HIGHLY PARALLEL HEVC MOTION ESTIMATION BASED ON MULTIPLE TEMPORAL PREDICTORS AND NESTED DIAMOND SEARCH

Esmaeil Hojati, Jean-François Franche, Stéphane Coulombe, Carlos Vázquez
École de technologie supérieure, Montréal, Canada

ABSTRACT

Rate-constrained motion estimation (RCME) is the most computationally intensive task of H.265/HEVC encoding. Massively parallel architectures, such as graphics processing units (GPUs), used in combination with a multi-core central processing unit (CPU), provide a promising computing platform to achieve fast encoding. However, the dependencies in deriving motion vector predictors (MVPs) prevent the parallelization of prediction units (PUs) processing at a frame level. Moreover, the conditional execution structure of typical fast search algorithms is not suitable for GPUs designed for data-intensive parallel problems. In this paper, we propose a novel highly parallel RCME method based on multiple temporal motion vector (MV) predictors and a new fast nested diamond search (NDS) algorithm well-suited for a GPU. The proposed framework provides fine-grained encoding parallelism. Experimental results show that our approach provides reduced GPU load with better BD-Rate compared to prior full search parallel methods based on a single MV predictor.

Index Terms— HEVC, rate-constrained motion estimation, GPU, massively parallel architecture

1. INTRODUCTION

The latest hybrid video compression standard, H.265/HEVC, was developed by the Joint Collaborative Team on Video Coding (JCT-VC) [1]. Although HEVC doubles the compression efficiency of H.264/AVC without degrading the visual quality, its computational complexity is considerably higher [2]. Fortunately, HEVC defines several high-level parallelization tools, such as wavefront parallel processing (WPP) and tiles, which allow processing several coding tree unit (CTUs) in parallel. These high-level tools were designed for multicore processors, but they cannot provide enough parallelization for many-core CPUs or heterogeneous CPU/GPU architectures. Moreover, GPUs are appropriate for data-parallel algorithms. Therefore, while fast search methods are less complex, their conditional execution structure is not suitable for a GPU's single instruction multiple thread (SIMT) model. Subsequently, full search (FS) method is mostly used for GPU implementation [3].

In the literature, several methods were proposed to process, in parallel, the rate-constrained motion estimation (RCME), identified as the most complex task of a video encoder. Some of them process several prediction units (PUs) in parallel [4–9]. The main challenge with these methods is determining the best motion vector (MV) for a PU without knowing its motion vector predictors (MVPs). They usually estimate these MVPs from already encoded CTUs' MVs as in Yu et al. [4] and Yan et al. [5] where such estimation is based on spatial information. These methods permit the parallel processing of all the PUs within a CTU. However, to provide parallel RCME for all the CTUs in a frame, the spatial MVP dependency must be removed completely. Moreover, Wang et al. [6] and Fan Wang et al. [7] proposed methods by transferring the calculated distortion values to CPU. Chen et al. [8] proposed to ignore the MVPs when determining MVs in parallel at the frame level, resulting in reduced rate-distortion (RD) performance. Shahid et al. improved the RD performance by using MVs from a previous frame to form a temporal prediction of MVPs [9]. Although these methods achieve fine-grained parallelism suitable for a GPU, MVP prediction errors lead to noticeable and undesirable RD performance losses.

In this paper, we propose a novel highly parallel RCME method for a fast search algorithm using multiple temporal MV predictors. The method, targeted at CPU/GPU heterogeneous architectures, performs RCME for all the PUs of a CTU in parallel. The use of multiple temporal MV predictors ensures good RD performance while the proposed fast nested diamond search (NDS) is well-suited to the GPU's data-parallel paradigm and provides significant complexity reduction compared to FS. Moreover, our approach can be combined with high-level tools such as WPP, tiles and slices to reach a higher degree of parallelization and speed.

The paper is organized as follows. Section 2 presents the RCME process in HEVC and its dependencies. The parallel encoding framework for a CPU/GPU heterogeneous architecture is presented in section 3. Section 4 contains the experimental results. Finally, Section 5 concludes this paper.

