Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution.

-Albert Einstein, (1929)

University of Alberta

ENHANCEMENTS TO RECONSTRUCTION TECHNIQUES IN COMPUTED TOMOGRAPHY USING HIGH PERFORMANCE COMPUTING

by

Steven Nicholas Eliuk

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

To my wife, mother and father, and entire family.
Thank you for the support and teaching me more than I ever thought possible.

Abstract

Computers have been used in diagnostic imaging for decades, but High-Performance Computing (HPC) in diagnostic imaging is rather rare because of the high cost associated with traditional HPC. Recent advancements in shared memory computers and the large market for commodity graphic cards have provided the means for technological evolution at a more rapid pace and decreased cost. These improvements can provide a detailed 3D view of the human body, near instantaneous reconstruction for Computed Tomography (CT), more intricate reconstruction algorithms, and reduced radiation exposure to patients through the use of iterative reconstruction techniques. The use of HPC in diagnostic imaging is realizable with current commodity hardware. For this reason a new reconstruction, visualization, and interaction environment has been constructed to decrease reconstruction time, decrease radiation exposure, and reconstruct images from raw attenuation values in medical-CT. The thesis explores the use of different computational tools in the Advanced Reconstruction Environment for Medical Imaging (AREMI). AREMI uses multi-core and multi-GPU programming for reconstruction of real-world clinical raw attenuation values from Siemens Definition Flash CT scanners. The use of this environment, analysis of various reconstruction techniques, and the ability to output High Dynamic Range (HDR) has provided a means to analyze various aspects of CT reconstruction in medical applications. The thesis also provides a concise review of filtered back-projection and algebraic methods in CT-reconstruction and provides insight into programming advanced CT-reconstruction concepts in a serial, multi-CPU, GPU, and multi-GPU environment that are not formally explained in literature; therefore, making this topic accessible and valuable to a broad audience.

Acknowledgements

I greatly appreciate the support from Dr. Boulanger - University of Alberta Department of Computing Science, the University of Alberta Department of Radiology and Diagnostic Imaging, the National Sciences and Engineering Research Council, Nvidia, SGI, and Servier Laboratories for the support of myself and research project. The hardware resources donated, on several occasions, from Nvidia were appreciated and led to much of the analysis presented in this thesis. A special acknowledgement to Dr. Noga - University of Alberta Department of Radiology and Diagnostic Imaging, is deserving for her advice, expertise, and support of the research. Likewise, the University of Alberta Department of Radiology and Diagnostic Imaging for the use of specialized scanners, expertise, and their ability to schedule use of people or technology.

There are many great collaborators not mentioned below that I have had the pleasure to work with and I would like to formally thank them. Most importantly though, I would like to recognize Dr. N. Birkbeck as he was instrumental on the evaluation of concepts and practical implementation details. We often collaborated on various projects and I truly appreciate his willingness to discuss topics disjoint to his own. I also appreciate the discussion with Dr. P. Toft, his 1996 dissertation was incredibly useful and his correspondence helped develop the breadth and thoroughness of this thesis. Dr. V.V. Vlček was kind enough to discuss his thesis on many occasions and those discussions led to a better understanding of the topic area. Benjamin Keck has many influential publications in iterative reconstruction techniques and we talked on several occasions concerning difficulties with iterative methods, his willingness for discussion was pivotal in the development of my own conceptual ideas presented in this thesis and for this I am grateful. Dr. K. Stierstorfer, from Siemens, provided assistance in the understanding of the raw format of Siemens CT-scanners, subtleties of parameters in scans, data layout, and always answered questions with the utmost professionalism and was always willing to help on non-proprietary information. Dr. C. Mendl was an important collaboration partner and we produced one joint paper on the comparison of various HPC methods for convolution weighted back-projection seen in Section 6.3. Lastly, Keith Ruiter for his help in editing publications and this document, your time was always appreciated. Without the considerable help of these people, I honestly believe, the completion of this thesis would have required much more time.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Symbols

$f(x,y)$	2D Image Slice	18
$g(l,\theta)$	sinogram or Radon transform	18
I_d	integrated X-ray intensity for a given detector	18
$S_0(E)$	X-ray spectrum	18
$\mu(s;E)$	linear attenuation coefficient	18
I_0	reference intensity of detector	18
$p(\gamma,\beta)$	sinogram for fan-beam geometry	17
γ	detector channel angular offset in fan-beam geometry	17
β	X-ray source angular offset	17
ϱ	spatial frequency	23
F_{1D}	1D Fourier transform	23
$\lvert\varrho\rvert$	ramp filter	25
D	source distance to isocenter	28
$\pm\,\tau$	region of interest value for restricting recon size	28
$c(\gamma)$	convolution of channel array	28
$q(\gamma,\beta)$	1D convolution of projection values with source angle change	29
$R\{\}$	Radon transform	38
b	linear vector representing a sinogram	38
f	linear vector representing recon image	38
A	linear vector representing Radon transform system matrix	38
$\rho^{'}$	distance of ray to pixel center location	43
k	linear channel dimension of sinogram	38
l	linear projection dimension of sinogram	38
m	linear x dimension of recon image	39
n	linear y dimension of recon image	39
λ	relaxation parameter	54

Chapter 1

Introduction

The ability to produce a volumetric depiction of the inner properties of an object was a revolutionary achievement for industry and medicine. There are many modalities that these digital images can be acquired from, and each has strengths and weaknesses. The modality of choice for this thesis is fan-beam Computed Tomography (CT), as it is widely used in medical imaging and has a number of difficult problems that can be addressed through the use of High-Performance Computing (HPC). In this thesis we show the difficulties in modelling a real-world problem, as in our problem much of the information is unavailable. A solution is outlined, and many different aspects of fan-beam CT-reconstruction are evaluated. This implementation has the ability to reconstruct raw-attenuation values in a timely fashion by specific algorithm choices and an implementation targeted at multi-GPU environments is described. We are then able to perform qualitative and quantitative analysis on all methods implemented. The ultimate goal is to construct a valid 2D or 3D model of X-ray attenuation given a series of measurements, at various angles, of a specific resolution, and varying levels of noise. The task is a difficult problem computationally. Likewise, much of the information is typically not available from specific scanning mediums. In this chapter we discuss some of the ethical and theoretical motivation, contributions to the research community, and applicability to different readers.

1.1 Ethical Motivation

The introduction of medical-CT in the 1970s was revolutionary as it changed diagnostic medicine. Surgeons could now see problems in the brain and body on diagnostic scans prior to operating, whereas formerly they had to operate to see what was wrong with the patient. It is now widely used in the developed world with close to 70 million CT scans per year acquired in the United States alone [2, 7]. However, there are apparent risks associated with CT because X-rays are used to acquire the volumetric depiction of the inner anatomy of the patient and the radiation used is linked to cancer. Furthermore, clinical use of CT exposes the patient to comparatively high levels of X-ray radiation [54]. Specifically, an X-ray beam is composed of photons of various energies in the range of kilowatts. The photons are attenuated by different materials encountered. Based on the level of attenuation, one can create an image representing the inner properties encountered by the X-ray beam. The attenuation can be thought of as absorption, and this absorption, when significant, can be linked to cancer if DNA is damaged.

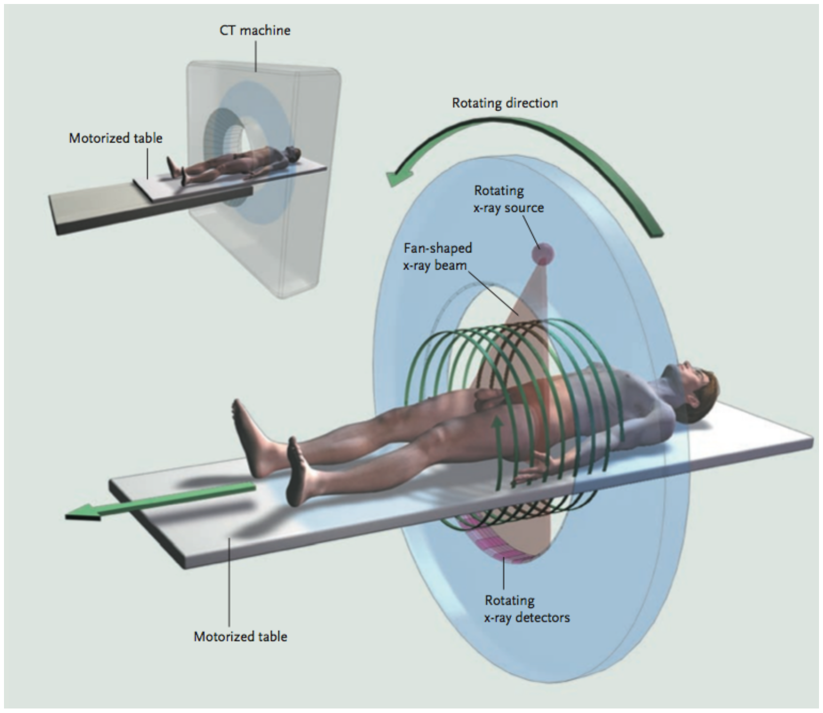
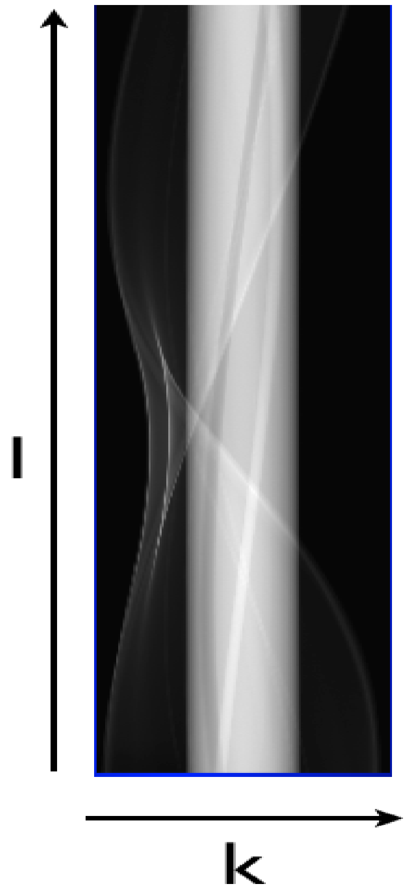


Figure 1.1: Multidetector helical CT acquisition [7].

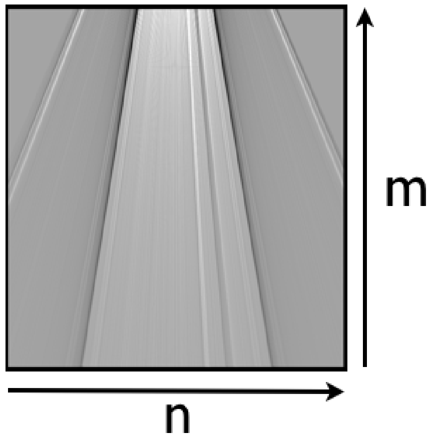
The radiation exposure in CT is classified as high when compared to traditional X-ray technologies. This is understandable because CT acquires a volumetric representation of the body, see Figure 1.1, rather than just a single image as seen in a typical chest X-ray. For instance, when comparing a typical adult abdominal CT, and the relevant organ dose, to dental or lateral chest radiography, the abdominal CT has a relevant dose of 200x more than dental radiography and 67x more than the lateral chest radiography [7]. The larger amount of radiation used in CT, and the escalating use of CT in clinical environments (3-fold since 1993), have led to many studies that show cancer rates do increase substantially with the exposure to CT [2, 7]. Given the high risk associated with a CT study, the research in this thesis has been devoted to reducing the amount of radiation needed (up to 50% in iterative techniques), providing reasonable runtime, and providing a good visual reconstruction environment to build highly detailed images of the inner properties of the body.

1.2 Problem Statement

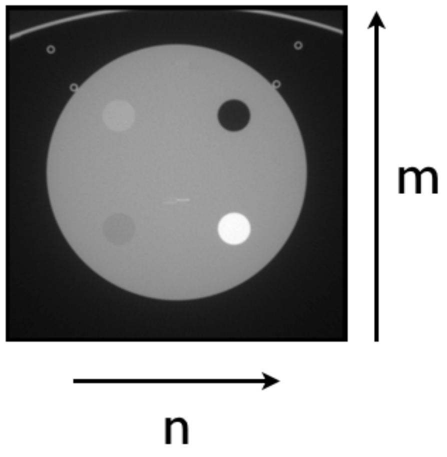
Although there are many details described in this thesis, the core problem that is modelled is that of the Radon transform, and the use of this transformation information to reconstruct the original im-



(a) Sinogram of 1.2c



(b) Projection of angle 90° through image



(c) Reconstructed Phantom

Figure 1.2: (1.2a) represents the sinogram of 1.2c, known numerically as $p(\gamma, \beta)$, for fan-beam geometry where γ represents the offset in the detector fan or channel and β represents the offset of the X-ray source and l and k are the rectilinear coordinates of the sinogram; (1.2b) single projection of angle 90° through reconstruction region; (1.2c) complete back-projection of l=1152 projections and k=736 radial channels, representing a complete sampling of sinogram, also known numerically as $f(x, y)$ where m and n are the rectilinear coordinates of the image.

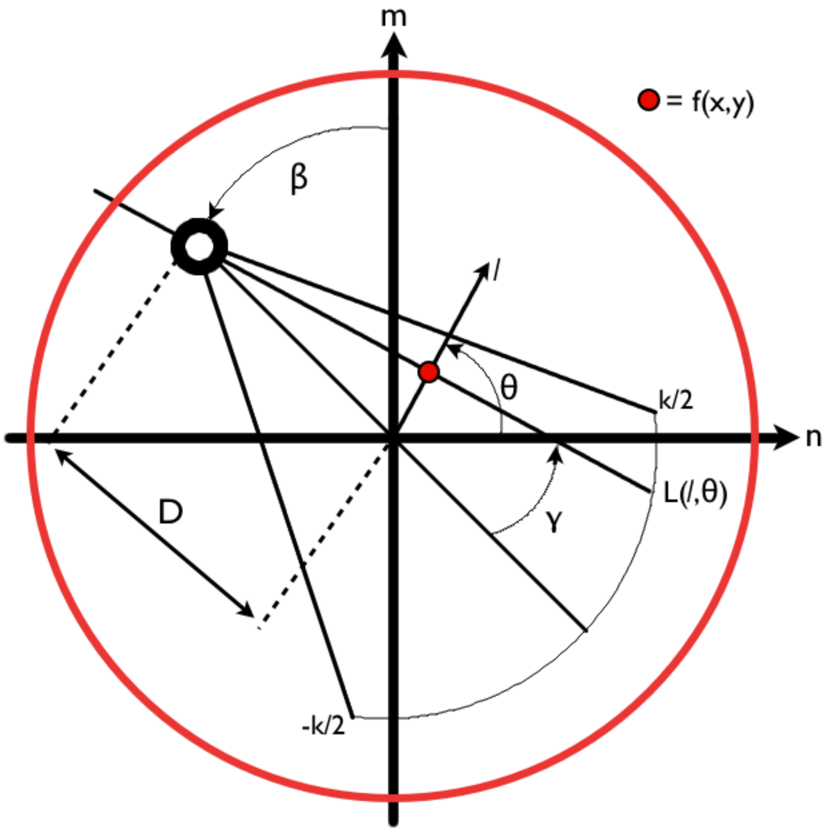


Figure 1.3: Representation of basic fan-beam CT geometry. Where γ = angle index of sensor array for given red pixel location, θ = angle of pixel from origin, D = distance from isocenter to source, β = angle of source, l = distance from isocenter to pixel location, $f(x,y)$ location in reconstruction region, m and n designates y and x-axis of reconstruction region, k number of radial detector channels, and the red circle designates angular projections $(2\pi/\theta_l)$.

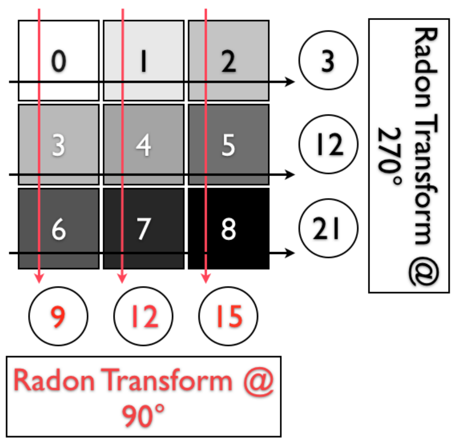


Figure 1.4: Simplified Radon transform for heterogeneous (having different weights) image consisting of nine pixels. The Radon transform is seen involving three rays at 90° and 270°, with resultant sums.

age [79]. The main difference from the mathematical model is that the Radon transform is typically a continuously sampled domain, but it can only be modelled discretely when using real data. Therefore, the discrete input data for reconstruction algorithms is a sinogram with rectilinear coordinates k and l, as seen in Figure 1.2a. The figure depicts k representing the radial detector channels of the scanner and l as the projection data at a specific β-angle. The sinogram is the discrete output of the Radon transform of $f(x, y)$, as seen in Figure 1.2c, where the basic line-intergration principle of the Radon transform is seen in Figure 1.4. Therefore, the resolution and noise properties contained in the sinogram can impact the quality of a reconstruction.

The output of the algorithm is typically a reconstructed image $f(x, y)$ with rectilinear coordinates n and m respectively. The definition of $f(x, y)$ can heavily influence the quality of reconstruction as it controls the sampling of the sinogram in reconstruction algorithms. Therefore, adjusting the size, definition of pixels, ray-width, and interpolation method used can impact the reconstruction, as they are all related to either the image ($f(x, y)$) or the sinogram ($p(\gamma, \beta)$).

1.2.1 Fan-Beam Geometry

Figure 1.3 depicts fan-beam medical-CT geometry, which is the modality of choice for this thesis. The dimension of the reconstruction region can be seen where m and n represent the rectilinear reconstruction coordinates of the image ($f(x, y)$). A given pixel location ($f(x, y)$) is seen and is defined by the location (γ) in the radial arc of detector channels, which ranges from ($\pm k/2$), and the angle of the source (β) during the sample that ranges from $0 \rightarrow 2\pi$.

1.2.2 Parameters

There are a number of algorithmic parameters that are presented in this thesis, such as:

1. number of projections (l),
2. projection ordering,
3. relaxation of algebraic methods,
4. constraints of maximal and minimal value of line-integrals from the Radon transform,
5. reconstruction region (square or circular),
6. pixel or voxel size,
7. parallel workload policies (blocking structures, threads, etc),
8. number of *ART*-iterations,
9. distance metric reconstruction region (number of pixels or physical length based),
10. definition of convergence.

This only represents a subset of the available parameters, but illustrates some of the more important ones for reconstruction techniques. We will investigate some of these parameter choices and how they influence a reconstruction, qualitatively and quantitatively.

1.2.3 Reconstruction Methods

The ability to provide an accurate reconstruction is heavily influenced by the reconstruction method used. There are methods that have reduced computational complexity and can reconstruct images faster at the expense of accuracy or amplification of noise. Likewise, there are methods that are computationally slow, but accurate at representing the physical system. These methods provide superior reconstructions, but are computationally expensive. We investigate the use of algorithms designed for parallel computational devices to make the problems more manageable and provide quicker reconstructions. Specifically, we analyze how the complexity of reconstruction methods varies based on the number of processors or GPUs used.

1.3 Theoretical Motivation

There are a wide variety of problems that face the medical imaging community relating to CT. Mainly:

1. acquisition of good quality scans,
2. choosing the most reasonable frequency kernels,

3. applying frequency kernels in a timely manner,
4. choosing a reconstruction type,
5. reconstruction of scan in a timely manner,
6. choosing the most reasonable spatial kernels,
7. applying spatial kernels in a timely manner,
8. reduction of artifacts,
9. and translation of visual perception of acquired images into a mental conceptual understanding of the anatomy.

These issues are never independent and, in reality, are usually directly dependent on each other. That is to say, without a good acquisition medium, coupled with good quality frequency and spatial kernels, and a quick reconstruction with minimal artifacts, a mental model of the depicted anatomy, with high confidence, is not possible. Likewise, we mention timely manner, as without reasonably fast computation time, in terms of seconds or minutes, in items 3, 5, and 7, the personnel costs and throughput of the scanning modality is decreased. The techniques implemented in this document tend to focus more on the enhancement of items 2 to 9. By focusing the attention to these areas we are able to provide a dynamic visualization environment for those reviewing the reconstructions or trying to understand the qualitative and quantitative impact of parameters.

Item 4 in the above list is important to the correct representation of the material encountered by X-rays. There are two variations of reconstruction methods in CT, which will be reviewed in Section 2.6 and Section 2.12.5. The most commercially used reconstruction scheme is Filtered Back-Projection (FBP), and variations, proposed initially by Feldkamp, Davis, and Kress (FDK) [20]. The less common technique is referred to as the algebraic method to solve the inverse Radon transform and will be shown to be computationally intensive. There are also variations that use an approximation to the solution to converge on a reasonable result through an iterative update method.

The complexity of the simplest reconstruction algorithms, FDK, is $O(n^4)$, where n is the the rectilinear coordinate of the reconstruction region and approximately equal to the number of projections, and considered to be the least computationally demanding method [61]. From the view of computational analysis, the simplest is already challenging and, until the advent of Field Programmable Gate Arrays (FPGAs), reconstruction often took more than ten minutes for a 256^3 volume [61], and even several hours to reconstruct hi-res multi-slice volumes. Looking inside today's commercial-CT machines, one can generally find specialty devices such as the ASIC chip from Terarecon Inc. or FPGAs, as these boards were designed for fast execution of the FDK type algorithms [61]. The use of the specialty hardware devices was groundbreaking, as reconstruction times of tens of minutes were reduced to seconds for 256^3 volumes. These devices are expensive and lacked

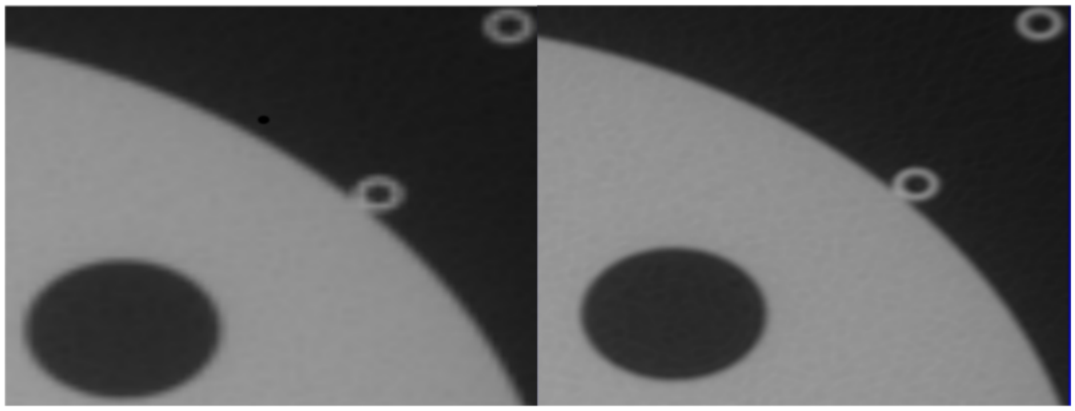


Figure 1.5: Left image: Siemens Definition Flash+ 128, image size $(m = n = 256)$, reconstruction with standard grid and unknown interpolation scheme. Right image: implemented fine-grid reconstruction, equal image size, linear interpolation. Obvious aliasing is seen in the left image, less noticeable in the right image. No edge enhancing kernels were used during reconstruction.

the ability to change algorithmic procedures because the hardware was specifically designed for a specific algorithm. Also, commercial-CT algorithms appear to lack fine-grid sampling and have possibly relied upon proprietary interpolation schemes and filters to reconstruct at high quality. This is likely as even the simple higher order interpolation scheme seen in Figure 1.5 seems to provides a better quality, less apparent aliasing, reconstruction than the commercial algorithm. These coarser grained grids are most likely used to cut down reconstruction time, though a solid answer is difficult to find as most of the commercial techniques are proprietary and closely guarded secrets. The Nyquist-Shannon sampling theorem shows that, although we are reconstructing an ($m \times n$ image), there should generally be a more computationally expensive finer grid used that is based upon the detector spacing and image size [69, 90]. The benefits of the finer grid can be seen in Figure 1.5. Oversampling in a continuous domain should never introduce aliasing, which is only introduced when one under-samples without reducing spatial frequencies. Full review of the various FBP techniques will be discussed in Chapter 2.2, and an exploration of the techniques implemented in this document will be discussed in Chapter 3-6.

The class of FBP algorithms are only approximate solutions to the inverse Radon transform and are, as noted, computationally complex. However, linear algebra techniques and iterative techniques are substantially more elaborate. The complexity stems mainly from the forward-projection of rays through pixel locations, seen in Figure 1.6. A lookup table can be used for this process, but the most expensive process is the solving of the linear system. For example, using linear algebra techniques, a reconstruction of a $(4 * (m \times n \ = \ 256))$ image, 736 detectors, fine grid technique that follows Nyquist-Shannon Sampling theorem, and 1152 projections would require a transformation matrix with dimensions, $m = 256{*}4$, $n = 256{*}4$, $k = 736$, and $l = 1152$ projections for one reconstructed image. In this example, $m,$ and n are the rectilinear coordinates of the reconstruction region seen

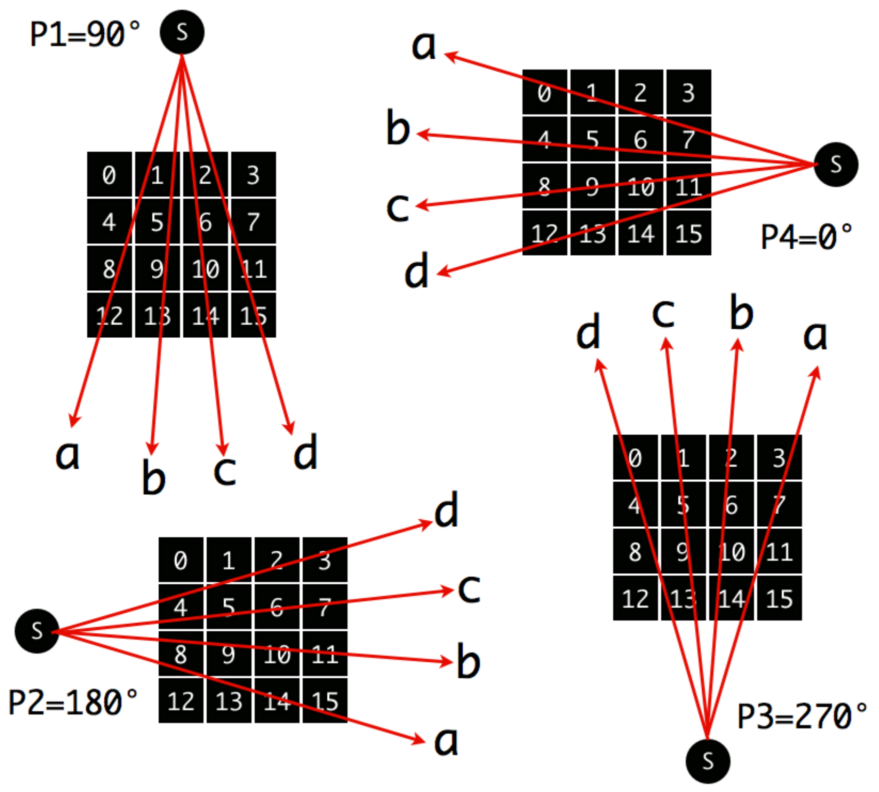


Figure 1.6: Linear method in CT, four projection angles at 90°, 180°, 270°, and 0°. Likewise, four rays cast through each discretized image with intersecting pixels is depicted.

later as the x, and y coordinates of Figure 2.3c. Likewise, Figure 2.3a shows a sinogram that has rectilinear coordinates k, and l. The radial channel detectors (k) are represented by the x coordinate and the projection angular samples (l) are seen as the y coordinate of the sinogram. This does not initially look like a large matrix, but the actual reconstruction requires a matrix with dimensions ($mn \times kl$). Considering 64 to 256 images are usually being reconstructed makes for a considerable amount of computation.

Although the linear methods are accurate, computationally they are expensive, requiring iterative techniques to solve a large linear system. Traditional methods to solve systems of linear equations break down as the problem size increases, as in the problem size previously considered. Solving large sparse matrices is a research topic in itself and is generally not real-time. For this reason, approximate methods have been proposed, these methods are typically referred to as iterative methods in CT. The iterative method tries to satisfy a given condition of accuracy, tolerance value, or set number of iterations. If the condition is not met another iteration is completed, and so forth, until the condition is met. However, the determination of when convergence has been fulfilled is difficult to understand when using real data, as the true solution is not known. These techniques are beginning to be implemented in current CT machines such as Siemens IRIS reconstruction technique [28] or GE ASIR. There has been little use of these iterative methods in clinical-CT, as the results have not been accepted and no guidelines for dose reduction have been established at this time. Likewise, the ability to use reduced projection count is not possible because commercial-CT machines typically have a continuous source and intermittent exposure is not possible. In essence, there are no methods to strobe the continuous source of X-rays.

FBP and Algebraic Reconstruction Techniques (ART) used for reconstruction in CT will be investigated thoroughly in this document. The ability to test these techniques with a highly optimized multi-CPU (64 cores) and multi-GPU (2 Nvidia Quadro 4000) hardware architecture provides a good test platform for the development and analysis of a highly optimized real-time environment for CT reconstruction. The research conducted is unique as we investigate one of the most complicated imaging modalities, fan-beam CT. As there are no standard qualitative or quantitative metrics defined previously we introduce a few. We also are able to show the process of taking real scanner specifics and incorporating those specifics into a reconstruction environment. The results show a enhancement is available to clinical-CT when more robust image quality is needed and real-time manipulation ability must be used to gain a clearer image of the inner anatomy represented by the data acquired. The use of HPC in medical imaging is rather rare because of the associated costs. The closest relation to the techniques proposed in this document is from Partners Research group based in Massachusetts General Hospital and North Eastern University where a NVIDIA Tesla S870 that contains four independent GPUs was used to perform reconstructions [42]. However, their paper is really only a preliminary report and has no specifics on algorithm design or quantitative noise analysis. Likewise, the problem explored is much smaller than what is to be investigated in this thesis. We will show this is the norm in the field as there is very little algorithmic details in publications.

The ultimate goal of the methods established is to reconstruct raw-attenuation values from fan-beam CT in an efficient manner. Specifically, achieving high-quality reconstructions in a timely fashion. This is important for two reasons, mainly: analysis of the techniques can be performed in a more timely fashion and to facilitate the use of low-dose techniques in clinical-CT fast and accurate reconstruction are required.

1.4 Contributions

There are several contributions contained in this thesis. Six formal contributions are included, mainly:

1. *we present a concise literature review of the Radon transform and provide many solutions to the inverse problem when considering current HPC for the reconstruction of fan-beam CT.* Although there are a number of publications concerning the reconstruction of CT, there lacked a good source of information on current implementation techniques and comparison of methods. Therefore, the production of a concise literature review detailing these methods is a substantial contribution to the research community by itself.

2. *a detailed description of the Advanced Reconstruction Environment for Medical Imaging (*AREMI*), which is based on a multi-core and multi-GPU C++ implementation that contains a decoupled visualization and reconstruction routine.* The environment is used to investigate the use of parallel computation on conventionally unmanageable problems in an attempt to make them tractable. The environment is capable of High Dynamic Range (HDR) output, and *can be used to reconstruct raw attenuation data from medical-CT*. We illustrate some of the difficulties encountered with reconstruction from raw-data and *show a 3D version of Siddon's algorithm developed specifically for a GPU or multi-GPU environment*.

3. *we establish qualitative and quantitative metrics for evaluation of reconstruction techniques*. These evaluation techniques analyze the specific frequency space noise, convergence of iterative algorithms, and the importance of projection ordering for iterative methods. *We show an alternative projection ordering technique that is able to achieve faster convergence in our implementation. We establish that different reconstruction regions result in difference levels of convergence and noise*.

4. *we have investigated the use of an open multi-GPU environment that does allow faster reconstructions when compared to our CPU or single GPU implementations.* In this, we present a GPU distributable SART algorithm with an evaluation of different forward and backward-projection techniques. We illustrate the importance of data ordering on the GPU with a approximation to the back-projection algorithm that requires no mutually-exclusive

write-operation. There are no practical implementation details of a multi-GPU environment for fan-beam CT-reconstruction and, because of the computational problems associated with algebraic reconstruction, a complete overview of the framework is given.

5. *we use* AREMI *through a multi-GPU implementation to analyze a computationally complex complete system matrix derivation, $O(n \times m)$, on current hardware and compare to Toft's previous implementation.* This complete system matrix derivation required for ART, from Toft's dissertation, was deemed to be too computationally expensive at the time [97]. We also provide an investigation into automated blocking-strategies for GPU task delegation that can realize substantial improvements in our test environment.

6. *we show a comparison of a DirectX implementation of Convoluted Weight Back-Projection (CWBP) with a serial, parallel, GPU, and multi-GPU CUDA based implementations.* The comparison highlights runtime, noise analysis, and the effects of numerical precision on each technique.

1.5 Applicability

We believe the document compiled is useful to many people in different disciplines, mainly:

1. ***Computing Scientists*** interested in the use of GPUs for real-world problems, as seen in Chapter 3 and Chapter 6. The techniques discussed for GPU implementation considers the numerical precision of technique, efficient GPU programming, blocking strategies, data sharing, and multi-GPU development.

2. ***Engineer, Computing Scientists, and Mathematicians*** requiring to solve large systems of equations as these are very computationally-demanding tasks and using the GPU can reduce runtime. Likewise, those interested in solving similar matrices that are sparse, overdetermined, or underdetermined, as seen Chapter 2.

3. ***Medical Imaging Physicists*** who are interested in reconstruction techniques from raw-attenuation values and some of the difficulties involved, seen in Chapter 3.

4. ***Radiology and Diagnostic Imaging Specialists*** who are interesting in low-dose CT methods. These techniques are detailed in Chapters 3 - 6 and can provide the basis for understanding concepts and complexity of reconstruction in medical-CT.

The ideas implemented have provided a solid foundation for future research in all modalities of medical imaging and are not restricted to CT. The algorithms, and enhancements, show improvements in runtime, image resolution, dose reduction, noise reduction, and artifact removal in our implementation. Likewise, they provide a concise review of techniques and difficulties concerning implementation. The ethical benefits are based upon the reduction of radiation dose in CT, in turn reducing the cases of cancers attributed to CT-studies, while still constructing high quality images for diagnosis. Through the use of the developed environment more computationally demanding filters, reconstruction grids, numerical precision levels, and computationally expensive reconstruction routines can be utilized. Through the use of these more computationally demanding techniques implemented in AREMI, radiation dosage can be reduced, because these methods do not require the same number of projections as typical FBP. The exact reduction in the likelihood of developing cancer, given the use of methods that require less projections, is not well established. However, we believe it is reasonable to postulate there would be a reduction in the occurrences of cancer given the use of these methods as they require less radiation. Eventually, we believe these computationally expensive methods, based on reduced projection count, will be useful to clinical-CT.

1.6 Organization and Reading Guidelines

The thesis is organized in a linear fashion and is intended to be read from beginning to end, as we assume in later chapters the reader has read the previous chapters. The remainder of the thesis is organized as follows: Chapter 2 provides the necessary background to understand the physical process that occurs during a fan-beam CT scan. The focus of this chapter is on the development of conceptual understanding of FBP and algebraic techniques to solve the inverse Radon transform. We then outline the multi-GPU reconstruction environment AREMI in Chapter 3. Next, in Chapter 4, we establish: qualitative and quantitative metrics, along with a method to realize when convergence has been reached, various projection ordering techniques and how they affect convergence, and illustrate the effects of reconstruction region on convergence and noise. Chapter 5 describes methods to align Flying Focal Spot (FFS) modes and we illustrate the importance of various methods for understanding ray and pixel interaction in our model. Chapter 6 represents the some experimental results and provides three unique evaluations. We assess SART techniques in a multi-GPU environment using the information learned from the previous chapters. We then analyze a approach to a complete method in ART, not based on ray-tracing, and compare to a previous complete technique from Toft [97]. Last, we compare various implementations of FBP techniques in an attempt to understand the effects of HPC modality, and numerical precision, on runtime and noise. Chapter 7 provides the conclusion and reiterates some of the results, contributions, and future work.

Chapter 2

Literature Review

2.1 Introduction to Computed Tomography

The advent of computed tomography in the 1970s has revolutionized diagnostic medicine. The ability to depict a good, although approximate, computer image of the inner structure of the human body aids in the ability to diagnose various ailments. Recently it has been brought to the attention of the medical imaging community that the radiation a patient is exposed to, originally thought to be low, has been found to be high enough to cause cancer. For this reason, research into alternative reconstruction techniques that require less radiation exposure could be beneficial. There are a number of factors that are attributed to greater risk of cancer development from CT, namely: the age of the patient, and the total amount of radiation dose. If the patient is young, there is a longer manifestation time for cancer development [6, 7]. Likewise, if total dose reaches even moderate exposure levels, cancer is more evident, as seen in Figure 2.1. There are many supporting publications dealing with increased cancer risk with low dosage of radiation, as similar levels of radiation have been studied for those who work in nuclear facilities and from Japanese cancer rates resulting from the atomic bombs in WWII. For further reading on the affects of radiation and CT on cancer rates please see

	All Solid Cancer		**Leukemia**	
	Females	Males	Females	Males
Excess cases (including non-fatal cases) from exposure to 100 mSv	800 (400–1600)	1300 (690–2500)	100 (30–300)	70 (20–250)
Number of cases in the absence of exposure	455500	36900	830	590
Excess deaths from exposure to 100 mSv	410 (200-830)	610 (300-1200)	70 (20-220)	50 (10-190)
Number of deaths in the absence of exposure	22100	17500	710	530

Figure 2.1: Depicts the estimated cancer cases and deaths expected to result in 100,000 people exposed to 100 mSv of radiation. Where 95% confidence intervals are shown in parentheses. The confidence intervals are important because they include statistical variations, uncertainty in dose reduction results, see [99] for more details.

[78, 2, 9, 76, 75, 99].

The investigation into image quality and noise reduction has led to the ability to lower CT dose and still construct good quality CT images for diagnostic purposes. Techniques, such as iterative reconstruction, linear methods, enhanced display technology, HPC on shared memory computers, and HPC using GPUs all provide the means to enhance image quality and reduce reconstruction time. In the following chapter, we will review current techniques in CT reconstruction and provide a clear picture of the current research and development in computed tomography. Many of these techniques have been implemented, modified, and investigated in the development of this thesis, and are truly instrumental to this investigation.

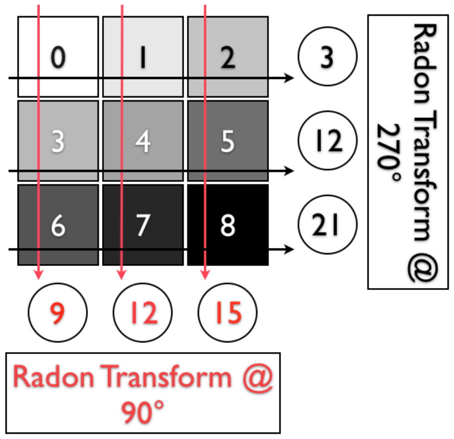


Figure 2.2: Simplified Radon transform for heterogeneous (having different weights) image consisting of nine pixels. The Radon transform is seen involving three rays at 90° and 270°, with resultant sums.

2.2 Problem Description and Representation

Figure 2.2 depicts the basic principle of all CT acquisitions and reconstruction techniques. The figure shows the basic Radon transform with the integration of values encountered by each ray at a specific angle. These values are what constructs a sinogram $(g(l, \theta))$ that has rectilinear coordinates k, and l. Where k, the x dimension of the sinogram, defines the radial detector channels and l, y dimension of the sinogram, defines the angular projections. The inverse of this problem is what we wish to solve and is a computationally demanding task. That is, given the sinogram of measured attenuation values, we wish to reconstruct a 2D model $(f(x, y))$ that accurately represents the measured attenuation values. Where the reconstruction has rectilinear coordinates (n, m).

The reconstruction should consider noise, machine geometry, sensor width, and prior knowledge concerning what is being imaged can all help in the goal of accurate reconstructions.

In this thesis we investigate how the reconstruction problem can be made tractable using parallel processing. That is, given an algorithm designed for parallel computation, how runtime, convergence, and noise can be affected. We investigate the use of the Graphics Processing Unit (GPU) and conventional High-Performance Computing (HPC) in the form of shared-memory parallel computing.

There are generally three forms of reconstruction techniques currently in CT. They are classified into classical and mainstream FBP methods, linear reconstruction tactics and iterative methods, and statistical approaches. This document thoroughly explores the first two, mainly: FBP and iterative reconstruction techniques for linear methods.

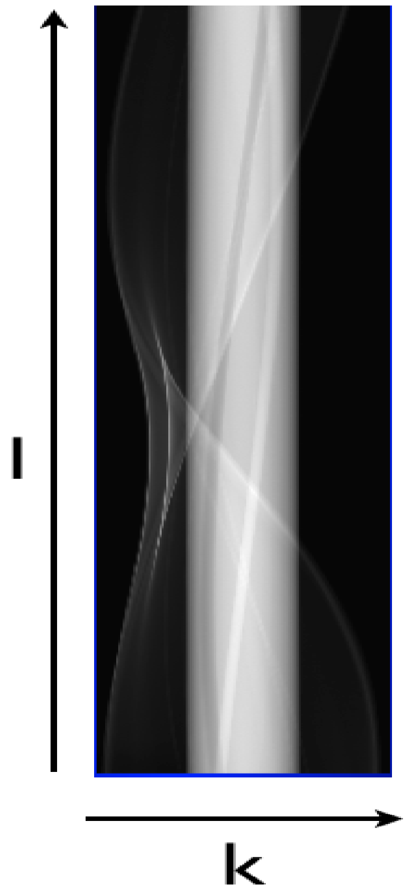
The following sections begin with the basis of reconstruction, the Fourier method and the projection slice theorem, which helps derive FBP and Convolution Weighted Back-Projection (CWBP) algorithm [20]. We discuss the basic methodology behind linear and iterative methods, mainly the forward and back-projection of rays through pixel locations. Although the linear methods are not new, they are becoming popular as they have the ability to better reduce noise than conventional FBP techniques[47, 97]. Previously those techniques were too computationally expensive and not widely used, but with the advent of the GPU they are slowly becoming feasible. One should also note that although iterative techniques are now being researched, and some commercial software from Siemens and the General Electric Company use these methods, the procedures for iteration count and dose reduction are not well defined and generally only being used in research.

