A Review on Parallel Medical Image Processing on GPU

CONFERENCE PAPER · AUGUST 2015

3 AUTHORS, INCLUDING:

Siau-Chuin Liew

Universiti Malaysia Pahang

24 PUBLICATIONS **30** CITATIONS

A Review on Parallel Medical Image Processing on GPU

Hui Liang Khor , Siau-Chuin Liew and Jasni Mohd. Zain
Faculty of Computer Systems and
Software Engineering
University Malaysia Pahang
Lebuhraya Tun Razak
26300 Gambang, Kuantan
Pahang Darul Makmur, Malaysia
hlkhorr@yahoo.com, liewsc@ump.edu.my, jasni@ump.edu.my

***Abstract*—An efficient implementation are necessary, as most medical imaging methods are computational expensive, and the amount of medical imaging data is growing .Graphic processing units (GPUs) can solve large data parallel problems at a higher speed than the traditional CPU, while being more affordable and energy efficient than distributed systems. This review investigates the use of GPUs to accelerate medical imaging methods. A set of criteria for efficient use of GPUs are defined. The review concludes that most medical image processing methods may benefit from GPU processing due to the methods' data parallel structure and high thread count. However, factors such as synchronization, branch divergence and memory usage can limit the speedup.**

Keywords—parallel computing; Medical image processing; GPU; multicores

I. INTRODUCTION

Technology advancement in parallel processing has demonstrated a potential in improving medical image processing performance. Parallel processing or computing, is a form of computation in which many calculations are carried out simultaneously. It is based on the concept that by dividing a large and complex problems into smaller and manageable parts, which are then solved concurrently [1]. Parallelism has been employed for many years, mainly in high- performance computing (i.e., supercomputers such as Cray, and PC-based cluster and grid computing), but the more recent rise of multicore processor technology such as Intel Quad-Core technology, which delivers four complete execution cores within a single processor, has brought the power of parallel processing down to the server and subsystem level.

Modern Graphic Processing Units (GPUs) used for general-purpose computations have a highly data parallel architecture. They are composed of a number of cores, each of which has a number of functional units, such as arithmetic logic units (ALUs). One or more of these functional units are used to process each thread of execution, and these groups of functional units are called thread processors. All thread processors in a core of a GPU perform the same instructions, as they share a control unit. This means that GPUs can perform the same instruction on each pixel of an image in parallel. Lee et al. (2010) has make a fair comparison between a CPU and GPU program as demonstrated in Table I. [2]

TABLE I. THE DIFFERENCE BETWEEN CPU AND GPU

GPU	**CPU**
Constructed to fit many thread processors on a chip	Designed with advanced control units and large caches
Have around 20 to 40 cores	Have around 4 to 12 cores
GPU cores share a control unit	CPU cores has a separate control unit
GPU cores perform the same instruction on each pixel of an image in parallel	CPU cores perform different instructions on different data in parallel

II. KEY FACTORS THAT DETERMINE THE SUITABILITY OF AN ALGORITHM TOWARDS A GPU IMPLEMENTATION

Several aspects define the suitability of an algorithm towards a GPU implementation. In this review, five key factors have been identified: Data parallelism, thread count, branch divergence, memory usage and synchronization. The following sections will discuss each of these factors, and explain why they are important for an efficient GPU implementation. Furthermore, several levels are defined for each factor (e.g. low, medium, high and none / dynamic), thereby creating a framework for rating to what extent an algorithm can benefit from GPU acceleration.

A. Data parallelism

TABLE II. TWO TYPE OF PARALLELISM

Task parallelism	**Data parallelism**
An algorithm execute the same instructions on multiple data elements in parallel	An algorithms execute different instructions in parallel
Suitable for GPUs	Suitable for multi-core CPUs

There are two type of parallelism as listed in table II. The serial part of an algorithm restricted the speedup in parallelization. Based on Amdahl's law (Amdahl, 1967), with a constant processing time in serial part, the maximum theoretical speedup of an algorithm (95% portion executed in parallel) is X 20 speedup despite of the number of cores possessed[3]. However, the practical speedup is often higher than the theoretical limit. This may due to: (1) the serial part of the algorithm is not fully optimized. (2) The parallel part of the algorithm may use the memory cache more efficiently. The degree of parallelism can be rated as illustrated in table III.

