Low-dose X-ray CT Image Reconstruction on GPUs

Aditya Maheshwari
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
aditya.maheshwari@scs.carleton.ca

October 2, 2019

1 Introduction

This project focuses on model based iterative methods for XCT image reconstruction on GPUs. The startup paper cuMBIR ([4]) provides a theoretical state of the art solution that allows each thread to handle the reconstruction of one X-ray (slice) of the 3D object, and then suggests using one graph colouring algorithm and one sub-graph partitioning algorithm that groups different X-ray projections first to reduce memory collisions and then reduce non-coalesced memory accesses. Using this mapping, the GPU is then used to iteratively update the image by using either iterative coordinate descent or stochastic gradient descent to minimize the objective function that can produce the image required. In this project I will attempt to recreate the results of the paper on the In this project I will first attempt to recreate the results of the paper (but likely will have to change the 3rd component in the case that the architectures do not match, they use [NVIDIA Volta 100 GPU, resulting in a 1.48X speedup]), and then further explore the first point related to parallelism granularities and setting different hyper-parameters for ICD and SGD, and also exploring some methods of further parallelizing ICD because of some properties of X-ray data. After that if time permits I will explore to see if there is a way to parallelize this (or another) graph colouring/partitioning step potentially so it can run in batches. Finally, (again if time permits) I would like to use the new method on some slightly different datasets.

2 Literature Review

Computed tomography is a non-destructive way method of obtaining three-dimensional visualizations by using the variation of X-ray attenuation within objects to approximate their densities [3]. Minimizing radiation dose and increasing reconstruction speed and accuracy in XCT are important as they reduce wait time for patients, minimizes energy required per scan (in a world where the number of scans are frequently increasing), and protects patients as well (in the case of medical XCT scans) [**?**]. Furthermore, with the amount of detailed and quality data advanced X-ray technology can provide, we are now able to use these methods to better understand the skeletal system and blood supply by examining bone tissue at all five orders of magnitude of size required to get an accurate picture [1]. These solutions will allow for solutions to many more forms of disease both experimentally and theoretically, but can only work if the data can be processed in a reasonable amount of

time. This paper focuses on speeding up image reconstruction time by parallelizing MBIR methods on Graphical Processing Units (GPUs).

XCT methods involve an X-ray source, and detectors which measure the extent an X-ray has been attenuated (diminished) by passing through the object. By sending X-rays through the object from a series of views (slices or planes through the object), and measuring how much the X-ray attenuates through each slice, we can combine these slices to get a three-dimensional approximation of the objects thickness and properties. The decrease in intensity across linear paths defined by the position of the X-ray source at every slice is characterized by Beer's Law:

$$I = I_0 exp(-\mu x)$$

where μ is the material's linear attenuation coefficient, I_0 is the initial intensity, and x is the length of the X-ray path. If there are multiple materials present, μ can be indexed, and through the many different X-rays taken this can still be approximately solved [3]. The tomographic data shows which voxels (3-D pixels) of the object each X-ray (slice) travels through as well as the different attenuation coefficients associated. Currently, most reconstructions are model-based iterative methods (MBIR), which focus on updating each voxel in batches based on which slices arrive. Once the scanned object is discretized, we are effectively taking an inner product where the first vector is the intercepts of the pixels along the path and the second vector is the attenuation coefficients of these voxels. This is a multiplication of matrix and vector, where the matrix holds the paths of all X-rays, (i.e holds a 1 at all rows/columns where the X-ray projection i intersects the internal image voxel j, and 0 if this does not intersect), and the vector holds the attenuation coefficients. Because only a fraction of the voxels are intersected by any individual X-ray, this is a sparse matrix.

The first challenge in speeding up this process is to select a GPU-friendly solver to minimize the errors in reconstruction of the image as well as converge in a reasonable time; the solvers considered here are iterative coordinate descent (ICD) and stochastic gradient descent (SGD), along with exploring different parallelism granularities for the GPU kernel design to fully exploit all GPU resources. The second challenge is dealing with the irregular geometric relationship between different X-rays in terms of the parts of the scanned object they help recreate; this is solved by building a thread mapping algorithm (which is in fact a graph colouring problem followed by a graph partitioning algorithm) to map projections to threads and to reduce memory collisions and non-coalesced memory accesses. The third optimization done involves placing read-only information (like tomographic data) in to read-only texture memory; since we do not change the data recorded at any point this will improve memory throughput.

