GPU Accelerated Ultrasonic Tomography Using Propagation and Backpropagation Method

Pedro D. Bello[1], Yuanwei Jin[2], and Enyue Lu[3]
[1]Department of Electrical and Computer Engineering, Florida International University, FL
[2]Department of Engineering and Aviation Sciences, University of Maryland Eastern Shore, MD
[3]Department of Mathematics and Computer Science, Salisbury University, MD

***Abstract*—This paper develops implementation strategy and method to accelerate the propagation and backpropagation (PBP) tomographic imaging algorithm using Graphic Processing Units (GPUs). The Compute Unified Device Architecture (CUDA) programming model is used to develop our parallelized algorithm since the CUDA model allows the user to interact with the GPU resources more efficiently than traditional Shader methods. The results show an improvement of more than 80x when compared to the C/C++ version of the algorithm, and 515x when compared to the MATLAB version while achieving high quality imaging for both cases. We test different CUDA kernel configurations in order to measure changes in the processing-time of our algorithm. By examining the acceleration rate and the image quality, we develop an optimal kernel configuration that maximizes the throughput of CUDA implementation for the PBP method.**

***Index Terms*—Medical Imaging, Ultrasonic Tomography, GPU, CUDA, Parallel Computing**

I. Introduction

GRAPHIC Processing Units (GPUs) are computation-dedicated hardware that can significantly accelerate execution of algorithms for different applications. In 2007, NVIDIA released their Computed Unified Device Architecture (CUDA) programming model, which allows C/C++ users to use the GPU resources in a simpler and friendly manner. This new programming model is being used, nowadays, for a wide range of applications such as medical imaging, online gaming, etc. due to its flexibility and high performance.

1.1 Compute Unified Device Architecture CUDA

The CUDA programming model enables a channel that connects the CPU with the GPU. Functions that are needed to run in parallel are launched from the CPU and executed on the GPU. If data is stored on the CPU, but needs to be processed on the GPU, memory copies and allocation must be programmed. After the data is stored on the GPU, kernel functions are called to process the data in parallel.

The kernel functions assign threads to process the same instructions in different parts of the data. Algorithms that follow this pattern are called Single Instruction Multiple Data (SIMD) algorithms. To follow the SIMD implementation, the CUDA programming model needs to establish a kernel configuration for a specific application. Kernel configuration refers to how threads are grouped into blocks and how blocks are organized to cover a whole grid of data at each execution run in a parallel mode. The strategy to determine the size of blocks (i.e. dimensionality) and the format of block organization impacts greatly the computation time of a CUDA application. Our goal is to find the kernel configuration that achieves the fastest implementation for our imaging application.

1.2 Previous Work on Medical Imaging using GPUs

Much research in the medical imaging field using GPUs has been done for many years. In [1], Copeland et al. used 2D images, obtained by X-ray projections, to reconstruct 3D+T cerebral angiography information. The total processing time of their algorithm was about six hours. Later on in [2], his team developed a new version of the algorithm that processed the data in parallel using GPUs. According to their results, improvements of 12x were achieved for a "naive" approach using CUDA. Then, using the sparsity of the problem, and by adapting the memory accessing of the information they were able to achieve 246x improvement. Wen-mei et al. in [3], implemented a magnetic resonance image reconstruction algorithm that took advantage of the GPU resources. They started by implementing a "naive" algorithm that was slow, and then they exploited the memory-accessing optimizations that the CUDA programing model offers. At the end, their results showed improvements of 22.5x.

In [4], we developed a new iterative image reconstruction algorithm called PBP method. This method utilizes the multiple-input multiple-output (MIMO) technique to achieve faster imaging reconstruction. In this paper, we develop a GPU implementation that exploits the parallelism structure of the algorithm to further accelerate image reconstruction.

II. GPU Accelerated PBP Algorithm

2.1 PBP Imaging Algorithm

The process of reconstructing images from ultrasonic information starts with the following acoustical wave equation:

$$\frac{\partial^2}{\partial t^2} u(\mathbf{x},t) = c^2(\mathbf{x}) \triangle u(\mathbf{x},t) + \sum_{l=1}^{J_m} s(\mathbf{x},\mathbf{s}_l,t) \qquad (1)$$

where the acoustic field $u(\mathbf{x},t) \in \Omega \times [0,T]$. J_m is the number of simultaneous excitation sources. $c(\mathbf{x}) = c_0\sqrt{1+f(\mathbf{x})}$ is

The work is supported in part by the National Science Foundation under grant no. CCF-1156509 and CMMI-1126008, and the U.S. Army Research Laboratory, the Office of Naval Research, and the Army Research Office under grant no. W911NF-11-1-0160.

P. Bello did his work as a REU (Research Experience for Undergraduate) student at Salisbury Univeristy in Summer 2012.

the propagation speed of the acoustic wave in the medium. $f(\mathbf{x})$ is the acoustic potential function that needs to be reconstructed. The imaging problem described by (1) can be formulated as an inverse problem:

$$R_j\left(f\left(\mathbf{x}\right)\right) = g_j \tag{2}$$

where g_j is the measurement data collected at the acoustic sensors, and $R_j(\cdot)$ is the non-linear operator governed by (1). The solution of (2) can be solved using the PBP method [4], which takes the form of

$$f^{k+1}\left(\mathbf{x}\right) = f^k\left(\mathbf{x}\right) + R_j'\left(f^k\left(\mathbf{x}\right)\right)^*\left[g_j - R_j\left(f^k\left(\mathbf{x}\right)\right)\right] \tag{3}$$

where $k = 0, 1, 2, \cdots$ is the iteration timestep.

