operations for its adjacent CUs. Calculate prediction threshold mainly executes the calculation operations for its adjacent CUs by formula (1). Compare prediction threshold and partition threshold mainly compares the difference value between prediction threshold and partition threshold. Execute/Terminate the standard partition algorithm mainly executes or terminates the operations of the standard partition algorithm according to the difference value between prediction threshold and partition threshold.

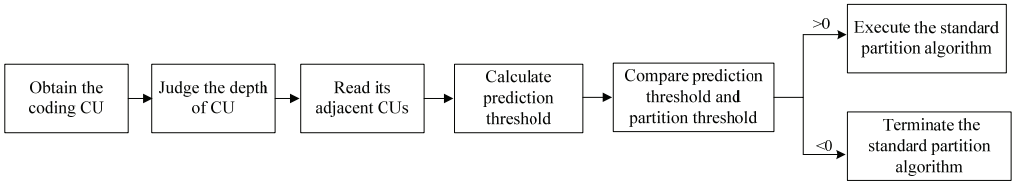


Fig. 4. Realizing process for our proposed scheme.

3.4 Realizing Steps

Since the division of LCU in HEVC is adopted by recursion quadtree, our proposed method in this paper mainly optimizes the CU partitioning process by depth order from large to small. The detailed realizing steps for our proposed scheme can be summarized into the following steps:

Step 1: Obtain the current prediction CU and its depth;

Step 2: Read its partition mode of b_L, b_U, and b_{L-U} of the current CU in depth 3, and judge whether it uses the partition mode PART_N×N as its optimal partition mode. Then it calculates the prediction threshold by using formula (1), and compare the difference value between prediction threshold and partition threshold by using formula (4) above. If the difference value is greater than 0, the current CU needs to be further divided with PART_N×N mode. Otherwise it does not need to be further partitioned.

Step 3: Calculate the total RdCost value for b_L, b_U, and b_{L-U} and the current CU, and compare it with the whole coding block constituted by them in depth 3 to judge whether the whole coding block needs to be further partitioned. The whole coding block will execute the partition operations when the former is larger than the latter. Otherwise it does not execute the partitions operations.

Step 4: Execute the partition strategy to calculate for b_L and b_U by taking the same step 2 and step 3 above, then get their partition mode information and save them respectively.

Step 5: Obtain the partition mode for b_L, b_U, and b_{L-U} in depth 2, then calculate their prediction threshold values by using formula (1) and compare their difference value between prediction threshold and partition threshold by using formula (3) above. If the difference value is greater than 0, the current prediction block needs to be further partitioned. Otherwise it does not need to be partitioned.

Step 6: Calculate the total RdCost value for b_L, b_U, and b_{L-U} and the current CU, and compare it with the whole coding block constituted by them in depth 2 to judge whether the whole coding block needs to be further partitioned. The whole coding block will execute the partition operations when the former is greater than the latter. Otherwise it will not execute the partitions operations.

Step 7: Take the same partition operations for b_L and b_U by taking the same step 2 to step 6, then calculate and save their partition mode information.

Step 8: Calculate the partition mode information for b_L, b_U, and b_{L-U}, then calculate their prediction threshold value by using formula (1), then compare their difference value between prediction threshold and partition threshold by using formula (2) above. If the difference value is greater than 0, the current prediction block needs to be further partitioned. Otherwise it does not need to be partitioned.

Step 9: Calculate the total RdCost value for b_L, b_U, and b_{L-U} and the current CU, and compare it with the whole coding block constituted by them in depth 0 to judge whether the whole coding block needs to be further partition. It will execute the partition operations when the former is greater than the latter. Otherwise it will not execute the partition operations.

3.5 Application Instance

Take partitioning and determining its optimal partition mode from a LCU, for example, to illustrate the detailed realizing process for our proposed scheme, which is shown in Fig. 5.

