IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 10, 2018, accepted October 30, 2018, date of publication November 6, 2018,
date of current version December 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2879816

A Novel Search Area Data Reuse Method for Memory Access Optimization of Fast Search Motion Estimation

HONGJIE LI[1], YANHUI DING[1], WEIZHI XU[1], HUI YU[2], AND FANGAI LIU[1]
[1]School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China
[2]School of Management Science and Engineering, Shandong Normal University, Jinan 250014, China
Corresponding authors: Yanhui Ding (yanhuiding@126.com) and Weizhi Xu (xuweizhi@sdnu.edu.cn)

This work was supported in part by the Primary Research & Development Plan of Shandong Province under Grant 2017GGX10112, in part by the NNSF of China under Grant 61602285, Grant 61772321, Grant 61520106005, and Grant 61602284, in part by the Shandong Natural Science Foundation under Grant ZR2015FQ009 and Grant ZR2016FP07, and in part by the Project of Shandong Province Higher Educational Science and Technology Program under Grant J16LN05.

ABSTRACT Motion estimation (ME) is a time-consuming algorithm to find a matching block in the search area for video applications, such as video compression. Motion estimation algorithm includes full search algorithm and fast search algorithm. If the width of the motion estimation algorithm search area is twice the size of the block, half of the search area for adjacent blocks overlaps. In view of this, this paper proposes a search area data reuse method for fast search motion estimation. With the proposed data reuse implementation, only the first block needs to read the data of the entire search area from the off-chip memory. The other blocks only need to read half the original search area. In this way, time of access to external memory is reduced, and running time of the algorithm is also decreased. Experimental results of diamond search show that the search area data reuse methods can reduce the running time by 40% to 60% compared with the algorithm of no use data reuse, and it can also reduce the power consumption by 62% to 73%. Compared with other methods in the literature, the proposed method also performs better on running time and power consumption.

INDEX TERMS Motion estimation, search area data reuse method, diamond search, CUDA, power consumption.

I. INTRODUCTION

Motion Estimation is an important part of video applications such as video coding and frame rate up-conversion. In video coding, the function of motion estimation is to eliminate the time redundancy between adjacent frames. In frame rate up-conversion, the interpolation method based on motion estimation is one of the most effective methods. ME requires very high computational complexity, and it usually takes up most of the running time for these video applications. For example, in video coding, motion estimation is closely related to the complexity of the coding. H.264/AVC [1] is highly compressed digital video coding standard, and multi-view video coding (MVC) is a coding standard extension of H.264/AVC. The complexity of motion estimation and disparity estimation is about 99% of the whole MVC [2]. The H.264 JM encoder shows that the integer ME (IME) consumes nearly 60% of the total encoder time [3]. Therefore, it is necessary to optimize the motion estimation to reduce its running time. This is very important for real-time video applications.

Motion estimation is the process of finding the matching block. The most similar block to the current block is found in the reference frame according to the matching principle. The motion estimation algorithm includes the full search algorithm and the fast search algorithm. Full search motion estimation algorithm [4] is an accurate motion estimation algorithm, but the search time consumption can be several times as fast search motion estimation algorithm. Therefore, full search motion estimation is usually used in comparison with fast search algorithms by running time and accuracy. The fast search algorithms include three step search algorithm [5], new three step search algorithm [6], four step search algorithm [7], hexagon search algorithm [8], etc. Compared with full search, fast search algorithm has the advantage of less running time and the disadvantage of low search precision.

In addition, fast search method is usually irregular. The irregular memory access of the fast search motion estimation algorithm destroys the locality of the data. Thus, the fast search algorithm still takes up a lot of time and needs to be optimized.

Data reuse is a very effective way to improve the performance of motion estimation. In previous studies, the data reuse method has been widely used in full search algorithm, but it was seldom considered for fast search motion estimation algorithm. Although the fast search algorithm is faster than the full search algorithm. It still takes a lot of time [9]. The main reason is the irregular memory access of the fast search algorithm, which increases the off-chip memory traffic.

In this paper, a new search area data reuse method for fast search motion estimation is proposed for memory access optimization. The method reuses the overlapping data between the search areas of two adjacent blocks. The overlapping data are further divided into two parts, a definite data reuse area and a possible data reuse area. The proposed reuse methods can effectively reduce the number of memory access. And the experimental results on GPU platform show that the running time of the algorithm can be reduced by 40% to 60% after using the proposed methods of search area data reuse. Furthermore, the proposed methods can also reduce the power consumption by 62% to 73%.

The rest of this paper is organized as follows. Section 2 introduces the related work of data reuse for motion estimation. And Section 3 proposes the new search area data reuse methods for fast search motion estimation. Then, Section 4 gives the implementation of the reuse method in the shared memory of GPU. The experimental results are presented in Section 5. Finally, a brief conclusion is given in Section 6.

II. RELATE WORK

A. DATA REUSE OF MOTION ESTIMATION ALGORITHM

Data reuse is an effective method to optimize the performance of motion estimation [10], such as data reuse by selective search area reuse algorithm [11] and single reference frame multiple current macro-blocks scheme [12]. Shim *et al.* [13] proposed the search area selective reuse algorithm to reuse the data of the existing frame in memory, which reduces the memory access time effectively. Tuan *et al.* [14] studied the data reuse attributes of full-search block-matching (FSBM), analyzed the memory bandwidth requirement of ME, and explored the problem of data reuse. Four levels (A, B, C and D) were defined according to the reuse degree of the previous frame access. The effects of different levels of data reuse are different. Through the highest level of data reuse (Level D), one access to frame data can be achieved and the memory bandwidth requirement can be greatly reduced. Depending on the method of Level C data reuse, Chen *et al.* [15] proposed a scheme of n-stitched zigzag scan for Level C+, which can effectively reduce the memory bandwidth requirement and solve the corresponding conflict. The method proposed in the literature [25] is based on full-search motion estimation, and inter-frame data reuse method with better storage bandwidth is proposed.

