

A New Optimization Algorithm for HEVC

Hongcheng Liu^{1(✉)}, Hang Lei¹, Yunbo Rao¹, Ping Chen²,
and Jun Shao²

¹ School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, People's Republic of China
liuhongcheng667@qq.com

² Digital Home Industry Technology Research Institute of Sichuan Hong Dian, Chengdu 610041, People's Republic of China

Abstract. In this paper, we propose a new optimization algorithm utilizing the texture feature and correlation of adjacent frames for newly proposed video standard High Efficiency Video Coding (HEVC). For intra prediction, the complexity of pictures' texture to perform different levels of simplification on Most Probable Mode (MPM) selection is scaled. For inter prediction, current Coding Unit (CU) depth information with that information of temporally adjacent frame's co-located CU is initialized. Experimental results show that the proposed algorithm improves the efficiency of encoder with more than 30 percent of encoding time decrease and nearly negligible increment in bit-rate.

Keywords: HEVC · Texture feature · Adjacent correlation

1 Introduction

The previous standard H.264/AVC achieves superior compression performance for ordinary definition video sequences. And HEVC standard [1] comes up to increase the efficiency when coding high definition video contents. However, concerns about high computational complexity that HEVC standard's tedious encoding process bring raise in real-time applications.

As the main process of encoding, prediction process of intra is quite different from inter. Unlike inter prediction constructing prediction units through motion estimation and motion compensation [2], intra prediction traverses all the prediction modes to pick out the best one to build blocks which are used in subsequent process. In order to reduce the complexity that traversing all the intra prediction modes brings, MPM is used to pre-processing all the modes using STAD [3] model in HEVC, it can reduce the range of candidate modes. In this paper, R-D cost [4] in STAD model is calculated as

$$J_{mode} = SATD(SAD) \lambda \times B_{pred} \quad (1)$$

where J_{mode} denotes the R-D cost of current mode, SATD denotes the sum of absolute values of the value from the Hadamard transform of distortions between prediction samples and original samples [5]. λ denotes a multiplier corresponding to Quantization Parameter (QP) and Picture Order Count (POC) of the slice current CU located in. B_{pred} indicates the number of bits cost when coding frame index.

During the R-D cost calculate process described above, an array *RdModeList* is created to temporally store all the modes in STAD ascending order. According to the size of Prediction Unit(PU), the first 3 or 8 optimal modes in this array will be selected as the first part of final candidate set. Then MPM process gets the two prediction modes of PUs locating on the left and upside of current CU, and adds the two modes to the final candidate set if they are not the same as the modes that already be selected. So the final set of MPM is the fusion of the first 3 or 8 optimal candidates in *RdModeList* and prediction modes of two spatial neighbors.

Slice include three components: I Slice, P Slice, and B Slice. Slice is divided into numbers of CTU whose size is defined to be 64×64 . A CTU will be recursively divided into four sub-CUs until its best splitting judged by R-D cost. Basic unit used for carrying the information in the prediction process is called PU. For inter prediction process, CU may be predicted with one or more PUs depending on 8 different partition modes. The one with the minimum R-D cost is chosen as optimum mode, which is calculated as

$$J_{mode} = SSD_{luma} + w_{chroma} \times SSD_{chroma} + \lambda \times B_{mode} \quad (2)$$

where SSD_{luma} and SSD_{chroma} denote the Sum of Squared Differences (SSD) between the original blocks and the reconstructed blocks of the luma and chroma parts, respectively. w_{chroma} is the weighting factor. B_{mode} specifies the bit cost of mode decision, and λ denotes the Lagrangian multiplier.

