Maximum A Posteriori Estimation-based Adaptive Search Range Decision for Accelerating HEVC Motion Estimation on GPU

Seoung-Jun Oh[1*] and Dongkyu Lee[2]
[1] Department of Electronics Engineering, Kwangwoon University
Seoul, 01897 - Korea
[e-mail: sjoh@kw.ac.kr]
[2] Hyundai Mobis
Yongin, Gyeonggi, 16891 - Korea
[e-mail: rangkyu87@naver.com]
*Corresponding author: Seoung-Jun Oh

Received December 11, 2018; revised February 21, 2019; accepted March 25, 2019; published September 30, 2019

Abstract

High Efficiency Video Coding (HEVC) suffers from high computational complexity due to its quad-tree structure in motion estimation (ME). This paper exposes an adaptive search range decision algorithm for accelerating HEVC integer-pel ME on GPU which estimates the optimal search range (SR) using a MAP (Maximum A Posteriori) estimator. There are three main contributions; First, we define the motion feature as the standard deviation of motion vector difference values in a CTU. Second, a MAP estimator is proposed, which theoretically estimates the motion feature of the current CTU using the motion feature of a temporally adjacent CTU and its SR without any data dependency. Thus, the SR for the current CTU is parallelly determined. Finally, the values of the prior distribution and the likelihood for each discretized motion feature are computed in advance and stored at a look-up table to further save the computational complexity. Experimental results show in conventional HEVC test sequences that the proposed algorithm can achieves high average time reductions without any subjective quality loss as well as with little BD-bitrate increase.

Keywords: adaptive search range (ASR); High Efficiency Video Coding (HEVC); GPU computung; motion estimation (ME); parallel reduction; motion vector difference (MVD)

A preliminary version of this paper appeared in IWAIT2018, January 7-10, Chiang Mai, Thailand. This version includes a concrete analysis and supporting implementation results on a MAP estimator for ASR. This research was partly supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-2016-0-00288) supervised by the IITP(Institute for Information & communications Technology Promotion) and the work reported in this paper was conducted during the sabbatical year of Kwangwoon University in 2016.

http://doi.org/10.3837/tiis.2019.09.015 **ISSN : 1976-7277**

1. Introduction

High efficiency video coding (HEVC) is targeted for efficient compression of high resolution and 3D videos [1-3], which was developed together by both ISO/IEC Moving Picture Experts Group (MPEG) and the ITU-T Video Coding Expert Group (VCEG). Compared with the MPEG-4 Advanced Video Coding (AVC) standard, HEVC can reduce the bitrate by almost 50% with a similar perceptual video quality. Since HEVC uses a more diversified block structure, intra prediction and motion compensation, and two in-loop filters, up to 8K-UHD video contents can be efficiently encoded [4,5]. That achievement in coding gain results mainly from its more flexible block partition mechanism at the cost of high computational complexity [4-7].

In the encoding process, as shown in **Fig. 1**, each picture is divided into coding tree units (CTUs), which are the base units in HEVC [8,9]. The size of a CTU can be chosen as 64×64, 32×32, 16×16, or 8×8. A CTU is composed of a luma coding tree block (CTB), two chroma CTBs, and the associated syntax elements. The luma and chroma CTBs can be further partitioned into smaller blocks using a quad-tree structure. The leaves of the CTBs are specified as coding blocks (CBs). One luma CB and its corresponding two chroma CBs, together with the syntax elements, form a coding unit (CU). The CU shares the identical prediction mode (intra, inter, skip, or merge), and it acts as the root for a prediction unit (PU) partitioning structure.

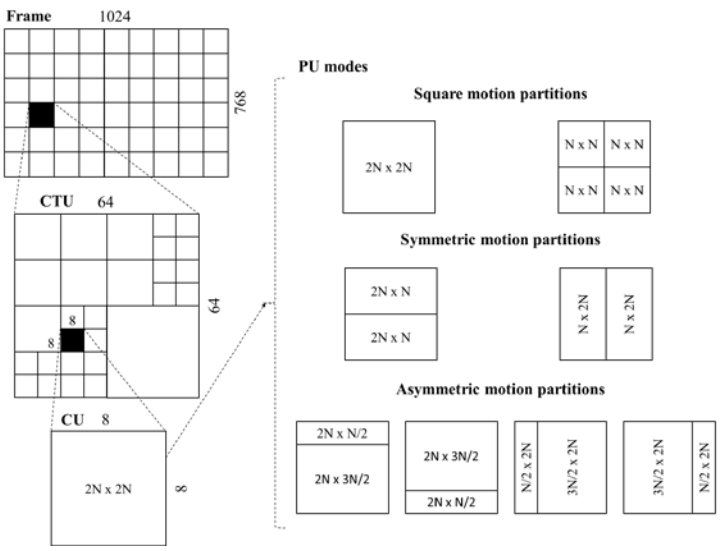


Fig. 1. Flexible block partitioning in HEVC: CTUs, CUs, and PUs

Fig. 1 shows all possible PU modes. The PU is composed of prediction blocks (PBs) in which the same prediction process is applied for its luma and chroma PBs. In the PU partitioning structure of HEVC, each luma and chroma CB can be further partitioned into one, two, or four rectangular-shaped PBs. In HEVC, it adopts square motion partitions, symmetric motion partitions, and asymmetric motion partitions, as shown in **Fig. 1**. It means that every

CU undergoes motion predictions by various types of PU partitions. With this flexible block partitioning mechanism, the motion estimation (ME) itself consumes more than 50% coding complexity or time to encode [6]. Thus, it is requied to design a fast ME algorithm for a real time HEVC codec.

In order to implement a fast video encoder, a number of researches have been made to explore fast sequential motion estimation algorithms [10-19]. One of the methods of reducing computation is to stop the ME process early [10-13]. The efficiency of early termination algorithms is improved by determining the adaptive threshold based on the ratedistortion (RD) cost of highly correlated blocks. Another way to reduce complexity of ME process is either to reduce search points within a fixed search region [14] or to perform ME process within an adaptive search range (ASR) [15-19].

Nowadays, there have been many researches on implementing a parallel processing-based video encoder in order to achieve very high encoding performace [20-35]. One of them is to use a Graphics Processing Unit (GPU) for ME [26-35]. Since various search patterns are not preferable for parallel environment such as GPU due to both their irregular data flow and data dependency [8,14], they have generally adopt full search algorithm which searches all the points within a search range (SR). Thus, the computational complexity for the GPU-based ME strongly depends on the size of the SR. Most GPU-based ME methods adopt CTU-level parallelism requring independent CTU processing and perform hierarchical-SAD (H-SAD) computing [26] which requires a unique SR for a CTU. It is very important for GPU-based ME algorithms to adaptively decide the size of the SR for each CTU without dependencies between neighbouring CTUs in order to accelerate the motion estimation process on GPU.

In this paper, we propose an ASR decision algorithm for accelerating HEVC integer-pel ME on GPU which estimates the optimal SR for each CTU using a MAP (Maximum A Posteriori) estimater, which is called GPU_ASR. GPU_ASR can remove deta dependency between CTUs shown in **Fig. 1** with a negligible RD penalty, which enhances the performance as compared to our previous work [33]. There are three main contributions in this paper as compared to our previous work. First, we define the motion feature as the standard deviation of motion vector difference (MVD) values in a CTU, which is discretized to easily compute the estimator in the GPU. Second, the MAP estimater which theoretically estimates the motion feature of the current CTU using the motion feature of a temporally adjacent CTU and its SR is proposed, which removes data dependency. Thus, the SR for the current CTU is parallelly determined. Finally, the values of the prior distribution and the likelihood for each discretized motion feature are computed in advance and stored at a look-up table to further save the computational complexity. A threshold for motion feature value is used as a user parameter that controls the coding efficiency and the speed-up performance.

