A HIGHLY PARALLEL CODING UNIT SIZE SELECTION FOR HEVC

Liron Anavi, Avi Giterman, Maya Fainshtein, Vladi Solomon, and Yair Moshe

Signal and Image Processing Laboratory (SIPL)
Department of Electrical Engineering, Technion – Israel Institute of Technology
yair@ee.technion.ac.il, http://sipl.technion.ac.il/

ABSTRACT

The High Efficiency Video Coding (HEVC) standard provides a substantial improvement in coding efficiency over previous video coding standards at the cost of a higher computational complexity. HEVC employs a quadtree based image structure by partitioning the image into coding units (CUs). Finding the optimal CU size in terms of rate-distortion is one of the most computationally challenging parts of any HEVC encoder. Previous works for fast CU size selection are usually based on data dependency between neighboring CUs and therefore limit the degree of possible parallelism. In this paper, we present a fast CU size selection method that does not depend on any data from other CUs in the same frame, thus allowing utilization of the high parallel processing capability of many-core processors, such as a GPU. Experimental results show that the proposed method incurs only a negligible loss in rate-distortion performance compared with counterpart methods that limit parallelism.

Index Terms— HEVC, coding unit (CU), fast video coding, parallelization, GPU

1. INTRODUCTION

High Efficiency Video Coding (HEVC) is the newest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). It enables substantially increased compression performance, compared with previous standards, at the expense of increased computational complexity [1]. To achieve high compression performance, HEVC replaces macroblocks, which were used in previous video coding standards, with a quadtree structure, enabling a high level of flexibility in the encoding process. HEVC initially divides each frame into non-overlapping coding tree units (CTUs) that can be of size 16x16, 32x32 or 64x64 pixels. Each CTU can be recursively further divided into four smaller quadratic blocks called coding units (CUs), down to blocks of size 8x8. This recursive subdivision composes a quadtree structure and its levels are usually called depths. The quadtree structure allows high compression efficiency as smooth stationary areas are encoded with large blocks and detailed areas with complex motion are encoded with small blocks, resulting in a small side-information overhead. Selecting a good quadtree partitioning, in terms of rate-distortion, is extremely important for achieving high compression efficiency. However, since there are many possibilities for partitioning a CTU, and since each partitioning under consideration requires a rate-distortion optimization of different motion estimation and intra prediction modes, this procedure incurs a high computational complexity. Therefore, it is required to design a clever fast method for CU size selection.

Several methods for CU size selection have been proposed in the literature. Some proposals to speed up CU decision are based only on information from the current CU. For example, a mathematical model for reducing the number of depth checks, based on Bayesian decision rule and estimation methods, is introduced in [2]. This algorithm has an offline stage applied to a training set of videos. The results depend on the provided training set, which may not qualify for certain types of videos. Early termination is considered to be a simple and efficient technique to reduce encoding time. A top-down partitioning of the HEVC quadtree is usually performed, pruning the quadtree once a threshold of rate-distortion cost is reached. An early termination method based on analysis of previous SKIP mode CUs is proposed in [3] and implemented in the HM4.0 reference software. It is based on the observation that if the SKIP mode is chosen as the best mode for the current CU, then no further splitting is required. This method can only be applied to inter prediction and is elaborated for non-SKIP CU modes in [4].

Other proposals to speed up CU decision use information from the neighboring CUs to select candidate CU sizes. A content-based fast CU decision algorithm was developed in [5]. The ratio of utilized CUs to total number of CUs in different depths at frame level is analyzed and rarely used CUs with specified depths are skipped. The analysis used in this method is data dependent and is affected by encoding configurations. A method that uses an early termination approach has been proposed in [6] based on the adaptive weighted average rate-distortion cost of adjacent SKIP mode CUs. Methods based on depth information correlation between spatial and temporal adjacent CUs and the current CU are proposed in [7-10]. They speed up calculation by reducing the depth search range. [10] is a revision of the method [9] with favorable results compared with counterpart algorithms.

