RESEARCH ARTICLE OPEN ACCESS

Implementation of Diamond Search Algorithm Using Parallel Processing Architecture

K. Shuma Roshini[1], M. Tejaswi[2],
[1]*M.Tech in Embedded Systems, Gudlavalleru Engineering College, Gudlavalleru, A.P.INDIA.*
[2]*Assistant Professor in ECE Department, Gudlavalleru Engineering College, Gudlavalleru, A.P.INDIA.*

Abstract
In video communication whole content of video cannot be stored without processing. So there is a need to compress the video before transmission and storage this process is called as video compression. Video compression plays an important role with regard to real-time scouting/video conferencing applications. Regarding the entire motion based video compression process, movement estimation is the most computationally expensive and time consuming process. Motion estimation is the key element in video compression. The Motion Estimation is a process which determines motion between two or more frames and finds best possible macro block. There are several algorithms on block matching to name a few, Full Search Motion estimation [FS], Three Step Search Motion Estimation [TSS], New Three Step Search Motion Estimation [NTSS], Four Step Search Motion Estimation [FSS], Diamond Search Motion Estimation [DS].Instead of trying to further reduce computational complexity of these algorithms it is better to implement these algorithms on parallel processing architecture. In this paper Diamond Search Algorithm is implementation on CPU and GPU.
Key Words: Motion Estimation, video compression, DS, GPU,CUDA.

I. INTRODUCTION

A raw video requires 2 to 3 GB of memory for one minute depending on resolution of the frame, it is impractical to store and transfer this much amount of video, for this purpose video compression is used. By using the technique of motion estimation video can be encoded, motion estimation is the process which determines motion between two or more frames and it is used to find best possible macro block. Video sequences consists of high level of redundancy between consecutive frame it means changes between frames are very less. In temporal redundancy the reduction of redundancy involves encoding of a first reference frame and the current frame, while the current frame encodes only the difference from the reference frame so this require a lot of computational complexity. In order to reduce computational complexity of ME [1] algorithms, a number of Block Matching Algorithms (BMA)[2] came into existences. At some point huge Computational complexity for all the algorithms is going to increase. So, Diamond Search algorithm is implemented on GPU to reduce compression time of a video.

A **Graphics Processing Unit** (**GPU**)[3] is a electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. The exhaustive motion search is implemented on GPU, instead of other fast ME algorithms, because of its regular memory access pattern. Although other fast ME algorithm can certainly be implemented, need an additional layer of texture to specify the target searching position and this results in random memory access pattern and dependent texture read. The repercussion of these are quite significant in modern graphics architecture.

Motion Estimation

Mainly in video editing motion estimation [4] is a type of video compression scheme. The motion estimation process is done by the coder to find the motion vector pointing to the best prediction macro block in a reference frame. For compression redundancy between adjacent frames can be exploited where a frame is selected as a reference and subsequent frames are predicted from the reference using motion estimation. The motion estimation process analyzes previous or future frames to identify blocks that have not changed, and motion vectors are stored in place of blocks.

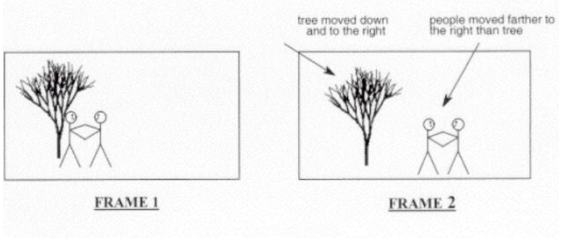


Figure 1: Comparing two Frames

Figure 1 shows an example of a frame with 2 stick figures and a tree. The second half of this figure is an example of a possible next frame, where panning has resulted in the tree moving down and to the right,

and the figures have moved farther to the right because of their own movement outside of the panning. The problem for motion estimation to solve is how to adequately represent the changes, or differences, between these two video frames.

It will probably worth mentioning that although motion estimation is also employed in many other disciplines an example would be computer vision, target tracking, and industrial monitoring, the techniques developed particularly for image coding a variety of in certain respects. The intention of image compression would be to lessen the total transmission bit rate for reconstructing images for the receiver. Hence, the motion information should occupy simply a small amount of the transmission bandwidth additional onto the picture contents information. So long as the motion parameters obtain can effectively reduce the total bit rate, these parameters need not function as true motion parameters. Besides, the reconstructed images for the receiving end are sometimes distorted. Therefore, in case the reconstructed images are employed for estimating motion information, an extremely strong noise component must not be ignored.