This chapter is organized as following,

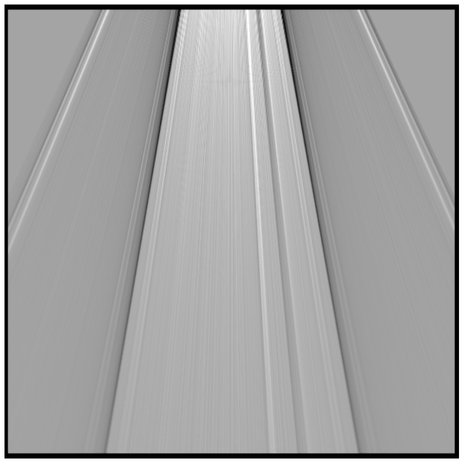
1. complete review of the Radon transform, FBP, and the implementation in parallel environments,

2. a detailed review of linear algebra formulation of Radon transform, Algebraic Reconstruction Technique (ART) [27], Simultaneous Algebraic Reconstruction Technique (SART) [1], and a implementation in parallel environments.

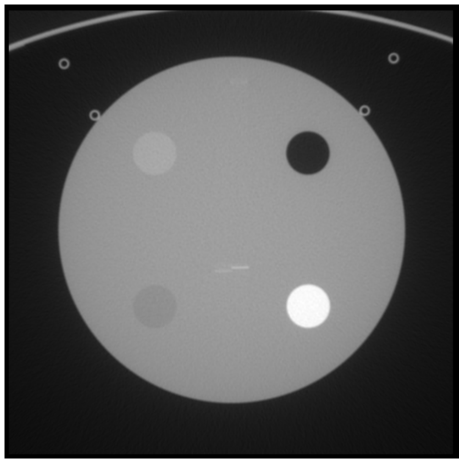
The Radon transform provides the basis for all reconstruction techniques as it represents the data available to be reconstructed into an image [79]. For this reason, we will start with some basic principles of CT, then move onto the Fourier slice theorem, and finally conventional FBP as these techniques provide the core understanding to the final review section on linear algebra and iterative reconstruction methods. The following section provides an introduction to the inner workings and complexity involved with reconstruction in CT from simulated and raw data.



(a) Sinogram of 2.3c



(b) Projection of angle 90° through image



(c) Reconstructed Phantom

Figure 2.3: (2.3a) represents the sinogram of a phantom, known numerically as $p(\gamma, \beta)$, for fan-beam geometry where γ represents the offset in the detector fan or channel and β represents the offset of the X-ray source; (2.3b) single projection of angle 90° through image; (2.3c) complete reconstruction of 1152 projections representing complete sampling of sinogram also known numerically as $f(x, y)$. All data is raw format from Siemens 128 slice Definition Flash+ CT scanner.

2.3 Fundamental Concepts of CT-Reconstruction

Given a 2D slice $f(x, y)$ seen in Figure 2.3c, and given an angle θ relative to a coordinate system in the x-y plane, $g(l, \theta)$ represents the 2D Radon transform, Figure 2.3a, for the slice $f(x, y)$ and can be thought of as representing an individual projection at angle θ of $f(x, y)$, where l is the parameterization of the line, seen in Figure 2.4. In reality, the 2D Radon transform actually represents the attenuation values measured by the sensor array and $f(x, y)$ represents the unknown function or slice one wishes to reconstruct [47, 77, 24]. A sinogram is the pictorial representation of $g(l, \theta)$ with l and θ as the rectilinear coordinates of the image, scene in Figure 2.3a. Simply put, an image representation of the Radon transform of $f(x, y)$.

This measured attenuation values are characterized by the following equations:

$$g(l, \theta) = -\ln\left(\frac{I_d}{I_0}\right), \tag{2.1}$$

$$I_d = \int_0^{E_{max}} S_0(E) \exp\left[-\int_0^d \mu(s; E) \mathrm{d}s\right] \mathrm{d}E, \tag{2.2}$$

where I_d represents the integrated X-ray intensity for a given detector, $S_0(E)$ is the X-ray spectrum, and $\mu(s; E)$ is the linear attenuation coefficient along the line between the X-ray source and a given detector for an incident photon energy E [77]. Likewise, I_0 is characterized as the reference intensity of a given detector. The reference values for the entire detector assembly is usually performed in a calibration step [77] and can fluctuate with the X-ray tube age. For this reason frequent calibration is required for optimal reconstructions.

Although Equation 2.2 is correct, it is a mathematically intractable for CT reconstruction purposes [77]. The intractability stems from the maximal energy integration that is computationally difficult. Referring to the definition of intractability, complexity theory shows that the solution cannot be solved in polynomial time [40]. For this reason, the energy term in Equation 2.2 is replaced by an *effective energy* ($\bar{E}$) term and the outer integral is replaced with the reference detector I_0 seen here:

$$I_d = I_0 \exp\left[-\int_0^d \mu(s; \bar{E}) \mathrm{d}s\right]. \tag{2.3}$$

The goal of the abstraction is simpler computation, this is accomplished by the fact $\bar{E}$, for a given material, will produce the same measured intensity from a monoenergetic source as would be measured using the actual polyenergetic source [77]. For this reason, the importance of calibration of I_0 for each detector is crucial. Commonly, for fan-beam CT machines there is a reference detector that is situated at the end of the detector array in order to guarantee that only air is between the specific detector and source. The reference detector measured value is used along with the pre-calibrated values for the active detectors in order to determine the reference intensity for each detector [77]. Using this method, the machine is able to constantly adjust this ratio to adapt to a dynamic environment of temperature, air density, etc, giving more robust and consistent scans.

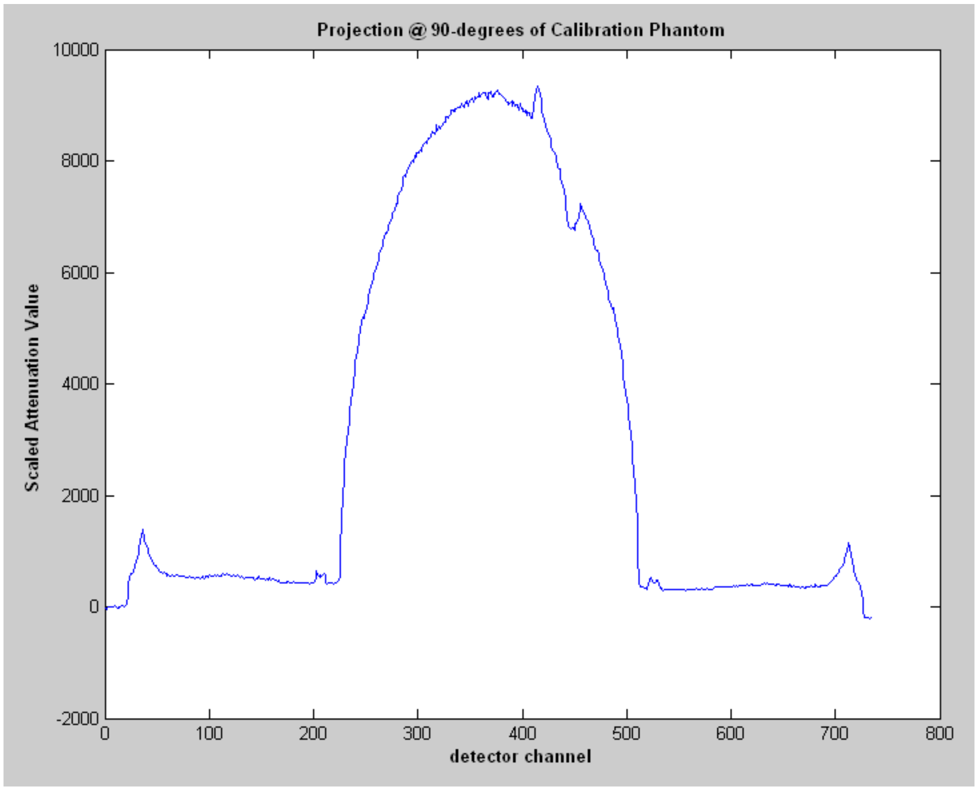


Figure 2.4: Siemens 128 Definition Flash+ raw sensor attenuation values for slice one, 90° source. These values represent individual $g(l, \theta)$ of a physical calibration phantom seen in Figure 2.3c.

The raw attenuated values obtained from a Siemens 128 Slice Definition Flash+ can be seen in Figure 2.4. The data used in much of the research community is typically the simulated Shepp-Logan phantom data, as seen in Figure 2.5 [91]. There are a number of significant differences between simulated data and actual raw data obtained from a machine. For instance, the Radon transform computes line integration as accurately as the computer's precision, and results in a near perfect plot with relatively smooth representation of the projection as seen in Figure 2.6. This plot depicts, on the outer sides of the graph, the air portion of the scan, black outside border. A perfect representation of this region is seen with a flat zero for this section depicted by detector channels 0-190 and 550-736 of Figure 2.6. In comparison, referring to Figure 2.4, one can see evidence in channel locations 85-195 and 540-690 a noisy representation of air space from the physical scan of the calibration phantom seen reconstructed in Figure 2.3c. This creates a number of challenges because the data is relatively noisy when reconstructing from actual sensor data.

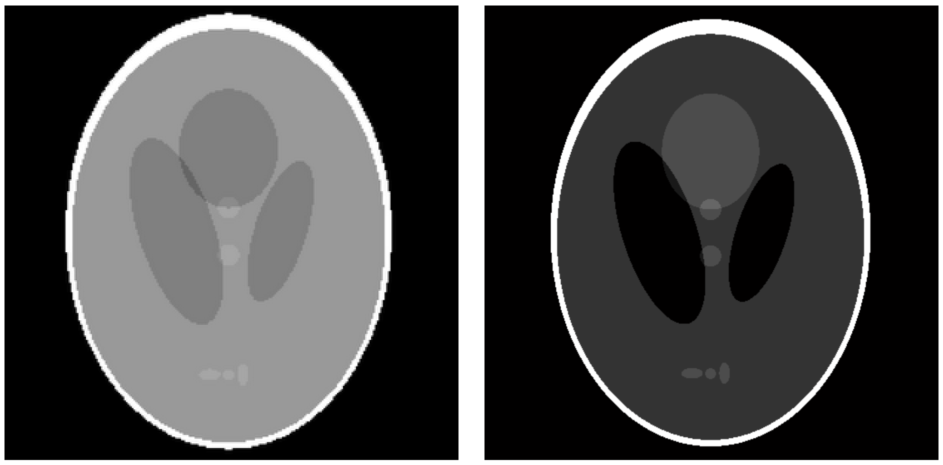


Figure 2.5: Left Image: Typical simulated Shepp-Logan Phantom usually defined according to hounsfield units, Right: Simulated Shepp-Logan data in our simulator. The contrast is slightly different given our ability to illustrate the ability of AREMI to output HDR.

2.4 Projection or Fourier Slice Theorem

The Fourier Slice theorem is instrumental to the development of both 3D helical reconstruction and the simpler 2D reconstruction in CT. The principle is derived from Bracewell's, originally in 1956, for radio astronomy problems, but best captured in his book on Fourier transforms (FT) and the applicability of the transform in many domains [4]. Conceptually, the transform states that if a two-dimensional function $f(x, y)$ is projected onto a line (one-dimensional space) a Fourier transform of that projection is equivalent to doing a 2D Fourier transform of the original function $f(x, y)$ and taking a slice in the Fourier domain at the origin and parallel to the projection line.

The 1D Fourier transform of a projection is equivalent to a line going through the origin of the 2D Fourier transform of the complete image at a specific angle corresponding to the projection angle. As an example, the left image from Figure 2.7, represents the projection of the data onto a line with the source at $135°$. Applying the 1D Fourier transform to the raw data, seen as the blue graph for this projection, results in the equivalence relation with the 2D Fourier transform of the complete function and the depicted red line through the origin that represents the identical data in the 2D domain. This relation is referred to as the Central Section or Projection Slice theorem. Reviewing the mathematical proof is useful as it develops the formal Projection Slice theorem we will use for the formulation of the conventional FBP, originally developed by Bracewell and Riddle [5] and later separately developed by Ramanchandran and Lakshminarayanan [81] with additional

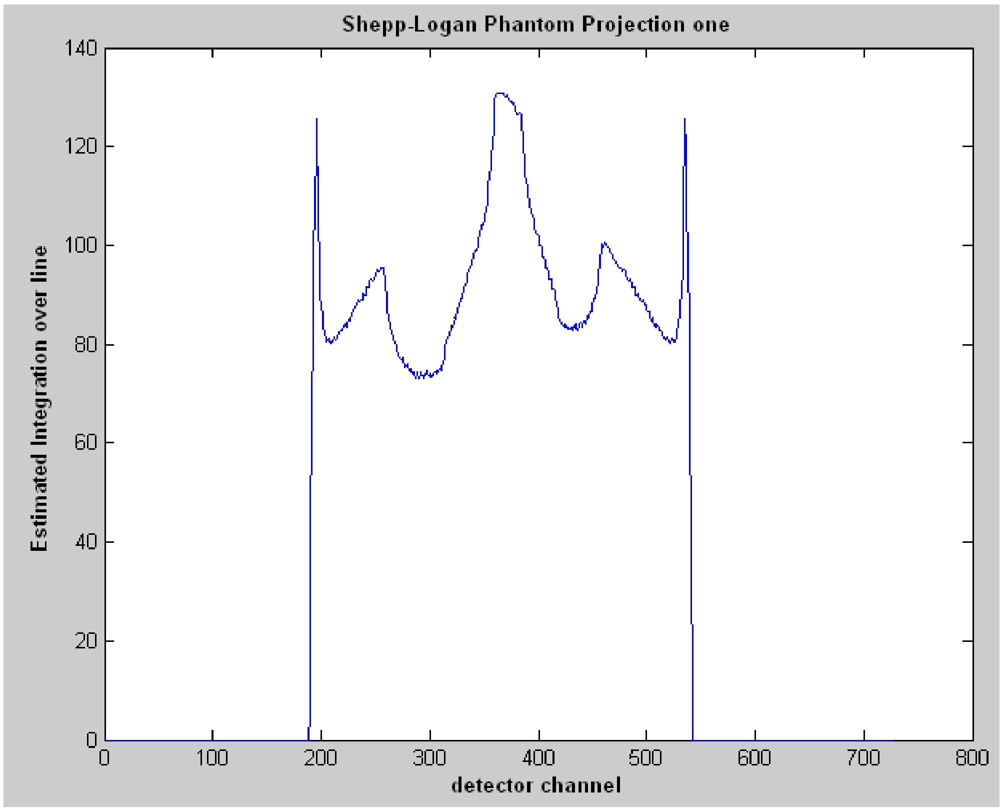


Figure 2.6: Radon transform of simulated Shepp-Logan Phantom analogous to the beam intensity integration along a line from the X-ray source to the detector in typical physical CT. Notice relatively smooth function with perfect zero value for detector channels 0-190 and 550-736, which represent airspace in the phantom model.

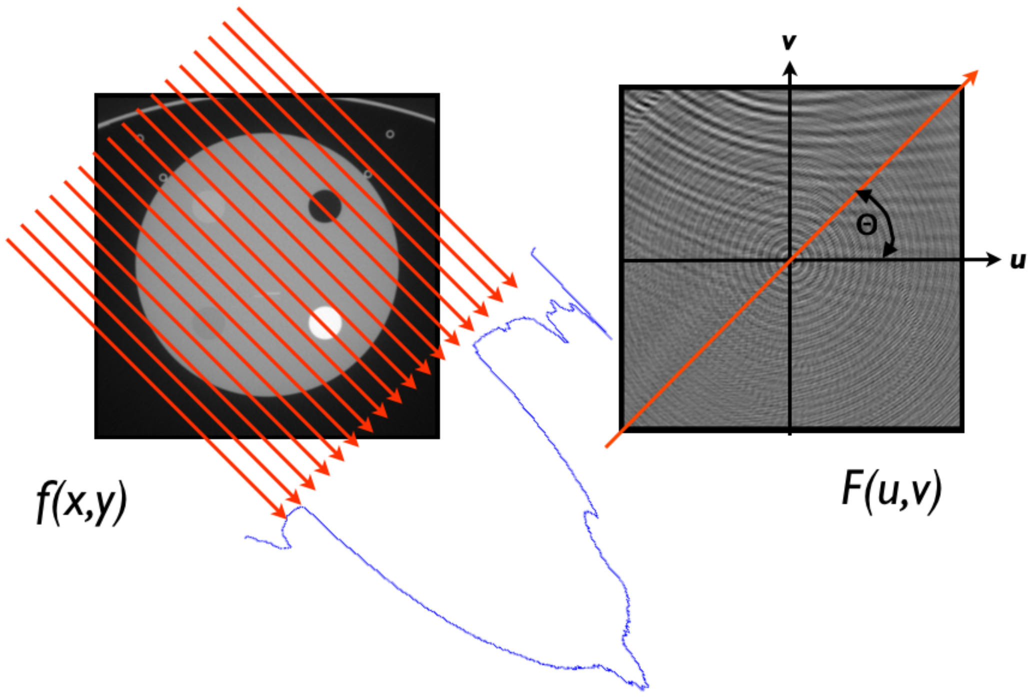


Figure 2.7: The left image represents the 1D transform and the result $g(l, \theta)$ at angle θ and the corresponding 2D Fourier transform of $f(x, y)$ and line at angle θ representing equal slice. Images generated from actual raw data at $135°$ and resulting 2D Fourier transform of $f(x, y)$ image.

contributions such as CWBP. The equation is:

$$G(\varrho,\theta) = F_{1D}\{g(l,\theta)\}, \tag{2.4}$$

$$= \int_{-\infty}^{\infty} g(l,\theta)e^{-j2\pi\varrho l}\,\mathrm{d}l, \tag{2.5}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - l)e^{-j2\pi\varrho l}\,\mathrm{d}x\mathrm{d}y\mathrm{d}l, \tag{2.6}$$

where, ϱ represents the spatial frequency and F_{1D} is the 1D Fourier components along l.

We can pull the integration of $f(x, y)$ out as it is not dependent on l, the result is:

$$G(\varrho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\int_{-\infty}^{\infty} \delta(x\cos\theta + y\sin\theta - l)e^{-j2\pi\varrho l}\,\mathrm{d}l\mathrm{d}x\mathrm{d}y, \tag{2.7}$$

This step is accomplished using the shifting invariant property of the delta function, refer to [77] for details. Equation 2.7 is now simplified to:

$$G(\varrho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi\varrho(x\cos\theta + y\sin\theta)}\,\mathrm{d}x\mathrm{d}y. \tag{2.8}$$

The final Equation is similar to the 2D Fourier transform of $f(x, y)$:

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(xu+yv)}\,\mathrm{d}x\mathrm{d}y. \tag{2.9}$$

Let $u = \varrho\cos\theta$, $v = \varrho\sin\theta$, and the equivalency equation is:

$$G(\varrho,\theta) = F(\varrho\cos\theta, \varrho\sin\theta). \tag{2.10}$$

2.5 Fourier Reconstruction Method

The Projection Slice theorem is instrumental in the development of the Fourier method for reconstruction and is the base of FBP and CWBP algorithm. The reconstruction process could take place directly now with sampling of the Fourier space values in either form, 1D or 2D, and creating a 2D Fourier based image, having decreased computational complexity of $O(n \times m \log n)$. The final step consists of taking the inverse 2D Fourier transform for this image to construct the image space representation. However, most of the literature shows the difficulty involved with interpolating frequency based polar data onto the Cartesian grid of the image of the domain [47, 77, 24]. This observation is evident when analyzing the 2D Fourier image on the right of Figure 2.7. Sampling the single projection seen along the line of the 2D spectrum image is expensive as it would require non-contiguous reads of memory, i.e. row major order or column major can be made coalesced and contiguous, and interpolating values from polar to cartesian grids would not be row or column major reads. Obviously the storage could be manipulated to provide contiguous reads for this type of memory fetch, it would, nonetheless, require those non-contiguous memory read to form this data structure as the raw data from the machine is in row major order. In general, sampling row major or column major lines is more effective at providing a high rate of cache hits in main memory and/or GPU memories as the sensor data acquired is generally stored in this manner. Likewise, as the sampling is non-uniform, and actually has decreased sampling as one moves from the centre of acquisition out, numerical error and interpolation error increase resulting in numerical stability issues. For this reason, there is an increased amount of error due to the 2D interpolation in the calculation when moving from low frequency to high frequency samples [47, 97, 102, 103]. Although mathematically the technique is sound in a continuous domain, i.e. continuous sampling over the domain, in practice, the technique suffers from a number of issues that prevent the practical applicability in a computational environment, see Toft's PhD dissertation for more details [97].

2.6 Filtered Back-Projection (FBP)

The basic back-projection algorithm can easily be inferred from the Projection Slice theorem and has increased computational complexity of $O(n \times m \times l)$ when compared to the Fourier method but does not suffer from the numerical stability issues found in the Fourier method. The back-projection operation is known as the most expensive operation in the reconstruction process. As every projection is literally smeared over the entire image space as depicted in Figure 2.3b where the single $\beta = 90°$ projection is sampled by all pixels in the image. Given that, typically, 1152 projections need to be applied over the entire image space accounts for significant computational costs for just one slice and reconstructions range from 12 slices to 256. We shall now construct the FBP method, beginning with the inverse FT of $F(u, v)$ in respect to polar coordinates, we have:

$$f(x,y) = \int_0^{2\pi}\int_0^{\infty} F(\varrho\cos\theta, \varrho\sin\theta) e^{j2\pi\varrho(x\cos\theta + y\sin\theta)} \varrho \, \mathrm{d}\varrho \mathrm{d}\theta, \tag{2.11}$$

incorporating the Projection Slice theorem equivalency equation derived in Equation 2.10 and substituting accordingly, we derive:

$$f(x,y) = \int_0^{2\pi}\int_0^{\infty} G(\varrho, \theta) e^{j2\pi\varrho(x\cos\theta + y\sin\theta)} \varrho \, \mathrm{d}\varrho \mathrm{d}\theta. \tag{2.12}$$

One can reduce the outer integral to $180°$ as the symmetric properties of the scan would be duplicated if we covered the whole $360°$ of rotation of the source angle β in the parallel beam reconstruction case. However, for the process of developing a CWBP algorithm for fan-beam reconstruction, originally from Lakshminarayanan [52], we will keep the $360°$ of rotation as we require it for a complete reconstruction using fan-beam sensor data. One can also limit the inner integration of ϱ to only the image space we are interested in, eventually yielding the inner integration between $\pm\,\gamma$ that represents the sensor array and the maximal radius of the focal spot circle produced by the scanner. One can limit this value even further in the implementation stage by restricting the radius to an even smaller area of interest, one designates this restriction by $\pm\,\tau$ on the inner integration as in:

$$f(x,y) = \int_{0}^{2\pi} \int_{-\gamma+\tau}^{+\gamma-\tau} G(\varrho,\theta) e^{j2\pi\varrho(x\cos\theta + y\sin\theta)} \varrho \, \mathrm{d}\varrho \mathrm{d}\theta. \tag{2.13}$$

The integration over ϱ of $(x\cos\theta + y\sin\theta)$ is a constant term and can be be substituted with $l = (x\cos\theta + y\sin\theta)$ showing that the inner integral is simply the 1D Fourier transform, as depicted in Equations 2.4 - 2.5, and expressed by:

$$f(x,y) = \int_{0}^{2\pi} \left(\int_{-\gamma+\tau}^{+\gamma-\tau} G(\varrho,\theta) e^{j2\pi\varrho l} \varrho \, \mathrm{d}\varrho \right) \mathrm{d}\theta. \tag{2.14}$$

Although Equation 2.14 represents back-projection, and a significant improvement over the previously mentioned Fourier reconstruction technique, it is missing a important filtering value, namely, $|\varrho|$. The filter is required because of over sampling of low-frequencies, Figure 2.8 shows the effect when no filter is used on the projection data.

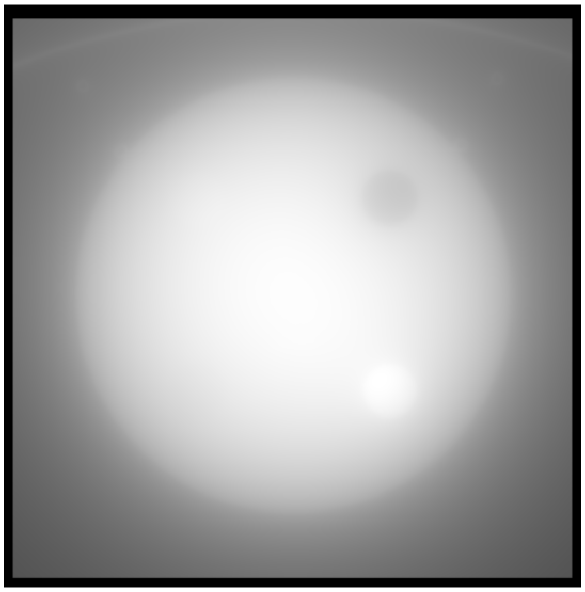


Figure 2.8: Depicts conventional FBP reconstruction with no ramp filter, still contains Hamming filter. The reconstruction with ramp filter can be seen in Figure 2.3c.

Adding a filter $|\varrho|$ to Equation 2.14, we have:

$$f(x,y)=\int\limits_{0}^{2\pi}\left(\int\limits_{-\gamma+\tau}^{+\gamma-\tau}|\varrho|G(\varrho,\theta)e^{j2\pi\varrho l}\varrho\,\mathrm{d}\varrho\right)\mathrm{d}\theta. \tag{2.15}$$

The results of the filtering can plainly be seen in Figure 2.3c versus the unfiltered reconstruction in Figure 2.8. The basic insight into filtering in the reconstruction process is twofold. First, the use of the ramp filter ($|\varrho|$) in Equation 2.15 is used to deal with the higher sampling at lower frequencies. Therefore, one needs to add appropriate weight to the high frequency samples or decrease the weight of the lower frequency samples. Second, filters used on the raw data, in addition to the ramp filter, have the ability to preserve edge features, prevent aliasing, apply different regularization parameters, and more depending on the required reconstruction. The foundations for the equations in this section were based on those found in [47, 77, 24].

2.7 Special Consideration – Fourier Transform

In this section, we will review the effects of the 1D FT from the spatial domain to frequency domain and the inverse FT from frequency domain to spatial domain, as we will be using these techniques to facilitate more efficient computation in Fourier frequency domain, which has complexity $O(k \log k)$, as the filtering operation is simply a point-wise multiplication, rather than a more expensive symmetric convolution in the spatial domain having complexity $O(k^2)$. There are some notable issues one should account for if one wants to transform a spatial filter or projection data to the frequency domain. In general, if a 1D filter is constructed in the spatial domain and has a sampling size n, transforming the filter to the frequency domain, for the more computationally efficient point-wise multiplication, can have side effects. Mainly, due to the finite space available during the transformation of 1D filter of size n, there can be a loss of high frequency information during the transform, as the intervals in the frequency domain are smaller than that in the spatial domain. To decrease these effects, the literature often suggests to pad the data with zeros up to the next power of two, given the original data was of size n [47, 77, 24]. However, experimentally padding with zeros tends to increase high frequency drop-offs where the padding begins. There are mainly three successful techniques found to alleviate this problem in raw CT reconstruction. One can choose to repeat the last value, as this does not create any high frequency anomalies. Second, one can linearly decrease the values from the boundary points of the measured sensor values to zero and then zero pad the data. This also alleviates the problem of adding noise to the measured data, as the outer boundary of the sensor array typically measures air values and provides no meaningful information for reconstruction. Last, the most effective method is to apply, when possible, the filter in the frequency domain directly.

Likewise, transformation of projection data to and from the frequency domain should be handled by either filling the last measured value up to the next power of two of the original size or linearly decreasing from the last value to zero and appending zeros.

For further discussion on this topic please refer to [47, 102, 97] as a more comprehensive review is conducted.

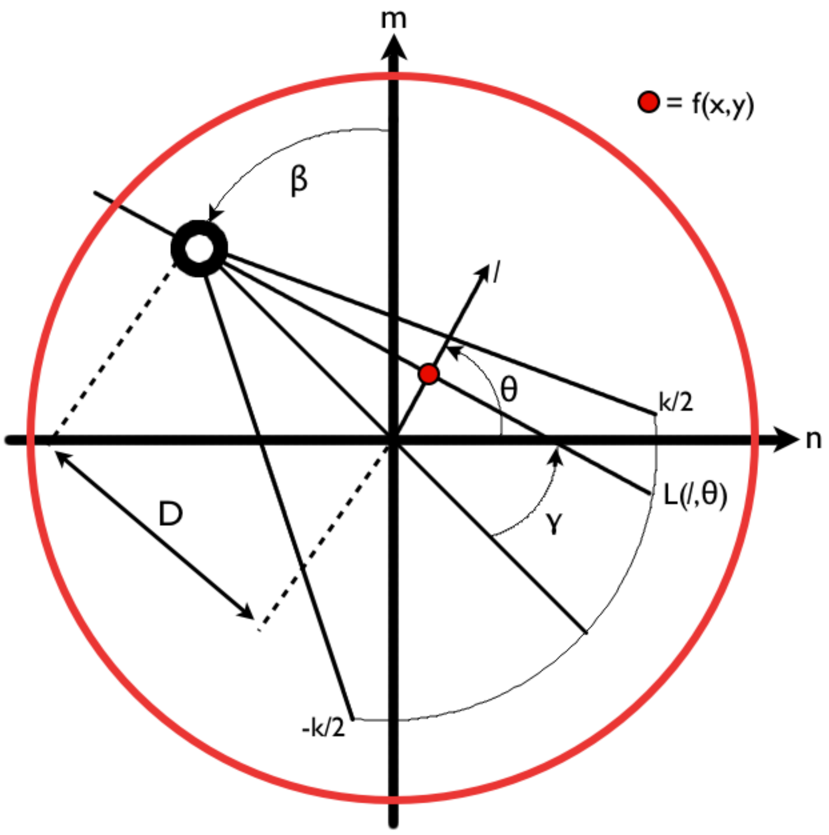


Figure 2.9: Representation of basic fan-beam CT geometry. Where γ = angle index of sensor array for given red pixel location, θ = angle of pixel from origin, D = distance from isocenter to source, β = angle of source, and l = distance from isocenter to pixel location.

2.8 Fan-Beam Convolution Weighted Back-projection (CWBP)

Equation 2.15 was chosen as it makes the transition to CWBP, in the Fan-Beam case, easier as we limit the bounds of the inner integral to only those required for the reconstruction from actual discrete data where only a finite amount of measurements are seen. The Equation is slightly different than our previous derivation. Mainly, the equations are defined with the geometry of fan-beam machines accounted for, where the pixel locations $f(x,y)$ of the reconstruction image is defined as:

$$f(x,y) = \frac{1}{2}\int_{0}^{2\pi}\int_{-\gamma+\tau}^{+\gamma-\tau} p(\gamma,\beta)c(D'\sin[\gamma'-\gamma])D\cos\gamma \,\mathrm{d}\gamma\mathrm{d}\beta, \tag{2.16}$$

where, γ is the valid discrete locations on the sensor array, β is the discrete measured angles as seen in Figure 2.9, D is the distance from iso-centre to the source, D' is the distance from voxel or pixel location to source, and l is the euclidean distance from isocenter to the given pixel location. Let $\pm\tau$ be the sensor values one wishes to omit given the absolute values of x and y needed in the reconstruction; there is no benefit in including these values as they are never sampled given the angles that are visited during the reconstruction. The maximal reconstruction size is determined by the effective area of the sensor array. Secondly, remembering the definition of $g(l,\theta)$ from Equation 2.1, one can change from parallel beam to fan-beam geometry easily. Specifically, $\theta = \beta + \gamma$ and $l = D\ \sin\gamma$, where D is the distance from isocenter to the source. The Jacobian of the coordinate transformation is simply $D\cos\gamma$ and the fan-beam formula is now given by Equation 2.16. The third instrumental change from Equation 2.15 in Equation 2.16 is the transformation of the convolution integral $C(\star)$. The convolution theorem states that the product of the Fourier transform of two signals $f(x,y)$ and $g(x,y)$, namely $F(u,v)$ and $G(u,v)$, is equal to the Fourier transform of the convolution of the two signals in the spatial domain $\mathscr{F}_{2D}(f(x,y)\star g(x,y)) = F(u,v)G(u,v)$ [77, 24]. The computational attractiveness of this theorem is that the high efficiency of the Fourier transform can be used to speed up the required convolution by a simple point-wise multiplication in the frequency domain rather than on expensive convolution in the spatial domain. The following Equations will lead to separation of the back-projection process from the convolution process $c(\gamma)$:

$$c(D'\sin\gamma) = \left(\frac{\gamma}{D'\sin\gamma}\right)^2 c(\gamma), \tag{2.17}$$

$$c_f(\gamma) = \frac{1}{2}\left(\frac{\gamma}{\sin\gamma}\right)^2 c(\gamma), \tag{2.18}$$

where D' represents the distance of the pixel to the source.

Using Equation 2.17 and 2.18, we are able to isolate the convolution into one efficient step in the Fourier domain by:

$$f(x,y) = \int\limits_0^{2\pi} \frac{1}{(D')^2} \int\limits_{-\gamma+\tau}^{+\gamma-\tau} \tilde{p}(\gamma,\beta)c_f(\gamma-\gamma')\,\mathrm{d}\gamma\,\mathrm{d}\beta, \tag{2.19}$$

where $s = \tilde{p}(\gamma,\beta) = \cos\gamma\, D\, p(\gamma,\beta)$ and $t = c_f(\gamma-\gamma')$.

Computationally this equation is too expensive as the convolution requires a $O(k^2)$ algorithm versus the FFT method with computational complexity $O(k\log k)$. Therefore, we manipulate this equation to bring a more efficient computational model. Since the convolution in the spatial domain is expensive even though the convolution is symmetric, as the filter is a reflection of itself, one can reduce it to a simpler correlation operation. The process is still too costly considering the amount of projections. For this reason, we will exploit the previously defined convolution theorem and perform the convolution in the frequency domain with a simple point-wise multiplication. Mainly, $S = \mathscr{F}_{1D}(s)$, $T = \mathscr{F}_{1D}(t)$, and $q = \mathscr{F}_{1D}^{-1}[S\,D]$ being the inverse Fourier transform of the point-wise multiplication of S and D. This process eliminates the inner integral and we are left with a relatively simple equation:

$$f(x,y) = \int\limits_{0}^{2\pi} \frac{1}{(D')^2}\, q(\gamma,\beta)\, \mathrm{d}\beta. \tag{2.20}$$

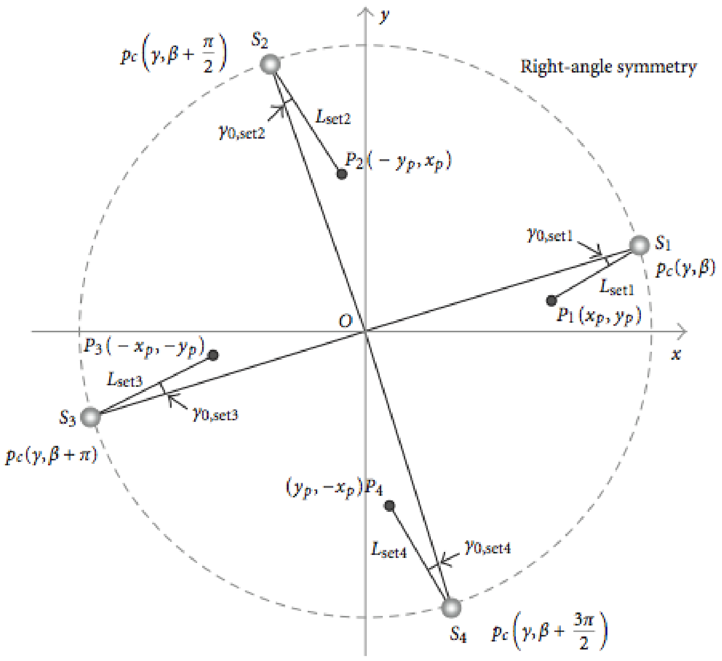
The equation is called Convolution Weighted Back-Projection, and is appropriate as D' is defined as the distance of the pixel location from the iso-centre and, hence, the weighting is inversely proportional, as the distance from iso-centre increases, seen in Equation 2.20. Likewise, the convolution $q(\gamma,\beta)$ is applied as coarser sampling is seen in fan-beam data in higher frequency areas and is filtered appropriately using a standard ramp filter. Other frequency-based filters can be applied at this point in the frequency domain such as the Hamming filter that is often used to smooth out aliasing problem and is similar to a cosine filter [18]. Likewise, any frequency based filter can be used, even those that are not appropriately defined in the spatial domains.

2.9 Variations of FBP Algorithms

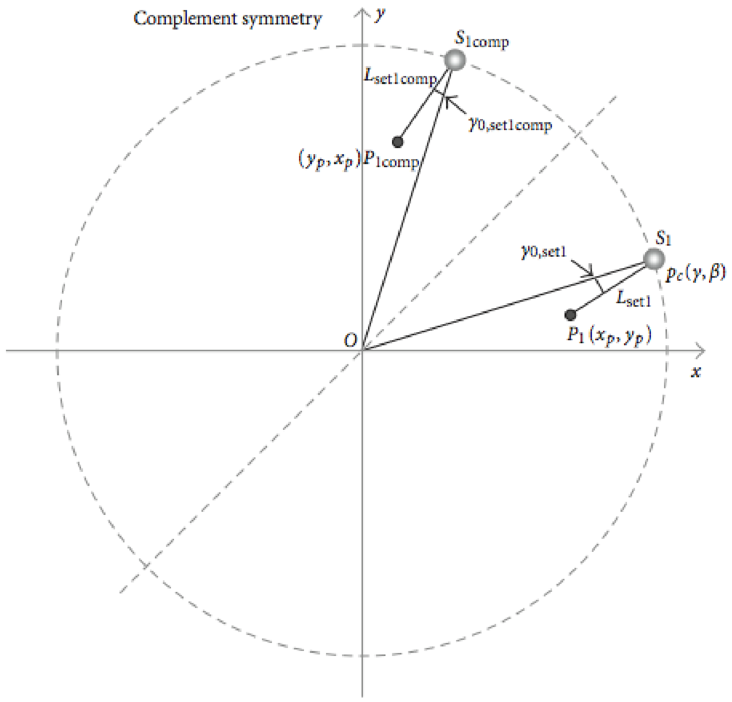
As we have seen, there are many different algorithms to compute the inverse Radon transform. Likewise, there are many commercial algorithmic variations of FBP, but they generally are not published works and cannot be implemented for comparison basis. These techniques are regarded as proprietary techniques in CT reconstruction and will only be lightly discussed in this thesis. For this reason, this review focuses on published works that give some algorithmic insight into the technique.

2.10 Kernels used in FBP Algorithms

The filtering process in FBP is one of the most important stages in reconstruction. Generally, most research is spent in the image domain, where the DICOM image is modified using available commercial image-space kernels, these techniques take place after the image is reconstructed with a conventional ramp-filter coupled with a Hamming filter to give a smoother image with less aliasing. However, there is a lot of potential if one uses novel filters in the reconstruction stage on the raw attenuation values in the frequency or spatial domains. Shtok *et al.* 2008 publications depicts the use of an adaptive, spatially-variant linear filter instead of the typical ramp filter [92]. The technique uses three novel kernels, specifically the author argues that the use of neighboring projections kernels, angle dependence kernels, and distant dependent convolution kernels. Shtok *et al.* accounts for some of the inadequacies of the discrete nature of scanning, as the FBP is built in a continuous domain and apparent problems are seen when implemented in discrete settings. Shtok *et al.* was



(a) Right-angle symmetry



(b) Complement symmetry

Figure 2.10: (2.10a) depicts typical cartesian similarities of angle γ and length L for four different pixel locations. Namely, $P_1(x_p, y_p)$, $P_2(-y_p, x_p)$, $P_3(-x_p, -y_p)$, $P_4(y_p, -x_p)$. (2.10b) shows that the two pixel locations $P_1(x_p, y_p)$ and $P_{comp}(y_p, x_p)$ share the same L value and the γ, the sign of γ is only negative [113].

able to reduce the Signal to Noise Ratio (SNR) ratio by approximately 7 dB. The whole technique is interesting and novel in particular the training stage on desired family of attenuation images is always difficult as, in a clinical setting patients differ widely. The goals of this thesis is to use parallel processing to enhance image quality and runtime for raw data reconstruction not to explore new filters.

2.11 Multi-core Parallel Reconstruction

Zeng *et al.* [113], discuss a number of computational improvements in CWBP. The techniques touch upon four main points. First, two points exploit symmetric properties of cartesian coordinates seen in Figure 2.10. The next enhancement uses a multi-core algorithm to distribute work among processors. Last, a proprietary Intel optimized compiler for C++ is used to improve the speed. Using the parallel computation, symmetric properties, and the optimized compiler the technique is able to accomplish a 40x speedup over conventional CWBP. Although impressive, the technique suffers from many limitations.