This work was funded by Vantrix Corporation and by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Development Program (NSERC-CRD 428942-11). Emails: {esmaeil.hojati-najafabadi.1, jean-francois.franche.1}@ens.etsmtl.ca,{stephane.coulombe, carlos.vazquez}@etsmtl.ca

2. MOTION ESTIMATION IN HEVC

HEVC utilizes a quadtree structure called CTU to partition each frame. This structure consists of blocks and units with a maximum size of 64×64 pixels. A block is composed of a rectangular area of picture samples with related syntax information. A CTU can be recursively divided into coding units (CUs). The information associated with the prediction process of a CU is stored in the PUs. Partitioning and selection of best PU modes are performed by an RD optimization (RDO) process.

In the HEVC Test Model (HM) [10], RCME is a process which consists in estimating the best temporal prediction parameters based jointly on the rate and distortion for each PU. It is typically performed at integer precision followed by fractional refinement. The sum of absolute differences (SAD) and the sum of absolute transformed differences (SATD) are used as a distortion measures (Dist) for integer and fractional pel precision respectively.

2.1. Full search RCME

This approach examines all the possible MVs in a rectangular area. To find the best mode, the cost function is defined as:

$$\mathrm{J_{ME}}\,(mv, mvp) = \mathrm{Dist}(mv) + \lambda \cdot \mathrm{Rate}(mvp - mv) \qquad (1)$$

where $\mathrm{Dist}(mv)$ is the distortion associated with mv, $\mathrm{Rate}(mvp - mv)$ is the bit-cost, which is a function of the difference between the MV and the MVP. The constant λ is a Lagrange multiplier. Using Eq. 1, the prediction parameters for RCME are obtained as follows:

$$\begin{aligned} \mathrm{P_{ME}} &= (mv^*, mvp^*) \\ &= \underbrace{\arg\min}_{\substack{\forall mv \in MV_{search}, \\ mvp \in \{mvp_A, mvp_B\}}} \{\mathrm{J_{ME}}\,(mv, mvp)\} \end{aligned} \qquad (2)$$

where mvp_A and mvp_B are determined from neighboring PUs, MV_{search} is defined as the search region composed of the set of integer MVs covering a square area.

2.2. Fast search RCME

This approach examines a reduced set of MVs to decrease the computational complexity compared to FS but may find suboptimal MVs. Among those, the Test Zone search (TZS) has been adopted in the HM. It first determines the center of the search by evaluating several candidates from neighboring CUs [11]. Then a search pattern is employed to find the global minimum. After that, if the distance between the best point and center is less than 2, the search process ends, else the new best point is assigned as the center and the search is repeated.

As we can anticipate, the number of TZS iterations depends on the selected search center and on the distortion values, producing unequal number of iterations for different PUs. For instance, a spatial area of a frame with unpredictable or complex motion will require more iterations than a relatively still area. We propose a modified execution model for TZS to prevent GPU execution inefficiencies due to possible different execution paths for different PUs.

3. PROPOSED FAST MULTI-PREDICTOR RCME

In this section, we present our proposed fast parallel framework for RCME in HEVC. It comprises a method to efficiently execute fast search on GPU and a novel multi-temporal-predictor RCME (MTP-RCME) which reduces the rate-distortion penalty due to MVP prediction errors.

3.1. GPU fast search RCME

As mentioned, TZS performs a variable number of iterations for each PU, making it inefficient for a GPU. In this section, we explain the reasons for this inefficiency considering the GPU architecture and propose a fast search algorithm.

3.1.1. GPU architecture considerations

The GPU hardware is designed to execute parallel programs using the SIMT computing model. This model executes the same copy of a parallel program (kernel) on different data. In this paper, we use the Open Computing Language (OpenCL) [12][13] terminology to describe our method. Each instance of a program is run by a *workitem* or thread. *Workitems* are grouped into workgroups. However, because of the scheduler architecture in the GPU, *workitems* are executed in clusters. The number of *workitems* that are processed in a cluster is 32 for NVIDIA (a thread warp) and is 64 for AMD (a thread wavefront). To use resources efficiently, the number of *workitems* in a workgroup should be a multiple of the wavefront thread number.

