An Adaptive Search Range Decision Algorithm for Parallel Motion Estimation

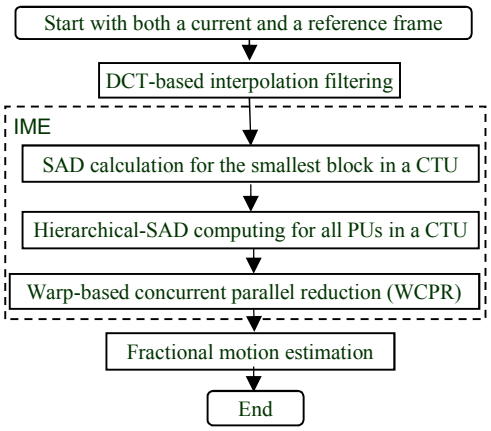
Dongkyu Lee	Chang-Beom Ahn	Youngchul Chung	Seoung-Jun Oh
Hyundai Mobis, Gyeonggi-do, Korea	Dept. of Electrical Engineering, Kwangwoon University, Seoul, Korea	Dept. of Electronics and Comm. Engineering, Kwangwoon University, Seoul, Korea	Dept. of Electronics Engineering, Kwangwoon University, Seoul, Korea
rangkyu87@naver.com	cbahn@kw.ac.kr	ychung@kw.ac.kr	sjoh@kw.ac.kr

***Abstract*—We introduce an adaptive search region decision algorithm to improve the speed of GPU-based motion estimation which generally adopts the full-search. The motion feature of the current block is estimated by MAP (Maximum A Posterior) estimation with the motion feature of the temporally adjacent block and its SR. In our MAP estimation, the prior distributions and the likelihoods are represented by the probabilistic modeling of the data extracted from the training set. The SR is adaptively decided depends on the estimated motion feature. In the motion estimation on GPU for an HEVC encoder, the proposed method with different change rates achieves 22.3% and 31.2% of average time reductions in the integer-pel ME with on average 0.25% and 1.08% BD-bitrate increases, respectively.**

Keywords— HEVC, GPGPU, CUDA, motion estimation, parallel reduction, search range.

I. INTRODUCTION

As the demand for digital video contents increases due to the development of mass storage media, internet streaming technology, and various multimedia devices, the spread of video related services has increased rapidly. However, there exists a limitation in storing and transmitting video data since they require a huge bandwidth. In order to solve these problems, a video compression technique for reducing the size of video data has been developed. As part of these activities, the Video Coding Experts Group (VCEG) in ITU-T and the Moving Picture Experts Group (MPEG) in ISO/IEC formed JCT-VC (Joint Collaborative Team on Video Coding) and established the HEVC (High Efficiency Video Coding) video compression standard in 2013.[1] However, these tools in an HEVC codec have greatly increased the complexity, which is the biggest problem in the commercialization of HEVC.[2] Thus, researches on fast parallel algorithm design for the H.264 and the HEVC codecs have been actively conducted.[3-6]. Recently, a parallelization technique using a Graphics Processing Unit (GPU) is attracting attention.[7] We propose an CUDA-based adaptive search region decision algorithm to improve the speed of GPU-based motion estimation which generally adopts the full-search. CUDA (Compute Unified Device Architecture) is a parallel programming model for GPUs developed by Nvidia.[8] Since ME takes the highest complexity in conventional standard video encoders, a fast ME process is essential for implementing a real-time HEVC encoder. Lee and his colleagues proposed the GPU-based parallel ME algorithm for HEVC shown in Fig. 1 (a).[9] The execution flow of the proposed ME is shown Fig. 1 (b). However, they used a fixed size SR (search region), which may prevent the performance of their algorithm. The performance may be increased if the size of SR can adaptively be decided.



(a) Flow chart of the ME

(b) Execution flow of the ME

Fig. 1. The flow chart and the execution flow of the ME in [9]

In this paper, we introduce an adaptive SR decision algorithm to improve the ME performance introduced in [9]. The performance improvement can be achieved by removing the spatial dependency problem that occurs when GPU parallelization is applied to ME by using temporal data. The standard deviation of motion vector difference (MVD) values highly related to the SR is used as the motion feature. In Section II, an ME method using a GPU is described. In Section 3, we propose an adaptive search range determination method to improve the performance of the ME method described in Section III. From now on, let the proposed algorithm be called

978-1-5386-2615-3/18/$31.00 ©2018 IEEE

GPU-ASR. The performance of GPU-ASR is analyzed based on the experimental results in Section IV, and conclusion is followed in Section V.