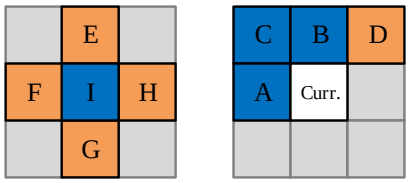

Fig. 1. Neighboring CTUs of the current CTU used in [10] for CU size selection. $\boldsymbol{\alpha} = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{I}\}$, $\boldsymbol{\beta} = \{\boldsymbol{D}, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}\}$

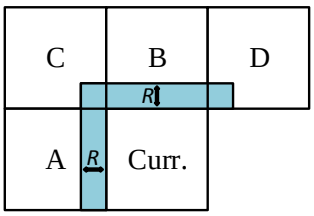

Fig. 2. The depth information extracting region decided by $\boldsymbol{R}$ in the current frame according to [10].

Since the aforementioned fast CU size selection methods assume serial computation, the acceleration they can achieve is limited. With the recent development of many multicore parallel computing platforms, it is possible to further accelerate video encoding through parallelization. Nowadays, personal computers are typically equipped with a highly parallel, powerful, and cost-effective Graphics Processing Unit (GPU). Hence, some previous works in the literature have proposed to accelerate HEVC encoding using a GPU [11-15]. The most computationally demanding part of a video encoder is motion estimation. Therefore, most works implement this feature on the GPU, while leaving the rest of the encoding process for the CPU. Although high speedups are achieved by GPU-based motion estimation algorithms, further acceleration is still required for practical video services. In particular, after accelerating motion estimation, CU size selection becomes a major bottleneck. In addition, reducing the depth search range of CU sizes may prevent unnecessary computations and reduce the computational load on both the CPU and the GPU. Parallelizing an HEVC encoder CU size selection is difficult due to many dependencies at CTU-level within the same frame. One approach for CU size selection on many-core processors, such as a GPU, is to schedule computations by building a directed acyclic graph of dependencies among neighboring CTUs [16].

In this paper, we propose a highly parallel method for CU size selection. In contrast to [16], we do not map dependencies but remove some of them. The method does not depend on any data from other CTUs in the same frame, thus allowing easy parallelization for a GPU.

2. FAST SERIAL CU SIZE SELECTION

The CU size selection method proposed in [10] is designed for serial computation and gives superior results compared with counterpart serial methods. In this section, we describe this method in brief as it is a basis for the parallel method we propose later in this paper.

The method described in [10] exploits spatial and temporal correlations of a video. CU size selection relies on nine previously coded CTUs, divided into two groups: α and β, as shown in Fig. 1. Out of the CTUs that have already been coded in a serial order, these CTUs are the best candidates to predict which depths are more probable than the others for the current CTU quadtree partitioning [9]. All the CTUs in group α have higher correlation to the CTU being evaluated than those in group β [9]. The four CTUs in group α are first checked for the depths they adopted. Then, if needed, the five CTUs in group β are also checked for the depths they adopted. One of the major contributions of this method compared with counterpart methods is that depths are checked in neighboring CTUs only in CUs that are in a small strip of size $R = 8$ around the CTU being evaluated. This strip is depicted in Fig. 2. Using depth information only from CUs that are in a small strip around the CTU being evaluated, raises spatial correlation, leading to more accurate depth information.

According to the quadtree depths adopted by neighboring CUs in a strip of size R, four similarity degree classes are defined - high similarity, medium-high similarity, medium-low similarity and low similarity. Low similarity indicates that all depths are adopted in group α, hence no conclusive decision can be made on possible depths of the evaluated CTU. In this case, 3 out of the 4 possible depths are checked, discarding depth 0 (no partitioning of the CTU) or depth 3 (smallest possible CUs). The depth to discard is selected based on the depths of I, the collocated CTU in the reference frame. High similarity indicates that all the CTUs in group α adopted a single depth. If all the CTUs in group β also adopted the same depth, this depth is selected for the evaluated CTU. Otherwise, an additional neighbor depth is checked. In many cases, low or high similarity conditions are not met since two or three depths are adopted by the CTUs in group α. Such medium-similarity CTUs have a tendency to be either more neighbor-dependent, or not. Those more correlated to neighboring CTUs, i.e., the CTUs in their group α adopted two depth levels, are classified as medium-high similarity. Medium-low similarity is used to handle cases where group α adopted three depth levels. For medium-low similarity CTUs, since a single depth is not adopted by any of the CTUs in group α, this depth is discarded. A second depth can be further discarded if one depth is adopted by all the CTUs in group α, and another depth is adopted only by a single CTU in group α. If two depths are adopted each by one CTU in group α, the depth discarded is the one with the greater distance to the CTU adopted by all the neighboring CTUs. For medium-high similarity, although two depths are not adopted by group α, one, two or three depths may be checked. Three depths are checked if group β adopted a depth

Similarity	Depths checked	Group β used?
low	3	no
medium-low	2 or 3	no
medium-high	1 or 2 or 3	yes
high	1 or 2	yes

Table 1. Summary of the methods for fast CU size selection used in [10] and in this paper. The degree of similarity between the CTU under evaluation and neighboring CTUs in group α determine the number of depths to be checked and whether to also use information from neighboring CTUs in group β.