Furthermore, a wavefront executes *workitems* in parallel using the SIMT processing model across the processing unit. Thus, the execution time is affected by divergences in the execution flow. Branching, for example, is achieved by combining all execution paths into a unique sequence of instructions. This implies that the total time to execute a multipath branch is the sum of the execution time of each individual path. As a consequence, even if only one *workitem* in a wavefront diverges, all the *workitems* in the wavefront will execute the diverging branch [14].

3.1.2. Proposed GPU fast nested diamond search method

Considering these constraints, to map the TZS algorithm into an efficient data-parallel model, we define a fixed search pattern with 64 MV positions. This search pattern is a modified diamond search pattern as depicted in Fig. 1. This pattern is concentrated in the center and surrounded by four embedded diamond patterns with 8 pixels step. Furthermore, one wavefront performs block matching for all the positions. After each iteration, if the termination condition is not met, the center is moved to the best position and another search iteration is performed by the same workgroup.

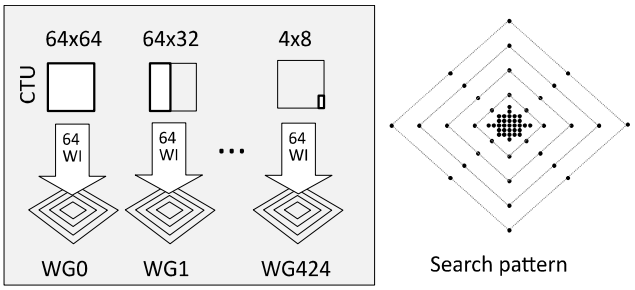

Fig. 1. Workitem (WI) and workgroup (WG) mapping

Moreover, each CU consists of several PUs but each PU might require a different number of iterations. The algorithm requires special arrangements to prevent performance loss. Thus, we split the CU into all the possible PUs and assign a workgroup to each PU. To match the GPU's data-parallel model, the RCME for each PU is defined by a data structure containing the arguments for this process. For each PU, the job structure contains the position of the PU along with its block dimensions and the result is a distortion and MV pair. For a CTU, these job structures are precomputed and stored into arrays. When asymmetric mode partitioning (AMP) is not enabled, each CTU consists of 425 possible PUs and accordingly a workgroup is assigned to each PU. Fig.1 depicts job scheduling and work mapping of a CTU.

To exploit even more the GPU's processing capabilities, we also perform interpolation filtering and fractional pel refinement after the integer motion estimation in the GPU. The reference frames are updated and interpolated in the GPU right after the reconstruction of each frame. The interpolation filter in GPU is implemented as a separable filter. For each pixel, sixteen sub-pel samples are generated. The image is partitioned into one-pixel wide columns with 256 one-pixel rows, and interpolation of each column is done by one workgroup consisting of 256 *workitem*s. Each *workitem* is calculating sixteen subsamples for each pixel. Algorithm 1 summarizes the GPU kernel for the NDS method.

3.2. Multi-temporal-predictor RCME

In HEVC, the derivation of the MVP from neighboring PUs prevents a high degree of parallelism. Also, using an improper MVP in the RCME process will produce an incorrect rate cost that will in turn lead to incorrect optimal MV selection. To achieve a high degree of parallelism while preserving a high-coding efficiency, we propose a method that evaluates the cost as Eq. 1 on a list of probable MVPs collected from the past encoded frame MVs. These MVs eliminate dependencies between all the CTUs of the current frame. Using multiple probable MVs will result in better compression performance than one MVP. This proposed method is called multi-temporal-predictor RCME (MTP-RCME), which was presented for FS in [15].

Algorithm 1. Proposed nested diamond search method kernel

```
1:  WG ← get_group_id()        ► PU index (idx)
2:  WI ← get_work_id()         ►Position idx in search pattern
3:  PUjob ← PUArray[WG]        ► PU size and position
4:  SearchPos ← PosArray[WI]►Search position
5:  BestMVI ← MVP_i , iter ← 0
6:  do
7:      Center ← BestMVI
8:      J_ME[WI] ← SAD(PUjob, Center + SearchPos)
9:      BestMVI ← argmin(J_ME[0…63]) ► After barrier
10:     Iter ← iter + 1
11: while (iter < 4 and abs(Center - BestMVI) ≥ 2)
12: BestMV = fractionalRefinement(BestMVI)
```