The rest of the paper is organized as follows: in Section 2 briefly introduces the previously presented GPU-based ME algorithm [33] and ASR decision algorithms for HEVC ME. Section 3 shows details about our approach. Experiments are conducted to test the accuracy and the speed-up of the proposed ASR decision algorithm and the results are shown in Section 4. Finally, Section 5 concludes the paper.

2. Realted Works

For a HEVC ME, Lee and his colleagues proposed the GPU-based parallel ME algorithm for HEVC [33]. They partitioned a frame into two subframes for high pipelined execution in the GPU. In the integer-pel ME (IME) stage, there exist three modules: SAD calculation, H-SAD computing, and warp-based concurrent parallel reduction (WCPR). They introduced a

representative search center position (RSCP) to solve a dependency problem in parallel execution at the first module. The RSCP was determined by using motion vectors of a co-located CTU in a previously encoded picture. They defined their own MVD as the difference between a current motion vector and a search center vector which is a vector between a current block and a search center point (SCP). Its value is denoted by $mvd = mv - scv$, where mv is the current motion vector and scv is the search center vector. The SCP is determined by the RSCP decision method introduced in [33]. Furthermore, they insisted that the MVD is more closely related to the search region than the motion vector since the search region is set around the SCP. H-SAD computing was followed to reduce computational complexity by data reuse. **Fig. 2** shows the basic concept of H-SAD computing [33]. 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.

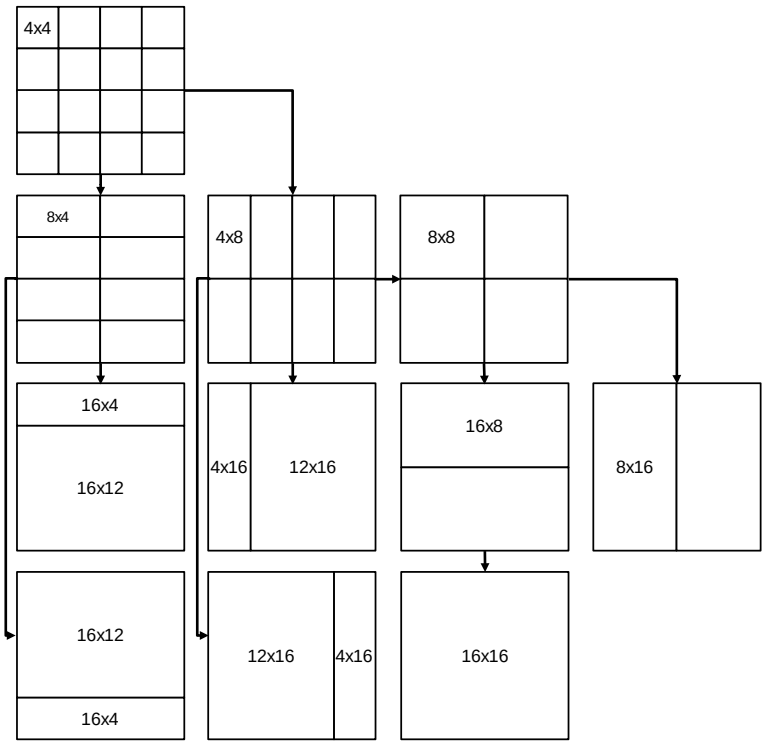


Fig. 2. Concept of H-SAD computing

They insisted that their encoder reduced total encoding time by 56.2% with 2.2% BD-bitrate increase against an HM encoder for Classes B and C of MPEG test sequences. The proposed ME showed substantial improvement of on average 130.7 times than that of the HM encoder. The proposed WCPR provided 70.6% and 17% improvement with respect to sequential parallel reduction (PR) and concurrent PR, respectively.

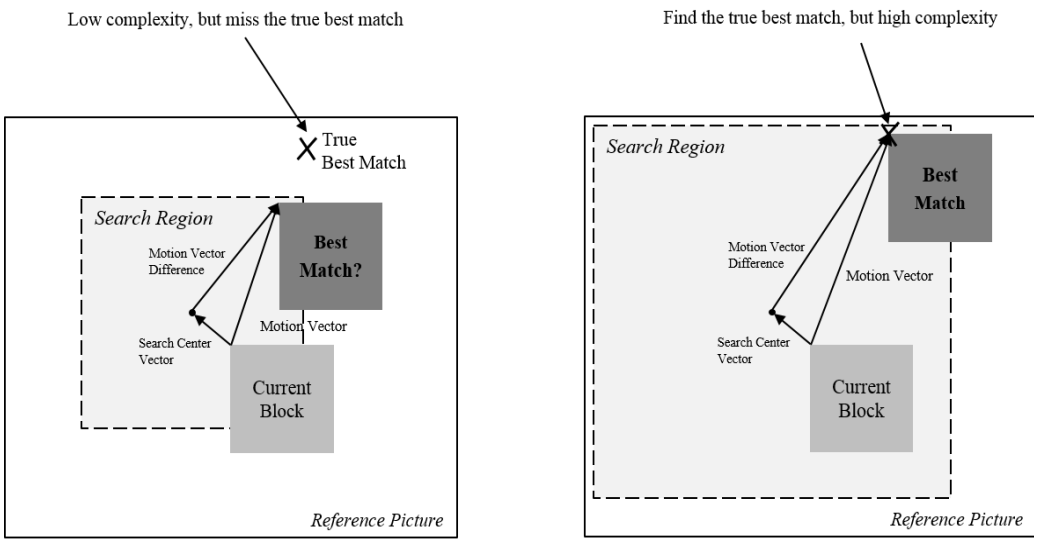
However, they used a fixed-size SR, which degraded the performance of the algorithm. Thus, performance may be improved if the size of the SR can be adaptively determined. Although some ASR decision methods for HEVC were presented [15-19], they cannot be directly applied to GPU-based ME process. In [15], the statistical results of the MVD

prediction distribution is used to decide ASR based on depth level. In [16], temporal correlation between the depth map and the motion in texture is used to form a tailor-made SR. Requiring depth map for ASR decision prevents it from being applied to non-multiview video encoding. In [17,18], ASR deicision is made on PU rather than CTU, which cannot be appied to most of GPU-based ME algorithms. On the other hand, the proposed GPU_ASR is designed for GPU-baed ME which requries data independency between CTUs and a unique SR for a CTU by using temporal data and assigning a SR for a CTU on GPU so that it can be applied to other parallel ME algorithms. Lee and his colleagues proposed ASR for GPU-based ME which is preliminary version of this paper [19].

3. Proposed Adaptive Search Range Decision Algorithm for GPU-based HEVC ME: GPU_ASR

3.1 Motion features for adaptive search range decision

Full search (FS) has generally been used in the area of parallel ME [26-35]. Since all the locations in the SR are searched in FS, s_{sr}, the size of the SR, is the key parameter which determines the complexity. If the SR is small as shown in **Fig. 3 (a)** and does not contain the true best match of the block, the other best match block in the SR is selected as the best match block. In this case, although both the complexity and the accuracy of the motion estimation is low, the coding efficiency is reduced. In contrast, if the search region is too large to include the true best match block and search for many other areas as shown in **Fig. 3 (b)**, the position of the true best match block can be searched to improve both accuracy and coding efficiency.