The data reuse method has a beneficial effect on the full search motion estimation, but there is little research on the fast search motion estimation. For the fast search motion estimation, Kim and Sunwoo [16] proposed the sub-region partitioning method by studying the influence of the search order, and using the optimized order to improve the reuse of each sub-region partitioning in the sub-region, effectively reducing redundant data loading. A run-time prediction algorithm is proposed to effectively identify the most-frequently accessed memory regions in the search window(s) for processing individual coding tree units (CTUs) [30]. Kim and Sunwoo [16] proposed the method of 2-D data reuse supported by horizontal and vertical reference SRAMs.

This paper mainly studies the data reuse method of fast search motion estimation, and puts forward a novel method of search area data reuse. The method discovers overlapping data of adjacent search areas for reuse, so that the two adjacent current blocks can load as little data as possible. At the same time, according to the algorithm search process, the overlapping data are further divided into two parts: a definite data reuse area and a possible data reuse area. Finally according to the characteristics of different fast search methods, different reuse data are selected to improve the performance of the algorithm.

B. GPU/CUDA PLATFORM

The cache of the CPU is a high-speed, small-capacity memory in the hierarchy of the computer storage system. When the CPU accesses memory, the control system automatically calculates the data that are frequently used in the memory, and then stores these frequently used data in the cache for reuse. However, the cached data can only be determined automatically by the hardware and cannot be programmed. Taking into account this feature of the CPU, this paper chooses to conduct research and experiments on GPU, which can set the reuse data that need to be cached. The main idea of this paper focuses on data reuse instead of parallelization. The proposed data reuse method is suitable not only for GPU, but also for other platforms (for example, ASIC). GPU is adopted because it has programmable shared memory to facilitate the implementation of the proposed data reuse method.

Universal coding requires a large amount of memory for hardware implementation. However, it is not easy to implement these algorithms in hardware [17]. The compute unified device architecture (CUDA) launched by NVIDIA can combine CPU with GPU to realize these algorithms [18]. The CUDA program contains kernel functions and serial code, and the kernel function is called by the CPU and loaded onto GPU as the co-processor [19]. Typically, the kernel function is executed in parallel by a thread block. Many threads exist in the same thread block, and these threads have the same

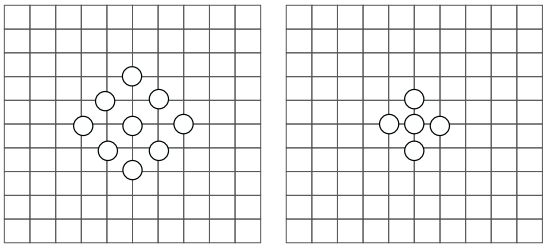


FIGURE 1. Large diamond search pattern and small diamond search pattern.

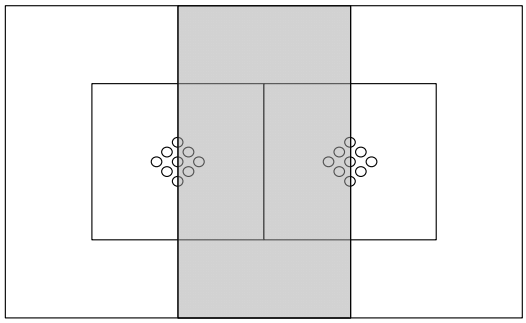


FIGURE 2. Search areas of two adjacent current blocks.

instruction address. The program can be executed in parallel, and different threads can communicate through shared memory (SM) and synchronize with each other using the barrier operation. CUDA contains different types of memories such as global memory, registers, shared memory, and constant memory. The characteristics of different memories are different. Each thread has its own register. Each thread blocks has a shared memory. All threads can access the global memory. Global memory is off-chip memory and its access overhead is considerably.

This paper chooses to use shared memory to implement the proposed method of search area data reuse. The reason is that shared memory can be read and written by all threads in the same thread block programmatically, and the proposed search area data reuse method can be realized. Access to shared memory is almost as fast as a register, so it is possible to optimize the performance of a fast search algorithm by using shared memory for data reuse.

C. DIAMOND SEARCH

Diamond search is a typical quick search method [20], so this paper selects this algorithm to verify the proposed search area data reuse method. However, the proposed method can also be used by other fast search methods. The diamond search is based on a diamond template to find the block that best matches the current block. Diamond search attempts to generate a circular search by considering almost all possible directions to search for motion vectors. The diamond search uses two search templates, namely a large diamond search pattern (LDSP) and a small diamond search pattern (SDSP) [21]. As shown in Figure 1, these search templates all come from search points within a circle with a radius of 2 pixels. The LDSP contains nine search points to form a large rhomboid, and the SDSP consists of five search points to form a smaller rhombus. These two templates make full use of the center offset feature of the motion vector, and can find the matching block quickly and accurately. Diamond search starts with a large diamond search template, until the minimum point of the search is the center point of the search, and then the small diamond search template is used. Otherwise, the large diamond search template is always used [22]. Diamond search using LDSP and SDSP templates can improve the efficiency and accuracy of the algorithm.

