

Developing a Massively Parallel Forward Projection Radiography Model for Large-Scale Industrial Applications

Matthew Bauerle, University of Michigan Ann Arbor
 Advisor: Edward Jimenez
 Sandia National Laboratories

August 22, 2014

Abstract

This project utilizes Graphics Processing Units (GPUs) to compute radiograph simulations for arbitrary objects. The generation of radiographs, also known as the forward projection imaging model, is computationally intensive and not widely utilized. The goal of this research is to develop a massively parallel algorithm that can compute forward projections for objects with a trillion voxels (3D pixels). To achieve this end, the data are divided into blocks that can each fit into GPU memory. The forward projected image is also divided into segments to allow for future parallelization and to avoid needless computations.

1 Introduction

Radiography is the use of electromagnetic radiation other than visible light for imaging. In this work, an object is probed with x-rays and the strength of the transmitted x-rays is measured. We model the transmission of the x-rays with Lambert-Beer's law [1]. Given the material linear attenuation coefficient $\mu(\varepsilon, \vec{x})$ as a function of x-ray energy, ε , and location, \vec{x} , the transmitted x-ray amplitude, $I(\varepsilon)$, is given by

Lambert-Beer's Law of Attenuation

$$I(\varepsilon) = I_0(\varepsilon) \exp \left(-\|\vec{b} - \vec{a}\| \int_0^1 \mu(\varepsilon, t\vec{b} + (1-t)\vec{a}) dt \right)$$

where \vec{a} is the x-ray source location, \vec{b} is the location of the appropriate detector element, and $I_0(\varepsilon)$ is the initial x-ray amplitude at energy ε . Forward projection is not often utilized for industrial sized Computed Tomography (CT) images as the trillion voxel (teravoxel) images are too large to fit in device memory. The goal of CT is to use multiple X-Ray images of an object from different angles to reconstruct the 3 dimensional attenuation data of the object [1]. Forward projection is useful to create an iterative algorithm because it allows the computer to compare the X-Ray images that would be created by the reconstructed object with the actual X-Ray images from the CT scanner.

1.1 GPU architecture

There are many differences between GPU and CPU architectures. GPUs generally outperform CPUs on a strict FLOPS per cost or power basis. However, it is more difficult to utilize the greater processing power in a GPU. GPUs generally operate with a single instruction multiple data (SIMD) architecture while CPUs generally operate as multiple SISD processors. GPUs have many more cores than CPUs but each individual core is less powerful with less memory and a slower clock speed. Thus, while GPUs are more effective at performing parallel computations on large data sets, CPUs are generally more efficient at processing data which is highly sequential or branching [2].

2 Methods

We attempt to evaluate the transmission for every path that starts at the radiation point source and ends at a pixel on the detector plane, which is illustrated in figure 1. The object we are imaging has an attenuation value at every position, but because the computations are done on a finite computer, the attenuation is sampled at a finite number of positions. However, evaluating the transmission integral requires attenuation values at locations which are not exactly specified by the data. Thus, we use trilinear interpolation to get these intermediate points as shown in figure 1. GPUs have efficient hardware interpolation due to the usefulness of interpolation in

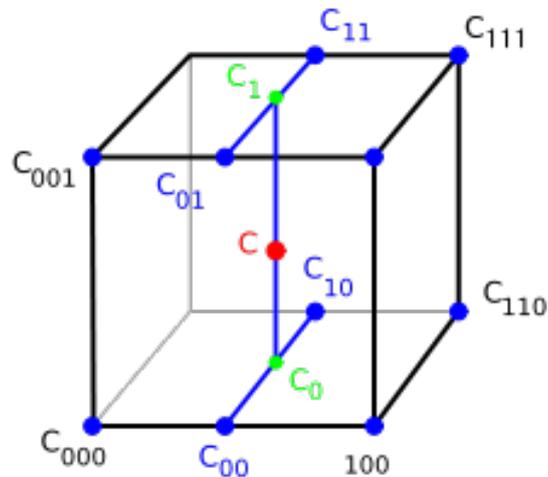


Figure 1: Trilinear Interpolation

computer graphics. However, it is necessary to use texture memory to make use of the hardware interpolation. Any portion of global memory can be declared to be texture memory. One limitation is that a 3D texture cannot have more than 4096 elements in any dimension. Also, since it is global memory, it is limited by the GPU memory which is a few gigabytes.