The CU size decision is recursively performed on the quad-tree. We denote a CU in the depth i as CU_i . That means CU_0 is the CTU, root of the quad-tree. CUs with the size $2N \times 2N$, $N = 8; 16; 32$ can be divided into four $N \times N$ CUs. For each CU size, to judge whether a CU should be split or not, R-D costs are calculated in situation where CU is un-split and split, respectively. The final R-D cost of CU_i is calculated as

$$J_{min}(CU_i) = \min\left(J(CU_i)_{un-split}, J(CU_i)_{split}\right) \quad (3)$$

where $J_{min}(CU_i)$ denotes the minimal R-D cost of CU_i . $J(CU_i)_{un-split}$ and $J(CU_i)_{split}$ denote the R-D costs of current CU encoded in the un-split manner and the split manner, respectively. $J(CU_i)_{split}$ is the sum of R-D costs of the four sub-CUs or PUs in the depth $i + 1$. During the splitting process of CTU, RDO technique described above is performed recursively until all parts of a CTU find out optimum splitting with the minimal R-D cost.

According to analysis described above, the proposed optimization algorithm includes two parts: (1) Detailed simplified MPM algorithm cut down the number of candidate modes for intra. (2) Simplified CU splitting algorithm achieves a comprehensive increase in coding efficiency with assistance modification of QP relationship and Reference Frame Selection (RFS) mechanism for inter.

2 The Proposed Algorithm

2.1 Proposed MPM Algorithm

HEVC performs 35 intra prediction modes including DC mode, planar mode and 33 angular modes. Among 35 intra prediction modes, DC and planar work well when predicting pictures with abundant smooth areas. If a candidate mode is equal to DC or planar, this partition corresponding to the candidate mode is supposed to be smooth. The prediction directions are irregular and angles between them are likely to be quite wide.

In this paper, complexity level is defined to scale texture features of CUs, four parameters consisting of first elements of STAD ascending ordered array *RdModeList* denoted by $R [0]$ and $R [1]$, and two spatial adjacent CUs' prediction modes denoted by *left* and *above*. Figure 1 shows the complexity level process of current CU.

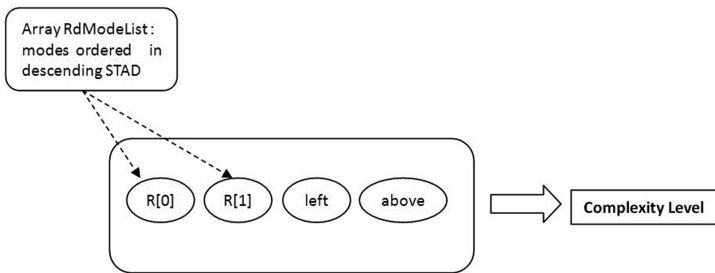


Fig. 1. Complexity level decision process.

The complexity level decided by four parameters directly decides complexity level of current frame. According to the number of candidate modes selected from array *RdModeList*, optimized algorithm performs detailed different cutting down on the candidate modes collection of each complexity level. To guarantee that proposed algorithm dose not reduce prediction accuracy compared with previous algorithm, number of modes in optimized candidate collection for every level is classified in detail with plenty of experiments.

If any of these four parameters is equal to DC or planar mode, the proposed algorithm classifies current CU into smooth texture complexity level, ranges from level 0 to level 2, and perform DC and Planar mode to be the major part of final candidates collection.

In other cases when all the parameters are equal to angular mode, current CU is classified into complex texture level, ranges from 3 to 5. To scale the CUs of complex texture, we select the first two elements of MPM candidates array in STAD ascending order as Group 1. When the two modes in Group 1 are all angular mode, the difference between two modes' index indicates the angle between two modes' prediction direction. We try to figure out the probability distribution when threshold value for mode index diffidence M is valued as 5,8,12 and 16, which corresponds to different angles. This experiment is conducted on CUs selected from video sequence named BasketballDrill, FourPeople, and BasketballDrive. These CUs are divided into two parts, half

of them whose average splitting depth is not smaller than 2.5 as Part 1 and the other whose average splitting depth ranges from 1.5 to 2.0 as Part 2, denoting normal and complex texture, respectively. The result is given in Table 1.