In recent years, there have been some improvements to the diamond search algorithm. Tham *et al.* [23] proposed an unrestricted center-biased diamond search algorithm (UCBDS) with good accuracy and efficiency. Shah and Dalal [24] proposed a hardware efficient double diamond search algorithm (HEDDS), to more quickly find the best matching block in the search window, and minimize the number of iterations of the search. Thereby this algorithm can reduce the memory read and time consumption, and provides a fairly good coding quality and efficiency.

III. SEARCH AREA DATA REUSE METHOD FOR FAST SEARCH MOTION ESTIMATION ALOGRITH

Taking diamond search algorithm as an example, this section introduces the method of search area data reuse for fast search motion estimation. The new method can effectively reduce the number of accesses to off-chip memory and improve the performance of the algorithm. The search areas of fixed block size ME and variable block size ME are the same, so the proposed search area data reuse method can be applied to both of them. The proposed method is suitable not only for H.264 but also for HEVC. The reason is that there is little difference between motion estimation algorithms in H.264 and HEVC.

A. DATA REUSE BETWEEN THE SEARCH AREAS

Data reuse of the fast search motion estimation algorithm can be achieved by reusing the overlapping data between the search areas of the current adjacent blocks. Taking the diamond search algorithm as an example (Fig 2.), the block size is 16×16 and the search area size is 32×32. As shown in Fig 2, the smaller blocks are two adjacent blocks, the small circle represents the diamond search strategy for each search point, and the large blocks are the search areas of the two adjacent blocks. The search area of the left block contains half the search area on the right, which is the overlapping area between the two search areas. By reusing the data of the overlapping area, searching for two adjacent blocks can read the data from the off-chip memory to the on-chip memory only once instead of twice, thereby reducing the number of off-chip memory accesses. This will greatly reduce the

algorithm running time and optimize the performance of the algorithm.

B. POSSIBLE+DEFINITE SEARCH AREA DATA REUSE METHOD

Unlike the full search motion estimation, the fast search motion estimation has uncertain search steps. Because the first step of the fast search motion estimation algorithm is fixed (take the corresponding location of the current block as the search center), there is a definite data reuse area between the search areas of the two current blocks (Fig. 3). However, the search center or search direction of the second step is uncertain. When the fast search algorithm performs the second step, the search center is determined by the result of the first step. Therefore, there is also a possible data reuse area. This paper divides the reusable data of the fast search motion estimation algorithm into the possible reuse data and definite reuse data according to the algorithm's search process to improve the performance of the algorithm. Use of possible and definite reuse data areas for data reuse is known as Possible + Definite search area data reuse method.

Fig 3. give the search points or search areas of two current blocks. Numbers in the small circles are the search steps of diamond search. There must be a definite reuse data area wherever the search direction of the adjacent current block is, because the search center of the first step is fixed. If the second step of the two current blocks goes as in Fig. 3. There will be a possible reuse data area. When the on-chip memory size is large enough, we can put both definite reuse data and possible data reuse data on chip, which will further reduce the off-chip memory traffic.

IV. IMPLEMENTATION OF SEARCH AREA DATA REUSE METHOD IN MEMORY

In the case of diamond search, we suppose the block size is 16×16 and the search area size is 32×32 in Fig. 4. The left block part is the reusable part of the first two adjacent blocks. Therefore, when the algorithm searches for the matching block of the first two blocks, the data of the overlapping region can be read only once from the off-chip memory, and then the two adjacent blocks can use the overlapping data together. The search area overlaps of the first three current blocks the algorithm searches are also shown in Fig 4. The left block rectangle represents the overlapping search area between the first two current blocks. And the gray rectangle on the right represents the overlapping search area of the second and third current blocks. When the algorithm searches the matching block of the first three current blocks, the overlapping areas are two different rectangles. When the current fourth block starts its search, the overlapping area will slide to the right accordingly. In order to store the overlapping data only once, this paper studies the operation process of the algorithm in detail, analyses the possible reuse area, and divides the search process into several cases. The overlapping

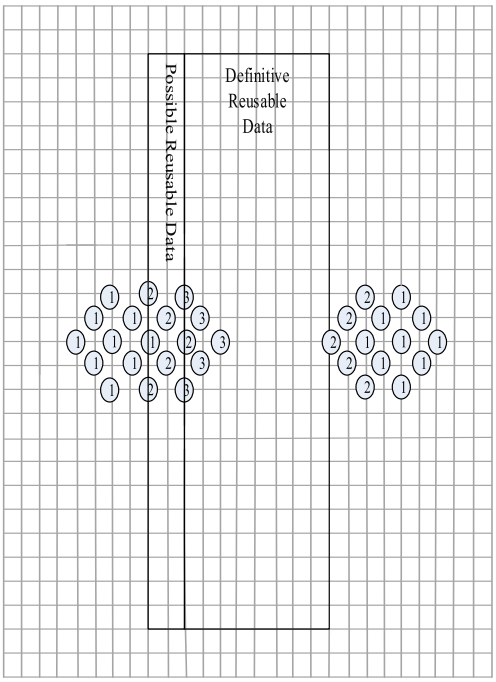


FIGURE 3. Definite data reuse area and Possible data reuse area.

