Fast GPU-Based Approach to Branchless Distance-Driven Projection and Back-Projection in Cone Beam CT

Daniel Edward Schlifske
Marquette University

A FAST GPU-BASED APPROACH TO BRANCHLESS DISTANCE-DRIVEN PROJECTION AND BACK-PROJECTION IN CONE BEAM CT

by

Daniel Schlifske, B.S.

A Thesis submitted to the
Faculty of the Graduate School, Marquette University, in Partial Fulfillment of the
Requirements for the Degree of Master of Science

Milwaukee, Wisconsin

December 2015

ABSTRACT

A FAST GPU-BASED APPROACH TO BRANCHLESS DISTANCE-DRIVEN PROJECTION AND BACK-PROJECTION IN CONE BEAM CT

DANIEL SCHLIFSKE, B.S.

MARQUETTE UNIVERSITY, 2015

Modern CT image reconstruction algorithms rely on projection and back-projection operations to refine an image estimate in iterative image reconstruction. A widely-used state-of-the-art technique is distance-driven projection and back-projection. While the distance-driven technique yields superior image quality in iterative algorithms, it is a computationally demanding process. This has a detrimental effect on the relevance of the algorithms in clinical settings.

A few methods have been proposed for enhancing the distance-driven technique in order to take advantage of modern computer hardware. This study explores a two-dimensional extension of the branchless method, which is a technique that does not compromise image quality. The extension of the branchless method is named “pre-projection integration” because it gets a performance boost by integrating the data before the projection and back-projection operations. It was written with Nvidia’s CUDA framework and carefully designed for massively parallel graphics processing units (GPUs).

The performance and the image quality of the pre-projection integration method were analyzed. Both projection and back-projection are significantly faster with pre-projection integration. The image quality was analyzed using cone beam CT image reconstruction algorithms within Jeffrey Fessler’s Image Reconstruction Toolbox. Images produced from regularized, iterative image reconstruction algorithms using the pre-projection integration method show no significant artifacts.

TABLE OF CONTENTS

I. INTRODUCTION

This section serves as an introduction to the context, goals, and organization of this study. First, there is a brief introduction to Computed Tomography (CT) and iterative image reconstruction. In this section, the key operations examined in this study, projection and back-projection, are explained. Then, the next section outlines the high-level objectives of this study. Finally, the organization of this document is outlined.

A. Brief Introduction to Computed Tomography

The purpose of this section is to provide readers with enough background to understand the concepts of projection and back-projection. It explains data collection in CT and also defines terms such as X-ray source, detector, image, and sinogram. Furthermore, this section briefly introduces iterative image reconstruction, which necessitates advanced projection and back-projection.

In CT, an X-ray tube and detector rotate around an object. The X-ray tube emits X-rays, which are attenuated by the object between the tube and the detector. There are three main modes of data collection in CT: parallel beam, fan beam, and cone beam (Hsieh, 2009). All three modes are shown in Fig. 1. In parallel beam mode, the X-ray source is modeled as a line of equally spaced sources, each of which emits parallel beams of X-rays that hit a row of detector cells. The fan beam mode differs from the parallel beam mode in two ways. First, there is only one X-ray source, which is modeled as a single point. Second, the X-ray beams emitted by the source “fan-out” and thus are not parallel. Finally, the third mode of data collection is the cone beam mode. It is essentially a 3D collection of many fan beams.

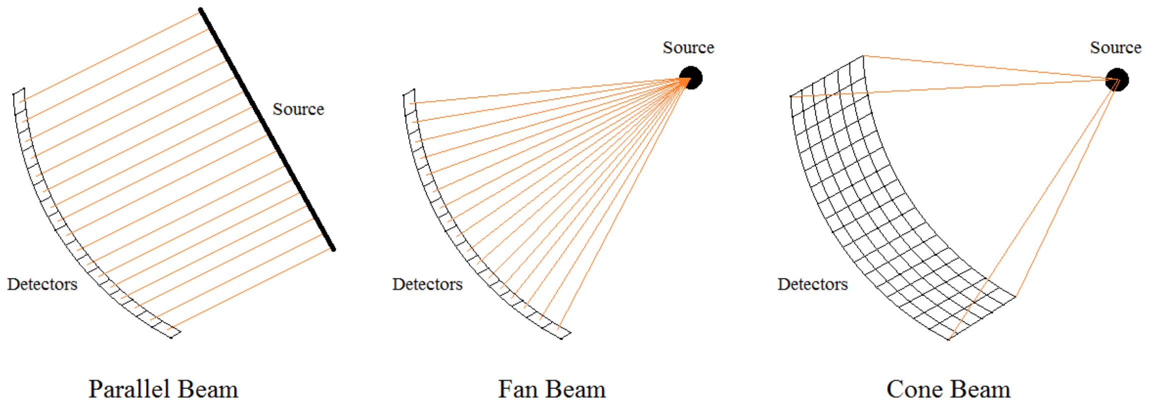


Fig. 1. Parallel beam (left), fan beam (center), and cone beam (right) CT geometries.

Data collected by the detector is stored in a sinogram. A sinogram is a 3D stack of data. Each piece is called a view and contains the data collected when the X-ray tube and detector are located at a certain angle. For any particular view, the sinogram stores the signal collected from the detector along its rows and channels. Image reconstruction is the task of using the data in the sinogram to generate an image of the radiodensity of the object between the X-ray source and the detector. There are two main classes of reconstruction algorithms: analytical and iterative.

Analytical reconstruction algorithms are generally algorithms that are non-iterative and require a closed-form solution (Fessler J. A., Analytical Tomographic Image Reconstruction Methods, 2009). These algorithms are frequently based on some variant of a Filtered Back-Projection (FBP) technique (Fessler J. A., Analytical Tomographic Image Reconstruction Methods, 2009) (Kak & Slaney, 2001). One particular example of an analytical reconstruction algorithm that does *not* use FBP is the direct Fourier method, which is based on the Fourier slice theorem (Fessler J. A., Analytical Tomographic Image Reconstruction Methods, 2009). FBP transforms the sinogram data back into an image by first filtering the data in frequency space and then smearing the data back along the projection lines (Fessler J. A., Analytical Tomographic Image Reconstruction Methods, 2009) (Kak & Slaney, 2001). FBP algorithms are relatively quick but generally only produce acceptable results when the data is uniformly spaced and closely sampled (Thibault J.-B. , Sauer, Bouman, & Hsieh, 2003). Despite their simplicity, certain analytical algorithms based on FBP have been very successful. One prominent example is the Feldkamp, Davis, and Kress (FDK) algorithm for cone beam CT (Feldkamp, Davis, & Kress, 1984). Decades after its introduction, the FDK algorithm is still being optimized for the latest computer hardware (Scherl, Keck, Kowarschik, & Hornegger, 2007) (Scherl, et al., 2007).

Iterative reconstruction algorithms attempt to find the image that best fits the data that was actually acquired. They use statistical techniques, model imperfections in the system, and take into account the discrete acquisition of the data (Hsieh, 2009). Unlike analytical methods which rely solely on back-projection, iterative methods also use projection. In projection, the image estimate is projected onto the detector to simulate data acquisition. Fig. 2 shows the relationship between an image volume and a sinogram as well as the projection operations used to generate one from the other. Iterative algorithms generally perform multiple iterations of projection and back-projection in order to converge towards a refined image.

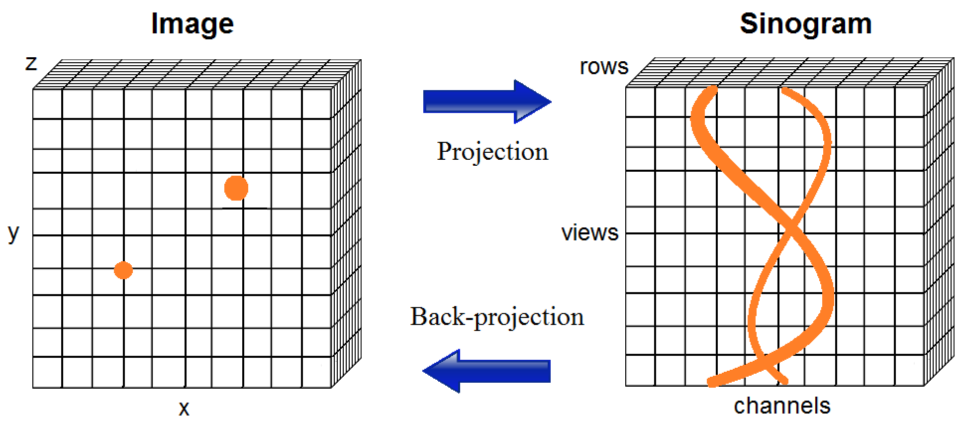


Fig. 2. An image volume (left) and a sinogram (right). The blue arrows illustrate projection and back-projection operations.

B. Study Objectives

The objective of this work is to explore a novel technique for projection and back-projection in cone beam CT that significantly speeds up image reconstruction algorithms while not compromising the image quality. There are many different models for projection and back-projection, and most of today's high quality image reconstruction algorithms rely on fairly sophisticated such models. Therefore, a key objective of this study is to focus on a model for projection and back-projection that is currently being used in high quality image reconstruction algorithms. Another goal of this study is to focus on characterizing the speed or performance of projection and back-projection for state-of-the-art processor technologies. Massively parallel graphics processing units (GPUs) are increasingly being used for computationally intensive tasks such as image reconstruction. Therefore, this study will focus on projector and back-projector performance on GPUs. Finally, the last objective of this study is to do a thorough analysis of the image quality effects of the techniques proposed. This analysis should examine the image quality effects of new or modified operations as well as the image quality of the output of the entire image reconstruction algorithm.

C. Organization of Thesis

The remainder of this document is organized as follows. Section II is a survey of the work that has been done in the field of projection and back-projection as well as the optimization of those operations. Section III describes the pre-projection integration algorithm, which is the technique focused on in this study. Section IV contains the results from the study, which includes both performance results and image quality results. Finally, Section V is a conclusion which summarizes the key points and findings of the study.

II. RELATED WORK

Projection and back-projection are integral to tomographic imaging modalities such as CT. These operations have become very sophisticated over time. As shown in this section, a great deal of analysis has been devoted not only to projection and back-projection operations, but also to their application to certain algorithms and their efficiency on different computational platforms.

This section is organized as follows. First, it explores a few of the more noteworthy projection models. Second, it examines state-of-the-art image reconstruction algorithms to determine which projection models are the most prevalent. While more accurate models may exist, distance-driven projection emerges as the overwhelming favorite of most modern algorithms. Third, this section explores ways in which distance-driven projection has been optimized for parallel computation platforms such as GPUs. And finally, this section examines how other projection methods such as ray-driven or pixel-driven have taken advantage of GPUs. While the focus of this study is distance-driven projection, it is nevertheless useful to have an understanding of the techniques used to optimize other projection methods.

A. Projector Models

The pixel-driven method is popular for implementing the back-projection operation, especially in FBP algorithms. In pixel-driven back-projection, the output for a given pixel is essentially the sum along the sinusoid corresponding to that pixel in the sinogram (Hsieh, 2009). Because the sinogram elements do not correspond exactly to a pixel, interpolation of the sinogram elements must be performed. Pixel-driven back-projection is illustrated in Fig. 3.

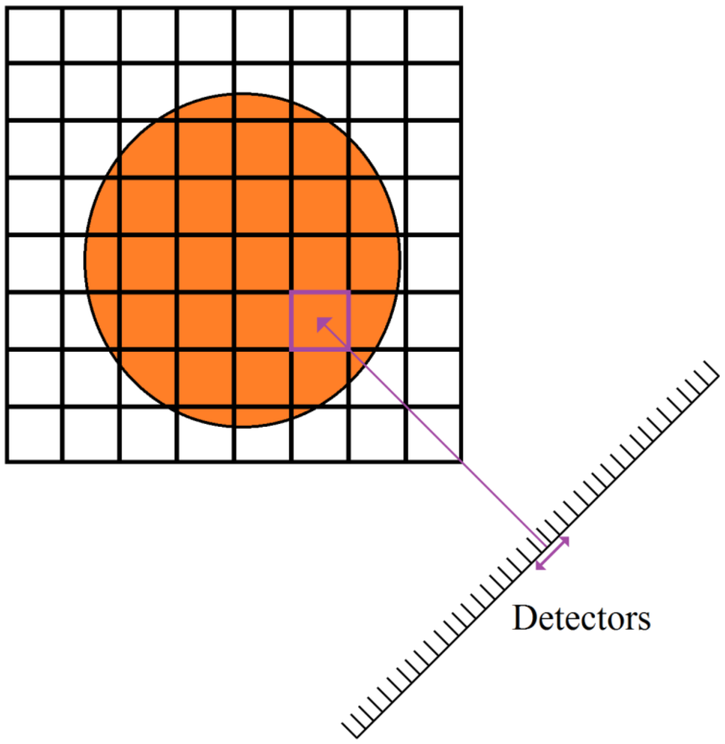


Fig. 3. Pixel-driven back-projection. The center of a pixel is mapped onto the detector. The values of the nearest projection data elements are interpolated to calculate the value of the pixel.

Ray-driven projection is a frequently used model. In CT, ray-driven projection consists of using the attenuation values in image voxels to calculate the attenuation of an X-ray moving from the source to the detector (Hsieh, 2009). It produces significant artifacts when used for back-projection, so it is typically used as a model for projection only (Fessler J. A., Analytical Tomographic Image Reconstruction Methods, 2009). In projection, an image volume is projected onto a detector. Ray-driven projection is illustrated in Fig. 4.

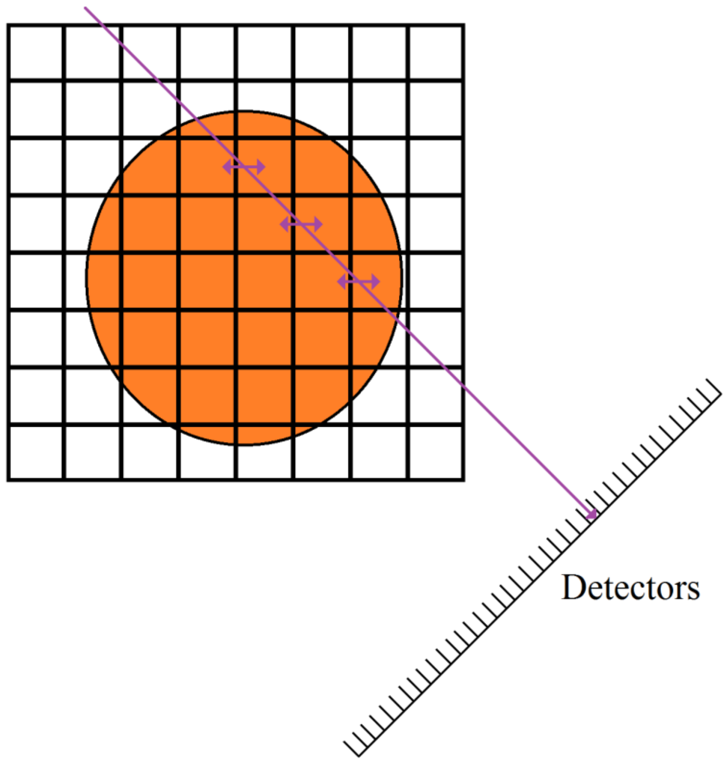


Fig. 4. Ray-driven projection. The center of each detector cell is projected through the image. At each row, the image intensities are interpolated and added to the projection data.

The intuitive approach to ray-driven projection is the "grid-interpolated" scheme (Xu, Fast implementation of iterative reconstruction with exact ray-driven projector on GPUs, 2010) (Xu & Mueller, A comparative study of popular interpolation and integration methods for use in computed tomography, 2006). In the grid-interpolated scheme, the image volume is sampled at many different points over a ray. Interpolation is performed at each point, and the values are summed up along the ray. The trapezoid rule is then used to calculate the line integral of the ray. The grid-interpolated method is not an exact method: the nonzero distance between samples and the interpolation leads to imperfections.

In 1984, Siddon published a method to calculate the exact radiological path through a 3D image (Siddon, 1985). He did not introduce the concept of projection based on "rays," but no prior publication offered an exact solution to the 3D problem that was both fast and consistent. For each ray, Siddon’s algorithm finds the intersections with voxels. For each intersection, the length of the ray within the voxel is multiplied by the value of the voxel. These values are then summed over the entire ray.

De Man and Basu introduced the concept of distance-driven projection and back-projection (De Man & Basu, Distance-driven projection and backprojection, 2002) and later extended it to 3D (De Man & Basu, Distance-driven projection and backprojection in three dimensions, 2004). Distance-driven projection is essentially a combination of ray-driven projection and pixel-driven techniques. In 2D distance-driven projection, pixel boundaries and detector boundaries are mapped to a common axis. Then, the length of the overlap between detector and pixel boundaries is used to determine the weight used in the projection of a pixel or the back-projection of a detector element. Fig. 5 shows the overlap created when detector boundaries are mapped to an image grid. In De Man and Basu’s 3D distance-driven model, a common plane is used instead of a common axis. The boundaries of voxels in both the transaxial and axial directions are mapped to the plane along with the boundaries of the detector channels and rows. The area of the overlap is used to determine projection weights.

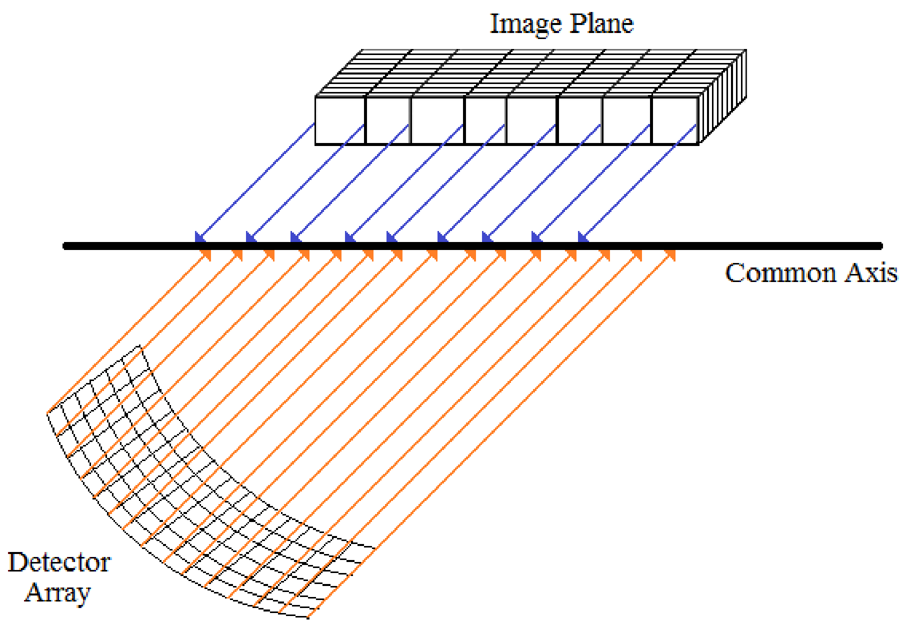


Fig. 5. Voxel boundaries (blue) and detector boundaries (orange) are mapped onto a common axis in the distance-driven model. The length of the overlap determines the projection or back-projection weight.

De Man and Basu demonstrated that there are two main advantages to distance-driven projection. First, distance-driven projection produces fewer image artifacts than previous projection methods because the areas of overlap are used for projection weights (De Man & Basu, Distance-driven projection and backprojection in three dimensions, 2004). Second, the distance-driven projection and back-projection operations are matched, or symmetric. That means that they can be used in iterative reconstruction algorithms where many projection and back-projection operations are done (De Man & Basu, Distance-driven projection and backprojection in three dimensions, 2004).

The separable footprint method is a more recent projection model that shares some similarities with distance-driven projection (Long, Fessler, & Balter, 2010). In the separable footprint method, the footprint of a pixel onto a detector is approximated by a trapezoidal function in the transaxial direction and either rectangular or trapezoidal functions in the axial direction, depending on the cone angle. When a trapezoidal function is used for both the transaxial and axial footprints, the method is called “SF-TT.” When a trapezoidal function is used for the transaxial footprint but a rectangular function is used for the axial footprint, the method is called “SF-TR.”

