RISE:2012
RESEARCH, INNOVATION AND SCHOLARSHIP EXPO

Graduate
Category: Engineering and Technology
Degree Level: PhD
Abstract ID# 216

Northeastern University

CDSP

Accelerating 3D CT Reconstruction Using GPUs

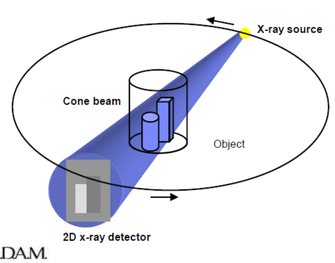
Saoni Mukherjee†, Miriam Leeser†

† Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA

Abstract

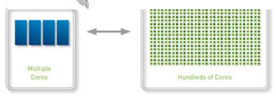
Biomedical applications with large datasets can benefit from acceleration. Graphic Processing Units(GPUs) are particularly useful in this context as they can produce high fidelity images in faster time. An image algorithm to reconstruct conebeam computed tomography(CT) using 2D projections is implemented using GPUs. The implementation takes slices of the target, weighs the projection data and then filters the weighted data to backproject the data and creates the final 3D construction. The implementation is tested using mathematical phantoms to evaluate its performance.

Background – State of the art

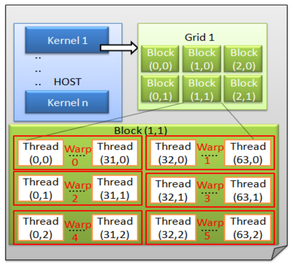


ADAM.

GPU and CUDA



- Sequential part runs on CPU and computationally intensive part is accelerated by GPU
- Massively multithreaded multicore chips
- Users across Science and Engineering achieving 100x or better speedup on GPUs
- Theoretical peak performance: 518 GFLOPS
- Enabling heterogeneous computing
- Minimal extension to C/C++ environment
- Serial **host** code runs in one host thread and parallel **kernel** code runs in several device threads across multiple GPU threads, all written in C.



Goal

The serial application has already been implemented and the implementation does not provide real time images. The goal is to make the reconstruction as close as real-time.

Feldkamp Algorithm

The Algorithm used to implement this application is Feldkamp Algorithm.

Feldkamp reconstruction of cone-beam data can be formulated as a weighted filtered backprojection. First, the cone-beam projections are individually weighted and ramp filtered, producing the modified cone-beam projections $q(p,\vec{t})$, corresponding to source positions θ_p. Next, the 3D image is reconstructed by a weighted backprojection.

To define the backprojection operation, let $\vec{r} = [x, y, z]^T$ denote position in the 3D image (i.e., object), and let $\vec{\tau}(\theta, \vec{r}) = [\tau_1(\theta, \vec{r}), \tau_2(\theta, \vec{r})]^T$ denote the t_1, t_2 position of the intersection with the detector plane of the source ray passing through point $\vec{r}$ and with source angle θ. Then

$$\tau_1(\theta, \vec{r}) = \frac{D(y\cos\theta - x\sin\theta)}{D + x\cos\theta + y\sin\theta}$$

$$\tau_2(\theta, \vec{r}) = \frac{Dz}{D + x\cos\theta + y\sin\theta}.$$

The weighted backprojection is then

$$f(\vec{r}) = \sum_{p=0}^{P-1} W(p\Delta\theta, \vec{r})\, g[p, \vec{\tau}(p\Delta\theta, \vec{r})]\Delta\theta,$$

where $W(\theta, \vec{r})$ is an appropriate weight function. The backprojection formula is identical to the 2-D fan-beam case, except that in the latter τ is scalar, and $\vec{r}$ two dimensional. The computational cost of 3D cone-beam backprojection for an $N \times N \times N$ image with P projections is cN^3P, because the contributions of P projections have to be accumulated in for each of the N^3 image voxels. The constant c depends on implementation details such as complexity of the interpolation. In contrast, the computational cost of weighting and ramp filtering is only $O(PN^2 \log N)$ when the convolution is performed using FFTs. Therefore, the cost of backprojection dominates the cost of conventional cone-beam reconstruction, which has cost $O(PN^3)$, or $O(N^4)$, when, as is often the case, $P = O(N)$. The situation is similar in 2D fan-beam reconstruction, where the complexities of the filtering and backprojection steps are $O(N^2 \log N)$ and $O(N^3)$, respectively.

* The above algorithm is an excerpt from [2]

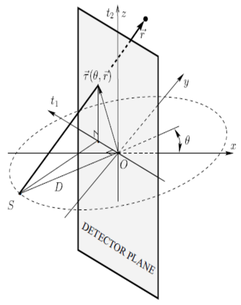


Fig. Cone-beam tomography with circular source orbit and planar equispaced detector.

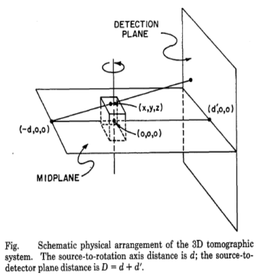


Fig. Schematic physical arrangement of the 3D tomographic system. The source-to-rotation axis distance is d; the source-to-detector plane distance is $D = d + d'$.

Serial vs. Parallel

Serial implementation goes through each slice one after another, whereas the GPU kernel code contains the operation inside for loop. That kernel is called from the host code. The rest is handled by GPU. GPU executes the same operations multiple times and sends back the result to host.

Goal Achieved?

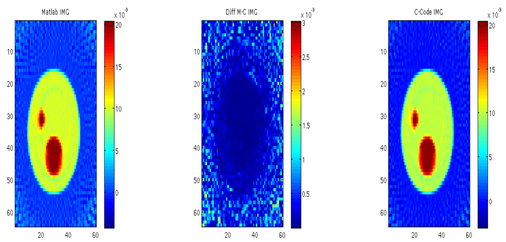
The serial version takes 3.04 seconds to run (on average)
The GPU version takes 0.2 seconds to run (on average)

Speed-up = 15 times

Future Works

- Apply to real world data.
- Improve speed up.
- Accelerate other biomedical imaging applications.

Results



References

1. L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," *J. Opt. Soc. Am.*, vol. 1 (A), pp. 612-619, 1984.
2. S Xiao, Y Bresler, Munson Jr. DC, "Fast feldkamp algorithm for cone-beam computer tomography". *Proceedings IEEE International Conference on Image Processing (ICIP)* 2003; 819–22.
3. NVIDIA CUDA C programming guide, CUDA 4.1 documentation.

Acknowledgements

James Brock, Nicholas Moore.

This work was supported in part by the National Science Foundation Engineering Research Centers Innovations Program, Biomedical Imaging Acceleration Testbench (Award Number EEC-0946463).

NSF