2.1 GPUs

A GPU has several streaming microprocessors (SMs) connected by a global memory through an interconnection network. Each SM has many registers, some shared memory, and single instruction multiple data (SIMD) pipelines. The global memory has large amounts of fixed size continuous segments; when code that is executed needs global memory, it coalesces (groups together) the memory accesses of threads within that particular warp into one or more memory transactions depending on the size and distribution of memory accessed by each thread. If multiple non-atomic instructions write to the same memory address for more than one of the threads, only one of the threads can finish the write operation (usually one is chosen "at random"), resulting in a memory collision. A collision can also

occur when threads from different warps try writing to the same global address at the same time. If memory accesses within a warp reside in multiple segments, this will require multiple memory transactions with data that is not necessary for each individual thread to access; this is known as non-coalesced memory access. Any memory transaction usually causes some latency, therefore irregular memory accesses are usually the largest factor in degrading performance.

GPUs are extremely effective at speeding up repetitive data processing seen in loops and apply statements. These can generally work very well for situations like XCT reconstruction, because we are processing each X-ray's associated data in a very similar manner. However, a few general rules are important to follow when using GPUs to take advantage of improved performance [5]:

1. Each pipeline should focus on outputting a single data element (one voxel) rather than many output data elements (multiple voxels), so that many inputs are processed in parallel to produce one output, rather than being forced to write accross the entire memory into many outputs

2. Conditional statements significantly restrict the number of workers that can run

3. Have sufficient data to fill pipelines as GPUs tend to be abundant in bandwidth instead of latency

4. Perform extra computation rather than lookup values from a table as GPUs are extremely fast at computation and slow down significantly when they have to search accross a lot of memroy at each step

Modern GPUs also come with texture memory, which were originally designed for graphics applications where memory access patterns have a spatial locality [5]. Because all the measurements from the data are read-only, these can sit in texture memory (or the L1 cache), and therefore can be accessed faster than they would be as a part of global memory.

Next we look in more detail at the general structure of MBIR methods, and then explain how these can be efficiently processed on GPUs.

2.2 MBIR

MBIR methods minimize an objective function by iteratively updating an image, using Bayesian estimation and regularization [7]. MBIR provides a much higher quality image than other methods as well as a significantly lower X-ray dose, but requires significantly larger amounts of computation than other methods, rendering it currently impractical for most use cases [7]. Because the multiple X-rays (slices) have an irregular geometric relationship (they have a "random" pattern of collisions where two slices update the same voxel), this can traditionally be hard to paralelize for a GPU; as GPUs do not perform efficiently when it constantly has to search across memory for a value to update or when multiple threads try to write to the same memory value at the same time. However, multi-core CPUs tend to take an extremely long time doing this update, for example on a 3D volume with size 512x512x512 they can take more than half an hour to reconstruct the image [9]. Mathematically, this objective function can be written as minimizing

$$E(f) = D(g, Af) + \alpha R(f)$$

where $D(g, Af)$ is the fidelity term based projection measurements, $R(f)$ is the regularizer based on the prior knowledge, and α is the weight parameter of regularization. This regularizer is different for different models. In the paper [4] they use the Mumford-Shah functional as the regularizer, but it makes sense to test multiple regularizers.

In summary, first we we propose unified thread mapping algorithm to optimize memory collision problem and non-coalesced memory access problem (which occurs due to the irregular geometric relationship between X-rays), then we use parallel ICD or SGD to minimize the objective function above as well as explore different parallelism granularities for GPU kernel design (to minimize the objective function), and finally we propose a series of architecture level optimizations to further improve performance including mixed-precision computing and on-chip memory optimization. In the cuMBIR paper [4], using a NVIDIA Volta 100 GPU they are able to achieve a 1.48X speedup on the previous state of the art result [8]. Because our architecture is a little different, and we can use different sub-optimizations at each step.

Next, we discuss the thread mapping algorithm used to reduce memory collisions and non-coalesced memory accesses for this problem.

2.3 Thread Mapping Optimization

Different X-rays will intersect on some voxels, but we don't want multiple threads updating the same voxels simultaneously - whether these are threads in the same warps or different warps. Threads in different warps generally will not attempt to update the same voxel, because the GPU has a scheduler which is fairly optimized and will avoid such collisions. This means the real challenges is preventing memory collisions from the same warp meaning we need a way to group the X-rays so that two X-rays in the same group do not attempt to write to the same voxels at the same time. At the same time, if we choose X-rays that are far away to update in the same warp, we will have many non-coalesced memory accesses which can also slow down performance. Therefore, a collision friendly scheme results on non-coalesced accesses, and the same vice versa, meaning this is not a trivial problem to solve.