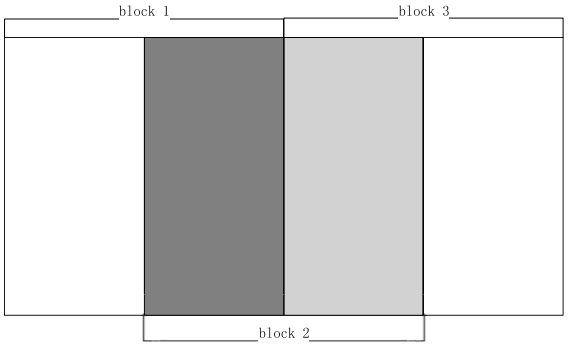


FIGURE 4. The search area reuses of the first three blocks.

data are stored according to the specific situation. As a result, the search area data reuse method is implemented, so that data can be stored only once and used multiple times.

The method of data reuse in the search area is divided into three cases: (1) the method of data reuse in the search area of the first block (2) the method of data reuse in the search area of the even block (3) the method of data reuse in the search area of the odd block (except the first block).

A. METHOD OF DATA REUSE IN SEARCH AREA OF THE FIRST BLOCK

The storage of the first block is simple. After the search of the first block ends, the reusable data in the latter part are stored for the search of the second block. Shared memory (SM) on GPU is divided into two buffers, buffer1 and buffer2. Fig. 5 represents the whole SM, the left rectangle represents buffer1, and the right rectangle represents buffer2. The circle1 represents the front half data of the first block's search

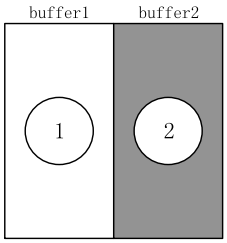


FIGURE 5. Storage of the first current block's search area in SM.

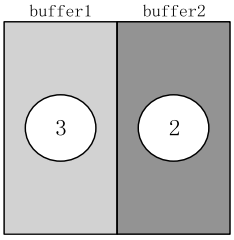


FIGURE 6. Storage of the second current block's search area in SM.

area. The circle2 represents the data in the latter half of the first block's search area, which is also the reusable data of the second block.

B. METHOD OF DATA REUSE IN SEARCH AREA OF THE EVEN BLOCK

To explain how to store the search area of even blocks in SM, we can take the search area of the second block as an example. Fig. 6 represents the entire SM. Circle2 represents reusable data which are already in buffer2 and not needed to be loaded from off-chip memory. Circle3 represents the data that need to be loaded from off-chip memory to make up another half of the second block's search area. Data in buffer1 and buffer2 combine to form the complete search area of the second block. The search order for the second (even) block is contrary to the search order for the odd block. For the even block, the data in the second half of the search area (Circle2) are searched first, and then the first half of the search area (Circle3).

C. METHOD OF DATA REUSE IN SEARCH AREA OF THE ODD BLOCK

To explain how to store the search area of odd blocks in SM, we can take the search area of the third current block as an example. Fig. 7 shows the whole SM. Circle3 in buffer1 shows the stored reusable data of the second block's search area, and circle4 in buffer2 shows the data that need to be loaded from off-chip memory. Buffer1 and buffer2 combine to form the complete search area of the third block.

In summary, with the proposed data reuse implementation, only the first block needs to read the data of the entire search area from the off-chip memory. The other blocks only need to read half the original search area. In this way, the time of access to the external memory is reduced, and the running time of the algorithm is also decreased.

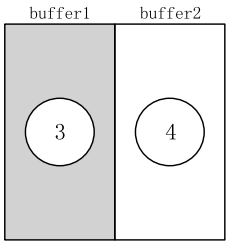


FIGURE 7. Storage of the third current block's search area in SM.

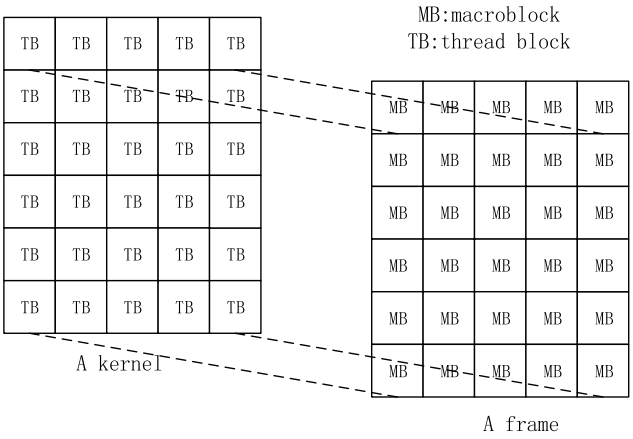


FIGURE 8. Parallel implementation of a kernel. Each thread block in the kernel is in charge of a current macroblock.

D. POSSIBLE + DEFINITE SEARCH AREA DATA REUSE METHOD

The process of possible + definite search area data reuse method is similar to the process of search area reuse method, which also uses the shared memory to store both the definite reuse data and the possible reuse data. This will cause the stored data to be more than the definite data reuse, so it needs to ensure that the storage space of SM is large enough. Storing possible reuse data and definite reuse data in SM further increases the scope of data reuse, compared with only storing definite reuse data in SM. Therefore, the loading of redundant data is reduced and the running time of the program is also decreased. As a result, the introduction of possible reuse data can further improve the performance of the algorithm.