The symmetry exploited in this algorithm is impressive as approximately only $1/8$ of the calculations are needed as the values for L and γ are reused for different pixel locations. The complexity of the computation of L and γ can be seen in the following equations:

$$\begin{aligned}
L_{set1} &= \sqrt{D^2 + (x_p^2 + y_p^2) - 2\,Dr\,\cos(\beta - \tan^{-1}(x_p, y_p))},\\
&= \sqrt{D^2 + (x_p^2 + y_p^2) - 2\,Dr\,\cos(\beta + \frac{\pi}{2} - \tan^{-1}(y_p, -x_p))},\\
&= L_{set2},\\
&= \sqrt{D^2 + (x_p^2 + y_p^2) - 2\,Dr\,\cos(\beta + \pi - \tan^{-1}(-x_p, -y_p))},\\
&= L_{set3},\\
&= \sqrt{D^2 + (x_p^2 + y_p^2) - 2\,Dr\,\cos(\beta + \frac{3\pi}{2} - \tan^{-1}(-y_p, x_p))},\\
&= L_{set4}.
\end{aligned} \tag{2.21}$$

$$\begin{aligned}\gamma_{0,set1} &= \sin^{-1}\left(\frac{r\,\sin(\tan^{-1}(x_p,y_p)-\beta)}{L_{set1}}\right),\\ &= \sin^{-1}\left(\frac{r\,\sin(\tan^{-1}(-y_p,x_p)-\beta-\frac{\pi}{2})}{L_{set1}}\right),\\ &= \gamma_{0,set2},\\ &= \sin^{-1}\left(\frac{r\,\sin(\tan^{-1}(-x_p,y_p)-\beta-\pi)}{L_{set1}}\right),\\ &= \gamma_{0,set3},\\ &= \sin^{-1}\left(\frac{r\,\sin(\tan^{-1}(-y_p,x_p)-\beta-\frac{3\pi}{2})}{L_{set1}}\right),\\ &= \gamma_{0,set4}.\end{aligned} \tag{2.22}$$

There are computationally expensive square-roots, divisions, arcsin, arctan operations. Figure 2.10 shows the right angle symmetry for the four pixel locations are equivalent. Likewise, the complement symmetry is equivalent for depicted pixels when the sign of γ is reversed. The derivations of these equivalences can be seen in Equations 2.21 - 2.22. Although the technique is theoretically valid, in order to make the technique computationally efficient one would require the data to be reordered as the technique would require coalesced reads to be efficient, as non-coalesced reads are computationally inefficient. In general, this technique reads from different projections, β-angles, that are $45°$ and $90°$ apart in memory and would require the data to be reordered to provide efficient reading of data. Likewise, Zeng's paper assumes the projection data is completely available, they perform a costly data reordering routine to improve cache hits when reading from different projections (β-angles). This routine is expensive and is not practical for a real-world system where the data is acquired one projection at a time. A fair argument for this technique, that was not mentioned in the paper, would be post-acquisition filtering where knowledge of the complete system is necessary to make an educated guess concerning noise, artifacts, or any filters that requires complete knowledge of the data before processing. The paper also makes note of a required re-binning of helical data to single slice data. This process is also costly in terms of runtime and it is a rough approximation. Considering the discrete nature of the measured data and the fact that it is already an approximation to the continuous system, further approximation is generally not desired.

Secondly, using specialized proprietary compilers is expensive and typically reserved for industry application as one cannot really claim algorithmic enhancement or speedups based on using a more effective compiler for a specific hardware. The more substantial claim would be concerning the compiler and the speedups based on the compiler.

Last, the paper does use the classical Shepp-Logan phantom, seen in Figure 2.5. The research community typically does use the Shepp-Logan phantom as real attenuation values are difficult to obtain; however, the goal of research is to produce an effective technique to solve real problems, and using the Shepp-Logan phantom does not reflect real-world problems, such as those previously discussed concerning measured versus actual air values, seen in Figure 2.4.

Although Zeng *et al.* provide novel enhancements to the CWBP reconstruction algorithm, the most important contributions are the realization that data reordering is not the optimal solution. Second, the understanding that, with the addition of more processing cores, a substantial speedup can be achieved.

2.12 FBP Parallel Techniques

There are generally three forms of parallel techniques used in FBP algorithms. There are those based on multi-core parallel approaches that are designed for the CPU, such as those from Zeng *et al.* discussed previously [113]. The second parallel strategy is the graphics processing unit (GPU) as a parallel processor as seen in Scherl *et al.*'s work from 2007 [84]. Finally, there are Field Programmable Gate Arrays (FPGA) based parallel techniques. For this reason, we will first discuss parallel approaches designed for CPUs and then move on to parallel techniques formulated for GPUs and then we will give a brief overview of methods that use FPGAs.

2.12.1 CPU-based Parallel Techniques for FBP

The FBP code is parallelizable, as the outer integral in Equation 2.20 is the summation of rays and can be distributed easily over many processors. The allotment of work can be influenced by a variety of design choices, such as work queues, automated optimized blocking strategies, and divide and conquer techniques. Work queues are favorable on large or diverse systems where there is often a variety of processors working and no optimal blocking strategy is used to control the workload per process. Each node may have a different processor and/or cache sizes and could be competing for resources with other local processes. The work queue scenario usually employs a management routine that controls the allocation of available work. When a thread is launched it requests work from the queue, if work is available the management thread issues a workload; if not, the thread waits for work or is possibly terminated if no work exists. Large, diverse work schedulers often have the choice to issue the same work to multiple systems, in the hope the work will be completed sooner, as in Google's commodity computing mentality [23, 14]. System discovery techniques can be used on a system sharing the same type of processors. This technique often finds a near-optimal solution to workload distribution, given the available memory, cache, and registers on the system, and is seldom competing for resources with other processes. Divide and conquer techniques are usually the simplest, as these techniques look at how much work is required and divide that work evenly over the number of available workers. All techniques are feasible for the FBP, as the work required to compute the integrals is easily distributed.

Zeng *et al.* uses the simpler divide and conquer technique as their method of choice for parallelism [113]. They choose to assign specific threads a range of values for the outer integral seen in

Equation 2.20. The method was effective and resulted in a runtime speedup of approximately 1.89x. The speedup is reasonable, but more interesting would to test concurrent execution on greater than two processors, as this would provide more information on the level of parallelization available for their algorithm. This thesis provides more extensive testing with processor count greater than two.

2.12.2 GPU Based Parallel Techniques for FBP

The work from Vlček provides an algorithm for the inverse Radon transform on the GPU [100]. As mentioned previously, the Radon transform is the primary method used for forward and backward projection in CT-reconstruction. The inverse Radon transform is specifically the method used for development of CWBP algorithms. Vlček publication used the more primitive hardware shaders to carry out the inverse Radon transform. Specifically, the technique used the CPU to perform the 1D Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) filtering operations, as the convolution is too computationally expensive to perform serially on a CPU. Likewise Vlček, performed the outer integration (FBP) using a hardware shader. The hardware shader was limited in the size of execution code, dynamic loops were impossible to program, and video memory was limited. Nonetheless, this technique was able to accomplish a 29x speedup over a single CPU version of the inverse Radon transform on a matrix of 1024 elements [100]. The technique provided the foundation and future development of the field and proved that a substantial speedup was possible for FBP if video memory size increased and dynamic loops were possible. The constant writing of data to the same texture output is expensive in hardware shaders and a considerable improvement to this method is now available through the use of CUDA, where writing to the same location, as in the outer integration or summation process' is made much more efficient. The publication lacked usability as the FFT and IFFT code was performed on the CPU and was costly.

Vlček's 2004 work on the inverse Radon transform performed on the GPU naturally led to the publication from Vlček in 2005 [101] that detailed the programming of FBP, FFT, and IFFT on the GPU using hardware shaders. The main difference over the previous publication was the addition of the FFT and IFFT filtering on the GPU instead of the traditional CPU implementations. Overall, the method, almost completely implemented on the primitive GPU, was 4x faster than the serial execution on a CPU.

The significant oversight of the two publications from Vlček [100, 101] was a discussion concerning numerical precision of the GPU results when compared to the CPU results. The hardware based shaders, at the time, used glorified hardware accelerated mathematical operations and were not following the IEEE-754 standards. Luckily, GPUs have progressed over time and the use of non-fastmath can be specified on the GPU for numerically sensitive computation. User studies would need to be conducted to analyze the effect on image quality with radiologists when differences, based on numerical precision, are too large. Difference maps are important to the evaluation of image quality as they show concretely where the distinction, based on numerical precision of the computation, is depicted for image processing. However, they are not that useful as a qualitative metric unless there is a known image.

Scherl *et al.* discuss the implementation of the FDK algorithm on a volume-based reconstruction on both cell-processors and a CUDA enabled GPU [84]. This technique was developed for cone-beam reconstruction and not fan-beam. Generally, cone beam reconstruction techniques are simpler as they mostly do not deal with table transitions. Nonetheless, the technique showed the GPU was approximately 13% faster than an optimized Cell Broadband Engine Architecture developed by Scherl *et al.* [85]. As with most publications in the area, there is little focus on difference maps, image quality, and relative error between techniques. Likewise, the design overview of all techniques was never explicitly described. There was a general overview in a previous publication of Scherl from 2008 [83] with some small code snippets. These pieces of code simply show rudimentary initialization of worker threads for a given task. What is more interesting is the design changes to standard serial FBP or FDK volume-based reconstruction algorithms when efficiently implementing on multi-core CPUs, GPUs, or Cell Broadband Engine Architecture (CBEA) [74] that are used in their reconstruction framework.

Despres *et al.* poster presentation from the 35th conference in Medical Physics shows a brief discussion of a 100x speedup using GPUs for FDK type algorithms, but does not detail results and no formal publication record can be found on his CV [15]. For this reason, no extensive evaluation or discussion of the techniques can be included.

Okitsu *et al.* made a significant publication to the research community, as they provide a thorough comparison of implementations on various GPU hardware [70]. The implementation is common and unique at the same time, as it does use the typical FDK algorithm, but does so in a multi-GPU environment where data sharing and updating is required. The technique follows typical implementation where filtering is achieved through the conventional Shepp-Logan filter [91], and the laborious task is the back-projection to the slice or volume. As seen in the previous description there is no data dependence between projections or pixel/voxels. Likewise, projection ordering is not relevant. For this reason an efficient distribution can be accomplished through simple divide and conquer technique. The authors also attempt to limit the amount of data for GPU storage, as they like to use true low cost commodity cards with low memory, approximately 512MB. For this reason they choose to only store the volume data and update the projection data when required. Although this technique has performed well on commodity GPUs, current mid-range GPUs have much larger memory, approximately 6GB on a Tesla c2070, and huge performance gains can be realized through storage of both the projection data and the volume data.

2.12.3 ATI Stream Processors and FBP

Wang *et al.* show a novel use of ATI's stream processing pipeline in the reconstruction stage of FBP [105]. Wang *et al.* was able to conclude that an optimized Computer Abstract Layer (CAL) implementation on ATI hardware represents approximately the same runtime characteristics of a reasonable Nvidia CUDA implementation [105, 111]. One should note, the CAL implementation is a low-level programming model that is more complex to code than the high-level CUDA Application Programming Interface (API). When comparing ATI's high-level programming model Brook+, the runtime was almost 4x slower than the CUDA technique implemented in Wang *et al.* and originally found in [111]. Wang's *et al.* publication lacks any comparison concerning error, or quality, and; therefore, is difficult to evaluate. Likewise, the exact implementation of FBP is unavailable and simply represents a comparison of runtime of the same technique implemented in different hardware settings. The ATI stream processor architecture is impressive, as they have a large amount of stream processors. There appears to be a lot of potential for ATI GPUs in the near future as the advent of OpenCL has made programming ATI GPU devices easier in non-pipeline scenarios.

2.12.4 FPGA Based Parallel Techniques for FBP

There is a rather large body of publications on FPGAs and the speedups obtainable using this specialized hardware. However, the FPGA technology was revolutionary in 2002 when the publication by Leeser *et al.* was conducted, but rather outdated now, when comparing against current GPUs [12]. Specifically, the techniques were about 100x faster than serial processing on the CPU. The paper showed that, using specialized FPGAs, they were able to produce a worst-case relative error of 0.015% when compared to the pre IEEE-754-2008 floating-point implementation IEEE-754-1985 on the CPU. There is a wealth of publications in the last five years that compare FPGA and GPU implementations. The general argument is, FPGAs are expensive to produce and are only beneficial if the algorithm requires detailed low-level hardware control operations and is complicated in nature [93]. Even with these requirements, GPUs have still been shown to outperform FPGA in some implementations, as seen in [93], if the GPU implementation can be optimized through the use of cached memory or other GPU memory tricks such as page-locking, see Appendix A for more details. However, specialized FPGA implementations are efficient, providing almost 3x the performance when compared to a GPU in [96]. This publication also explores the intricacies involved with programming each of the hardware architectures, FPGAs, GPUs, and CPUs. This is important as efficient algorithm design on the CPU is different than that on the GPU and of that for a FPGA.

2.12.5 Summary of Parallel Techniques for FBP

There are many techniques for parallelizing runtime for FBP. Generally, most techniques try to harness either multicore CPUs, GPUs, FPGAs, or cell-processors. However, many of the techniques never really exploit all available computational power. They often neglect the use of CPUs when developing for GPUs, and likewise neglect GPUs when developing for CPUs. Furthermore, most techniques seldom perform adequate error analysis and provide little comparison of techniques besides runtime. Most important to clinical settings is, not specifically how long the reconstruction takes, but how accurately does the reconstructed image represent the actual inner properties of the object. As long as the runtime is reasonable, meaning not exceptionally long, clinical settings would prefer the most accurate depiction rather than the fastest. The techniques often lack the required details for implementation and are generally technical overviews of research. Although there have been many articles published concerning FBP using parallel machines, most miss the complete evaluation of the technique. *For this reason, the techniques shown in this thesis have been thoroughly analyzed in terms of error, runtime, noise, and use in a clinical settings*. The data used for evaluation are raw attenuation values from CT-scanners and simulated data. This provides the research community a substantial contribution, as the results shown are more applicable to real-world settings and the problems encountered under those environments.

2.12.6 Alternative to FBP Algorithm

Although FBP is the most widely used reconstruction technique in commercial-CT, there is an alternative method to reconstruction, known as linear algebraic techniques [47]. The algebraic techniques can also be used to compute the inverse of the Radon transform. Specifically, the direct inverse techniques, as seen in Section 2.2, represent only direct inverse formulas to the Radon transform. Although these techniques work well, they are based on a continuous domain, and one typically only uses discrete sampling in real-world problems. Therefore, these solutions are only approximations to the true sampled system. Likewise, the sampling is more sparse as one moves from isocenter to the outer diameter of the image area. Also, when working with the direct inverse techniques, noise is not easily modelled. Last, they typically require longer acquisition time because of the required oversampling of the interior to have adequate sampling of the periphery.

The alternative discussed here, algebraic techniques, rest on the fact the Radon transform is a linear transform and the discrete version of the Radon transform is also a linear transform [97]. The algebraic techniques are not new, reviewing literature from early CT development in the 1970s, it is evident that these technique were well developed [97, 27, 25, 34, 35]. The number of projections required for algebraic reconstructions can be greatly reduced when compared to FBP [30, 64, 65]. Guan *et al.* showed that the required number of projections for FBP is $1.57 * n$ and ART requires $0.67 * n$, where n represents the required radial sample $((2 * L)/\pi)$ [30]. This requires consideration of a determined matrix rather than in FBP where minimum sampling rate on the periphery must be maintained. The decreased number of projections means the table translation along z can be increased. This results in less radiation exposure, decreased scan time, and increased throughput of the device. Therefore, the investigation into these methods is important as they can reduce radiation exposure to patients and provide a greater image quality for diagnosis.

2.13 Realization of a Linear Algebra Representation of the Radon Transform

Linear techniques attempt to model the Radon transform of the image $R\{f(x,y)\}$, where $f(x,y)$ is the original image, or in the case of a CT acquisition scan this would be the 2D slice of the object scanned. Remembering the relation of attenuation values, $g(l,\theta)$, from Equation 2.1 and $p(\gamma,\beta)$ we see:

$$\begin{aligned} g(l,\theta) &= R\{f(x,y)\}, \\ \downarrow \quad & \quad\downarrow \quad\; \downarrow \\ \mathbf{b} \quad &= \mathbf{A} \quad\; \mathbf{f}. \end{aligned} \tag{2.23}$$

As these attenuation values are simply the Radon transform of the image $f(x,y)$. Where R is the forward Radon transform, $f(x,y)$ is the 2D image slice, and $p(\gamma,\beta)$ represents the sinogram of the attenuation values. Likewise, the relation to the algebraic methods can be seen from Equation 2.23, originally formulated by Gordon [27] and reproduced by many authors [34, 35, 97, 102], where the vector $\mathbf{b}$ is the non-square sinogram $p(\gamma,\beta)$, which is the result of the Radon transform of the 2D image, $f(x,y)$ is the image we wish to reconstruct represented by $\mathbf{f}$ in linear vector formulation, and $\mathbf{A}$ is the system matrix that represents the transformation of $\mathbf{f}$ and is the relation to the Radon transform. For an easier conceptual understanding we restrict the definition to parallel beam geometry, as the fan-beam case is complex and will be detailed in the methodology Section 3.2.2. When considering this transition from fan-beam, one should remember that rebinning adds more non-uniformity into the projection space [36, 37, 3]. However, interpolation can be used to obtain a uniformly sampled domain. This interpolation can actually reduce the image quality as the fan-beam data is already a discrete representation of the continuous transformation [39].

However, as we are using real-world sampling, only a discrete representation of Equation 2.23 is possible, mainly $\bar{g}(k,l) = g(l_k,\theta_l)$. The same can be said of the discrete samples of the image $\bar{f}(m,n) = f(x_m,y_n)$. Now only considering discrete samples of Equation 2.23, one quickly notices that $\mathbf{b}$ is a vector and not a non-square matrix or sinogram, as shown in Figure 2.3a. The vector $\mathbf{b}$ represents the sinogram stored in vector form as defined by:

$$\mathbf{b}_{kl} = p(\rho,\theta)_{k\times l}, \tag{2.24}$$

$$\mathbf{b} = \{a_1,a_2,a_3,a_4,b_1,b_2,b_3,b_4 \dots ,o_1,o_2,o_3,o_4\}, \tag{2.25}$$

$$\mathbf{b}_i = b_{kL+l}, \tag{2.26}$$

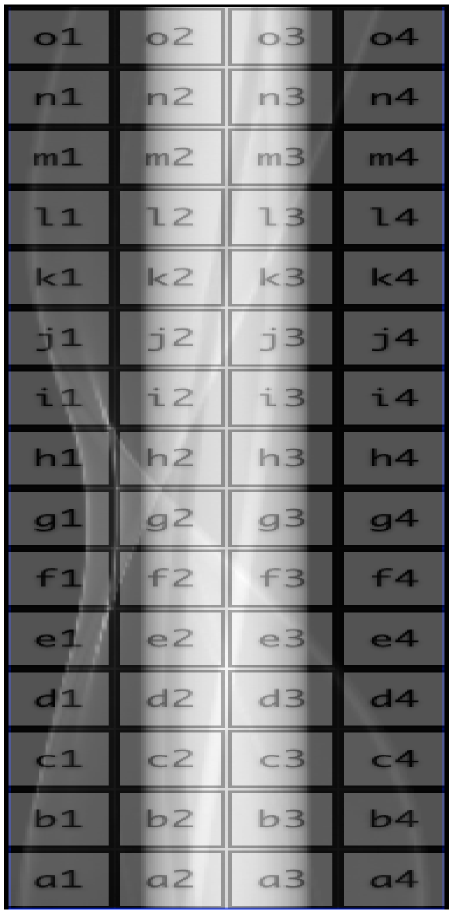
where k and l are the dimensions of the sinogram in matrix form, as seen in Figure 2.11. Therefore, vector $\mathbf{b}$ always has dimension $I = k \times l$. Likewise, the vector $\mathbf{f}$, seen in Equation 2.27, correlates to the image $f(x,y)$, seen in Figure 2.11c, and the mapping is:

$$\mathbf{f}_{mn} = f(x,y)_{m\times n}, \tag{2.27}$$

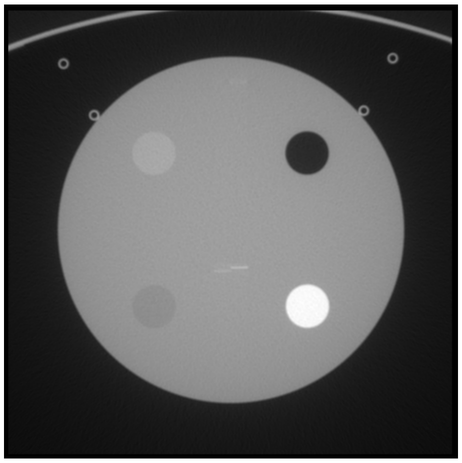
$$\mathbf{f} = \{a_1, a_2, \ldots, a_8, a_9, b_1, b_2, \ldots, b_8, b_9, \ldots, i_1, i_2, \ldots i_8, i_9\}, \tag{2.28}$$

$$\mathbf{f}_j = x_{mN+n}. \tag{2.29}$$

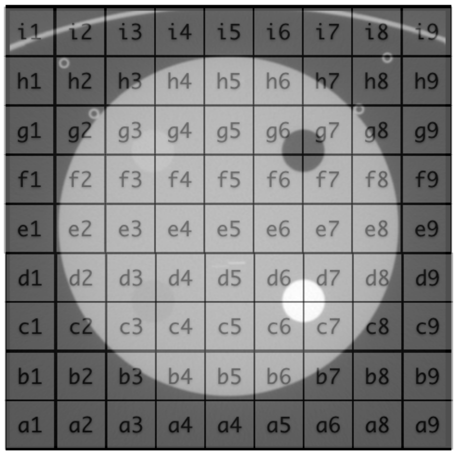
Therefore, vector $\mathbf{f}$ always has a dimension $(m \times n)$ where m and n are the dimensions of the reconstructed image.



(a) Pixelated sinogram of 2.11c



(b) $f(x,y)$ or image


i1 i2 i3 i4 i5 i6 i7 i8 i9
h1 h2 h3 h4 h5 h6 h7 h8 h9
g1 g2 g3 g4 g5 g6 g7 g8 g9
f1 f2 f3 f4 f5 f6 f7 f8 f9
e1 e2 e3 e4 e5 e6 e7 e8 e9
d1 d2 d3 d4 d5 d6 d7 d8 d9
c1 c2 c3 c4 c5 c6 c7 c8 c9
b1 b2 b3 b4 b5 b6 b7 b8 b9
a1 a2 a3 a4 a4 a5 a6 a8 a9

(c) Pixelated $f(x,y)$ used to construct vector x for linear algebra techniques

Figure 2.11: Linear method in CT, (2.11a) shows the simplified decomposition of non-square sinogram $p(\gamma,\beta)$, this decomposition is used in the vector representation $\mathbf{b}$ for linear algebra techniques seen in Equation (2.24). Likewise, (2.11c) represents the pixelated decomposition of (2.11b).

The final step in a linear algebra representation is the realization that the transformation matrix $\mathbf{A}$ is actually the discrete Radon transform operation. The transformation matrix is large, specifically defined by the dimensions of vectors $\mathbf{f}$ and $\mathbf{b}$. Therefore, the transformation matrix $\mathbf{A}$ has dimension $I \times J$ and, in turn, a total of $m \times n \times k \times l$ elements [97, 102]. Although a simplified explanation, one should always account for sensor geometry on the physical acquisition modality. The sensor size and geometry have a direct correlation to accurate reconstructions. For instance, although most clinical-CT reconstructions are $m \times n = 512 \times 512$, in reality, a finer grid should be used. The derivation of the transformation matrix $\mathbf{A}$, defined as:

$$f(x,y) = \sum_m \sum_n f(m,n)\, \phi(x - x_m, y - y_n). \tag{2.30}$$

Taking the Radon transform of the interpolated image $f(x,y)$ from Equation 2.30 leads to:

$$\hat{g}(k,l) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\, \delta(\rho_k - x\cos\theta_l - y\sin\theta_l) \mathrm{d}x\, \mathrm{d}y. \tag{2.31}$$

By substituting the expansion functions from Equation 2.30 into Equation 2.31, we have

$$\begin{aligned} \hat{g}(k,l) = &\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sum_m \sum_n f(m,n)\, \phi(x - x_m, y - y_n) \\ &\delta(\rho_k - x\cos\theta_l - y\sin\theta_l) \mathrm{d}x\, \mathrm{d}y, \qquad (2.32) \\ = &\sum_m \sum_n f(m,n) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \phi(x - x_m, y - y_n) \\ &\delta(\rho_k - x\cos\theta_l - y\sin\theta_l) \mathrm{d}x\, \mathrm{d}y. \qquad (2.33) \end{aligned}$$

The matrix elements of $\mathbf{A}$, $i = k \times L + l$ and $j = n \times M + m$, for the transformation matrix $\mathbf{A}$ are calculated by,

$$\begin{aligned} a_{i,j} &= a_{kL+l,nM+m}, \qquad (2.34) \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \phi(x - x_m, y - y_n)\, \delta(\rho_k - x\cos\theta_l - y\sin\theta_l) \mathrm{d}x\, \mathrm{d}y, \qquad (2.35) \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \delta((\rho_k - x_m\cos\theta_l - y_n\sin\theta_l) - x\cos\theta_l - y\sin\theta_l) \\ &\qquad \phi(x,y) \mathrm{d}x\, \mathrm{d}y, \qquad (2.36) \\ &= \hat{\phi}(\rho_k - x_m\cos\theta_l - y_n\sin\theta_l, \theta_l). \qquad (2.37) \end{aligned}$$

in Equations 2.30 - 2.37, is originally found in Toft's 1996 work on the Radon transform [97], and later from Vlček [102]. Equation 2.30 is an interpolation function function for a given image $f(x,y)$ by a discrete value $\hat{f}(m,n)$ with an interpolation function, the expansion function directly models the interpolation scheme used and can be seen, as only a discrete number of samples are possibly, mainly $f(m,n)$. $\mathbf{A}$ is now defined in terms of discrete values of the interpolation function.

Example: Referring to Linear Technique depicted in 2.12 for realization of $\mathbf{b} = \mathbf{Af}$ equation setup.

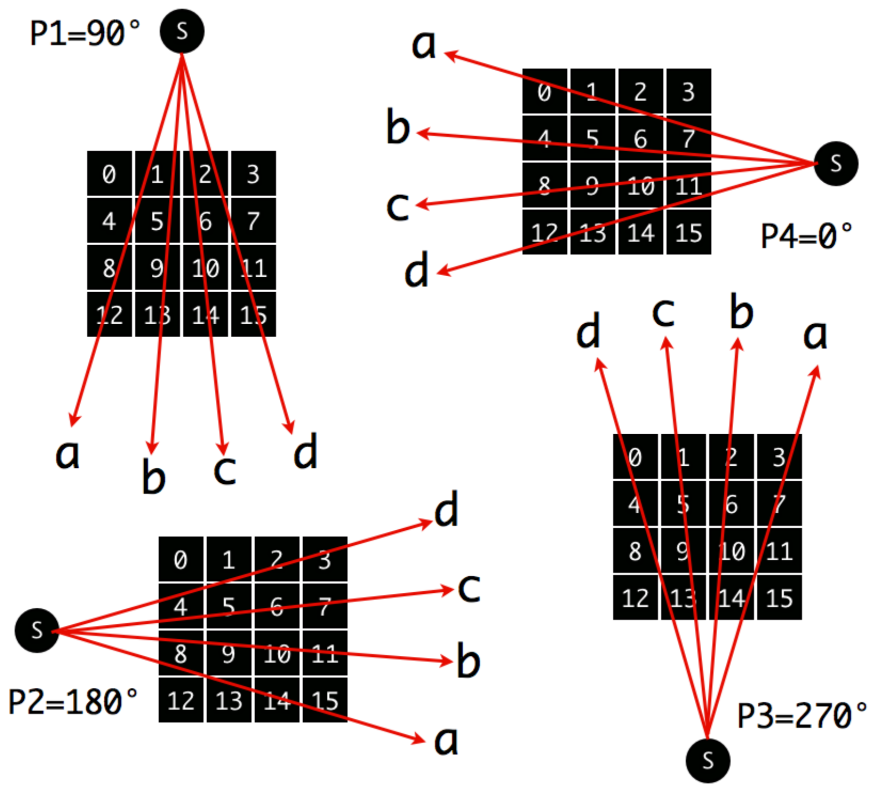


Figure 2.12: Linear method in CT: four projection angles at $90°$, $180°$, $270°$, and $0°$. Likewise, four rays cast through each image with intersecting pixels is seen. One can see from the ray $P1_a$ there is an intersection with pixels 1, 4, 5, 8, and 12. The pixel intersections for this ray are represented by the value one, in expanded notation found in below.

$$
\begin{aligned}
P1 &= \{a_1, b_1, c_1, d_1\}, \text{where,}\\
a_1 &= \{0,1,0,0,1,1,0,0,1,0,0,0,1,0,0,0\},\\
b_1 &= \{0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0\},\\
c_1 &= \{0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0\},\\
d_1 &= \{0,0,1,0,0,0,1,1,0,0,0,1,0,0,0,1\}.\\
P2 &= \{a_2, b_2, c_2, d_2\}, \text{where,}\\
a_2 &= \{0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,1\},\\
b_2 &= \{0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0\},\\
c_2 &= \{0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0\},\\
d_2 &= \{0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0\}. \qquad (2.38)\\
P3 &= \{a_3, b_3, c_3, d_3\}, \text{where,}\\
a_3 &= \{0,0,1,1,0,0,0,0,0,0,0,1,0,0,1,0\},\\
b_3 &= \{0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0\},\\
c_3 &= \{0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0\},\\
d_3 &= \{1,0,0,1,0,0,0,1,0,0,0,0,0,1,0,0\}.\\
P4 &= \{a_4, b_4, c_4, d_4\}, \text{where,}\\
a_4 &= \{1,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0\},\\
b_4 &= \{0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0\},\\
c_4 &= \{0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0\},\\
d_4 &= \{0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0\}.
\end{aligned}
$$

2.14 Derivation of the Discrete Matrix Elements of A

As $\mathbf{A}$ is a representation of the discrete Radon transformation, there are many ways to discretize the image $\hat{f}(m,n)$. A viable solution to the estimation is a classical nearest neighbour approximation [10, 97, 102]. Four interpolation techniques were evaluated, mainly Nearest Neighbour Approximation (NNA), Fast Nearest Neighbour Approximation (FNNA), approximation by the Radon transform of a square, and sinc interpolation. FNNA provided the best approximation of the matrix $\mathbf{A}$ with lower error relative to the source image $\bar{f}(m,n)$ [102]. Likewise, it had the least error for an individual projection. We will now continue with our review of the literature concerning discretization techniques from [97, 102].

Discretization of the System Matrix A Using a Nearest Neighbour Approximation

An example of simplified NNA and the resultant vector entries can be seen in Figure 2.12 and Equation 2.38. The simplified techniques for NNA described in Yair's early work [10], and later in Toft's dissertation [97], and Vlček dissertation [102], describes a binary approach. The simplified method states that if a pixel and a ray are incident then 1 is used, if not a 0 is recorded for the entry in the system matrix $\mathbf{A}$. The problem is evident when $P1 = 90°$ and ray a in the simplified depiction in Figure 2.12. In particular, the same weight is given to pixels that are incident with ray a, namely pixels 1, 4, 5, 8, and 12. However, it is shown that pixel-four's mid-point is far from $P1_a$, and that pixel twelve's mid-point is much closer to the finite-width ray ($P1_a$). For this reason we know this simplified technique found in the literature has significant shortcomings. One should also consider when using a binary scheme for incident and non-incident pixel/ray interaction, that a rescaling is required and one must solve the modified system $\triangle\mathbf{f}\mathbf{A}\mathbf{f} = \mathbf{b}$, where $\triangle\mathbf{f}$ represents the size of the pixel in the image $f(x, y)$ [102]. For completeness, we show the simple process involved in the calculation for matrix elements of $\mathbf{A}$ by:

$$\rho^{'} = \rho_r - x_m \cos\theta_l - y_n \sin\theta_l, \tag{2.39}$$

$$\text{if,} \quad \frac{\triangle x}{2} > |\rho^{'} \cos\theta_l| \;\text{ and }\; \frac{\triangle x}{2} > |\rho^{'} \sin\theta_l| \Rightarrow a_{i,j} = \triangle x, \tag{2.40}$$

$$\text{else}, a_{i,j} = 0. \tag{2.41}$$

A depiction of NNA can be seen in Figure 2.13. This figure represents a pixel and ray interaction, one can see that $\rho^{'}$ is the distance of the ray from the $p = [x_m, y_n]$. If $\rho^{'}$ is within the bounds set by Equation 2.40, the ray is incident with the pixel. This technique can efficiently be implemented, row-by-row or column-by-column, to build the system matrix $\mathbf{A}$.

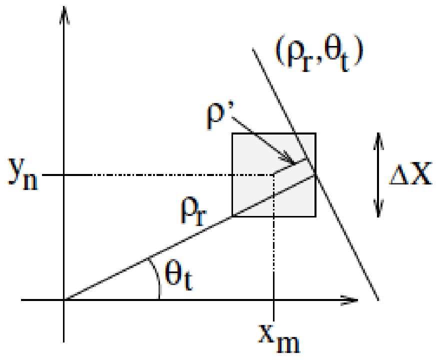


Figure 2.13: Nearest neighbour approximation is shown with pixel $p = [x_m, y_n]$. Each pixel is defined by a center point and the size that is represented by $\triangle X$. Where ρ_k represents the offset on the detector, θ_l represents the angle of the source, $\rho^{'}$ represents the distance of the p to the ray [97].

Discretization of the System Matrix A Using a Discrete Radon Transform

The discrete Radon transform provides a novel way to discretize the image; given that a $p = [x_m, y_n]$ can be represented by a square pixel of size Δx^2 and a constant amplitude of 1. Assuming a square centered at $(0, 0)$ and of size 2×2, then using the rules for scaling and translation, see [97, 102] Sections B.6, and Theorem 2.5 respectively, one can represent the discretized value through the simple formula:

$$a_{i,j} = \frac{\triangle x}{2} \hat{g}_{square} \left(2 \frac{\rho_k - x_m \cos\theta_l - y_n \sin\theta_l}{\triangle x}, \theta_l \right), \tag{2.42}$$

where, $\hat{g}_{square} = \hat{g}(r, t)$ is the discrete Radon transform of the 2×2 square pixels centered at $(0, 0)$. There are variations to this interpolation scheme that uses several rays and average the results from each ray. The ability to closely model the real geometry of the machine used during acquisition, such as real detector size, can be incorporated into this discretization scheme [97]. The approximation by the Radon transform of a square provides a novel technique for the discretization and lower error than NNA. Toft's dissertation showed the technique performs well and does not suffer from storage problems and long computational time as does the Sinc interpolation method. Likewise, similar results were found in [102], where the Radon Kernel performed better, in terms of L2-error, than NNA and is slightly better than a Sinc interpolation techniques defined by:

$$\text{sinc}(x) = \frac{\sin(x)}{x}. \tag{2.43}$$

Discretization of the System Matrix A Using Sinc Interpolation

The Sinc interpolation scheme is naturally derived from signal processing and representation of the signal in terms of sampling rate up to the optimal Nyquist sampling level. The Sinc interpolation method will not be used in this thesis as there is a large amount of processing time associated with the generation of **A** with this method. Likewise, when creating a finer discretization, the memory requirements double. Furthermore, the technique has the ability to produce negative entries, which is physically impossible [97, 102]. Last, the technique does not perform well in terms of error when compared to the discrete Radon or FNNA techniques. For further reading on the Sinc interpolation method, see Toft's dissertation [97].

Discretization of the System Matrix A Using Fast Nearest Neighbour Interpolation (FNNA)

The most interesting interpolation technique is FNNA [102], as it performs well compared to the most common techniques like NNA, discrete Radon, and Sinc interpolation. Vlček's description uses some different variable definition. Mainly, R and T are anogalous to our K and L. The FNNA approach is based on the direct Radon transform found in [102] and is expressed by:

$$T = 2\,t_{min} + 1, \tag{2.44}$$
$$\text{origin}_x = \lfloor \frac{M-1}{2} \rfloor, \tag{2.45}$$
$$\text{origin}_y = \lfloor \frac{N-1}{2} \rfloor, \tag{2.46}$$
$$t_{min} = \lceil ((M-1-origin_x)^2 + (N-1-origin_y)^2)^{\frac{1}{2}} + 1 \rceil, \tag{2.47}$$

where $[origin_x, origin_y]$ represent the center of the image, t_{min} is the minimal value of t.

As discussed previously, the system matrix $\mathbf{A}$ has dimension $IJ = m \times n \times k \times l$, where $m \times n$ is the dimension of the source image and $k \times l$ is the dimension of the sinogram. For this reason, the system matrix size is directly correlated with the size of the source image and sinogram. Furthermore, because of this correlation the speedup in FNNA is attributed to the fact that the loop over the variable t is omitted and directly calculated. This does limit the arbitrary choice of t and therefore the value needs to be calculated as seen in Equations 2.44 - 2.47.

Vlček provides experiments that show a further discretization of the pixels is not beneficial to the reconstruction in terms of maximal error of approximation. The results in Vlček's publication are likely attributed to the use of the float data type and the limited precision available. We assert that the use of double precision data type, and more accurate modelling of the numerical error, would result in less maximal error with great pixel subdivision. This assertion is feasible as increased sampling of any function does not create aliasing as suggested by the analysis by Vlček. The basic algorithm from Vlček is included in Algorithm 2.1, as it completes the definition of the FNNA interpolation technique.

Algorithm 2.1 Calculate System Matrix A by FNNA Interpolation [102]

```
for $r = 0$ to $R - 1$ do
    for $m = 0$ to $M - 1$ do
        $c_m := (x_m - origin_x)\cos\theta_r$
    end for
    for $n = 0$ to $N - 1$ do
        $s_n := (y_n - origin_y)\sin\theta_r$
    end for
    for $m = 0$ to $M - 1$ do
        for $n = 0$ to $N - 1$ do
            $\bar{t} = c_m + s_n - t_{min}$
            $t = \lfloor \hat{t} \rfloor$
            $\delta = \hat{t} - t$
            $A_{rT+t,nM+m} = A_{rT+t,nM+m} + \frac{1-\delta}{sizeof\,sub-pixel}$
            $A_{rT+t+1,nM+m} = A_{rT+t+1,nM+m} + \frac{\delta}{sizeof\,sub-pixel}$
        end for
    end for
end for
```

Bresenham's Line Algorithm, an Alternative to a System Matrix Representation

Although specific models can account for the development of various interpolation schemes, they are expensive to produce, even in parallel, when considering constructing the complete system matrix $\mathbf{A}$. The system matrix has $m \times n \times k \times l$ elements, which is a huge matrix. One can notably produce a sparse representation of the system matrix, as only 0.1% of the entries are non-zero [97, 102]. However, when the scanner geometry changes, the type of scan changes in terms collimation, table movement, dual source, etc and one must recreate the system matrix at a great cost. Producing all possible system matrices would be too costly and possibly more expensive than storing the non-sparse representation of one system matrix. The main cost associated with the process is the interaction of a ray with the pixel or voxel. In the case of voxels, one can imagine the extra complexity when considering a volume vs. a 2D image or pixel as this would add an even greater level of computation, as more voxels would need to be visited to correctly determine the row entry of the system matrix to accurately define the process. An alternative method that is fast is based on Bresenham's line algorithm for digital plotters [8]. Although the technique is slightly modified in this thesis, it plays an instrumental role in the development of a fast method for iterative techniques used to solve the linear equations, the basic line plotting algorithm can be seen in Algorithm 2.2. Although the technique cannot be used directly in CT reconstruction, it can be modified to suit our interest. The modified algorithm will be presented in Chapter 3 and analysis in Chapter 6. As seen in Figure 2.14, only those pixels in 2D, or voxels in 3D, visited by the ray are sampled. Conversely, in the standard techniques, all $m \times n$ elements are visited in the 2D case, and each voxel in the 3D case. The basic Bresenham's algorithm was adopted by Siddon in 1985 to calculate pixel interaction with rays in a more efficient manner [94]. Jacob's *et al.* publication revisited the implementation details of Siddon's algorithm and was able to speed-up the technique substantially [41]. As iterative methods for reconstruction rely on efficient implementations, these techniques have truly shown to be instrumental in the development of fast iterative algorithms for CT. As shown by Xu *et al.*, the Siddon method produces the most accurate forward and backward-projections [19] and, for this reason, it is the principle method used for forward and backward ray-driven techniques used in this thesis.