Step 1: Obtain a current prediction CU and its depth;

Step 2: Read its partition mode information of b_LB, b_UC and b_{L-U}A from the current CU D in depth 3, and judge whether B, C and A coding CU uses the partition mode PART_N×N as their optimal predicted mode. Then it will calculate the prediction threshold by using formula (1), and compare the difference value between prediction threshold and partition threshold by using formula (4) above. If the difference value is greater than 0, the current CU does not need to be further partitioned. Otherwise it needs to be further partitioned with PART_N×N mode.

Step 3: Calculate the total RdCost value for b_LB, b_UC and b_{L-U}A and the current CU D, and compare it with the whole coding block E constituted by them to judge whether the whole coding block E needs to be further partitioned. The whole coding block E will needs to be partitioned when the former is smaller than the latter. Otherwise it does not need to be partitioned.

Step 4: Execute the partition strategy to calculate for b_LF and b_UG by taking the same step 2 and step 3 above, then get and save their partition information.

Step 5: Obtain the partition mode information for b_LF, b_UG, b_{L-U}E in depth 2, then calculate their prediction threshold value by using formula (1) and compare their difference value between prediction threshold and partition threshold by using formula (3) above. If the difference value is greater than 0, the current CU H needs to be further partitioned. Otherwise it does not need to be further partitioned.

Step 6: Calculate the total RDCost value for b_LF, b_UG, b_{L-U}E and the current CU H, and compare it with their whole coding block I constituted by them in depth 2 to judge whether the whole coding block I needs to be further partitioned. The coding block I will execute the partition operations when the former is greater than the latter. Otherwise it will not execute the partitions operations.

Step 7: Execute the same partition operations for b_LJ and b_UK by using the same step 2 to step 6 above, then calculate and save their partitions information.

Step 8: Calculate the partition mode information for b_LJ, b_UK, b_{L-U}I in depth 1, then calculate their prediction threshold value by using formula (1), and compare their difference value between prediction threshold and partition threshold by using formula (2) above. If the difference value

is greater than 0, the current prediction block L needs to be further partitioned; Otherwise it does not need to be partitioned.

Step 9: Calculate the total RDCost value for b_LJ, b_UK, b_{L-U}I and the current CU L, and compare it with the whole coding block LCU in depth 0 to judge whether the LCU needs to be further partitioned. The LCU will execute the partition operations when the former is greater than the latter. Otherwise it will not execute the partition operations.

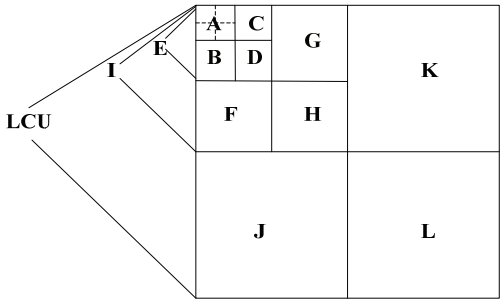


Fig. 5. Fast CU size partition optimal instance with our proposed scheme.

4. Experimental Results and Analysis

In order to verify the effectiveness of our proposed algorithm above, some experiments are performed in this section. In our experiment, our designed experiments mainly include the following two contents: building the experimental environment and testing the experimental results.

4.1 Building the Experimental Environment

The simulation environment was conducted on an Intel 2-GHz processors, 2-GB memory capacity, Intel Windows XP operating systems. The test conditions, configurations and test sequences in our experiment are built based on the JCT-VC [20]. We use 12 different official test sequences of HEVC above with six different class resolutions: Class A (2560×1600), Class B (1920×1080), Class C (1280×720), Class D (1024×768), Class E (832×480), and Class F (416×240) as our experimental test objects. And their frame rate and frame number are set to 25 and 100, respectively. Other some general configuration parameters are shown at Table 2.