TABLE III. THE DEGREE OF PARALLELISM IN AN ALGORITHM

The degree of parallelism in an algorithm		
Low	Medium	High
0 – 49%	50 – 74%	75 – 100%

B. Thread count

A thread is an instance of a kernel. A high threads count in GPU is required for achieving a substantial speedup in an algorithm. This is because: 1) the clock speed of the CPU is higher than that of the GPU, and 2) global memory access may require several hundred clock cycles[4], potentially leaving the GPU idle while waiting for data. CPUs attempt to hide such latencies with large data caches. GPUs on the other hand, have a limited cache, and attempt to hide memory latency by scheduling another thread. Thus, a high number of threads are needed to ensure that some threads are ready while the other threads wait. Data parallelism as previously described, is the percentage of the algorithm that is data parallel. Thread count is how many individual parts the calculation can be divided into and executed in parallel. For most image processing algorithms, each pixel or voxel can be processed independently. This leads to a high thread count, and is a major reason why GPUs are well suited for image processing. For example, an image of size 512x512 would result in 262,144 threads, and a volume of size 256x256x256, almost 17 million threads. The rating of the thread count is defined as follows:

TABLE IV. THE RATING OF THE THREAD COUNT

Rating	Description
High	The thread count is equal to or more than the number of pixels/voxels in the image
Medium	The thread count is in the thousands
Low	The thread count is less than a thousand.
Dynamic	The thread count changes during the execution of the algorithm

C. Branch divergence

Threads are scheduled and executed atomically in groups on the GPU. AMD calls these groups wave fronts while NVIDIA calls them warps. However, in this review they will be referred to as an atomic unit of execution (AUE). An AUE is thus a group of threads that are all executed atomically on thread processors in the same core. The size of these groups may vary for different devices, but at the time of writing it is 32 for NVIDIA GPUs[5] and 64 for AMD GPUs [4]. Branches (e.g. if-else statements) are problematic because all thread processors that share a control unit have to perform the same instructions. To ensure correct results, the GPU will use masking techniques. If two or more threads in an AUE execute different execution paths, all execution paths have to be performed for all threads in that AUE. Such a branch is called a divergent branch. If the execution paths are short, this may not reduce performance by much.

The following levels are used for branch divergence:

TABLE V. THE RATING OF THE BRANCH DIVERGENCE

Rating	Description
High	More than 10% of the AUEs have branch divergence and the code complexity in the branch is substantial.
Medium	Less than 10% of the AUEs have branch divergence, but the code complexity is substantial.
Low	The code complexity in the branches is low.
None	No branch divergence

D. Memory usage

At the time of writing, GPUs with 2 to 4 GB memory are common while some high-end GPUs have 6 to 16 GB. Nevertheless, not all of this memory is accessible from a GPU program, as some of the memory may be reserved for system tasks (e.g. display) or used by other programs. This amount of memory may be insufficient for some segmentation methods that operate on large image datasets, such as dynamic 3D data. The system's main memory can be used as a backup, but this will degrade performance due to the high latency of the PCIe bus. For iterative methods, this limit can be devastating for performance as data exceeding the limit would have to be streamed back and forth for each iteration. Defining N as the total number of pixels/voxels in the image the rating of memory usage is:

TABLE VI. THE RATING OF THE MEMORY USAGE

Rating	Description
High	More than 5N.
Medium	From 2N to 5N.
Low	2N or less.

E. Synchronization

Most parallel algorithms require some form of synchronization between the threads. One way to perform synchronization is by atomic operations. An operation is atomic if it appears to happen instantaneously for the other threads. This means the other threads have to wait for the atomic operation to finish. Thus, if each thread performs an atomic operation, the operations will be executed serially and not in parallel. Global synchronization is synchronization between all threads. This is not possible to do inside the kernels on the GPU except using atomic operations. Thus global synchronization is generally done by executing multiple kernels which can be expensive. This is due to the need for global memory read and write, double buffering and the overhead of kernel launches. Local synchronization is to perform synchronization between threads in a group. This can be done by using shared memory, atomic operations or the new shuffle instruction [6]. The rating of synchronization is defined in this review as follows:

TABLE VII. THE RATING OF THE SYNCHRONIZATION

Rating	Description
High	Global synchronization is performed more than hundred times. This is usually true for iterative methods.
Medium	Global synchronization is performed between 10 and 100 times.
Low	Only a few global or local synchronizations.
None	No synchronization.