(a) (b)

Fig. 3. Relationship between s_{sr} and ME performance:
(a) a small search region case and (b) a large search region case

However, searching too many locations can cause performance degradation. Therefore, s_{sr} should be appropriately determined according to the characteristics of the image in order to efficiently determine the position of the true best match block. The standard deviation of MVDs in the current CTU is chosen as a motion feature, which may highly be related to s_{sr}.

The magnitude of MVD and its standard deviation are denoted by $d_{mvd} = |mvd|$ and σ_{mvd}, respectively. GPU_ASR proposed in this paper adaptively determines s_{sr} according to the characteristics of the image. The motion feature in a previously coded CTU is used to determine s_{sr} for the current CTU. Since the motion estimation uses a H-SAD value shown in **Fig. 2**, all the PUs within a CTU must have the same search range. Therefore, the motion feature of the currently coded CTU must be estimated by using motion features of the co-located CTU in the previously coded picture. Then, the motion feature can be used to determine the SR for the current CTU.

Fig. 4 shows that the MVD is more closely related to the SR than the motion vector itself; For the block shown in **Fig. 4 (a)**, a small SR is required since d_{mvd} is small in case that the accuracy of *scv* is high even if *mv* is large. For the block shown in **Fig. 4 (b)**, a large SR is required since d_{mvd} is large in case that the accuracy of *scv* is low even if *mv* is small. As shown in **Fig. 5**, the large σ_{mvd} means that the SR for all MVDs must be large, and the small σ_{mvd} means that all MVDs can be included in a small SR. That is, it can be said that the required s_{sr} is proportional to σ_{mvd}. Thus, an appropriate s_{sr} can be determined using the estimated σ_{mvd}. There may exist one exception in our assumption. If all d_{mvd} in the CTU are similarly large, σ_{mvd} is small. Since d_{mvd} is large, a large s_{sr} is required, but σ_{mvd} is small. Thus, our assumption is wrong in this case. However, according to our experiment on test sequences in HEVC CTC (Common Test Condition) shown in **Table 1**, the probability of this situation occurring is close to zero. Therefore, this case is not considered in GPU_ASR.

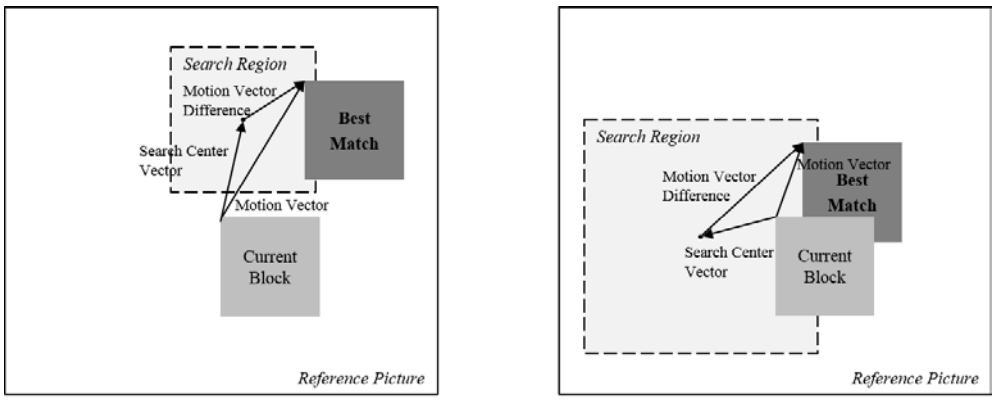


Fig. 4. Relationship between MVD and the search region size. (a) small MVD and small search region and (b) large MVD and large search region

In order to estimate the motion feature of the current block, MAP estimation is performed using the motion feature of a temporally adjacent block and s_{sr}. The SR of the current block is determined adaptively based on the estimated motion feature. In our estimation, we collect data from a training set to represent a prior distribution and likelihood. The training set used for probability modeling consists of the test streams shown in bold in **Table 1**. All the video streams in Table 1 are used as a test set of the designed adaptive search range decision method.

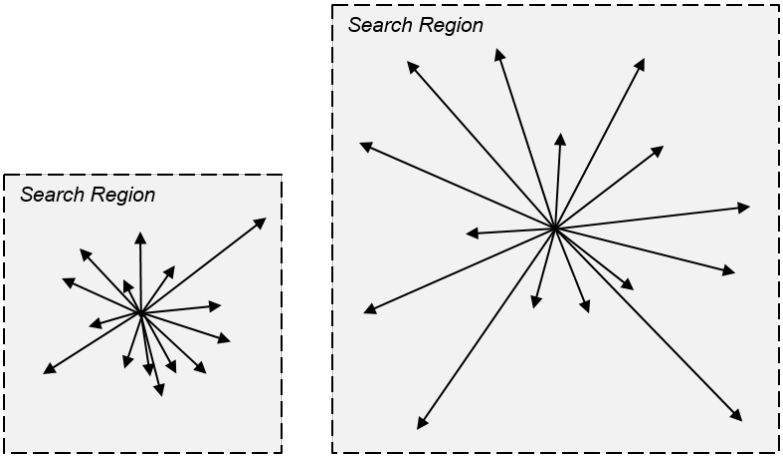


Fig. 5. Relationship between σ_{mvd} and the search region size: (a) search region for small σ_{mvd} and (b) search region for large σ_{mvd}

Table 1. Test video streams for the probabilistic modeling

Class	Sequence	The number of frames
Class B (1920×1080)	Kimono	240
	ParkScene	240
	Cactus	500
	BasketballDrive	500
	BQTerrace	600
Class C (832×480)	**BasketballDrill**	500
	BQMall	600
	PartyScene	500
	RaceHorses	300

3.2 Motion-based motion feature estimation model

The problem of estimating Σ_{mvd}^{t}, the motion feature of the *t*-th picture, can be expressed as the MAP estimation problem in the Bayesian framework shown in Eq.1:

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

where $\Sigma_{mvd}^{t} = \{\sigma_{mvd,1}^{t}, \sigma_{mvd,2}^{t}, \ldots, \sigma_{mvd,N}^{t}\}$, $\sigma_{mvd,i}^{t}$ 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, and s_i^{t-1} represents the search range of the *i*-th CTU in the (*t*-1)-th picture. The search region size is $2M \times 2M$ in case of $s_i^{t-1} = M$. The estimated value $\hat{\Sigma}_{mvd}^{t}$ of Σ_{mvd}^{t} can be expressed as following:

$$\hat{\Sigma}^{t}_{mvd} = \underset{\Sigma^{t}_{mvd} \in \Xi^{t}_{mvd}}{\operatorname{argmax}} \left\{ p(\Sigma^{t-1}_{mvd} \mid \Sigma^{t}_{mvd}, S^{t-1}) p(\Sigma^{t}_{mvd}, S^{t-1}) \right\}, \tag{2}$$

where Ξ^{t}_{mvd} represents all possible combinations of Σ^{t}_{mvd}.