Group	CTU	Corr. for $R = 8$
α	A	0.767
	B	0.772
	C	0.759
	I	0.789
β	D	0.735
	E	0.725
	F	0.729
	G	0.725
	H	0.722

Table 2. Depth correlations between the current CTU and neighboring CTUs, taken from [10].

not adopted by group α. If group β adopted two such depths, only the depth with a greater number of occurrences is additionally checked. One depth is checked if group β adopted no new depths and CTU C is the only CTU to adopt one of the two depths. This is done due to the relatively low correlation in group α of CTU C to the evaluated CTU. Otherwise, two depths are checked. The aforementioned procedure for fast CU selection is summarized in Table 1.

3. FAST PARALLEL CU SIZE SELECTION

A GPU is a parallel many-core programmable processor able to provide high computational performance in comparison with a CPU. Its tremendous computational capability can be used not only for accelerating computer graphics algorithms, but also for general purpose computing (GPGPU). Therefore, there has been a strong demand for using a GPU as a coprocessor to assist the CPU with data-intensive applications. A GPU is designed for problems that can be expressed as a parallel computation. Its programming is based on running short code sections, called kernels. Each of these kernels defines a task that can be executed multiple times and independently from other kernel calls. The overhead of initiating and loading a kernel onto the GPU is high. Therefore, using a GPU for executing high complexity calculations becomes efficient only when running an initiated kernel many times. A natural implementation of an HEVC encoder on a GPU or any other many-core processor, may

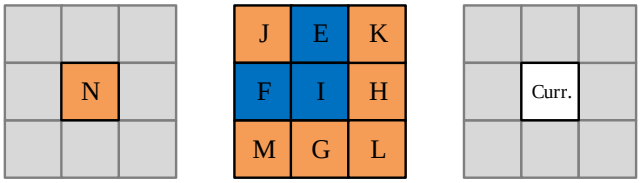

Fig. 3. Neighboring CTUs of the current CTU used in this paper for CU size selection.
$\boldsymbol{\alpha} = \{\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{I}, \boldsymbol{I}\}$, $\boldsymbol{\beta} = \{\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{J}, \boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}, \boldsymbol{N}\}$

create many kernel calls by allocating computational tasks for each CTU to a different kernel call. In order to be efficient, the GPU must run the kernel calls independently. As a result, this architecture makes it impossible to rely on information from other CTUs in the same frame, since computational tasks for all kernel calls should execute in parallel. However, as most fast CU size selection methods proposed in the literature assume serial computation, and therefore use information from spatial adjacent CTUs, they are not suitable for execution on a GPU. In this section, we propose a method for fast CU size selection that does not use information from other CUs in the same frame. This method has no spatial dependencies in the same frame thus suitable for highly parallel implementation on a many-core processor.

Depth correlations between the current CTU and neighboring CTUs were collected in [10] and are summarized in Table 2. From this table we can see that, as we take information about quadtree depths from previously coded frames, the correlation between the evaluated CTU and neighboring CTUs decreases. This brings us to the assumption that predicting the optimal quadtree partitioning of a CTU based on the depths of neighboring CTUs from previous frames only is expected to hurt coding efficiency. In order to verify this assumption and to quantify the loss of coding efficiency, we have created a "temporally shifted" version of the method proposed in [10]. Namely, we use the method as in [10] with $R = 8$. However, instead of using group α and group β with CTUs of the current frame N and of the reference $N-1$ frame, we use CTUs of the reference $N-1$ frame and of the $N-2$ frame respectively, at the same locations. Results of this naive "temporal shift" on the sequences given in Table 3, show on average an increase of about 2% in bit-rate with a negligible degradation in PSNR, compared with [10]. These results show that, even with a simple change from serial methods, it is possible to gain significant acceleration of CU size selection for HEVC encoding by high parallelization with only a small penalty in coding efficiency.