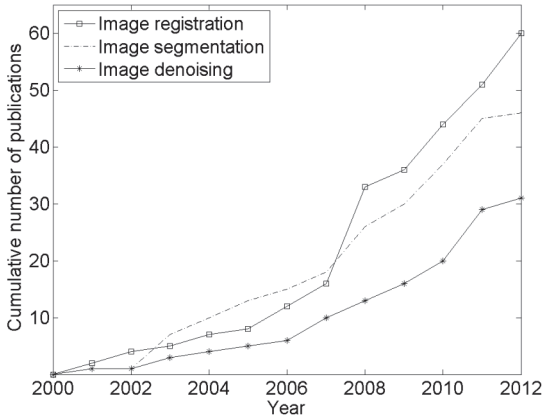
III. GPU COMPUTING ON MEDICAL IMAGES

Graphic processing units (GPUs) were originally created for rendering graphics. However, in the last ten years, GPUs have become popular for general-purpose high performance computation, including medical image

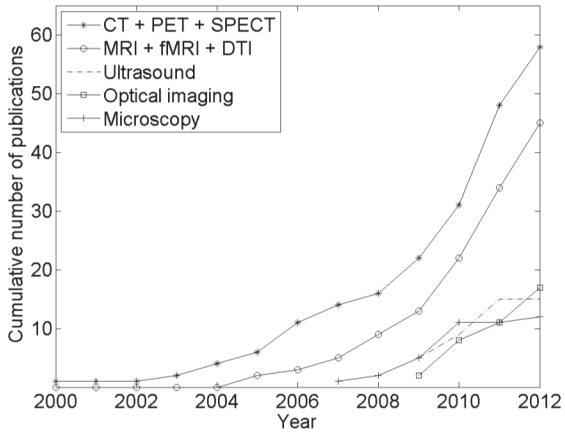
processing. This is most likely due to the increased programmability of these devices, combined with low cost and high performance. Shi et al. (2012) recently presented a survey on GPU-based medical image computing techniques such as segmentation, registration and visualization [7]. Pratx and Xing (2011) provided a review on GPU computing in medical physics with focus on the applications image reconstruction, dose calculation and treatment plan optimization, and image processing [8]. A more extensive survey on medical image processing on GPUs was presented by Eklund et al. (2013) [9]. They investigated GPU computing in several medical image processing areas such as image registration, segmentation, denoising, filtering, interpolation and reconstruction. So et al. (2011) made a comparison between CPUs and GPUs for ultrasound systems, in terms of power efficiency and cost effectiveness. The conclusion was that a hybrid CPU-GPU system performed best [10]. The table VIII demonstrated the potential of parallel computation in medical imaging and visualization in a wide range of applications including image reconstruction, image denoising, motion estimation, deformable registration, diffeomorphic mapping, and modelling.

TABLE VIII. PARALLEL COMPUTATION IN MEDICAL IMAGING

Researcher	**Medical Imaging Process on GPU**	**Speed up in relative to CPU**
E. van Aart et al [11]	accelerated algorithm for brain fiber tracking	40 X
L. Ha et al. [12]	three-dimensional deformable registration algorithm for mapping brain datasets	3 X
F. Lecron et al. [13]	Heterogeneous computing for vertebra detection and segmentation in X-ray images	3 to 22 X
M. Xu and P. [14]	image reconstruction on CT machines	5 X
M. Schweiger [15]	GPU-accelerated finite element method for modelling light transport in diffuse optical tomography	10 to 20 X
D. Kim et al [16]	MRI reconstruction	5 X
Eklund et al [17]	True 4D image denoising on the GPU	6 X
D. J. Tward et al [18]	simulation and phantom modeling	20 X
L. D'Amore et al [19]	Numerical solution of diffusion models in biomedical imaging on multicore processors	10 X
J. Thiyagalingam et al [20]	On the usage of GPUs for efficient motion estimation in medical image sequences	60 X
A.Eklund et al [21]	Fast random permutation tests enable objective evaluation of methods for single subject fMRI analysis	10 X



(a) Algorithms



(b) Modalities

Fig. 1. Cumulative number of publications versus (a) GPU acceleration of algorithms and (b) modalities.