Table 2. General configuration parameters

Profile	Main
GOP (group of pictures) structure	Low delay, Random access
Max CU size	64×64
Max CU depth	4
Motion search mode	TZsearch
Motion search range	64
QP (quantization parameter)	22, 27, 32, 37
Encoder.cfg	encoder_intra_rnain.cfg
Video frame	I frame
Reference Software Profile[21]	HM16.1

In our experiment, BD-rate performance and coding time saving rate are taken as our comparison indicators, where BD-rate [21] is mainly used to measure the loss of coding efficiency; coding time saving rate is used to measure the reduction rate of computational complexity for HEVC. The less coding time saving rate represents the higher computational complexity efficiency for HEVC. Two important statistical performances, average and STD, are introduced into our experiment to evaluate the performance of our method. Where the average represents the average value of BD-rate performance and coding time saving rate by repeated random experiments in our experiments, and it can reflect the concentration trend of different methods under the same conditions; STD represents the standard deviation of BD-rate performance and coding time saving rate by repeated experiments in our experiments, and it can evaluate the robustness of different methods under the same conditions. The coding time saving rate can be expressed as follows:

$$Coding\ time\ saving\ rate = \frac{\left|(coding\ time_{HM16.1)} - coding\ time_{our})\right|}{coding\ time_{HM16.1)}} \times 100\% \tag{5}$$

where coding $time_{HM16.1}$ and coding $time_{our}$ represent the total encoding decision time of the standard reference software of HM16.1 and our proposed algorithm in this paper, respectively.

In our experiment, two general comparison schemes, the standard reference HM scheme [22] and Ha's scheme [9] are selected to compare with our proposed method from two aspects, BD-rate performance and coding time saving rate. Where the standard reference HM scheme mainly used neighborhood correlation of neighborhood CUs to quickly determine the optimal partition mode for the current CU; Ha's scheme mainly used texture similarity of neighborhood CUs to quickly determine the optimal partition mode for the current CU, while our proposed method mainly used neighborhood prediction of neighborhood CUs in different depth with prediction threshold value to quickly determine the optimal partition mode for the current CU.

4.2 Testing the Experimental Results

In this subsection, we mainly test the performance of coding time saving rate and BD-rate increase compared with the standard reference HM scheme [22] and Ha's scheme [9]. The related experimental results are shown in Tables 3 and 4.

Table 3 is the coding time saving rate and BD-rate in different test sequences with QP=27. Table 4 is the coding time saving rate and BD-rate in different test sequences with QP=37. From Tables 3 and 4, we can see clearly that our proposed method in this paper have good perform in coding time compared with the standard reference software of HM16.1 under the same test conditions, reaching about 19.0% coding time saving, and only adding 0.102% BD-rate. Ha's scheme has the similar good perform in coding time, reaching about 20.0% coding time saving, but it needs to add about 2.06% BD-rate. The standard reference software of HM16.1presents the worst coding performance in three schemes above. From Tables 3 and 4, we can also see clearly that our proposed method in this paper have good perform of STD value in BD-rate performance and coding time saving rate compared with the Ha's scheme under the same test conditions. Fig. 6 is the coding time saving rate for Traffic test sequence with different QP value. Fig. 7 is the coding time saving rate for BQMal test sequence with different QP value. From Figs. 6 and 7, we can also see that our proposed method in this paper has good coding time saving rate in the different QP values compared to the other two methods.

Table 3. Time saving rate in different test sequences with QP=27

Picture class	Sequence name	Coding time saving (%)		BD-rate (%)	
		Our proposed	Ha's scheme	Our proposed	Ha's scheme
Class A	PeopleOnstreet	-17.73	-19.24	+1.22	+1.96
	Traffic	-15.98	-21.35	+1.01	+2.12
Class B	BasketballDrive	-22.84	-22.26	+0.99	+1.87
	Cactus	-21.67	-24.83	+1.09	+1.77
Class C	RaceHorses	-18.42	-17.58	+0.78	+1.96
	BQMall	-16.65	-19.36	+0.86	+2.08
Class D	BasketballPass	-17.84	-20.57	+0.94	+1.94
	BlowingBubbles	-18.63	-19.83	+1.11	+1.80
Class E	Fourpeople	-17.58	-17.74	+1.34	+2.26
	Johnny	-17.61	-16.68	+0.96	+1.93
Class F	BasketballDrillText	-18.24	-22.34	+1.32	+1.84
	Slideshow	-19.56	-18.79	+1.27	+1.99
Average		-18.57	-20.05	+1.02	+1.85
STD		-19.64	-22.45	+0.18	+0.20