We assume that each CTU in a picture is independent from each other. According to this assumption, Eq. (2) can be decomposed as following:

$$\hat{\Sigma}^{t}_{mvd} = \underset{\Sigma^{t}_{mvd} \in \Xi^{t}_{mvd}}{\operatorname{argmax}} \left\{ \sum_{i} \left(p(\sigma^{t-1}_{mvd,i} \mid \sigma^{t}_{mvd,i}, s^{t-1}_{i}) p(\sigma^{t}_{mvd,i}, s^{t-1}_{i}) \right) \right\}. \tag{3}$$

It can be seen that the solution of Eq. (3) is consistent with that of the estimation problem for each CTU. Therefore, Eq. (3) can be expressed as the MAP estimation problem for CTU as follows:

$$\hat{\sigma}^{t}_{mvd} = \underset{\sigma^{t}_{mvd} \in \varsigma^{t}_{mvd}}{\operatorname{argmax}} \left\{ p(\sigma^{t-1}_{mvd} \mid \sigma^{t}_{mvd}, s^{t-1}) p(\sigma^{t}_{mvd}, s^{t-1}) \right\}, \tag{4}$$

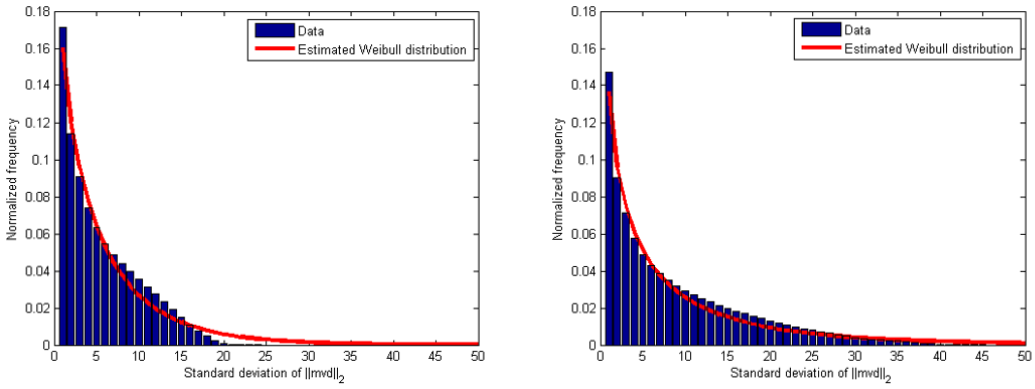
where ς^{t}_{mvd} represents all possible cases for σ^{t}_{mvd}. By solving Eq. (4), the estimated value $\hat{\sigma}^{t}_{mvd}$ of σ^{t}_{mvd} can be obtained, and the search range s^{t} for the corresponding CTU can be determined.

Test video streams are coded with a CTC low-delay P structure using an HM 10.0 encoder [36]. Each video is encoded four times with different QPs: QP = 22, 27, 32, and 37. In this paper, we use two search range sizes such as 8 and 16, most commonly chosen in the HEVC encoder, to simplify the problem and ease of implementation without losing any generality. GPU_ASR is applied to the motion estimation method proposed in [33] where $S^{t} = \{16\}$ is used. If it is possible to select $s^{t}_{i} = 8$ instead of $s^{t}_{i} = 16$ as many as possible with negligible image quality loss, the complexity can clearly be reduced by using a high degree of parallel processing in GPU environment as compared with the case where $s^{t}_{i} = 16$ is only used. Performance evaluation for both search ranges will be discussed later. Each σ^{t}_{mvd} in the *t*-th picture is rounded off to an integer value for ease of implementation and then expressed as a continuous probability distribution through probability modeling.

Since we use $s^{t}_{i} \in \{8, 16\}$ in consideration of GPU parallelism and coding efficiency, two prior probabilities such as $p_{8}(\sigma^{t}_{mvd})$ and $p_{16}(\sigma^{t}_{mvd})$ must be specified first, where $p_{M}(\sigma^{t}_{mvd})$ is the prior probability for $s^{t}_{i} = M$. **Fig. 6 (a)** and **(b)** show the normalized histograms, that is, the prior probabilities for $s^{t}_{i} = 8$ and 16, respectively. While it has a value between 0 and 25 for $s^{t}_{i} = 8$, σ^{t}_{mvd} has a value between 0 and 48 for $s^{t}_{i} = 16$. As expected, σ^{t}_{mvd} has a high probability of occurrence in the low range, and the probability of occurrence in the high range decreases exponentially. The two prior probabilities are experimentally modeled by the Weibull distribution shown in Eq. (5):

$$f(x;\lambda,k) = \begin{cases} \dfrac{k}{\lambda}\left(\dfrac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^{k}} & x \geq 0 \\ 0 & x < 0 \end{cases}, \tag{5}$$

where $k > 0$ is a shape parameter, and $\lambda > 0$ is a scale parameter. Thus, the two prior probabilities are $p_{8}(\sigma^{t}_{mvd}) = f(\sigma^{t}_{mvd}; 0.8696, 5.3548)$ and $p_{16}(\sigma^{t}_{mvd}) = f(\sigma^{t}_{mvd}; 0.7145, 7.1864)$ as shown in **Fig. 6**.



(a) (b)

Fig. 6. Weibull distribution modeling for two prior distributions: (a) $p_8(\sigma^t_{mvd})$ and (b) $p_{16}(\sigma^t_{mvd})$

Likelihood $p(\sigma^{t-1}_{mvd} \mid \sigma^t_{mvd}, s^{t-1})$ can be considered to represent the similarity of σ^{t-1}_{mvd} and σ^t_{mvd} when temporally adjacent CTUs are encoded using s^{t-1}. The likelihood of $p(\sigma^{t-1}_{mvd} \mid \sigma^t_{mvd}, s^{t-1})$ are also determined as a probability distribution modeled using data collected from the training set. **Fig.'s 7** and **8** show the likelihoods for σ^t_{mvd} when s^{t-1} is 8 and 16, respectively. Those likelihoods are experimentally modeled by the Gaussian distribution as following:

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{6}$$

where μ is a mean and σ^2 is a variance. **Fig.'s 7** and **8** show that each mean of the likelihoods is very close to the given σ^t_{mvd}. In case of $s^{t-1} = 8$, (μ,σ) in each likelihood for σ^t_{mvd} = 5, 10, 15, and 20 are (4.44,2.11), (9.59,2.56), (14.45,2.00), and (19.20,1.56), respectively. In case of $s^{t-1} = 16$, the values of (μ,σ) in each likelihood for σ^t_{mvd} = 5, 10, 15, and 20 are (4.08,2.71), (13.89,4.96), (23.83,4.20), and (34.94,1.34), respectively.

Fig. 9 is a flowchart of GPU_ASR. GPU_ASR performs motion estimation process on a per-picture basis in the GPU [33]. Each CTU in the *t*-th picture is allocated to each thread block having 32 threads to determine its search range. Each thread block computes $\sigma^{t-1}_{mvd,i}$ of the co-location CTU in the (*t-1*)-th picture. Two variables, $\sigma^{t-1}_{mvd,i}$ and s^{t-1}_i, are applied to Eq. (4) to find $\hat{\sigma}^t_{mvd,i}$, the estimated value of $\sigma^t_{mvd,i}$. The threshold values are set to $TH_i = TH_8$ and $TH_i = TH_{16}$ for determining s^t_i in case of $s^{t-1}_i = 8$ and $s^{t-1}_i = 16$, respectively. If $\hat{\sigma}^t_{mvd,i}$ is greater than TH_i, $s^t_i = 16$ is chosen. Otherwise, $s^t_i = 8$ is chosen.

Since it determines the selectivity of the search range based on $\hat{\sigma}^t_{mvd,i}$, TH_i is a 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 in Section 4. In order to easily implement the adaptive search range decision method in the GPU, the continuous value $\sigma^{t-1}_{mvd,i}$ is quantized to a discrete value. We compute the values of the prior distribution and the likelihood according to the discrete $\sigma^{t-1}_{mvd,i}$ value in advance. After

computing it according to $\sigma_{mvd,i}^{t-1}$ and s_i^{t-1} using Eq.(4), every $\hat{\sigma}_{mvd,i}^{t}$ is stored in a lookup table. The computation time for this operation is less than 1 ms using Geforce GTX 780 GPU.