E. PARALLEL IMPLEMENTATION OF ME ON GPU

Parallel implementation of the ME algorithm makes good use of GPU's many-core architecture. Each thread blocks (TB) in the GPU kernel processes one macroblock (MB) in the current frame (Fig. 8). Each thread in a TB is in charge of a search point or reference block and all threads execute parallel search to find the best matching block for the current block (Fig. 9).

V. EXPERIMENTAL RESULTS

A. RUNNING TIME

The experimental environment is the Windows 10 operating system. CPU is Intel G530, and GPU is NVIDIA GeForce

TABLE 1. Comparison between Definite search area data reuse and No data reuse.

Number of frames	Sequence	Size	No data reuse (s)	Definite search area data reuse (s)	Rate(%)
1	Football	176×144	0.073053	0.061476	15.84
	Foreman	352×288	0.254124	0.149723	41.08
	News	352×288	0.253877	0.149040	41.29
	Mobisode2	416×240	0.231372	0.139689	39.62
	Johnny	1280×720	2.514633	1.012222	59.74
	SlideShow	1280×720	2.518903	1.015279	59.69
50	Football	176×144	3.604546	3.011479	16.46
	Foreman	352×288	12.734084	7.491019	41.17
	News	352×288	12.706494	7.490260	41.05
	Mobisode2	416×240	11.554281	6.974855	39.63
	Johnny	1280×720	126.007574	50.887801	59.61
	SlideShow	1280×720	125.999923	50.875493	59.62
100	Football	176×144	7.166847	5.981729	16.53
	Foreman	352×288	25.398413	14.960719	41.09
	News	352×288	25.398413	14.955715	41.11
	Mobisode2	416×240	23.086618	13.925909	39.67
	Johnny	1280×720	252.023172	101.701277	59.64
	SlideShow	1280×720	252.015944	101.738036	59.63

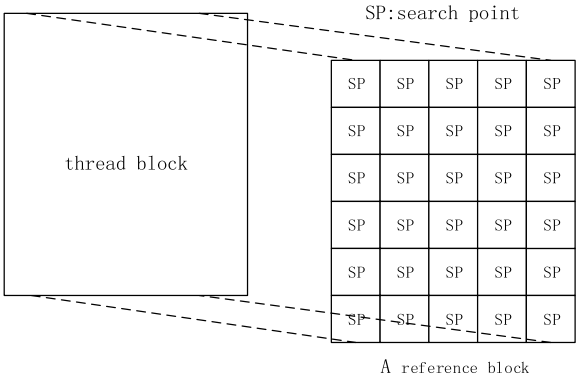


FIGURE 9. Parallel implementation of a thread block. Each thread in the thread block is in charge of a search point or reference block.

GTX 750 Ti with CUDA 8.0. The program is run and debugged in Visual Studio 2013.

The diamond search is used as an example in the experiment. The block size is 64 × 64 and the search area size is 128 × 128. The diamond search algorithm is one typical fast motion estimation algorithm. The diamond search can find matching blocks faster than full search but its search accuracy is not as good as the full search. The proposed search area data reuse method will not affect the search accuracy of diamond search while it can effectively improve the performance of the algorithm.

The experimental results of diamond search are shown in Table 1 and Table 2. Six test sequences of Football, Foreman, News, Mobisode2, Johnny, and SlideShow were tested. In Table 1, Sequence represents different tested sequences. Size represents image size of the test sequence. No data reuse (s) represents the running time in seconds when no data reuse is adopted. In this case, GPU cache is used instead of shared memory because cache is adopted when shared memory is not used for GPU. Definite search area data reuse(s) is that only definite reusable data is considered and reused in shared memory. Possible+Definite search area data reuse(s) means that both definite reusable data and possible reusable data are loaded to share memory for reuse. Rate(%) in Table 1 represent the percentage of running time improvement for Definite search area data reuse method over the method of No data reuse. Rate (%) in Table 2 represent the percentage improvement in time between Possible + Definite search area data reuse method and No data reuse method.

Table 1 show the running time comparison between Definite search area data reuse and No data reuse. In this paper, the running time of 1 frame, 50 frames and 100 frames for

TABLE 2. Comparison between Possible+Definite search area data reuse and No data reuse.

Sequence	Size	No data reuse (s)	Possible+Definite search area data reuse (s)	Rate(%)
Football	176×144	0.073053	0.060625	17.01
Foreman	352×288	0.254124	0.146758	42.24
News	352×288	0.253877	0.146849	42.15
Mobisode2	416×240	0.231372	0.138105	40.31
Johnny	1280×720	2.514633	0.977472	61.12
SlideShow	1280×720	2.518903	0.991458	60.63

diamond search algorithm is tested. The results show that compared with the method of no data reuse, Definite search area data reuse method has a significant improvement in running time. Most of the test sequences have a time reduction of over 40%, and sometimes even have a reduction of over 60%. This proves that the proposed method of data reuse can greatly reduce the running time of the algorithm. The running time of 1 frame, 50 frames, and 100 frames is shown in Table 1. It is found that time reduction rate is not sensitive to the number of frames. Take the sequence of News as an example, time reduction rates are 41.29%, 41.05%, and 41.11% for 1 frame, 50 frames, 100 frames respectively. Besides, time reduction rates of Mobisode2 in one frame, 50 frames, and 100 frames are 39.62%, 39.63%, and 39.67%, respectively. This indicates that the optimization result of the data reuse algorithm proposed in this paper is not related to the number of frames used. From the experimental results in Table 1, it is found that different resolutions show different time reduction rates with the same number of frames. For example, the time reduction rate of News (352 × 288) is 41.29%, while Johnny (1280 × 720) is 59.74%. It is proved that as the resolution becomes larger and larger, reuse efficiency of the data reuse method for the sequence becomes higher and higher, and the effect of the data reusing method becomes better and better.