2.14.1 Summary of Interpolation Methods

The discretization of the system matrix $\mathbf{A}$ through interpolation is important to accurately reconstruct the image $f(x, y)$ because, without a valid representation of the forward-projection process, a high quality reconstruction is impossible. Implementation of the various techniques were consistent with previous conclusions in [97, 102] and basic linear interpolation has been shown to be adequate [47, 82, 19, 51]. The focus in this thesis is to development of an efficient parallel environment for CT reconstruction of raw attenuation values from clinical-CT, outlining differences in programming methodologies in the various environments, comparison of error of implemented

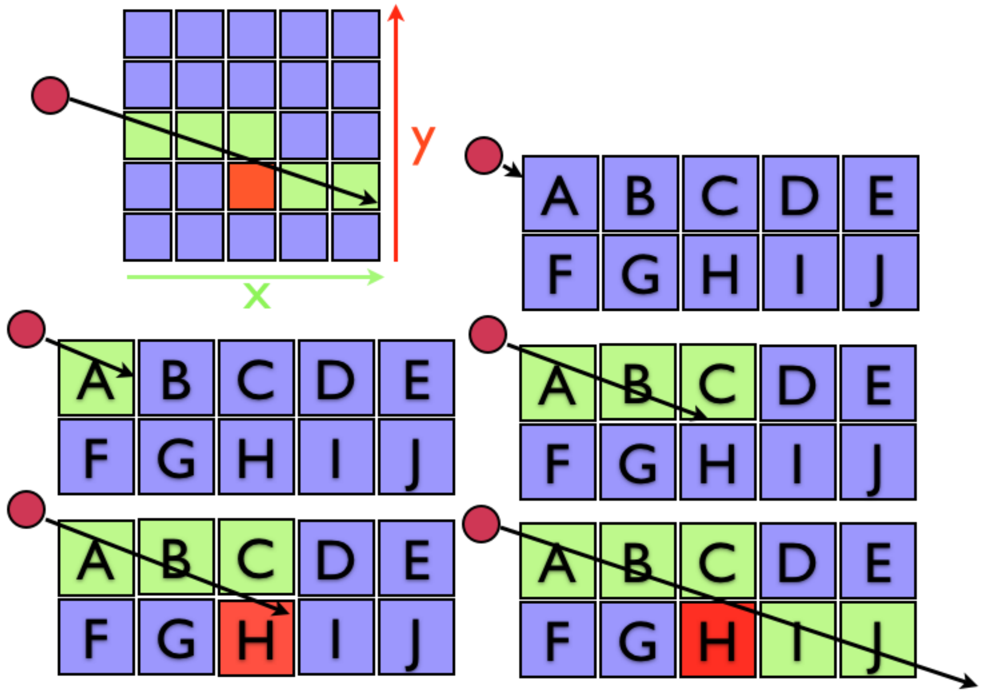


Figure 2.14: A ray is cast, the x-index and y-index are set at the location of interception with the square. The x-index is incremented three times as the $\triangle x > \triangle y$. Once the $\triangle y > \triangle x$ then the y-index is decremented once followed by two more increments of the x-index. This continues until the ray exits the square, volume, or screen. [8]

Algorithm 2.2 Bresenham's Line Algorithm [8]

```
procedure BRESENHAM($x0, y0, x1, y1$)
    $dx := abs(x1 - x0)$
    $dy := abs(y1 - y0)$
    $sx := sy := 1$
    if $x0 > x1$ then
        $sx := -1$
    end if
    if $y0 > y1$ then
        $sy := -1$
    end if
    err := dx-dy
    while $x0 \mathrel{!=} x1$ && $y0 \mathrel{!=} y1$ do
        WRITEPIXEL($x0, y0$)
        e2 := 2*err
        if $e2 > -dy$ then
            err := err -dy
            x0 := x0 +sx
        end if
        if $e2 > dx$ then
            err := err + dx
            y0 := y0 +sy
        end if
    end while
end procedure
```

techniques, and adding insight as there are more efficient programming techniques that make use of parallel architectures, multi-GPU development, and HI-RES display technology. Nonetheless, some of the most common interpolation schemes have been shown. Although, the subject has been revisited in an attempt to use the large complete system matrix, the results show, even with current techniques such as parallel and multi-GPU computing, these algorithms are still computationally challenging with current hardware and will likely be challenging far into the future as advancement in processor speeds, memory speeds, and overall computing have been slow from 2003 to 2011. Luckily, the advent of the GPU has provided a new parallel framework for programming in a highly concurrent manner and, for this reason, iterative ray-tracing implementation have been successful. As previously discussed, linear interpolation has been deemed accurate and current GPUs have linear interpolation and trilinear interpolation built-in through hardware accelerated operations and, for this reason we exclusively show results using linear interpolation when possible.

2.15 Methods to Solve Linear System in CT Reconstruction

The system matrix $\mathbf{A}$ dimensions are defined in terms of the reconstruction region or source image (m and n) and the sinogram (k and l). The large system matrix $\mathbf{A}$ creates a number of difficulties for solving the linear equations $\mathbf{Af} = \mathbf{b}$. For instance, consider a typical reconstruction with a source image $f(x, y)$ ($m = n = 512$), and a sinogram $\bar{g}(k, l) = g(l_k, \theta_l)$ size $k = 736$ and $l = 1152$.

This represents a typical clinical reconstruction for a single axial slice, and a system matrix $\mathbf{A}$ with 231,928,233,984 elements. Assuming typical four byte entries for the sparse matrix, one would have four bytes per element and approximately 900GB of storage requirement for a single slice. This example is incredibly unrealistic on current and near-future computer hardware, as one generally has 128 or more slices per scan. Likewise, denser interpolation techniques are required and one could likely quadruple this memory requirements. Therefore, techniques to solve the linear equations need to be efficient in terms of memory and computation, as without efficient storage techniques, the major time in computation is spent transferring data. Likewise, one could almost remove the storage requirement and simply calculate the entries of $\mathbf{A}$ as needed. This process would result in redundant work as these entries are needed many times over when solving the system. Luckily, as previously mentioned, $\mathbf{A}$ is known to be sparse. Given the above example, and FNNA interpolation, it is shown that less than 0.1% of the entries in $\mathbf{A}$ are non-zero: approximately 231,928,234 non-zero entries and approximately 900 MB [102]. Therefore, sparse matrix storage techniques are a good choice as they have lower memory requirements and the computation time for correct indexing is good. Remembering the considerations set forth by the large size of $\mathbf{A}$, the following sections review common techniques for solving the linear system $\mathbf{Af} = \mathbf{b}$ after a quick primer on the connections between common matrix operations and the Radon transform.

2.16 Duality of Matrix Operations and the Radon Transform

Toft's dissertation, Section 9.3, describes a close correlation between matrix operations and the Radon transform [97]. We describe some of them here, but Toft deserves credit, as most literature does not recognize these simple relations and they are truly essential in the complete understanding of how the mathematical techniques are linked and the methodology's soundness.

In the previous section, we described in detail the steps involved in the alternate representation of the Radon transform in a more common linear algebra framework where matrix and vector operations can be used to reconstruct $f(x, y)$. Formally, Equation 2.23 shows the transformation from $g(l, \theta) = R\{f(x, y)\}$ to the linear system $\mathbf{Af} = \mathbf{b}$. Therefore, common matrix operations have correlating Radon operations. For instance, the Equation $\mathbf{Af} = \mathbf{b}$ is the Radon transform ($\mathbf{A}$) of the discrete image $f(x, y)$ to discrete sinogram $\mathbf{b}$. Another common operation is the scalar product between a row of $\mathbf{A}$, namely $\mathbf{a_i}$ with the discrete image $\mathbf{f}$, i.e. $\mathbf{a}_i^T\mathbf{f}$. This is equivalent to the Radon transform of the source image at a specific angle into the parameter domain or one row of the sinogram. This duality makes the transition to vector form relatively straight forward, seen by:

$$\mathbf{b_i} = \mathbf{a_i^T f} \; \leftrightarrow \text{Radon transform of image, at one angle,} \\ \text{to one row in sinogram.} \tag{2.48}$$

Another common operation is the back-projection from the sinogram into the image domain. The matrix operation is $\tilde{\mathbf{f}} = \mathbf{A}^{\mathbf{T}}\mathbf{f}$, the transpose correlates to the back-projection operation.

$$\tilde{\mathbf{f}} = \mathbf{A}^{\mathbf{T}}\mathbf{b} \leftrightarrow \text{Adjoint Radon transform of sinogram into image domain.} \tag{2.49}$$

The correlation between the forward and backward projections can now be seen. Toft also shows how, under the non-valid assumption of full-rank of $\mathbf{A}$, FBP can be modelled. One cannot assume the full-rank of $\mathbf{A}$ because of non-zero mean values in the frequency spectrum are set to zero during filtering. The mathematical model is nonetheless accurate. Interested readers should consult [97] for a more detailed review.

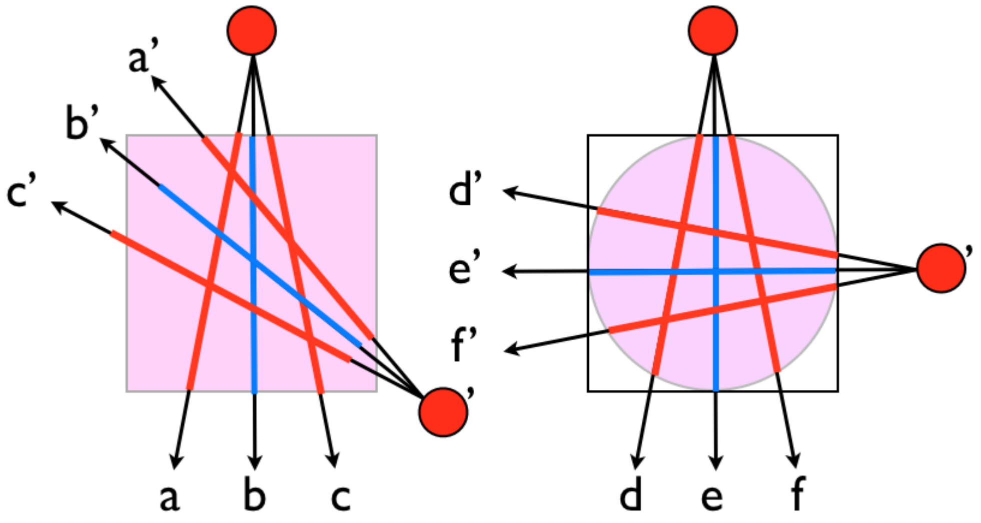


Figure 2.15: The left image shows the reconstruction region in purple with non-constant length rays are evident after rotation. Likewise, consider a right-triangle, Pythagorean theorem shows the red rays (hypotenuse) are obviously longer than the blue central ray. The right image shows non-constant ray-length reconstruction where each ray does not have equal intersection length with the circular reconstruction region depicted in purple

2.17 Algebraic Reconstruction Technique (ART)

ART represents one of the earliest methods used to solve the inverse Radon transform. The technique was developed by Gordon *et al.* in 1970 and relies on an iterative method to update the solution to the large system of linear equations [27]. The general algorithm found in ART is actually originally developed by Kaczmarz in 1937, see Kacsmarz's algorithm for more details [46]. The general idea of the algorithm is the ability to satisfy each equation of the linear system $\mathbf{Af} = \mathbf{b}$ in the sub kth iteration [27, 97, 102]. Iterations in ART are not typical iterations, so two iterations would equate to $2 \times rows$ of the system matrix $\mathbf{A}$. One complete iteration of ART is completed once all rays and all projections have been forward-projected, relaxed, and back-projected. One update rule for $f(x, y)$

is defined as:

$$\mathbf{f^{k+1}} = \mathbf{f^k} + \lambda * \left(\frac{\mathbf{b_i} - \mathbf{a}_i^T \mathbf{f}^k}{\mathbf{a}_i^T \mathbf{a}_i} \right) \mathbf{a}_i, \tag{2.50}$$

where, $\mathbf{f}^{k+1}$ represents the update image after one ray is forward-projected $\mathbf{a}_i^T \mathbf{f}^k$, relaxed by λ, and back-projected $\mathbf{a_i}$, to produce the update image $\mathbf{f^{k+1}}$. Recalling from the previous section that $\mathbf{a_i^T}$ represents the row in the system matrix and $\mathbf{f^k}$ is the current image being projected through. For the remainder of the text the term *ART*-iteration will be used when referring ART-type iterations. The relaxation parameter λ is usually constrained between $(0, 1]$ and can suppress noise as $\lambda \to 0$ at the cost of slower convergence [47]. Last, the denominator ($\mathbf{a_i^T a_i}$) represents the length of the ray through the reconstruction region.

There are some authors that consider this length to be a constant and choose to ignore it. Although, this is not entirely true as it is only constant length from source to detector. Figure 2.15 depicts this rather well as one can see in the first image with a square reconstruction region and two projections from the source. It is evident that ray-b is shorter than rays-a and ray-c. Secondly, after rotation from $90°$ to $330°$ rays a', b', c' have different lengths than their counter parts a, b, c at $90°$. The second image is where the lengths of all rays are non-equal when using a circular arc detector array and a circular reconstruction region, as any rotation around the reconstruction region would result in equal length from source to detector but the intersection length is still not equal. Although the Keck's *et al.* do not explicitly state this assumption, the authors do note this assumption in a later paper on the topic [48] and was further validated through correspondence. We will demonstrate in this thesis that this assumption is invalid. Measuring an accurate pixel length is relatively expensive, but it does provide slightly more accurate results and should be considered when defining a parallel reconstruction algorithm. Specifically there are two forms of length that were considered in this thesis, mainly those pixels that were intersected which gives a pixel count, and the actual length of the intersect with the reconstruction region. Take for instance the ray with $\gamma = 0°$, this would be the central ray on the fan, or central channel, and would be directly along the x-axis if the source was at $0°$. This ray has region intersection length that is constant throughout all projections, but when the source is at $45°$ there is a great number of pixels across the diagonal that the projection value is distributed across. One should consider these metrics when designing the algorithm and should formally describe what method is used.

ART is accurate for reconstruction when modelled in a precise way [27]. We will show that a naive implementation will result in poor performance and poor reconstructions in Section 4.1.3. For instance there are two methods considered in both Toft's and Vlček dissertations for *ART*-iteration ordering, mainly cyclic and random. The cyclic is based on the model of the actual machine and in fact it acquires attenuation values for each projection in a cyclic manner. The random technique simply chooses a random projection among the available projections, assuring every projection is used only once. They note the random choice is experimentally better for convergence and provides reasonable output in only one complete *ART*-iteration. Reviewing the literature, it is evident that choosing projection rays that are maximally orthogonal to each other is the best choice and provides much faster convergence [29]. The original publication, from Guan *et al.*, was in 1994, several years before either dissertation was produced. The justification for maximal orthogonality between updates is considering basic ART in a cyclic manner will update low frequency content first and high frequency components are recovered late and slowly [29]. Using projections that are of greater angular distance from each other results in faster convergence, which is depicted in Figure 2.16. As the angular distances increase, less *ART*-iterations are required and faster convergence is realized.

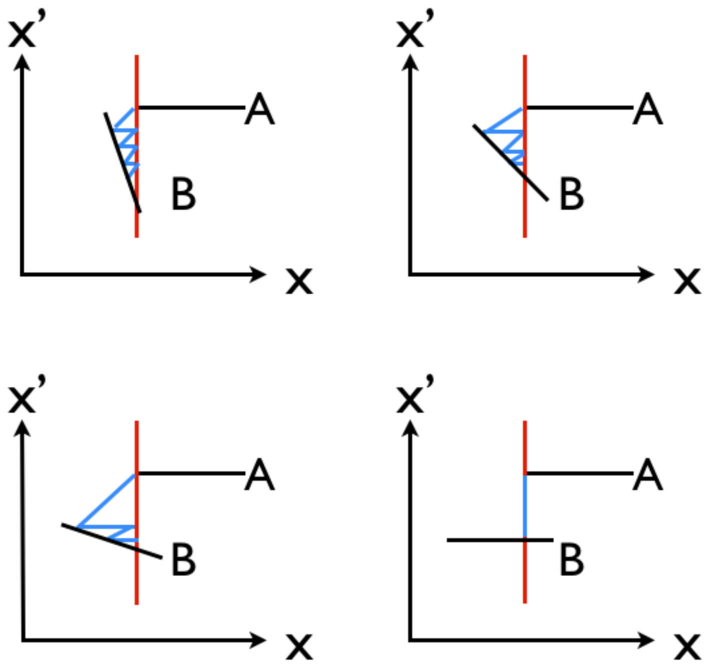


Figure 2.16: Four images are shown with increasing angle of orthogonality between rays. Four angles are depicted and the number of steps to reach the intersecting point can be seen in blue, decreasing as the angle increases until orthogonal.

ART is a good reconstruction algorithm for low projection-count, noisy, and reduced X-ray radiation exposure and will consistently provide better reconstructions under these conditions [97]. The decreased number of projections, and/or decreased X-ray radiation level, for a given CT scan has a drastic correlation on the reduction of the lifetime risk of cancer associated with a CT scan. That being said, there are a number of problems encountered during implementation. First, the number of *ART*-iterations required is not well defined, as it is defined by the scanning geometry and no specific criteria has yet been set for dosage levels. Second, as the number of *ART*-iterations increase so can the noise. Therefore, careful relaxation and regularization parameters need to be considered. ART has computational complexity $O(k \times l \times n \times m)$, where k is the number of channels in the sinogram, l is the number of projections, and n and m are the square reconstruction region of f(x,y); and therefore making the technique challenging. ART has provided a strong algorithm for underdetermined scans in the goal of high quality CT reconstruction and has led to the development of the Simultaneous Algebraic Reconstruction Technique (SART) [1].

2.18 Simultaneous Algebraic Reconstruction Technique (SART)

An important consideration in any reconstruction scheme is the scanner geometry used during acquisition. Today's CT scanners rely on multi-channel and multi-row detector to create a volume of x-rays that has exactly $channels \times rows$ of detectors. Therefore, satisfying a given equation in the *k*th sub-*ART*-iteration, as in ART, does not closely model the current hardware because a given detector value would be the summation of attenuation values of neighboring rays. Therefore, an alternative scheme to model the system is SART [1]. The technique relies on a projection-wise update to the reconstructed image or volume, rather than a ray-wise update to the reconstructed image and can formally be described as:

$$\mathbf{\hat{f}_j^{k+1}} = \mathbf{\hat{f}_j^{k}} + \lambda * \left(\frac{\sum_i \mathbf{a}_{ij} \left(\frac{\mathbf{b_i} - \sum_j \mathbf{a_{ij}} \mathbf{\hat{f}_j^{k}}}{\sum_j \mathbf{a_{ij}}} \right)}{\sum_i \mathbf{a}_{ij}} \right), \tag{2.51}$$

where $\mathbf{\hat{f}}$ represents the estimated image of the true reconstruction $f(x, y)$, rather than the previous definition where $\mathbf{\hat{f}}$ was an estimate to the true sinogram. The parameter λ is a relaxation parameter that, as in ART, is constrained to $(0, 1]$ and has the ability to suppress noise at the expense of a speedy convergence. The $\sum_i \mathbf{a_{ij}}$ represents the number of rays contributing to a given pixel estimate, and the $\sum_j \mathbf{a_{ij}}$ represents the length of a ray and can only be considered constant for a given ray when one considers the physical distance from source to detector. The non-constant length between rays, given different reconstruction regions, can be seen in Figure 2.15. $\mathbf{b}$ represents the true sensor values for a given projection.

The updates to the image estimate $\mathbf{\hat{f}}$ are performed on a projection basis instead of ray-by-ray as in ART. Therefore, noise is drastically reduced and convergence is generally quicker. The global convergence of SART was proved by Jiang *et al.* when one assumes the attenuation values are non-negative [43]. However, there is no method to estimate, for a given reconstruction based on raw attenuation values, how many *ART*-iterations will be required to converge on a solution. This makes the computational complexity of the algorithm difficult to assess. Nonetheless, given one *ART*-iteration of SART the computational complexity is $O(k \times l \times n \times m)$.

2.19 Heuristic in ART or SART

One can provide a heuristic to the algorithm, if knowledge about the domain can effectively be modelled. The back-projection of correction terms in the reconstruction region can be modelled by a Hamming window as seen in Figure 2.17. The uniformly back-projected value is replaced by the

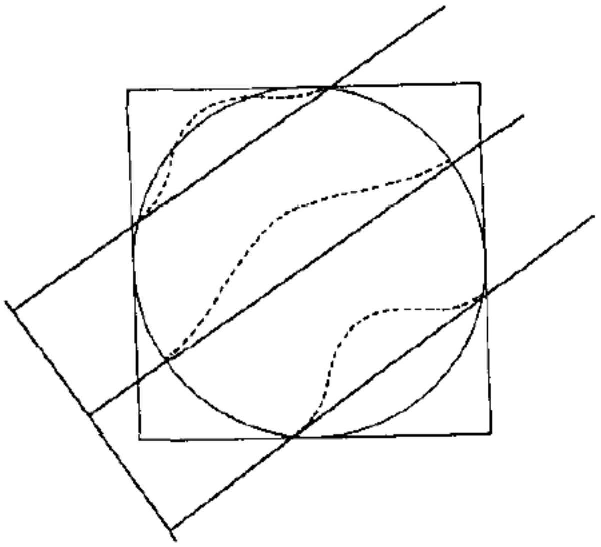


Figure 2.17: The Hamming window for circular reconstruction region based on ray length [47].

weighted Hamming window based on length through the reconstruction region [47]. The heuristic can help achieve faster convergence, as it more accurately models the ray attenuation through materials.

2.20 Relaxation, Constraints, and Regularization for ART-Type Algorithms

As discussed previously, the relaxation parameter λ in Equation 2.51 and 2.50 has the ability to suppress noise as it approaches 0 [47]. However, this leads to a slower convergence and is specific to a given scan. Let us consider the simple example where $\lambda = 1$. This would effectively update the solution space to satisfy the given ray in ART or the given projection in SART. However, it is easy to understand that in later iterations this absolute satisfaction could disrupt other rays or projections in the solution space. Therefore, a relaxed update where λ approaches zero can provide a less dramatic update to the solution space and better ability to satisfy alternative rays, or projections, depending on the update scheme. Therefore, the careful consideration of λ is important to convergence. Testing will show large dramatic updates, where higher λ values are used, can provide faster convergence if used in early iterations. Likewise, lower λ values in later iterations have the ability to refine higher frequency data as the updates are less dramatic.

Constraints can play another important roll as the ability to produce negative attenuation values is not physically possible. As previously mentioned, the system if often underdetermined and has an infinite number of solutions. Therefore, constraining the negativity of possible solutions helps create a more stable and viable solution, given the prior knowledge of the scanner characteristics.

Likewise, an upper bound is possible and constraining solutions to this upper bound will ultimately stabilize the solution with less error. Regularization in SART is advantageous for noisy data or limited projections [73]. Pan's *et al.* uses a single GPU SART implementation with total variation regularization [73]. The technique preserves straight, and sharp edges, making the regularization technique applicable to CT reconstruction. The technique is evaluated on rather limited datasets and small reconstruction sizes. Nonetheless, regularization has been shown to help reconstruction to suppress noise.

2.21 Singular Value Decomposition (SVD) and Conjugate Gradient Method

Toft's dissertation shows a description of why SVD and Conjugate Gradient methods are ill suited for CT reconstruction [97]. Specifically, the SVD method is too computationally demanding, computational complexity $O(4I^2J + 8J^2I)$ [97]. Remembering the derivation of I and J implies an approximate computational complexity of $O(n^6)$. This means even for simple problems in CT-reconstruction, computational advancements are required in order to use the technique in CT-reconstruction. Likewise, the conjugate gradient method generally relies on the storage of $\mathbf{A}^\mathbf{T}\mathbf{A}$, which is not a sparse matrix, requiring approximately 3.3TB for a 1024x1024 reconstruction using a single-source scanner with 1152 projections and 736 channels. Adding FFS-modes and dual-source technology further increases the size. As mentioned, there are other conjugate gradient methods that do not require this computation and storage. The low storage Conjugate Gradient methods are a research topic entirely on their own, for further reading see [97, 72].

2.22 Recent Parallel Computational Techniques for Linear Algebra Methods in CT

2.22.1 Parallel Implementations of ART-Class Algorithms

Melvin *et al.* provides two good parallel implementations of ART type algorithms [57, 58]. The problem is decomposed and implemented on a shared-memory machine with 6-processors in the 2008 publication. The reconstruction time was reduced significantly and the authors showed radiation dose could be reduced greatly, with no loss to reconstruction detail versus a typical serial implementation. One consideration of parallel computing under HPC environments is that the cost associated with double precision versus single precision is not significant. However, in GPU implementations this cost is significant and warrants investigation.

2.22.2 GPU Implementations of ART-Class Algorithms

The GPU has brought advancements to many disciplines in recent years, as it provides parallel computation at commodity hardware costs. Using the GPU effectively in an iterative reconstruction algorithm is a challenging task as the data size is large and GPUs typically have small amounts of memory. Specifically, current GPU hardware has a maximal memory size of 6GB on a Nvidia Quadro 6000. Likewise, divergence in algorithms severely effects performance, as does atomic writing to memory, non-coalesced reads, host-to-device memory transfers, and device-to-host memory transfers, see Appendix A for further reading on these subjects. For these reasons, careful and generally non-conventional algorithm design is needed.

There are three large and active research groups in iterative reconstruction on the GPU, mainly Mueller *et al.* [61, 109, 110], Keck *et al.* [49, 50, 48], and Pien *et al.* from Partner's Research Group of Massachusetts General Hospital, in collaboration with North Eastern University and Siemens Advanced Imaging [42].

All groups typically do not give their exact forward or backward ray-projection techniques, but one could argue they are generally using some form of Siddon's algorithm, as it can model the ray-projection and grid traversal accurately [94, 41], or possibly Joseph's [44].

There are two forms of the back-projection technique used, ray-driven or voxel/pixel driven back-projection. Using a ray-driven forward-projection (RD-FP) and a ray-driven backward-projection (RD-BP) is considered a matched pair [112]. However, it has been shown by Zeng *et al.* that an unmatched pair, where the back-projection is less constrained, can produce a faster algorithm and still keep a valid reconstruction [112, 49]. Typically, the unmatched pair is constructed by a RD-FP and voxel-driven back-projection (VD-BP). The benefits of a VD-BP is twofold. First, the VD-BP does not require atomic writing, which drastically slows down iterative methods, as a pixel is projected onto the sensor array and the sensor array is linearly interpolated or bilinearly interpolated. Atomic writing, as previously used, is the process of serially updating a value and is costly as threads must wait and obtain locks on registers before updating a value. Second, the VD-BP has only $m \times n$ lookups, where m and n are the width and height of the reconstructed image. On the contrary the RD-BP approximately has $numberOfRays \times ((ray\_length)/(voxel\_size))$ and the ray and volume, or slice, intersection is costly, not to mention the requirement of atomic writes as many rays can update one voxel element. Having an unmatched pair also removes ringing artifacts that are typically found in matched-pair implementations [112].

There are performance enhancements possible from tuning the blocking strategy used to perform work on the problem [106]. Significant performance gains can be achieved if a good blocking strategy is used. For instance, Weinlich *et al.* showed, when using the NVIDIA GEFORCE 8800GTX, best runtime was seen at 16x16 computational blocks on the GPU . These blocks can be thought of as workgroups for a given volume. If the data requested are relatively localized in the same area of memory for all workers then the chance of data already being loaded is greater.

Xu presents a fast forward-projection algorithm programmed on the GPU through the OpenGL Shading Language (GLSL) [108]. The technique uses a GPU accelerated forward-ray tracing algorithm based on Siddon's algorithm and a voxel driven technique for the back-projection technique. The technique speaks of a number of operations required but no specifics on the actual forward-projection or backward-projection technique for GLSL. Therefore, making comparison difficult. The technique is also not analyzed in terms of noise, error, or convergence. These metrics are important as creating a faster reconstruction is only beneficial if the reconstruction is accurate. Finally, the technique is only built for using one GPU.

Last, for completeness, we again mention Despres's *et al.* presentation, as it does describe a ray-tracing algorithm that achieved a 47x speedup. However, as previously mentioned in the back-projection portion of this thesis, the article lacks implementation details and no subsequent publication was found on the groups' CVs [15].

The use of a multi-GPU environment is currently relatively rare for iterative reconstruction environments because sharing data among GPUs is expensive, and the cost associated with efficient multi-GPU environments is higher. However, Jang's *et al.* work from the Northeastern University realized substantial performance increases from a multi-GPU environment consisting of four GPUs [42]. The difficulty with this work is that there are no implementation details only results. Therefore, this publication is in the same domain as commercial-CT reconstruction algorithms, as there is no way to duplicate or analyze the algorithms.

2.22.3 Summary of Parallel Techniques in Iterative Reconstruction

Parallel techniques in iterative reconstruction are important as they provide a means to reduce computation time to a computationally intense problem. By reducing computation time, the method is made more usable for clinical-CT as scan reconstruction is needed rather quickly. If the computational time is reduced, the method will be used more, as it has been shown to require less than 50% of the number of projections required for FBP [63, 64, 65]. Although the techniques shown are computationally demanding, using parallel techniques can reduce runtime dramatically. There are a number of problems with storage of the large system matrix $\mathbf{A}$ on current hardware, but Siddon's algorithm, or those that provide good indexing schemes to grids, can produce fast ray-tracing techniques that build on the previously mentioned ray-driven forward and backward projection methods. Although multi-GPU environments are not common, we are able to show their applicability in this problem domain and present an efficient implementation in Chapter 6 which dramatically affects runtime.

2.23 Conclusion

The concepts presented in this chapter have been instrumental in the development of a CT-reconstruction environment based on serial, multi-CPU, GPU, and multi-GPU programming. Although the concepts seem relatively complex, programming the methods to avoid common problems such as serial writes, data sharing, data and workload distribution, and the sheer scale of data is notably more complicated. The key conceptual idea one must grasp from the described methods is that FBP is the method of choice on commercial-CT. FBP relies on oversampling the interior frequency region to get reasonable reconstruction of the outer edge, or perimeter, of the reconstruction region because the slice theorem positions the projections in a polar grid layout in frequency space [63, 64, 65]. This oversampling equates to increased radiation exposure to patients [2, 7]. ART type algorithms require less than 50% of the projections when compared to FBP, equating to approximately half of the radiation exposure. The computational problems with ART are often an underdetermined matrix, resulting in an infinite number of solutions, and difficulty constructing efficient forward and back-projection algorithms. Also, the computational model requires a large system matrix $\mathbf{A}$ that is difficult to store on current computers. Furthermore, parallel sparse representations are difficult to produce and precomputing all possible system and scan geometries is near impossible. There are a number of iterative reconstruction publications but they typically lack the detail for reproducing the results. Some key concepts, such as specific ray-tracing techniques and grid interpolation schemes are often missed and only pseudocode is given which is generally not useful. The attractiveness of ART type algorithms is clear, but truly fast reconstruction, when compared to FBP, is still not common. *In the following chapters we will show a thorough analysis of these concepts, implemented on current technology, and show some unconventional methods to solve many of these computational problems in a fast manner.*

Chapter 3

Advanced Reconstruction Environment for Medical Imaging (AREMI)

Through extensive research and development an advanced parallel reconstruction and visualization environment has been created. The environment is specialized, because of the parallel nature of the design, and has a number of computational control parameters, such as the number of cores to use, the use of the GPU, and/or multi-GPU. Likewise, the environment has many algorithmic control parameters, such as relaxation, Field-Of-View (FOV), geometry layout, regularization, padding, filtering, and more. As there is no publication that gives a clear and concise depiction of current implementation details and shortfalls, this thesis provides an original and substantial contribution to the fields of CT-reconstruction, parallel, GPU, and multi-GPU development. Likewise, there are many subtleties concerning reconstruction from raw attenuation values that are never discussed, such as effectively handling of FFS scans, non-parallel beam data, table movement, noise estimation, etc. All of these points have been addressed in this thesis through the development of the Advanced Reconstruction Environment for Medical Imaging (AREMI), this alone being a notable contribution for any research group or person wishing to understand or implement reconstruction algorithms from actual attenuation values from different manufacturers. The following sections detail the structural layout of AREMI, the use and layout of raw-scanner data and understanding scanner geometry. There are a number of implementation details for single-core, multi-core, single-GPU, and multi-GPU templated reconstruction environments for CWBP and algebraic techniques for reconstruction. The fine detail of implementations will be explained and the results of some parameter settings will be outlined.

3.0.1 Data, Runtime, and Analysis

As this is the first chapter where analysis is conducted we introduce some specifics concerning data, runtime, analysis, and iterations.

Data

The data used for all experiments is raw-attenuation values from a Siemens Definition Flash+ 64 or 128 slice scanner. The attenuation values are typically interlaced depending on FFS modes, which will be discussed in Section 3.2. To keep consistent in the document we use 1152 projections, and 736 detector channels unless otherwise noted. This corresponds to a sinogram ($g(l_k, \theta_l)$) with rectilinear coordinates of $k = 736$ and $l = 1152$, respectively. Last, various reconstruction sizes are evaluated and are typically square with $N = M$.

Runtime and Analysis

Wall-clock time was used for all testing where runtime was analyzed because I/O operations can dramatically influence the results. Runtime results were always conducted over five experiments and averaged. The machines used were dedicated machines and the standard deviation was typically negligible. Nonetheless, is depicted with error bars or specified with $\pm value$ after average runtime, when greater than two.

Iterations

Iterations in this document are in respect to *ART*-iteration as seen in Chapter 2.1, Section 2.17. As a reminder, one *ART*-iteration is completed once all rays and all projections have been forward-projected, relaxed, and back-projected.

3.1 AREMI, a Multi-Core, Multi-GPU Reconstruction and Visualization Environment

AREMI is a multi-core and multi-GPU C++ based visualization workspace for evaluation of various reconstruction or visualization techniques. In order to provide universal programming access to AREMI, the Open Graphics Library (OpenGL) was chosen for visualization. As the actual visualization process is not considered demanding, only one CPU-core and one GPU is designated to this task. However, the ability to use more CPU cores or GPUs dedicated to visualization is possible for more demanding visualizations, such as stereo or temporal datasets. The remaining CPU cores and GPUs are used for reconstruction and can be specified at runtime by parameters. The visualization routine is a decoupled visualizer. This means the hardware resources used for visualization purposes can be temporarily suspended in order to facilitate faster or more precise reconstructions. This temporary suspension can result in significant speedups in some cases and, for this reason, is considered an important feature of AREMI. For instance, the ability to show how the projections are acquired and the resulting sinogram built is incredibly important and a real-time visualization of this process is most important, not the runtime. On the other hand, in the reconstruction of an infant at low radiation dose, the highest detail and the quickest time is most important, not the visualization process. These two scenarios justify the ability to decouple the visualization from the reconstruction process.

The computational data flow and hardware utilization diagram is presented in Figure 3.1. The Reconstruction Manager (RM) is the distributor of workload to threads and GPUs. The workload is generally the input data and the output is stored to the output data, but has the ability to work on the output data if required. The RM also has the ability to launch standard threads to work on problems or subproblems if required. The Visualization Manager (VM) has the ability to work on output data or the original data and be decoupled if the GPUs are required for reconstruction. To enable decoupling of the VM a parameter is set at runtime. We found a consistent decrease in reconstruction time when using the decoupled visualization routine with methods that employed the GPU. The justification is rather obvious: the hardware resource used to speed up computation is the parallel architecture of the GPU. When the GPU is used as a graphics pipeline by OpenGL for visualization some of this hardware is in use, and the contention for the hardware is what causes the increased runtime for GPU-based reconstructions. The contention for resources is not limited to only the GPU, without decoupling there is contention for the PCI and main-memory bandwidth. Therefore, reducing the contention for these limited resources can increase throughput and reduced computation time.

3.1.1 AREMI Reconstruction Manager (RM)

The AREMI RM has the ability to delegate tasks or workload to GPUs and specific threads if required. The RM has control parameters that define the size of workload per GPU or thread. These parameters define the workload based on the actual reconstruction algorithm. For instance, CWBP is a super parallelizable algorithm because there is no contention for data and no need for sharing of data or atomicity for reads or writes. For this reason, distributing the workload across GPUs is simpler and breaks down to a near-optimal blocking strategy that can either be set or learned. One hypothesis that we investigated was the use of a mixed CPU and GPU environment to speed-up reconstruction believing the extra computational resources would aid throughput. A job estimate is given to the processing power available on the available CPUs and GPUs by hardware queries. The algorithm then distributes workload to each thread, based on the previous estimate. Threads that use CPU resources for partial computation are launched and alternate threads that will use GPU resources are launched. This technique was not effective as the joining of results from various computational engines was too expensive. Nonetheless, distribution of the workload across many GPUs uses most of the available computational power, in terms of gigaflops, of the machine and can provide significant speedups. This will be shown in Chapter 6, where the runtime analyzed for various algorithms and levels of precision in a multi-GPU environment. As one can see, the runtime is only significantly affected by this use of multiple GPUs if the actual workload is large enough. When the workload can be distributed and achieve full-concurrency, full blocks of workload per sequential

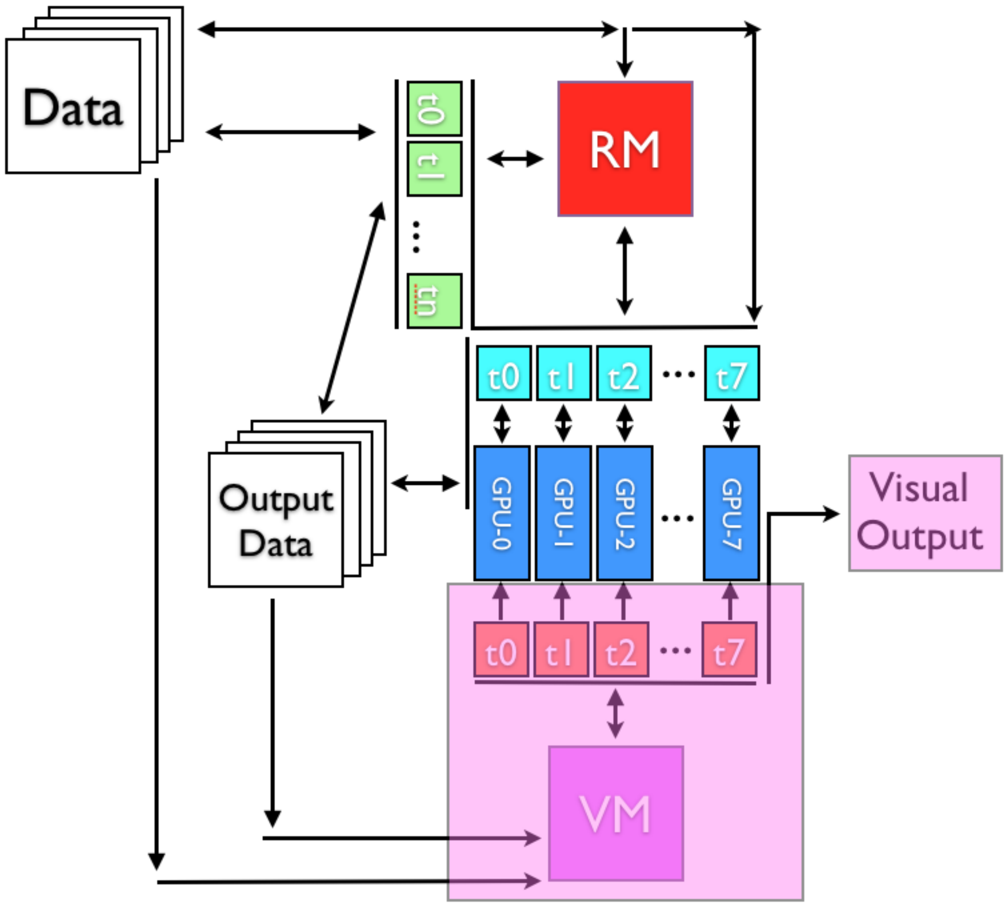


Figure 3.1: AREMI can manage up to eight-GPUs, current limit on one shared memory system. The reconstruction manager (RM) delegates tasks to threads and GPUs. The Visualization Manager (VM) delegates visualization duties to threads and can be decoupled, seen in the perimeter of the VM and the Visual Output, if faster reconstruction is required.

multi-processor (SM), on the GPU will the additional use of CPU resources help reduce the computational runtime. However, the overall runtime was typically slower as merging results was costly. See Appendix A.2 for more details on SMs. When the problem fits computationally on one GPU there is actually a performance decrease by scaling across multiple GPUs and/or CPUs. One can also notice that the error is stable when using a GPU, multi-GPU, and CPU. For this reason, we believe this type of mixed-mode is justifiable for larger problems. We believe this to be the only work that provides analysis of numerical stability given precision, noise, and runtime for various reconstruction algorithms on raw-attenuation values for CWBP, making the results a substantial contribution to research and development as better spatial resolution scanners, with increasingly larger datasets, become more prevalent, as this technique can create faster reconstruction without affecting image quality under our test environment.

Lee *et al.* uses a hybrid CPU and GPU method for different reasons, mainly to find air regions in the scan and omit them from GPU computation through a CPU based K-means clustering algorithm [53]. This is not a true mixed-mode algorithm as the CPU and GPU are not working on the same problem, or even concurrently, as in the mixed-mode CWBP algorithm.

3.1.2 AREMI Visualization Manager (VM)

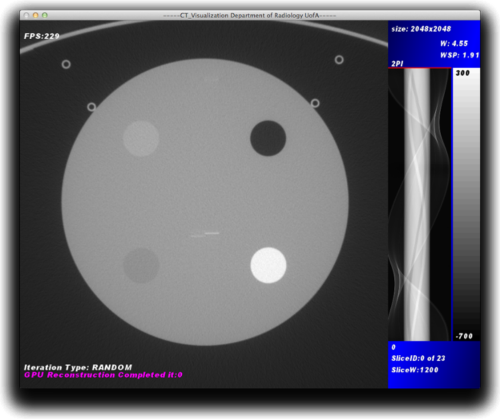
The AREMI VM is important to the visualization of reconstructions. The VM provides an efficient framework to easily add or remove features through keyboard and mouse based input. The VM is important as it provides the ability to adjust image windowing, control image fidelity, dynamic *ART*-iteration control, and adjust scanner parameters which reflect the geometry of the scanner. These concepts will be reviewed in the remainder of this section.