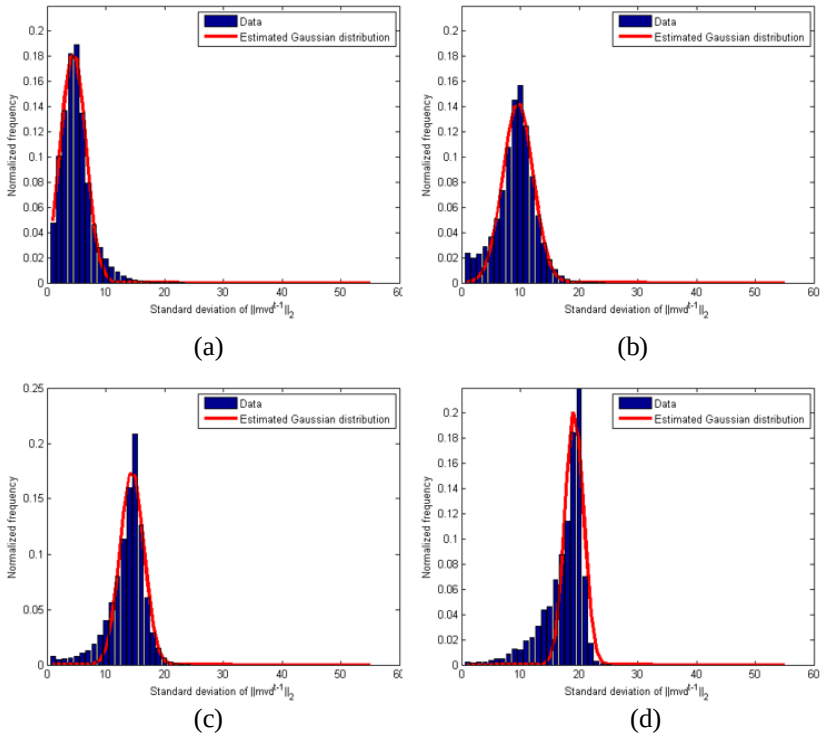


Fig. 7. Likelihoods for the search range of 8, $s^{t-1}=8$: (a) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=5, s^{t-1})$, (b) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=10, s^{t-1})$, (c) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=15, s^{t-1})$, and (d) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=20, s^{t-1})$

4. Experimental Results and Analysis

We use Intel Core i7-2600 @ 3.4 GHz CPU, 8 GB memory, Geforce GTX 780 with 3 GB DRAM, Visual Studio 2012, CUDA Toolkit version 6.5 and graphics card driver version 350.12 on Windows 8 64-bit operating system. The GPU-based motion estimation in [33] is applied to the encoder of the CTC environment in the low delay P structure with the search range of 16, and this is represented by E_{SR16}. An encoder with a search range of 8 is represented by E_{SR8}. Experiments are conducted by changing the threshold value TH_i in the HEVC encoder using GPU_ASR. There are three thresholds used for each search range, and the encoder to which each threshold value is applied is denoted by E_{ASRD_TH1}, E_{ASRD_TH2}, and E_{ASRD_TH3}. **Table 2** shows the threshold used in this work. Four QP values are used: 22, 27, 32, and 37. Among the test sequences used in MPEG, Class B (1920 × 1080) and Class C (832 × 480) sequences are used for the experiments.

The performance and the coding efficiency of GPU_ASR are measured using time reduction rate (TR) and BD-bitrate [37], respectively:

$$TR = \frac{T_{\text{ref}} - T_{\text{test}}}{T_{\text{ref}}} \times 100 \ (\%), \tag{7}$$

where T_{ref} and T_{test} represent the execution time of the reference and the compared algorithm, respectively. This performance is denoted by IME-TR in this paper. **Table 3** shows coding efficiencies and IME-TRs of E_{SR8} against E_{SR16}. This table shows the coding efficiency reduction and the time reduction rate when the search range is changed from 16 to 8, and therefore corresponds to the lower limit of the proposed algorithm. When the search range is changed to 8, the average BD-bitrate is increased by 1.8% and the reduction rate of the IME time is 42.2%. Among nine test streams, BD-bitrates are increased to more than 3% in BasketballDrive, BasketballDrill, and RaceHorses. The coding efficiency of the chroma component is lower than that of the luma component.

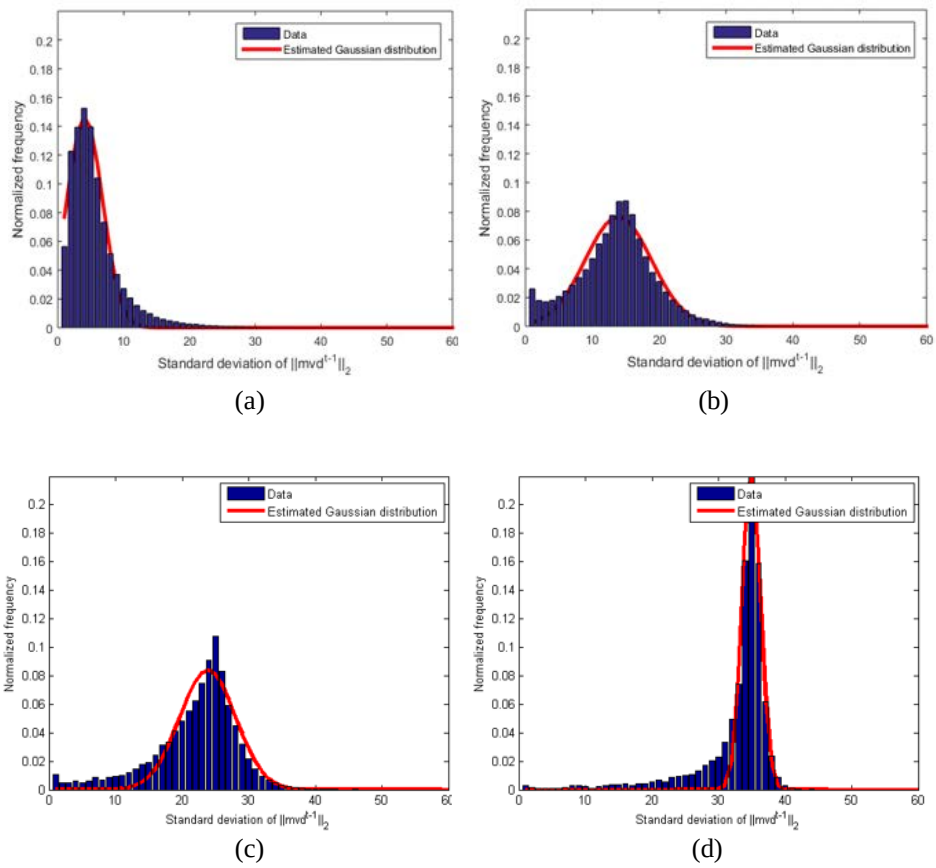


Fig. 8. Likelihoods for the search range of 16, $s^{t-1}=16$: (a) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=5, s^{t-1})$, (b) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=15, s^{t-1})$, (c) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=25, s^{t-1})$, and (d) $p(\sigma_{mvd}^{t-1} \mid \sigma_{mvd}^{t}=35, s^{t-1})$