The separable footprint method is effective for a two reasons. One reason is that its trapezoidal footprint functions are successful at modeling detector blur. Detector blur occurs because voxels have finite size. Fig. 6 illustrates the concept of detector blur. At projection angles where the transaxial voxel footprint is more triangular than rectangular, the separable footprint method is more accurate than the distance-driven method (Long, Fessler, & Balter, 2010).

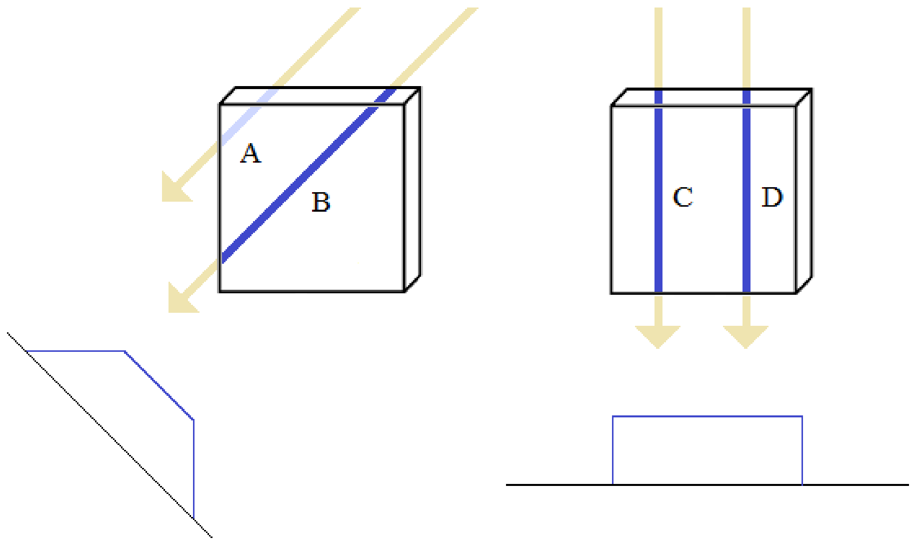


Fig. 6. A simplified, parallel beam illustration of detector blur. On the left is a projection angle of 45˚. Because the length of ray A within the voxel is smaller than the length of ray B within the voxel, the voxel’s footprint blurs, resembling a trapezoid. On the right is a projection angle of 90˚. Rays C and D travel the same distance through the voxel, so the voxel’s footprint more closely resembles a rectangle.

Another reason for the effectiveness of the separable footprint method is that the transaxial and axial footprints can then be treated as 2D separable functions. With this optimization, the separable footprint method can then be optimized for highly parallel hardware such as commercial GPUs. Wu and Fessler showed that the separable footprint projectors can run two to three times faster on a dual-GPU system than on a 24-thread CPU (Wu & Fessler, 2011). Despite its desirable characteristics, most systems tend to favor distance-driven approach over the separable footprints method for reasons discussed below.

B. The Distance-Driven Method in Modern Algorithms

When analyzing how a projection method fits into a reconstruction algorithm, one needs to consider a fundamental trade-off between computational complexity and accuracy. Neither the distance-driven nor the separable footprints methods are exact. Both simplify the footprint of a voxel on a detector: distance-driven uses rectangles and separable footprints uses trapezoids. While the separable footprints method may be slightly more accurate, the distance-driven method can provide sufficient accuracy to meet the image quality goals of an image reconstruction algorithm. Indeed, some very advanced iterative image reconstruction algorithms have used the distance-driven method with success.

For example, Positron Emission Tomography (PET) image reconstruction is very similar to CT image reconstruction. One particular high-quality PET image reconstruction algorithm is based on Ordered Subset Expectation Maximization (OSEM) which uses distance-driven projectors (Manjeshwar, Ross, Iatrou, Deller, & Stearns, 2006). Because most PET detectors are comprised of detector blocks that contain a grid of crystals, the detector boundaries are not evenly spaced. The distance-driven model supports uneven spacing of detectors, allowing the projectors to accurately model the system.

Additionally, Thibault et al. describe how they used the distance-driven method of De Man and Basu with some minor changes in a 3D regularized iterative reconstruction algorithm called "MAP-ICD" (Thibault J.-B. , Sauer, Bouman, & Hsieh, 2007). Instead of mapping detector boundaries onto an image plane, the algorithm projects the voxel's center onto a detector array and then flattens it to create 2D rectangular footprints as shown in Fig. 7. They found that this method produces images without artifacts related to projection.

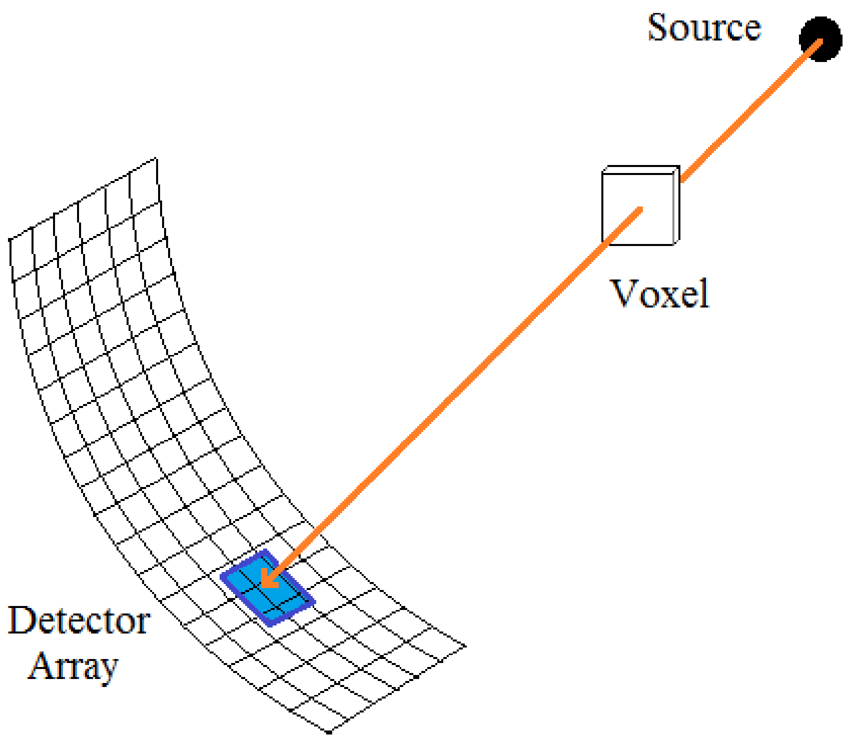


Fig. 7. The projection of a voxel in the MAP-ICD algorithm is shown. The center of a voxel is projected onto a detector array and flattened to create a rectangular footprint.

Therefore, while the distance-driven method may not be the most accurate projection method, it has been very effective in commercially successful iterative image reconstruction algorithms. Additionally, the simplicity of the rectangular footprints of the distance-driven method allows for numerous optimization opportunities. For these reasons, this study will focus on distance-driven projection.

C. Optimizations of Distance-Driven Projectors for GPUs

One of the first efforts at optimizing the distance-driven projectors was attempted on the highly parallel IBM Cell Broadband Engine (Cell BE) processor. Chevalier and Drapkin ported the PET distance-driven projectors to the Cell BE processor in order to make Time-of-Flight PET relevant in clinical settings (Chevalier & Drapkin, 2009). The Cell BE processor contains a PowerPC core and eight Synergistic Processing Elements (SPEs), which are 128-bit vector processors.

Chevalier and Drapkin's first step was to vectorize the inner most loop of the distance-driven algorithm to take advantage of the 128-bit SPEs. Another particularly effective enhancement for the Cell BE was the use of double-buffering to hide the latency associated with moving data from the CPU to the Cell BE. With double-buffering, the data for a future operation is sent to the processor (in this case the Cell BE) while the processor is computing the data from the prior operation. Therefore, double-buffering minimizes the amount of time that the Cell BE is waiting for the data to execute the "compute" steps of the algorithm. In addition, they discovered significant performance improvements by changing the algorithm to work on 2D blocks of the image rather than on one row at a time. This reduced the number of times the data had to be transferred to the Cell BE memory. Overall, their optimized Cell BE code ran ten times faster than code optimized for the Intel Woodcrest CPU.

Gross et al. also have addressed the challenge of distance-driven projection on GPUs (Gross, Heil, Schulze, Schoemer, & Schwanecke, 2009). However, their work focused on novel ways to use GPUs to map voxels onto detector grids rather than on optimizing the distance-driven overlap kernel. They describe how the vertex shaders and texture units on GPUs can be used to quickly transform the parallelogram of a voxels's projection onto a detector grid into a rectangle.

Klaus Mueller has completed a significant amount of work in the field of accelerating projection methods on modern GPU hardware (Mueller & Yagel, Rapid 3-D cone-beam reconstruction with the simultaneous algebraic reconstruction technique (SART) using 2-D texture mapping hardware, 2000) (Mueller, Yagel, & Wheller, Fast implementations of algebraic methods for three-dimensional reconstruction from cone-beam data, 1999). Mueller and Xu surveyed different interpolation and integration methods associated with projections and looked at the distance-driven model, which they referred to as the “box-beam-integrated” method (Xu & Mueller, A comparative study of popular interpolation and integration methods for use in computed tomography, 2006). They briefly discussed using Summed Area Tables as a way to simplify the box-beam integrated technique. However, they ultimately decided to explore enhanced ray-driven techniques.

However, all of the previously mentioned works failed to address one of the most significant shortcomings of the distance-driven method. The distance-driven projection method contains an overlap kernel, in which the pixel and detector boundaries are traversed (De Man & Basu, Distance-driven projection and backprojection in three dimensions, 2004). The overlap kernel contains branches in its code due to different code paths for detector boundaries and pixel boundaries. Branch divergence is inefficient for Single Instruction Multiple Data (SIMD) processors like GPUs and also for pipelined CPUs.

With these inefficiencies in mind, the originators of distance-driven projection, De Man and Basu, proposed a "branchless" approach (Basu & De Man, 2006). The branchless method factors the overlap kernel into three operations: integration of the input signal, linear interpolation of the integrated signal to obtain values at detector locations, and digital differentiation of the integrated signal at the interpolated detector locations.

What follows is a brief explanation of the theory behind branchless distance-driven projection. Recall that for projection, the distance-driven method needs the integral of an image between the boundaries, d_j and d_{j+1}, of a detector. De Man and Basu represent this as shown in Eq. (1), where $p(x)$ is the image intensity at position x.

$$d_{j,j+1} = \frac{1}{d_{j+1} - d_j} \int_{d_j}^{d_{j+1}} p(x)dx \tag{1}$$

They then write the 1D integral of a row of the image as *P*. See Eq. (2). Note that *C*, the constant of integration, is an arbitrary value, which can be chosen according to algorithmic convenience. The variable p_j represents sample point *j* in the image, which is a pixel boundary. The intensity of pixel *j* is represented by $p_{j,j-1}$.

$$P(p_j) = \begin{cases} P(p_{j-1}) + p_{j,j-1} * (p_j - p_{j-1}) & j > 1 \\ C & j = 1 \end{cases} \tag{2}$$

For projection, the sample points are the evenly spaced pixels. Thus, the difference between adjacent pixels $(p_j - p_{j-1})$ is equal to 1. Then, Eq. (2) can be simplified as Eq. (3) shows.

$$P(p_j) = \begin{cases} P(p_{j-1}) + p_{j,j-1} & j > 1 \\ C & j = 1 \end{cases} \tag{3}$$

After linearly interpolating P to obtain values of the function at detector locations d_j and d_{j+1}, the integral from Eq. (1) can be found more simply with Eq. (4).

$$d_{j,j+1} = \frac{1}{d_{j+1} - d_j}\left[P\left(d_{j+1}\right) - P\left(d_j\right)\right] \tag{4}$$

Their experimental results showed that the error from the branchless method compared to the original method was on the order of the numerical precision of the calculations. They also noted that the branchless method requires evenly spaced pixels for projection and evenly spaced detectors for back-projection. An even spacing of pixels is almost always the case. Additionally, in parallel beam and cone beam geometries, the detectors are evenly spaced. A solution is proposed for the 2D fan beam geometry case in which the detectors are not evenly spaced (Basu & De Man, 2006). However, since the focus of this study is cone beam CT, it will not be detailed here.

The pre-projection integration method discussed in Section III is an extension of the branchless method. Pre-projection integration has the potential for more significant performance improvements because it integrates the input signal in two dimensions, not one. However, this also means that more elements contribute to the integral, which in turn creates a need to analyze and possibly mitigate any sources of precision loss. Additionally, several aspects of the pre-projection integration method are designed specifically for modern GPUs.

D. Optimizations of Other Projection Methods for GPUs

The focus of this study is the optimization of distance-driven projection and back-projection on GPUs. However, it is still useful to examine the techniques that have been used to optimize other projection and back-projection methods on GPUs. Some techniques are directly applicable to the distance-driven method. Others are indirectly applicable and therefore can be used to guide the optimization process.

Due to the highly parallel architecture of GPUs, one of the most effective things that can be done is to divide the operation into tens of thousands of tasks (or "threads") that can be computed in parallel. Modern GPUs have thousands of cores. Therefore, to keep the entire device busy, it is generally advisable to split the operation into as many small threads as possible. In addition, there should very few dependencies between threads. GPUs provide synchronization mechanisms and atomic operations. However, they are generally too slow to use. Therefore, a thread's ability to independently execute its own tasks is very important. This strategy has been effectively used to accelerate the filtering and back-projection in both CUDA and Cell BE implementations of the FDK algorithm (Scherl, Keck, Kowarschik, & Hornegger, 2007) (Scherl, et al., 2007).

Limiting memory transactions is also important when optimizing any GPU program. One way to limit memory transactions is to use a GPU's on-chip registers to store incremental updates rather than write them to memory. Ideally, as many registers per thread should be used until it begins to change the multiprocessor occupancy, which is the amount of threads that can simultaneously run on a core. This tradeoff was nicely characterized for a CUDA-based FDK implementation (Scherl, Keck, Kowarschik, & Hornegger, 2007). Christiaens, et al. demonstrate another way to limit memory transactions in their implementation of a Siddon-based projector (Christiaens, De Sutter, De Bosschere, Van Campenhout, & Lemahieu, 1999). Their projector was not designed specifically for GPUs; it was designed to achieve better cache performance on CPUs. Still, they show that computing the Siddon weights incrementally is preferable to storing them in large arrays as Siddon originally proposed.

Another GPU optimization technique is the use of texture memory. Texture memory is used to store textures (sometimes called “image objects”), which originated from graphics programming. In most ways, texture memory behaves exactly like normal memory. However, some GPUs cache texture memory on chip in L1 and L2 caches but do not cache normal memory. The benefits of having data cached can be significant. However, to take full advantage of the cache, threads need to be organized for spatial locality. Spatial locality means that threads with neighboring indices need to read neighboring elements in the textures. This ensures that the elements needed have already been brought into the cache. This has shown to be an effective technique for projection in cone beam CT (Perez, Jimenez, & Thompson, 2014).

In addition to caching, there is another benefit of Texture memory that can be used to give projection and back-projection a performance boost. Modern GPUs perform linear filtering of texture memory with devoted hardware called Texture Units (Wilt, 2013). Consider a bilinear interpolation. A normal computer program would read in the four closest elements, perform the linear weighting in each direction, and calculate the interpolated value. With the Texture Units, a floating point value can be specified as the index for a texture read operation. The GPU uses dedicated hardware to linearly interpolate the four nearest values, which generally makes the operation very quick and efficient. This technique has been effective for ray-driven projection (Xie, Hu, Chen, & Shi, 2015) (Perez, Jimenez, & Thompson, 2014), voxel-driven back-projection (Xie, Hu, Chen, & Shi, 2015), and Siddon-based projection (Xu, Fast implementation of iterative reconstruction with exact ray-driven projector on GPUs, 2010). Hardware-based interpolation provided by Texture Units has also been successfully used to speed up 3D OSEM PET reconstruction on GPUs (Pratx, Chinn, Habte, Olcott, & Levin, 2006).

III. PRE-PROJECTION INTEGRATION ALGORITHM

In this section, the details of the Pre-Projection Integration algorithm are explained. First, in Section III.A, a theoretical overview of the algorithm is given by explaining the data integration and overlap calculation in mathematical terms. Then, in Section III.B, the implementation details, including design and optimization details for GPUs, are discussed.

A. Overview

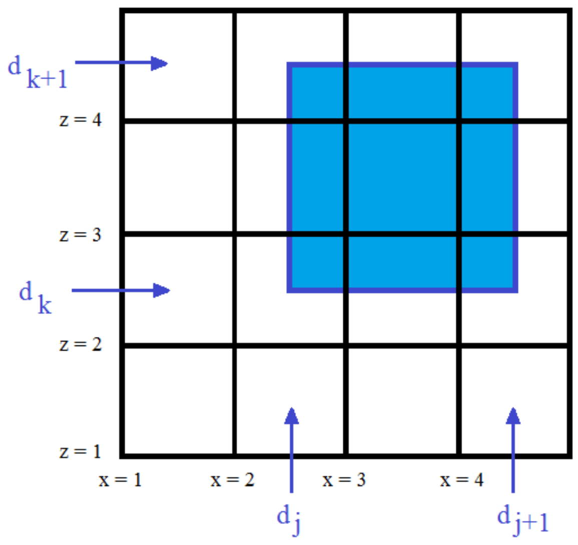
As mentioned in Section II.B, the distance-driven model for projection and back-projection has become the standard for iterative image reconstruction algorithms. During projection, the algorithm requires the sum of the intensities of the voxels that fall within rectangular detector boundaries. Similarly, during back-projection, the algorithm requires the sum of the intensities of sinogram elements that fall within rectangular voxel boundaries. The detector boundaries on the *x*-axis are d_j and d_{j+1}. The detector boundaries on the *z*-axis are d_k and d_{k+1}. The function *p(x, z)* represents the image intensities. Both operations consist of a 2D integral where the bounds of integration are the detector boundaries during projection and the voxel boundaries during back-projection. This is shown in Eq. (5).

$$d_{j,j+1,k,k+1} = \frac{1}{d_{j+1} - d_j} \frac{1}{d_{k+1} - d_k} \int_{d_j}^{d_{j+1}} \int_{d_k}^{d_{k+1}} p(x,z)\, dz\, dx \tag{5}$$

The function $p(x, z)$ represents the image intensities, and thus is piecewise constant. Therefore, the double integral can be calculated as shown in Eq. (6). An intersection operator is used in Eq. (6) to create an overlap kernel. Essentially, all of the image, *p(x,z),* that falls within the rectangular boundary created by the detector boundaries d_j, d_{j+1}, d_k, and d_{k+1} is summed.

$$d_{j,j+1,k,k+1} = \frac{1}{d_{k+1} - d_k} \frac{1}{d_{j+1} - d_j} \sum_x \sum_z p(x, z) \cap d_{j,j+1,k,k+1} \qquad (6)$$

The overlap kernel can be computed as follows. Consider the example in Fig. 8 below. The detector boundaries $d_{j,j+1}$ and $d_{k,k+1}$ are projected onto an image slice with pixel boundaries $x = 1$ through $x = 4$ and $z = 1$ through $z = 4$. The pseudocode on the right of Fig. 8 shows how the overlap kernel is traditionally computed. There are two nested loops that iterate over the image coordinates containing the detector boundaries in both dimensions. For each image coordinate, a weight is calculated that corresponds to the length of the detector boundary within that particular voxel. The voxel's intensity, proportional to both of those weights, is added to a sum.



```
z_start = floor( $d_k$ );
z_end = floor( $d_{k+1}$ );

x_start = floor( $d_j$ );
x_end = floor( $d_{j+1}$ );

for z = z_start to z_end
    weight_z = min( z + 1, $d_{k+1}$ ) - max( z, $d_k$ );
    for x = x_start to x_end
        weight_x = min( x + 1, $d_{j+1}$ ) - max( x, $d_j$ );
        sum += weight_x * weight_z * p(x,z);
```

Fig. 8. Blue detector boundaries projected onto a 2D image slice (left) and pseudocode for the overlap kernel (right). The overlap kernel calculates the sum of the weighted image intensities within the rectangular overlap region.