Image Windowing

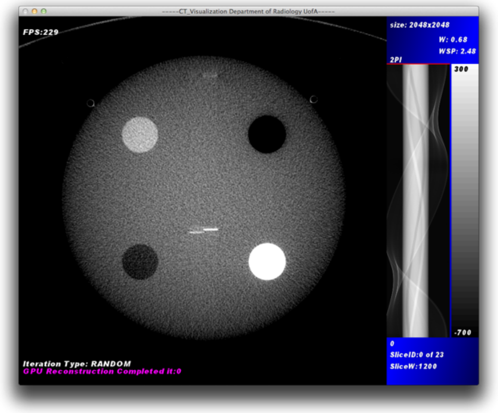
The VM provides a windowing feature that works on the reconstructed image and is essential to clinical-CT use, as windowing provides the ability to conceptualize subtle features or materials based on a specific window-size (WS) and window start point (WSP). Each material encountered by an X-ray attenuates the response. Using image-windowing one can narrow the visual representation to only a limited number of attenuation values. This is important as only a limited amount of grayscale is available on the display medium and materials that are very close in attenuation response could not be seen without this feature. The basic formula to compute the resulting value based on the WS and WSP is given by:

$$f(l) = max(0, min(1, (l - WSP)/WS)). \tag{3.1}$$

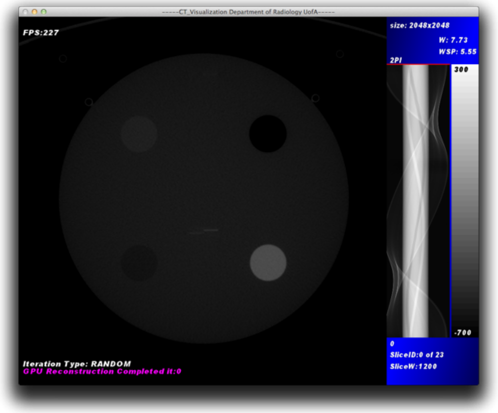
The windowing feature is implemented in the CG shader environment and manipulates 32-bit floating point numbers to the requested WS and WSP via a pass-through vertex program and computational fragment program. The pass-through vertex program simply passes vertices to the fragment


```
------CT_Visualization Department of Radiology UofA------
FPS:229
size: 2048x2048
W: 4.55
WSP: 1.91
2PI
300
-700
0
SliceID:0 of 23
SliceW:1200
Iteration Type: RANDOM
GPU Reconstruction Completed it:0
```

(a) AREMI VM FBP window= 4.55 and windowSP= 1.91


```
------CT_Visualization Department of Radiology UofA------
FPS:229
size: 2048x2048
W: 0.68
WSP: 2.48
2PI
300
-700
0
SliceID:0 of 23
SliceW:1200
Iteration Type: RANDOM
GPU Reconstruction Completed it:0
```

(b) AREMI VM FBP window= 0.68 and windowSP= 2.48


```
------CT_Visualization Department of Radiology UofA------
FPS:227
size: 2048x2048
W: 7.73
WSP: 5.55
2PI
300
-700
0
SliceID:0 of 23
SliceW:1200
Iteration Type: RANDOM
GPU Reconstruction Completed it:0
```

(c) AREMI VM FBP window= 7.73 and windowSP= 5.55

Figure 3.2: Three different window sizes and window start points (WSP) are depicted. Different features are suppressed or enhanced depending on interest.

program after applying the current model-view projection matrix, the graphics pipeline can be seen in Figure A.5. The computational fragment program normalizes the data to the interval [0,1] after applying the WS and WSP. For those GPUs with NVIDIA's High Precision High Dynamic Range (HDR) technology, output of the full size 32-bit number results in HDR output on capable display devices.

Better Image Fidelity

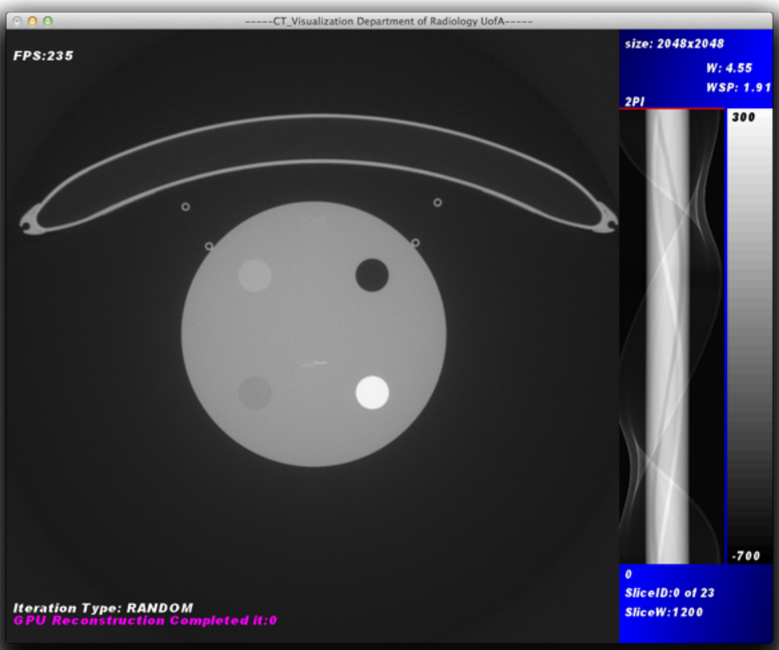
The conceptual development of a mental model of the reconstructions is important and, in order to facilitate better image understanding, the user has the ability to remove toolbars to decrease the level of background light interference with the actual reconstruction. This aids image fidelity because the dynamic range of the reconstructed image is represented better.

Dynamic *ART*-iteration Control

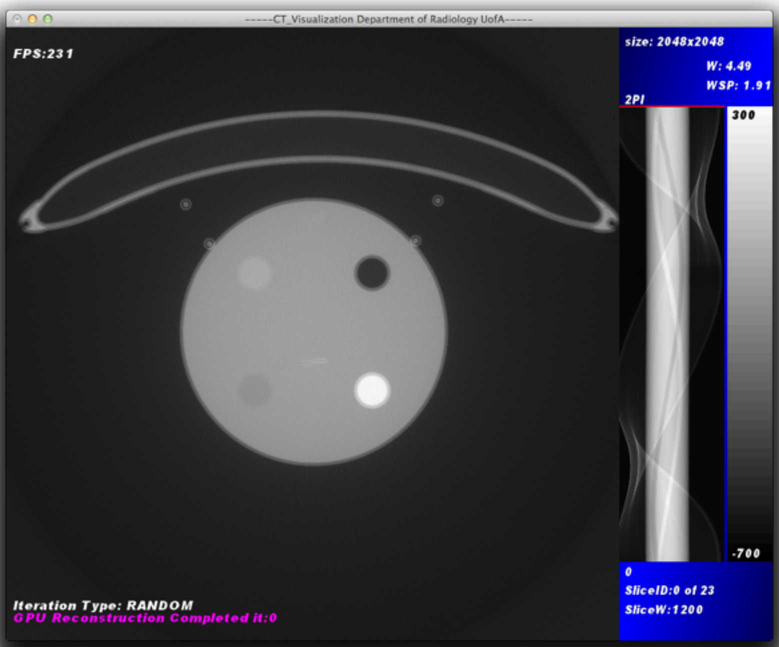
For iterative reconstruction methods, the VM gives the user the ability to add more *ART*-iterations to the reconstruction. This feature is important to real clinical scans because the actual image is not known, and for the underdetermined system, there is an infinite number of solutions; therefore, the exact solution needed is not known and numerous sources of noise exist. For this reason, the user can then specify more *ART*-iterations if required. The ability to use output data in the form of a reconstruction image, and provide more *ART*-iterations to enhance details, is an example of the RM dynamic input parameters.

Control of Scanner Geometry Parameters

The scanner geometry often changes or is somewhat different than what the actual scanner definitions specify. The VM gives the ability to change the isocenter detector bin parameter value on the fly. This feature gives the ability to correct scans or learn the optimal isocenter bin parameter. For instance, Siemens technical document entitled ”Somatom Definition Format Description”, which is not publicly available, but can be requested, specifies the center channel of the detector array to be channel number 367.125 for a Definition Flash CT-scanner and results in a reconstruction as seen in Figure 3.3b. The geometry is obviously not correct, as can be seen by the blurriness on edges and double rings on circular objects, more evident in Figure 3.4b. The user must incrementally adjust, through the keyboard, the reconstruction until the desired calibration is achieved, the calibrated results can be seen in Figure 3.3a and Figure 3.4b. Although the image based reconstruction show a qualitative improvement we also wanted to show a quantitative metric to describe the need for calibration. Therefore, we sampled 24 pixels where a significant transition between air and a highly-attenuated region is seen. We then plotted the results to establish a quantitative metric as seen in Figure 3.4. The area between the estimated-ideal plot and the ISO-calibrated red plot is much less than that of the ISO-uncalibrated green plot. This shows a quantitative metric if one were

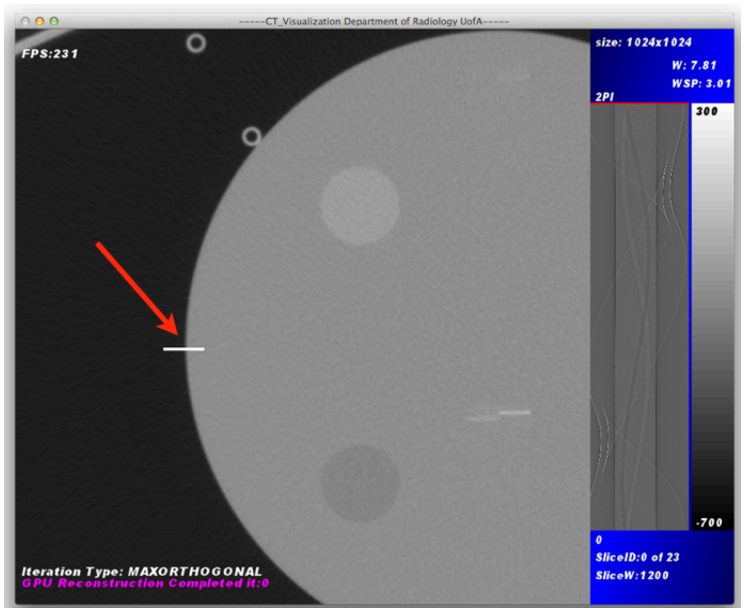


(a) AREMI VM calibrated isocenter detector bin 367.224994

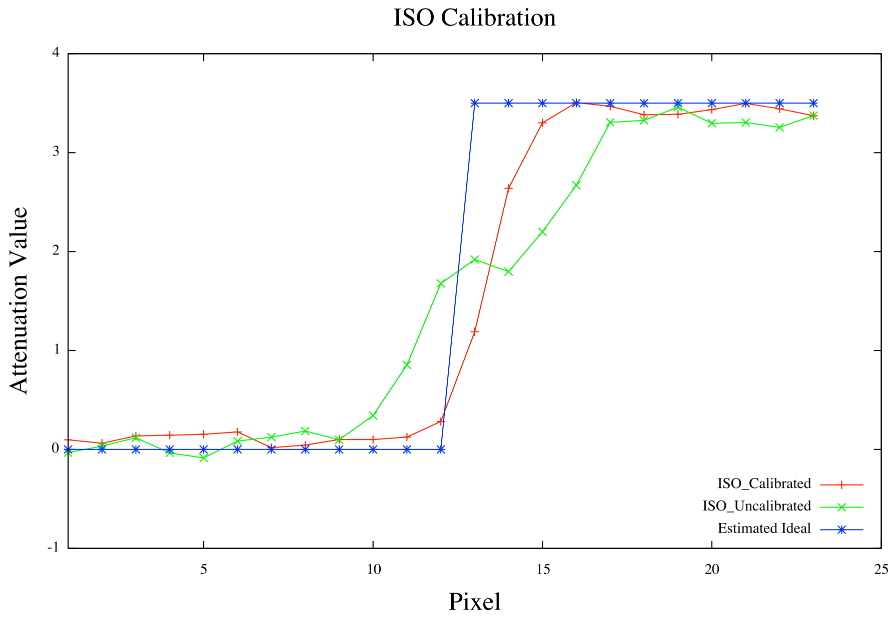


(b) AREMI VM uncalibrated Siemens specified isocenter detector bin 367.125

Figure 3.3: The calibrated isocenter detector bin provides a much sharper reconstruction (3.3a), using the Siemens specified detector bin value provides a reconstruction that is not as sharp and an obvious misalignment is seen (3.3b).


-----CT_Visualization Department of Radiology UofA-----
FPS:231
size: 1024x1024
W: 7.81
WSP: 3.01
2PI
300
-700
0
SliceID:0 of 23
SliceW:1200
Iteration Type: MAXORTHOGONAL
GPU Reconstruction Completed it:0

(a) Pixel Sampling



(b) Graph of Pixel Samples

Figure 3.4: The pixels sampled for (3.4b) are designated by the white line in (3.3b). Ringing is notable in the ISO-uncalibrated reconstruction seen in Figure 3.3b. The ringing is more noticeable when graphed as seen in the green two step plot in (3.4b).

to consider the squared-error between each technique and the estimated-ideal plot as the larger area would cause an increase squared error value.

We found the best method, under AREMI, was to disable any interpolation used during the calibration step as the interpolation can mask the double step found for the uncalibrated example in Figure 3.4b. The manual calibration of the scan is not ideal, but it does have the ability to obtain more precise calibration, as in the previous example, and could eventually lead to the ability to modify other scanner attributes, preventing the need to repeat a scan and the patient being exposed to more radiation. The reconstruction framework developed in this thesis is more than just a method to quickly reconstruct images, it also provides the visualization environment and toolset to work with the reconstruction that is beneficial and necessary. The noted calibration issue is just one use and a subtlety that is often overlooked when describing implementation details.

3.2 AREMI CT Reconstruction Techniques

The following section defines the raw data layout, the description of Flying Focal Spot (FFS) from Siemens [21, 45], and the 2D algorithm, modified from Siddon [94], responsible for ray and slice intersection. The description and considerations of these topics are essential to the understanding and implementation of good reconstruction algorithms on computer hardware and these are investigated below.

3.2.1 Flying Focal Spot (FFS) and Raw Attenuation Data

Let us start with the basic description of the raw data obtained from current fan-beam CT machines. The fan-beam geometry is based on the geometrical layout found in Figure 3.5. There are a number of challenges that must be addressed in the algorithmic implementation, as the attenuation values obtained in raw data format can be interlaced FFS or noninterlaced scan data, as seen in Figure 3.6. The purpose of the FFS mode is to give increased spatial resolution or a thinner slice for reconstruction without requiring the sensor array to decrease detector size and increase sensor density, both being technically difficult for manufacturing. For instance, a Siemens 64-slice Definition Flash+ CT machine has 32 detectors, but, with FFS, is advertised as 64-slice machine as the deflection of the X-ray beam enables finer 64-slice functionality in the longitudinal direction [21]. The actual angular movement of the focal spot is proprietary information and depends on machine specifics yet there are analytical methods for calculating the value for cone-beam CT [45]. The accuracy in the definition of this value enables better reconstruction. For this reason, special attention is needed for this process and methods are borrowed from [45]. The methods for the estimation of the deflection in FFS-mode will likely become more readily available or released from manufacturers overtime. The algorithms implemented in AREMI easily adapt to encompass these natural progressions.

The deflection of the electron beam results in, not only longitudinal movement in the z-direction, but also movement in the radial direction. The radial direction is equal to movement in the projection

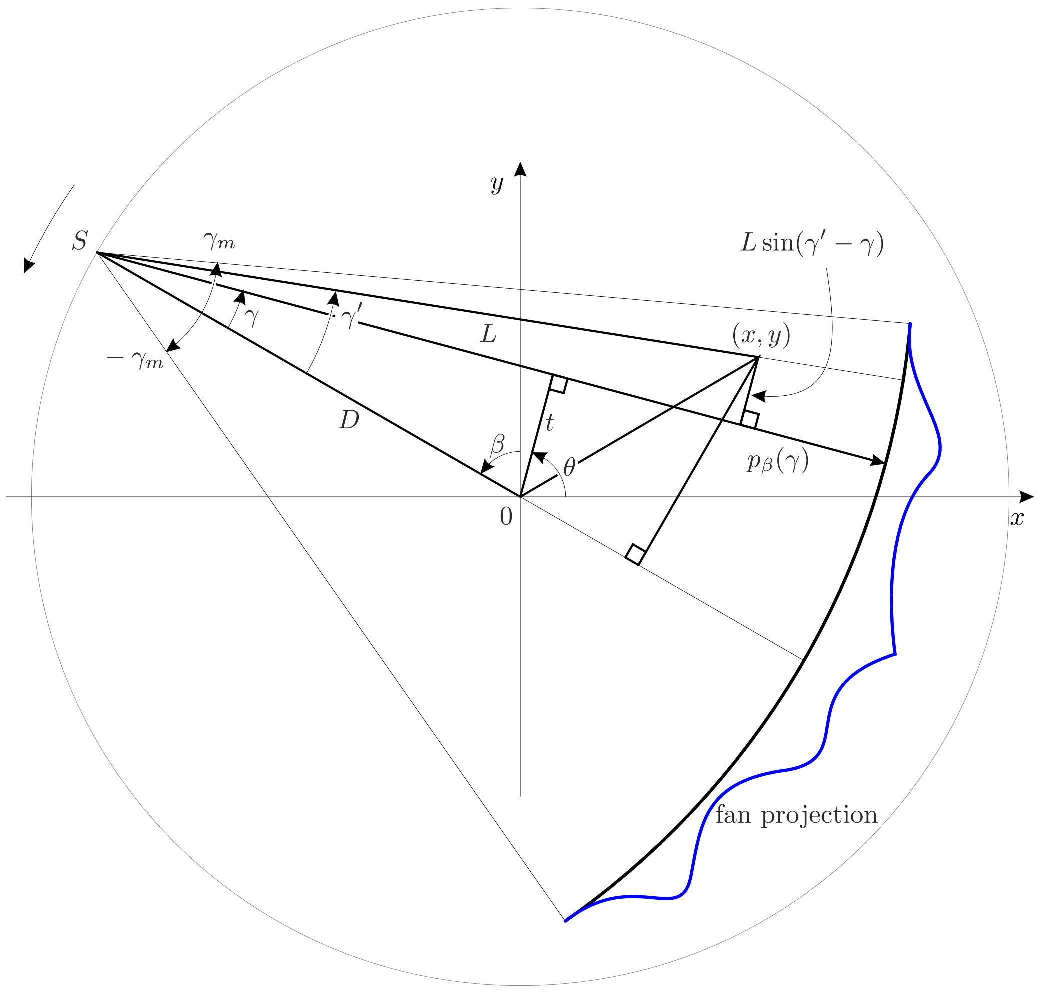
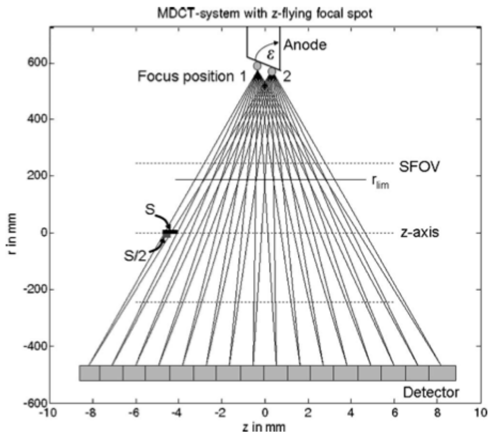


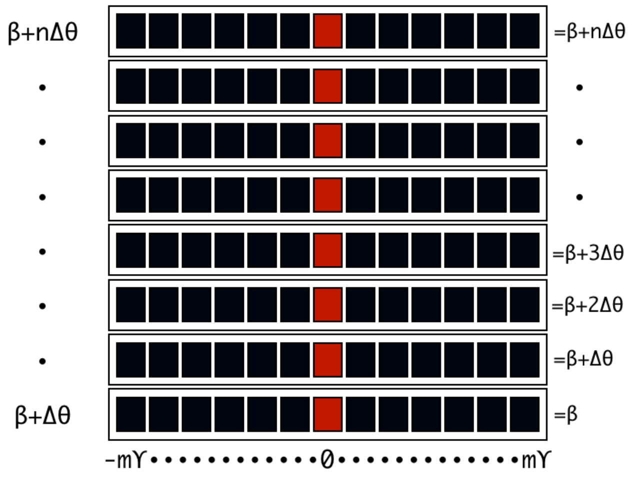
Figure 3.5: Fan-beam geometry (adapted from [47]): the nondiffracting source S emits X-ray radiation, which is dimmed when passing the patient (at the origin of the coordinate system, not shown) and then reaches the detectors on the opposite side. These are arranged on a circular fan, such that each individual detector is uniquely identified via its fan-angle γ. During the scan, the whole geometry rotates around the patient, i.e., the rotation angle β traverses 360°. We identify each ray $p_\beta(\gamma)$ by β and γ.

slices' center detector channel, as seen in Figures 3.7 and 3.8. The raw data layout for non-FFS and FFS modes can be seen respectively in Figure 3.6b and 3.6c. The layout has a simple odd and even indexing, where the odd index equates to a non-FFS scan and the even index relates to a FFS scan.

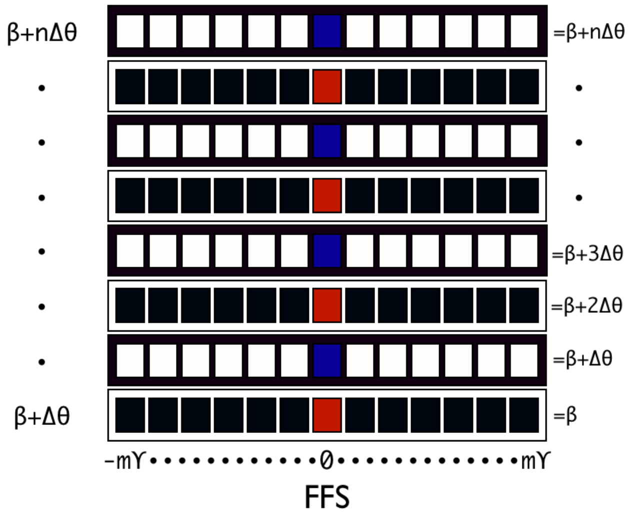
We hypothesized that, for efficient computation, it would be beneficial to reduce the number of conditional statements and branches in the algorithm design for all methods by resorting the data to be contiguous non-FFS projections and contiguous FFS projection before computation on the GPU, as suggested by [13]. However, testing showed that this layout was not beneficial in runtime because of the limited size of the radial detector array for a projection, also known as detector channels. A typical reconstruction of $f(x, y)$ where $m = n = 512$, would create a maximal lateral, or hypotenuse, length of $\sqrt{m^2 + n^2} = 724$ which is approximately the number of required detector



(a) Basic depiction of FFS machine [21]



(b) Raw projection data non-FFS mode



(c) Raw projection data with FFS mode

Figure 3.6: Raw Projection Data with and without FFS. FFS has alternating data layout with blue and white sensor values representing FFS mode and red and black representing non-FFS mode.

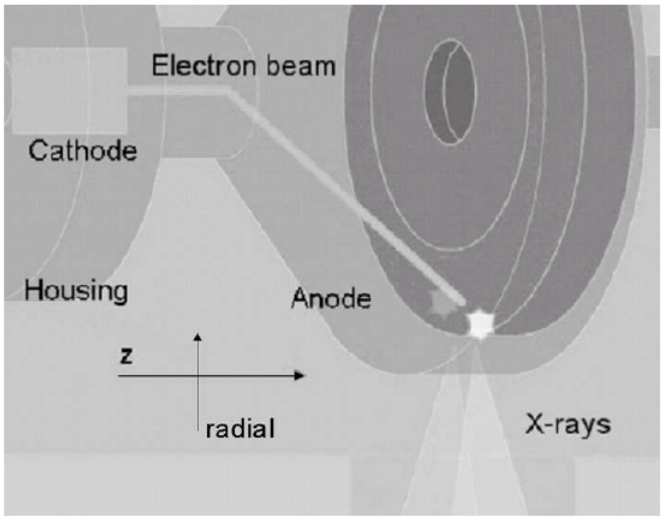
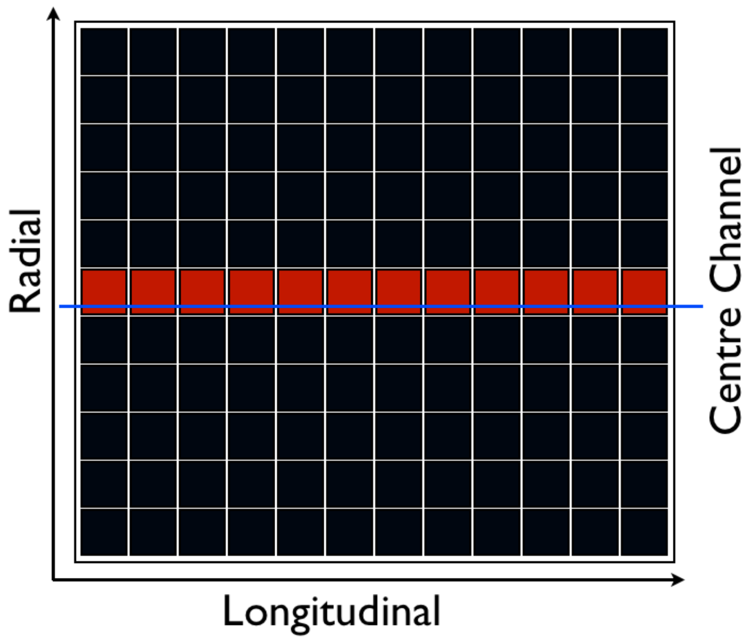
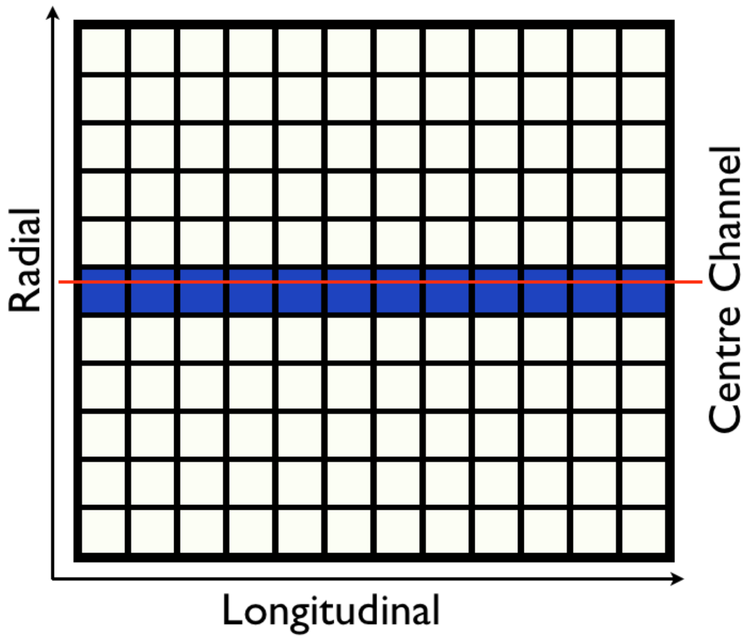


Figure 3.7: Basic depiction of FFS in the longitudinal direction. A continuous electromagnetic deflection of the electron beam results in the focal spot wobbling between two different positions [21]. Sampling at this exact frequency results in higher spatial resolution and the reduction of ring artifacts [21].

channels needed for a complete scan. In the case of the Siemens Definition Flash+ scanner, one has 736 detector channels. For ray-driven forward or backward process 736 threads can be distributed on the GPU maximally for SART, as each thread is associated with one of the 736 detector channels and would be ray-traced through the reconstruction region. The update' formula requires periodic updates to the volume or image after a complete projection update. However, there was no speedup with the data contiguously ordered and one can remove branch statements in the GPU algorithm. The justification for the results is the basis for the discussion in Appendix A.3. Formally, if an algorithm can achieve a fully active *warp*, that is 32 active threads, the algorithm will perform efficiently. However, if the *warp* is not fully active, less than 32 active threads in a *warp*, the algorithm will not be as efficient because of the branch and limited threads on that branch that would be contained in the not fully-active *warp* as there are idle CUDA-Cores. Processing is scheduled in groups of 32 threads in a *warp*. When there are not enough threads on a branch to fill a *warp* it is inefficient because there are idle CUDA-cores. In the simplest form, the data is already well organized, ensuring a near-fully active *warp*. For example, consider the distribution of the 736 threads, one for each ray/detector, on two Nvidia Tesla c2070 GPUs, each containing 448 CUDA-Cores and 14 SMs. An efficient implementation would be to have all 14 SM's on the first GPU, which equates to $14 \times 32 = 448$ threads or ray-tracing forward or backward projectors. There would be a remaining 288 rays needing projection through the volume, equating to 9 SM's active on the second GPU. At any point, all *warps* would be completely active because the conditional branch would never be performed, because all data is specific to one projection and 736 detector channels in non-FFS mode. Only after the GPU computation is completed for a given projection would data be



(a) Non-FFS mode Detector Layout



(b) FFS mode Detector Layout

Figure 3.8: The non-FFS centre channel, blue line, can be seen in (3.8a) and the FFS mode centre channel is seen in (3.8b) by the red line. There is deflection in the Radial direction the centre detector channel changes from 0.625 for non-FFS to 0.125 for FFS. Likewise, there is also longitudinal movement in the z-direction that is not depicted. (This information was gathered from the Siemens Somatom Definition VA30 Data Format Description for Definition Flash+ scanner)

loaded that would activate the conditional branch because of the FFS mode. Only fully-active *warps* are used because all 736 channels are FFS mode data, and the method to calculate the distribution of workload to GPUs ensures this. The switch between non-FFS data and FFS data continues until all *ART*-iterations are completed. Although branches are frowned upon in GPU programming, they are often needed and do not affect the efficiency of the algorithm drastically, provided the algorithm ensures a good data layout and a good balance between thread processing time in each *warp*. Finally, one typically wishes to achieve higher occupancy through scheduling more threads than cores. This is only beneficial if threads are accessing memory frequently. If threads are not accessing memory frequently, as in the forward-projection, scheduling more threads to get higher occupancy does not hide memory latencies.

Considering the FFS mode in FBP and algebraic methods is as follows, the FFS causes a shift in the longitudinal-direction and radial-direction. The exact shift angles are proprietary information from each manufacturer and are usually not available. However, Kachelrie *et al.* provides a description of how to obtain a good estimate to these shifts [45]. Two new symbols will be introduced, αFFS that defines the radial directional shift or channel shift and zFFS that defines the longitudinal shift or detector row shift. As the formal acronym FFS stands for Flying Focal Spot, there is no geometric changes in the definition of the detector in either fan-beam or parallel-beam cases. This can be seen in the three-dimensional derivation of the detector defined by:

$$x_1 = -(\bar{RF} + \bar{RD}) * \cos(\beta + \gamma) + \bar{RF} * \cos(\beta), \tag{3.2}$$

$$y_1 = -(\bar{RF} + \bar{RD}) * \sin(\beta + \gamma) + \bar{RF} * \sin(\beta), \tag{3.3}$$

$$z_1 = \zeta + Tmz, \tag{3.4}$$

where $\bar{RF}$ and $\bar{RD}$ correspond to the distance from isocentre for the source and detector respectively, β defines the angle of the source and γ defines the offset angle of the detector in the channel array, and $\zeta + Tmz$ represents the longitudinal coordinate in the detector array and the table movement, see Figure 2.9 for a definition in the case of the simpler 2D-geometry. The definition of the source is reformulated from [45] and defined by:

$$x_0 = (\bar{RF} + \triangle\bar{R_F}) * \cos(\beta + \triangle\bar{\beta}), \tag{3.5}$$

$$y_0 = (\bar{RF} + \triangle\bar{R_F}) * \sin(\beta + \triangle\bar{\beta}), \tag{3.6}$$

$$z_0 = ZS + \triangle\bar{z}, \tag{3.7}$$

where $\triangle\bar{R_F}$ represents small variation to the deflected focal spot to isocenter, and $\triangle\bar{\beta}$ and $\triangle\bar{z}$ being the deflection angle and length as from [45]. The equations for non-FFS mode, $\triangle\bar{\beta} = \triangle\bar{z} = 0$, and is defined by:

$$x_0 = \bar{RF} * \cos(\beta), \tag{3.8}$$

$$y_0 = \bar{RF} * \sin(\beta), \tag{3.9}$$

$$z_0 = TMz. \tag{3.10}$$

The previous derivations have formally defined a ray's starting and ending point. This is essential for reconstruction methods where the accurate intersection of this ray with a given volume, or slice, defines the preciseness of the reconstruction. The exact method to estimate these changes due to FFS will be described later in, Chapter 5, as this method only provides a good starting point.

3.2.2 Ray-Driven Forward and/or Backward Projection by Siddon's Algorithm

The following section describes Siddon's Algorithm and a slightly faster implementation from Jacob *et al.* [94, 41]. Specifically, the algorithms define the ray intersection with a volume in 3D or slice in 2D and the associated numerical indexes, i, j, k for a given volume, or i, j for a given slice. The technique is incredibly efficient because it does not visit each pixel or voxel as with other interpolation techniques. The ability to not visit each pixel, and consequently test it based on the interpolation method used, significantly enhances the forward and backward ray projection process. Jacob's *et al.* describes an algorithm that does not require an ordered list of index points, as in the classical Siddon's algorithm; rather, the technique calculates only the first $i, j,$ and k index points then incrementally updates the positions. The enhancements result in a 7.5x speedup when calculating radiological paths and a 5x speedup when considering total reconstruction time [41]. Jacob's *et al.* [41] provides a novel improvement to Siddon's algorithm [94], which is related to Bresenham's line algorithm [8]. As the Siddon-type exact solution is considered the *gold-standard* for interpolation of ray intersection with a volume or grid [19], the following addresses the description of the algorithm in a fan-beam 3D reconstruction that includes FFS mode and shows subtle performance and image quality-based enhancements to the technique. The use of the method and derivation of the technique has never been explicitly given for a fan-beam 3D reconstruction incorporating FFS. This technique is formulated for parallel computation as it has the ability to reformulate the computation for a specific ROI or rays. The break-down of the algorithm is formulated into three parts that would all be contained in one *kernel* call or one thread's serial execution on the CPU. The technique given is based on an implementation in any shader-based pipeline, OpenCL, or CUDA.

Derivation of Ray Endpoints with FFS

The pseudocode presented in Algorithm 3.1 represents the definition of the end-points of a ray and shows the use of FFS modes in both the radial and longitudinal directions, seen in lines 7-12. The definition is for that of fan-beam geometry, as lines 10-11 show the required angle γ is needed for

Algorithm 3.1 Enhanced Siddon's 3D fan-beam ray-tracing algorithm to define ray endpoints pseudocode [94]. The parameters are defined as β the angle of the source, γD[] angle of detector cell in the XY-coordinate, zD[] angle of the detector cell in the Z-coordinate, {x0,xN,y0,yN,z0,zN} numerical endpoints of the ROI for reconstruction, {numX,numY,numZ} are the number of elements in each direction respectively, {voxX,voxY,voxZ} are the voxel size for each dimension respectively, $\triangle\bar{R_F}$ FFS radial deflection, and $\triangle\bar{z}$ longitudinal deflection.

```
 1: procedure BACKWARDPROJECTIONENDPOINTS
 2:     $ix := blockDim.x * blockIdx.x + threadIdx.x$        ▷ channel id in radial direction
 3:     if $ix > nChannels$ then
 4:         return
 5:     end if          ▷ ensures when nonPow2 channels, the extra threads exit gracefully
 6:     $projVal := projRay[ix]$                          ▷ hide load of projection value
      $\rightarrow$ Definition of the Source under FFS, endpoint 1.
 7:     $X1 := (RF + \triangle\bar{R_F}) * cos(\beta + \triangle\bar{\beta})$
 8:     $Y1 := (RF + \triangle\bar{R_F}) * sin(\beta + \triangle\bar{\beta})$
 9:     $Z1 := ZS + \triangle\bar{z}$
      $\rightarrow$ Definition of $\gamma$, loading should be avoided. If possible, do not load, as in the second
    definition of $\gamma$. Not always possible,
10:     $\gamma := \gamma D[ix]$
11:     $\gamma := (halfDetector - ix) * machineSpecs.angleGridSeparationRadians$ $\rightarrow$ Definition
    of the Detector under FFS, endpoint 2.
12:     $X2 := RF * cos(\beta + \gamma)$
13:     $Y2 := RF * sin(\beta + \gamma)$
14:     $Z2 := TMz$
15:     $xu := yu := zu := 1$
16:     if $X1 > X2$ then
17:         $xu := -1$
18:     end if
19:     if $Y1 > Y2$ then
20:         $yu := -1$
21:     end if
22:     if $Z1 > Z2$ then
23:         $zu := -1$
24:     end if
25: end procedure
```

precise definition of the detector. These values are pre-calculated and stored in an array. However, in line 11 the value is calculated. In the basic example, either method works, but the calculated method is much faster as no loading from memory is required. Therefore, if possible calculate and if the calculation is too complex store the value. The small code-snip has subtle, but important, ordering of calculation and loads that can realize substantial improvements in runtime versus a typical implementation.

First, the pseudocode can be used by a single or multi-GPU system, as the number of projections distributed to a given kernel call, or GPU, is dictated by the calling thread. Second, by specifying the memory load of the ray value early, namely the first call, one actually hides the load behind the computation of numerical values. This means, the value is not required initially after the load, and; therefore, much computation can be completed during the latency of the load. Finally, in the third code snippet in Algorithm 3.3, the value loaded is used. This technique, of early load, is rather simple, but prevents all threads waiting for the load of required data, consequently preventing idle stream processors waiting for data to continue computation. Specifically, testing showed a 2% reduction in computation time on a multi-GPU SART algorithm, ray-driven matched pair with five *ART*-iterations using dual Nvidia Quadro 4000 GPUs and 1024x1024x32 volume. One initially would postulate that the Nvidia CUDA 4.0 compiler would order this load in a more respective manner, to prevent the idle stream processor waiting for the data, but testing shows consistent improvement using this technique of early load. One should also note, although the forward and back-projection ray-driven techniques are similar, the early load can only be realized in the ray-driven back-projection process as the loaded projection value is smeared over pixels encountered during the ray-tracing algorithm. In the forward-projection process, seen in Algorithm 3.3, the loads are completed when needed because, as the ray visits a pixel, the value encountered is needed for the ray-sum. The direction of the ray, in terms of x, y, and z, is important as it dictates the increment value for the indices of the 3D matrix representation of the volume. Lines 15-24 define the direction of the ray in terms of x, y, and z. For instance, if $X1 > X2$, one knows when tracing the ray through the volume from $X1 \rightarrow X2$ the associated index through the volume will cause the index pointer to move down not up; therefore, requiring a negative value for the xu and hence the redefinition of the increment value to -1. Only if $X1 < X2$ would one use a position xu, as the $x - index$ would grow, not decrease, as the ray moves through the volume from $X1 \rightarrow X2$.

Calculation of Initial Indexes and Incremental Updates

The incremental description of backward-projection continues, and the main differences from Siddon's algorithm and Jacob's *et al.* enhancements are now apparent. There are some important considerations in the definition. First, as the pseudocode is formulated for the GPU, branches should be avoided when possible as they cause divergence in the algorithm and *warps* that are possibly not fully active, see the CUDA guide for more details [13]. Careful initialization of variables to avoid *if*

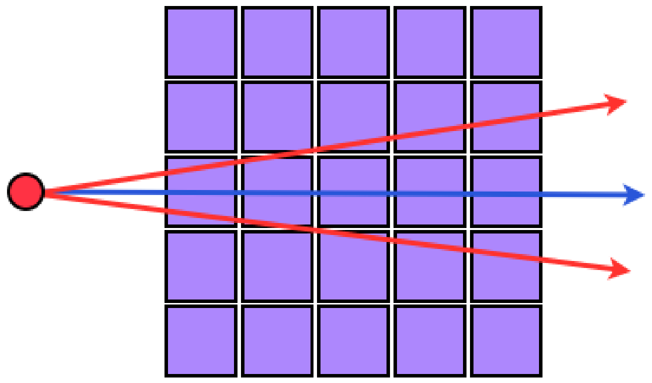


Figure 3.9: Although all rays have equivalent length, the number of pixels intersected is not equivalent. The red rays intersect six pixels and the blue ray only intersects five.

and *else* statements is important as it decreases the likelihood of threads diverging. Obviously this is not always possible. Second, there are three choices for ray-length, seen in line 18 from Algorithm 3.2, mainly the length of the ray can be considered from source-to-detector and constant for all rays because of the detector-array is on an arc and has equal distance to the source and is what is used in the pseudocode. Two alternative definitions for length that can and should, be considered and the resulting conclusions of these alternative ray lengths are now explained. The first alternative ray length can be based on pixel interactions as in Toft's dissertation [97]; therefore, if we consider past points, there is no guarantee that the same amount of pixels are encountered in the discrete representation. For instance, consider Figure 3.9 which shows that the number of pixel interactions are not equivalent among all rays, only the geometric length of each ray is equivalent. The second alternative ray length to be evaluated is the length of the ray with the ROI. This length is obviously not constant among rays and also varies depending on the angle of the source. Likewise, the number of pixel interactions is not constant and varies, based upon the source angle. The increased computational complexity is based on the requirement to store/write the number of pixel interactions in a separate array, this requires another read from memory, as well as a mutually-exclusive writing operation. The pseudocode given is for the basic ray-length, as insightful ray-lengths can lead to faster convergence and/or less noise in the reconstruction under our implementation, but has been found to be very subjective, and thus no absolute conclusion should be drawn.