In Table 2, Possible + Definite search area data reuse method and the running time without the data reuse method is given. Because the optimization results of the search area data reuse method are not related to the number of frames used in the experiment, only the results of 1 frame are given in Table 2. The results show that compared with the method of No data reuse, Possible + Definite search area data reuse method can significantly reduce the running time. For example, Johnny with Possible + Definite search area data reuse method has a running time reduction rate of up to 61.12%. At the same time, Johnny with Definite search area data reuse method has a reduction rate of 59.74%. This shows that compared with the Definite search area data reuse method, Possible + Definite search area data reuse method is more efficient for data reuse, and can further shorten the running time of the algorithm.

In summary, experimental results show that the proposed search area data reuse methods (Definite search area data reuse and Possible + Definite search area data reuse) effectively improve the performance compared with No data reuse. Improvements of most test sequences are over 40%, and some sequences even reach 61.12%. For all the sequences, the method of Possible+Definite search area data reuse need less running time than the method of Definite data reuse. This indicates that it is effective to putting possible reuse data in the shared memory besides definite reuse data. Moreover, as the resolution of the test sequence becomes larger, the data reuse method for the sequence becomes more efficient.

B. POWER CONSUMPTION

$$Bandwidth = Ra \times W \times H \times f \tag{1}$$

$$P_{DRAM} = P_{DRAM\_static} + \alpha 1 \times Throughput_{read} + \alpha 2 \times Throughput_{write} \tag{2}$$

Xu *et al.* calculated the storage bandwidth of level A-D and calculated the power consumption of each reuse level for full search motion estimation. According to the formula used by Xu *et al.* [25] (Formula (1)), the storage bandwidth of diamond search is calculated, where Ra is redundant access factor, W is the width of a frame, H is the height of a frame, and f is the frame rate. According to Formula (2), the power consumption of No data reuse, Definite data reuse and Possible + Definite data reuse for diamond search algorithm are calculated. Because the power consumption difference between different data reuse methods is mainly caused by the difference of access memory mode, this paper only tests the power consumption of DRAM according to Formula (2) [26].

For different data reuse methods, the operations for DDRx of CPU are the same, including sending the images to GPU and receiving MVs from GPU. Therefore, only GPDDRx for GPU is considered in power model. For different data reuse levels, it is assumed that the power consumption of PDRAM_static and $\text{Throughput}_{\text{write}}$ is the same. Therefore, this paper only compares DRAM reading power, where $\alpha 1 = 1.12\text{Watt/(GB/s)}$, and $\text{Throughput}_{\text{read}}$ is equal to the Bandwidth in Formula (1). The video resolution is 1080 p,

TABLE 3. comparison with other methods.

Sequence	[16] (s)	[30](s)	[31](s)	Definite data reuse (s)	Possible+Definite data reuse (s)
Football	0.164104	0.054789	0.064206	0.061476	0.060625
Foreman	0.331108	0.190593	0.223349	0.149723	0.146758
News	0.336936	0.190407	0.223132	0.149040	0.146849
Mobisode2	0.307315	0.173529	0.203352	0.139689	0.138105
Johnny	2.369454	1.885974	2.210110	1.012222	0.977472
SlideShow	2.320201	1.889177	2.213863	1.015279	0.991458

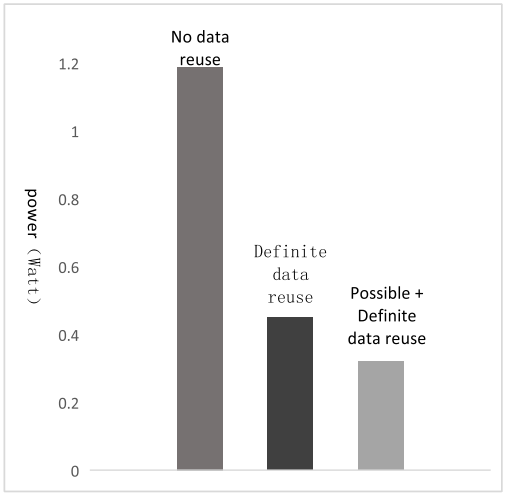


FIGURE 10. Power of data reuse methods (1080p, 30fps, SRH = SRV = 32, N = 16).

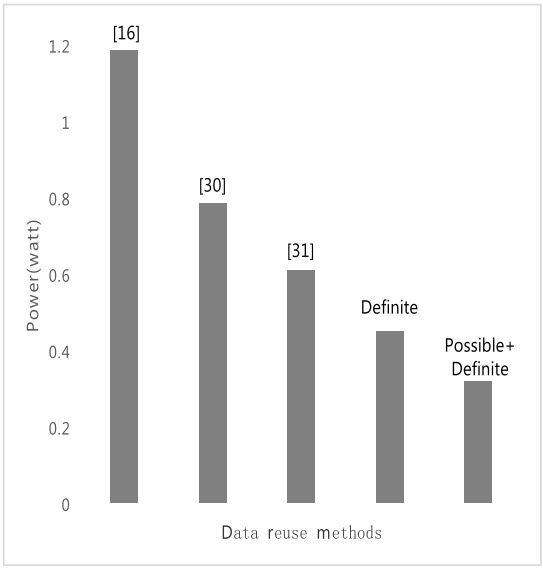


FIGURE 11. Comparison of power consumption (1080p, 30fps, SRH = SRV = 32, N = 16).