The overlap kernel can be fairly slow on GPUs. Due to their highly parallel SIMD architectures, GPUs need to have all threads executing the same instructions to achieve peak performance. Branch divergence occurs when threads that are supposed to be executing the same instructions take different branches. When branch divergence occurs on a GPU, certain threads execute “no-op” instructions while waiting for other threads. This leads to longer kernel execution times.

A branchless way to calculate the area of overlap in Eq. (5) would provide a significant speedup for projection and back-projection operations. In order to solve this problem, all of the data is integrated *before* the entire projection or back-projection operation. In effect, this creates a Summed Area Table (SAT), or “integral image” (Crow, 1984). We use the SAT to represent the image during projection and the sinogram during back-projection. See Eq. (7). The bounds of integration, *X* and *Z*, are the variables of *P(X,* *Z)*. So, *P(X, Z)* represents the integral of the image intensities, $p(x, z)$, with coordinates less than *X* and *Z*.

$$P(X, Z) = \int_{0}^{X}\int_{0}^{Z} p(x, z) dz\, dx\ + C \tag{7}$$

Any value for the constant of integration can be chosen. For reasons discussed later, the mean value of the slice, μ, is the optimal value to subtract before integration. See Eq. (8) and Eq. (9). The variables *nx* and *nz* represent the dimensions of the 2D image slice.

$$\mu = \frac{1}{nx * nz} \int_{0}^{nx}\int_{0}^{nz} p(x, z) dz\, dx \tag{8}$$

$$P(X, Z) = \int_{0}^{X}\int_{0}^{Z} [\, p(x, z) - \mu] dz\, dx \tag{9}$$

The use of an SAT could substantially speed up the projection operations. An SAT allows for quick and efficient calculation of the sum of intensities in an arbitrary rectangular region of an image (Crow, 1984). SATs have existed for three decades as a solution to a variety of problems. Indeed, a considerable amount of work has been devoted to efficient SAT creation (Hensley, Scheuermann, Coombe, Singh, & Lastra, 2005). SATs have also been applied to problems in computer vision and computer graphics. For example, prominent examples of SAT usage in computer vision are normalized cross-correlation (Lewis, 1995) and the Viola–Jones object detection framework (Viola & Jones, 2001). The SAT approach can also be used to efficiently compute image histograms for real-time object tracking (Medeiros, Holguín, Shin, & Park, 2010). In computer graphics, SATs are commonly used for surface rendering (Lacroute & Levoy, 1994). However, to the best of our knowledge, SATs have not been applied to any medical imaging problems.

The proposed pre-projection integration method uses an SAT to efficiently calculate Eq. (5). After integrating the data as shown in Eq. (9), the sum of the intensities within the area of overlap, s, can be calculated using the SAT formula illustrated shown in Eq. (10). Fig. 9 shows how Eq. (10) can be used to compute the area in the overlap region from Fig. 8.

$$s = P(j+1, k+1) - P(j+1, k) - P(j, k+1) + P(j, k) \tag{10}$$

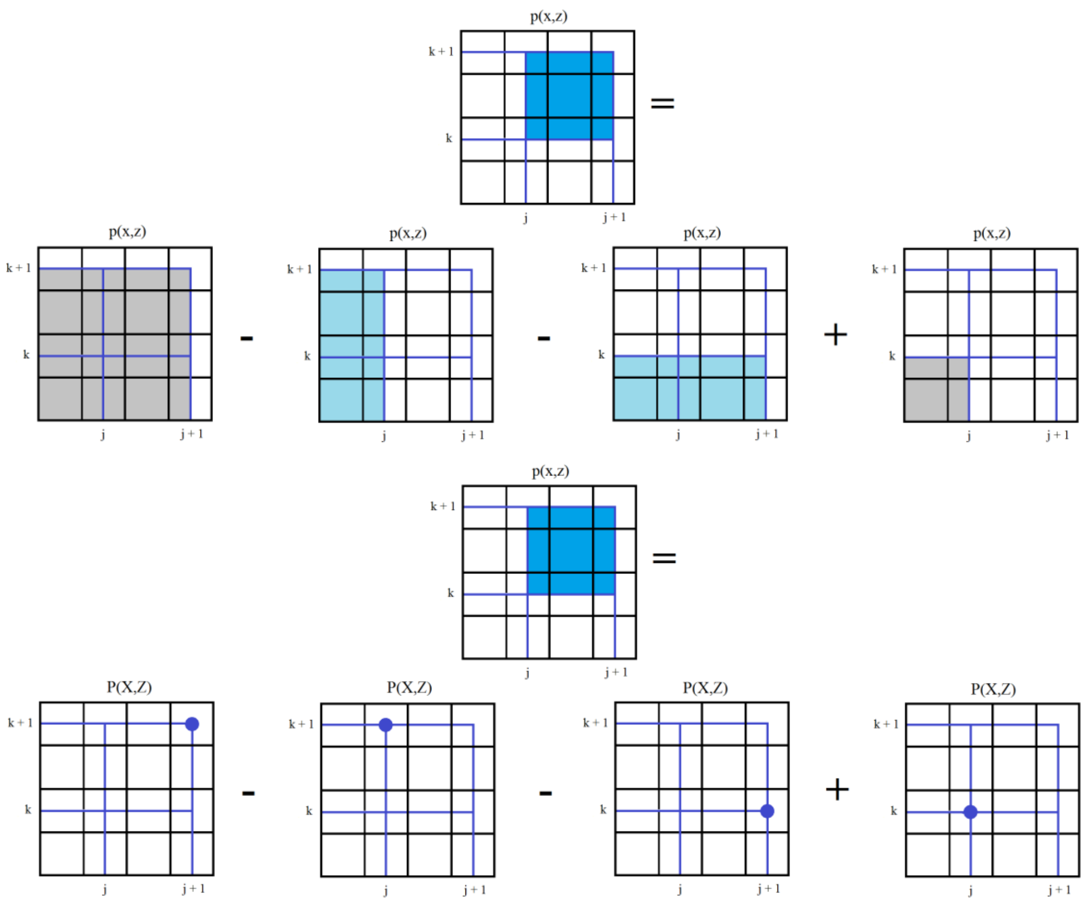


Fig. 9. The sum, *s*, of the intensities of an arbitrary rectangular region defined by boundaries *j*, *j+1*, *k*, and *k+1* can be found by subtracting the intensities of the two light blue regions (defined by *P(j, k+1)* and *P(j+1, k)*) from the intensities of the dark blue region (defined by *P(j+1, k+1)*) and then re-adding the intensities of the gray region (defined by *P(j,k)*) since it was subtracted twice as part of both light blue regions.

However, recall that the mean value of the image was subtracted before integrating the data to create an SAT. Therefore, after calculating *s*, the mean value of the image, μ, needs to be added back in proportion to the area of overlap. Then, this can be inserted into back into Eq. (5). The result is shown in Eq. (11).

$$d_{j,j+1,k,k+1} = \frac{1}{d_{j+1} - d_j}\frac{1}{d_{k+1} - d_k}\left[s + \mu * \ (d_{j+1} - d_j) * (d_{k+1} - d_k)\right] \qquad (11)$$

Unlike the overlap kernel of the traditional distance-driven implementation, Eq. (11) can be calculated with code that does not contain branches. All the parallel threads would perform the same memory operations although with different coordinates. All threads would also carry out the same arithmetic operations to determine the sum of intensities in a voxel-detector overlap. For these reasons, the use of SATs helps distance-driven projection achieve optimal performance on GPUs.

Finally, note that the use of SATs is solely intended to speed up the execution of distance-driven projection and back-projection, not to change the final output of the operations. This preserves the superior image quality produced by the distance-driven model while substantially reducing the execution times.

B. Detailed Method

The use of pre-projection integration applies to both distance-driven projection and back-projection operations. However, for the sake of illustration, consider only the projection operation as described by De Man and Basu. Back-projection involves the exact same steps, but reversed with respect to the sinogram and image. The pre-projection integration method consists of two new steps. First, the data needs to be integrated to convert the image volume into SATs. Second, the projector needs to use the SATs to calculate the sum of the voxels within the rectangular detector footprint. These two steps are described in detail below.

1. SAT Generation

The projection operation is given a 3D image volume as its input. An SAT must be generated for each 2D slice in the volume. The steps to generate an SAT on a GPU are listed in Algorithm 1 and then explained in detail below.

```
Algorithm 1: SAT generation
Input: 2D image slice
Output: SAT
   Place a black border around the image slice
   Find the mean value of the image slice and subtract it from each element
   Run parallel prefix sum
   Run transpose
   Run parallel prefix sum
   Run transpose
```

First, a black (zero-valued) border is placed around each 2D image slice. The black borders are necessary to perform read operations at the edges of the SATs (Hensley, Scheuermann, Coombe, Singh, & Lastra, 2005). There are certainly other ways to handle edge cases, but a black border is a simple approach that is appropriate for the needs of this study.

Next, the mean value of each 2D slice is subtracted from each element within the slice. This compensates for precision loss when the values across a large image in two dimensions are added to create the SATs. The absolute precision of floating-point numbers decreases as the magnitude of the values increase. Since the magnitude of the values increases as they are integrated, this is a source of precision loss that would not be seen if the original image were used to find the sum in a rectangular area. There are a few ways to mitigate this. One method is to subtract the mean value from the image before the SAT is created (Hensley, Scheuermann, Coombe, Singh, & Lastra, 2005), compensating for the subtraction of the mean when the SAT is read during projection. The impact of this method is analyzed in detail in Section IV.A.2.

After the subtraction of the mean, the actual generation of an SAT is completed with successive parallel prefix sum operations and transpose operations (Hensley, Scheuermann, Coombe, Singh, & Lastra, 2005) (Nguyen, 2007). A parallel prefix sum operation, a transpose, another parallel prefix sum operation, and another transpose will generate a 2D SAT. Both the parallel prefix sum and transpose are operations that have previously been optimized on GPUs.

The parallel prefix sum operation, also known as a “scan” operation, sums up the data along each row of an image. It is an “inclusive” operation. Therefore, each element in the output contains its own value plus the value of all preceding elements. The recursive doubling algorithm shown in Fig. 10 can be used to compute the parallel prefix sum operation in O(log n) time (Hensley, Scheuermann, Coombe, Singh, & Lastra, 2005). However, the algorithm proposed by Hensley et al. is not *work efficient*. While the algorithmic complexity is O(log n), the number of addition operations that it requires is O(n log n). More complicated algorithms for the parallel prefix sum use balanced tree approaches (Nguyen, 2007). These have O(log n) complexity and require O(n) addition operations, making them work efficient as well. The pre-projection integration method uses this work efficient implementation.

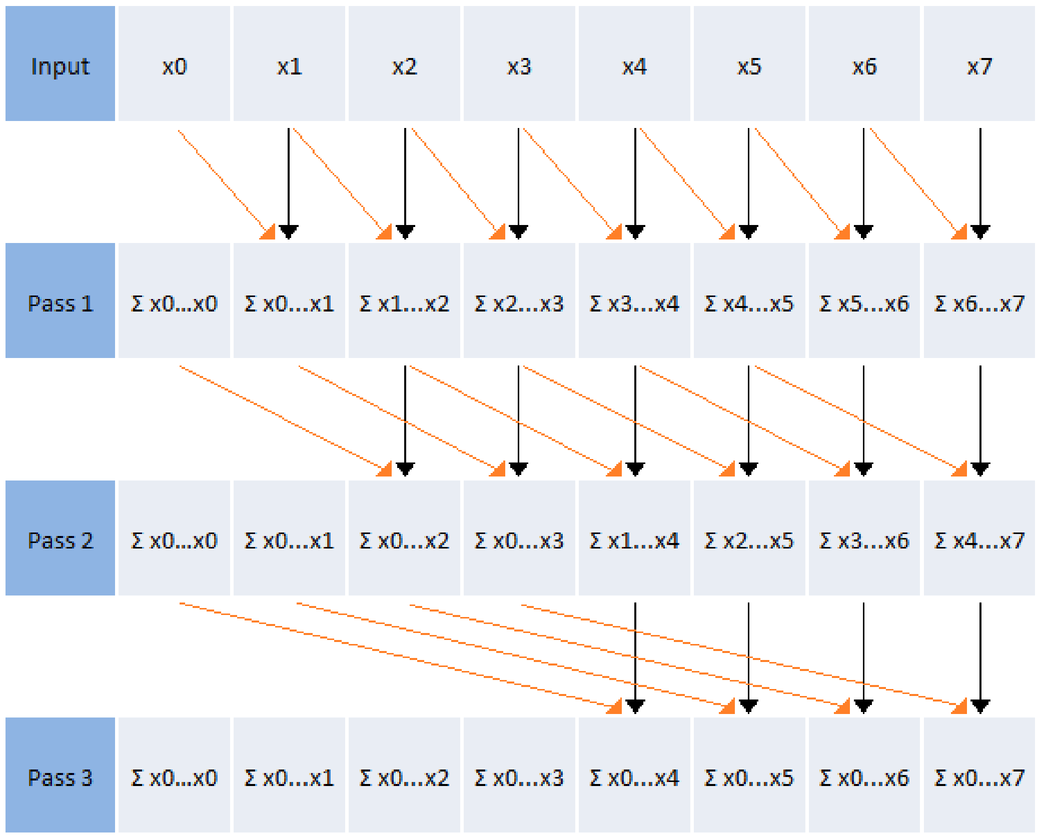


Fig. 10. Recursive doubling algorithm for a row of eight elements. The output can be computed in O(log n) time, or three passes in this case.

A transpose is then performed after the parallel prefix sum operation. For performance reasons, the transpose operation used in pre-projection integration makes strategic use of the local memory on a GPU. Threads work on small tiles and coordinate the use of local memory in order to avoid reading all values from global memory. Fig. 11 shows the first two steps of the SAT generation: parallel prefix sum and transpose.

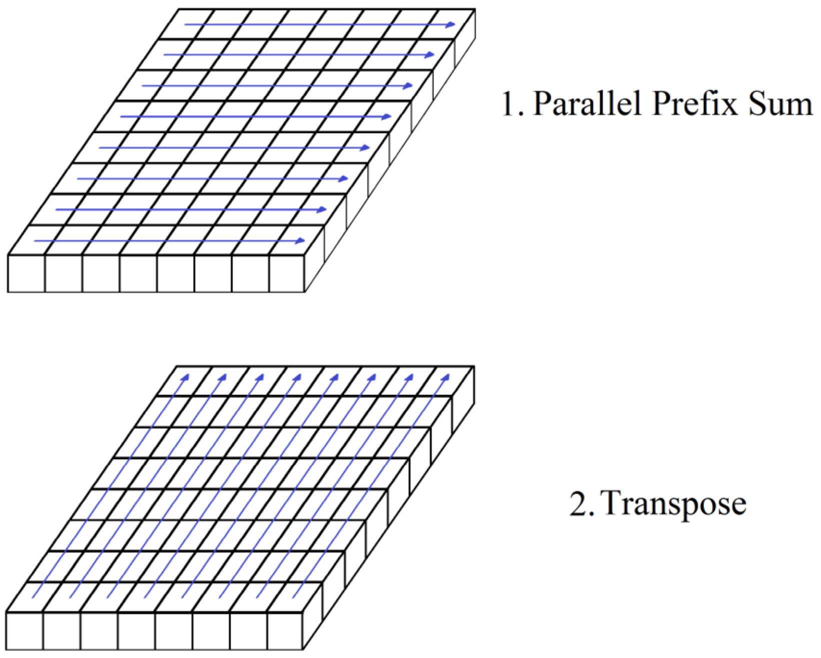


Fig. 11. The first two steps of SAT generation. A parallel prefix sum operation sums the data along each row in an image slice. After that, a transpose is performed to turn what were originally the columns of the slice into the rows.

After the first transpose operation, the second parallel prefix sum operation sums the data along what were originally the columns of the image. Another transpose operation brings the image back to its original orientation. After the second transpose, the SAT generation is complete: each element contains the sum of all elements preceding it in both the vertical and horizontal dimensions. Note that this means that the sum of all values is located in the upper right corner of the image slice.

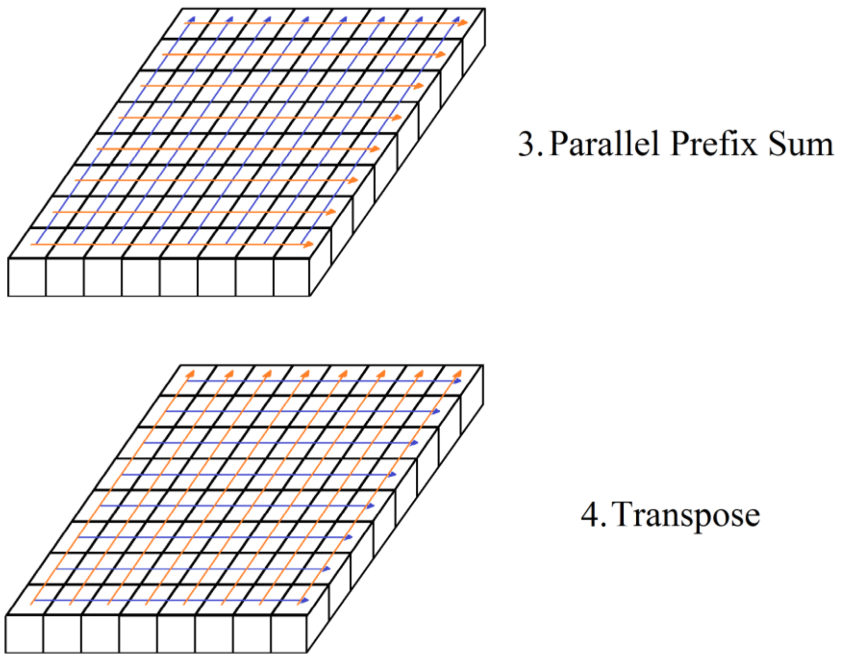


Fig. 12. The final two steps of SAT generation. After the transpose in the second step, another parallel prefix sum operation sums the data along what were originally the columns. Finally, a transpose is performed to bring the data back to the original orientation.

This entire process must be done for each 2D image slice in the 3D image volume. In a simple implementation, each slice can be done sequentially. It is possible to parallelize the computation along the slices of the 3D image volume as well. The significance of the performance benefits would depend on the size of the image slices and the number of cores in the GPU. For many scenarios, there is enough work in a 2D image slice to fully utilize the GPU.

Because SAT generation is not a step that normally takes place during projection or back-projection, it is crucial that the time it takes is minimal compared to the time the actual projection or back-projection takes. Fortunately, SAT creation on GPUs has already been optimized by others (Hensley, Scheuermann, Coombe, Singh, & Lastra,

2005) (Nguyen, 2007). The performance of the SAT generation is explored in Section IV.A.1.

2. Using an SAT for Overlap Calculations

After creating the SATs, the original distance-driven projection is completed. However, instead of calculating the sum of an image within a detector footprint with an overlap kernel, the SAT is used. The steps to use a GPU to perform this calculation are listed in Algorithm 2 and then explained in detail below.

```
Algorithm 2: Calculation of incremental projection updates using an SAT
Input: SAT and coordinates of rectangular detector-voxel overlap
Output: Incremental update for projection
   Read the SAT values at the four corners of the overlap
   Use Texture Units on the GPU to do bilinear interpolation
   Calculate the area using Eq. (10)
   Add the mean value proportional to the area
   Share SAT reads for adjacent detector footprints
```

In 3D distance-driven projection, the voxels and detector cells are mapped onto a common plane to approximate the sum in the area of overlap. Suppose we need the sum of the voxels over the area of overlap defined by mapping the detector onto an image plane as shown in Fig. 13.