Ray-based Walk Through Region of Interest (ROI)

The ray-based walk through the ROI, whether square or circular as seen in Figure 2.15, and the GPU is really what makes the computationally demanding iterative methods feasible. The code snippet in Algorithm 3.3 demonstrates the walk of the ray through the ROI. Like the previous two code snippets, there are some important points to be discussed. First, the basic task is the walking

Algorithm 3.2

```
procedure BackwardProjectionEntryExitPnts
    $\alpha_{x0} := \alpha_{y0} := \alpha_{z0} := 0$
    $\alpha_{xN} := \alpha_{yN} := \alpha_{zN} := 1$
    $\alpha_{xu} := \alpha_{yu} := \alpha_{zu} := 0$
    if $X1! = X2$ then
        $\alpha_{x0} := (x0 - X1)/(X2 - X1)$
        $\alpha_{xN} := (xN - X1)/(X2 - X1)$
        $\alpha_{xu} := voxX/abs(X2 - X1)$
    end if
    $\rightarrow$ similar for Y and Z
    $\alpha_{xmin} := min(\alpha_{x0}, \alpha_{xN})$; $\alpha_{xmax} := max(\alpha_{x0}, \alpha_{xN})$;
    $\rightarrow$ similar for Y and Z
    $\alpha_{min} := max(\alpha_{xmin}, max(\alpha_{ymin}, \alpha_{zmin}))$
    $\alpha_{max} := min(\alpha_{xmax}, max(\alpha_{ymax}, \alpha_{zmax}))$
    if $\alpha_{min} < 0||\alpha_{max} > 1||\alpha_{max} < 0||\alpha_{max} > 1||\alpha_{max} > \alpha_{min}$ then
        return                                    ▷ ray does not intersect region
    end if
     $\rightarrow$ calculate distance of ray
    length := sqrt((X2-X1)*(X2-X1) + (Y2-Y1)*(Y2-Y1) + (Z2-Z1)*(Z2-Z1))
    $\rightarrow$ only the pseudocode for the definition of x is included
    $\alpha_x := \alpha_y := \alpha_z := \alpha_{min}$
    $\alpha_x + = \alpha xu$; $\alpha_y + = \alpha yu$; $\alpha_z + = \alpha zu$;
    $minX := minY := minZ := 0$
    if $X1 < X2$ then
        if $\alpha_{min}! = \alpha_{xmin}$ then
            $minX = \lceil((X1 + \alpha_{min} * (X2 - X1) - x0)/voxX)\rceil$;
            $\alpha_x := (minX * voxX - X1 + x0)/(X2 - X1)$
        end if
    else if $X1 > X2$ then
        if $\alpha_{min}! = \alpha_{xmin}$ then
            $maxX = \lfloor((X1 + \alpha_{min} * (X2 - X1) - x0)/voxX)\rfloor$;
            $\alpha_x := (maxX * voxX - X1 + x0)/(X2 - X1)$
        else
            $maxX := numX$
        end if
    else
        $maxX := \lfloor(X2 - x0)/voxX\rfloor$
        $minX := maxX + 1$
        $\alpha_x := \alpha max + voxX$
    end if
    $\rightarrow$ similar for Y and Z
    $\alpha_{mid} := (min(\alpha_x, min(\alpha_y, \alpha_z) + \alpha_{min})/2$
    $xIdx := (X1 + \alpha_{mid} * (X2 - X1) - x0)/voxX$
    $\rightarrow$ similar for Y and Z
end procedure
```

through indices that represent the walk of the line from $\alpha_c \rightarrow \alpha_{max}$. A naive implementation would simply write the value of the projection to reconstructed image pixel. However, this results in poor reconstruction quality. Two enhancements are possible. First, the variable α_D defines the length of intersection with the pixel. Therefore, as seen in line 28, the projection value is scaled by this length of intersection. This technique more closely represents the actual area-based detector and the region a ray intersects from beginning to end from source to detector. The second improvement is the application of a heuristic, namely using a Hamming filter on the ray-length as seen in Figure 2.17, where the Hamming filter for the circular reconstruction region is scaled by the typical Hamming filter function $w(n) = 0.54 - 0.46 * \cos((2\pi n)/(N-1))$. This filter effectively enhances the contribution to the points closest to isocenter and deemphasizes those points towards the perimeter of reconstruction.

Algorithm 3.3

```
1: procedure BACKWARDPROJECTIONRAYWALK
2:     projVal *= length;
3:     $\alpha_c := \alpha_{min}$
4:     while $\alpha_c <= \alpha_{max}$ do
5:         if $xIdx < 0||xIdx > numX - 1||yIdx < 0||yIdx > numY - 1||zIdx < 0zIdx > numZ - 1$ then
6:             imageIdx := -1
7:         else
8:             imageIdx := numX*numY*zIdx + numX*xIdx + yIdx        ▷ 3D index in volume
9:         end if
10:        $min := min(\alpha_x, min(\alpha_y, \alpha_z))$
11:        if $min == \alpha_x$ then
12:            $\alpha_D := \alpha_x - \alpha_c$        ▷ distance of ray intersect with pixel or voxel
13:            $xIdx := xIdx + xu$
14:            $\alpha_c := \alpha_x$
15:            $\alpha_x := \alpha_x + \alpha_{xu}$
16:        else if $min == \alpha_y$ then
17:            $\alpha_D := \alpha_y - \alpha_c$
18:            $yIdx := yIdx + yu$
19:            $\alpha_c := \alpha_y$
20:            $\alpha_y := \alpha_y + \alpha_{yu}$
21:        else
22:            $\alpha_D := \alpha_z - \alpha_c$
23:            $zIdx := zIdx + zu$
24:            $\alpha_c := \alpha_z$
25:            $\alpha_z := \alpha_z + \alpha_{zu}$
26:        end if
27:        if $imageIdx > -1$ then
28:            $atomicAdd(reconImage[imgIdx], \alpha_D * projVal)$
29:        end if
30:    end while
31: end procedure
```

Summary of Implementation of Siddon's Algorithm

The previous sections represent the computational involvement and efficient programming of a ray-driven forward and backward-projection through a volume in 3D or an image in 2D. The technique is formally based on Siddon's algorithm [94], uses some of the enhancements from Jacob's *et al.* [41], and *our noted refinements to further accelerate the ray-projection process in a multi-GPU environment.* These techniques substantially reduce runtime in our implementation when applied and compared on current hardware, as seen in Section 6.2. When compared to complete techniques, such as those from Toft [97], we see a huge decrease in run-time. The implementation included is for that of the backward-projected ray, as it is more complicated and requires an atomic addition operation because many threads may write to the same location. This is evident because many rays may intersect a given pixel. This situation is more likely, as the distance of a pixel-to-source decreases, as seen in Figure 3.9 where three rays intersect the pixel closest to the source and, as the distance from the source increases, the likelihood of multi-ray pixel interaction decreases. The number of rays contributing to a given pixel value in the back-projection process is important and must be accounted for. This value is realized in the denominator of the back-projection equation, mainly $\sum_i \mathbf{a}_{ij}$, and should not be omitted. The forward-projection is similar, luckily there is no need for atomic operators that slow the computation and the extra storage requirements are not needed to save the number of ray intersections, remembering that atomic operations require parallel threads to behave in a serial manner. Mainly line 6 is not required from Algorithm 3.1 and line 28 from Algorithm 3.3 is replaced with the summation or accumulation value $accum = accum + \alpha_D * image[imgIdx]$. After the *while* loop in Algorithm 3.3 is completed, the accumulation value is written to projection array at the specific $\gamma[ix]$ index, $proj[\gamma[ix]] = accum * rayLength$ which represents the forward-projection and estimation of the ray through the volume or image. *An official definition has never been given for the exact method to represent the ray projection for fan-beam geometry layout with FFS, and the subtleties required for efficient computation on a GPU or multi-GPU environment and; hence, is a significant contribution to the research community.* The formal evaluation of these techniques will be done in Chapter 6. AREMI in itself is a significant contribution, as *the environment demonstrates the use of multi-core and multi-GPU programming in an efficient way.*

Through the use of AREMI, algorithmic enhancements, and the many computational resources available, a huge speedup can be realized for CWBP, iterative techniques, filtering, and visualization and will be explicitly shown in Chapter 6. The speedups in these areas means faster results are obtained and qualitative and quantitative analysis based on parameters can be completed quicker. Likewise, in a clinical settings quick results are needed for maximal throughput and efficiency of resources.

3.3 Summary

AREMI represents a framework that is suitable when large computational resources are required for more computationally expensive CT reconstruction methods. This framework uses non-traditional HPC hardware resources to provide HPC level computational performance. *We presented an open implementation of the* AREMI *framework and a detailed, enhanced FFS-mode Siddon's algorithm for the GPU that will be shown to perform well in a multi-GPU environment.* This chapter was required to conduct the analysis in the following chapter. In the next chapter we will demonstrate a quantitative metric to measure noise, establish some criteria for determining convergence for iterative algorithms, and finally detail effects of different reconstruction regions in terms of convergence and noise.

Chapter 4

Noise, Convergence, and Projection Ordering

4.1 Evaluation of Techniques

The following chapter is used to define a quantitative metric for evaluation of reconstructions in AREMI. First, a metric is established to measure the noise in a homogenous region of a reconstructed image. The method proposed has the ability to isolate the noise in specific frequencies. Second, we demonstrate the limited range of Hounsfield Units (HU), and how with raw attenuation values one can extend this range to promote better image fidelity on capable devices. We also investigate what convergence means to iterative algorithms and a method to realize when convergence has been reached. Last, we illustrate the importance of projection ordering to iterative methods in a qualitative and quantitative study with numerous projection ordering techniques. These methods are all used in Chapter 6 where further experimental analysis is conducted. We remind the reader of the definition of the data used, runtime, and analysis detailed in Section 3.0.1 as they are not explicitly repeated in this chapter.

4.1.1 Establishing a Quantitative Noise Metric

The ability to estimate noise in a scan is useful as it provides a quantitative metric for the reconstruction algorithm. Two techniques will be described: first the use of the noise power spectrum (NPS), and the peak signal-to-noise ratio (PSNR) in a discrete setting. Typically PSNR is used in signal compression as a metric to evaluate how closely the compressed signal represents the true signal. However, it can be used in alternate settings. However, as one is not reconstructing from a known image, the noise and quality of reconstruction is subjective and the ability to use the PSNR is not possible for raw data, as no known image is available and; therefore, PSNR formula is not well defined. However, as the common practice is reconstructing simulated data where a known image is available, the use of PSNR is practical and can be useful and is mentioned for that reason. As we target reconstruction from raw attenuation values, the utilization of the NPS metric is useful as no

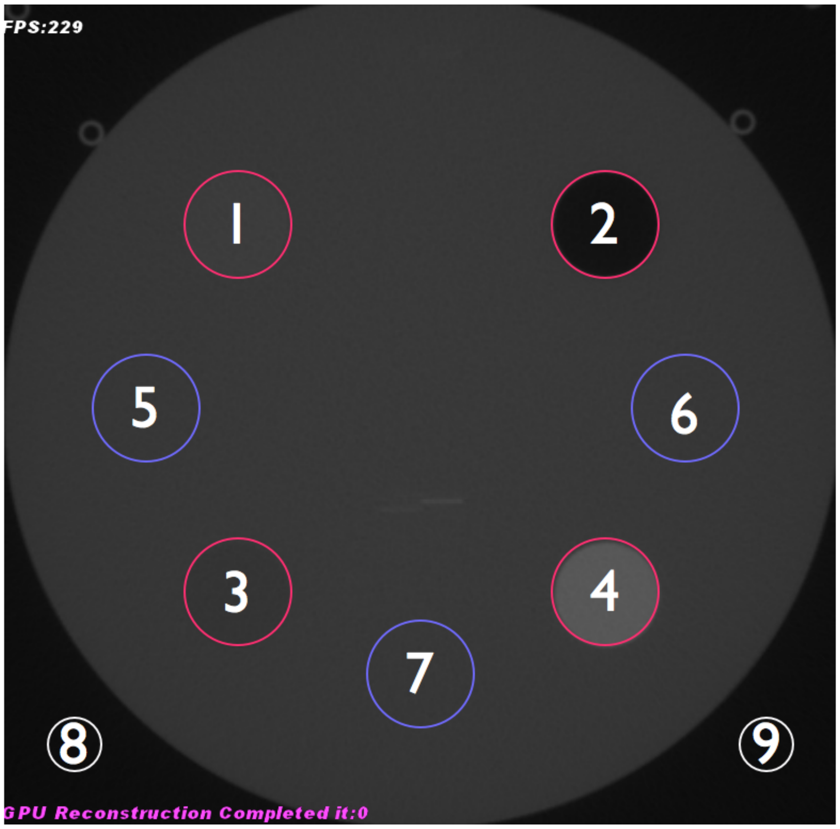


Figure 4.1: The estimation of noise is accomplished through the use of nine homogenous locations in the scan and varying radius for homogenous section. Two smaller outer homogenous areas, (8,9), are specified by the white circles and represent air. Seven internal homogenous sections are used, the three depicted in blue, (5,6,7), represent an internal homogenous area and the red circles, (1,2,3,4), represent four different internal homogenous areas. The square sample of each homogenous section is taken, typically 16×16, in order not to sample the boundary of the homogenous area and corrupt the estimated-NPS.

known image is required. By exploiting homogenous areas in the scan, one can estimate the noise in a relatively precise manner.

Noise Estimation

A phantom, as seen reconstruced in Figure 2.3c and physically in Figure 4.2, is desired as it has several known homogenous areas, specifically an ACR accredited phantom was used. The phantom represented a good tool to perform qualitative analysis because it contains many regions of known composition and geometrical layout, see Appendix B.5 for more details. When using real human scans it is difficult to perform qualitative analysis on the reconstructions because the known composition is not available. Figure 4.1 depicts nine homogenous regions: the outer two smaller white circles represent air; the three blue inner circles represent the same homogenous properties; and the four red circles represent unique homogenous areas. The homogenous regions provide a

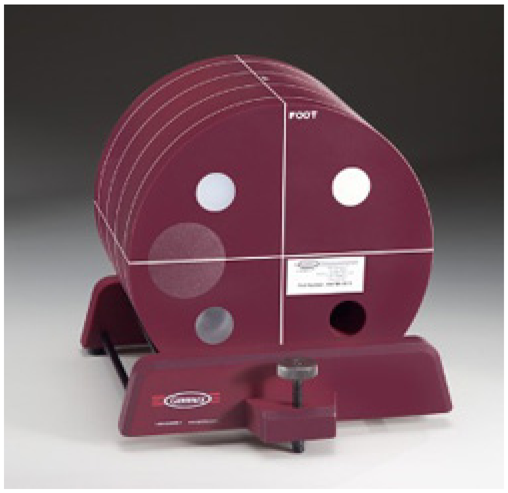


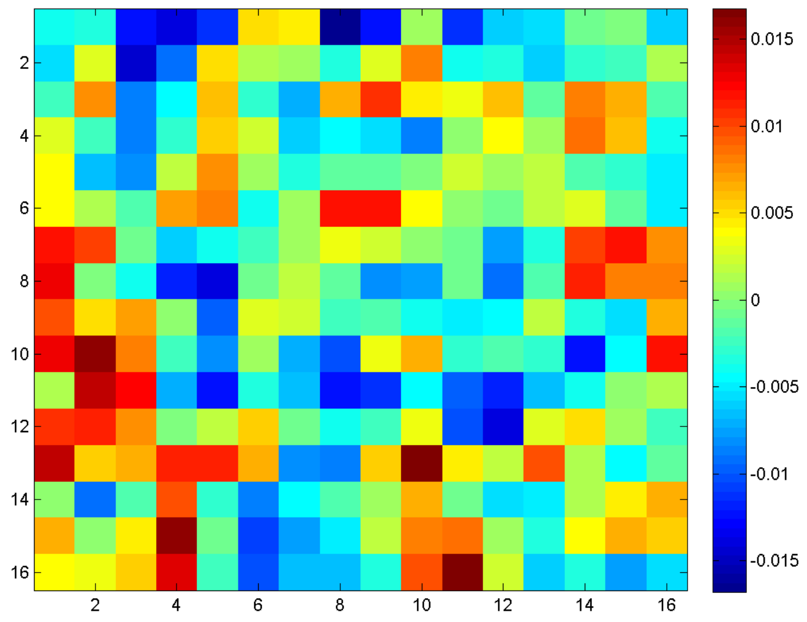
Figure 4.2: An ACR accredited phantom was used for scanning.

stable, known value, which can be used for noise analysis through the use of NPS. Using an ACR-accredited phantom, homogenous areas of the scan are known to have various compositions, such as solid water, acrylic, and polyethylene to name a few. From the nature of CT scanners they are quantitative devices that measure the amount of attenuation in a specific detector channel [56]. This attenuation reflects the physical properties of the material within the area scanned and, using this knowledge, one can construct relatively accurate quality scores and noise estimates.

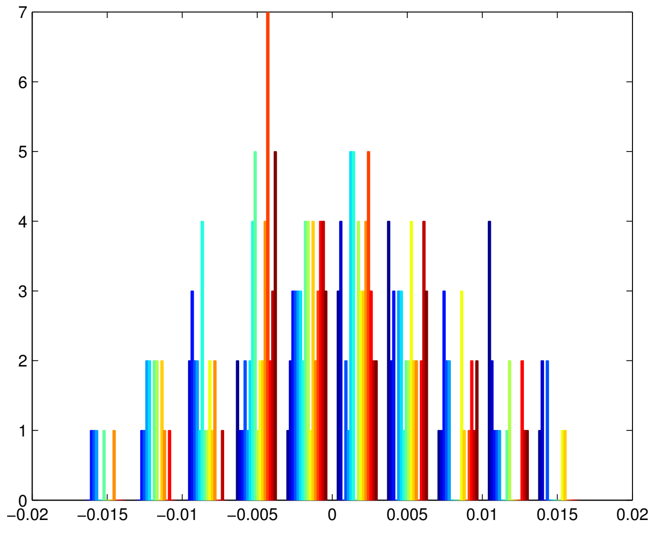
We now introduce methods from Kijewski and Judy that illustrates the use and validity of NPS in CT [51], when having several scans of the same object. However, one typically does not have several hundred images to do the analysis on. For instance, common practice would be to estimate the NPS on a single image with one or more ROI. The NPS analysis can provide a valid image-standard or quality-metric that describes the amount, and frequency, of noise contained in a reconstructed image, as seen in Figure 4.4. However, the NPS provides noise estimates at specific frequencies, and one generally would like one value for comparison purposes. Therefore, as the noise is always positive, the summation of NPS frequency values provides a good metric and similar to the L2-norm. Nonetheless, we have access to frequency-specific noise information that can be evaluated and seen in Figure 4.4. We use the simplified discrete NPS formulation, represented by:

$$\tilde{W_d}[k,l] = \frac{x_0 y_0}{N_x N_y} | \sum_{m,n} \triangle \tilde{a}[m,n] e^{-2\pi i(km/N_x + ln/N_y} |^2. \tag{4.1}$$

This equation appears complex but is relatively simple, as it is the 2D Fourier transform of the image components scaled by the pixel size of the x and y, divided by the number of components. In terms of programming, one can easily formulate Equation 4.1 into the pseudocode found in Algorithm 4.1.



(a) ROI 1 for Estimated-NPS

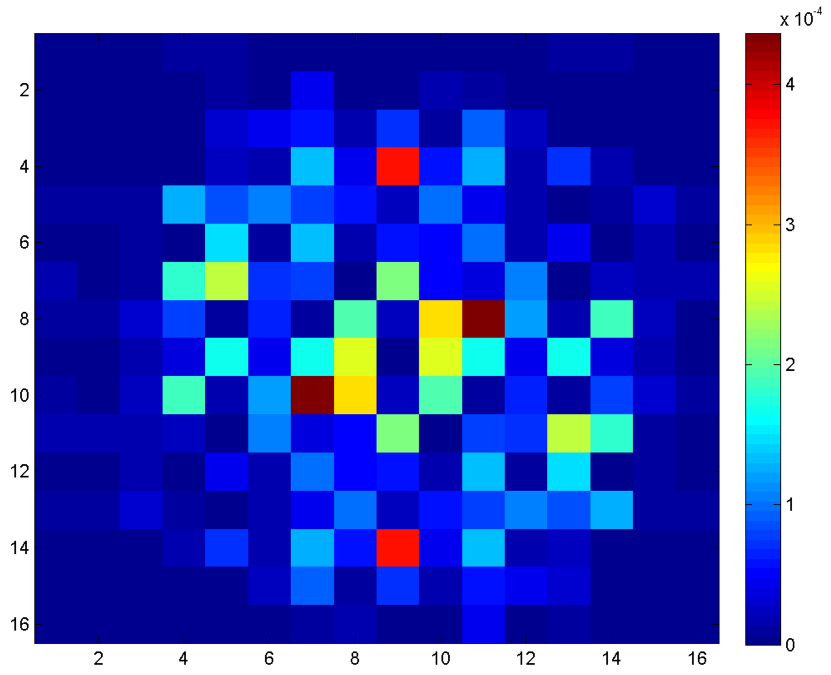


(b) Histogram for ROI

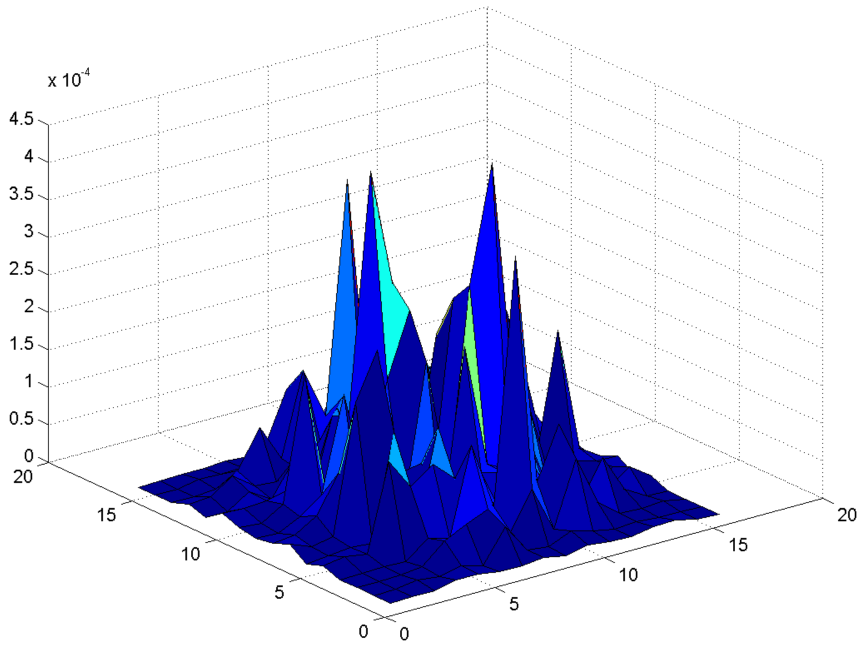
Figure 4.3: (4.3a) depicts the ROI and (4.3b) shows the histogram of the ROI.

Algorithm 4.1 Estimated-NPS for Homogenous ROI

```
1: procedure ESTIMATENPS
2:     $scale := (voxSizeX * voxSizeY)/(numX * numY)$
3:     $NPS := absf(powf(fft2d(image), 2)) * scale$
4: end procedure
```



(a) Estimated-NPS of ROI 1 - DC Centered



(b) Estimated-NPS of ROI 1 - DC Centered 3D

Figure 4.4: (4.4a) and (4.4b) show the 2D and 3D DC component centered. The specific noise at a given frequency can easily be seen.

Table 4.1: DICOM Reconstruction from Siemens – Estimated-NPS for ROIs

Dicom Reconstruction Techniques, results NPS units^2	
Reconstruction Technique	Siemens DICOM
ROI 1	0.00000052
ROI 2	53.5977
ROI 3	69.1033
ROI 4	0.000000511
ROI 5	66.4297
ROI 6	63.0912
ROI 7	47.0969
ROI 8	0.00000004
ROI 9	0.000000014

However, an even simpler version can be derived:

$$\begin{aligned} NPS = absf(powf(fftd(\text{image}), 2)) * \text{scale}, & & (4.2)\\ x' = (x * x - y * y) &= x^2 - y^2, & (4.3)\\ y'i = (x * y + y * x) &= 2xy, & (4.4)\end{aligned}$$

where, $z = x + yi$ and $|z| = (x^2 + y^2)^{1/2}$, therefore substituting x' and $y'i$,

$$\begin{aligned} &((x^2 - y^2)^2 + (2xy)^2)^{1/2} & & (4.5)\\ &((x^2 - y^2)(x^2 - y^2) + 4x^2y^2)^{1/2} & & (4.6)\\ &(x^4 + y^4 - 2x^2y^2 + 4x^2y^2)^{1/2} &= (x^4 + y^4 + 2x^2y^2)^{1/2}, & (4.7)\\ &((x^2 + y^2)^2)^{1/2} &= x^2 + y^2, & (4.8)\\ &(x^2 + y^2) * scale &= NPS, & (4.9)\end{aligned}$$

where x and y are the real and imaginary portion of the transformed value. This derivation, although simplistic in nature, is computationally more efficient.

One must be careful as shown by Kijewski and Judy [51], that adequate size is available for the FT, otherwise frequency bleeding is possible. In general, if higher frequencies cannot be represented in the transform, the values are aliased onto other portions of the spectrum [51]. Therefore, our need to zero-pad the array before transforming to at least the next-power-of-two.

Using the reconstructed DICOM data from the Siemens Definition Flash+ scanner, we performed the estimated-NPS on the nine regions. Surprisingly, the DICOM image had low frequency noise in the heaviest attenuated regions. Specifically, the Siemens reconstruction was able to produce an unbelievable zero in these regions, as seen in Table 4.1; however, they performed poorly in alternate regions, as seen in Table 4.1. When compared with reconstruction techniques developed in this thesis, we consistently obtained substantially better estimated-NPS in ROIs 2, 3, 5, 6, and 7, seen in

Table 6.6. As we do not have access to the proprietary code we do not want to draw any conclusion, as the Siemens reconstruction algorithm and filters are legendary. Nonetheless, the results are worth mentioning. We have shown how to establish a quantitative metric that can be measured to give insight into the level of noise in a particular reconstruction. This noise metric will be used in the following chapters to illustrate how the estimated-NPS value can be influenced by a reconstruction method, region, and various algorithmic parameters.

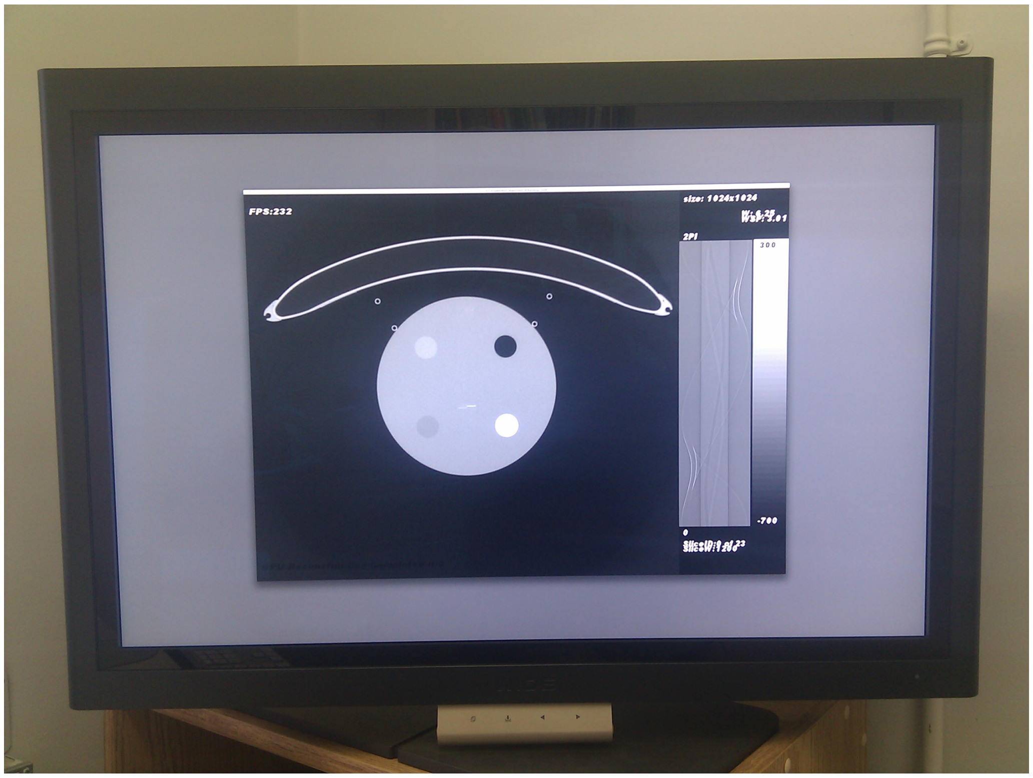


Figure 4.5: AREMI is seen displayed on the 14-bit PAC display in monochrome mode. The ability to output HDR to the monitor is likely beneficial to diagnostic imaging specialists.

4.1.2 AREMI Image Quality Enhancements

FBP is the most widely used reconstruction technique in CT, as it can be modelled relatively easily on a computer and is computationally efficient for image sizes of $m = n = 512$. This thesis redefines this algorithm in a parallel and multi-GPU environment. Through the use of the raw attenuation values, AREMI is able to perform any frequency-based filtering on actual raw data in addition to conventional image based filtering. Likewise, one can extend the numerical precision of the defined images to 14-bit, which provides a more detailed image on a specialized visualization hardware. As the framework uses raw data to complete the reconstruction, the algorithm can control the level of precision displayed in terms of computational precision used, and in terms of the images displayed for visual analysis. AREMI is not limited to the HU, 4096 values, and can adjust according to hardware available. The HDR available from Nvidia on select GPUs provides 10-bit output to compatible devices. However, the WIDE PAC display from DoubleBlack imaging further extends this range to 14-bit through a specialized and environment calibrated look-up-table (LUT). AREMI is shown on this monitor in monochrome HDR mode in Figure 4.5.

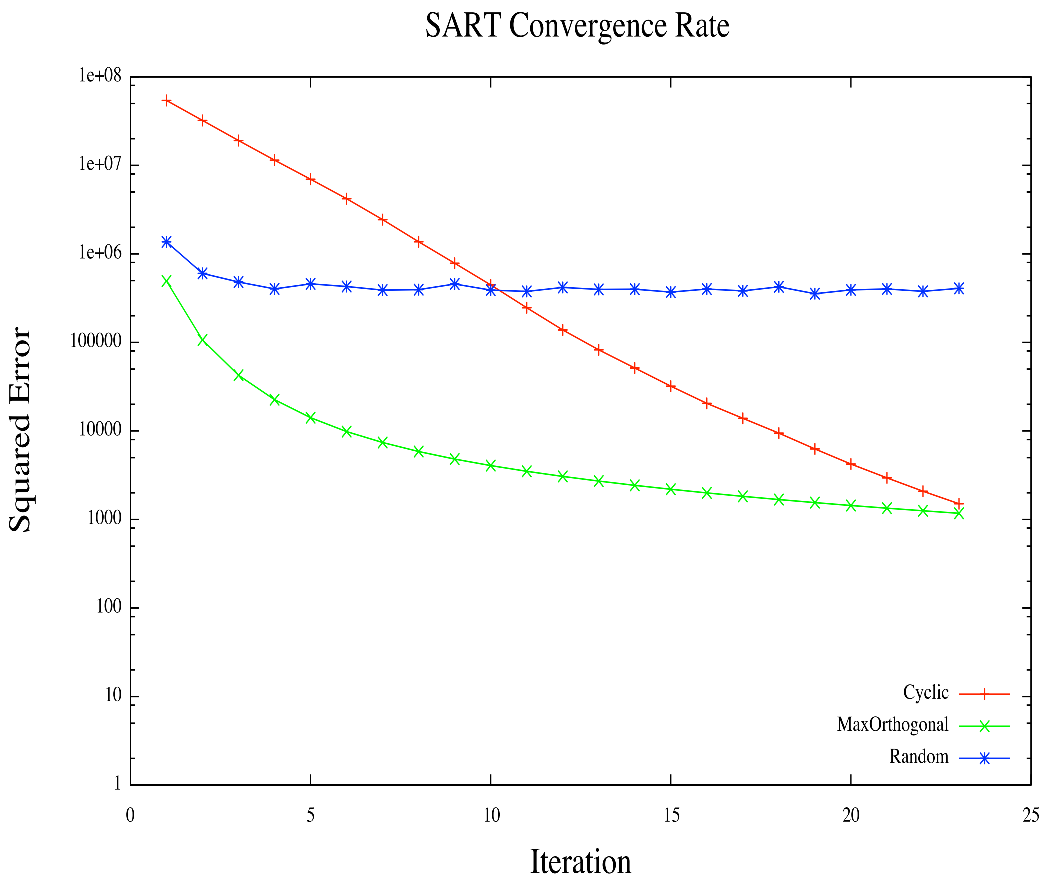


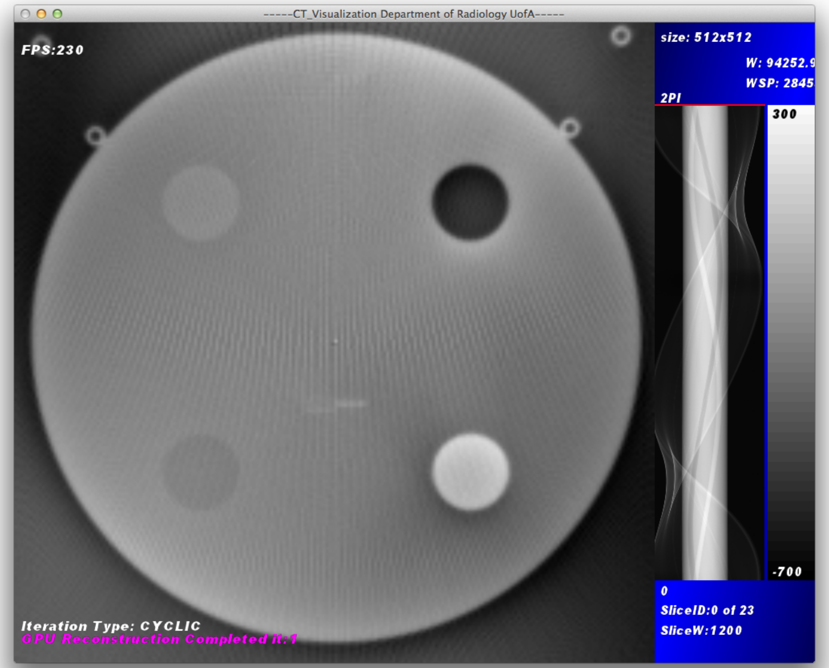
Figure 4.6: Three projection ordering techniques are evaluated for approximately 25 *ART*-iterations. The *cyclic* ordering produces a steady convergence though slow, *random* ordering produces unstable convergence, and the *maxOrthogonal* technique has rapid convergence after only eight *ART*-iterations.

4.1.3 AREMI Iterative Algorithms and Convergence

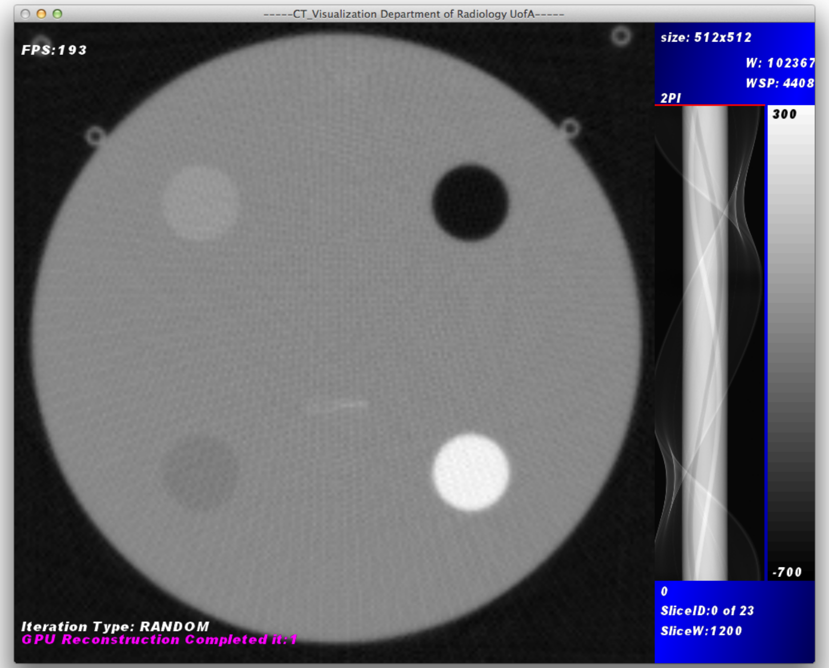
The convergence of ART type algorithms is an interesting topic area that is often neglected in many research publications. The faster the convergence rate to an acceptable solution, the more applicable the technique is in a clinical environment. In the following section we analyze the reconstruction of a 512×512 slice using different projection ordering techniques and two forms of area reconstruction. The *Cyclic* projection ordering samples the projections in the order they were acquired in the scan and recovers low-frequency components first, rather quickly as seen in Figure 4.7a. The high-frequency components are recovered later and considerably slower. *Random* projection ordering randomly samples the projections and can usually provide a reasonable reconstruction after one complete *ART*-iteration as seen in Figure 4.7b.

The first, and simplest method to achieve quicker convergence is the ordering of the projections in a manner that achieves a higher-level of orthogonality between updates to the solution, as originally shown by Guan *et al.* [29, 31]. This projection ordering techniques is referred to as the *maxOrthogonal* method in this paper, but original termed *MLS* [29] . The technique shows that, if the orthogonality between subsequent projections is large, the solution is achieved in less steps, as seen geometrically in Figure 2.16 and in the reconstruction in Figure 4.8. We present and analyze four forms of projection ordering: *cyclic*, *random*, *maxOrthogonal*, and a *hybrid* technique that combines the initial rapid update found in *maxOrthogonal*-method with the steady update found in later *ART*-iterations in the *cyclic*-method under AREMI, seen in Figure 4.6 - 4.11. These figures show that a reasonable solution to the problem is found when using *maxOrthogonal* ordering after only one *ART*-iteration of SART, where several *ART*-iterations are required for *cyclic* and *random* when reconstructing the phantom from raw attenuation data. The results are consistent with previous results from Guan *et al.* [29] when reviewing Figure 4.6.

In order to further validate the assertion that the best technique for convergence is *maxOrthogonal*, testing under different relaxation values was required. Figure 4.9 revealed that although the *maxOrthogonal* technique provided great initial convergence, the updates in later *ART*-iterations were relatively slow and *cyclic* technique actually performed better in terms of obtaining convergence. Using this information, a hypothesis was made that through a *hybrid* technique, which combined the rapid early convergence of the *maxOrthogonal* technique with the refinement ability of *cyclic* method, even faster convergence could be possible. Figure 4.10 shows the results of four hybrid techniques, mainly the techniques use the *maxOrthogonal* method until a set *ART*-iteration value, where the technique changes to the *cyclic* method. This resulted in faster convergence as seen in Figure 4.11 where the *hybrid30* performs the best and obtains the fastest convergence. These results are contrary to current literature where *maxOrthogonal*-method performs best [29]. The *hybrid30*-method achieves a substantial improvement in convergence time. The results were verified on various scans and consistent results were obtained. Therefore, the *hybrid* techniques are good, as it appears that the method is efficient in updating high frequency content in the beginning by using the *maxOrthogonal* method, but updates the low frequency content of the image in a more efficient manner by use of the *cyclic* method. Reviewing Figure 4.11, it is interesting to note that, for the *hybrid* techniques, it appears the more *ART*-iterations used under *maxOrthogonal* the steeper the descent after the transition to the *cyclic* method. One notices the *hybrid10* and *hybrid20* are relatively smooth, where the *hybrid30* and *hybrid40* have a more dramatic update to the solution. *hybrid40* - *hybrid60* perform worse than *hybrid30*, but still better than the pure *cyclic*, *random*, and *maxOrthogonal* projection ordering techniques. Faster convergence results in quicker runtime and greater applicability of linear algebra techniques in clinical-CT because convergence is reached



(a) AREMI SART reconstruction using *cyclic* projection ordering.



(b) AREMI SART reconstruction using *random* projection ordering.

Figure 4.7: *Cyclic* projection ordering after one *ART*-iteration of SART is depicted in (4.7a). The *random* ordering results in initial rapid convergence and a reasonable reconstruction seen in (4.7b).

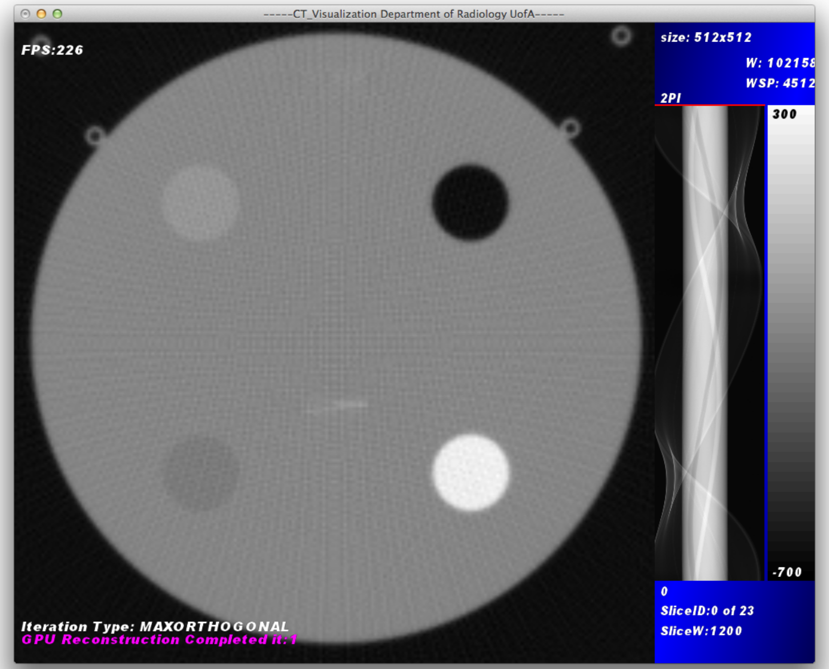

-----CT_Visualization Department of Radiology UofA-----
FPS:226
size: 512x512
W: 102158
WSP: 4512
2PI
300
-700
0
SliceID:0 of 23
SliceW:1200
Iteration Type: MAXORTHOGONAL
GPU Reconstruction Completed it:1

Figure 4.8: The *maxOrthogonal* projection ordering method results in a good reconstruction after only one *ART*-iteration.