Table 1. Threshold selection text result

Depth condition	Avg_depth ≥ 2.5		1.5 ≤ Avg_depth ≤ 2.5	
	P(ΔM < TH)	P(ΔM ≥ TH)	P(ΔM < TH)	P(ΔM ≥ TH)
5	11.58 %	88.42 %	67.22 %	32.78 %
8	13.36 %	86.64 %	85.64 %	14.46 %
12	29.02 %	71.98 %	86.46 %	13.54 %
16	48.20 %	51.80 %	87.02 %	12.98 %

According to the results in Table 1, probability distribution indicates that when the threshold value for difference between two compared modes' index is set to be 8, it achieves a balance in distinguishing normal and complex textured CUs compared to 5, 12 and 16. And the threshold 8 for index different corresponds to 45 degree for angle according to the mapping of intra angular mode index and prediction direction. In this paper 45 degree is chosen as the threshold angle to distinguish normal and complex texture for CUs, as is shown in Fig. 2.

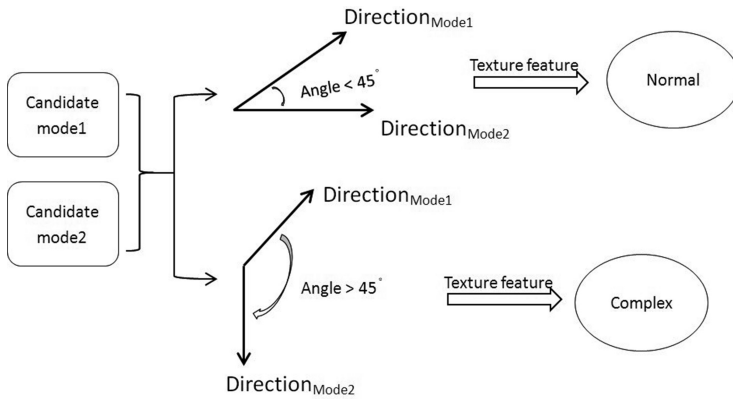


Fig. 2. Normal and complex texture feature judgment rules.

The difference between $R[0]$ denoting $RdModeList[0]$ and $R[1]$ denoting $RdModeList[1]$ and the difference between $left_mode$ and $above_mode$ will be calculated to denote the angle between two angular modes' prediction direction. If that difference is bigger than the specific threshold value described above, current encoding CU corresponding to these modes being compared will be judged as a higher access to complexity level. Six complexity levels in proposed algorithm are described in Table 2.

Table 2. Complexity level decision standard table

Level	R [0]	R [1]	Left	Above	MPMs	
					NumCandidates = 3	NumCandidates = 8
0	$\text{Num}_{\text{DC}} + \text{Num}_{\text{Planar}} \geq 3$				Planar, DC, R[0]	
1	$\text{Num}_{\text{DC}} + \text{Num}_{\text{Planar}} = 2$				DC, Planar, R[0,1]	DC, Planar, R[0,1,2,3]
2	$\text{Num}_{\text{DC}} + \text{Num}_{\text{Planar}} = 1$				DC, Planar, R[0,1,2]	DC, Planar, R[0,1,2,3,4,5]
3	R[0]-R[1] \geq Threshold left-above \geq Threshold				NC	
4	left-above $<$ Threshold				L,A,L(N), A(N),R[0]	L,A,L(N), A(N),R[0,1]
5	R[0]-R[1] $<$ Threshold left-above \geq Threshold				L,A, R[0](N),R[0]	L,A,R[0](N), R[1](N),R[0,1]

In Table 2, *level* denotes the complexity serial number. If a large serial number indicates a more complex hierarchy. *R[0]*, *R[1]*, *left*, *above* denote four parameters used to decide *level*. Num_{DC} and $\text{Num}_{\text{Planar}}$ denote the number of parameters whose values are equal to DC and planar. *Threshold* denotes the integer mode difference value reflecting threshold angle between two modes, and is set to be 8 in our work. $\text{Num}_{\text{Candidates}}$ denotes the number of candidate modes selected. *L* and *A* is shorthand of *left_mode* and *above_mode*. *NC* denotes no change from previous algorithm. And *X* (*N*) denotes the directional adjacent angular mode of mode *X* (*X* can be *A*, *L*, *R[0]*, *R[1]*). Simplified rules for MPM mechanism is clearly shown. The proposed MPM method can be described in algorithm 1.