Figure 1 illustrates the popularity of GPUs application in medical imaging field. The Figure 1 (a) illustrate the number of publications used in variety medical imaging algorithms, while the Figure 1 (b) presents the number of publications for a specific medical image modality. It shows that GPU application in medical imaging has increased in 2008, after the release of compute unified device architecture (CUDA) in 2007. The CT was the earliest modality that utilized graphics hardware as compare to the MRI modality. This may due to CT data normally requires more demanding algorithms (e.g. an inverse Radon transform) as compared to MRI reconstruction algorithms (e.g. inverse fast Fourier transform). GPUs have also applied on ultrasound modalities in real-time processing and visualization. Optical imaging and microscopy have utilized GPUs, in speeding up reconstruction algorithms.

IV. CONCLUSION

A high demand of powerful in computer games has increased the demand of high performance's GPUs as compared to CPU demands. It is because GPU can differ by a factor ten in theoretical performance. Most Image processing algorithms exhibit a parallel behavior, which GPU is an ideal framework for them. In certain cases a hybrid CPU-GPU implementation may give a better performance. For example, image registration algorithms, where the GPU calculate the similarity in parallel while CPU runs a serial optimization algorithm. Generally, for an algorithm to perform efficiently on a GPU it has to be data parallel, have many threads, no divergent branches, use less memory than the total amount of memory on the GPU and use as little synchronization as possible. However, there are several other factors affecting GPU performance, such as kernel complexity, ALU to fetch ratio, bank conflicts etc. Most of medical image processing methods are data parallel with a high amount of threads, which makes them well suited for GPU acceleration. However, factors such as synchronization, branch divergence and memory usage can limit the speedup over serial execution. GPU optimization techniques are required in order to reduce the impact of these limiting factors.

General purpose GPU frameworks such as Open Computing Language (OpenCL) and compute unified device architecture (CUDA) have attracted a lot of users in recent years. Their popularity is likely to increase, as they ease the programming of GPUs compared to shader programming. OpenCL enables efficient use of both GPUs and CPUs. It is likely that more hybrid solutions that use GPUs for the massively data parallel parts, and the CPU for the less parallel parts will appear. The challenge with these hybrid solutions is efficient sharing of data. At the time of writing, sharing data has to be done explicitly by memory transfer over the PCI express bus. However, this seems to be an issue that both major GPU manufacturers want to improve. It is also likely that there will be an increase in GPU libraries with commonly used data structures and algorithms such as heaps, sort, stream compaction and reduction. Libraries and frameworks that aid in writing image processing algorithms as well as scheduling, memory management and streaming of dynamic image data will probably become more important as more algorithms and image data are processed on the GPU. One framework that aims to aid the design of image processing algorithms for different GPUs is the Heterogeneous Image Processing Acceleration Framework (HIPAcc). There are two main GPU manufacturers, NVIDIA and AMD, provide some details of the future development of their GPUs. However, these details are subject to change. In general, the trend in GPU development has been increasing the number of thread processors, the clock speed and the amount of on-board memory. This allows more data to be processed faster in parallel.

REFERENCES

[1] Jian-Ming Wang, William F.Eddy, "Parallel Algorithm for SWFFT Using 3D Data Strcture" Circuits, System and Signal Processing, April 2012, Volume 31, Issue 2, pp 711-726.

[2] Lee, V., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A., Satish, N., Smelyanskiy, M., Chennupaty, S., Hammarlund, P., Singhal, R., Dubey, P., 2010. Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU. In: Proceedings of the 37th Annual International Symposium on Computer Architecture, pp. 451‑460.

[3] Amdahl, G.M., 1967. Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of AFIPS ' 67 (Spring). ACM Press, New York, New York, USA, pp. 483‑485, http://dx.doi.org/10.1145/ 1465482.146556

[4] Advanced icro Devices, 2012. AMD Accelerated Parallel Processing OpenCL Programming Guide. Technical Report July

[5] NVIDIA, 2010. OpenCL best practices guide. Technical Report.