The motion estimation problem, involves two related sub-problems: i) identify the moving object boundaries, so-called motion segmentation, and ii) estimate the motion parameters of each one moving object, so-called motion estimation in strict sense. Within our use, a moving object is basically a team of contiguous pels that share the same variety of motion parameters. This definition does not necessarily equal the ordinary meaning of object. For instance, within the videophone scene, the still background may include wall, bookshelf, decorations, etc. Given that these merchandise are stationary (sharing the same zero motion vector), they could be considered as one single object in the case of motion estimation.

An appealing point in studying the motion estimation techniques for standards is the fact that the current video standards specify for only the decoders. Assuming motion displacement information (motion vectors) are generated through encoder and transmitted to the decoder, the decoder only performs the motion compensation operation—patch the to-be-reconstructed picture making use of the known (already decoded) picture(s). The encoder, which performs the motion estimation operation, is not really explicitly specified among the standards. Hence, it is more than possible use different motion estimation techniques to produce standards-compatible motion information. At the decoder, excluding the resources of coded pictures which can be used for motion compensation, fundamentally the same block-based motion compensation operation is designed by most of the popular video standards, H.261, H.263, MPEG-1, and MPEG-2[5][6].

II. LITERATURE SURVEY

As motion estimation happens to be the most compute intensive operate in an H.264 encoder, extensive researches most certainly been completed to accelerate the motion estimation operate in GPU. Since full search algorithm has more parallelism compared to any search algorithm, it's usually used to parallelize motion estimation in GPU [7]. But can often obtain huge performance gain in observation to the JM encoder, it is not clear if their result is yikes sufficient to be applied practically. In the other search algorithms, motion vector prediction (MVP) is made to increase the compression ratio. In MVP, the initial motion vector (MV) of the current macro block (MB) is predicted beginning with the MVs of one's neighbor macro blocks, assuming that they have been already encoded in a sequential raster order. And so the motion vector associated with a block cannot be predicted if MB encoding is performed in parallel. Since MVP limits parallelism, several approaches are proposed to extend the parallelism. If simply don't MVP, it can be observed that the coding efficiency is significantly degraded, resulting in a larger output size. Kung has proposed a block-based ME algorithm which uses 4x4 blocks as a substitute for 16x16 blocks. For being frame is separated into finer blocks, the total number of blocks that could be executed in parallel is increased. Since dynamic behavior like early termination introduces poor performance in GPU resulting from load imbalance, they proposed a fresh search algorithm based on three step search algorithm which has no early termination technique. Schwalb et al. have proposed a different approach. They ignored MVP to completely exploit macro block (MB) level parallelism. Instead, they use other predictors provided by Forward Dominant Vector Selection (FDVS) and Split and Merge techniques. In FDVS, it predicts the motion vector of n 1th frame by reusing the motion vectors of n previous frames. In Split and Merge, it exploits correlation of movement vectors between variable block sizes. Their result separated they will be able to achieve competitive quality and coding efficiency in comparison with UMHexagons Search algorithm even if they use Diamond Search algorithm [8]. These techniques can easily be applicable to our approach that if use multiple reference frames. Currently, only consider one reference frame. Search algorithm that proposed is the same as our algorithm in making use of a diamond pattern for search points. While their algorithm considers only a 3x3 square pattern, Here assume holistic diamond pattern (nxn) and of course the value of n that optimizes the performance within a target GPU is about through measurements. Within this particular approach, work with early termination techniques and prevent the load imbalance by carefully thinking about the underlying GPU architecture. To fully exploit MB level parallelism, here do not consider MVP that poses dependencies among the sequence of MBs encoding. To bypass the lack of MVP, Propose a new search algorithm fitted to the GPU architecture, where more computations are

performed yet in along-side maintain as much coding efficiency as MVP would supply.

III. PROPOSED SYSTEM

Compute Unified Device Architecture (CUDA) CUDA[9][10] serves as a parallel programming framework for utilizing GPU for general computing. Typical GPUs consist of hundreds of processing cores very effective at achieving immense parallel computing performance.Today all of NVIDIA's GPUs are CUDA GPUs. CUDA is not computer architecture in the sense of a definition of an instruction set and a set of architectural registers; binaries compiled for one CUDA GPU do not necessarily run on all CUDA GPUs. More specifically, NVIDIA defines different CUDA compute capabilities to describe the features supported by CUDA hardware. The first CUDA GPUs had compute capability 1.0. In 2011 NVIDIA released GPUs with compute capability 2.1, which is known as Fermi" architecture.