II. GPU-BASED ME ALGORITHM

In [9], a frame was partitioned into two subframes for CUDA pipelined execution to achieve high GPU utilization. In the integer-pel ME (IME) stage, there exist three modules: SAD calculation, hierarchical-SAD (H-SAD) computing, and warp-based concurrent parallel reduction (WCPR). At the first module, the representative search center position decision (RSCP) was introduced, which solved a dependency problem in parallel execution by using motion vectors of a co-located CTU in a previously encoded frame. H-SAD computing was followed to reduce computational complexity by data reuse. Then, WCPR executed several PR operations in parallel, which could minimize latency by both increasing thread utilization from 20% to 89% and eliminating thread synchronizations. They insisted that their encoder reduced total encoding time by 56.2% with 2.2% BD-bitrate increase against an HM encoder for HEVC test sequences in Classes B and C. The proportion of the ME process could be reduced to nearly zero. It also provided 1.1% performance improvement over the encoder integrated with CPR-based ME. The proposed ME showed substantial improvement of on average 130.7 times than that of the HM encoder. The WCPR provided 70.6% and 17% improvement with respect to sequential PR and CPR, respectively. However, they used a fixed size SR, which may prevent the performance of their algorithm. Therefore, the performance may be increased if the size of SR can adaptively be decided.

III. THE PROPSED ALGORITHM

The standard deviation of MVD values, σ_{mvd}, in the CTU highly related to the SR is used as the motion feature. The motion feature of the current block is estimated by MAP (Maximum A Posterior) estimation with the motion feature of the temporally adjacent block and its SR. In our MAP estimation, the prior distributions and the likelihoods are represented by the probabilistic modeling of the data extracted from the training set. The SR is adaptively decided depends on the estimated motion feature. The problem of estimating σ_{mvd} can be expressed as the MAP estimation problem in the Bayesian framework shown in Eq.1.

$$p(V_{mvd}^{t} \mid V_{mvd}^{t-1}, S^{t-1}) = \frac{p(V_{mvd}^{t-1} \mid V_{mvd}^{t}, S^{t-1})\, p(V_{mvd}^{t}, S^{t-1})}{p(V_{mvd}^{t-1}, S^{t-1})}, \quad (1)$$

where $V_{mvd}^{t} = \{\sigma_{mvd,1}^{t}, \sigma_{mvd,2}^{t}, \ldots, \sigma_{mvd,N}^{t}\}$, and $\sigma_{mvd,i}^{t}$ is σ_{mvd} of the i-th CTU in the t-th picture. $S^{t-1} = \{s_1^{t-1}, s_2^{t-1}, \ldots, s_N^{t-1}\}$ is a set of search ranges, where s_i^{t-1} represents the SR of the i-th CTU in the (t-1)-th picture. The search region size is $2M \times 2M$ in case of $s_i^{t-1} = M$, where M=8, 16 in this paper. Thus, the estimated value of σ_{mvd}^{t} can be expressed as follows:

$$\hat{\sigma}_{mvd}^{t} = \underset{\sigma_{mvd}^{t} \in \Sigma_{mvd}^{t}}{\arg\max} \left\{ p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}, s^{t-1})\, p(\sigma_{mvd}^{t}, s^{t-1}) \right\}, \quad (2)$$

where Σ_{mvd}^{t} represents all possible cases for σ_{mvd}^{t}. By solving Eq. (2), the value of $\hat{\sigma}_{mvd}^{t}$ can be obtained, and s^t for its corresponding CTU is determined.

In this work, we collect data from a training set to represent a prior distribution and likelihood, and perform probability modeling using those collected data. Since we use $s_i^t \in \{8, 16\}$ in consideration of GPU parallelism and coding efficiency, two prior probabilities such as $p_8(\sigma_{mvd}^{t})$ and $p_{16}(\sigma_{mvd}^{t})$ must be specified first, where $p_M(\sigma_{mvd}^{t})$ is the prior probability for $s_i^t = M$. Two prior probabilities can empirically be expressed by Weibull distribution. Likelihood. $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}, s^{t-1})$ can be considered to represent the similarity of σ_{mvd}^{t-1} and σ_{mvd}^{t} when temporally adjacent CTUs are encoded using s^{t-1}, which can empirically be expressed by Gaussian distribution.