Algorithm 1. Optimized MPM algorithm.

1. Input: current CU *rpcCU*, STAD R-D cost ascending ordered array *RdModeList*, spatial left PU's prediction mode *left_mode*, spatial above PU's prediction mode *above_mode*.
 2. Output: MPM final candidate set.
 3. Algorithm detail steps:
 - Step 1: Judge the size of *rpcCU*, decide value of $\text{Num}_{\text{Candidates}}$ denoting number of selected candidates from *RdModeList*.
 - Step 2: Get spatial left and above adjacent PUs' prediction modes *left_mode* and *above_mode*, initialize array *Preds*.
 - Step 3: *RdModeList[0]*, *RdModeList[1]*, *left_mode* and *above_mode* decide the *level* in Table 1.
 - Step 4: Update the content of *Preds* according to complexity level. Merge *Preds* with specific elements of *RdModeList* into final candidate collection and remove repetitions.
-

2.2 Proposed CU Splitting Algorithm

Texture features of pictures may be one of main factors which influence cost and depth of splitting process. Many experiments indicate complex textured CUs always need precise splitting with small size sub-CUs, which means splitting depth of these pictures is always equal to 2 or 3. In contrast, CUs with simple texture perform a smaller partition depth 0 or 1 in most areas.

CU splitting process starts to divide CU from a depth called start splitting depth. And CU splitting process end with a depth called end splitting depth. Values of depth ranges from 0 to 3 corresponding to different sized CUs. The default value of start splitting depth is 0 and default value of end splitting depth is 3 as defined.

Our work takes influence of QP into consideration of depth correlation of adjacent frames. If the start splitting depth of current CU $depth_{start}$ with minimum depth of temporally previous co-located CU, $depth_{start}$ minus 1 under these situations that depth relationship is not conform to QP relationship, which can be described as

$$depth_{start}(n) = \begin{cases} depth_{start}(n) - 1; & \text{if } (QP(n) > QP(n-1) \text{ and } depth_{start}(n) > Avg_depth(n-1)) \\ depth_{start}(n); & \text{otherwise} \end{cases} \quad (4)$$

The first correction in optimized algorithm is the start splitting depth correction according to Eq. (4). And in consideration of the poor correlation of adjacent frames' maximum depth, limit condition is set as Eq. (5), trying to narrow the scope of maximum depth optimization to reduce the loss of splitting accuracy.

level	R [0]	R [1]	left	above	MPMs	
					NumCandidates=3	NumCandidates=8
0	Num _{DC} +Num _{Planar} ≥ 3			Planar, DC, R[0]		
1	Num _{DC} +Num _{Planar} = 2			DC, Planar, R[0,1]	DC, Planar, R[0,1,2,3]	
2	Num _{DC} +Num _{Planar} = 1			DC, Planar, R[0,1,2]	DC, Planar, R[0,1,2,3,4,5]	
3	R[0]-R[1] ≥ Threshold left-above ≥ Threshold			NC		
4	left-above < Threshold			L,A,L(N), A(N),R[0]	L,A,L(N), A(N),R[0,1]	
5	R[0]-R[1] < Threshold left-above ≥ Threshold			L,A, R[0](N),R[0]	L,A,R[0](N), R[1](N),R[0,1]	