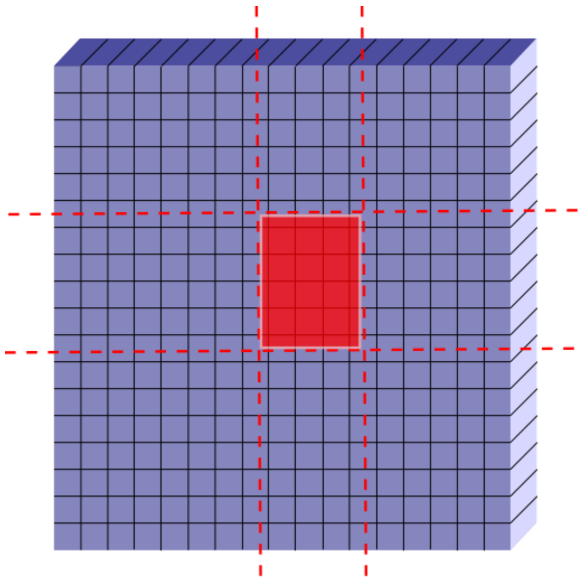


Fig. 13. Detector boundaries (dashed red lines) mapped onto a 2D image slice.

The traditional distance-driven projector would use the overlap kernel to produce a result. However, if an SAT is used, it can generate the value of the 2D integral with four image reads as shown in Fig. 14.

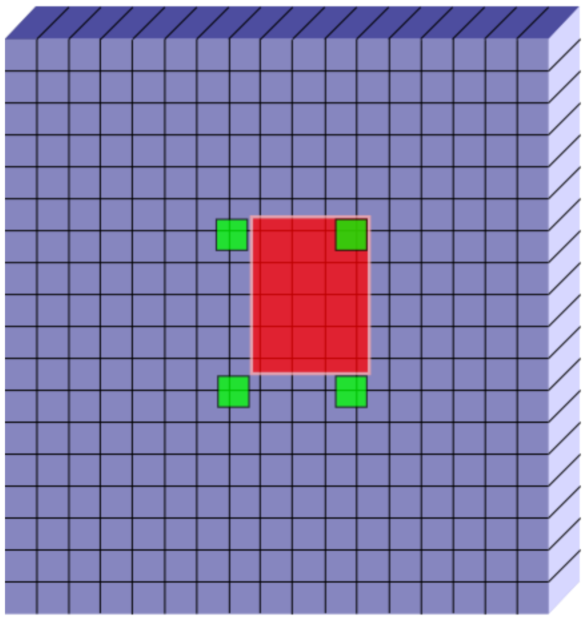


Fig. 14. Four SAT read operations (green) are required to get the sum of the voxel intensities in the area of overlap (red).

As Fig. 14 shows, the detector boundaries usually do not line up perfectly with the image grid. Therefore, a bilinear interpolation of four SAT elements is necessary to get the value we need for each corner of the SAT. In order to efficiently do this, the interpolation hardware in GPU Texture Units is used. The SATs are stored as textures and *floating point coordinates* are specified for each read operation. The GPU then uses a Texture Unit to do bilinear interpolation of the four closest voxels. Because the work is done on dedicated hardware, this is much quicker than doing the interpolation manually with software instructions.

While GPU Texture Units are very efficient, there is a loss of precision when they are used to perform linear interpolation. Modern GPUs can represent textures with 32-bit values compliant with IEEE 754 (IEEE Standards Committee, 2008). However, the coefficients used to interpolate between texture values are generally not IEEE 754 compliant. For example, on modern Nvidia GPUs, the interpolation coefficients are 9-bit fixed point values with one bit as the sign bit and eight bits of fractional value (Wilt, 2013). If the texture is stored in 32-bit floating point format, precision can be lost by interpolating the values with 9-bit fixed point coefficients. Nevertheless, using the Texture Units for the interpolation is preferable due to their speed. We explore the effects of the precision loss caused by the Texture Units in Sections IV.B.2 and IV.C.2.

With the GPU performing bilinear interpolations, the result is easily calculated. The outputs from the four SAT read operations are named LL (lower left), LR (lower right), UL (upper left), and UR (upper right). Then, if the mean had not been subtracted before creating the SATs, the sum of the intensities within the rectangular detector boundary, *s*, could be calculated as shown in Eq. (12).

$$s = UR - UL - LR + LL \tag{12}$$

However, Eq. (12) is modified slightly to correct for the subtraction of the mean, μ, that was done before the SAT was created. To do this, it is necessary to know the length of the overlap in both the X direction (*x_l*) and the length of the overlap in the Z direction (*z_l*). The sum of the intensities, *s*, is then calculated as shown in Eq. (13).

$$s = (UR - UL - LR + LL) + (x\_l * z\_l * \mu) \tag{13}$$

Finally, the number of global memory accesses can be further reduced by sharing SAT reads for adjacent detectors. When detector boundaries are mapped to an image plane, adjacent detectors usually have adjacent detector boundaries. Therefore, it is possible to do fewer than four reads per detector by sharing reads between adjacent detectors. Fig. 15 shows how two detector boundaries share SAT reads. The actual “sharing” of the SAT values can be achieved by either using GPU local memory or by having a thread calculate the projection of multiple sinogram elements and storing the values in private memory.

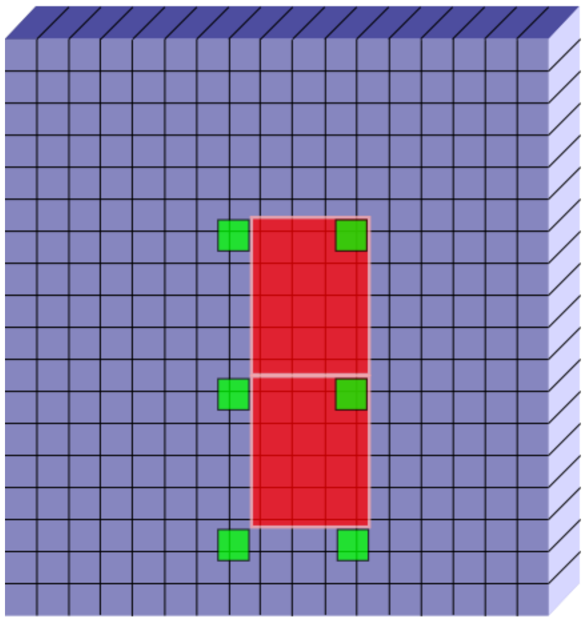


Fig. 15. SAT reads (green) for adjacent detector elements (red rectangles). The two middle SAT values are required for the calculation for both detector elements, so they can be shared.

IV. RESULTS

There are three main operations discussed in Section III: 1) SAT generation, 2) projection, and 3) back-projection. An objective of this study was to analyze these operations both as independent processes and as part of an image reconstruction algorithm. Therefore, the results in this section are presented as follows. First, Sections IV.A, IV.B, and IV.C contain the results of the individuals operations of SAT generation, projection and back-projection, respectively. The analysis for each operation is done for performance, which means execution time, and image quality. Then, in Section IV.D, all operations are analyzed together as part of an iterative image reconstruction algorithm.

In order to generate the results in this section, the geometry of a cone beam X-ray CT system was simulated. The system has an arc detector with N_s = 888 detector channels and N_t = 32 detector rows. There are N_β = 984 views over 360˚. The size of each detector cell is Δ_s = 1.024 mm by Δ_t = 1.099 mm. The source to detector distance is D_{sd} = 946.75 mm and the source to rotation center distance is D_{s0} = 538.52 mm. A quarter detector offset in the s direction was also used. The 3D image volume has dimensions of N_X , N_Y, and N_Z, the sizes of which vary according to the experiment. The experiments were carried out on a 64-bit Linux workstation with two 4-core Intel Xeon CPUs and an Nvidia Tesla K20 workstation graphics card.

A. SAT Generation

SAT generation was analyzed for both performance and precision loss. The performance analysis characterizes how quickly a stack of 2D SATs can be generated from both a 3D image volume and a 3D sinogram. Then, the precision loss analysis characterizes the factors that contribute to precision loss in SATs.

1. SAT Generation Performance

A simple SAT generator was implemented with CUDA. Generating an SAT on a GPU is a problem that has previously been solved and optimized (Nguyen, 2007). It was possible to create a simple SAT generation program by slightly modifying Nvidia's sample parallel prefix sum and transpose kernels. It was also necessary to create a kernel ("Subtract mean") that found the mean value of each slice and subtracted it from all elements in the slice.

For projection, the SAT generation step consists of generating a 2D SAT for each slice in the volume. A 48x512x512 volume is used in this experiment, so an SAT is generated in each 48x512 slice of the 512-slice volume. As previously mentioned, it is convenient to put a black (zero-valued) border around each 2D slice so that texture reads outside the slice return zero. This enables reconstruction of image pixels at the edges of the slice. The size of the volume thus becomes 50x514x512. Additionally, the parallel prefix sum kernel requires the dimensions to be powers of two, so the 50x514x512 volume is padded to be 64x1024x512. The generation of the SATs for the entire 64x1024x512 volume takes about 12 milliseconds, excluding memory transfers to and from the GPU device. However, all other operations of the reconstruction algorithm would also be performed on the GPU, and hence the memory transfers would have to occur anyway. Table 1 below shows the execution times for the individual kernels in the SAT generation process.

Table 1. SAT kernel execution times for a 64x1024x512 dataset representing the image volume.

Operation	**Execution Time (milliseconds)**
Subtract mean	2.57
Parallel prefix sum 1	1.85
Transpose 1	2.82
Parallel prefix sum 2	1.76
Transpose 2	2.83
Total	**11.8**

For back-projection, the SAT generation step consists of generating an SAT for each 32x888 slice in the 984-slice sinogram. Again, a black border around each 2D slice is used for convenience, increasing the size of the sinogram to 34x890x984. Additionally, the SAT needs to be padded to get dimensions that are powers of two, so the sinogram dimensions increase to 64x1024x984. The generation of the SATs for the entire 64x1024x984 volume takes about 20 milliseconds, excluding memory transfers to and from the GPU device. Again, since all other operations of the reconstruction algorithm would also be performed on the GPU, the memory transfers would have to occur anyway. Table 2 below shows the execution times for the individual kernels in the SAT generation process.

Table 2. SAT kernel execution times for a 64x1024x984 dataset representing the sinogram.

Operation	**Execution Time (milliseconds)**
Subtract mean	2.55
Parallel prefix sum 1	3.56
Transpose 1	5.38
Parallel prefix sum 2	3.38
Transpose 2	5.51
Total	**20.4**

2. SAT Precision Loss

In addition to the calculation of performance results, an analysis of precision loss from SAT generation was performed. Note that this experiment does not involve any projection steps. The inputs were square images of the Shepp-Logan phantom (Shepp & Logan, 1974). Gaussian noise was added to the images to make them more realistic. A variety of image matrix sizes were used. To quantify how much precision is lost, an SAT is generated from the image. Then, the original image is simply recreated using the SAT. The SAT error is the difference between the recreated image and the original. It is important to note that recreating the original image does not require the use of any interpolation.

The loss of precision that occurs when generating SATs is attributed to the fact that the absolute precision of floating-point numbers is lost as the magnitude of the values increase. Specifically, floating-point numbers of greater magnitude have less absolute precision than floating-point numbers of smaller magnitude. See Fig. 16.

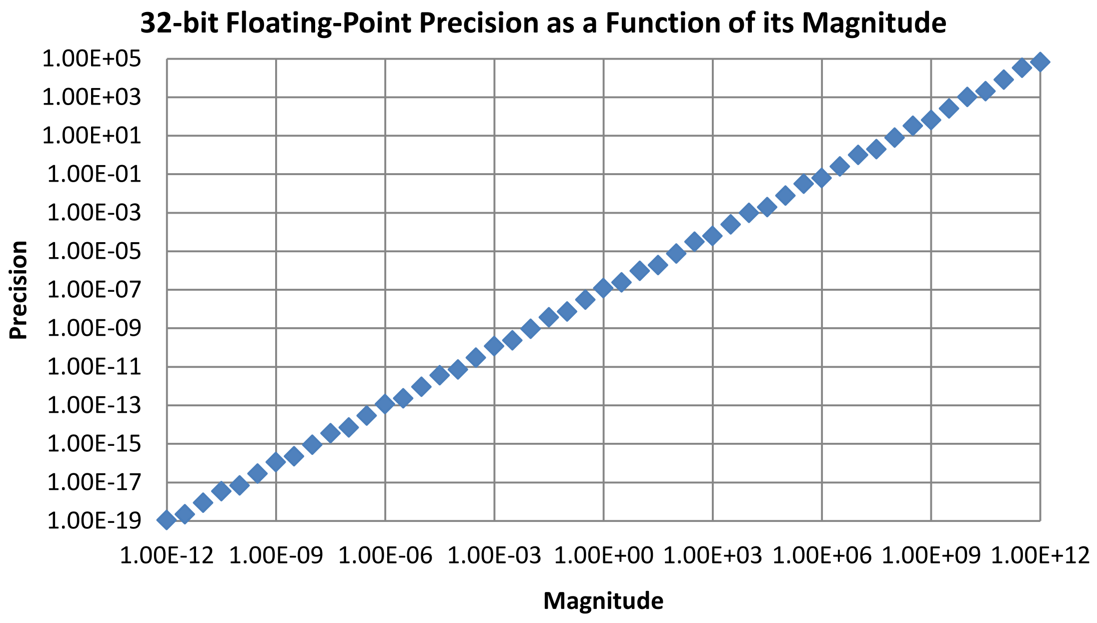


Fig. 16. 32-bit floating-point precision as a function of its magnitude. As the magnitude of a floating-point value increases, absolute precision is lost.

When the values of an image are accumulated to generate an SAT, precision is lost because the values being manipulated have greater magnitude than they did before they were accumulated. Therefore, the relationship between the maximum value in the SAT and the maximum SAT error is examined. Fig. 17 shows two plots. The first shows the maximum SAT error as a function of the maximum value in the SAT. The second is a subset of the 32-bit floating-point precision from Fig. 16. The correlation of the two plots demonstrates that the maximum SAT error is on the order of the precision of the maximum value in the SAT.

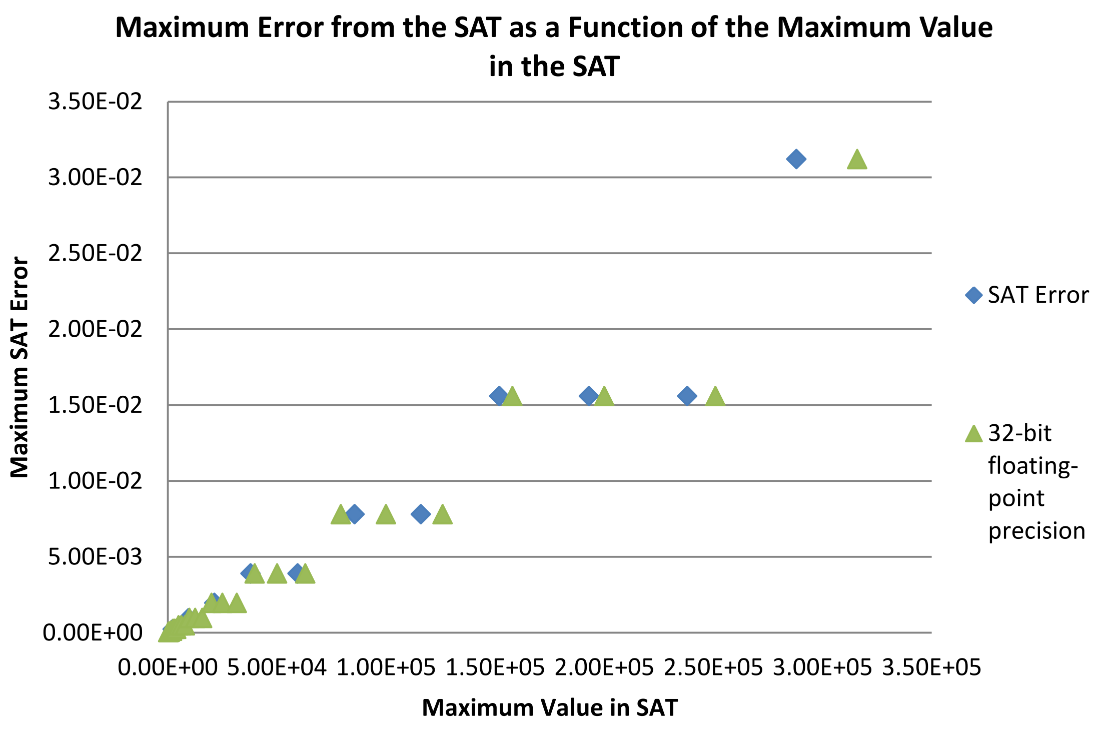


Fig. 17. Maximum Error from SAT as a Function of the Maximum Value in the SAT. The maximum error from an SAT is on the order of the precision of the maximum value in the SAT.

As stated previously, in order to mitigate the error caused by using SATs, the mean value of the image can be subtracted before creating the SAT. This has two effects. First, it reduces the maximum value from the SAT, which therefore reduces the maximum error as shown in Fig. 17. Second, it decreases the average error from the SAT, which can be more important than the maximum error. The average error from the SAT is data dependent and therefore cannot be as easily characterized as the maximum error in the SAT. Still, Fig. 18 shows how the average error is proportional to the area of the SAT. It also shows how the slope of that relationship changes when the mean is subtracted.

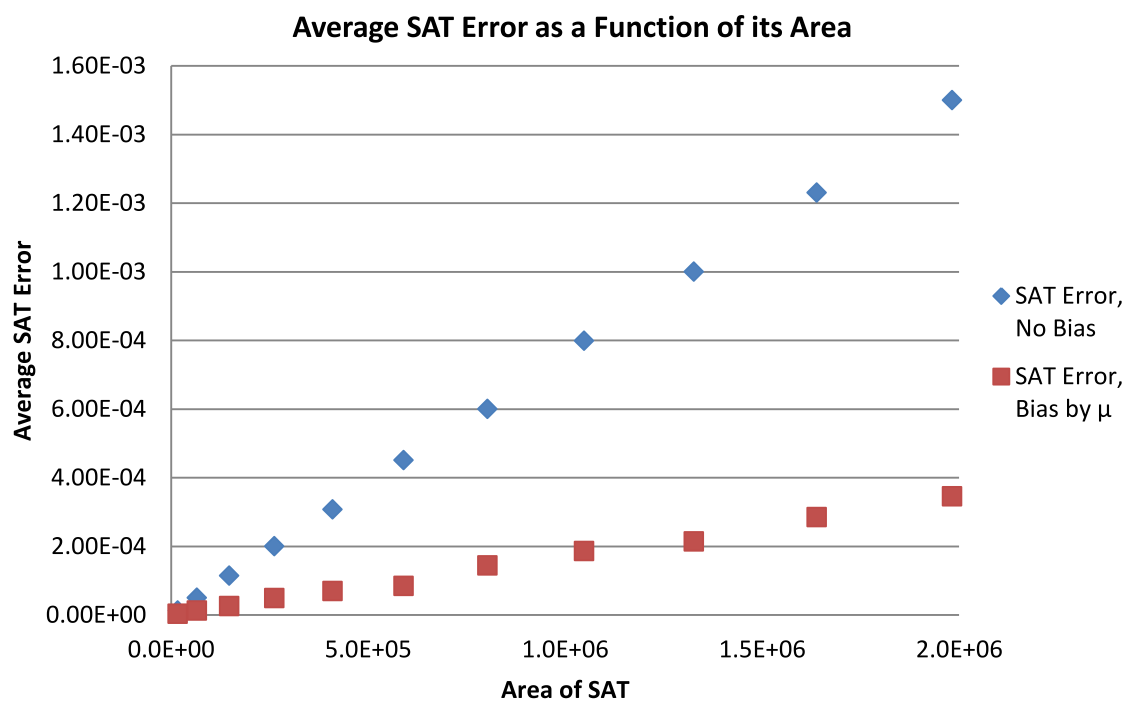


Fig. 18. Average SAT Error as a Function of its Area. As the area of an SAT increases, its average error also increases. However, the subtraction of the mean before creating the SAT has a substantial impact on the average SAT error.

Subtracting the mean before creating the SAT also has another useful effect. It prevents the error from accumulating in the corner of the image. Fig. 19 shows the original image, a 512x512 Shepp-Logan phantom with a mean of 0.1233, on the left and a difference image on the right that comes from using an SAT to recreate the original image. The histogram of the difference image was stretched to allow visualization of the error. The error increases monotonically from the origin (top left) to the end of the table (bottom right). The average error is $2.5 * 10^{-4}$.