Table 4. Time saving rate in different test sequences with QP=37

Picture class	Sequence name	Coding time saving (%)		BD-rate (%)	
		Our proposed	Ha's scheme	Our proposed	Ha's scheme
Class A	PeopleOnstreet	-17.23	-18.94	+1.14	+2.02
	Traffic	-17.70	-18.47	+0.96	+2.28
Class B	BasketballDrive	-22.24	-23.34	+1.07	+1.95
	Cactus	-21.87	-24.55	+1.01	+1.82
Class C	RaceHorses	-22.28	-21.67	+0.87	+1.89
	BQMall	-18.81	-20.83	+0.85	+2.03
Class D	BasketballPass	-17.35	-17.22	+0.78	+1.99
	BlowingBubbles	-18.38	-20.47	+1.03	+1.76
Class E	Fourpeople	-17.03	-16.65	+1.24	+2.23
	Johnny	-17.27	-19.48	+0.89	+2.05
Class F	BasketballDrillText	-18.76	-21.82	+1.21	+1.87
	Slideshow	-19.49	-20.34	+1.32	+2.10
Average		-19.03	-20.32	+0.91	+2.06
STD		-20.01	-22.44	+0.16	+0.20

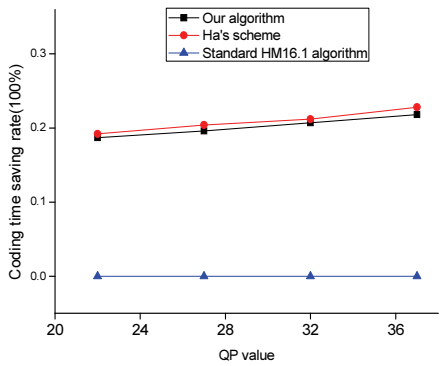


Fig. 6. Coding time saving rate for Traffic sequence.

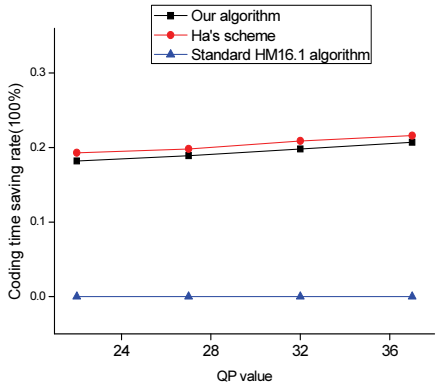


Fig. 7. Coding time saving rate for BQMal sequence.

The main reasons for them are that, in our proposed scheme, we mainly use the relevant partition information between neighbor CUs in different depth to quickly determine the optimal partition mode for the current CU, which can reduce many unnecessary prediction and partition operations for the current CU, therefore saving about 19% computational complexity and only adding about 0.102% BD-rate for HEVC. Ha's scheme can save about 20% computational complexity due to the use of texture-based technology in its scheme, which can reduce many unnecessary computations for the current CU due to texture similarity, thus saving much computational complexity for HEVC. But at the same time, it needs to add about 2.06% BD-rate for HEVC. The standard reference software of HM16.1 costing the most coding time lies in its independent prediction and partition operations for each CU to find its optical partition mode, which needs to take many unnecessary CU partition and prediction operations, thereby leading to the highest computational complexity for HEVC.

From Figs. 6 and 7, we can also observe that the coding time saving rate presents to slightly rise with the increase of QP value in our proposed algorithm and Ha's scheme, which shows that the two schemes can save much more coding time in high QP value compared to lower QP value under the same test condition. The similar results can be also found in the Ha's scheme [9]. The main reason for it is that Ha's scheme and our proposed scheme have a different impact on different QP value, thus influencing different encoding time. However, the standard reference software of HM16.1 does not present the phenomenon above.