We can make this a mathematical problem by creating a graph where each node represents an X-ray, and each edge between two nodes holds a weight representing how many voxels they share. We then want to find a colouring to reduce inter-warp memory collisions as this causes the largest delay, and then find a partition within each colour minimizing the non-coalesced memory accesses. The initial colouring problem for inter-warped memory can be done with any graph colouring algorithm - in the paper [4] they use the Welsh-Powell algorithm. Each subgraph with a different colour now needs to be optimized to reduce non-coalesced memory accesses, where the edge between two nodes represents the number of memory addresses that can be coalesced. For two X-rays, this is the same as the number of voxels that can reside in the same segments. We want to ideally partition nodes from the same colour into sub-graphs containing 32 nodes (representing 32 X-rays, meaning each thread in the warp is responsible for one X-ray). After all components have been processed, we can map the X-rays into a vector to form warps and then distribute warps into multiple thread blocks evenly (all of which can be done offline). This scheme gets transferred to the GPU and only needs to be transferred once as the whole algorithm is only related to the geometry of the XCT scanner and object size.

Next, we address parallel SGD and ICD and GPU Kernel Design in the context of this problem.

2.4 Parallel SGD and ICD and GPU Kernel Design

Stochastic Gradient Descent and Iterative Coordinate Descent are algorithms which are both designed to find the local minimum of an objective function. Stochastic gradient descent works by taking the gradient with respect to all the arguments for the objective function and then moving a pre-determined step size in the opposite direction of the gradient until the net change in position falls inside of some predetermined error value ϵ. Iterative coordinate descent find which variable has the steepest descent and then moves until the gradient of that particular value is zero but only in one direction at one time. We can see in [6] that we are able to parallelize the process of minimizing a sparse objective function in an asynchronous manner.

Asynchronous parallel SGD in the context of this problem will choose one projection measurement to update at each step. At each step all voxels on the path of one X-ray are updated in order to minimize the objective function [8]. ICD, on the other hand, will pick a particular voxel to update the image at each step with every single X-ray that goes through that voxel. Because each voxel that is updated in SGD is visible to all other threads immediately, there are less chances of a memory collision when using SGD, and less information about the entire matrix A is required at each step as well. There has been work on multi-voxel updating for ICD in [2] by identifying independent voxels offline ahead of time, which in particular use cases can work well. A common problem that remains with ICD is the resulting image does not look as smooth as when using SGD, but this still merits further exploration, largely because ICD methods theoretically can converge more quickly than gradient descent in many situations and do not depend on any preselected learning rates. We set each thread as a worker on one particular X-ray, and use the thread mapping algorithm described above to ensure that memory is accessed efficiently.

References

[1] Anika Grüneboom, Lasse Kling, Silke Christiansen, Leonid Mill, Andreas Maier, Klaus Engelke, Harald H Quick, Georg Schett, and Matthias Gunzer. Next-generation imaging of the skeletal system and its blood supply. *Nature Reviews Rheumatology*, pages 1–17, 2019.

[2] Sungsoo Ha and Klaus Mueller. A gpu-accelerated multivoxel update scheme for iterative coordinate descent (icd) optimization in statistical iterative ct reconstruction (sir). *IEEE Transactions on Computational Imaging*, 4(3):355–365, 2018.

[3] Richard A Ketcham and William D Carlson. Acquisition, optimization and interpretation of x-ray computed tomographic imagery: applications to the geosciences. *Computers & Geosciences*, 27(4):381–400, 2001.

[4] Xiuhong Li, Yun Liang, Wentai Zhang, Taide Liu, Haochen Li, Guojie Luo, and Ming Jiang. cumbir: An efficient framework for low-dose x-ray ct image reconstruction on gpus. In *Proceedings of the 2018 International Conference on Supercomputing*, pages 184–194. ACM, 2018.

[5] Klaus Mueller, Fang Xu, and Neophytos Neophytou. Why do commodity graphics hardware boards (gpus) work so well for acceleration of computed tomography? In *Computational Imaging V*, volume 6498, page 64980N. International Society for Optics and Photonics, 2007.

[6] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[7] Amit Sabne, Xiao Wang, Sherman J Kisner, Charles A Bouman, Anand Raghunathan, and Samuel P Midkiff. Model-based iterative ct image reconstruction on gpus. *ACM SIGPLAN Notices*, 52(8):207–220, 2017.

[8] Xiao Wang, Amit Sabne, Sherman Kisner, Anand Raghunathan, Charles Bouman, and Samuel Midkiff. High performance model based image reconstruction. In *ACM SIGPLAN Notices*, volume 51, page 2. ACM, 2016.

[9] Xiao Wang, Amit Sabne, Putt Sakdhnagool, Sherman J Kisner, Charles A Bouman, and Samuel P Midkiff. Massively parallel 3d image reconstruction. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 3. ACM, 2017.