Temporally previous co-located [6] CU's depth information includes minimum, maximum, and average splitting depth. min_pre_dpt is used to initialize start splitting depth of current CU, avg_pre_dpt is used for judgment that decide the value of some other variable, and max_pre_dpt is used for maximum depth optimization. Here, $flag_m$ is defined to decide whether maximum depth optimization should be applied or not, it is defined as

$$flag_m = (scale \geq 1 \mid \mid preQP < currQP) \& max_pre_dpt < 3 \quad (5)$$

where $preQP$ denotes the temporally previous co-located CU's QP and $currQP$ denotes QP of current CU. If $preQP$ is smaller than $currQP$, average depth of current CU is smaller than the temporally previous co-located CU. In this case, maximum depth

optimization can be applied. And *scale* denotes the value of scale value of the distance between *avg_pre_dpt* and *max_pre_dpt* and the distance between *avg_pre_dpt* and *min_pre_dpt*, calculated as

$$scale = \frac{max - avg}{avg - min} \quad (6)$$

where *max* denotes *max_pre_dpt*, *min* denotes *min_pre_dpt*, and *avg* denotes *avg_pre_dpt*. If *flag_m* is false, *min_pre_dpt* is used to initialize the start splitting depth of current CU, *max_pre_dpt* will not be applied. If *flag_m* is true, it indicates partitions in smaller depth is more than those in larger depth, *max_pre_dpt* can be used to optimize splitting process by stop splitting in advance when depth equals *max_pre_dpt* before it increases to 3 as permitted.

In consideration that the fact many CUs' maximum depth is 3 may limit the effect of maximum depth optimization, the conditions that decide the value of *flag_m* set threshold 1 for *scale*, which indicates that this maximum depth optimization only performs when the number of sub-CUs whose depth is close to *min_pre_dpt* is more than the number of sub-CUs whose depth is close to *max_pre_dpt*, which decreases the loss of prediction accuracy that imprudent reduction of large depth precise splitting lead to.

Meanwhile, we propose a limitation for B-Slice to avoid prediction error that CU splitting optimization may lead to from RFS of HEVC. This condition is set to be conducted every 16 frames as follow: select the first previous and backward reference frame of current frame as *pre_Pic* and *suf_Pic*, respectively. $\Delta depth_{start}$, $\Delta depth_{end}$ and $\Delta depth_{avg}$ denote the difference of three depth information of *pre_Pic* and *suf_Pic*.

Experiments are conducted: three depth difference of the first previous and backward reference frame $\Delta depth_{start}$, $\Delta depth_{end}$ and $\Delta depth_{avg}$ are divided into 20 parts, and weight value for every depth information is set ranging from 1 to 20 when sum of three weight value is 20. At the same time, threshold for depth values is tested to range from 0 to 45 with 5 as interval. Results indicate that when weighting factor of these three depth information is set 7, 10, 3 and threshold is set 25, and POC difference greater than or equal to 4 as is shown in Eq. (7), it is calculated as

$$flag_g = 7 \times |\Delta depth_{start}| + 10 \times |\Delta depth_{avg}| + 3 \times |\Delta depth_{end}| \geq 25 \& |POC_{pre_Pic} - POC_{suf_Pic}| \geq 4 \quad (7)$$

if *flag_g* is true, the proposed method do not conducted. Otherwise, proposed CU splitting optimization algorithm will be applied in encoding process. Here, Table 3 shows CU splitting optimization with QP, depth distribution and RFS correction.

Meanwhile, Algorithm 2 also show the proposed CU splitting for inter. Almost all probabilities exceeding 90 % reveal depth correlation seems reliable with additional conditions described above.

Table 3. Depth correlation probability with all corrections

Probability		$P(dpt_{min}(n) \geq dpt_{min}(n-1))$ with formula (4)	$P(dpt_{max}(n) \leq dpt_{max}(n-1))$ with formula (5)
Basketball Drive	22	93.12%	91.14%
	27	91.52%	90.78%
	32	89.28%	92.28%
	37	91.56%	93.52%
Taffic	22	89.30%	95.34%
	27	92.12%	91.56%
	32	94.72%	96.60%
	37	98.06%	95.14%

Algorithm 2. Optimized depth traversing algorithm

1. Input: current splitting depth $uidepth$, temporally previous co-located CU $preCU$, current CU $rpcCU$.
2. Output: $BestCU$ denoting best prediction block of $rpcCU$.
3. Algorithm detail steps:
 - Step 1: Get all depth and QP information of $preCU$.
 - Step 2: Utilize minimum depth of $preCU$ to initialize start splitting depth $depth_{start}$.
 - Step 3: Judge the value of $flag_m$ according to Eq.(5). If $flag_m$ is true, stop splitting process of $rpcCU$ when $uidepth$ equal maximum depth of $preCU$.
 - Step 4: Recursively check best mode of sub-CUs to build $BestCU$ until $uidepth$ is equal to 0 again.