the frame rate is 30 fps, the block size is 16 × 16, and the search area is 32 × 32. As shown in Fig. 10, for the diamond search, the power consumption of No data reuse, Definite search area data reuse and Possible + Definite search area data reuse is 1.19 Watt, 0.45 Watt and 0.32 Watt respectively. The power consumption of Definite data reuse, Possible + Definite search area data reuse are significantly lower than that of No data reuse. Definite search area data reuse decreased by 62.18% compared with No data reuse. Possible + Definite search area data reuse has the lowest power consumption of 0.03Watt, reducing the power consumption by 73.10% compared with No data reuse.

C. COMPARISON WITH OTHER METHODS

The existing GPU implementation is mainly oriented to full search motion estimation [27]–[29]. Data reuse methods for full search and fast search are different, and the comparability is not very strong. Full search takes longer to run than diamond search, mainly because the number of search points for full search is much larger than fast search, despite the fact that fast search has irregular memory access. We choose [16], [30], and [31] to compare running time. Table 3 is the result of this method compared with other methods. Compared with the method in [16], the proposed method can greatly reduce the running time of motion estimation. Definite data reuse can reduce the time by 54.54% to 62.53%, and Possible + Definite data reuse can reduce the time by 55.06% to 63.05%. The reason is that the method in [16] can only utilize reusable data between search points while the proposed methods exploit the reusable data between search regions. Compared with the method proposed in [30], the methods proposed in this paper can decrease the running time of the motion estimation by 19.50% to 47.51%. Compared with the method proposed in [31], Definite data reuse can reduce the time by 32.96% to 54.13%, and Possible + Definite data reuse can reduce the time by 34.29% to 55.21%. The main reason is that the method of [31] is utilized for the data reuse between search points, which is not as effective as search area reuse. This shows that the proposed data reuse methods can effectively reduce the running time of the fast search motion estimation algorithm.

According to Formula (1) and (2), the power consumptions of [16], [30], and [31] are calculated, as shown in Fig. 11. Compared with [16], Definite data reuse can reduce the power

consumption by 62.10%, and Possible + Definite data reuse can reduce the power consumption by 73.10%. Compared with [30], the power consumptions of Definite data reuse and Possible + Definite data reuse decrease by 42.74% and 59.28%, respectively. Compared with [31], the data reuse methods proposed in this paper can effectively reduce the power consumption by 26.35% and 47.62% respectively. In conclusion, the proposed methods are more favorable to reduce the power consumption compared with other methods.

VI. CONCLUSION

In this paper, a novel search area data reuse method for fast search motion estimation is proposed. The overlapping data between the search areas of two adjacent current blocks are reused. The overlapping area is further divided into two parts, a definite data reuse area and a possible data reuse area. The proposed reuse methods can effectively reduce off-chip memory traffic. With this method, the running time of the algorithm is reduced, and the performance of the algorithm is further improved on the premise of guaranteeing the accuracy of the algorithm. Experimental results on GPU/CUDA platform show that the proposed methods can effectively reduce the running time and power consumption of the fast search motion estimation algorithm.

REFERENCES

[1] Y. Chen, Y. Zhou, and J. Wen, "Efficient software HEVC to AVS2 transcoding," *Information*, vol. 7, no. 3, p. 53, 2016.

[2] C. Jiang and S. Nooshabadi, "GPU accelerated motion and disparity estimations for multiview coding," in *Proc. IEEE Int. Conf. Image Process.*, Melbourne, VIC, Australia, Sep. 2013, pp. 2106–2110.

[3] X. Wen, O. C. Au, J. Xu, L. Fang, R. Cha, and J. Li, "Novel RD-optimized VBSME with matching highly data re-usable hardware architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 206–219, Feb. 2011.

[4] J.-N. Kim, S.-C. Byun, Y.-H. Kim, and B.-H. Ahn, "Fast full search motion estimation algorithm using early detection of impossible candidate vectors," *IEEE Trans. Signal Process.*, vol. 50, no. 9, pp. 2355–2365, Sep. 2002.

[5] P. Lakamsani, "An architecture for enhanced three step search generalized for hierarchical motion estimation algorithms," *IEEE Trans. Consum. Electron.*, vol. 43, no. 2, pp. 221–227, May 1997.

[6] D. Park, Y. Jang, and J. Lee, "A new fast three step search motion estimation algorithm in H.264," in *Proc. Int. Forum Strategic Technol.*, Ulaanbaatar, Mongolia, Oct. 2007, pp. 541–544.

[7] M. F. So and A. Wu, "Four-step genetic search for block motion estimation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Seattle, WA, USA, vol. 3, May 1998, pp. 1393–1396.

[8] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 5, pp. 349–355, May 2002.

[9] R. Fan, Y. Zhang, and B. Li, "Motion classification-based fast motion estimation for high-efficiency video coding," *IEEE Trans. Multimedia*, vol. 19, no. 5, pp. 893–907, May 2017.

[10] W. Xu, S. Yin, L. Liu, Z. Liu, and S. Wei, "High-performance motion estimation for image sensors with video compression," *Sensors*, vol. 15, no. 8, pp. 20752–20778, Aug. 2015.