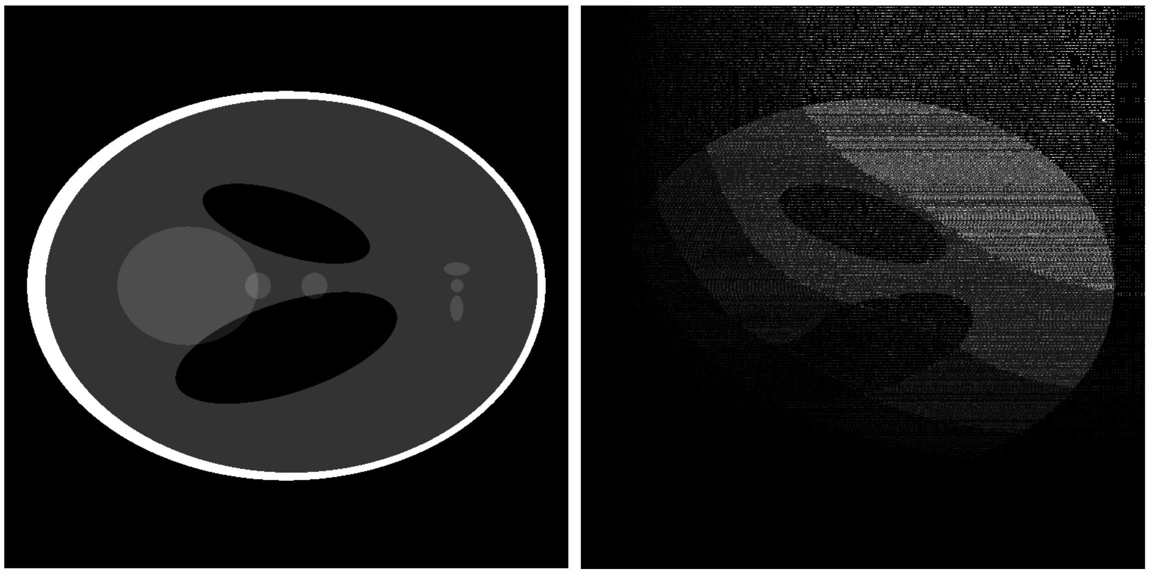


Fig. 19. Shepp-Logan phantom (left) and difference image (right) after recreating it from an SAT. The histogram of the difference image was stretched to allow visualization of the error.

Fig. 20 shows how subtracting the mean value prevents the error from accumulating towards the corner of the image. Again, note that the histogram of the difference image has been stretched to allow visualization of the error. The error is distributed over the entire image, not just in one corner. This method also decreases the mean of the error by a factor of 2.5 (from $2.5 * 10^{-4}$ to $1.0 * 10^{-4}$).

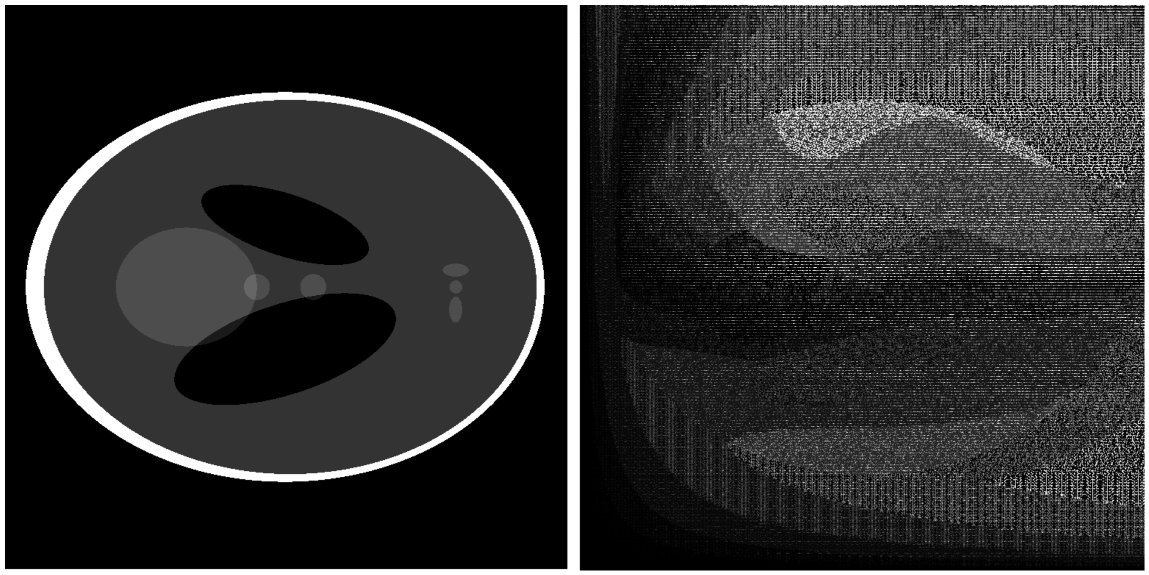


Fig. 20. Shepp-Logan phantom (left) and difference image (right) after recreating it from an SAT with the mean subtraction method. The histogram of the difference image was stretched to allow visualization of the error.

B. Projection

Within the context of projection, the pre-projection integration method was analyzed in terms of performance and image quality. The execution times for projection were measured for three versions of the projector: a single-threaded CPU version, a GPU-optimized version, and the pre-projection integration version. To verify the image quality of the pre-projection integration projector, its output was compared with the output of the GPU-optimized projector.

1. Projection Performance

A single-threaded CPU version of the distance driven projector served as the baseline. The GPU-optimized version of the projector was written in CUDA and was highly optimized for the Nvidia Tesla K20. Private memory (in the form of registers) was used for the projection sums in order to limit global memory transactions. However, using a large number of registers per thread meant that the threadblock size needed to be smaller in order to maximize the kernel occupancy. The image data was stored as a texture in order to take advantage of the texture caching on Nvidia GPUs. Finally, in order to get a high L1 cache hit rate on texture reads, the kernel was configured to prefer texture memory over local memory. This means that the GPU decreased the amount of local memory available in order to increase the amount of L1 cache available. The threads in the threadblock were also organized in order to maximize the spatial locality of the texture reads, further improving the L1 cache hit rate.

After optimizing it for the K20, the projector was enhanced with the pre-projection integration method. Table 3 shows the execution time for various implementations of the distance-driven projector. Fig. 21 shows the projection execution times from Table 3. Finally, Fig. 22 shows how the speedup from the pre-projection integration increases from 2.5x to 4.4x as the image size increases from 128x128x12 to 1152x1152x108.

Table 3. Distance-driven projection execution times. The duration of the SAT generation for the pre-projection integration method is omitted because it is negligible.

Image Size	Number of Voxels	Projection Execution Time (seconds)		
		Single-Threaded CPU	GPU-Optimized	Pre-Projection Integration
128x128x12	1.97E+05	15.4	0.28	0.11
256x256x24	1.57E+06	43.5	0.66	0.24
384x384x36	5.31E+06	89.4	1.17	0.41
512x512x48	1.26E+07	162.7	1.98	0.61
640x640x60	2.46E+07	242.9	2.99	0.85
768x768x72	4.25E+07	360.4	4.24	1.14
896x896x84	6.74E+07	513.8	5.81	1.47
1024x1024x96	1.01E+08	742.8	7.79	1.85
1152x1152x108	1.43E+08	935.9	10.16	2.30

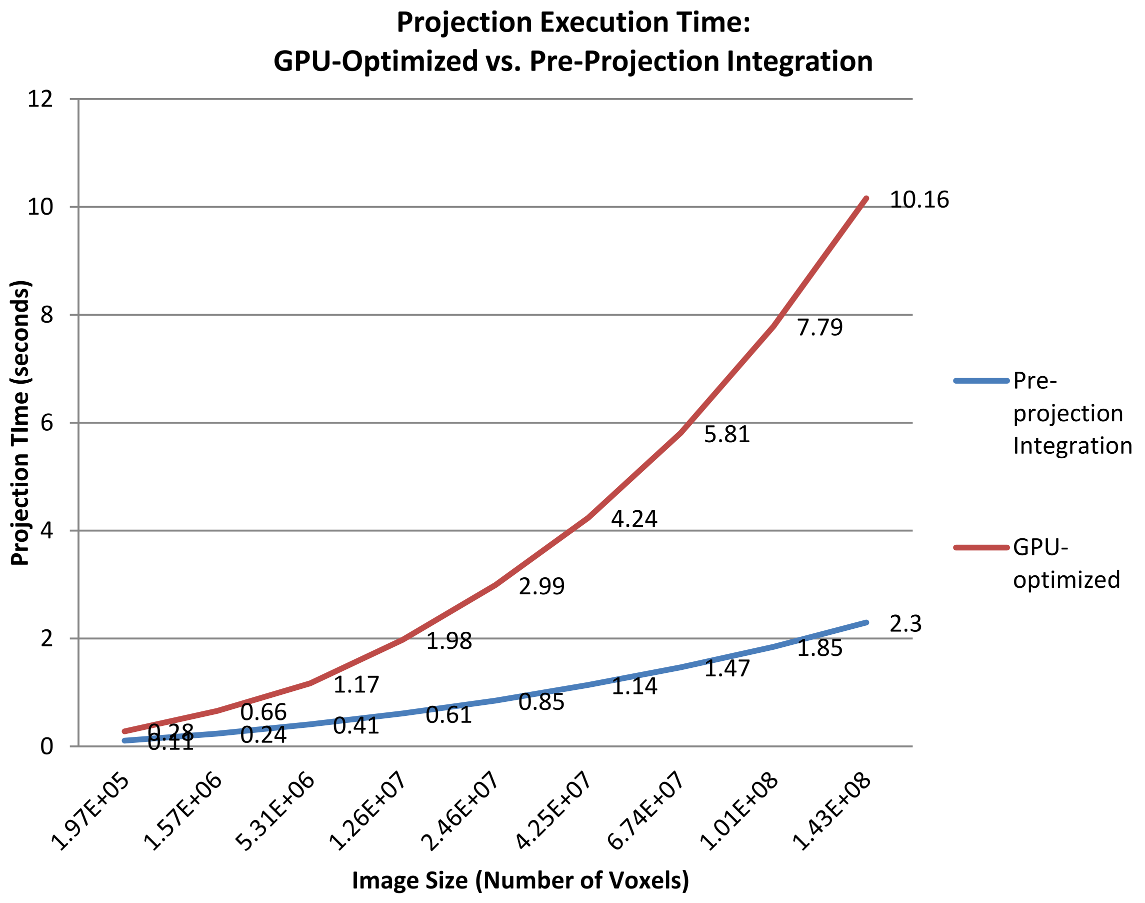


Fig. 21. A comparison of the projection execution times when using the GPU-optimized method and the pre-projection integration method.

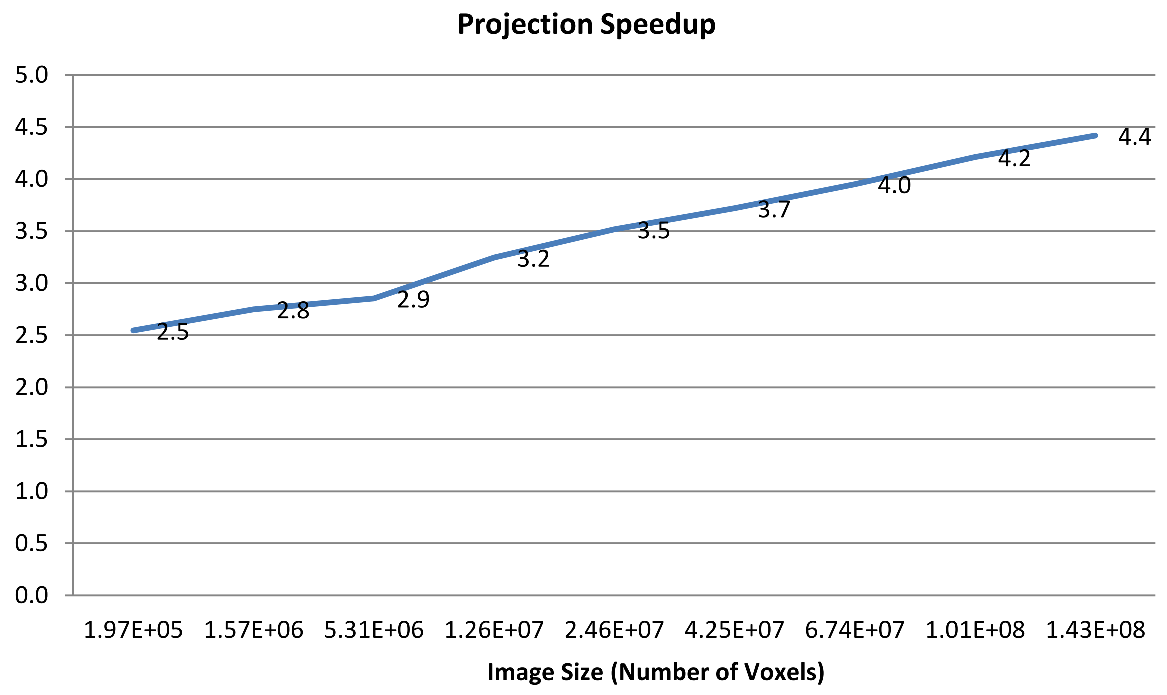


Fig. 22. Projection speedup from pre-projection integration. For smaller image sizes, the pre-projection integration method is 2.5x faster than the GPU-optimized method. For larger image sizes, the pre-projection integration method is 4.4x faster.

2. Projection Image Quality

The output of the pre-projection integration projector was compared with the GPU-optimized projector. The input image was a 512x512 version of the Shepp-Logan phantom, replicated 48 times to make 48 image slices. Fig. 23 shows a row of the sinogram from the pre-projection integration projector and the same row of the sinogram from the GPU-optimized projector. There are no visible differences. The image histograms are shown in Fig. 24. The differences in the image histograms are negligible. For example, the histogram of the sinogram produced by the pre-projection integration projector has a maximum value that is 0.004 larger than that of the sinogram produced by the GPU-optimized projector. Additionally, the mean value of the sinogram slice is exactly the same in the pre-projection integration and GPU-optimized output.

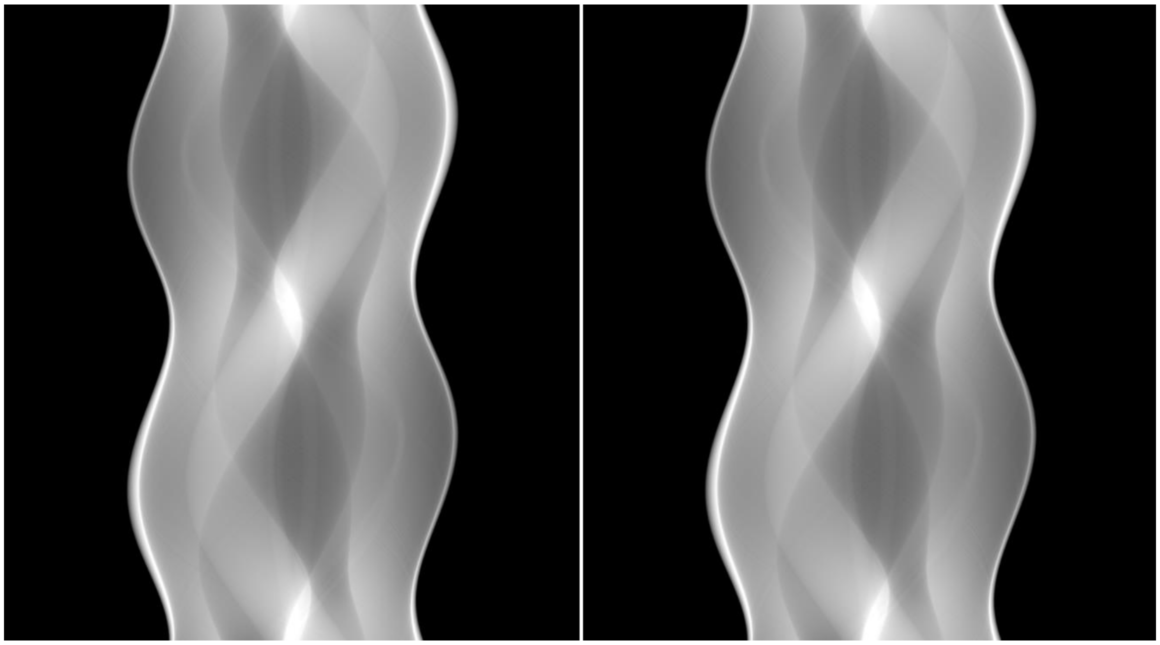
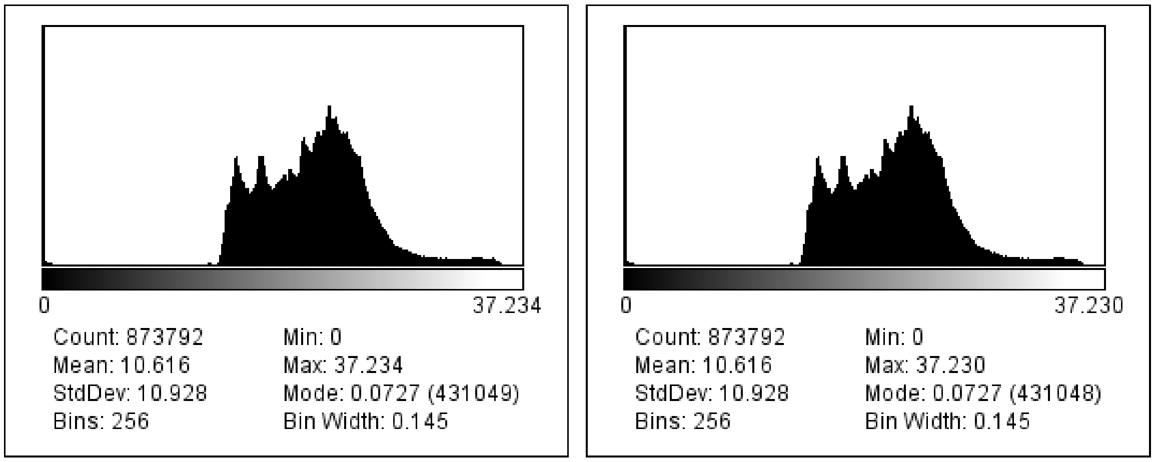


Fig. 23. A row of the sinogram from the pre-projection integration projector (left) and the GPU-optimized projector (right).



```
Count: 873792       Min: 0
Mean: 10.616        Max: 37.234
StdDev: 10.928      Mode: 0.0727 (431049)
Bins: 256           Bin Width: 0.145
```

```
Count: 873792       Min: 0
Mean: 10.616        Max: 37.230
StdDev: 10.928      Mode: 0.0727 (431048)
Bins: 256           Bin Width: 0.145
```

Fig. 24. Histograms for the sinogram slices shown in Fig. 23. Pre-projection integration is on the left and GPU-optimized is on the right.

In order to get a more detailed analysis of the image quality of the pre-projection integration projector, a difference image is required. Fig. 25 shows the pre-projection integration sinogram row subtracted from the GPU-optimized sinogram row. The histogram for that difference image is shown in Fig. 26. The difference ranges from about -0.06 to 0.05. These values are orders of magnitude smaller than the mean value of the sinogram slice, 10.616.