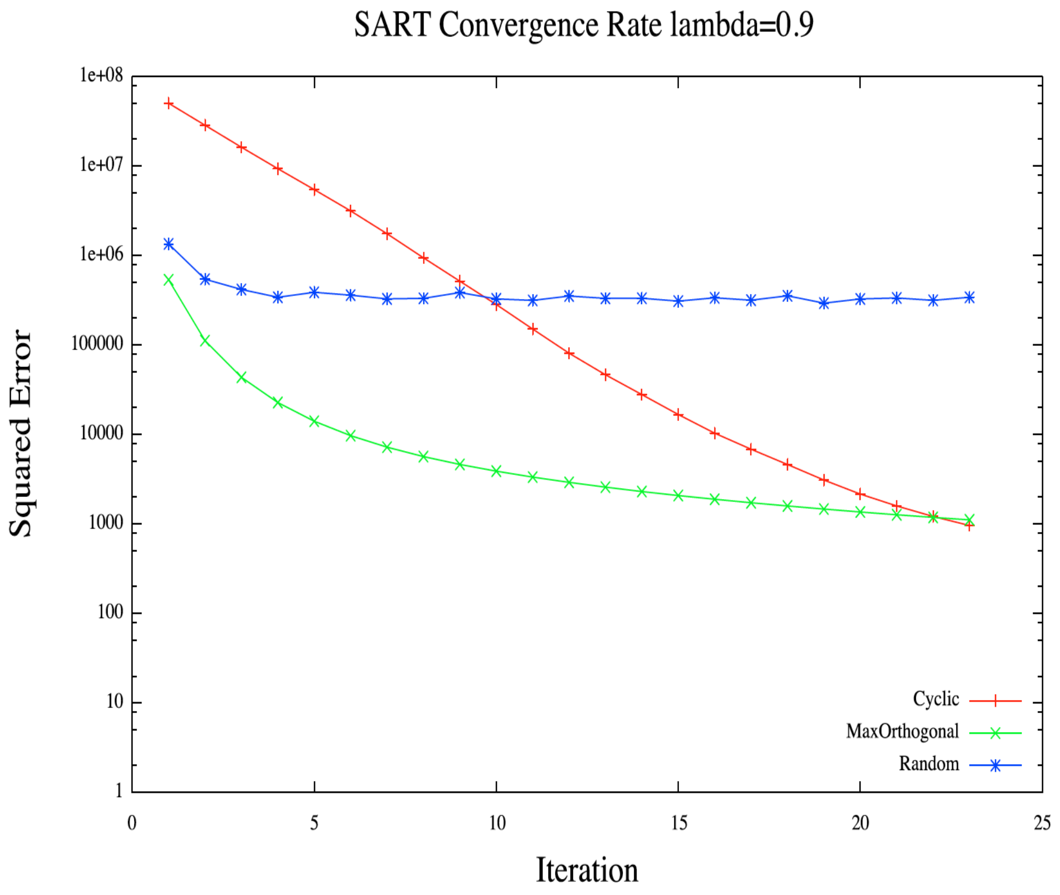


Figure 4.9: When using $\lambda = 0.9$ the *cyclic* technique obtains faster convergence than the *max-Orthogonal* method near 25 *ART*-iterations.

quicker.

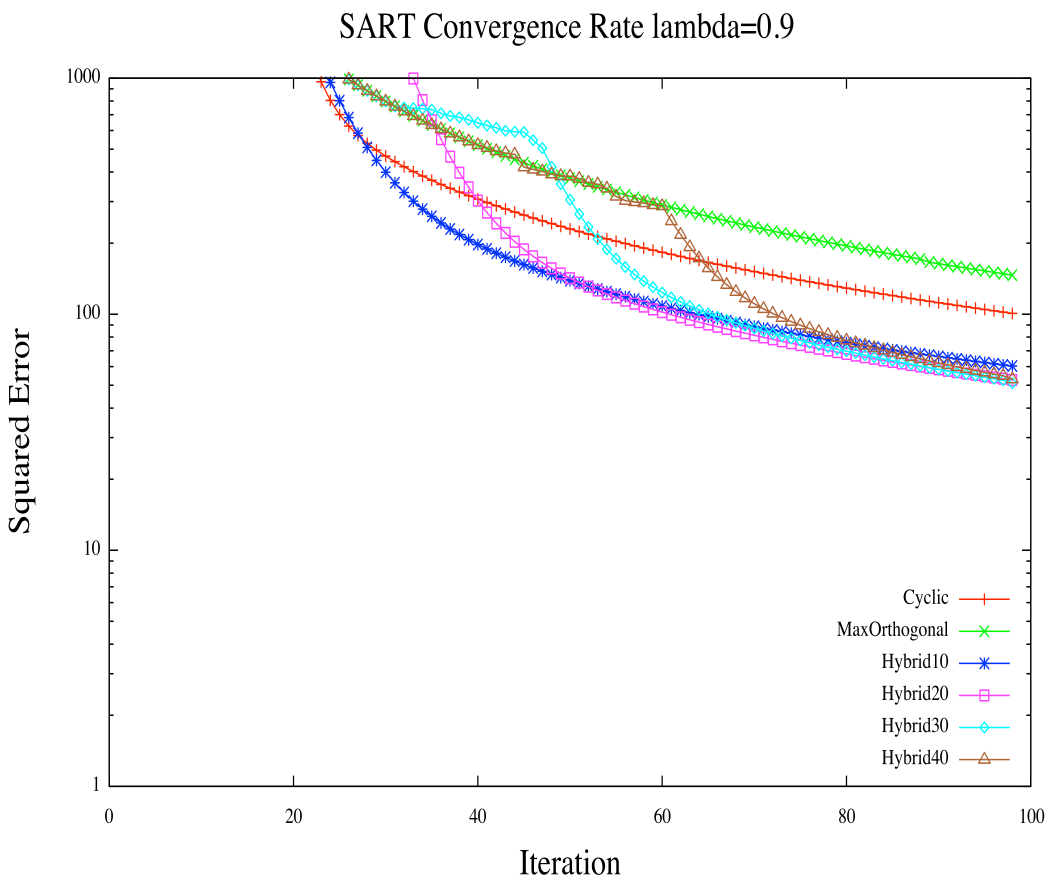


Figure 4.10: Several projection ordering techniques are evaluated for approximately 100 *ART*-iterations. The traditional *cyclic* ordering and *maxOrthogonal* methods are significantly slower at convergence than the *hybrid* methods. The *cyclic* method does not out perform the *maxOrthogonal* method in early *ART*-iterations only after *ART*-iteration 27.

4.1.4 Experimentation with Relaxation and ART

As mentioned previously, ART type algorithms attempt to converge towards a solution despite often using an underdetermined system. Therefore, the solution the algorithm may be converging on is one of many solutions, as there is at least one free variable available in the solution space. One can effectively reduce the amount of noise in the solution, or converge on a result that has less noise present, if relaxation is used [97, 38], as seen Figure 4.12. As an attempt to suppress noise in our data a $\lambda = 0.5$ was used, and experiments show an increased level of convergence versus the pure use of $\lambda = 0.9$ as seen in Figure 4.11. Through experimentation it is clear that there is no static best value for λ. However, it was consistently found on various reconstructions, that using a $\lambda \approx 0.8 - 0.95$ in the early *ART*-iterations and a lower value, $\lambda \approx 0.1 - 0.6$, resulted consistently in faster convergence with less noise. The justification for the results are: First, the higher value

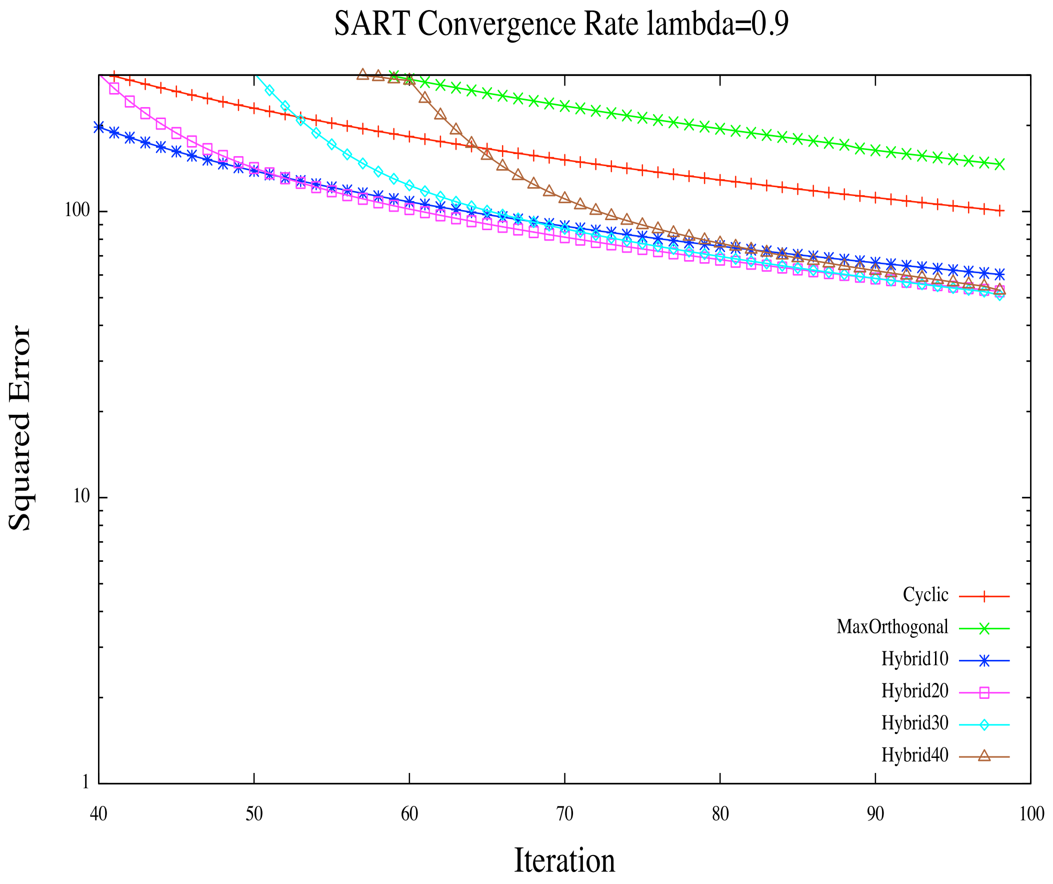


Figure 4.11: Six methods for projection ordering are evaluated and the final 60 *ART*-iterations are shown. The *hybrid* methods all outperform the traditional techniques. Specifically, the *hybrid30* method offers the fastest convergence. When comparing *hybrid30* to the *maxOrthogonal* method, the *maxOrthogonal* method would require 500 more *ART*-iterations to reach the value seen by *hybrid30* after only 100 *ART*-iterations.

of λ in the beginning requires the solution to be more constrained and dramatic updates result. Second, using a lower value of λ later during low frequency updates suppresses noise in the solution providing a greater degree of convergence when comparing L2-error.

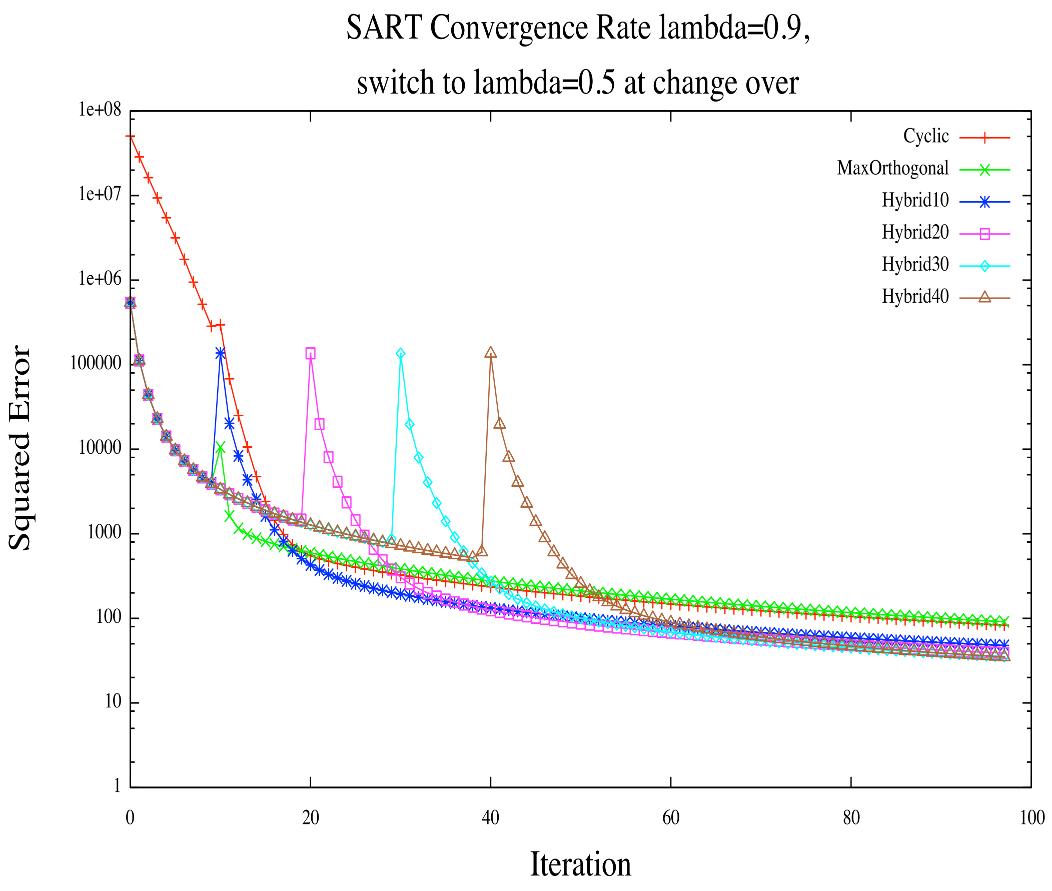


Figure 4.12: Experiment of switching over to $\lambda = 0.5$ at change over point from $\lambda = 0.9$ to suppress noise resulting in consistent results.

4.2 Analysis of Choice of Reconstruction Region

Convergence and Runtime

There are two methods for reconstruction in terms of ROI: square and circular. Specifically, one can reconstruct a complete square region or a circular region as seen in Figure 2.15. The purpose of choosing a circular reconstruction region is twofold. First, as we previously showed the reconstruction region is smaller when considering a circular reconstruction region versus a square region. This smaller size should result in computation speedup as not as many pixels are visited. Second, the model more accurately depicts the real scanning modality and assigns the zero attenuation weight value to those pixels not contained in the circular reconstruction region. One would

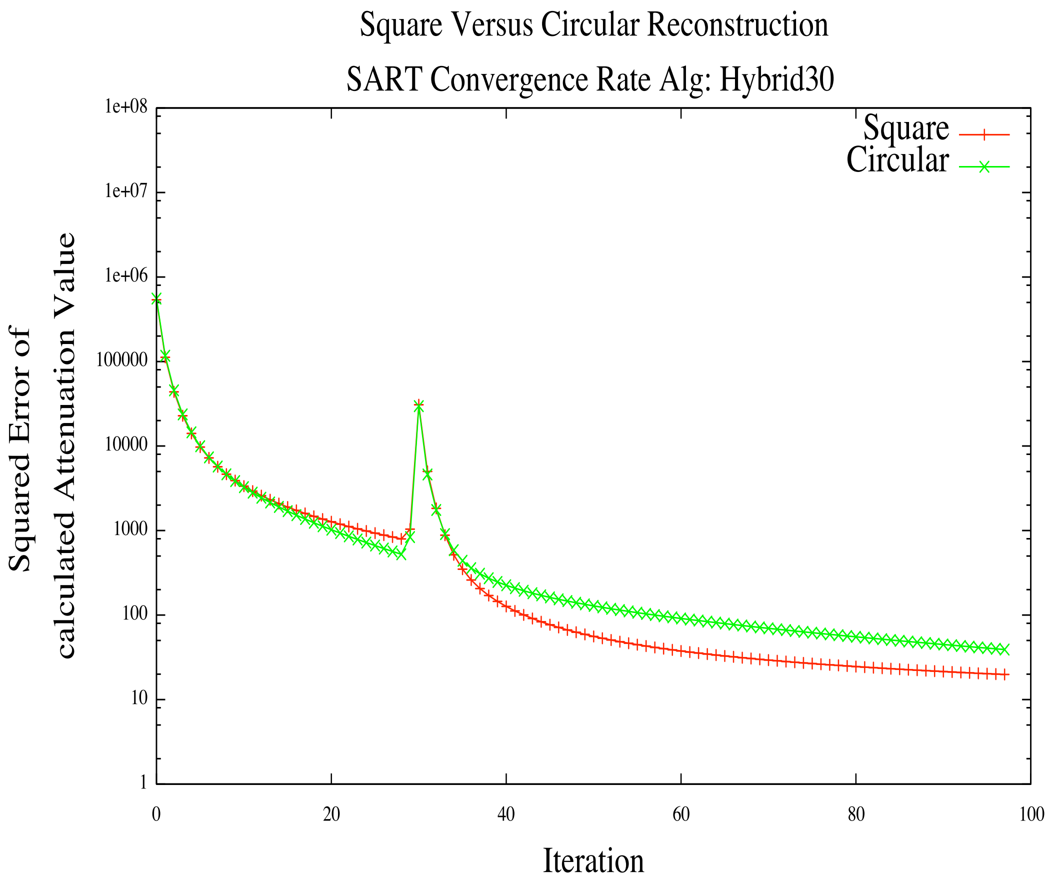


Figure 4.13: The square reconstruction technique results in significantly faster convergence when compared to circular reconstruction. The spike seen is caused by the change over point in the *hybrid* method when the technique switches from *maxOrthogonal* to the *cyclic*-technique.

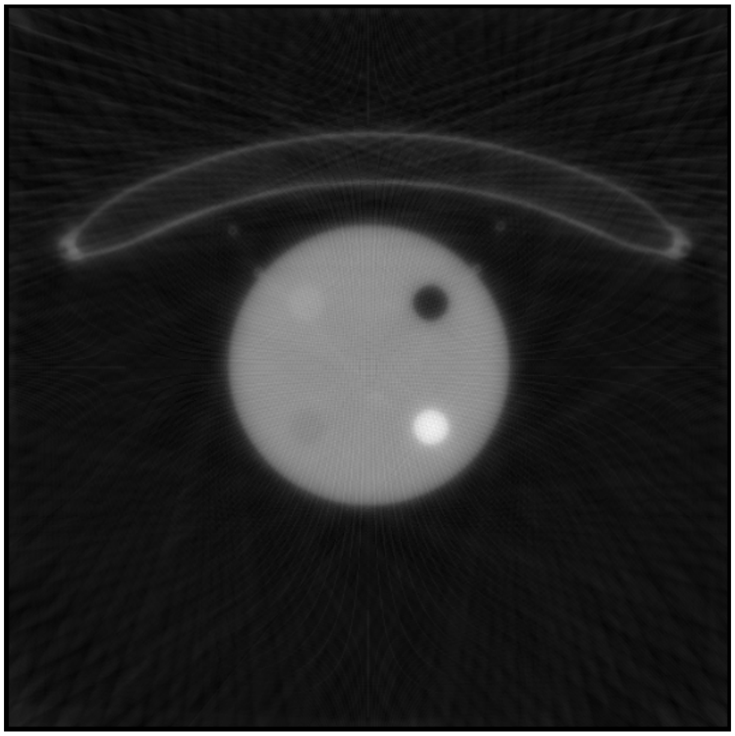
Table 4.2: Square vs. Circular Recon – Estimated-NPS for ROIs

Square and Circular Reconstruction Techniques, results NPS units$^2 10^{-5}$		
Recon Region	Square	Circular
ROI 1	1.89800	2.09291
ROI 2	8.80082	8.66552
ROI 3	2.15063	2.08043
ROI 4	88.8886	89.7349
ROI 5	1.25005	1.21675
ROI 6	1.22961	1.09228
ROI 7	1.09486	1.06625
ROI 8	171.932	38.7796
ROI 9	180.599	39.0146
Total Approx.	457.84	183.73

believe this to result in quicker convergence and possibly less error. However, contradictory results were obtained. Mainly, convergence was quicker with the square reconstruction region as seen in Figure 4.13. Second, runtime was actually quicker for the square reconstruction of 318s versus the circular reconstruction region of 344s for 100 ART-iterations for a 512×512 reconstruction. We believe the longer runtime is because of a more complex grid interpolation technique that is required for the circular reconstruction region to be mapped to our version of Siddon's algorithm.

Noise Analysis

The analysis of noise was drastically different when using the circular reconstruction region. The circular reconstruction region showed slower convergence but less estimated-NPS in the final reconstruction when compared to the square ROI as seen in Figure 4.13 and Figure 4.2. Although, in some ROIs the square reconstruction technique result in less noise, ultimately the summation of estimated-NPS was much lower for the circular reconstruction region. We believe the reduction of noise is due to the system more accurately modelling the actual scan and what we already know about the SNR for highly-attenuated regions [104]. For instance, when considering the square region there is noise present in ROI 8-9 seen Table 4.2. However, ROI 8-9, illustrated in Figure 4.1.1, should have low measured noise because this region has a very high SNR as it is air [104]. Nonetheless, ROI 8-9 show the highest estimated-NPS level. This noise is likely caused by noise in the highly attenuated interior region being distributed in the air region. The circular reconstruction does not distribute noise in the air region to the same degree and results in approximately 78% reduction in noise in ROI 8-9 in our implementation.


(a) AREMI SART Reconstruction using only 0.1% of 1152 projections.

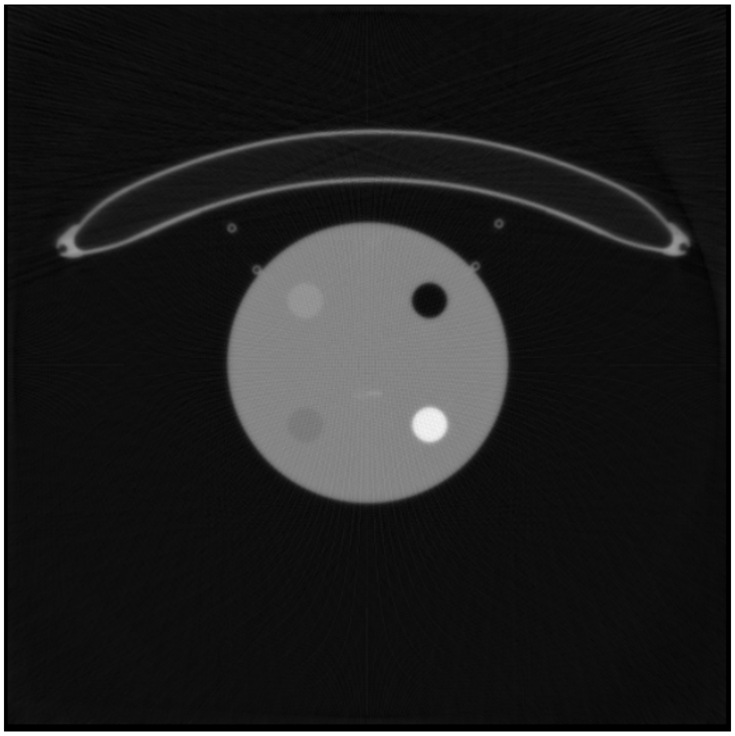
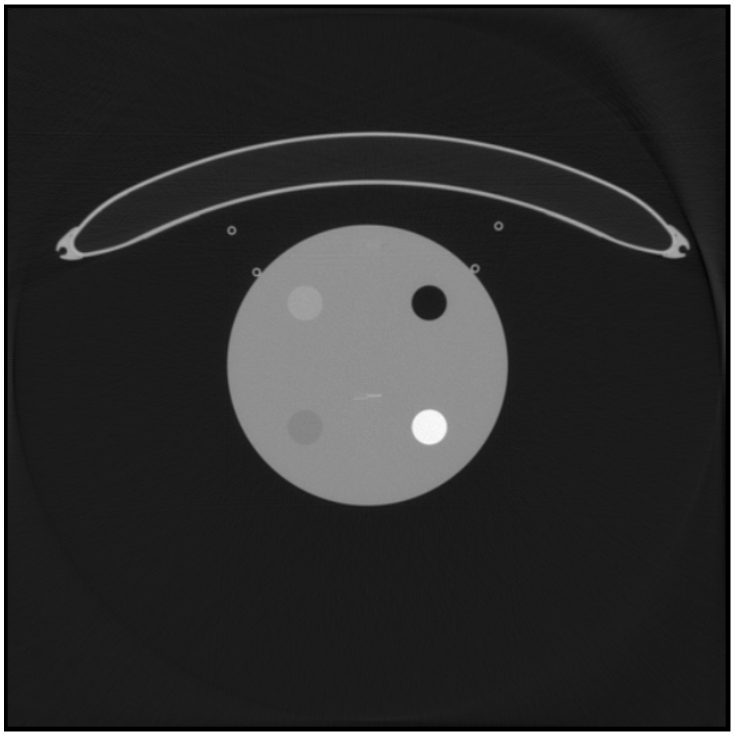
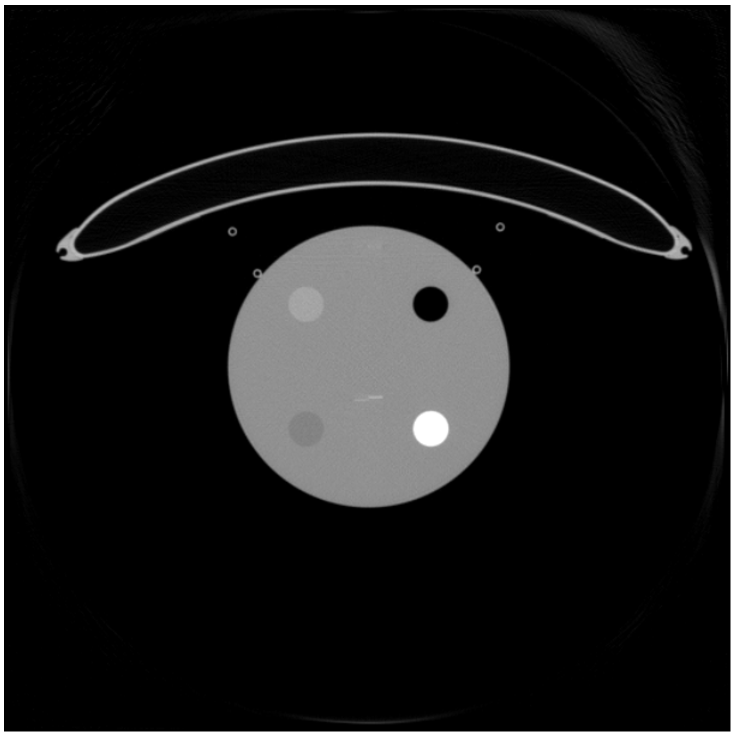

(b) AREMI SART Reconstruction using only 0.5% of 1152 projections.

Figure 4.14: AREMI has the ability to test subsets of the projections available. Using only 0.1% of the available projections results in the high-level of artifacts seen in (4.14a). Using 0.5% of the projections results in a reconstruction with less artifacts as seen (4.14b).



(a) AREMI SART Reconstruction using five complete *ART*-iterations.



(b) AREMI SART Reconstruction using 100 complete *ART*-iterations.

Figure 4.15: AREMI SART based full *ART*-iteration reconstructions are depicted. Using five complete *ART*-iterations of SART, i.e. $5 \times 1152 \times 736$, results in (4.15a). Using 100 *ART*-iterations results in an almost artifact free reconstruction, as seen in (4.15b), but with a slight amplification of noise in the right perimeter midpoint.

4.3 Convergence and Dynamic *ART*-iteration Count

The number of *ART*-iterations required in ART-type algorithms is dependent on the scan and no set *ART*-iteration count is acceptable for all reconstructions. Therefore, AREMI has the ability to perform more *ART*-iterations if required during visualization. The reconstructed images using 0.1%, 0.5%, 5, and 100 *ART*-iterations using *maxOrthogonal* projection ordering is seen in Figures 4.14 - 4.15. Using too few *ART*-iterations results in noise and artifacts in the reconstructions. However, using too many *ART*-iterations without relaxation can result in amplification of noise in the image [97].

The absolute convergence of SART was proved by Jiang *et al.* [43]. However, the change in solution space from $5 \rightarrow 100$ *art*-iterations is often very minimal in our experimental reconstructions, as seen in Figure 4.15. Likewise, artifacts, and noise, often become more pronounced when using very high *art*-iterations (> 500). Last, the squared error seen in Figure 4.12 after 100 iterations is only a couple points higher than after 500 iterations; making the increased iterations and computational time difficult to justify.

4.4 Summary

The current quantitative metric for establishing convergence in raw-data is based on square error between the current iteration and previous iteration. The definition of convergence could be enhanced by using a composite metric that would also use the estimated-NPS. However, this composite metric could only be defined when using a scan with known homogenous areas, and for this reason, is only mentioned for completeness. Nonetheless, by basing convergence on the level of noise in the reconstruction ROIs and squared error one could likely better define convergence. One could postulate that various weighting functions for the composite metric would also enhance the reconstruction.

We have established a quantitative metric for noise analysis through the use of an estimated-NPS for homogenous regions of the scan. We have demonstrated a means to evaluate convergence for iterative algorithms. We have shown an alternative mode for projection ordering that can result in convergence to a solution in a much more timely manner. We illustrated the effects of reconstruction region on convergence and noise.

In the next chapter we use the methods developed in the previous chapters to enable calculation of the radial and longitudinal deflection from FFS, given the Siemens Definition Flash scanner geometry. Second, we briefly review the representation of the system matrix $\mathbf{A}$. Last, we review the enhancements to Siddon's algorithm designed for a multi-GPU environment.

Chapter 5

Consideration for Reconstruction of Raw Data

5.1 CT-Reconstruction from Raw Data

Computed Tomography (CT) has many flavours of reconstruction algorithms and, surprisingly, in clinical environments these are typically based solely on manufacturer-available routines. The modern day CT-machine has the ability to export raw-attenuation values. These raw-attenuation values are based on the ray-sum of an area-based detector in a physical, non-simulated, setting as well as vendor specifications. Commercial reconstruction algorithms are typically based on FBP and variations such as Feldkamp, Davis, and Kress (FDK) [20]. The FBP technique has been shown to require $S_{FBP} = 1.57n$ projections, where n is the length of the diagonal grid of the reconstruction region [63, 64, 65]. There are various justifications of FBP use, such as, the methods have been in use for a number of years and many seasoned radiologists have become familiar with the reconstruction generated from by this technique. Likewise, the implementation details are relatively parallelizable and, for that reason, specialized hardware such as Field Programmable Gate Arrays (FPGA) are designed by vendors for near-instantaneous reconstructions in clinical settings. These specialized pieces of hardware are expensive and are based on specific algorithms, but lack the reconfigurable ability of CPUs or GPUs. However, they have achieved substantial speedups in runtime and were shown to be 100x faster than serial processing on the CPU [12]. The wide use of FPGAs is not generally seen in research because of their high-cost and have been, generally, only beneficial where the algorithm required detailed, low-level, hardware control operations [93]. There are alternative methods to FBP and FPGA-type implementations, but these are generally not seen in clinical environments because of a lack of motivation by manufacturers to move to alternative methods [71]. Likewise, there is an extreme deficiency in the understanding possessed by CT-design engineers in the recent advancements in computing and applied mathematics [71]. This means CT-design engineers are not truly aware of the possibilities of current computer hardware and mathematical concepts [71]. There are alternative methods, namely algebraic techniques, to CT-reconstruction that require less radiation exposure to patients, faster acquisition time, and a better reconstruction [71, 26, 63, 64, 65]. These methods are typically called iterative methods for CT-reconstruction. One class of algorithms, in this iterative reconstruction domain, is ART originally formulated by R. Gordon in 1970 [27]. The ART method relies on an iterative method to update the solution image to a large system of linear equations. The technique is computationally demanding but has been show to require approximately half the number of projections compared to typical FBP methods [63, 64, 65]. Specifically, for a single source scanner requiring $S_{ART} = 0.67n$ projections and for dual source scanners [88, 64], like the Siemens Definition Flash scanner, requiring $S_{ART} = 0.78n$ projections [63, 64, 65]. The reduction in the number of projections results in reduced acquisition times and, most importantly, reduced radiation exposure to patients' CT radiation; exposure having been correlated with increased risk in the development of cancer [2, 7]. Therefore, research and development into these alternative methods is important to patient health and the future of CT in diagnostic imaging.

There are a number of difficulties for new researchers to experiment with and overcome by developing innovative reconstruction algorithms. Specifically, understanding proprietary technologies, such as FFS, efficient system matrix layout, interpolation schemes, and hardware choices are but a few of the laborious areas that require an enormous amount of background research in this area. The aim of this section is to give a clear and insightful review of three essential aspects to CT-reconstruction that are complicated. Mainly we review FFS mode and how to effectively handle this technology in an ART domain and in conventional FBP. Second, we describe various interpolation techniques and an efficient system matrix representation. Last, we give a formal 3D implementation of Siddon's Algorithm on the GPU. Through careful presentation and analysis, this section provides a concise description of topics which are often overlooked and not well described in the literature, but essential in the development of efficient and precise reconstructions.

5.1.1 Background and Methodology

The following sections will provide insight into reconstruction of raw data from clinical-CT. Important topics such as FFS, system matrix representation, and quality metrics will be discussed.

5.1.2 Flying Focal Spot (FFS)

FFS modes are present on most commercial-CT scanners, as they provide increased spatial resolution in both radial and longitudinal directions, x and z respectively. The technology is revolutionary as it provides a means to extract an increased level-of-detail without shrinking hardware sensors, which are already small and closely packed. The technique uses a continuous electromagnetic deflection of the electron beam that results in the focal spot wobbling in radial and/or longitudinal directions as seen in Figure 3.7. The exact angle of deflection in the radial direction and length in the longitudinal direction are proprietary information. However, a close approximation has been found to work well by Kachelrie *et al.* in 2006 [45]. The FFS modes have the ability to substantially suppress artifacts in scans as seen in Kachelrie's *et al.* (see Figure 7 from [45]). The angle for the radial and longitudinal deflection can be approximated by [45]:

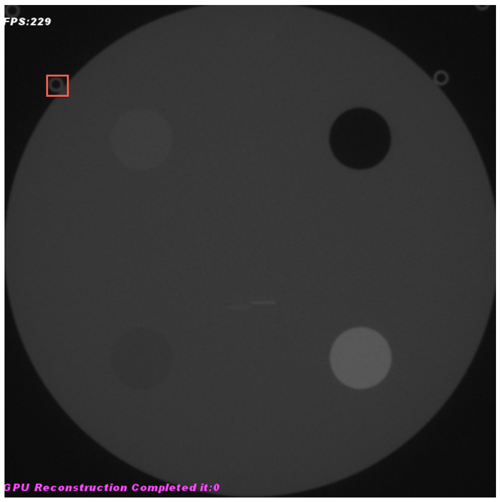
$$\triangle\bar{\beta} = (1/4) * (2\pi/\text{projectionCount}) * \frac{R_{FD}}{R_D}, \tag{5.1}$$

$$\triangle\bar{z} = (1/4) * (\text{sliceThickness}) * \frac{R_{FD}}{R_D}, \tag{5.2}$$

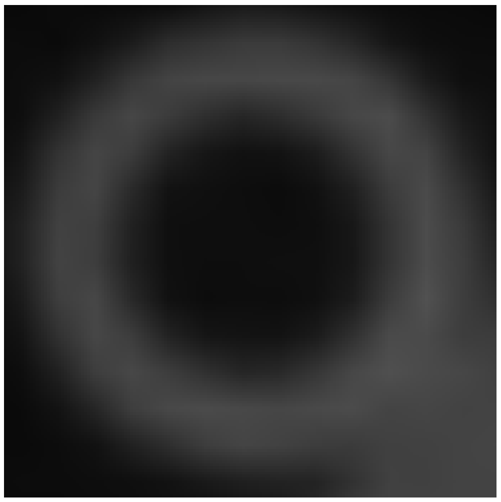
where $projectionCount$ is the number of projections, R_{FD} and R_D are the distance from source $\rightarrow$ detector and distance from isocenter→detector respectively. The *sliceThickness* corresponds to the collimation and nominal slice thickness, usually defined in the scan parameter file exported in raw format from the machine. As an example, given the test machine used, as defined in Appendix B, we have $R_{FD} = 1085.6\ mm$, $R_D = 490.6\ mm$, projection count of 4608, and nominal slice thickness is set at $0.6\ mm$. Under these parameter values, we have a radial deflection angle $\triangle\bar{\beta} \approx 0.04322°$ and a longitudinal length of $\triangle\bar{z} \approx 0.33192\ mm$. These values provide a good starting point for reconstruction under various FFS modes. However, to achieve more optimal settings for a specific machine, we present a simplistic user assisted method for calibration and an automated version. Using a phantom of known composition, under the assumption of a valid reconstruction environment, the user can simply adjust the radial angle and the longitudinal length through a keyboard shortcut until the desired results are achieved. The second method is an automated technique that requires a phantom of known composition and a specific line pair calibration. When the line pair is of known location and composition, the system can reconstruct and adjust until the error between the known and reconstructed image is below a tolerance value. Using Kachelrie's *et al.* approximation as an initial guess, either of the two suggested calibration techniques leads to good results, as seen in Figure 5.1. However, closer analysis of a feature of each reconstruction reveals the user-assisted calibration is much more accurate. Likewise, the spatial resolution as depicted in Figure 5.2 shows the calibrated reconstruction line intersection more closely resembles the actual line. Quantitatively the graph shows the area is much larger when comparing the non-calibrated method to the line-pair versus the calibrated method. This greater area would equate to a larger squared error metric. The calibration step is not needed for every reconstruction. After the initial calibration steps, these values can be saved and updated as needed. Generally, after the initial calibration, no further calibration is needed for a specific machine and the stored values can be used.

5.1.3 Representation of the Linear System

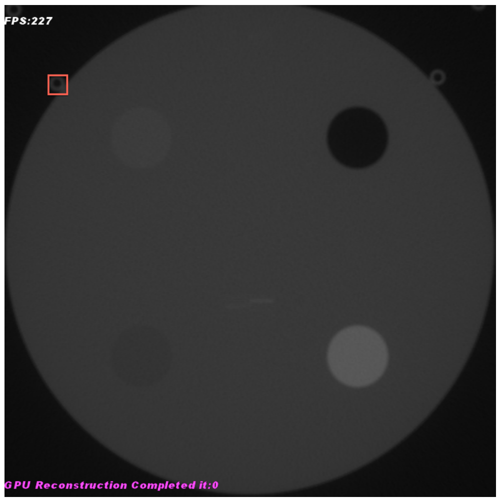
One alternative to FBP reconstruction algorithms is based on a linear algebraic representation of the Radon transform. Given the Radon transform, a matrix representation can easily be defined as in Section 2.12.5. The system matrix **A** represents the Radon transform of the image $f(x, y)$ for a sinogram $g(l, \theta)$. The matrix represents the forward and back-projection of rays through the image. Using Equation 2.23 means the 2D image and sinogram need to be represented in a vector form as



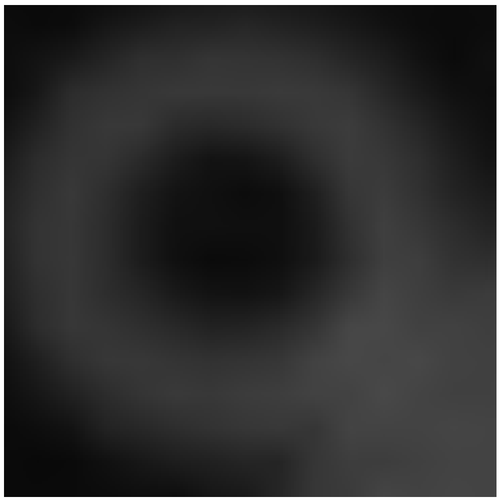
(a) Reconstruction with FFS Calibration



(c) Feature with FFS Calibration



(b) Reconstruction without FFS Calibration



(d) Feature without FFS Calibration

Figure 5.1: FFS calibrated reconstruction is shown in (5.1a), comparable to non-calibrated reconstruction seen in (5.1b). The feature of calibrated image seen in (5.1c) is sharper than the feature from the non-calibrated image in (5.1d). The features are depicted in the original images by the red highlighted box.