In HEVC, the MVP is derived from neighboring and collocated blocks. However, to eliminate the spatial dependency, the MTP list for a CTU consists of the set of encoded MVs belonging to the collocated CTU in the previously encoded frame. For a CTU of size 64x64, the encoder preserves sixteen temporal MVs [16]. Therefore, no overhead is induced to build the MTP list. The MTP list is:

$$mvp_i \in \{mv_{c1}, \dots, mv_{cN}\},\ N \leq 16 \tag{3}$$

where N is the number of different candidates that are derived from the previous frame and mv_{ci} is a MV in the collocated CTU in the previous frame. The same MTP list is used for all PUs of a CTU. In the proposed MTP-RCME method, Eq. 2 is modified as follows:

$$mv_i = \underset{mv \in MV_{search}}{\arg\min} \{ \mathrm{J_{ME}}\,(mv, mvp_i)\} \tag{4}$$
$$\mathrm{D}_i = Dist(mv_i)$$

The resulting parameters from Eq. 4 are the best rate-constrained MV and the corresponding distortion for mvp_i. The pair of (D_i, mv_i) is stored to be used in the CPU. In the CPU, when the actual MVP is available, the best pair in terms of RD is determined by the following formula:

$$\mathrm{P_{ME}} = (mv^*, mvp^*) = \underset{\substack{mv_i, \mathrm{D}_i, \text{ with } i \in 1\dots N, \\ mvp \in \{mvp_A, mvp_B\}}}{\arg\min} \{\mathrm{D}_i + \lambda \cdot \mathrm{Rate}(mvp - mv_i)\} \tag{5}$$

In order to integrate our proposed method into HEVC, we separate the RCME calculation from the RDO mode decision procedure. Our proposed MTP-RCME performs the RCME using MTP list in the GPU. The result of this stage is used in the RDO process performed by the CPU. In the RDO stage, the actual MVP is available, and, using the prior GPU calculated results, the best decision is made using Eq. 5 with very low computational complexity in the CPU.

Furthermore, the frame encoding is executed by two separate threads that provide asynchronous CPU and GPU execution and communications without stalls. One thread is used for the RDO processing, and the other for offloading the workload and communicating with the GPU. After providing the GPU with data, the GPU calculates MTP-RCME for all PUs in parallel, while the CPU performs the RDO process when the required data is available. The MTP-RCME process flowchart in the CPU is depicted in Figure 2.

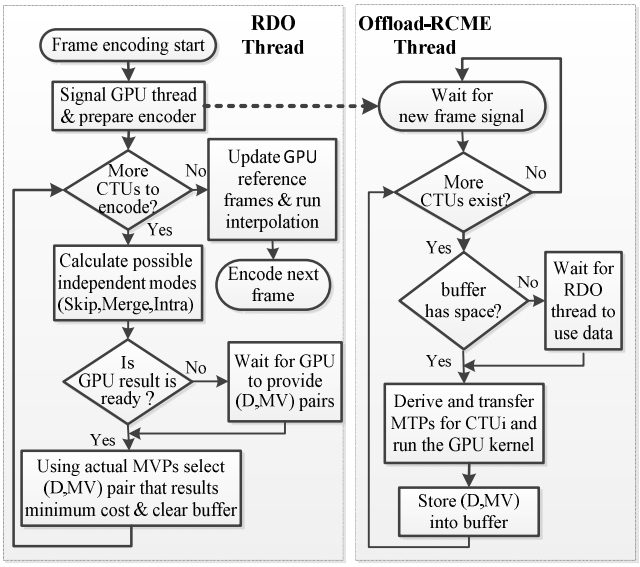

Fig. 2. Flowchart of RDO and offloading threads on the CPU

4. EXPERIMENTAL RESULTS

To validate the proposed MTP and NDS methods, we implemented them into the HEVC Test Model (HM 15.0) [10] by modifying the RCME part of the code. We executed simulations on an Intel® CPU i7-4770 running at 3.40GHz, and equipped with an AMD Radeon R9-270 GPU. These simulations consisted of encoding some standard video sequences defined in the *common HM test conditions* [17], with the "*Low-delay P*" configuration and quantization parameters (QPs) of 22, 27, 32, and 37.