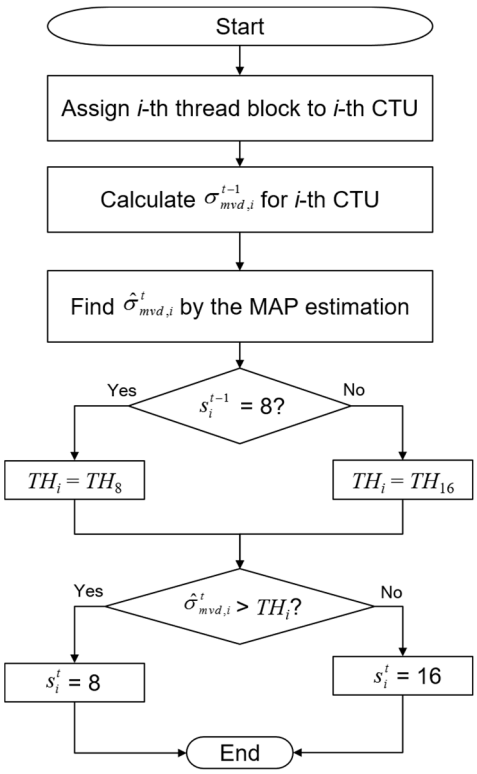


Fig. 2. Flow chart of the proposed method

Each CTU in the t-th picture is allocated to each thread block having 32 threads to determine its SR. Each thread block computes $\sigma_{mvd,i}^{t-1}$ of the co-location CTU in the (t-1)-th picture. Two variables, $\sigma_{mvd,i}^{t-1}$ and s_i^{t-1}, are applied to Eq. (2) to find $\hat{\sigma}_{mvd,i}^{t}$. The threshold values are set to $TH_i = TH_8$ and TH_{16} for determining s_i^t in case of $s_i^{t-1} = 8$ and $s_i^{t-1} = 16$, respectively. If $\hat{\sigma}_{mvd,i}^{t}$ is greater than TH_i, $s_i^t = 16$ is chosen. Otherwise, $s_i^t = 8$ is chosen. Since it determines the SR size based on $\hat{\sigma}_{mvd,i}^{t}$, TH_i is an important user parameter that controls the coding efficiency and the speed-up performance.

The coding efficiency and the speed-up performance according to TH_i will be discussed later. In order to easily implement the proposed algorithm in the GPU, the continuous value of $\sigma_{mvd,i}^{t-1}$ is quantized into a discrete value. We compute the values of the prior distribution and the likelihood according to the discrete value of $\sigma_{mvd,i}^{t-1}$ in advance, which are stored in a lookup table. The computation time for this operation is less than 1ms using Geforce GTX 780 GPU.

IV. Experimental Result

The adaptive search range determination method is applied to motion estimation using GPU to evaluate the performance and coding efficiency of the HM 10.0 encoder. The QP values used are 22, 27, 32, and 37. We use HEVC test sequences shown in Table 1. The GPU-based motion estimation in [**9**] is applied to the encoder of the CTC environment in the low delay P structure. The encoders using the fixed search range of 16 and 8 are represented by E_{SR16} and E_{SR8}, respectively. Experiments were conducted by changing the value of TH_i that determines the search range in the encoder applying the proposed method. There are two thresholds used for each search range, and the encoder to which each threshold value is applied is denoted by E_{ASRD_TH1} and E_{ASRD_TH2}, respectively as shown in Table 2.

Table 1. Test sequences for performance evaluation

Class	Sequences	The number of frames	Frame rate (fps)
Class B (1920×1080)	ParkScene	240	24
	BasketballDrive	500	50
Class C (832×480)	BasketballDrill	500	50
	RaceHorses	300	30

Table 2. Threshold of each SR

SR	TH_i	
	E_{ASRD_TH1}	E_{ASRD_TH2}
8	4	9
16	5	13

The performance is measured using a time reduction rate (TR) shown in Eq. 3, and the coding efficiency is evaluated by BD-bitrate.

$$TR = \frac{T_{\text{ref}} - T_{\text{test}}}{T_{\text{ref}}} \times 100\ (\%), \qquad (3)$$

where T_{ref} and T_{test} represent the execution time of the reference and the compared algorithms, respectively. This performance is denoted by IME-TR in this paper. When we measure coding efficiencies and IME-TRs of E_{SR8} against E_{SR16}, the average BD-bitrate is increased by 3.1% and the reduction rate of the IME time is 42.3%. The coding efficiency of the chroma component is lower than that of the luma component. based on the average BD-bit rate for the three components Y, U, and V. E_{ASRD_TH1} and E_{ASRD_TH2} can provide 22.3% and 31.2% time reduction rate on average with negligible loss of coding efficiency and without any subjective visual quality loss, respectively: the average BD-bitrate increase of 0.25%, and 1.08% was obtained by adjusting the threshold value corresponding to the user parameter. While the encoding efficiency tends to decrease gradually from E_{ASRD_TH1} to E_{ASRD_TH2}, the time reduction rate tends to increase gradually because the threshold value of the two encoders becomes larger, so that the possibility of selecting the search range from 8 to 16 is increased.