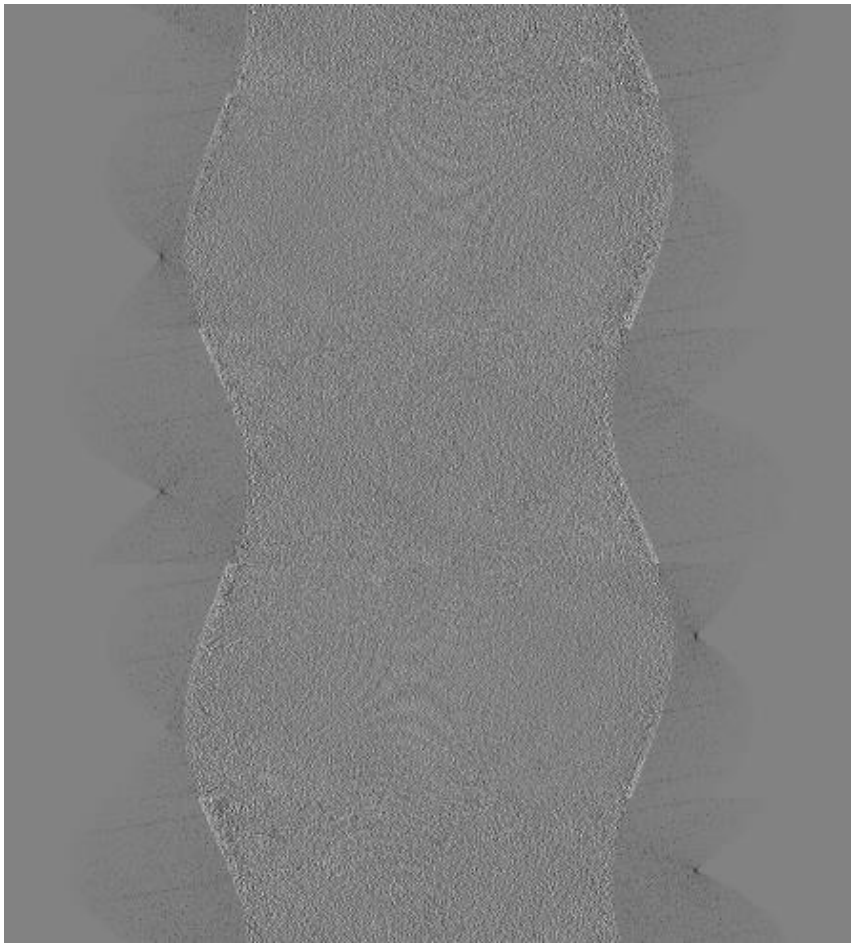


Fig. 25. A row of the pre-projection integration sinogram subtracted from the same row of the GPU-optimized sinogram. The histogram has been stretched to enhance the visualization of the error.

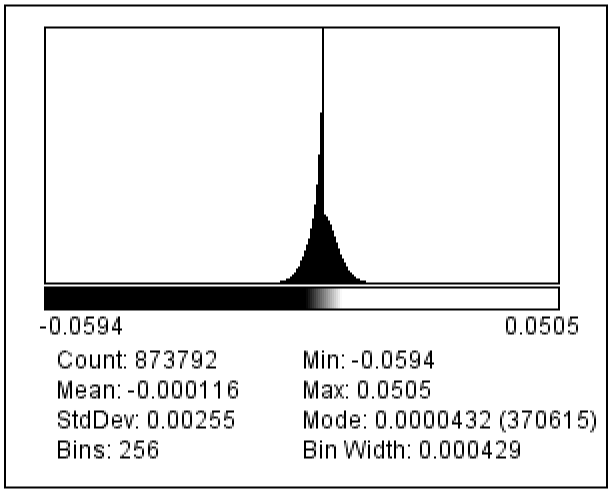


Fig. 26. Histogram of the difference sinogram in Fig. 25.

A close visual inspection of the difference between the sinogram row generated by the pre-projection integration projector and the sinogram row generated by the GPU-optimized projector in Fig. 25 shows some aliasing artifacts. While the minimum and maximum differences are very small compared to the mean values in the sinogram, it is still important to understand the root cause of any artifact. The aliasing artifacts are a result of using the limited precision hardware interpolation provided by the GPU's Texture Units. It is likely that this is due to the quantization of the interpolation coefficients which are represented in the Texture Units as 9-bit fixed point numbers. A version of the pre-projection integration projector was written that manually performs the bilinear interpolation of the SAT reads with software. This projector produces output that does not contain the aliasing artifacts. Fig. 27 contains the same difference image from Fig. 25, but software was used to do bilinear interpolation instead of hardware. Fig. 28 shows a histogram of the image in Fig. 27. The histogram shows that, in addition to eliminating the aliasing artifacts, the software interpolation also decreases the error by an order of magnitude.

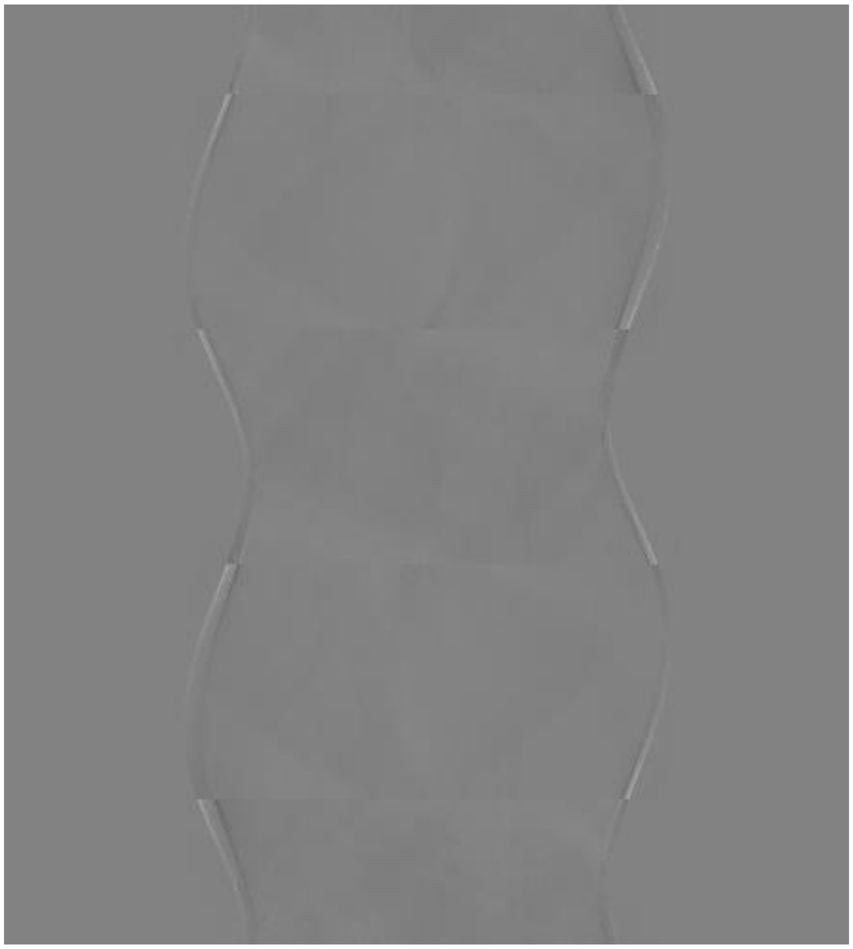
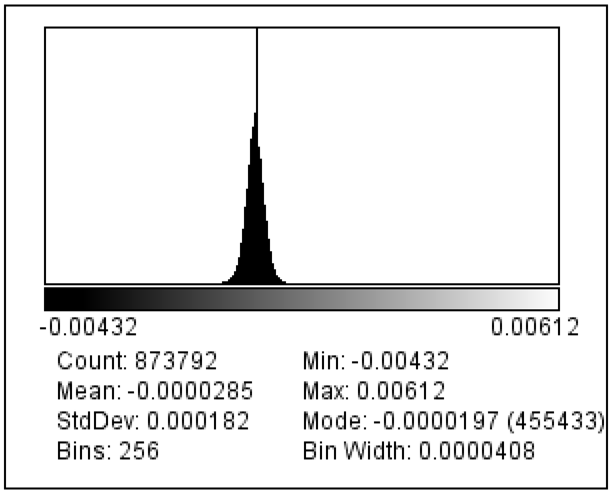


Fig. 27. A row of the pre-projection integration sinogram subtracted from the same row of the GPU-optimized sinogram. In this figure, the pre-projection integration version uses a software-based interpolation method instead of the hardware-based Texture Unit method. The histogram has been stretched to enhance the visualization of the error.



```
Count: 873792             Min: -0.00432
Mean: -0.0000285          Max: 0.00612
StdDev: 0.000182          Mode: -0.0000197 (455433)
Bins: 256                 Bin Width: 0.0000408
```

Fig. 28. Histogram of the difference image in Fig. 27

C. Back-Projection

Similar to projection, the pre-projection integration back-projector was analyzed in terms of both performance and image quality. The execution times for back-projection were measured for three versions of the back-projector: a single-threaded CPU version, a GPU-optimized version, and the pre-projection integration version. To verify the image quality of the pre-projection integration back-projector, its output was compared with the output of the GPU-optimized back-projector.

1. Back-Projection Performance

The performance analysis of the back-projector was carried out in a way similar to that of the projector. A single-threaded CPU version of the distance driven back-projector served as the baseline. Like the GPU-optimized projector, the GPU-optimized back-projector was written in CUDA and was highly optimized for the Nvidia Tesla K20. The threadblock size was tuned to maximize the kernel occupancy. The sinogram data was stored as a texture in order to take advantage of the texture caching. As was the case with the projector, the kernel was configured to prefer texture memory over local memory in order to get a high L1 cache hit rate on texture reads. Additionally, the threads in the threadblock were organized in order to maximize the spatial locality of the texture reads, further improving the L1 cache hit rate.

After optimizing for the K20, the back-projector was then enhanced with the pre-projection integration method. Table 4 shows the execution time for various implementations of the distance-driven back-projector. Fig. 29 shows the execution times from Table 4. Fig. 30 shows how the speedup from the pre-projection integration back-projector *decreases* from 5.3x to 1.2x as the image size increases from 128x128x12 to 1152x1152x108. While the speedup decreases as the image size increases, the pre-projection integration method still provides a speedup of at least 20%.

Table 4. Distance-driven back-projection execution times. The duration of the SAT generation for the Pre-projection Integration method is omitted because it is negligible.

Image Size	**Number of Voxels**	**Back-Projection Execution Time (seconds)**		
		Single-Threaded CPU	**GPU-Optimized**	**Pre-Projection Integration**
128x128x12	1.97E+05	15.0	0.21	0.04
256x256x24	1.57E+06	43.3	0.47	0.19
384x384x36	5.31E+06	89.5	1.23	0.67
512x512x48	1.26E+07	162.5	2.30	1.49
640x640x60	2.46E+07	242.8	4.22	3.34
768x768x72	4.25E+07	360.0	6.59	5.12
896x896x84	6.74E+07	516.3	10.04	7.97
1024x1024x96	1.01E+08	739.0	14.52	11.75
1152x1152x108	1.43E+08	936.6	20.53	17.05

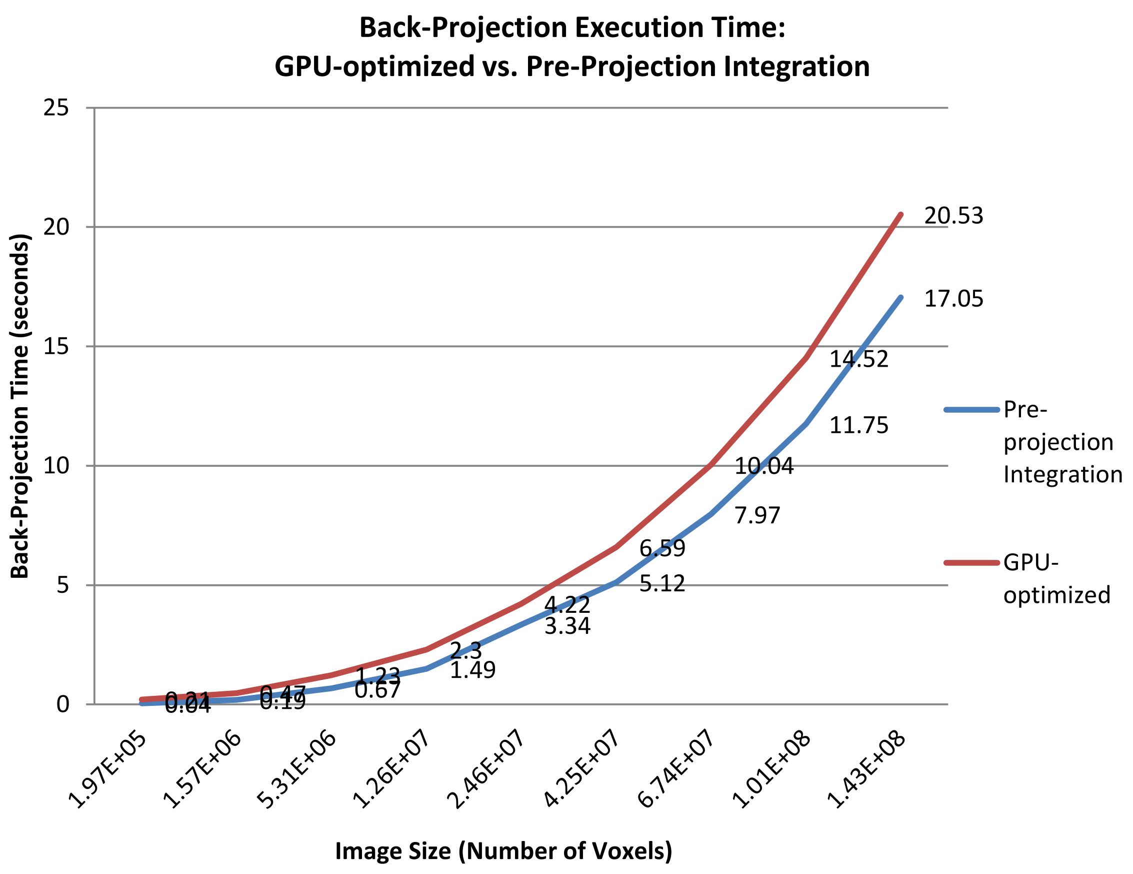


Fig. 29. A comparison of the back-projection execution times when using the GPU-optimized method and the pre-projection integration method.

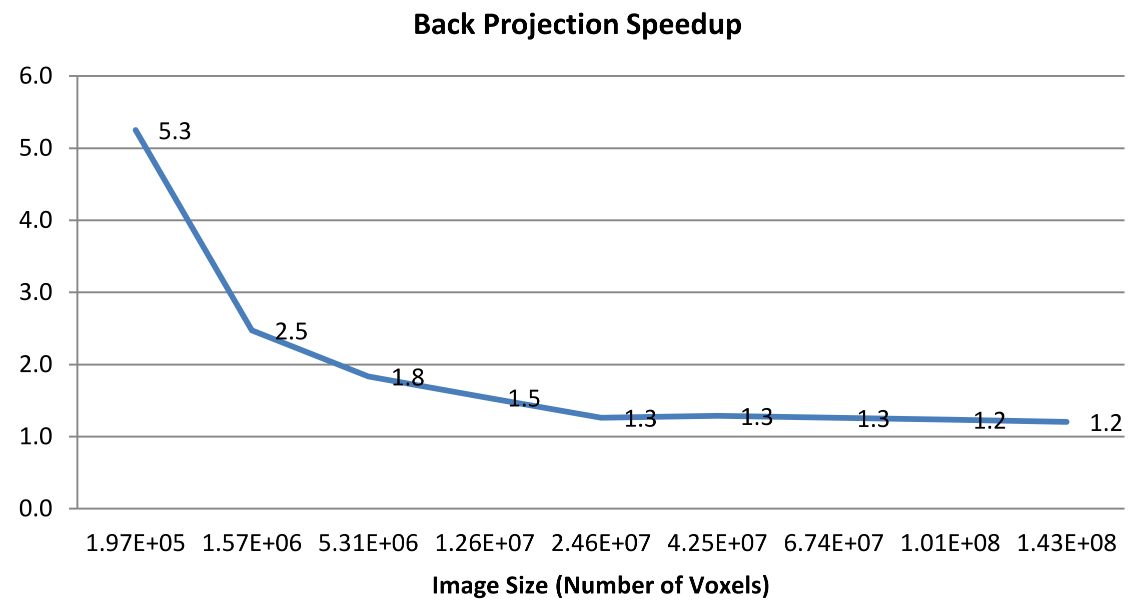


Fig. 30. Back-projection speedup from pre-projection integration. For smaller image sizes, the pre-projection integration method is 5.3x faster than the GPU-optimized method. For larger image sizes, the pre-projection integration method is 1.2x faster.

2. Back-Projection Image Quality

The output of the pre-projection integration back-projector was compared with the GPU-optimized back-projector. The input was a 32x888x984 projection of the Shepp-Logan phantom and the output was a 512x512x48 image volume. Fig. 31 shows a slice of the image from the pre-projection integration back-projector and the same slice of the image from the GPU-optimized back-projector. There are no visible differences. The image histograms are shown in Fig. 32. The histograms show almost no differences in the distribution or range of the data. For example, the difference of the mean value for each image slice is 0.134. This is five orders of magnitude smaller than the mean value of the GPU-optimized image slice, $1.1957 * 10^4$. The differences in the minimums and maximums are also similarly negligible.

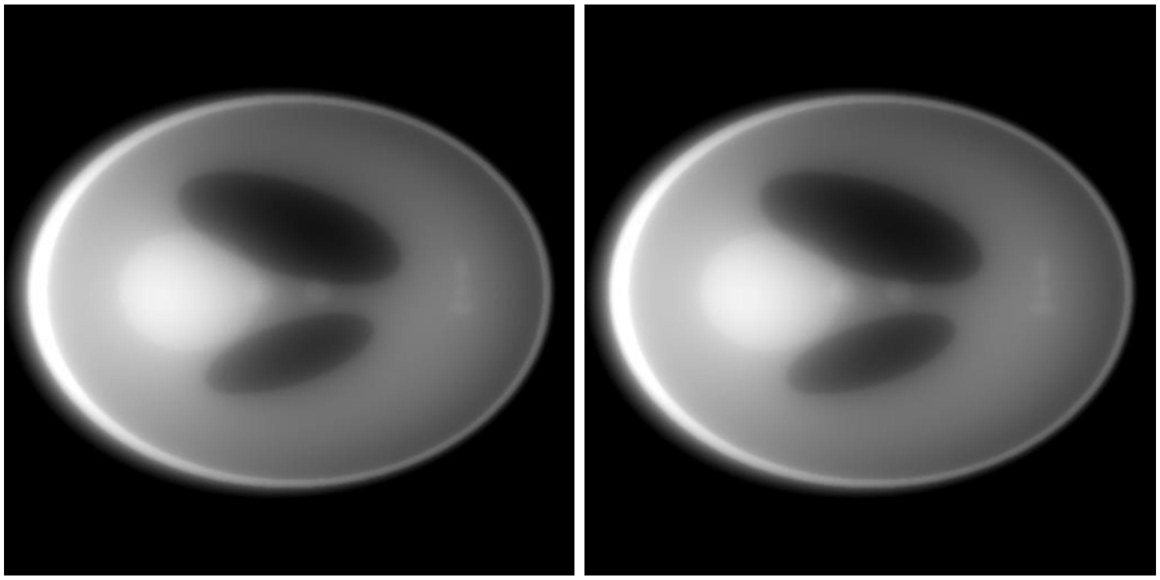
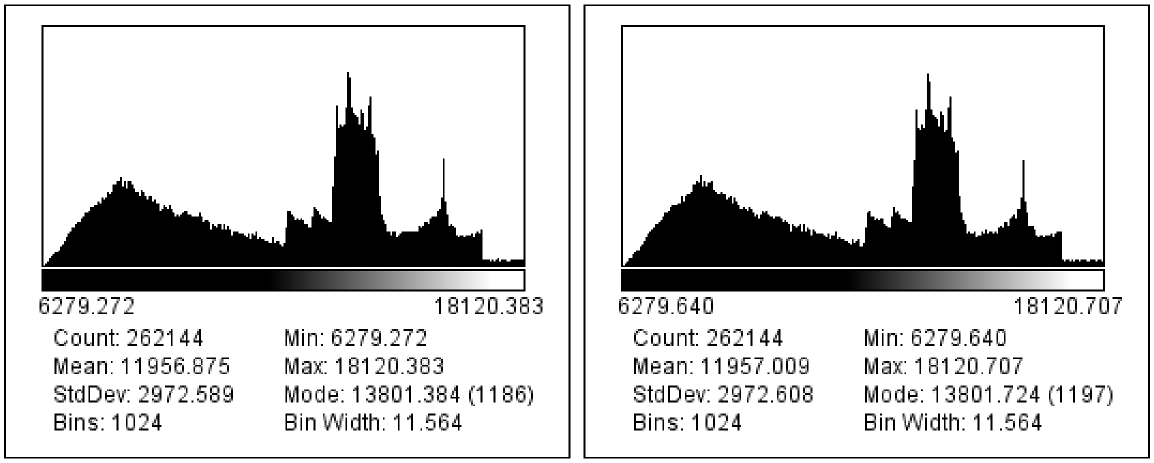


Fig. 31. A slice of the image from the pre-projection integration back-projector (left) and the GPU-optimized back-projector (right).



```
Count: 262144        Min: 6279.272
Mean: 11956.875      Max: 18120.383
StdDev: 2972.589     Mode: 13801.384 (1186)
Bins: 1024           Bin Width: 11.564
```

```
Count: 262144        Min: 6279.640
Mean: 11957.009      Max: 18120.707
StdDev: 2972.608     Mode: 13801.724 (1197)
Bins: 1024           Bin Width: 11.564
```

Fig. 32. Histograms for the image slices shown in Fig. 31. Pre-projection integration is on the left and GPU-optimized is on the right.