With these encouraging results in mind, we will now describe a fast CU size selection method that exploits better the correlations with CTUs in previous frames. The CTUs used by this method are depicted in Fig. 3. For designing the method, we use two observations. First, as seen in Table 2,

Class	Sequence	[10]		proposed	
		BDrate [%]	ΔT [%]	BDrate [%]	ΔT [%]
B	BQTerrace	0.63	-41.70	3.31	-66.14
C	BasketballDrill	1.37	-38.19	0.80	-61.30
	BQMall	1.00	-38.31	1.95	-59.08
	PartyScene	0.16	-32.27	1.18	-56.41
	RaceHorses	0.50	-30.88	0.59	-55.36
D	BasketballPass	0.52	-34.74	2.45	-52.83
	BQSquare	-0.10	-27.63	2.03	-54.30
	BlowingBubbles	0.36	-25.29	1.59	-54.54
	RaceHorses	0.41	-24.26	0.98	-52.76
Average		**0.54**	**-32.58**	**1.65**	**-56.99**

Table 3. Results of the proposed CU size selection method compared with [10]. For each method, change in coding performance in BD-rate, and change in (serial) coding time, are given compared to the HM16.2 reference software.

out of the neighboring CTUs in frame $N-1$, the collocated CTU I has substantially higher correlation with the evaluated CTU. In order to exploit this higher correlation, we introduce a weighted version of group α. This group contains now three instead of four CTUs but CTU I is included twice to give it a double weight in the similarity degree class decision. CTUs E and F are also included in this group due to their high correlation with the evaluated CTU. A second observation is that, as we use CTUs from a frame that has already been coded, we can use neighboring CTUs not only to the left and to the top of the evaluated CTU but also to its right and to its bottom (CTUs G, H, L, M). This allows us to compensate for the decrease in CTU correlation by adding more CTUs that were not available in the serial algorithm and introduce a larger group β. The new group β includes seven CTUs instead of five - the six neighboring CTUs in the $N-1$ frame not included in group α (CTUs G, H, J, K, L, M), and the collocated CTU from the $N-2$ frame – CTU N. This CTU has a high correlation with the evaluated CTU, estimating according to Table 2 a loss of 0.038 in correlation with CTU I. The method we propose for parallel CU size selection is similar to the one proposed in [10] with different CTUs in group α and in group β. It is important to note that, although the description of the method we propose is similar to the description of [10] in Section 2 and in Table 1, due to the change in the group α and in group β, the decisions of the new method may be different. In particular, since group α in the proposed method includes CTU I twice, more CTUs on average may be assigned with a high similarity compared with [10], thus fewer depths are checked. This leads to a faster running time.

4. RESULTS

In order to evaluate the performance of the proposed method, we implemented it in the HM16.2 reference software and we also implemented the method proposed in [10]. We tested with video sequences in class B (1920x1080), class C (832×480) and class D (416×240). We used "low delay P, main" encoding configuration with $QP = 22, 27, 32, 37$. The experimental results are given in Table 3, in which the change in coding performance, measured by BD-rate [17], and change in (serial) coding time are given. Table 3 compares [10] and the proposed method to the reference software (implementing [3]). The proposed method outperforms the state-of-the-art method proposed in [10] and achieves on average 1.65% bitrate increase compared with the reference software. Even for a serial implementation, running time is reduced by 56.99% on average compared with the reference software and is lower than the running time of [10]. A much larger reduction in running time can be achieved by a highly parallel implementation on a many-core processor, such as a GPU. Notice that more time savings are achieved for video sequences of higher resolutions. The reason is that there is a stronger spatial correlation between neighboring CUs in images of high resolution. Fig. 4 shows a comparison of rate-distortion curves for the sequence BQTerrace (1920x1080) between the reference software and the proposed method. It can be seen that the rate-distortion difference is very small.

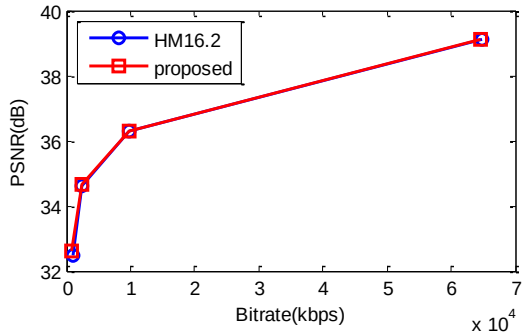

Fig. 4. Rate-distortion curves of the sequence BQTerrace.