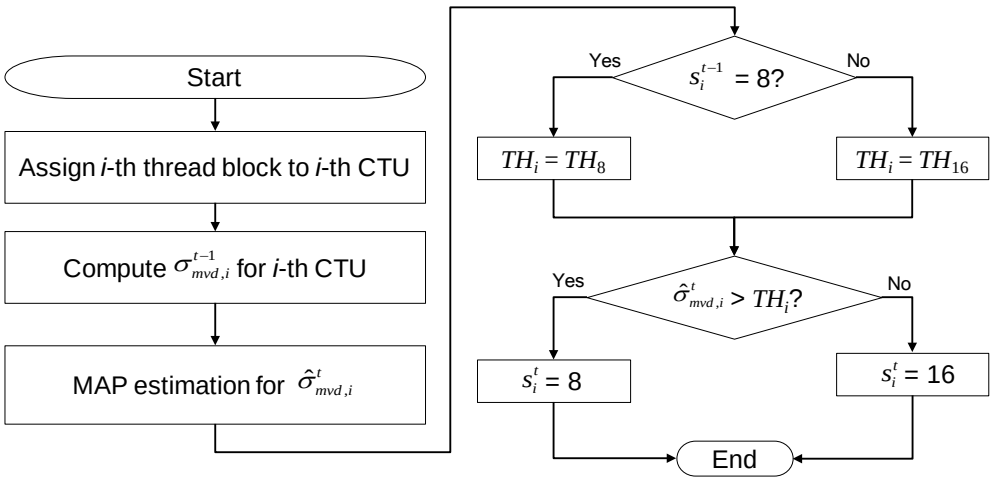


Fig. 9. Flow chart of deciding an adaptive search range

Table 2. Threshold values for adaptive search range decision

Search range	TH_i		
	E_{ASRD_TH1}	E_{ASRD_TH2}	E_{ASRD_TH3}
8	1	4	9
16	1	5	13

Tables 4 and **5** show the coding efficiencies and the time reduction rates of E_{ASRD_TH1}, E_{ASRD_TH2}, E_{ASRD_TH3}, and E_{SR8} with respect to E_{SR16}, respectively. The encoding efficiency is based on the average BD-bit rate for the three components Y, U, and V. E_{ASRD_TH1} shows a 15.4% time reduction rate on average without loss of coding efficiency. In BQTerrace, the coding efficiency is increased by 0.5% and the time reduction rate is 22.3%. The encoding efficiency tends to decrease from E_{ASRD_TH1} to E_{ASRD_TH3}, but the time reduction rate as well as the possibility of selecting $s_i^t = 8$ tends to increase. However, there is one exception case of BasketballDrive even though the difference is very small. The threshold value in GPU_ASR determines the trade-off between the coding efficiency reduction and the time reduction rate as a user parameter, so that an appropriate value can be selected according to the application program. **Tables 4** and **5** show that the coding efficiency decreases and the time reduction rate increases as the threshold increases. We use a rate-reduction (RD) curve shown in **Fig. 11** to explain that situation. In the RD curve, the reduction rate of the coding efficiency and the time reduction rate of E_{SR16} and E_{SR8} against to E_{SR16} is mapped to (0, 0) and the upper right end, respectively. Without losing geneality, it can be assumed that the time reduction rate represents the encoding performance. Thus, we can insist that GPU_ASR can enhance encoder performance with small amount of bitrate increase relatively since each point of E_{ASRD_THI} locates above the straight dash line in **Fig. 11**.

In order to evaluate the performance of GPU_ASR in terms of image quality, the bitstreams encoded with E_{SR16} and E_{ASRD_TH3} are decoded to compare the quality of the reconstructed sequences. Two sequences where BD-bitrate degradations are very large, BasketballDrive and RaceHorses, are compared. The average PSNR of each picture is measured in the reconstructed sequences, and pictures with the largest difference between PSNRs are shown in **Fig.'s 12-15** by QP. Both reconstructed sequences have a small PSNR

and very small picture quality differences so that it is difficult to distinguish them from each other.

Table 3. Coding efficiencies and IME-TRs of E_{SR8} against E_{SR16}

Class	Sequence	E_{SR8}				
		BD-bitrate (%)				*IME-TR* (%)
		Y	U	V	Ave.	
Class B	Kimono	0.9	0.8	1.5	1.0	42.0
	ParkScene	0.9	1.2	1.4	1.2	42.2
	Cactus	1.0	1.1	1.0	1.0	41.0
	BasketballDrive	2.6	4.3	3.7	3.5	42.1
	BQTerrace	0.7	1.7	-0.6	0.6	42.7
Class C	BasketballDrill	3.0	4.0	5.0	4.0	42.5
	BQMall	0.6	1.2	0.8	0.8	42.5
	PartyScene	0.4	0.6	0.5	0.5	42.6
	RaceHorses	3.1	4.0	4.2	3.8	42.5
Average (Class B)		**1.2**	**1.8**	**1.4**	**1.5**	**42.0**
Average (Class C)		**1.8**	**2.4**	**2.6**	**2.3**	**42.5**
Average (Total)		**1.5**	**2.1**	**1.9**	**1.8**	**42.2**

Table 4. Coding efficiencies of E_{ASRD_TH1}, E_{ASRD_TH2}, E_{ASRD_TH3} and E_{SR8} with respect to E_{SR16}

Classes	Sequence	AVERAGE BD-BITRATE (%)			
		E_{ASRD_TH1}	E_{ASRD_TH2}	E_{ASRD_TH3}	E_{SR8}
Class B	Kimono	0.1	0.0	0.4	1.0
	ParkScene	0.1	0.2	0.6	1.2
	Cactus	0.0	0.2	0.3	1.0
	BasketballDrive	0.2	0.1	1.0	3.5
	BQTerrace	-0.5	0.4	0.7	0.6
Class C	BasketballDrill	0.1	0.3	1.0	4.0
	BQMall	0.2	0.4	0.8	0.8
	PartyScene	-0.1	0.1	0.4	0.5
	RaceHorses	-0.1	0.4	1.7	3.8
Average (Class B)		**0.0**	**0.2**	**0.6**	**1.5**
Average (Class C)		**0.0**	**0.3**	**1.0**	**2.3**
Average (Total)		**0.0**	**0.2**	**0.8**	**1.8**

Table 6 summarizes the execution time and the time reduction rate of each module of IME according to the ratio that the search range is set to 8 by applying GPU_ASR to Class B sequences. As the selectivity increases from 25% to 100%, the time reduction rate increases by about 10% from 10% to 42.2%. In each module, the SAD module has the highest time reduction rate. This is a module that calculates the SAD value corresponding to all the search positions. Since the number of memory accesses is small and the computational complexity is high, the search range becomes small, and when the amount of computation to be performed is small, high performance can be obtained. The H-SAD module is a hierarchical SAD calculation module that adds several SAD values and stores them in memory. Therefore, the computational complexity is lower and the number of memory accesses is larger than that of the SAD module. WCPR is a process of finding the minimum value among all cost values, and performs only comparison operations. Since the data in memory moves frequently, it has a high memory access overhead, so it has low speed performance.