5. Conclusion and Future Work

In this paper, a fast CU size decision optimal algorithm based on neighborhood prediction is proposed for HEVC intra prediction. In our scheme, we mainly use the relevant information between neighboring CUs to quickly determine the optimal partition mode for the current CU with neighborhood prediction technology, which can reduce many unnecessary partition and prediction operations, thereby saving much computational complexity for HEVC. The simulation results show that our proposed fast CU size decision algorithm in this paper can achieve about 19.0% computational complexity, while incurring only 0.102% increase on BD-rate compared with the original standard reference software of HM16.1 in HEVC.

We plan to continue our future research in the following two directions. Firstly, we will extend our proposed method to suit the processing of inter prediction. Secondly, we will incorporate Bayes' theorem and machine learning of artificial intelligence (AI) in our proposed to test its whole coding performance.

Acknowledgement

The work was supported by the National Natural Science Foundation of China (No. 61602187, No. 6180405), the National Key Research and Development Plan (No. 2016YFD0200700), the Guangdong Science and Technology Projects (No. 2019B020219002), and Guangdong Laboratory of Lingnan Modern Agriculture.

References

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

[2] W. J. Han, J. Min, I. K. Kim, E. Alshina, A. Alshin, T. Lee, et al., "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1709-1720, 2010.

[3] G. J. Sullivan and J. R. Ohm, "Recent developments in standardization of high efficiency video coding (HEVC)," in *Proceedings of SPIE 7798: Applications of Digital Image Processing XXXIII*. Bellingham, WA: International Society for Optics and Photonics, 2010.

[4] J. Vanne, M. Viitanen, and T. D. Hamalainen, "Efficient mode decision schemes for HEVC inter prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 9, pp. 1579-1593, 2004.

[5] Y. Lu, Q. Zhang, and B. Wei, "Real-time CPU based H. 265/HEVC encoding solution with x86 platform technology," in *Proceedings of 2015 International Conference on Computing, Networking and Communications (ICNC)*, Garden Grove, CA, 2015, pp. 418-421.

[6] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: the new gold standard for video compression: how does HEVC compare with H. 264/AVC?," *IEEE Consumer Electronics Magazine*, vol. 1, no. 3, pp. 36-46, 2012.

[7] Joint Collaborative Team on Video Coding (JCT-VC), "Architectural outline of proposed High Efficiency Video Coding (HEVC) design elements," document JCTVC-A202, Dresden, Germany, 2010.

[8] T. Zhang, M. T. Sun, D. Zhao, and W. Gao, "Fast intra-mode and CU size decision for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1714-1726, 2016.

[9] J. M. Ha, J. H. Bae, and M. H. Sunwoo, "Texture-based fast CU size decision algorithm for HEVC intra coding," in *Proceedings of 2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Jeju, South Korea, 2016, pp. 702-705.

[10] X. Zhou, G. Shi, and W. Zhou, "Perceptual CU size decision and fast prediction mode decision algorithm for HEVC intra coding," in *Proceedings of 2016 IEEE International Symposium on Multimedia (ISM)*, San Jose, CA, 2016, pp. 375-378.

[11] Y. Song, Y. Zeng, X. Li, B. Cai, and G. Yang, "Fast CU size decision and mode decision algorithm for intra prediction in HEVC," *Multimedia Tools and Applications*, vol. 76, no. 2, pp. 2001-2017, 2017.

[12] D. Ruiz, G. Fernandez-Escribano, V. Adzic, H. Kalva, J. L. Martinez, and P. Cuenca, "Fast CU partitioning algorithm for HEVC intra coding using data mining," *Multimedia Tools and Applications*, vol. 76, no. 1, pp. 861-894, 2017.

[13] Y. F. Cen, W. L. Wang, and X. W. Yao, "A fast CU depth decision mechanism for HEVC," *Information Processing Letters*, vol. 115, no. 9, pp. 719-724, 2015.