This inter optimization algorithm performs from the second P or B slice to the end of sequence. It mainly consist of four parts in summary: initialize splitting depth, correct initial depth, judge and perform maximum depth optimization, and store current CU’s depth information. The algorithm process of the whole optimization algorithm in inter prediction can be described in algorithm.3.

Algorithm 3. Inter CU splitting optimized algorithm

1. Input: current splitting depth $uidepth$, temporally previous co-located CU $preCU$, current CU $rpcCU$.
 2. Output: $BestCU$ denoting best prediction block of $rpcCU$, the depth information of $rpcCU$.
 3. Algorithm detail steps:
 - Step 1: Judge if SliceType of $rpcCU$ and $preCU$ are all P or B Slice, jump to Step2. Otherwise, algorithm 3 is not allowed, original algorithm performs instead.
 - Step 2: Judge if SliceType of $rpcCU$ are B Slice, judge $flag_g$ using Eq.(7), if $flag_g$ is true, algorithm 3 is not allowed, original algorithm performs instead. If $flag_g$ is false or SliceType of $rpcCU$ are P Slice, Jump to Step3.
 - Step 3: Correct $depth_{start}$ in step 2 using Eq.(4).
 - Step 4: Perform algorithm 2.
 - Step 5: Judge if SliceType of $rpcCU$ is not I Slice, store splitting depth information in $rpcCU$.
-

3 Experimental Results

The performance of the proposed optimization algorithm is evaluated by comparing encoding time and image quality before encoding and after reconstruction, with original and proposed algorithm implemented on HM12.0 [7]. Benchmark recommended by the JCT-VC [8] has listed experimental conditions which are widely used to evaluate performance of video encoding standard. Original algorithm of HM12.0 and HM12.0 with proposed algorithm applied is tested and results are recorded, respectively. Experiments are conducted on PC installed with Intel Pentium CPU E6600 @3.06 GHz and 2 GB RAM.

The differences of image quality between proposed and pre-algorithm are measured by the Bjontegaard difference bit-rate (BD-bitrate) and Bjontegaard difference PSNR (BD-PSNR) [9] indicating the reduction of prediction accuracy, and the reduction of encoding time is calculated using average saving time percentage (ASTP) in our paper defined by:

$$ASTP = \frac{Enc.Time_{HM12.0} - Enc.Time_{Proposed}}{Enc.Time_{HM12.0}} \times 100\% \tag{8}$$

where $Enc.Time_{HM12.0}$ denotes the average encoding time through all the QP value of original HEVC Reference Software HM12.0, and $Enc.Time_{Proposed}$ denotes the average encoding time through all the QP value of HM12.0 with proposed algorithm applied in.

Test sequences with different resolutions are selected to measure the effect the proposed algorithm produce on frames of different texture complexity. Experimental results are given in Figs. 3 and 4. Figures 3 and 4 indicate the proposed algorithm always achieve more than 30 percent reduction on encoding time with almost negligible increase of BD-bitrate and decrease of BD-PSNR, even 40 percent reduction on encoding time as well as less than 2 percent increase in BD-bitrate.