2.1 Partitioning the Data

There are many different ways to partition the data to fit into GPU memory. The first method developed did not divide the memory into segments at all so the maximum size was approximately half a gigavoxel. The second method divided the data into slices which are small on one dimension. The size of the data processed was limited by the maximum texture size (4096^3 elements). This limited the volume size to $(2^{12})^3 = 64$ gigavoxels. The final method partitioned the data in all dimensions allowing arbitrary sized volumes.

2.2 Computing the Integrals

After the data is partitioned, we must compute the attenuation integrals. Without partitioning, we can evaluate the integrals in a single pass. However, with partitioning we need to divide the integral computation over each segment and sum the results. With the second method of partition, the integral limits are simple to compute because all rays intersect the voxel slices at the y faces. When we divide over all three dimensions, it is possible that the integration path will intersect any face of a voxel volume. In this case, the limits of integration are computed for each path at every voxel chunk. Another complication arises when the forward projection image is partitioned as well. Each voxel chunk will affect a portion of the projected image. A bounding box is computed for the region of influence for each voxel chunk. This data format should prove to be more useful when the code is utilized as it makes the incorporation of different viewing angles easier.

3 Evaluation of Computer

The algorithm was run on a computer with 16 gigabytes of RAM. The computer uses 2 Xeon E5540 processors clocked at 2.53 GHz with 4 cores and 8 threads each. The program only uses one core. The GPU was an nVidia GTX 690 which is a dual GPU card. Each GPU had 2 gigabytes of RAM, 1536 CUDA cores, and a base clock rate of 1006 megahertz.

4 Results

The program was able to generate simulated radiographs for volumes as large as 4 teravoxels. The program utilized 12-13 percent of the CPU processing power which corresponds to one core being fully utilized. The chart in figure 3 shows the computation time for voxel volumes from 500^3 to 8000^3 voxels. Figure 2 depicts a forward projection of a one gigavoxel data set depicting a 3 dimensional grid of spheres. Each sphere attenuates the X-Rays and the dark regions between spheres demonstrate attenuation when an X-Ray passes through multiple spheres. A basic analysis of the computation time showed that the algorithm was able to compute forward projections in a time that is proportional to the total number of voxels. Performance was better when the volume was divided into 500^3 subvolumes than 250^3 subvolumes.

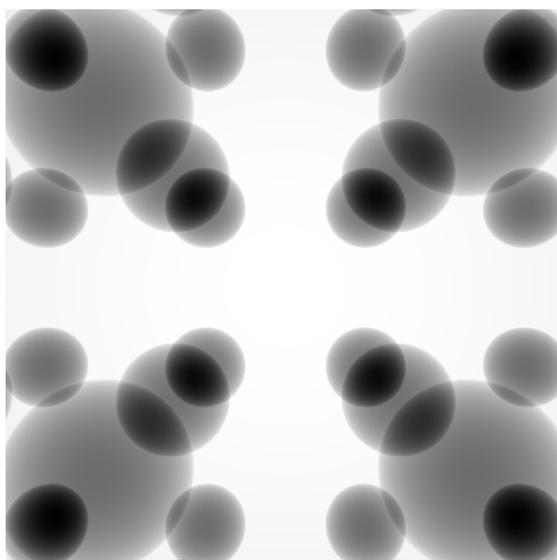


Figure 2: 1 Gigavoxel Projection

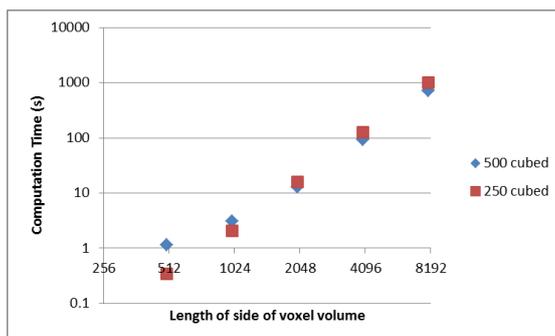


Figure 3: Computation Time

5 Other Accomplishments

I also performed several other tasks that were important to my supervisor and organization while not strictly related to designing the forward projection algorithm. I investigated the computational capacities of the Raspberry Pi, a credit card sized computer. I experimented with the numerical stability of math on the device and refreshed my C programming skills. I presented a poster on the forward projection research at the Student Intern Symposium. I networked with other researchers at Sandia and discussed future research topics. The most important skill I learned at this internship was parallel programming in CUDA.

6 Acknowledgments

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



References

- [1] Harrison H. Barrett and Kyle J. Myers. *Foundations of Image Science*. 2004.
- [2] Cliff Woolley. Cuda overview. <http://www.cc.gatech.edu/vetter/keeneland/tutorial-2011-04-14/02-cuda-overview.pdf>, 2011.