CUDA is based on the SIMD (Single Instruction Multiple Data) architecture and suited to exploiting various stages of data parallelism. A CUDA GPU consists of multiple so-called streaming multiprocessors (SMs).The threads executing a GPU program, a so-called kernel, are grouped in blocks. Threads belonging to one block all run on the same multiprocessor but one multiprocessor can run multiple blocks concurrently. Blocks are further divided into groups of 32 threads called warps; the threads belonging to one warp are executed in lock step, i.e., they are synchronized. CUDA is extension to C/C++ languages and allows programmers to write parallel programs for GPU.

Graphic Processing Unit is a computer chip that performs rapid mathematical calculations, primarily for rendering purpose. NVIDIA's GT GeForce 610 GPU has been chosen for implementing diamond search algorithm. This GPU consist of 48 processor cores where total work is allocated among all these processors. GPU is connected in a CPU using PCI slot of DDR3 memory type. It is also provided with boost clock where GPU can run at higher speed depending upon number of input factors that are used to determine whether it is a good idea to run at higher clock or not. Windows visual studio 2010 is used which is a Integrated development Environment (IDE) provides language services for all programming languages with code editor and debugger. For debugging purpose NVCC compiler is used which separate source codes and device code. NVCC generates both instructions for host and GPU, as well as instructions to send data back and forward between them. Device functions are processed by NVIDIA compiler and Host functions are processed by host compiler.

Specifications of GT GeForce 610

GPU Engine Specs:

- 48CUDA Cores
- 810Base Clock
- 1620Boost Clock
- 6.5Texture Fill Rate (billion/sec)

Memory Specs:

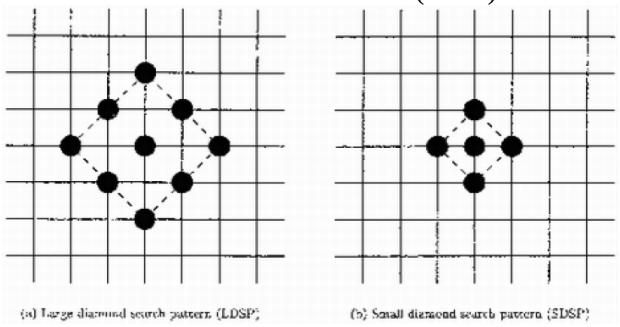
- 1.8 Gbps Memory Clock
- 1024MBStandard Memory Configuration
- DDR3Memory Interface
- 64-bitMemory Interface Width
- 14.4Memory Bandwidth (GB/sec)

Feature Support:

- 4.2OpenGL
- PCI Express 2.0Bus Support
- Certified for Windows 7
- DirectX 11, CUDA, PhysXSupported Technologies

Diamond Search Motion Estimation

The diamond search algorithm[8][11] has two search patterns first pattern consist of large diamond search pattern which consists of nine points from which eight points surrounds the center one to form a diamond shape which is called as Large Diamond Search Point (LDSP). Second pattern consist of five search points which forms smaller diamond and it is called as Small Diamond Search Point (SDSP).



(a) Large diamond search pattern (LDSP) (b) Small diamond search pattern (SDSP)

In searching procedure of the DS algorithm, first LDSP is searched repeatedly until minimum block distortion (MBD) occurs at the center point. If MBD occurs at the center point then search pattern is switched from LDSP to SDSP and search completes**.** Points yield from this search represents the motion vectors of best matching block.

Summary of DS algorithm:

Step 1: Initially LDSP is centered at the origin of the search window, and the 9 checking points of LDSP are tested. If MBD is located at the center then go to **step3**; otherwise **step2**.

Step 2: MBD point found in the previous search step is re-positioned as the center point to form a new LDSP. If the new MBD point obtained is located at the center position, go to **step3**; otherwise, recursively repeat this step.

Step3: Now switch the search pattern from LDSP to SDSP. The MBD point found in this step is the final solution of the motion vectors which points to the best matching block.