2.2 GPU Acceleration

We recognize that the iteration equation given by (3) can be executed in parallel at each spatial grid point of the imaging field. However, it cannot be programmed to calculate the $f(\mathbf{x})$ value at different time stages in parallel. Therefore, the final CUDA implementation algorithm becomes a combination of parallel and sequential tasks.

For the CUDA implementation of the tomographic imaging algorithm, image values at each grid point (i.e. pixels) are calculated in parallel. Each grid point is processed by a thread. Threads are then organized into blocks by CUDA. Although, it is believed that different kernel configurations impact the acceleration performance [5], to the best of our knowledge, there is very limited research regarding how block dimensionality affects the performance of CUDA applications, particularly for imaging algorithms. In this work, we conduct experimental tests on GPU of the PBP imaging algorithm and identify its optimal kernel configuration.

Our tests aim to determine which block configuration works better for our application by testing all the possible options. We modify the kernels of our program, making sure that the quality of the reconstructed image remains comparable. We measure the execution time of each test. Finally, we examine all the configuration results and analyze how timings change based on these configurations.

2.3 Imaging Results Implemented by GPU

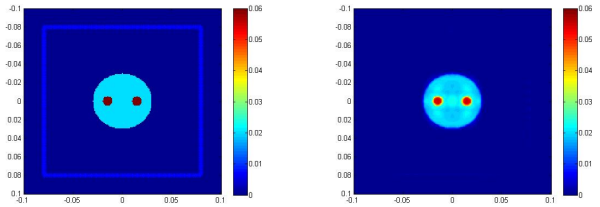
The GPU used for our testing is a GeForce GTX 580 which has 512 CUDA cores and 1.6 GB RAM DDR5. The CPU is a quad-core, desktop computer with a Xeon E5607 microprocessor running at 2.6 GHz and 8.0 GB RAM.

We benchmark our results using three different platforms. The original PBP algorithm was implemented in MATLAB [4]. Since the CUDA programming model is based on Microsoft Visual Studio C/C++, we propose to use both C/C++ and CUDA to test the performance improvement. Using these three programing models we test different behaviors that lead to different processing times with comparable image quality. Exact timings can be found in Table I.

Next, we implement our experimental tests in order to measure computation timings for different kernel configurations. We choose to test all the possible options so that our experimental results would be conclusive for the GPU that we use. According to our results, blocks of size 16 by 8 maximize the throughput of our application. Furthermore, kernels of 128 threads per block are faster than any other size for most of the cases. This can be accounted to the fact that, although it is possible to saturate the GPU resources using 256 or 512 threads per block, it is more efficient to process more blocks by each Streaming Multiprocessor (SM). The ground truth image and the reconstructed image implemented in GPU are shown in Fig. 1. Fig. 1a depicts a circular object to be imaged. The object is surrounded by a total of 640 acoustic sensors. Each sensor can transmit and receive acoustic wave signals. From the measured sensor data, the object image (see Fig. 1b) is reconstructed by the iteration equation given in (3).

MATLAB	C/C++	CUDA C/C++
06:06:27	00:56:55	00:00:43

Table I
TIME MEASUREMENTS (HOURS:MINUTES:SECONDS)



(a) Ground truth image (b) Reconstructed image

Figure 1. Reconstructed image by PBP method using GPU

III. CONCLUSIONS

We developed a fast algorithm that utilizes GPU resources to solve the imaging problem by MIMO-PBP ultrasonic tomography. The initial algorithm took 6 hours to complete, and our GPU implementation reduced the processing time to 43 seconds, thus achieving improvements of 515x faster than the initial MATLAB implementation, and 80x faster than the C/C++ implementation. Furthermore, we proposed a kernel configuration that maximizes the throughput of medical imaging algorithms using CUDA. This configuration, and the procedure we developed that leads to it, can be extended to other CUDA applications. By accelerating the algorithm using GPUs, tests that were not possible to perform before due to the long execution times are now possible.

REFERENCES

[1] A. Copelan, R. Mangoubi, M. Desai, S. Miter, and A. Malek, "Spatio-Temporal Data Fusion for 3D+T Image Reconstruction in Cerebral Angiography," *IEEE Transactions on Medical Imaging*, vol. 29, pp. 1238 – 1251, June 2012.

[2] Y. Xu, R. Mangoubi, M. Desai, A. Copelan, S. Mitter, and A. Malek, "GPU-based Real-time Implementation of 3D+T Image Reconstruction with Application to Cerebral Angiography," in *IEEE International Symposium on Biomedical Imaging*, 2011, pp. 401 – 406.

[3] W. Hwu, D. Nandakumar, J. Haldar, and I. Atkinson, "Accelarating MR Image Reconstruction on GPUs," in *IEEE International Symposium on Biomedical Imaging*, 2009, pp. 1283 – 1286.

[4] C. Dong, Y. Jin, M. Farrar, and K. Priddy, "A study of multi-static ultrasonic tomography using propagation and backpropagation method," in *Proccedings of SPIE*, vol. 8051, 2011, pp. 805 106–805 106–13.

[5] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-On Approach*. Morgan Kaufmann, 2010.