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

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.

Problem Description

Ultrasonic Tomography is widely used for medical imaging applications due to its great features. However, typical tomographic imaging algorithms are very slow due to the amount of computations that are required. Software efficient techniques have been proposed in order to improve the performance. In this poster, we propose a hardware technique that utilizes GPU resources to speed up the reconstruction process.

Reconstruction Model

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

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

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

$$R_j(f) = g_j \qquad (2)$$

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

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

where $k = 0, 1, 2, \cdots$ is the iteration time-step.

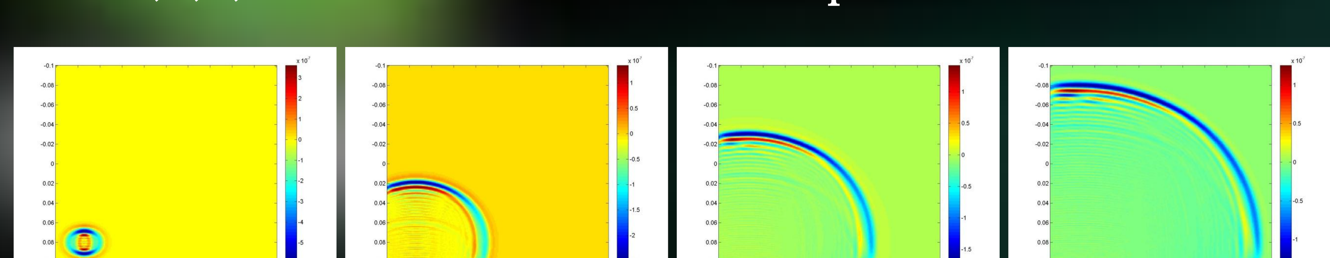


Figure 1. Acoustic wave propagation simulation

GPU Acceleration

For CUDA implementation of the tomographic imaging algorithm, image values at each grid point (i.e. pixels) are calculated in parallel (Fig. 2). 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, 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.

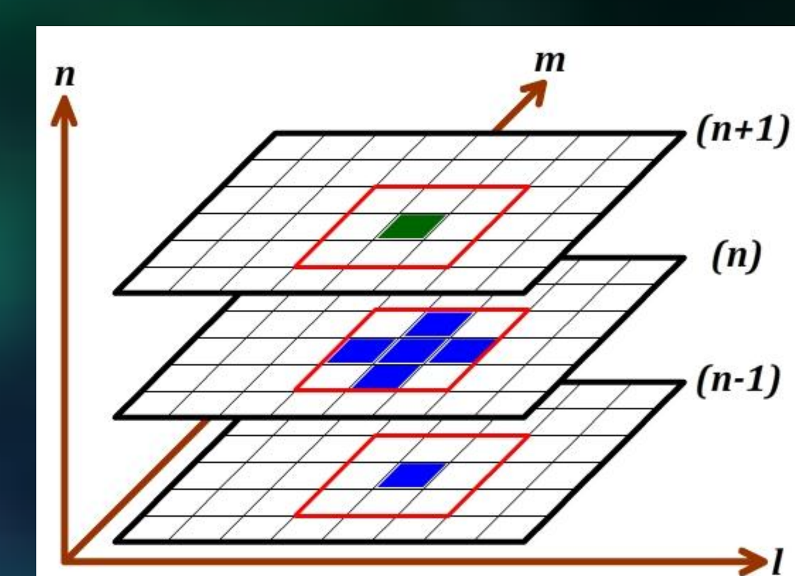
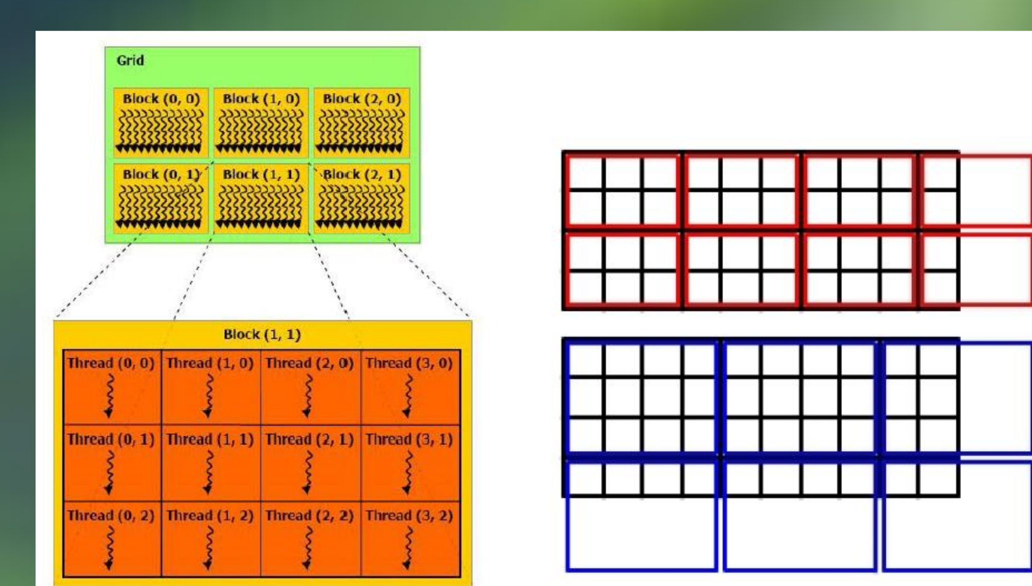