Table 5. IME-TRs of E_{ASRD_TH1}, E_{ASRD_TH2}, E_{ASRD_TH3} and E_{SR8} with respect to E_{SR16}

Class	Sequence	IME-TR (%)			
		E_{ASRD_TH1}	E_{ASRD_TH2}	E_{ASRD_TH3}	E_{SR8}
Class B	Kimono	12.5	28.2	38.3	42.0
	ParkScene	26.5	34.7	39.1	42.2
	Cactus	23.7	32.8	37.4	41.0
	BasketballDrive	9.9	22.1	33.4	42.1
	BQTerrace	22.3	35.1	39.3	42.7
Class C	BasketballDrill	12.3	18.6	26.6	42.5
	BQMall	12.7	24.9	30.9	42.5
	PartyScene	16.0	25.6	30.3	42.6
	RaceHorses	2.9	13.6	25.5	42.5
Average (Class B)		**19.0**	**30.6**	**37.5**	**42.0**
Average (Class C)		**11.0**	**20.7**	**28.3**	**42.5**
Average (Total)		**15.4**	**26.2**	**33.4**	**42.2**

Table 6. Time reduction of each module in IME according to the selection rate of search range of 8 for Class B sequences

Steps	0%	25%		50%		75%		100%	
	Time (ms)	Time (ms)	TR (%)	Time (ms)	TR (%)	Time (ms)	TR (%)	Time (ms)	TR (%)
SAD	10.0	8.5	14.4	7.0	29.3	5.7	42.7	4.2	57.5
H-SAD	5.8	5.2	10.7	4.6	20.3	4.0	30.6	3.4	40.8
WCPR	9.4	8.7	6.8	8.2	13.0	7.5	20.0	6.9	26.8
Total	25.1	22.4	10.7	19.8	21.2	17.2	31.4	14.5	42.2

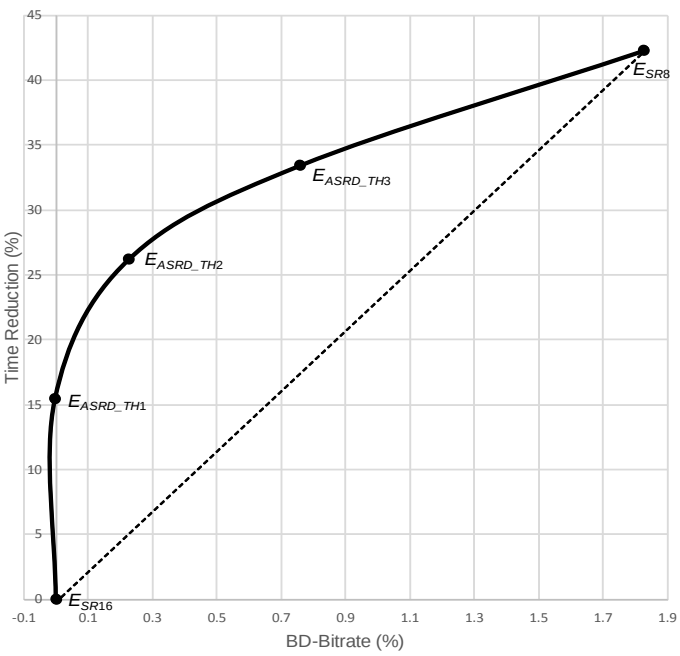


Fig. 11. Rate-reduction curve of adaptive search range decision

(a) (b)

Fig. 12. Decoded pictures of E_{SR16} and E_{ASRD_TH3} for *BasketballDrive* (QP : 22): (a) E_{SR16} (Y : 38.88 dB, U : 43.36 dB, V : 44.88 dB) and (b) E_{ASRD_TH3} (Y : 38.87 dB, U : 43.34 dB, V : 44.81 dB)

(a) (b)

Fig. 13. Decoded pictures of E_{SR16} and E_{ASRD_TH3} for BasketballDrive (QP : 37): (a) E_{SR16} (Y : 39.69 dB, U : 42.34 dB, V : 42.85 dB) and (b) E_{ASRD_TH3} (Y : 32.61 dB, U : 39.85 dB, V : 39.62 dB).

(a) (b)

Fig. 14. Decoded pictures of E_{SR16} and E_{ASRD_TH3} for RaceHorses (QP : 22): (a) E_{SR16} (Y : 39.37 dB, U : 41.36 dB, V : 42.64 dB) and (b) E_{ASRD_TH3} (Y : 39.39 dB, U : 41.31 dB, V : 42.48 dB)

(a) (b)

Fig. 15. Decoded pictures of E_{SR16} and E_{ASRD_TH3} for RaceHorses (QP : 37): (a) E_{SR16} (Y : 30.27 dB, U : 35.21 dB, V : 36.98 dB) and (b) E_{ASRD_TH3} (Y : 30.24 dB, U : 35.05 dB, V : 36.71 dB)

5. Conclusion

This paper proposed an adaptive search range decision algorithm to reduce the complexity of integer pixel motion estimation using global search: GPU_ASR. GPU_ASR reduced the complexity of integer pixel motion estimation while maintaining the coding efficiency by adaptively determining the search range according to the motion feature of the CTU.

There were three main contributions as compared to our previous work. First, we defined the motion feature as the standard deviation of MVD values in a CTU, which was discretized to easily compute the estimator in the GPU. Second, the MAP estimater which theoretically estimated the motion feature of the current CTU using the motion feature of a temporally adjacent CTU and its SR was proposed, which removed data dependency. Thus, the SR for the current CTU was parallelly determined. Finally, the values of the prior distribution and the likelihood for each discretized motion feature were computed in advance and stored at a look-up table to further save the computational complexity. The threshold for each motion feature value was used as the user parameter that controled the coding efficiency and the speed-up performance.

When GPU_ASR was applied to the HM 10.0 encoder, the average BD-bitrate increases of 0%, 0.2%, and 0.8% were respectively obtained with 15.4%, 26.2% and 33.4% reduction of the integer pixel motion estimation time by adjusting the threshold value corresponding to the user parameter. Using the rate-reduction curve defined in this paper, it was confirmed that GPU_ASR was more advantageous than the loss in terms of the time reduction rate and the reduction in the coding efficiency. GPU_ASR can be used in combination with an conventional motion estimation method for HEVC since it works without any dependence on major modules in the HEVC encoder.

Acknowledgements

This work was partly supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-2016-0-00288) supervised by the IITP(Institute for Information & communications Technology Promotion) and the work reported in this paper was conducted during the sabbatical year of Kwangwoon University in 2016.

References

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012. Article (CrossRef Link).

[2] ITU-T, High Efficiency Video Coding, Rec. ITU-T H.265 and ISO/IEC 23008-2, Oct. 2014. Article (CrossRef Link).

[3] G. J. Sullivan, J. M. Boyce, Y. Chen, J.-R. Ohm, C. A. Segall, A. Vetro, "Standardized Extensions of High Efficiency Video Coding," *IEEE Journal on Selected Topics in Signal Processing*, vol. 7, no. 6, pp 1001-1016, Dec. 2013. Article (CrossRef Link).

[4] W.-J. Han et al., "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1709–1720, Dec. 2010. Article (CrossRef Link).

[5] G. Corrêa, P. Assunção, L. Agostini, and L. A. da Silva Cruz, "Performance and computational complexity assessment of high-efficiency video encoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1899–1909, Dec. 2012. Article (CrossRef Link).

[6] F. Bossen, B. Bross, K. Suhring and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685-1696, Dec. 2012. Article (CrossRef Link) .

[7] M. Viitanen, J. Vanne, T. D. Hämäläinen, M. Gabbouj and J. Lainema, "Complexity analysis of next-generation HEVC decoder," in *Proc. of 2012 IEEE International Symposium on Circuits and Systems*, pp. 882-885, May 2012. Article (CrossRef Link).

[8] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block partitioning structure in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1679–1706, Dec. 2012. Article (CrossRef Link).