In a first set of simulations, we compared the RD performance of the following approaches: 1) the original HM using the fast search algorithm (TZS); 2) a parallel full search algorithm, performed in GPU, using a single MVP set to zero as described in [12] (Zero-FS); 3) a parallel full search algorithm using a single MVP derived by averaging four collocated MVs in a past frame as presented in [18] (AVG-FS); 4) the proposed parallel NDS method using null MVP of [12] (Zero-NDS); 5) the proposed NDS method using the AVG MVP of [18] (AVG-NDS) and; 6) the complete proposed method combining both MTP and NDS (MTP-NDS). The RD performance of these approaches are measured by the Bjøntegaard delta rate (BD-Rate) [19]. The anchor reference is the original HM using its full search algorithm. All approaches employ a search range of [-64, 64].

According to Table 1, the best RD performance for a parallel approach is achieved by our MTP-NDS approach with a 1.46% BD-Rate increase. In addition to slight BD-Rate improvement, the proposed MTP-NDS approach provides a significantly better GPU usage than a FS method. We have implemented a parallel FS approach similar to the one presented in [12]. For FS we measured an average GPU load of 86% compared to 52% for MTP-NDS.

In a second set of simulations, we analyzed the impact of the three MVP methods on the execution time when the proposed NDS algorithm is used. The speed impact was measured by the time reduction (TR) metric between the encoding time of the original sequential HM using the TZS and the execution time of the evaluated methods on the CPU/GPU architecture. Table 1 shows the proposed MVP method has no impact on the TR, on the level of 40%, compared to the Zero and AVG methods, while it achieves better RD performance. The TR is currently limited by the CPU. Faster speedups can be achieved by combining this method with high-level tools such as WPP, tiles and slices.

5. CONCLUSION

In this paper, we proposed a novel RCME method based on multiple temporal MV predictors and a fast NDS algorithm. The method provides the high degree of parallelization needed to efficiently exploit massively parallel architectures with increased RD performance.

Table 1. Rate distortion and time reduction comparison between proposed method and prior art methods

Video	BD-Rate (%) compared to HM full search						TR (%) compared to HM TZS				
	TZS	Zero-FS	AVG-FS	Zero-NDS	AVG-NDS	MTP-NDS	Zero-FS	AVG-FS	Zero-NDS	AVG-NDS	MTP-NDS
BQSquare (416×240)	0.29	1.72	1.41	2.13	1.34	0.68	36.8	38.2	37.1	37.6	37.4
BasketballPass (416×240)	0.41	2.01	1.42	2.24	1.82	0.98	37.7	37.3	37.9	37.7	37.5
BlowingBubbles (416×240)	0.18	1.78	1.65	1.56	0.77	0.46	37.7	37.5	38.2	37.7	37.6
RaceHorses (416×240)	0.95	2.60	2.07	3.84	3.52	2.15	36.2	36.8	36.4	36.4	35.5
BQMall (832×480)	1.18	1.94	1.65	3.28	2.76	1.74	41.0	40.7	41.4	41.5	40.9
BasketballDrill (832×480)	1.11	2.16	1.73	3.14	2.59	1.60	42.1	41.7	41.9	41.3	41.1
Flowervase (832×480)	0.31	2.11	1.52	2.08	1.49	0.64	37.9	39.0	37.7	38.6	38.4
RaceHorses (832×480)	1.08	2.97	2.28	3.80	3.39	2.56	38.9	37.8	38.5	37.5	37.6
FourPeople (1280×720)	0.81	2.15	1.67	3.02	2.39	1.43	43.7	43.4	43.2	43.6	43.1
Johnny (1280×720)	0.82	1.76	1.55	2.64	2.24	1.57	44.0	43.7	44.8	44.5	45.8
Cactus (1920×1080)	0.47	2.63	1.99	2.66	2.01	1.29	43.7	42.7	43.5	42.9	42.9
Kimono (1920×1080)	0.59	2.25	1.72	3.28	2.49	1.85	41.3	40.9	41.3	41.4	41.7
ParkScene (1920×1080)	0.48	2.61	1.93	3.18	2.62	1.68	42.6	41.5	42.8	42.1	42.5
PeopleOnStreet (2560×1600)	0.62	2.98	2.49	3.30	2.62	1.83	41.5	42.6	42.3	42.1	42.3
Average	0.66	2.26	1.79	2.87	2.29	1.46	40.4	40.3	40.5	40.3	40.1

6. REFERENCES

[1] B. Bross, W. J. Han, J. R. Ohm, G. J. Sullivan, Y. K. Wang, and T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10." document JCTVC-L1003, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), Jan-2013.