Figure 2. Image calculation process

Experimental Setup

Our tests consist of finding which kernel configuration works better for our application by testing all the possible options. We modify the kernels of our program (Fig. 3), 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 (Table 1).



(a) Grid of thread blocks (b) Two different blocks mapping

Figure 3. Kernel configurations

	1	2	4	8	16	32	64	128	256
1	1493.066	751.645	398.929	211.295	119.99	71.881	53.506	48.231	51.605
2	768.29	400.421	213.674	118.295	69.124	47.533	47.439	52.323	53.588
4	418.599	222.911	122.103	68.656	46.762	42.809	50.591	53.337	53.823
8	244.799	135.003	76.222	47.567	42.692	44.366	51.278	53.993	-
16	162.069	92.762	57.24	46.695	43.604	44.929	51.263	-	-
32	125.833	76.598	58.64	49.625	45.318	47.799	-	-	-
64	114.774	79.203	62.166	53.15	52.883	-	-	-	-
128	117.722	83.897	62.948	54.585	-	-	-	-	-
256	143.896	97.797	76.331	-	-	-	-	-	-

Table 1. Kernel timings for different block sizes

Results

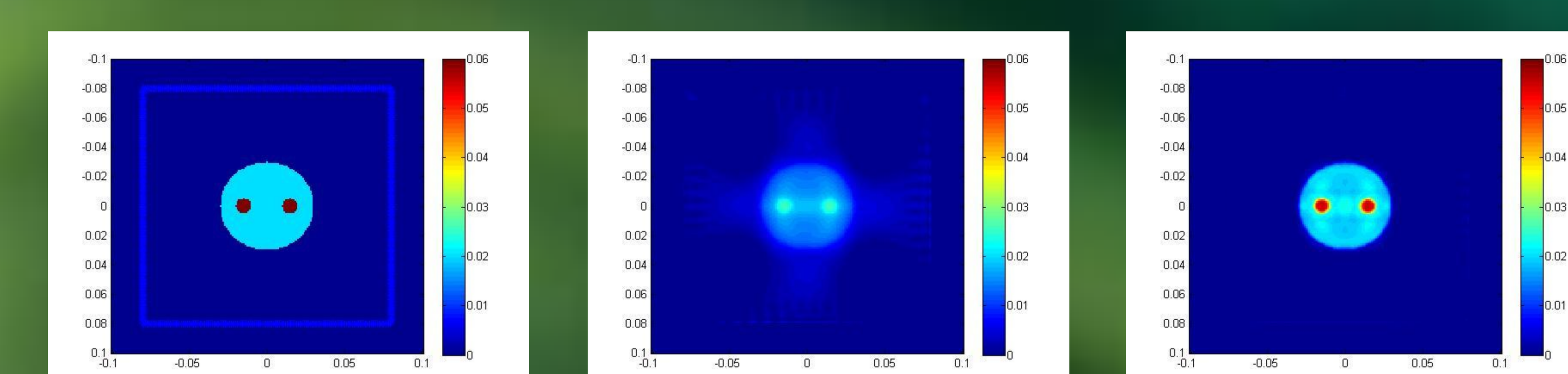
We benchmark our results using three different platforms. The original PBP algorithm was implemented in MATLAB. Since the CUDA programming model is based on C/C++, we propose to use both C/C++ and CUDA to test the performance improvement. Using these three programing models we test different imaging scenarios that lead to different processing times with comparable image quality. Execution times and performance improvement can be seen in Table 2 and 3, respectively. Final imaging results can be seen in Fig. 4.

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

Table 2. Processing times

	MATLAB	C/C++	CUDA C/C++
MATLAB	1	0.156	0.00194
C/C++	6.438	1	0.0125
CUDA C/C++	515.328	80.059	1

Table 3. Performance tabulation



(a) Ground truth image (b) Reconstruction after first iteration (c) Final reconstructed image

Figure 4. Reconstructed image by the PBP method using GPU

Conclusion

We developed a faster hardware implementation algorithm that utilizes GPU resources to accelerate the imaging problem by propagation and backpropagation ultrasonic tomography. The initial algorithm took 6 hours to finish, and we managed to reduce 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 the PBP imaging algorithms using CUDA.

Acknowledgments

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.