Similar to the image quality analysis done for the pre-projection integration projector, a difference image is used to get a better understanding of the image quality of the pre-projection integration back-projector. Fig. 33 shows the pre-projection integration image slice subtracted from the GPU-optimized image slice. The histogram for that difference image is shown in Fig. 34. The difference, which ranges from -4.4 to 4.8, is four orders of magnitude smaller than the mean value of the image slice.

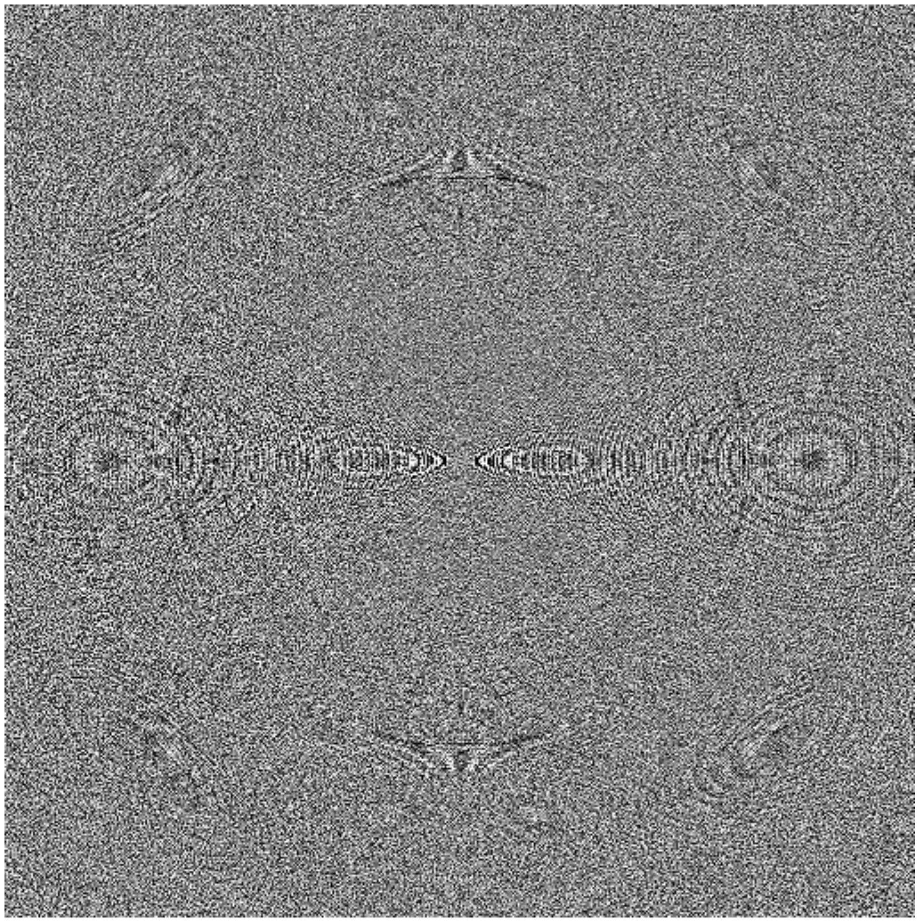


Fig. 33. A slice of the pre-projection integration image subtracted from the same row of the GPU-optimized image. The histogram has been stretched to enhance the visualization of the error.

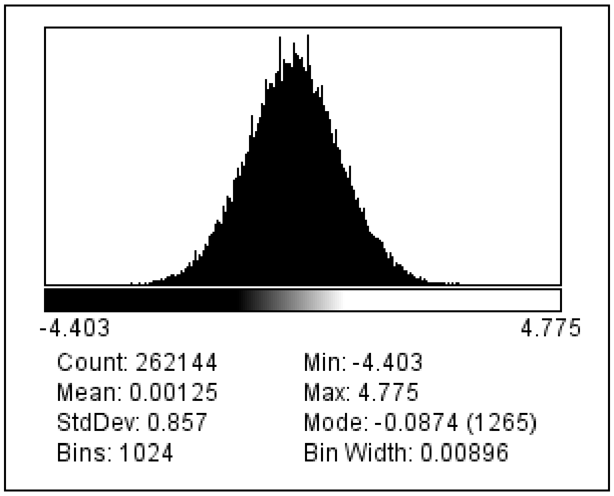


Fig. 34. Histogram of the difference image in Fig. 33.

However, similar to the difference sinogram in Fig. 25, the difference image in Fig. 33 shows aliasing artifacts when inspected visually. Although the magnitude of these artifacts is very small, it is important to understand their source. Again, it turns out that the cause is the quantization of the interpolation coefficients by the low precision hardware interpolation of the GPU's Texture Units. A version of the pre-projection integration back-projector was written that manually performs the bilinear interpolation of the SAT reads with software. This projector produces output that does not contain the aliasing artifacts. Fig. 35 contains the same difference image from Fig. 33, but software was used to do bilinear interpolation instead of hardware. Fig. 36 shows a histogram of the image in Fig. 35. The histogram shows that, in addition to eliminating the aliasing artifacts, the software interpolation also decreases the error by an order of magnitude.

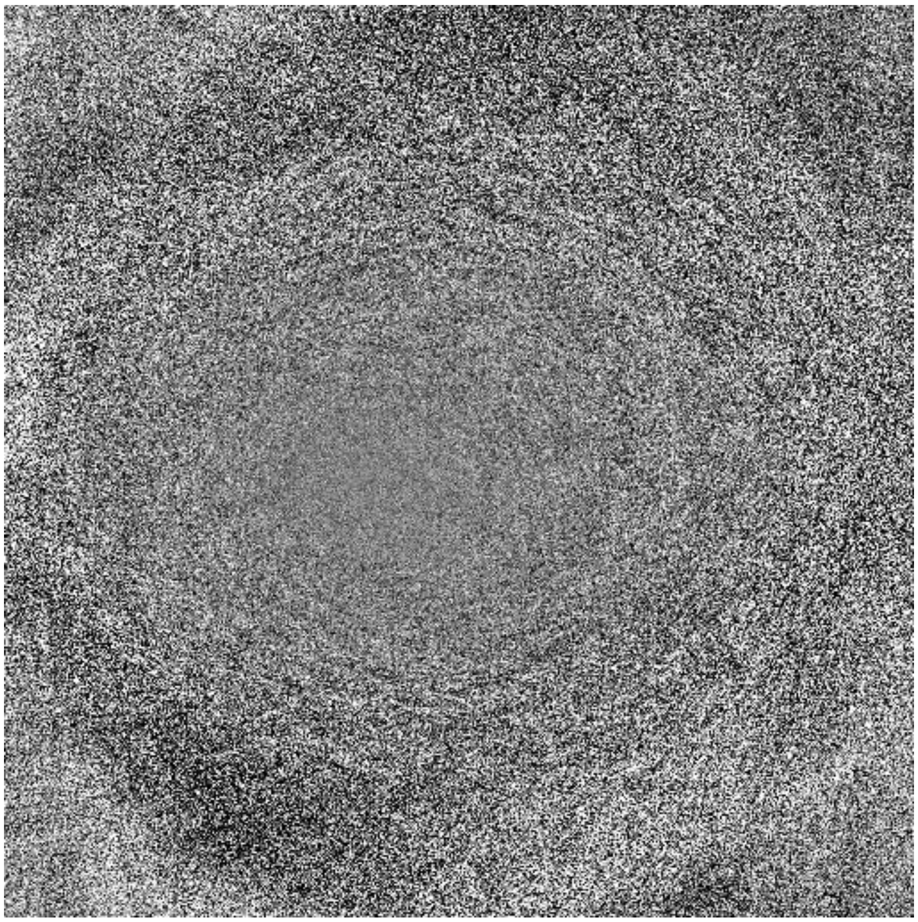


Fig. 35. A slice of the pre-projection integration image subtracted from the same row of the GPU-optimized image. In this figure, the pre-projection integration version uses a software-based interpolation method instead of the hardware-based Texture Unit method. The histogram has been stretched to enhance the visualization of the error.

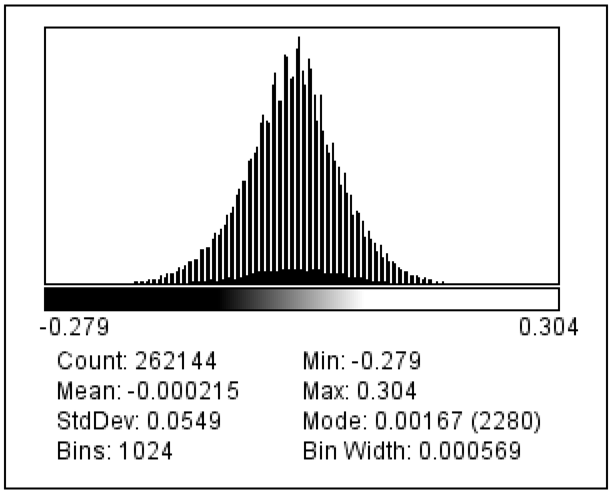


Fig. 36. Histogram of the difference image in Fig. 35.

D. Iterative Image Reconstruction with Pre-Projection Integration

An important question raised in Section IV is whether the level of error introduced by the pre-projection integration method is acceptable. In order to determine the answer, the pre-projection integration method was tested within an iterative image reconstruction algorithm. The GPU-optimized and pre-projection integration projector/back-projector pairs were implemented within Jeffrey Fessler's Image Reconstruction Toolkit (IRT) (Fessler J. A.). Note that while the pre-projection integration projector and back-projector are implemented in CUDA, the IRT runs in MATLAB. Therefore, in order to run the projectors/back-projector pairs within the IRT, they needed to be compiled as MATLAB Executable (MEX) programs. Fortunately, MATLAB provides mechanisms to create MEX functions that run CUDA code.

A Penalized Weighted Least-Squares (PWLS) algorithm was chosen to evaluate the pre-projection integration and GPU-optimized projector/back-projector pairs (Fessler J. A., Penalized weighted least-squares image reconstruction for positron emission tomography, 1994). The algorithm uses a Preconditioned Conjugate Gradient (PCG) method with a circulant pre-conditioner (Fessler & Booth, Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction, 1999). The input data came from ideal projections of the 3D Shepp-Logan phantom, so the true image was known. Three iterations of the algorithm were run. The results from the pre-projection integration projector/back-projector pair were compared to the results from the GPU-optimized projector/back-projector pair. Fig. 37 shows a slice from the image volumes produced using the two different methods for projection and back-projection. The contrast of the Shep-Logan phantom image is fairly low (Gonzalez & Woods, 2008).

Therefore, the histograms of the images in Fig. 37 are stretched to enhance visualization. The images are indistinguishable by visual inspection. Fig. 38 shows the image histograms for the slices in Fig. 37. The histogram statistics are nearly equivalent as well.

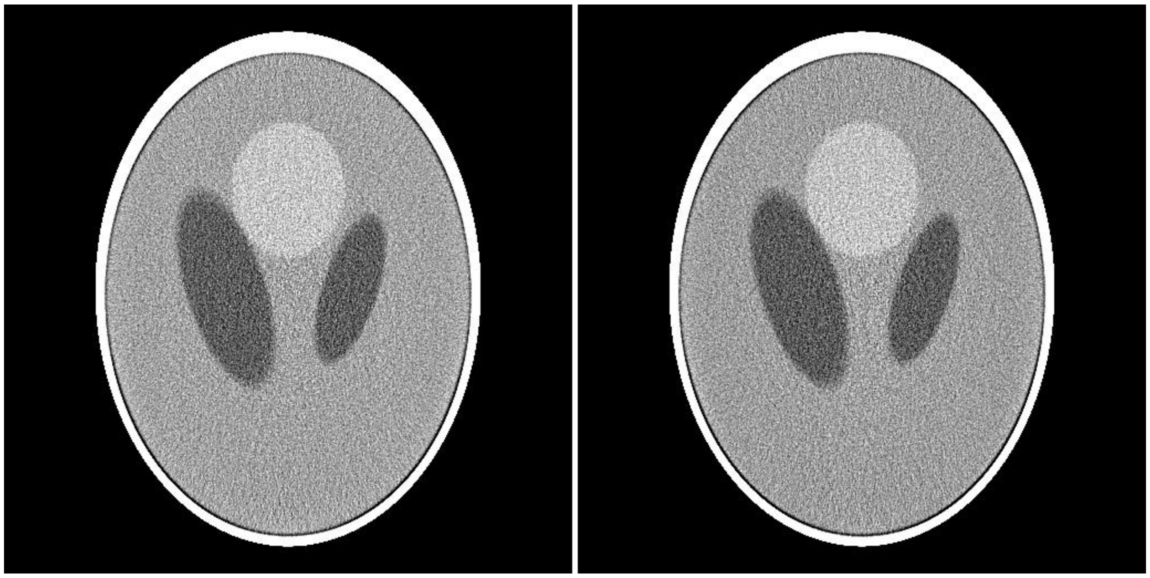
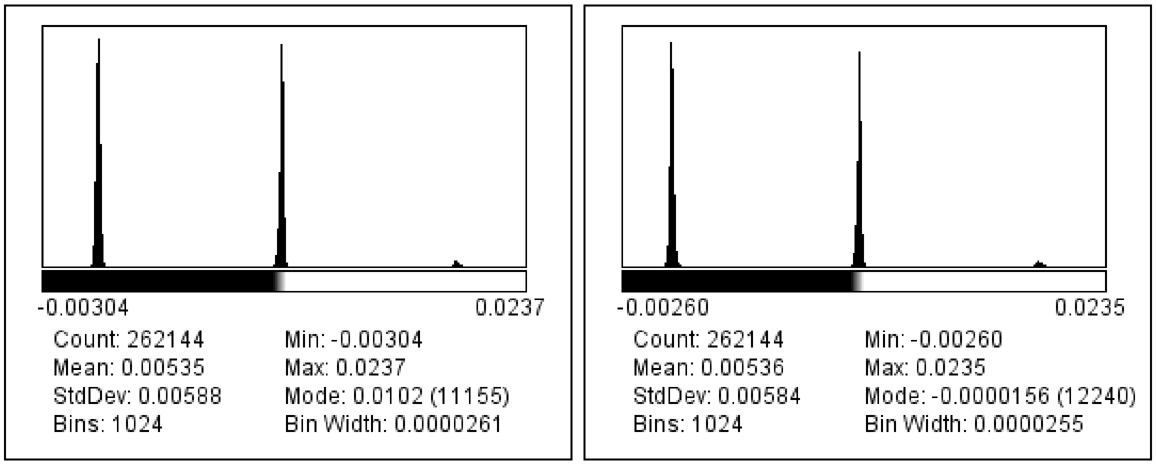


Fig. 37. Slices from image volumes created by the PWLS-PCG algorithm. The pre-projection integration method was used to create the slice on the left and the GPU-optimized projectors were used to create the slice on the right.



```
-0.00304                          0.0237
 Count: 262144        Min: -0.00304
 Mean: 0.00535        Max: 0.0237
 StdDev: 0.00588      Mode: 0.0102 (11155)
 Bins: 1024           Bin Width: 0.0000261
```

```
-0.00260                          0.0235
 Count: 262144        Min: -0.00260
 Mean: 0.00536        Max: 0.0235
 StdDev: 0.00584      Mode: -0.0000156 (12240)
 Bins: 1024           Bin Width: 0.0000255
```

Fig. 38. Histograms for the image slices in Fig. 37. The pre-projection integration method is on the left and the GPU-optimized is on the right.

A difference image was then created by subtracting the image created by the pre-projection integration projector/back-projector pair from the image created by the GPU-optimized projector/back-projector pair. This difference image is shown in Fig. 39 and the histogram is shown in Fig. 40. The difference image consists of low intensity noise inside the phantom. However, it shows a few ring artifacts where the edges of the phantom are located. The intensity of the ring artifact is small. In addition, both the GPU-optimized and pre-projection integration images show similar ring artifacts when compared to the true image. Fig. 41 shows two difference images: the true image minus the image created by the GPU-optimized method and the true image minus the image created by the pre-projection integration method. Because both of these difference images contain ring artifacts, it is likely not something that is introduced by the pre-projection integration method. Rather, it is likely that the PWLS-PCG algorithm is very sensitive to round-off errors while trying to preserve edges.

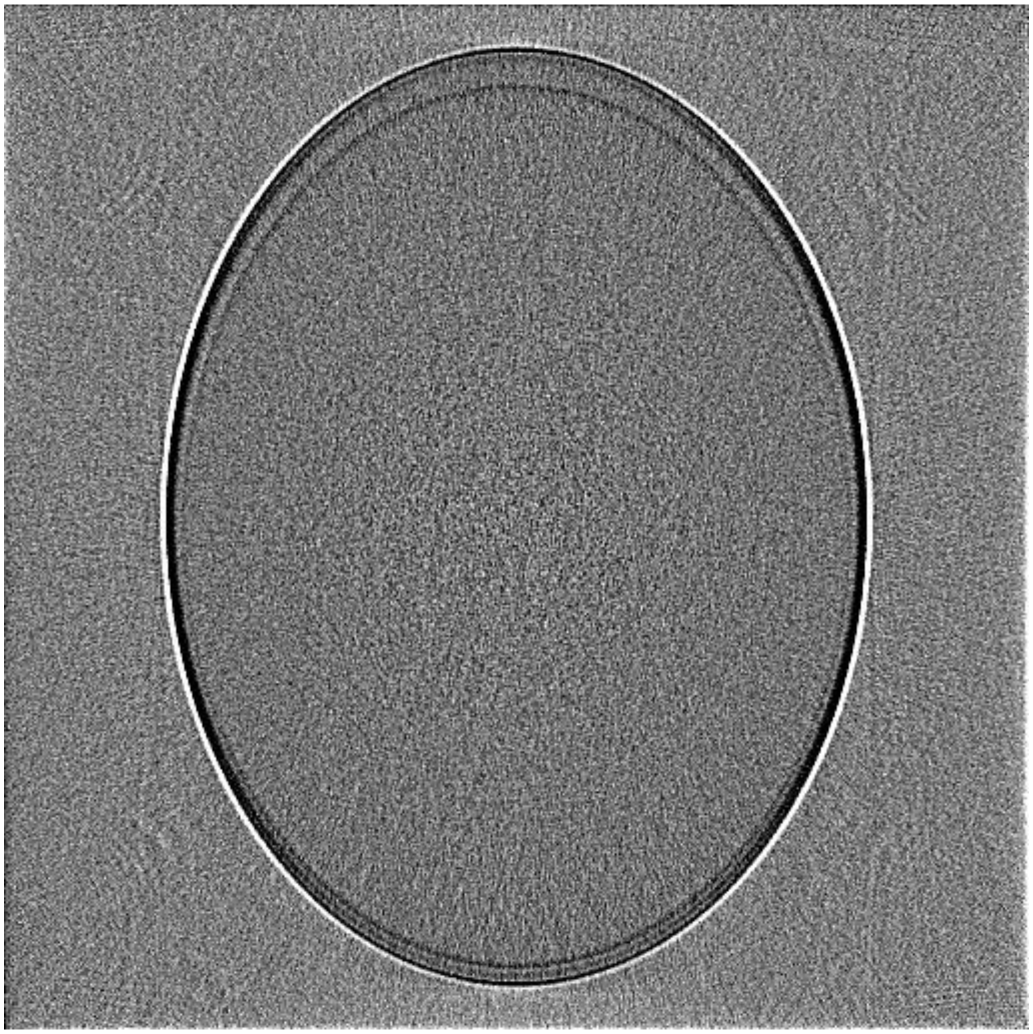


Fig. 39. GPU-optimized output minus pre-projection integration output. The histogram has been stretched to enhance the visualization of the error.