[11] H. Shim and C. M. Kyung, "Selective search area reuse algorithm for low external memory access motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 1044–1050, Jul. 2009.

[12] T. C. Chen, C. Y. Tsai, Y. W. Huang, and L. G. Chen, "Single reference frame multiple current macroblocks scheme for multiple reference frame motion estimation in H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 2, pp. 242–247, Feb. 2007.

[13] H. Shim, K. Kang, and C. M. Kyung, "Search area selective reuse algorithm in motion estimation," in *Proc. IEEE Int. Conf. Multimedia Expo*, Beijing, China, Jul. 2007, pp. 1611–1614.

[14] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[15] C.-Y. Chen, C.-T. Huang, Y.-H. Chen, and L.-G. Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 553–558, Apr. 2006.

[16] S. D. Kim and M. H. Sunwoo, "MESIP: A configurable and data reusable motion estimation specific instruction-set processor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1767–1780, Oct. 2013.

[17] K. Marumo, S. Yamagiwa, R. Morita, and H. Sakamoto, "Lazy management for frequency table on hardware-based stream lossless data compression," *Information*, vol. 7, no. 4, 2016, p. 63.

[18] 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 *Proc. IEEE China Summit Int. Conf. Signal Inf. Process. (ChinaSIP)*, Xi'an, China, Jul. 2014, pp. 349–353.

[19] C.-T. Lu and H.-M. Hang, "Multiview encoder parallelized fast search realization on NVIDIA CUDA," in *Proc. Vis. Commun. Image Process. (VCIP)*, Tainan, Taiwan, Nov. 2011, pp. 1–4.

[20] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," in *Proc. Int. Conf. Inf., Commun. Signal Process. Theme, Trends Inf. Syst. Eng. Wireless Multimedia Commun. (ICICS)*, vol. 1, Sep. 1997, pp. 292–296.

[21] N. N. Shah and U. D. Dalal, "Hardware efficient double diamond search block matching algorithm for fast video motion estimation," *J. Signal Process. Syst.*, vol. 82, no. 1, pp. 115–135, 2016.

[22] J. Luo and J. Peng, "An unsymmetrical diamond search algorithm for H.264/AVC motion estimation," in *Proc. Chin. Conf. Image Graph. Technol.*, 2013, pp. 54–65.

[23] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 369–377, Aug. 1998.

[24] N. N. Shah and U. D. Dalal, "Hardware efficient double diamond search block matching algorithm for fast video motion estimation," *Signal Process. Syst.*, vol. 82, no. 1, pp. 115–135, 2016.

[25] W. Xu, H. Yu, D. Lu, F. Liu, and Z. Liu, "A novel data reuse method to reduce demand on memory bandwidth and power consumption for true motion estimation," *IEEE Access*, vol. 6, pp. 10151–10159, 2018.

[26] J. Lin, H. Zheng, Z. Zhu, and Z. Zhang, "Thermal modeling and management of DRAM systems," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2069–2082, Oct. 2013.

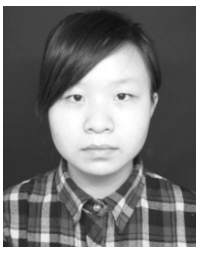
[27] F. Wang, D. Zhou, and S. Goto, "OpenCL based high-quality HEVC motion estimation on GPU," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, 2014, pp. 1263–1267.

[28] A. Heikkinen and L. Fono, "Parallel implementations of motion estimation algorithms using OpenCL," in *Proc. 18th Int. Conf. Digit. Signal Process. (DSP)*, Fira, Greece, Jul. 2013, pp. 1–5.

[29] Z. Liu, Y. N. Lu, T. Liu, and T. W. Yuan, "The improved full search algorithm for motion estimation with GPU accelection," in *Proc. Int. Conf. Comput. Eng. Inf. Syst.*, 2016, pp. 241–244.

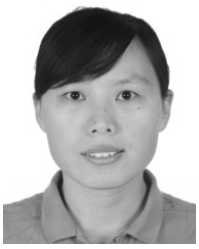
[30] C. Song, L. Ju, and Z. Jia, "Hybrid scratchpad and cache memory management for energy-efficient parallel HEVC encoding," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, New York, NY, USA, Oct. 2015, pp. 712–719.

[31] Y. Fan, L. Huang, B. Hao, and X. Zeng, "A hardware-oriented IME algorithm for HEVC and its hardware implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 2048–2057, Aug. 2018.



Hongjie Li was born in China in 1993. She is currently pursuing the master's degree with the School of Information Science and Technology, Shandong Normal University. Her main research interests include high performance computing.

YANHUI DING was born in China in 1981. He is currently an Associate Professor with the School of Information Science and Technology, Shandong Normal University. His research interests include Web data integration, medical image analysis, and data analysis.



HUI YU was born in China in 1983. She received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences. She is currently an Assistant Professor with the School of Management Science and Engineering, Shandong Normal University. Her research interests include signal processing and machine translation.

WEIZHI XU was born in China in 1982. He received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences. He was a Post-Doctoral Researcher with the Institute of Microelectronics, Tsinghua University. He is currently an Assistant Professor with the School of Information Science and Engineering, Shandong Normal University. His research interests include video processing and high performance computing.

FANGAI LIU received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently a Professor with the School of Information Science and Engineering, Shandong Normal University, China. His current research interests include parallel and distributed computing.