[2] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," *2013 Picture Coding Symposium (PCS)*. IEEE, pp. 394–397, 2013.

[3] C. Jiang and S. Nooshabadi, "GPU accelerated motion and disparity estimations for multiview coding," *2013 IEEE International Conference on Image Processing*. pp. 2106–2110, 2013.

[4] Q. Yu, L. Zhao, and S. Ma, "Parallel AMVP candidate list construction for HEVC," *Vis. Commun. Image Process.*, 2012.

[5] C. Yan *et al.*, "Efficient Parallel Framework for HEVC Motion Estimation on Many-Core Processors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 12, pp. 2077–2089, 2014.

[6] X. Wang, L. Song, M. Chen, and J. Yang, "Paralleling variable block size motion estimation of HEVC on multi-core CPU plus GPU platform," *Image Process.*, no. 10110502200, pp. 1836–1839, 2013.

[7] S. Fan Wang , Dajiang Zhou, "OpenCL based high-quality HEVC motion estimation on GPU," *IEEE Int. Conf. Image Process.*, pp. 1263–1267, 2014.

[8] W. Chen and H. Hang, "H.264/AVC motion estimation implementation on compute unified device architecture (CUDA)," *IEEE Int. Conf. Multimed. Expo*, pp. 697–700, 2008.

[9] M. U. Shahid, A. Ahmed, and E. Magli, "Parallel rate-distortion optimised fast motion estimation algorithm for H.264/AVC using GPU," *2013 Picture Coding Symposium (PCS)*. IEEE, pp. 221–224, 2013.

[10] "Joint Collaborative Team on Video Coding Reference Software, ver. HM 15.0." [Online]. Available: http://hevc.hhi.fraunhofer.de/.

[11] J. H. Jeong, N. Parmar, and M. H. Sunwoo, "Enhanced test zone search algorithm with rotating pentagon search," *2015 International SoC Design Conference (ISOCC)*. pp. 275–276, 2015.

[12] S. Momcilovic and L. Sousa, "Development and evaluation of scalable video motion estimators on GPU," *Signal Processing Systems, 2009. SiPS 2009. IEEE Workshop on*. IEEE, pp. 291–296, 2009.

[13] "The open standard for parallel programming of heterogeneous systems," *Khronos Group*. [Online]. Available: https://www.khronos.org/opencl/.

[14] AMD, "AMD Accelerated Parallel Processing OpenCL Programming Guide," 2013. [Online]. Available: http://developer.amd.com/wordpress/media/2013/07/AMD_Accelerated_Parallel_Processing_OpenCL_Programming_Guide-rev-2.7.pdf.

[15] E. Hojati, J. F. Franche, S. Coulombe, and C. Vázquez, "Massively Parallel Rate-Constrained Motion Estimation using Multiple Temporal Predictors in HEVC," *submitted to the 2017 IEEE International Conference on Multimedia and Expo (ICME2017)*. Nov-2016.

[16] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[17] F. Bossen, "JCTVC-L1100: Common HM test conditions and software reference configurations. JCT-VC Document Management System (April 2013)." 2013.

[18] J. Ma, F. Luo, S. Wang, and S. Ma, "Flexible CTU-level parallel motion estimation by CPU and GPU pipeline for HEVC," *Visual Communications and Image Processing Conference, 2014 IEEE*. IEEE, pp. 282–285, 2014.

[19] G. Bjøntegaard, "Improvements of the BD-PSNR model." ITU-T SG16/Q6 Video Coding Experts Group (VCEG), Document VCEG-AI11, Berlin, Germany, 16-Jul-2008.