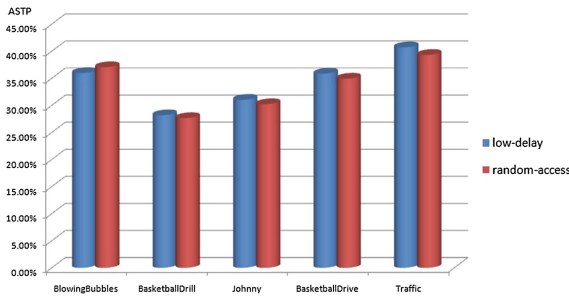


Fig. 3. ASTP change with the increase of frame size

Experiment results seems indicate that no matter what definition encoding sequence is, proposed algorithm always perform 30 percent to 40 percent reduction on encoding

time, averagely cutting down more than other optimizations that have been published in public with 20 percent to 30 percent reduction. If prediction accuracy of BD-bitrate is increased, it is not good in generally achieving 0.8 percent to 1.8 percent compared with other optimizations averagely range from 0.5 percent to 1 percent.

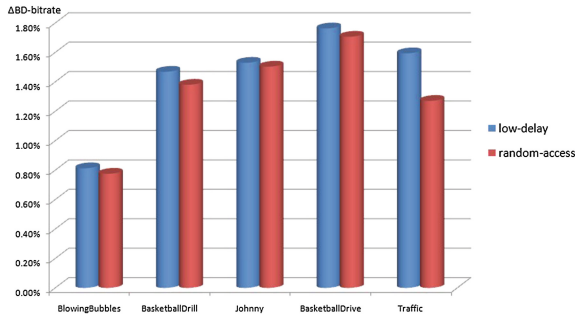


Fig. 4. BD-bitrate change with the increase of frame size

For holistic optimization, BD-bitrate increase 0.8 percent to 1.8 percent is not a bad result. For example, the fast inter CU selection method based on pyramid motion divergence [10] with BD-bitrate increase ranges from 1.1 percent to 2.7 percent. It shows the increase of BD-bitrate in our experiments is indeed acceptable.

4 Conclusion

We propose optimization algorithm for intra and inter prediction process based on texture feature and adjacent correlations. Our algorithm achieves remarkable encoding time reduction with almost negligible increment of bit-rate compared to the HEVC, especially applied in encoding high definition video sequences which have become more and more common in multimedia network video transmission.

Acknowledgment. This work is partly supported by science and technology support program of Sichuan Province (Grant No.2013GZ01 51).

References

1. Bross, B., Han, W.-J., Ohm, J.-R., Sullivan, G.J., Wiegand, T.: High Efficiency Video Coding (HEVC) text specification draft 10, document JCTVC-L1003. JCT-VC Geneva, Switzerland (2013)
2. Zhu, C., Lin, X., Chau, L.P.: An enhanced hexagonal search algorithm for block motion estimation. In: IEEE International Symposium on Circuits and Systems (ISCAS 2003), Bangkok, Thailand (2003)
3. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. *IEEE Trans. Circ. Syst. Video Technol.* **22**(12) (2012)

4. Lee, Y.M., Sun, Y.T., Lin, Y.: SATD-based intra mode decision for H.264/AVC video coding. *IEEE Trans. Circ. Syst. Video Technol.* **20**(3), 463–469 (2009)
5. Zhu, C., Xiong, B.: Transform-exempted calculation of sum of absolute Hadamard transformed differences. *IEEE Trans. Circ. Syst. Video Technol.* **19**(8), 1183–1188 (2009)
6. Zhou, C.T., Zhou, F., Chen, W.Y.: Spatio-temporal correlation-based fast coding unit depth decision for high efficiency video coding. *J. Electron. Imaging* **22**(4) (2013)
7. JCT-VC HEVC reference software version HM 12.0. <https://hevc.hhi.fraunhofer.de/svn/Software/tags/HM-12.0>
8. Bossen, F.: Common HM test conditions and software reference configurations, document JCTVC-L1100, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) (2013)
9. Bjontegaard, G.: Calculations of average PSNR differences between RD-curves, Doc. VCEG-M33 (2011)
10. Xiong, J., Li, H.L., Wu, Q.B., Meng, F.: A fast HEVC inter CU selection method based on pyramid motion divergence. *IEEE Trans. Multimedia* **16**(2), 559–564 (2014)