[9] J.-L. Lin, Y.-W. Chen, Y.-W. Huang, and S.-M. Lei, "Motion vector coding in the HEVC standard," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 957–968, Dec. 2013. Article (CrossRef Link)

[10] J. Vanne, M. Viitanen, and T. D. Hämäläinen, "Efficient mode decision schemes for HEVC inter prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 9, pp. 1579–1593, Sep. 2014. Article (CrossRef Link).

[11] Y.-G. Lee, "Early search termination for fast motion estimation," *EURASIP Journal on Image and Video Processing*, 2015(29), Sep. 2015. Article (CrossRef Link).

[12] R. Khemiri, N. Bahri, F. Belghith, F. Sayadi, M. Atri, and N. Masmoudi, "Fast motion estimation for HEVC video coding," in *Proc. of 2016 IEEE International Image Processing, Applications and Systems*, pp. 1-4, Nov. 2016. Article (CrossRef Link).

[13] Z. Pan, J. Lei, Y. Zhang, X. Sun, and S. Kwong, "Fast motion estimation based on content property for low-complexity H.265/HEVC encoder" *IEEE. Transactions on Broadcasting*, vol. 62, no. 3, pp. 675-684, June 2016. Article (CrossRef Link).

[14] S.-H. Yang, J.-Z. Jiang, and H.-J. Yang, "Fast motion estimation for HEVC with directional search," *Electron. Lett.*, vol. 50, no. 9, pp. 673–675, Apr. 2014. Article (CrossRef Link)

[15] H. Kibeya, F. Belghith, M. A. B. Ayed, and N. Masmoudi, "Adaptive motion estimation search window size for HEVC standard," in *Proc. of 2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, Dec. 2016. Article (CrossRef Link).

[16] T.-K. Lee, Y.-L Chan, and W.-C. Siu, "Adaptive search range for HEVC motion estimation based on depth information," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 10, pp. 2216–2230, Oct. 2017. Article (CrossRef Link).

[17] K. Singh, S. R. Ahamed, “Computationally efficient motion estimation algorithm for HEVC,” *Journal of Signal Processing Systems*, Springer, vol. 90, no. 12, pp. 1713–1727, Dec. 2018. Article (CrossRef Link).

[18] Y. Tian, J. Yan, S. Dong, and T. Huang, “PA-Search: Predicting units adaptive motion search for surveillance video coding,” *Computer Vison and Image Understanding, Elsevier*, vol. 170, pp. 14–27, May 2018. Article (CrossRef Link).

[19] D. Lee, C. -B. Ahn, Y. Chung, and S.-J. Oh, “An adaptive search range decision algorithm for parallel motion estimation,” in *Proc. of 2018 International Workshop on Advanced Image Technology (IWAIT)*, May 2018. Article (CrossRef Link).

[20] 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, 2013. Article (CrossRef Link).

[21] 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 *Proc. of IEEE Visual Communications and Image Processing (VCIP)*, pp. 1-5, 2013. Article (CrossRef Link).

[22] 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. Article (CrossRef Link).

[23] 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 on multicore graphics processors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 11, pp. 1692-1703, 2009. Article (CrossRef Link).

[24] 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. Article (CrossRef Link).

[25] S. Kim, D. Lee, Y. Ahn, T. J. Hwang, D. Sim and S. J. Oh, "DCT-based interpolation filtering for HEVC on graphics processing units," in *Proc. of the International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pp. 155-158, 2013.

[26] W. N. Chen and H. M. Hang, "H.264/AVC motion estimation implementation on compute unified device architecture (CUDA)," in *Proc. of the IEEE International Conference on Multimedia and Expo (ICME)*, pp. 697-700, 2008. Article (CrossRef Link) .

[27] Z. Jing, J. Liangbao and C. Xuehong, "Implementation of parallel full search algorithm for motion estimation on multi-core processors," in *Proc. of IEEE International Conference on Next Generation Information Technology*, pp. 31-35, 2011. Article (CrossRef Link).

[28] R. Rodríguez-Sánchez, J. L. Martínez, G. Fernández-Escribano, J. M. Claver and J. L. Sanchez, "Reducing complexity in H. 264/AVC motion estimation by using a GPU," in *Proc. of IEEE 13th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1-6, 2011. Article (CrossRef Link).

[29] D. K. Lee and S. J. Oh, “Variable block size motion estimation implementation on compute unified device architecture (CUDA),” in *Proc. of the IEEE International Conference on Consumer Electronics*, Las Vegas, pp. 635-636, Jan. 2013. Article (CrossRef Link).

[30] D. Lee, D. Sim and S. J. Oh, "Integer-pel motion estimation for HEVC on compute unified device architecture (CUDA)," *IEIE Transactions on Smart Processing and Computing*, vol. 3, no. 6, pp. 397-403, 2014. Article (CrossRef Link)

[31] X. Jiang et al., "High Efficieny Video Coding (HEVC) Motion Estimation Parallel Algorithms on GPU," in *Proc. of the IEEE International Conference Consumer Electronics*-Taiwan *(ICCE*-Taiwan*)*, pp. 115-116, 2014. Article (CrossRef Link).

[32] S. Radicke, J. Hahn, C. Grecos, & Q. Wang, “A highly-parallel approach on motion estimation for high efficiency video coding (HEVC),” in *Proc. of IEEE Int. Conf. on Consumer Electronics*, pp.187-188, 2014. Article (CrossRef Link).

[33] D. K. Lee, D. Sim, K. Cho and S. J. Oh, “Fast motion estimation for HEVC on graphic processing unit (GPU),” *Journal of Real-Time Image Processing, Springer*, vol. 12, issue 2, pp. 549-562, Aug. 2016. Article (CrossRef Link).

[34] Y.-G. Xue, H.-Y. Su, J. Ren, M. Wen, C.-Y. Zhang, and L.-Q. Xiao, “A highly parallel and scalable motion estimation algorithm with GPU for HEVC,” *Scientific Programming*, Hindawi, vol. 2017, pp. 1-15, Oct. 2017. Article (CrossRef Link).

[35] F. Takano, H. Igarashi, and T. Moriyoshi, “4K-UHD real-time HEVC encoder with GPU accelerated motion estimation,” in *Proc. of IEEE Int. Conf. on Image Processing (ICIP)*, Sep. 2017. Article (CrossRef Link).

[36] C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 16 (HM16) Improved Encoder Description Update 6," *Joint Collaborative Team on Video Coding (JCT-VC), JCTVC-X1002*, Jun. 2016. Article (CrossRef Link).

[37] T.K. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J. Ohm, and G. Sullivan, “Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 76-90, Jan. 2016. Article (CrossRef Link).

Seoung-Jun Oh was born in Seoul, Korea, in 1957. He received both the B.S. and the M.S. degrees in electronics engineering from Seoul National University, Seoul, in 1980 and 1982, respectively, and Ph.D. degree in electrical and computer engineering from Syracuse University, New York, in 1988. In 1988, he joined ETRI, Daejeon, Korea, as a senior research member. Later, he was promoted to a Program Manager of Multimedia Research Session, ETRI. Since 1992, he has been a professor of Department of Electronics Engineering, Kwangwoon University, Seoul, Korea. He has been a chairman of SC29-Korea since 2001. His research interests include image/video processing, computer vision, real-time video processing, and machine/deep learning for computer vision.

Dongkyu Lee was born in Seoul, Korea. He received his B.S., M.S. and Ph.D. degrees in Electronic Engineering from Kwangwoon University, Seoul, Korea, in 2012, 2014 and 2017, respectively. He has been a research engineer in Hyundai Mobis, Korea since 2017. His research interests are image/video processing, video compression, and machine learning for object tracking.