In order to evaluate the performance of the proposed method in terms of subjective image quality, the qualities of those decoded videos were compared after the bit streams encoded by E_{SR16} and E_{ASRD_TH2} were decoded. The average PSNR of each picture of the decoded video was measured for both the RaceHorses and the BasketballDrive sequences in which the decrease in the BD bit rate was very large. Fig's 3 and 4 show the frame with the largest PSNR difference in each video in case of QP=22. Nevertheless, it can be seen that there is little difference in image quality between the two pictures in each figure.

(a) E_{SR16} (Y : 39.37 dB, U : 41.36 dB, V : 42.64 dB)

(b) E_{ASRD_TH2} (Y : 39.39 dB, U : 41.31 dB, V : 42.48 dB)

Fig. 3 Decoded pictures for *RaceHorses* (QP=22): (a) E_{SR16} and (b) E_{ASRD_TH2}

V. Conclusion

This paper proposes an adaptive search range determination method to reduce the complexity of integer pixel motion

(a) E_{SR16} (Y : 38.88 dB, U : 43.36 dB, V : 44.88 dB)

(b) E_{ASRD_TH2} (Y : 38.87 dB, U : 43.34 dB, V : 44.81 dB)

Fig. 4. Decoded pictures for *BasketballDrive* (QP : 22): (a) E_{SR16} and (b) E_{ASRD_TH2}

estimation using global search. The proposed method reduces the complexity of integer pixel motion estimation while maintaining the coding efficiency by adaptively determining the search range according to the motion characteristics of the CTU. It is assumed that the standard deviation of the MVD size, which is determined according to the size of the search range, is used as the motion characteristic, and the standard deviation and the search range are proportional to each other. We use the MAP estimation method to estimate the motion characteristics of the current CTU, and model the pre - distribution and likelihood with Bayesian distribution and Gaussian distribution through probability modeling from training data. By comparing the estimated motion characteristics with the set threshold, the search range of the current CTU is adaptively determined.

The adaptive search range determination method is applied to motion estimation using GPU to evaluate the performance and coding efficiency of the HM 10.0 encoder. The average BD-bitrate increase of 0.25%, and 1.08% was obtained with 22.3% and 31.2% reduction of the integer pixel motion estimation time by adjusting the threshold value corresponding to the user parameter. It is shown that the proposed adaptive search range determination method can be used in combination with a conventional GPU-based motion estimation.

ACKNOWLEDGEMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00288) supervised by the IITP (Institute for Information & communications Technology Promotion).

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,* pp. 1649-1668, 2012.

[2] F. Bossen, B. Bross, K. Suhring and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, no. 12, pp. 1685-1696, 2012.

[3] Y. J. Ahn, T. J. Hwang, L. D, S. Kim, S. J. Oh and D. Sim, "Study of parallelization methods for software based real-time HEVC encoder implementation," *Journal of Broadcast Engineering,* vol. 18, no. 6, pp. 835-849, 2011.

[4] Y. J. Ahn, T. J. Hwang, D. G. Sim and W. J. Han, "Complexity model based load-balancing algorithm for parallel tools of HEVC," in *Proceedings of IEEE Visual Communications and Image Processing (VCIP)*, pp. 1-5, 2013.

[5] N. M. Cheung, O. C. Au, M. C. Kung, P. H. W. Wong and C. H. Liu, "Highly parallel rate-distortion optimized intra-mode decision," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 19, no. 11, pp. 1692-1703, 2009.

[6] B. Pieters, C. F. J. Hollemeersch, J. De Cock, Lambert P, W. De Neve and R. Van de Walle, "Parallel deblocking filtering in MPEG-4 AVC/H. 264 on massively parallel architectures," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 21, no. 1, pp. 96-100, 2011.

[7] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. Purcell. "A Survey of General-Purpose Computation on Graphics Hardware". *Computer Graphics Forum*, vol. 26, no. 1, pp. 80-113, 2007.

[8] NVIDIA, "CUDA C programming guide," NVIDIA Corporation, PG-02829-01_v8.0, 2016.

[9] D. Lee, D. Sim, K. Cho and S. -J. Oh, "Fast motion estimation for HEVC on graphics processing unit (GPU)," *Journal of Real-Time Image Processing,* vol. 12, no. 2, pp. 549-562, 2016.