[6] NVIDIA, 2013a. CUDA C programming guide. Technical Report July.

[7] Shi et al., "Survey of GPU-based medical image", computing Quantitative Imaging in Medicine and Surgery, Vol 2, No 3 September 2012

[8] Pratx, G., Xing, L., 2011. GPU computing in medical physics: A review. Medical physics 38, 2685–2697.

[9] Eklund, A., Bj¨ornsdotter, M., Stelzer, J., LaConte, S., 2013. Searchlight goes GPU - Fast multi-voxel pattern analysis of fMRI data, in: International Society for Magnetic Resonance in Medicine (ISMRM).

[10] So, H.K.H., Chen, J., Yiu, B.Y., Yu, A.C., 2011. Medical ultrasound imaging:

[11] To GPU or not to GPU? IEEE Micro 31, 54–65.

[12] Evert van Aart, Neda Sepasian, Andrei Jalba, and Anna Vilanova, "CUDA-Accelerated Geodesic Ray-Tracing for Fiber Tracking," International Journal of Biomedical Imaging, vol. 2011, Article ID 698908, 12 pages, 2011. doi:10.1155/2011/698908

[13] Fabian Lecron, Sidi Ahmed Mahmoudi, Mohammed Benjelloun, Saïd Mahmoudi, and Pierre Manneback, "Heterogeneous Computing for Vertebra Detection and Segmentation in X-Ray Images," International Journal of Biomedical Imaging, vol. 2011, Article ID 640208, 12 pages, 2011. doi:10.1155/2011/640208

[14] Meilian Xu and Parimala Thulasiraman, "Mapping Iterative Medical Imaging Algorithm on Cell Accelerator,"International Journal of Biomedical Imaging, vol. 2011, Article ID 843924, 11 pages, 2011. doi:10.1155/2011/843924

[15] Martin Schweiger, "GPU-Accelerated Finite Element Method for Modelling Light Transport in Diffuse Optical Tomography," International Journal of Biomedical Imaging, vol. 2011, Article ID 403892, 11 pages, 2011. doi:10.1155/2011/403892

[16] Daehyun Kim, Joshua Trzasko, Mikhail Smelyanskiy, Clifton Haider, Pradeep Dubey, and Armando Manduca, "High-Performance 3D Compressive Sensing MRI Reconstruction Using Many-Core Architectures,"International Journal of Biomedical Imaging, vol. 2011, Article ID 473128, 11 pages, 2011. doi:10.1155/2011/473128

[17] Anders Eklund, Mats Andersson, and Hans Knutsson, "True 4D Image Denoising on the GPU," International Journal of Biomedical Imaging, vol. 2011, Article ID 952819, 16 pages, 2011. doi:10.1155/2011/952819

[18] Daniel J. Tward, Can Ceritoglu, Anthony Kolasny, et al., "Patient Specific Dosimetry Phantoms Using Multichannel LDDMM of the Whole Body," International Journal of Biomedical Imaging, vol. 2011, Article ID 481064, 9 pages, 2011. doi:10.1155/2011/481064

[19] Luisa D'Amore, Daniela Casaburi, Livia Marcellino, and Almerico Murli, "Numerical Solution of Diffusion Models in Biomedical Imaging on Multicore Processors," International Journal of Biomedical Imaging, vol. 2011, Article ID 680765, 16 pages, 2011. doi:10.1155/2011/680765

[20] Jeyarajan Thiyagalingam, Daniel Goodman, Julia A. Schnabel, Anne Trefethen, and Vicente Grau, "On the Usage of GPUs for Efficient Motion Estimation in Medical Image Sequences," International Journal of Biomedical Imaging, vol. 2011, Article ID 137604, 15 pages, 2011. doi:10.1155/2011/137604

[21] Anders Eklund, Mats Andersson, and Hans Knutsson, "Fast Random Permutation Tests Enable Objective Evaluation of Methods for Single-Subject fMRI Analysis," International Journal of Biomedical Imaging, vol. 2011, Article ID 627947, 15 pages, 2011. doi:10.1155/2011/627947

[22] Donoho, D., 2006. Compressed sensing. IEEE Transactions on information theory 52, 1289–1306.