5. CONCLUSION

In this paper, we proposed a fast, highly parallel CU size selection method for HEVC that is suitable for implementation on a many-core processor, such as a GPU. Parallelism is achieved by removing dependencies and not using information from other CUs in the same frame. Nevertheless, the proposed method achieves only a negligible loss in rate-distortion performance and faster running times compared with counterpart serial methods that limit parallelism even when executed in a serial manner.

ACKNOWLEDGMENT

The authors would like to thank Prof. David Malah, head of SIPL, and Nimrod Peleg, chief engineer of SIPL, for their support, advice, and helpful comments. The work was supported in part by Harmonic. Special thanks are given to Raz Nitzan from Harmonic for encouraging this activity.

6. REFERENCES

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, pp. 1649-1668, 2012.

[2] X. Shen, L. Yu, and J. Chen, "Fast coding unit size selection for HEVC based on Bayesian decision rule," in *Picture Coding Symposium (PCS), 2012*, 2012, pp. 453-456.

[3] K. Choi and E. S. Jang, "Fast coding unit decision method based on coding tree pruning for high efficiency video coding," *Optical Engineering*, vol. 51, p. 030502, 2012.

[4] R.-h. Gweon and L. Yung-Lyul, "Early termination of CU encoding to reduce HEVC complexity," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, pp. 1215-1218, 2012.

[5] J. Leng, L. Sun, T. Ikenaga, and S. Sakaida, "Content based hierarchical fast coding unit decision algorithm for HEVC," in *Multimedia and Signal Processing (CMSP), 2011 International Conference on*, 2011, pp. 56-59.

[6] J. Kim, S. Jeong, S. Cho, and J. S. Choi, "Adaptive coding unit early termination algorithm for HEVC," in *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, 2012, pp. 261-262.

[7] H. S. Lee, K. Y. Kim, T. R. Kim, and G. H. Park, "Fast encoding algorithm based on depth of coding-unit for high efficiency video coding," *Optical Engineering*, vol. 51, p. 067402, 2012.

[8] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An effective CU size decision method for HEVC encoders," *Multimedia, IEEE Transactions on*, vol. 15, pp. 465-470, 2013.

[9] Y. Zhang, H. Wang, and Z. Li, "Fast coding unit depth decision algorithm for interframe coding in HEVC," in *Data Compression Conference (DCC), 2013*, 2013, pp. 53-62.

[10] R. Fan, Y. Zhang, Z. Li, and N. Wang, "An improved similarity-based fast coding unit depth decision algorithm for inter-frame coding in HEVC," in *MultiMedia Modeling*, 2014, pp. 529-540.

[11] X. Wang, L. Song, M. Chen, and J. Yang, "Paralleling variable block size motion estimation of HEVC on CPU plus GPU platform," in *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*, 2013, pp. 1-5.

[12] S. Radicke, J. Hahn, C. Grecos, and Q. Wang, "A highly-parallel approach on motion estimation for high efficiency video coding (HEVC)," in *Consumer Electronics (ICCE), 2014 IEEE International Conference on*, 2014, pp. 187-188.

[13] X. Jiang, T. Song, T. Shimamoto, and L. Wang, "High efficiency video coding (HEVC) motion estimation parallel algorithms on GPU," in *Consumer Electronics-Taiwan (ICCE-TW), 2014 IEEE International Conference on*, 2014, pp. 115-116.

[14] S. Kim, D.-K. Lee, C.-B. Sohn, and S.-J. Oh, "Fast motion estimation for HEVC with adaptive search range decision on CPU and GPU," in *Signal and Information Processing (ChinaSIP), 2014 IEEE China Summit & International Conference on*, 2014, pp. 349-353.

[15] F. Wang, D. Zhou, and S. Goto, "OpenCL based high-quality HEVC motion estimation on GPU," in *Image Processing (ICIP), 2014 IEEE International Conference on*, 2014, pp. 1263-1267.

[16] C. Yan, Y. Zhang, J. Xu, F. Dai, L. Li, Q. Dai, *et al.*, "A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors," *Signal Processing Letters, IEEE*, vol. 21, pp. 573-576, 2014.

[17] G. Bjontegaard, "Calcuation of average PSNR differences between RD-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.