[14] Q. Zhang, J. Zhao, X. Huang, and Y. Gan, "A fast and efficient coding unit size decision algorithm based on temporal and spatial correlation," *Optik*, vol. 126, no. 21, pp. 2793-2798, 2015.

[15] X. Shang, G. Wang, T. Fan, and Y. Li, "Fast CU size decision and PU mode decision algorithm in HEVC intra coding," in *Proceedings of 2015 IEEE International Conference on Image Processing (ICIP)*, Quebec City, Canada, 2015, pp. 1593-1597.

[16] L. Shen, Z. Zhang, X. Zhang, P. An, and Z. Liu, "Fast TU size decision algorithm for HEVC encoders using Bayesian theorem detection," *Signal Processing: Image Communication*, vol. 32, pp. 121-128, 2015.

[17] S. Ahn, B. Lee, and M. Kim, "A novel fast CU encoding scheme based on spatiotemporal encoding parameters for HEVC inter coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 422-435, 2015.

[18] 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.

[19] X. Shen, L. Yu, and J. Chen, "Fast coding unit size selection for HEVC based on Bayesian decision rule," in *Proceedings of 2012 Picture Coding Symposium*, Krakow, Poland, 2012, pp. 453-456.

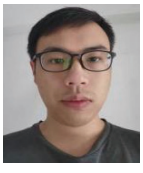
[20] F. Bossen, "Common Test Conditions and Software Reference Configurations," document JCTVC-G1200, Geneva, Switzerland, 2011.

[21] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," document VCEG-M33, Austin, TX, 2001.

[22] Joint Collaborative Team on Video Coding (JCT-VC), "Subversion Repository for the HEVC Test Model Version HM16.1," [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.1/.

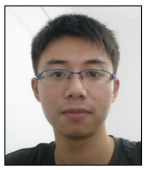
Jianhua Wang **https://orcid.org/0000-0001-9587-2845**

was born on February 6, 1982 in Guangdong, China. He received his B.S degree in Electronic Information Science and Technology from Shaoguan University, Guangdong, China, in 2006. He received his Ph.D degree in Control Science and Engineering at Guangdong University of Technology, Guangdong, China, in 2015. Currently he is a teacher of college of electronic engineering, south china agricultural university, Guangzhou, China. His research interests include wireless video transmission, cyber-physical systems and IoT.



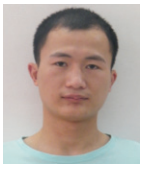
Haozhan Wang **https://orcid.org/0000-0001-9193-1158**

was born on June 6, 1997 in Guangdong, China. He received his B.Eng degree in software engineering from Guangdong University of Science and Technology, Guangdong China, in 2019. Currently, he studying for a master's degree in the school of electronic engineering, South China Agricultural University. The current research direction is intelligent information processing.



Fujian Xu **https://orcid.org/0000-0002-5653-8606**

was born on October 29, 199in JiangSu, China. He received his B.Eng degree in computer science and technology from Huaiyin Institute of Technology, JiangSu China, in 2017. Currently, he studying for a master's degree in the school of electronic engineering, South China Agricultural University. The current research direction is remote sensing image processing.



Jun Liu **https://orcid.org/0000-0002-5361-4247**

was born on October 11, 1986 in Hubei, China. He received his M.S degree in Control Science and Engineering from Guangdong University of Technology, Guangdong, China, in 2012. Currently he is a teacher of College of Automation, Guangdong Polytechnic Norma University, Guangzhou, China. His research interests include wireless sensor networks and cyber-physical systems.

Lianglun Cheng **https://orcid.org/0000-0003-1851-7473**

was born on August 22, 1964 in Hubei. He received his M.S and Ph.D degrees from Huazhong University of Science and Technology, Hubei, China in 1992 and Chinese academy of Sciences Jilin, china in 1999 respectively. He is a Prof and doctoral supervisor of Guangdong University of Technology. His research interests include RFID and WSN, IoT and CPS, production equipment and automation of the production process, embedded system, the complex system modeling and its optimization control, software of automation and information, etc.