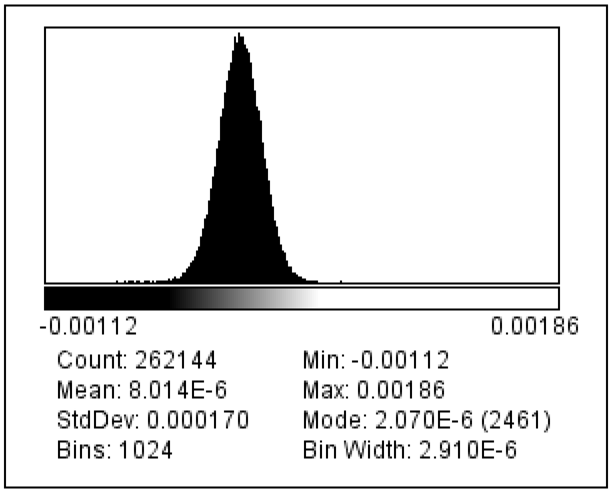


Fig. 40. Histogram of the difference image in Fig. 39.

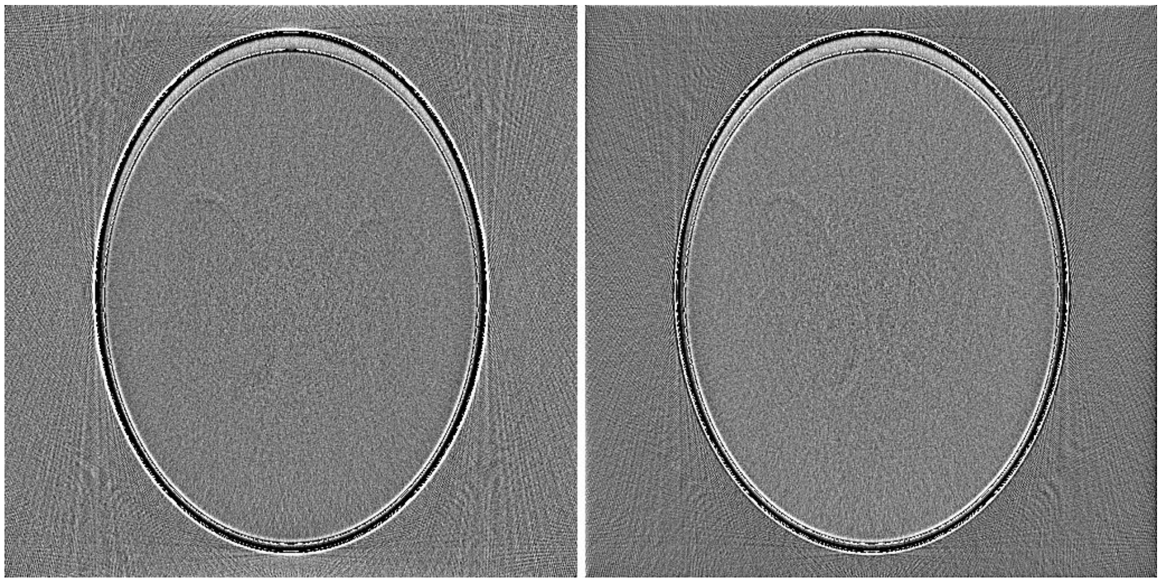


Fig. 41. True image minus pre-projection integration output (left) and true image minus GPU-optimized version (right). Note that both contain ring artifacts where the edge of the Shepp-Logan phantom is located. The histogram has been stretched to enhance the visualization of the error.

Finally, recall that Sections IV.B.2A and IV.C.2 highlighted the potential issue of using the low precision hardware interpolation of the GPU’s Texture Units. Therefore, a difference image was created that shows the difference between pre-projection integration with software interpolation and pre-projection interpolation with hardware interpolation. The difference image is shown in Fig. 42 and the histogram of that difference is shown in Fig. 43. The difference image is almost completely low intensity noise with no visible aliasing artifacts in either. This indicates that the image quality effects of low precision hardware interpolation can be mitigated when the projector and back-projector are used within an image reconstruction algorithm with regularization.

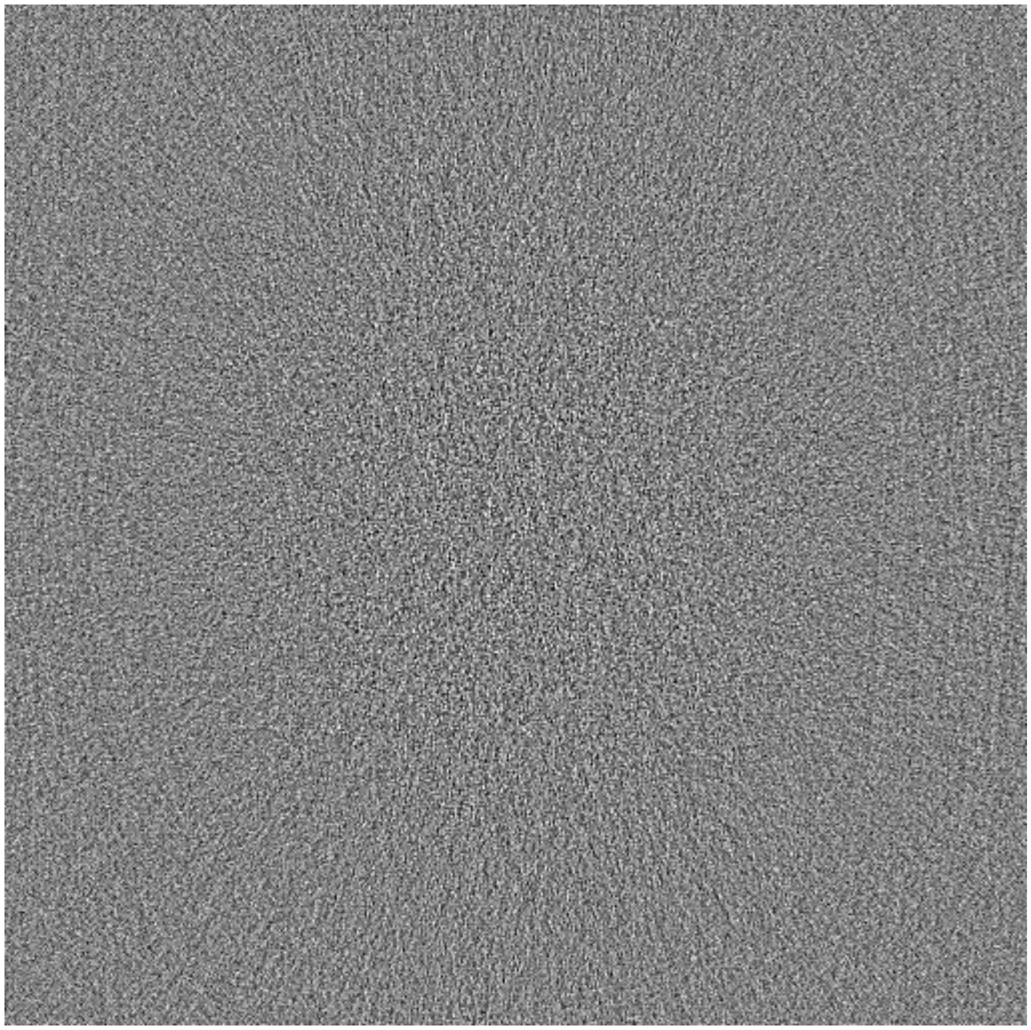
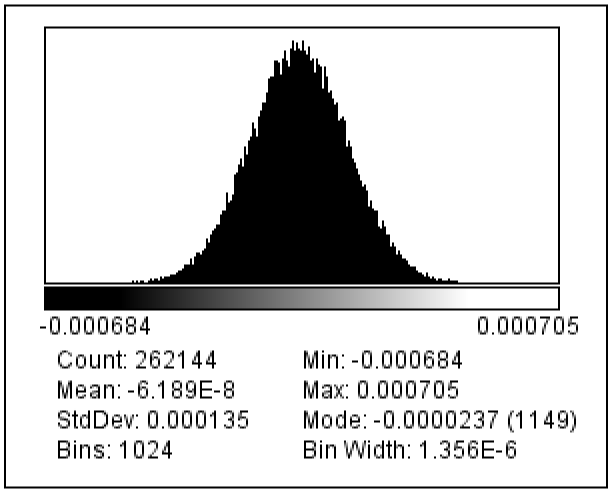


Fig. 42. Image showing the difference in the PWLS-PCG output image for software interpolation and hardware interpolation. The histogram has been stretched to enhance the visualization of the error.



Count: 262144
Mean: -6.189E-8
StdDev: 0.000135
Bins: 1024
Min: -0.000684
Max: 0.000705
Mode: -0.0000237 (1149)
Bin Width: 1.356E-6

Fig. 43. Histogram for the difference image in Fig. 42.

V. CONCLUSION

In this study, a novel technique called pre-projection integration was investigated with the goal of speeding up distance-driven projection and back-projection in cone beam CT. GPU-based projection and back-projection code was first enhanced with the pre-projection integration method. The pre-projection integration method also requires an additional step, SAT generation, which was implemented and characterized. Both the performance and image quality of the pre-projection integration method were carefully analyzed. Finally, the pre-projection integration method was tested within a regularized image reconstruction algorithm.

The pre-projection integration method has the potential to substantially decrease the execution times of regularized iterative image reconstruction algorithms for cone beam CT. Iterative image reconstruction makes use of projection and back-projection operations. Many different models for projection and back-projection have been studied. The focus of this study, however, is distance-driven projection and back-projection due to their high image quality and prevalence in modern iterative image reconstruction algorithms.

In the distance-driven model, both projection and back-projection involve the overlap kernel in which the sum of the elements within a region defined by detector or image boundaries is computed. Although the overlap kernel accounts for a significant portion of the overall projection and back-projection execution time, it has seen very little optimization. The pre-projection integration technique seeks to minimize the amount of time the overlap kernel takes when executed on GPUs while maintaining the high image quality of the distance-driven model. It uses SATs to quickly get the sum of the intensities in a region defined by detector or image boundaries. It also takes advantage of the ultra-fast hardware interpolation provided in a GPU's Texture Units.

The performance boost that pre-projection integration can deliver depends on the size of the CT detector and the image being reconstructed. GPUs perform at the highest level when memory accesses are limited. Using SATs for projections means that a constant, small number of global memory accesses are required. This can substantially reduce the number of reads needed. For example, images that have small voxels would typically require many reads to ascertain all values. Using pre-projection integration requires only four reads, even if the detector shadow falls upon tens of voxels. However, images that have larger voxels would not require as many reads in the traditional implementation. Thus, using pre-projection integration would save fewer reads. The speedup that pre-projection integration offers to projection increases from 2.5x to 4.4x as the image size increases. Conversely, the speedup that pre-projection integration offers to back-projection *decreases* from 5.3x to 1.2x as the image size increases. Still, both offer a speedup of over 4x for certain image sizes.

After verifying that the pre-projection integration method can deliver a substantial performance boost, it was necessary to verify that it does not have any adverse image quality effects. There are two main sources of error introduced by the pre-projection integration method: SAT error and hardware interpolation error. Both were carefully considered in this study. The SAT error was mitigated by subtracting the mean of the image before the integration step. The error introduced by the low precision hardware interpolation of the GPU Texture Units was not mitigated directly. However, using the IRT, it was shown that the aliasing artifacts produced by the hardware interpolation are mitigated indirectly by the regularized image reconstruction algorithm. Therefore, it is clear that the pre-projection integration method is a very promising technique for cone beam CT iterative image reconstruction algorithms.

Future studies could investigate a number of things related to pre-projection integration. First, the pre-projection integration method could be characterized on different hardware. AMD GPUs and Intel Coprocessors might treat texture interpolation differently, which could have an effect on performance and image quality. The pre-projection integration method could also be tested with different cone beam CT systems that have more detector cells. This could make the back-projection performance more favorable. Additionally, the SAT error could be further reduced by breaking up an image into smaller pieces and creating smaller SATs for each. That would require the handling of edge cases if a detector footprint overlapped with more than one SAT. Finally, the distance-driven method with pre-projection integration could be compared to the separable footprints method with regard to both performance and image quality. The tradeoff between performance and image quality would be very helpful for the design of future image reconstruction algorithms that require both quick execution and superior image quality.

BIBLIOGRAPHY

Basu, S., & De Man, B. (2006). Branchless distance driven projection and backprojection. *Electronic Imaging*, 60650Y-60650Y.

Chevalier, D. E., & Drapkin, E. (2009). PET 3D iterative reconstruction on Cell BE. *IEEE Nuclear Science Symposium Conference Record (NSS/MIC).*

Christiaens, M., De Sutter, B., De Bosschere, K., Van Campenhout, J., & Lemahieu, I. (1999). A fast, cache-aware algorithm for the calculation of radiological paths exploiting subword parallelism. *Journal of systems architecture 45(10)*, 781-790.

Crow, F. C. (1984). Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics 18(3)*, 207-212.

De Man, B., & Basu, S. (2002). Distance-driven projection and backprojection. *Nuclear Science Symposium Conference Record Vol. 3* (pp. 1477-1480). IEEE.

De Man, B., & Basu, S. (2004). Distance-driven projection and backprojection in three dimensions. *Physics in medicine and biology 49(11)*, 2463.

Feldkamp, L., Davis, L., & Kress, J. (1984). Practical cone-beam algorithm. *JOSA A 1(6)*, 612-619.

Fessler, J. A. (1994). Penalized weighted least-squares image reconstruction for positron emission tomography. *Medical Imaging, IEEE Transactions on 13(2)*, 290-300.

Fessler, J. A. (2009, November 19). *Analytical Tomographic Image Reconstruction Methods*. Retrieved September 2015, 14

Fessler, J. A. (n.d.). *Image Reconstruction Toolbox*. Retrieved from http://web.eecs.umich.edu/~fessler/code/index.html

Fessler, J. A., & Booth, S. D. (1999). Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction. *Image Processing, IEEE Transactions on 8(5)*, 688-699.

Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing.* Upper Saddle River, NJ: Pearson Prentice Hall.

Gross, D., Heil, U., Schulze, R., Schoemer, E., & Schwanecke, U. (2009). GPU-based volume reconstruction from very few arbitrarily aligned X-ray images. *SIAM Journal on Scientific Computing 31(6)*, 4204-4221.

Hensley, J., Scheuermann, T., Coombe, G., Singh, M., & Lastra, A. (2005). Fast Summed-Area Table Generation and its Applications. *Computer Graphics Forum 24(3)*, (pp. 547-555).

Hsieh, J. (2009). *Computed tomography: principles, design, artifacts, and recent advances.* Bellingham, WA: SPIE.

Hudson, H. M., & Larkin, R. S. (1994). Accelerated image reconstruction using ordered subsets of projection data. *Medical Imaging, IEEE Transactions on 13(4)*, 601-609.

IEEE Standards Committee. (2008). 754-2008 IEEE standard for floating-point arithmetic. *IEEE Computer Society Std, 2008*.

Kak, A. C., & Slaney, M. (2001). *Principles of computerized tomographic imaging.* Society for Industrial and Applied Mathematics.

Lacroute, P., & Levoy, M. (1994). Fast volume rendering using a shear-warp factorization of the viewing transformation. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (pp. 451-458). ACM.

Lewis, J. (1995). Fast normalized cross-correlation. *Vision interface 10(1)*, (pp. 120-123).

Long, Y., Fessler, J., & Balter, J. M. (2010). 3D forward and back-projection for X-ray CT using separable footprints. *Medical Imaging, IEEE Transactions on 29(11)*, 1839-1850.

Manjeshwar, R. M., Ross, S. G., Iatrou, M., Deller, T. W., & Stearns, C. W. (2006). Fully 3D PET iterative reconstruction using distance-driven projectors and native scanner geometry. *Nuclear Science Symposium Conference Record Vol. 5* (pp. 2804-2807). IEEE.

Medeiros, H., Holguín, G., Shin, P. J., & Park, J. (2010). A parallel histogram-based particle filter for object tracking on SIMD-based smart cameras. *Computer Vision and Image Understanding, 114(11)*, 1264-1272.

Medeiros, H., Park, J., & Kak, A. (2008). A parallel color-based particle filter for object tracking. *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on* (pp. 1-8). IEEE.

Mueller, K., & Yagel, R. (2000). Rapid 3-D cone-beam reconstruction with the simultaneous algebraic reconstruction technique (SART) using 2-D texture mapping hardware. *Medical Imaging, IEEE Transactions on 19(12)*, 1227-1237.

Mueller, K., Yagel, R., & Wheller, J. J. (1999). Fast implementations of algebraic methods for three-dimensional reconstruction from cone-beam data. *Medical Imaging, IEEE Transactions on 18(6)*, 538-548.

Nguyen, H. (2007). *Gpu gems 3.* Addison-Wesley Professional.

Perez, I. B., Jimenez, E. S., & Thompson, K. R. (2014). A high-performance GPU-based forward-projection model for computed tomography applications. *SPIE Optical Engineering+ Applications* (pp. 92150A-92150A). International Society for Optics and Photonics.

Pratx, G., Chinn, G., Habte, F., Olcott, P., & Levin, C. (2006). Fully 3-D list-mode OSEM accelerated by graphics processing units. *Nuclear Science Symposium Conference Record Vol. 4* (pp. 2196-2202). IEEE.

Scherl, H., Keck, B., Kowarschik, M., & Hornegger, J. (2007). Fast GPU-based CT reconstruction using the common unified device architecture (CUDA). *Nuclear Science Symposium Conference Record* (pp. 4464-4466). IEEE.

Scherl, H., Koerner, M., Hofmann, H., Eckert, W., Kowarschik, M., & Hornegger, J. (2007). Implementation of the FDK algorithm for cone-beam CT on the cell broadband engine architecture. *Medical Imaging* (pp. 651058-651058). International Society for Optics and Photonics.

Shepp, L. A., & Logan, B. F. (1974). The Fourier reconstruction of a head section. *Nuclear Science, IEEE Transactions on 21(3)*, 21-43.

Siddon, R. L. (1985). Fast calculation of the exact radiological path for a three-dimensional CT array. *Medical physics 12(2)*, 252-255.

Thibault, J.-B., Sauer, K. D., Bouman, C. A., & Hsieh, J. (2007). A three-dimensional statistical approach to improved image quality for multislice helical CT. *Medical physics 34(11)*, 4526-4544.

Thibault, J.-B., Sauer, K., Bouman, C., & Hsieh, J. (2003). High quality iterative image reconstruction for multi-slice helical CT. *International Conference on Fully 3D Reconstruction in Radiology and Nuclear Medicine.* Saint Malo, France.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on Vol. 1* (pp. I--511). IEEE.

Wilt, N. (2013). *The cuda handbook: A comprehensive guide to gpu programming.* Pearson Education.

Wu, M., & Fessler, J. A. (2011). GPU acceleration of 3D forward and backward projection using separable footprints for X-ray CT image reconstruction. *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, (pp. 56-9).

Xie, L., Hu, Y., Chen, Y., & Shi, L. (2015). A rapid parallelization of cone-beam projection and back-projection operator based on texture fetching interpolation. *SPIE Medical Imaging* (pp. 94122S-94122S). International Society for Optics and Photonics.

Xu, F. (2010). Fast implementation of iterative reconstruction with exact ray-driven projector on GPUs. *Tsinghua Science & Technology 15(1)*, 30-35.

Xu, F., & Mueller, K. (2006). A comparative study of popular interpolation and integration methods for use in computed tomography. *Biomedical Imaging: Nano to Macro. 3rd IEEE International Symposium on* (pp. 1252-1255). IEEE.