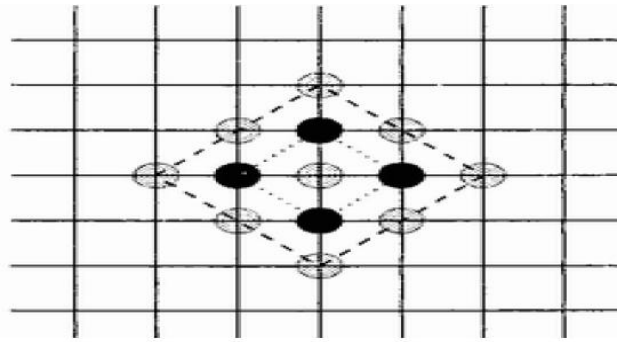


Figure 3: Light color dots represents the LDSP in this case lets us assume that center point contains least MBD then switch pattern shifts from LDSP to SDSP, thick color dots represents SDSP center point is final MBD which is best possible motion vector.

IV. RESULTS

GPU experimental results are compared with existing results which are taken from [12] in C environments with different configurations.

Table 1: Comparison of run times on CPU and GPU

Input video	Run time on CPU (seconds)	Run time on GPU (seconds)
Akiyo	17.33	3.67
Coast guard	17.26	3.86
Container	16.56	3.59
Foreman	17.28	3.88
News	16.77	3.75
Stefan	17.44	4.66

Simulation experimental video is given to GPU for implementing Diamond Search Algorithm for encoding purpose. In the above table run time of different videos are compared with run time of CPU. Encoding process of video occur faster in GPU. Table 2 shows by how much factor video can be compressed a video using this algorithm.

Table 2: Compression Factor of Videos

Video Name	Original File Size(MB)	Compressed file Size(KB)	Compression Factor(%)
Akiyo	10.8	427	38
Coast guard	10.8	467	42
Container	10.8	454	42
Foreman	10.8	471	42
News	10.8	471	42
Stefan	67.4	1035	51

V. CONCLUSION

In this paper Diamond Search Algorithm is implemented on parallel processing architecture for NVIDIA GPU. After observing all the communication overhead between host and device this proposed algorithm takes less run time compared to that of CPU with a 4x speed by only parallelizing motion estimation block in the encoding side.

VI. ACKNOWLEDGMENT

I am glad to express my deep sense of gratitude to Dr. M. Kamaraju, HOD of the ECE Department and entire staff of ECE Department, Gudlavalleru Engineering College, Gudlavalleru for their great support and encouragement in completion of this project.

REFERENCES

[1] Ahamdi, M. M. Azadfar "implementation of fast motion estimation algorithms & comparsion with full search method in H.264", IJCSNS international journal of computer science & Network security, vol 8, No.3, pp139-143, March 2008

[2] S. Immanuel Alex Pandian, Dr.G.josemin BalaBecky Alma George, / A Study on Block Matching Algorithms for Motion Estimation International Journal on Computer Science and Engineering (IJCSE) ISSN : 0975-3397, Vol. 3 No. 1 Jan 2011.

[3] John Nickolls, William j.Dally, " The GPU Computing Era," IEEE Annals of the History of Computing Publication(2010), pp.56-69, ISBN: 0272 -1732.

[4] R. Srinivasan and K. R. Rao, "predictive coding based on efficient motion estimation," IEEE Trans. Commun., vol. COMM-33,pp.888-896, Aug. 1985.

[5] K.R.Rao and j.j Hwang, techniques and standards for image, video and audio coding. Englewood Cliffs, NJ:prentice Hall,1996

[6] "MPEG-4 video verification model, ver. 14.0," ISO/IECJTC1/SC29/WG11/N29232, Oct. 1999.

[7] Zhou Jing, Jiao Liang bao, Cao Xuehong, " implementation of parallel full search algorithm for motion estimation on multi core Processors," (ICNIT) pp 31-35, ISBN 978-1-4577-0266-2, 2011

[8] D.v.manjunatha and Dr.sainarayanan "Comparsion and implementation of fast block matching motion estimation algorithms for video compression",ISSN:0975- 5462, Vol.3 No.10 October 2011 .

[9] W.Chen, H.Hang, "H.264/AVC motion estimation implementation on compute Unified Device Architecture(CUDA)", ICME 2008.

[10] Youngsub ko, youngmin yi and soonhoi ha, "an 9efficient parallel motion estimation algorithm and x264 parallelization in CUDA", ISBN: 978-1-4577-0620-2,page.no 91-98, IEEE,(2011).

[11] S.Zhu, K.ma, " A New Diamond search Algorithm for Fast Block-matching Motion Estimation", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL.9, NO.2, Feb.2000.

[12] D .rukmanidevi, p. rangarajan and j.raja paul, "A Novel search algorithm for variable block size motion estimation of H.264/AVC" ISSN: 1450-216X, vol 81 no 2(2012), pp. 160-167.