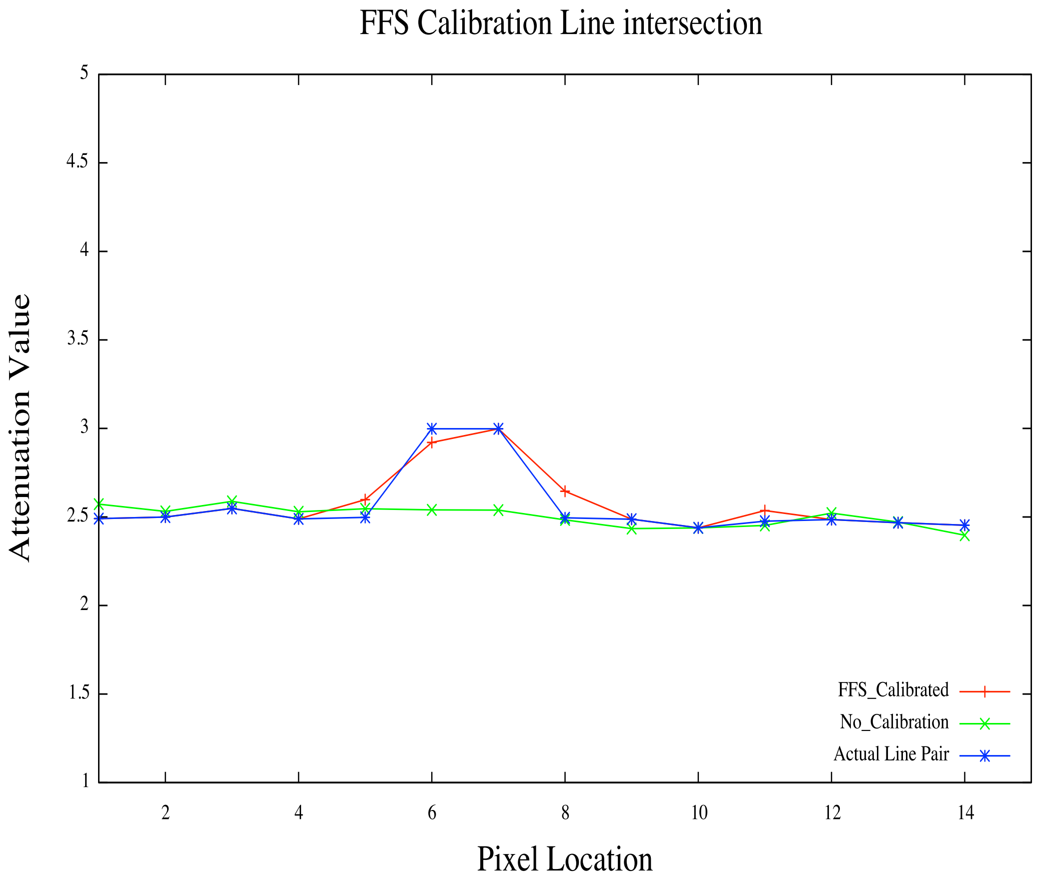


Figure 5.2: Graph depicting the intersection of line from reconstruction seen in Figure 5.1a and 5.1b with user-assisted calibration value versus approximation.

seen in Section 2.13, and summarized for clarity here for efficient computation on the GPU:

$$\mathbf{b}_{kl} = p(\rho,\theta)_{k\times l}, \tag{5.3}$$

$$\mathbf{b} = \{a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_1, c_2, c_3, c_4, \ \dots\ , o_1, o_2, o_3, o_4\}, \tag{5.4}$$

$$\mathbf{b}_i = b_{kL+l}, \tag{5.5}$$

$$\mathbf{f}_{mn} = f(x,y)_{m\times n}, \tag{5.6}$$

$$\mathbf{f} = \{a_1, a_2, \ \dots\ , a_8, a_9, b_1, b_2, \ \dots\ , b_8, b_9, \ \dots\ , i_1, i_2, \ \dots\ i_8, i_9\}, \tag{5.7}$$

$$\mathbf{f}_j = x_{mN+n}. \tag{5.8}$$

The vector form is relatively simple and one can easily access specific projections or rays by use of two operations; namely the modulus and division operators. For instance, assume we have the data stored in the efficient contiguous vector form and are performing updates to the solution vector, as in typical ART algorithms. We would like to carry out computation and; therefore, need to recover the source angle β and detector offset γ. Using the modulus operation in the following form $\gamma = currentrow\%k$, the detector offset is recovered. Using the division operation in the following form, $\beta = \lfloor \frac{currentrow}{k} \rfloor$, the source angle can be recovered. The requirement for a specific projection, row, or gamma offset is common in this representation and keeping the data in an efficient contiguous allocation in vector form keeps frequent accesses efficient and convenient with the mentioned operators.

The construction of the system matrix $\mathbf{A}$ is heavily researched and various interpolation schemes can be seen in [97, 102, 19]. The system is often underdetermined and the system matrix $\mathbf{A}$ is large. For instance, take $f(x,y)$ with dimension $m = n = 512$ matrix and $g(l,\theta)$ with dimension $l = 736$ and $k = 1152$ minimally. Matrix $\mathbf{A}$ is of dimension $m \times n \times k \times l =$ 222,264,557,568 elements. Representing this on a computer with single or double precision is expensive, in terms of memory. Luckily, if one chooses pre-computation of $\mathbf{A}$, then a sparse representation can be used that requires approximately 0.1% of the storage, because only a minimal amount of entries are non-zero. This sparse representation results in significant memory saving and speedup as seen in Toft's dissertation [97]. However, this sparse representation relies on a more complicated numerical procedure for various mathematical operations, because the sparse storage format needs to be considered when applying mathematical operations.

An alternative method to this dense representation can be seen in the use of Siddon's algorithm [94]. Specifically, we refer to the enhanced Siddon's algorithm described for FFS modes in Section 3.2.2 with our noted improvements.

A CUDA-based GPU implementation, with performance tuning, of the Jacob's enhancements to Siddon's algorithm was presented in Chapter 3. Nonetheless, it is interesting to investigate Toft's interpolation schemes from 1996 on current parallel hardware as will be seen in Section 6.2. Specifically, Toft asserted that the computational complexity of the calculation of ray and volume intersects is too complicated and requires an enormous amount of time on conventional computers. However, with the ability to use parallel processing, one can quickly compute the ray intersects with the given volume in a fast manner. The algorithm is perfectly scalable for GPU, multi-GPU, and mixed-modes environments of CPUs and GPUs. This is not the case for Siddon-type ray tracing algorithms where an upper-bound is present on the number of usable GPUs, as seen in Section 6.1 and 6.2. There is an upper-bound that is dictated by the available workload, as distributing a small workload over many CPUs and GPUs is not effective in reducing the runtime. Given an axial reconstruction the maximal distribution of approximately 1200 rays for dual source CT-scanners, over current Nvidia Fermie 6000 series GPUs, scales well to approximately three GPUs because each GPU has 448-stream processors. Figure 5.3 shows that, given a finite number of rays, distribution of more than one ray per stream processor results in idle stream processors on a GPU and decreases throughput. One can utilize smaller sub-volumes for ray intersects, as seen in Figure 5.4, in order to utilize more stream processors efficiently, but the merging of the results from each volume can be costly, as it requires transfer of information from GPUs through the PCI-bus then to main memory. This transfer of information tends to be the bottleneck of the technique and significantly hurts the throughput. Therefore, understanding of non-raytracing techniques is important and can achieve substantial speedups in the future as the problem can be distributed over many GPUs efficiently. Both techniques of ray-tracing and the complete technique will be seen in Section 6.1 and Section 6.2.

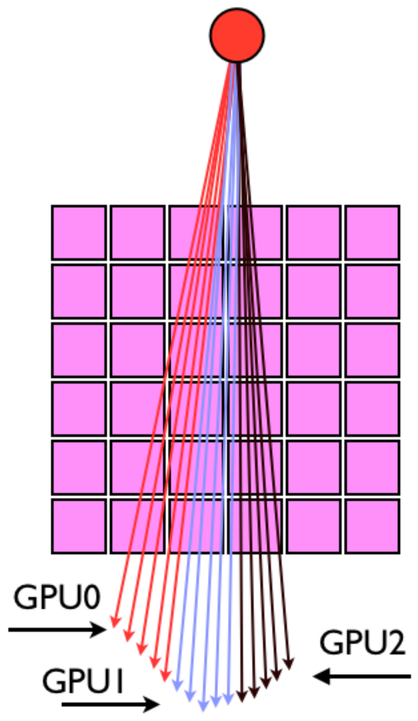


Figure 5.3: Siddon's algorithm using a ray-partitioning technique to distribute workload to multiple GPUs

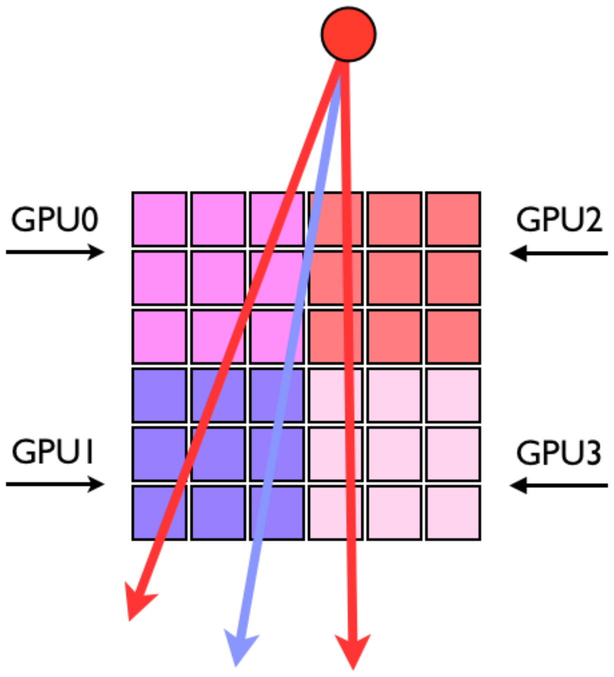


Figure 5.4: Siddon's algorithm using a volume-partitioning technique to distribute workload to multiple GPUs

2D and 3D Optimized Implementation of Siddon's Algorithm

Siddon's algorithm [94] is an alternative method to the laborious complete techniques, described previously in Chapter 2.2 and rather completely in Toft's dissertation [97]. The technique relies on the ray-based walkthrough the ROI and the intersection with x, y, z planes with the given ray. Although the technique is more efficient than the complete technique, Jacob *et al.* recognized the storage and merging of intersecting planes is inefficient [41]. Given a GPU implementation of Siddon's algorithm the merging of planes requires dynamic storage on the GPU, which is incredibly inefficient as the method requires storage of a dynamic list of intersecting planes. This dynamic list needs to be read and reading memory is a costly operation on the GPU. Therefore, this contribution serves to give 3D implementation details and an analysis of Jabob's *et al.* enhancements to Siddon's algorithm for a GPU and multi-GPU environment. In an attempt to keep a concise document, the implementation details of the enhanced Siddon's algorithm on the GPU will not be repeated, but the pseudocode can be seen in Section 3.2.2. We now present some of the performance analysis of subtle improvements on implementation that are reflected in runtime.

CUDA Algorithm Branching and Early Loads

The GPU represents an efficient parallel processing engine, but like any efficient engine, requires efficient layout of data and the most direct path, or simplest algorithm, is not what actually performs the best. First, branching should be limited, as a branch can result in idle CUDA-cores if there are not enough threads on a given branch. There are initializations of variables that could be replaced by typical *if/else* statements, such as line 15 from Algorithm 3.1 and lines 2-3 from Algorithm 3.2. One would suspect the compiler would structure the code in an efficient manner, but results show a performance improvement when one avoids the *else* statement in favor of initialization. Understanding how a *warp* operates in CUDA leads to a justification as, by avoiding the branch the SM keeps all SPs on the same path and fuller-*warps*. Likewise, if globally GPU addressed data is required, as in line 28 from Listing 3.3, perform the load early. This early load prevents idle SPs waiting for required data. Instead, if the load is performed early, computation continues as the operations do not require this loaded data. Finally, when the data is required, it has already been loaded, effectively hiding the load. On a HI-resolution reconstruction of 2048×2048, and 25 *ART*-iterations under SART, a consistent 2% reduction in runtime was realized following these simple suggestions. This reduction in time is significant as runtime can last from several seconds to hours. This equates to seconds, minutes, and possibly hours of runtime savings.

Further Enhancements to Siddon's Algorithm

One may recognize that the ROI required is often a circular or a spherical reconstruction. However, Siddon's algorithm only handles the case of square ROI. Considering Figure 2.17 and 2.15, one quickly realizes reconstructing a complete square is wasteful. Specifically, considering the area of a square versus a circle, or the volume of a cube or sphere, they differ by approximately 20% and 30% respectively. The ROI is generally a circular region and, for this reason, we hypothesized, not evaluating those locations that are not required for reconstruction should speedup computation. This non-important area can be seen as the white area in Figure 2.15. To accomplish this one can use two techniques. A simple check in the inner while-loop from Listing 3.3 for exit of the ray from the ROI by evaluation of the parametric equations for variables given the current α_c value. The alternate method would be to evaluate of the roots of the quadratic formula for ray and ROI intersection points. These points can then be used to find the required indices and a speedup would be likely, because there is a decrease in the number of pixels needed to be evaluated. The results of this hypothesis were shown in Section 4.2, and the hypothesis was shown to be invalid because of the increased cost of calculating the circular indices. However, it was shown that convergence, which is measured by squared-error, occurred quicker with the square reconstruction region. Nonetheless, the circular ROI resulted in a significantly less noise, approximately $1/3$, when compared to the square ROI technique.

Image Quality Metrics

As with noise estimation of raw data, there is no *gold-standard* for image quality. However, using the ACR-accredited phantom, we have the ability to measure spatial resolution in the terms of line-pairs and areas of known geometry, as seen in Figure 5.5. Although line-pairs provide a good-quality metric, they are based on user guided conceptual and perceptual understanding. Another possibility is a line-pair error metric where a line intersects the line-pair at a right-angle. The points are sampled, to the resolution of the reconstruction and plotted, seen in Figure 5.2. The closer the sampled line is to the known value, the more accurate the spatial resolution of the reconstruction is. Various error metrics can be used when comparing to the known value. Finally, when geometric properties of the phantom are of known specifications, one can easily test intersection of those regions with rays in order to calculate the exact geometric size. For instance, when a ray is defined and sampled it will spike as it intersects a circle or square, or any region of non-uniformity. The distance between the spikes represents the geometric size of the object and can easily be compared to a known value and a quality metric can be assigned, based on various error metrics. We have outlined two ways to access spatial resolution and geometric quality. However, although the technique is sound and used in many areas of research, it is flawed as a quality-metric because a good edge-preserving filter can be used that will boost this specific quality score. Nonetheless, these metrics are often used in the literature, but one must consider if such filters have been used when evaluating the quality of a reconstruction.

Although geometric evaluation is superficially interesting, it does not represent a fair quality metric as an algorithm can specifically target edge-enhancements. Nonetheless, the line pairs described previously can aid in the spatial resolution qualitative analysis and should be used.

Algorithm Convergence Rate

The ability to properly define convergence is a difficult task, as the system that one attempts to solve, in ART-type iterative methods, is often underdetermined. Therefore, as there is a great number of unknowns, versus restricting equations, it results in an infinite number of solutions. As mentioned previously, there are techniques such as compressive sensing where the number of non-zero elements is sparse. We did note in Section 2.14 that our system is sparse; however, compressive sensing is a research topic worthy of a dissertation on its own, and is out of the scope of this document, but we refer the reader to an informative webpage from Rice University on the topic [98]. However, we need a reasonable metric to evaluate convergence of the iterative algorithms. Therefore, one can measure the difference between the prior image and the reconstructed image using various error metrics. If the evaluated metric falls below the given tolerance, convergence to that tolerance has been accomplished. Lastly, as previously discussed in Chapter 3, the ordering of the projections is important and can result in a significant computational speedup and using the *hybrid30* method is strongly advised, as it results in faster convergence in our implementation, as seen in Figures 4.10 - 4.12.

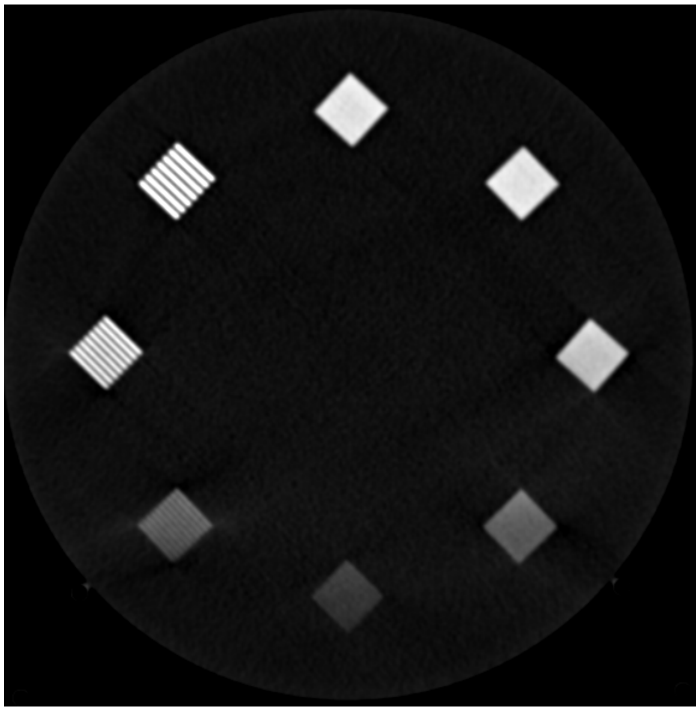


Figure 5.5: The image represents reconstructed line pairs, where spatial resolution is evaluated by what line pair can be seen.

5.1.4 Summary

The ability to conduct research in reconstruction algorithms, for raw attenuation values from refined commercial-CT, is important as it can lead to critical advancements in dose reduction, noise suppression, and image quality. This thesis *gives a description of the use and representation of raw data on current HPC computer hardware.* This parallel hardware has the ability to greatly reduce runtime if it is programmed in an efficient manner. As scans using FFS modes are becoming the norm, the ability to effectively reconstruct under these settings is important. *We have described a method that builds on those techniques from Kachelrie et al., but assisted by user-interaction to achieve a greater level of accuracy. Likewise, we have illustrated why pixel-to-ray interpolation techniques are still slower than the ray-driven Siddon algorithm, but believe there is promise in these techniques as they appear to scale to multiple GPUs easier, as the distribution is block like and simpler.* These methods will be evaluated in Chapter 6.1 and Chapter 6.2. Future advancements in parallel architectures may mean the pixel-to-ray interpolation technique will conceivably become the method of choice. As we are reaching the limits of size and heat on current processors, parallel processing is more commonly being utilized to obtain speedups. For this reason, Toft's complete interpolation technique may become the method of choice in the future as the scalability of ray-tracing techniques over multiple GPUs is difficult. The formal description of an enhanced 2D and 3D Siddon's algorithm, tailored for the GPU with pixel based ray-length scaling and the use of FFS modes, provides a high quality algorithm that is able to scale to multiple GPUs effectively and, in turn, reduce runtime, providing faster results for researchers or clinicians. The evaluation of clinical reconstructions is important, as understanding the quality of reconstruction in a non-simulated environment is required in order to evaluate the superiority of a reconstructions under various parameters. Overall the presentation of these topics is important, as it provides a concise description of implementation considerations and difficulties encountered during the reconstruction of raw attenuation values from commercial-CT; therefore, enabling a broad audience the ability to conduct research in a rather intricate and complicated topic area. The previous chapters are instrumental in developing the required theory in order to conduct qualitative and quantitative analysis in the final chapters of the thesis. We use the methods outlined to analyze various ray-driven techniques in Chapter 6.1, and complete techniques in Chapter 6.2. Last, we conduct a thorough quantitative analysis for CWBP using various HPC hardware in order to understand the impact of each modality on the reconstruction in Chapter 6.3.

Chapter 6

GPU Implementation and Experimental Results

The following chapter represents three formal contributions with an evaluation of techniques used included in the results portion of each section. A complete review of prior techniques is not explicitly detailed, as they were discussed extensively in the previous chapters and referenced accordingly. We first present our detailed multi-GPU Simultaneous Algebraic Reconstruction Technique (SART) based on our previously noted enhancements to Siddon's algorithm, seen in Section 3.2.2, that would be interesting to any medical imaging specialist or multi-GPU programmer. Second, we show a complete system matrix derivation is feasible and has reasonable runtime. We also compare the technique to Toft's previous implementation. Those concerned with complete methods and on-the-fly computation versus techniques requiring lookup tables or ray-tracing would benefit from reading this section. Last, as Filtered Back-Projection (FBP) techniques, seen in Section 2.6, represent most techniques currently implemented on clinical scanners; and therefore show various implementations of Convoluted Weighted Back-Projection (CWBP), seen in Section 2.8, and investigate runtime, effects of algorithmic numerical precision, and noise analysis for all techniques as this type of qualitative and quantitative analysis is not available in current literature. This section would be appropriate to those wishing to understand current implementations of CWBP for raw attenuation values and the numerical stability of CWBP on various HPC implementation.

We remind the reader the definition of the data used, runtime, and analysis is detailed in Section 3.0.1 as they are not explicitly repeated in this chapter. Unless otherwise noted in this chapter, the hardware test environment was that of a 12-core Mac Pro with dual Fermie based Quadro 4000 GPUs. For more information on the hardware and development environment see Appendix B.1.

6.1 Multi-GPU Iterative Reconstruction Algorithm for Medical Imaging

Iterative reconstruction algorithms have been shown to reduce the required number of projections for a complete reconstruction when compared to current conventional techniques like FBP [65]. Decreasing the number of required projections is important, as it would eventually reduce the level of X-ray radiation a patient is exposed to. The iterative methods are computationally demanding and require non-conventional programming, in terms of a multi-core and multi-GPU environment, for a timely reconstructions that are high quality. The environment presenter, AREMI, uses multi-core and multi-GPU systems for reconstruction and analysis. This environment can drastically reduce computational reconstruction time, and lead to faster analysis of parameter space and convergence. The environment is non-typical as it employs multiple resources in an effort to reduce runtime.

Solving large linear systems through conventional means was shown to be impractical in Section 2.13. Likewise computation of the inverse of $\mathbf{A}$ is computationally complicated given the enormous size and varying scanner setup. Likewise, the conjugate gradient algorithm is not well suited, due to the restriction on the system matrix. Toft's dissertation showed that the system matrix $\mathbf{A}^\mathbf{T}\mathbf{A}$ is not sparse and should be avoided [97]. Therefore, non-conventional methods are needed, such as SART. These techniques are computationally expensive but thankfully are parallelizable. Therefore, to provide efficient testing of parameter space, and to ensure fast convergence, AREMI was developed with parallel processing and efficient computation in mind. The environment utilizes available cores and GPUs to iterate to a solution of $\mathbf{f}$ from the transformation $\mathbf{b} = \mathbf{A}\mathbf{f}$.

There are methods from Keck *et al.* [49], and those from Xu *et al.* [109, 110, 62], that use a single GPU for SART type algorithms. Their exact implementation details are still notably absent in literature, but they show the use of a GPU for SART. Jang *et al.* showed a multi-GPU environment is possible and performs well, but their publication lacked any algorithmic description [42]. Comparison to alternative techniques is not currently possible as the specific algorithmic details are lacking and/or the imaging modality is different. Therefore, we have given a clear overview of our exact implementation details in our experiments in order to facilitate easier comparisons in the future.

AREMI has the ability to use multiple CPUs and GPUs in an attempt to reduce runtime for reconstruction and, in turn, provide the means to use higher *ART*-iteration count and more computationally expensive grids and filters to obtain reconstructions with less noise and artifacts. AREMI was developed for cross-platform interoperability as test hardware is composed of typical shared-memory HPC and multi-core / multi-GPU workstations. Through the use of AREMI we have significantly reduced runtime and have been able to test various parameter spaces and the use of AREMI results in faster convergence because more processing power is utilized.

6.1.1 Multi-GPU Algorithms for SART

Although AREMI has the ability to execute many different reconstruction algorithms, we will use SART as an example because SART has the ability to reduce the amplitude of noise in the reconstructions [1, 47]. SART updates the solution vector **f** on a per-projection basis, rather than a per-ray basis as in ART. We have chosen to illustrate the use of a ray-tracing algorithm for ray and ROI intersection, namely Siddon's algorithm [94], which employs enhancements by Jacob's *et al.* [41], and our specific improvements, refer to Section 3.2.2 for more details. We have also extensively tested complete interpolation techniques, as reviewed in Section 2.14 and Section 6.2; for more information beyond these sections see [27, 97, 102, 47]. The environment relies on the fact that each thread, whether on a CPU-core or CUDA-core, computes the intersection of a ray with the ROI.

We present three techniques for distribution of workload on the GPU and analyze how these methods perform in terms of runtime and noise. Specifically, we test the partitioning techniques on two different Back-Projection (BP) methods: a Ray-Driven (RD) BP methods, and a Voxel-Driven (VD) BP technique. The first partitioning technique divides the detectors over the number of devices and is referred to as Alg 1: ray-partitioning, where the second technique divides the ROI over the number of devices and is referred to as the Alg 2: volume-partitioning.

When considering a RD-BP both partitioning methods are required to use an atomic operation in the back-projection step. This atomic operation requires locking data and providing serial access to the variable. This lock is required for every pixel a ray encounters in the back-projection process to ensure a mutually-exclusive write operation. We will show the atomic operation requires approximately 20-percent of the processing time when reconstructing a 1024×1024 slice with 1152 projections and 736 detector channels. We hypothesized that reducing the likelihood of rays writing to the same pixel location would drastically improve performance, but likely increase the estimated-NPS because we can not guarantee the mutual-exclusiveness. To investigate this we developed Alg 3: RD-BP with no mutually-exclusive write operation. The technique relies on a reordering of rays to decrease the likelihood of threads writing to the same location.

When considering a VD-BP no mutually-exclusive write operation is required, and is considered an unmatched pair because the Forward-Projection (FP) is RD and the backward-projection is VD. The VD-BP relies on the projection of pixel elements onto the curved detector channels and linearly interpolating the results. This technique is similar to the CWBP process discussed in Section 2.8 and does not require the atomic operations as rays are not traced through the ROI in the back-projection process and there is no possibility of threads writing to the same location. Refer to Section 2.22.2 for more details on matched pair, unmatched pair, RD-BP, and VD-BP techniques for ART algorithms.

Although many different algorithms were tested, analysis will show that Alg 2: volume-partitioning performs best under our implementation in terms of runtime for both RD-BP and VD-BP techniques. Last, we will also show that although the VD-BP technique performs better in terms of runtime it performs worse in terms of noise in the reconstruction.

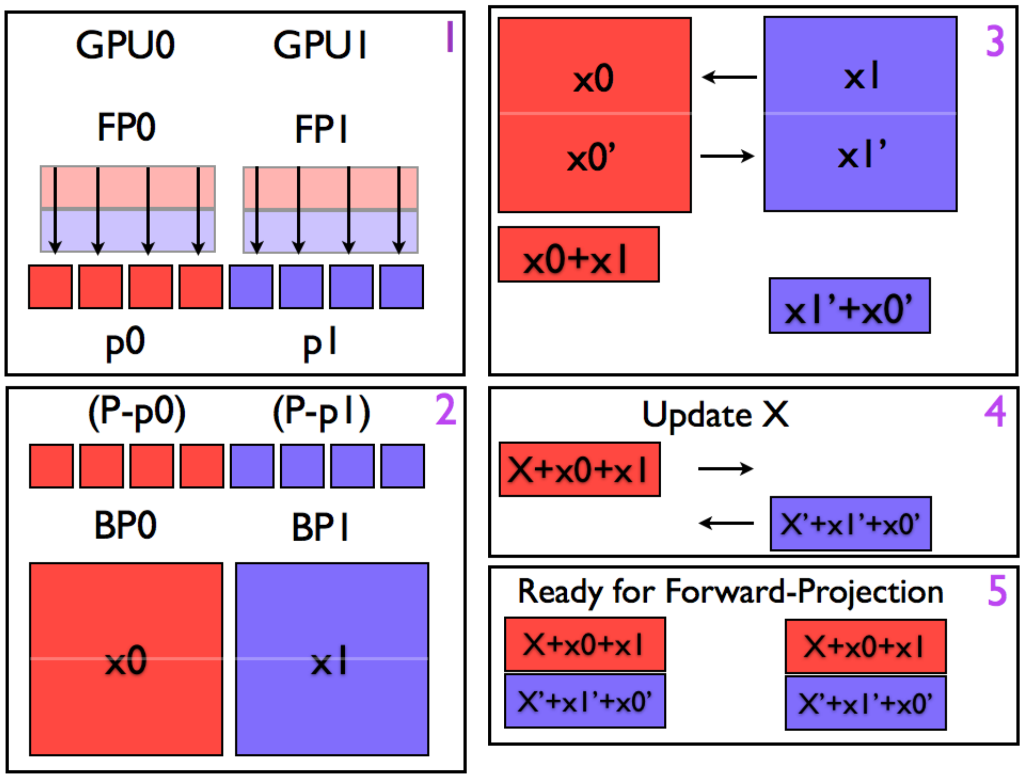


Figure 6.1: The SART update process in the multi-GPU environment is depicted. The FP is carried out by each GPU and creates a half-detector array worth of projection values, seen in (1). Those values are subtracted from the measured projection values P, seen in (2). Each GPU then carries out the BP process, seen in (2), and shares a portion of the result with other computing GPUs, stage (3). The image X is then updated with the back-projected values, seen in (3). Stage (4) shows the data each GPU was responsible for is shared with other processing GPUs. Stage (5) shows the data ready for FP and the entire process can start again, or exit if the required tolerance or ART-iteration count has been reached.

6.1.2 Algorithm 1: Ray Partitioning

Division and allocation of detector-channels to devices, in a multi-GPU environment, given fan-beam reconstruction, would equate to taking the total number of detector-channels and dividing that number by the number of GPUs. For consistency, we will consider the axial scan case where one-slice of the reconstruction region will be reconstructed. Therefore, we will now only consider the channel, or radial sample distribution, and not the complete multi-slice, or longitudinal distribution.

Given a typical scan from the Siemens Definition Flash+ scanner results in 736 channels, and using an example of two GPUs, this would equate to 368 channels or rays for one GPU and 368 channels, or rays, for the second GPU, as seen in Figure 6.3. This workload decomposition results in half the forward-projection process being delegated to each GPU. The technique requires no sharing of data in the forward-projection process, as the calculations are non-dependent. During the back-projection process we will also consider the ray-driven technique making the overall technique classified as a matched pair, rather than the simpler voxel projection technique, but, in this manner, multiple threads/rays can write to a given location and two resultant images need to be computed. As one of the most expensive processes, the movement of data from host-to-GPU or GPU-to-host, we rely on an in-place summation of back-projection textures. Therefore, during the update process there are two images built, updating the solution $\mathbf{f}$, as seen in Figure 6.1. As only a portion of the data is shared with computing GPUs, the technique results in a significant speedup as the ray-tracing algorithm is also distributed and the ray-tracing algorithm is the most computationally demanding portion of the execution being carried out in both the forward and back-projection process.

To ensure full utilization of a GPU, the distribution of threads should result in all GPU Sequential Multiprocessors (SMs) being active. Therefore, for an axial reconstruction with 736 channels, when using Nvidia Quadro 4000 GPUs, with 256 CUDA cores, a maximum of three GPUs could be used; the addition of a forth GPU would not speedup the execution time in this example, as all CUDA cores would not be active. We believe this assertion to be true as smaller test sizes showed when using two GPUs and 128 channels, no performance boost was realized. We currently lack the required hardware to conduct the four to eight GPU test under unified virtual address space (UVA) mode as cluster based GPU resources are inefficient because of the data transfer requirement between nodes. Likewise, a system that uses QuadroPlex units, where four GPUs share a single PCI slot, are also inefficient because there is a high contention for bandwidth as it is shared between all devices.

These techniques are not limited to axial scans and it is important to remind the reader that the algorithm described in Section 3.2.2 also is capable of using helical data. When considering the helical scans, the detector generally employs 736 channels radially and 12-128 detector rows that represent the longitudinal dimension and results in 8832-94202 rays requiring tracing through the ROI. In this situation a blocking strategy can be employed that is relatively simple for CUDA programmers.

The basic algorithm consists of:

1. distribution and memory transfer of detector channels for Forward-Projection (FP) to GPUs $(\frac{rays}{GPUs})$ and the FP of those rays through the Region-of-Interest (ROI), as seen in Stage 1 of Figure 6.1. This effectively results in a reduction of computational cost from approximately $O(k \times \sqrt{n^2+m^2})$ to $O(\frac{k}{GPUs} \times \sqrt{n^2+m^2})$.

2. given the sublist of detector channels each GPU is responsible for, subtract the measured value from the actual projection values P, (i.e. $O(k/GPUs)$ subtraction operations) and Back-Project (BP) those values through the ROI. The RD-BP technique has approximate computational complexity $O(\frac{k}{GPUs} \times \sqrt{n^2+m^2})$ and the VD-BP technique has computational complexity $O(n \times m)$, as seen in Stage 2 of Figure 6.1.

3. share specific portion $(ROI/GPUs)$ of ROI with neighbouring GPUs for summation operation on each GPU, as seen in Stage 3 of Figure 6.1.

4. each GPU is required to share the Stage 3 result with neighbouring GPUs and update their specific results with data received from neighbouring GPUs, as seen in Stage 4 of Figure 6.1.

5. last, the data is consistent on all GPUs and the FP process as seen in Stage 1 is carried out. ** note: ray partitioning is only carried out once as each GPU is always responsible for the same set of rays.

6.1.3 Algorithm 2: Volume Partitioning

Allocating a GPU device responsibility, for a specific region, is beneficial as it is simplistic to distribute a volume and merge results. The sub-volume distribution, seen in Figure 6.2 and Figure 6.4, depicts the SART process for sub-ROI distribution to GPUs. As shown, no explicit large sharing of data is required, but there are some subtleties that require sharing, like the distance metric as seen in Equation 2.51. This technique should scale rather well past two GPUs as it requires minimal data sharing, but we currently lack the hardware to validate this assertion. Generally though, because each GPU has the entire projection values for the forward ray-tracing algorithm and minimal data sharing is required, the technique should perform well in hardware environments with greater than two GPUs.

The basic algorithm consists of:

1. transfer all sinogram values to each GPU. Each GPU then allocates a portion of memory for the subROI it is responsible for $(ROI/GPUs)$, as seen in Stage 1 of Figure 6.2.

2. FP entire radial channel values through subROI and compute approximate p, as seen in Stage 2 of Figure 6.2. The computational complexity for the RD-FP is reduced from $O(k \times \sqrt{n^2+m^2})$ for the single-GPU technique to $O(k \times \frac{\sqrt{n^2+m^2}}{GPUs})$. **note: Each GPU computes actual ray-length through full reconstruction region, as well as subROI. This is required to weight the ray contribution to each pixel accurately in the FP of rays.

3. share p that was computed from the subROI with neighbouring GPUs and subtract updated p from known P, as seen in Stage 2 of Figure 6.2.

4. BP the results to obtain updated subROI, as seen in Stage 3 of Figure 6.2. The RD-BP technique has approximate computational complexity $O(k \times \frac{\sqrt{n^2+m^2}}{GPUs})$ and the VD-BP technique has computational complexity $O(\frac{n \times m}{GPUs})$. **note: RD-BP techniques requires computing the ray-length through the full reconstruction ROI, as well as the subROI. As in Stage 2, this is required to weight the ray contribution to each pixel accurately in the BP of rays through the ROI.

5. the ROI is then updated with the new projection values using a simple summation operations. as seen in Stage 4 of Figure 6.2.

6. last, the FP-process can be carried out again, as seen in Stage 2.

An interesting observation concerning this technique is that often many rays from the projection array do not intersect the region of interest. Although this does cause divergence in the ray-tracing algorithm, the data is well organized and diverges together, as depicted in Figure 6.4. Last, the technique requires some additional parameters to maintain consistency with the other implementations. Mainly, the total ROI value needs to be available to each GPU, not just the sub-ROI. This is necessary to scale the ray appropriately when applying the updates to the forward or back-projection process.

CUDA Device-to-Device Memory Transfers

In the previous sections, we discussed the process of transferring data between GPUs. There are various methods on the current Nvidia GPU to carry out this process, some being faster than others. Specifically, with the release of CUDA 4.0 and Fermie GPUs with UVA mode, it has become possible to have read/write access between GPUs and also the seamless copy of data from device-to-device without the need to directly use host-memory. This means there is not a separate address spaces for the host and each GPU. The UVA mode provides simplified programming, faster memory copies between GPUs, and less host overhead [89]. When possible these methods should be used and should dictate hardware purchases, as non-Fermie GPUs currently do not support this memory transfer method. For further reading on these topics see [13, 89].

6.1.4 Algorithm 3: RD-BP No Mutually-Exclusive Write Operations

We hypothesized that the mutually-exclusive write operation was likely costly. Therefore, we investigated how costly this guarantee was on the runtime. We removed the atomic operations from the RD-BP process and it was found to account for approximately 20% of the total runtime when reconstructing a 1024×1024 image, see Table 6.1. This is a significant portion of the runtime, as runtime typically lasts for minutes with the complete technique. For this reason, we believed, as

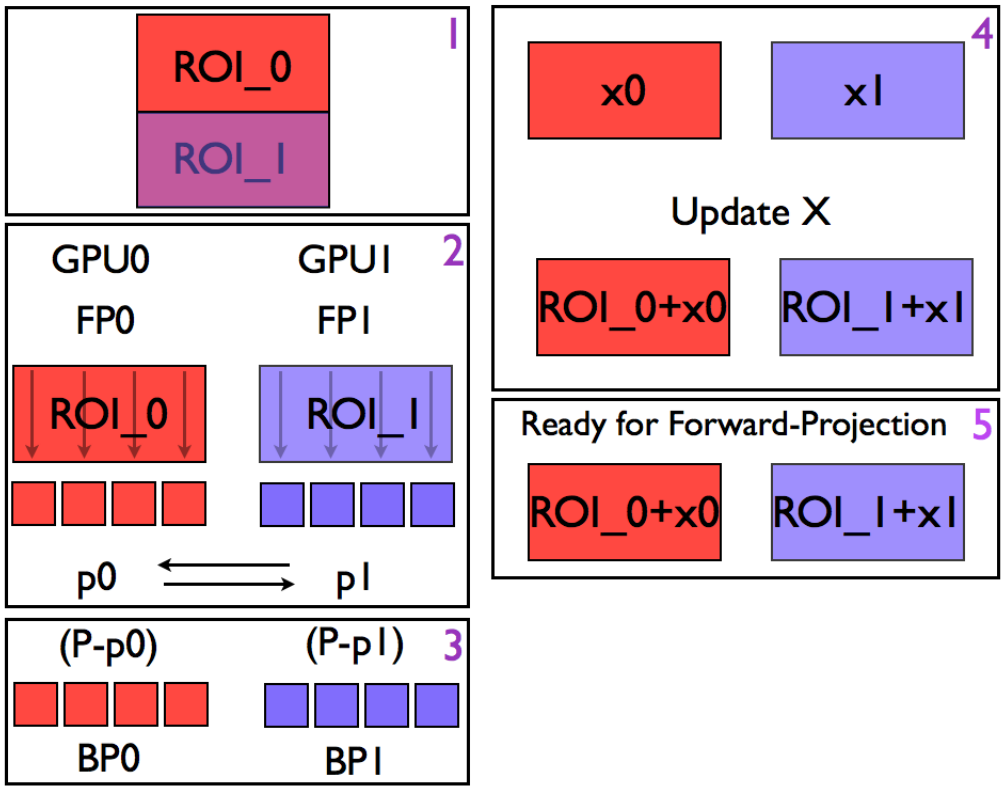


Figure 6.2: The SART update process in the multi-GPU environment is depicted. The FP is carried out by each GPU, but on a sub-ROI rather than the entire ROI as seen in (1-2). The BP process is seen in (4) and requires no explicit sharing of sub-ROI data. This technique eliminates the need to share actual reconstruction region data, which is an costly process as this data is large compared to sharing of the shared projection data.

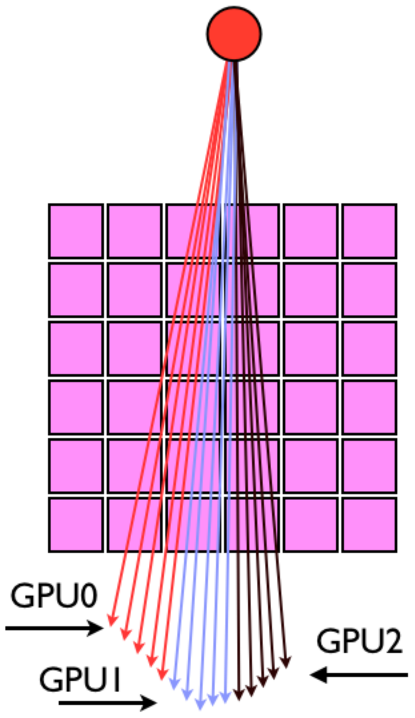


Figure 6.3: Siddon's algorithm using a ray-partitioning technique to distribute workload to multiple GPUs

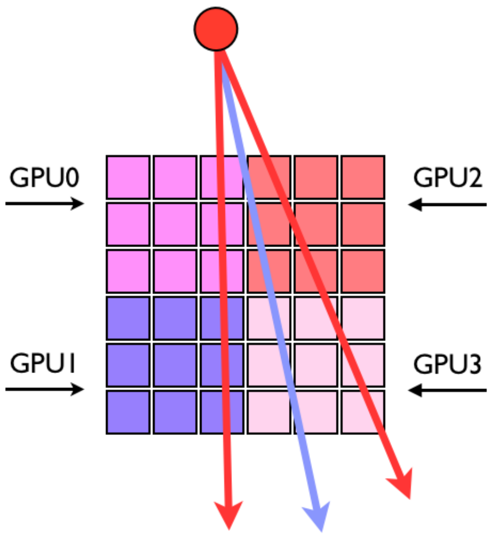


Figure 6.4: GPU0 ROI is depicted as the pink region in the top left. The exaggerated illustration shows all rays to the right of the isocenter channel, blue-ray, do not intersect the ROI of GPU0 and exit early from the ray-tracing algorithm seen on line 15 from Algorithm 3.2.

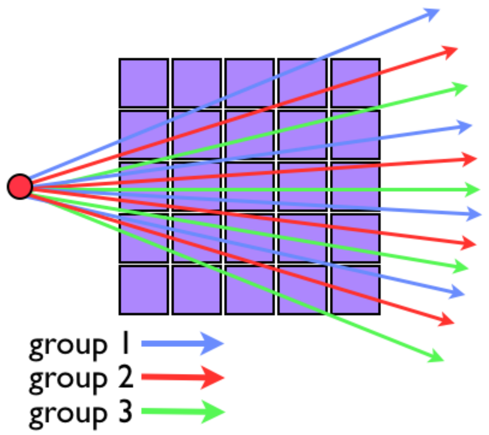


Figure 6.5: The ray-separation strategy is outlined, group. 1 is depicted by the blue rays and one can see there are less likelihood of rays writing to the same pixel locations. After group. 1 is completed group. 2 is processed and finally group. 3.

SART is only an approximation to the true solution, that further approximating the technique may not be entirely detrimental, and could likely produce similar reconstructions but faster. Therefore, a new technique based on ray-distribution was established. Mainly, the ordering of rays for computation was chosen to decrease the likelihood of rays writing to the same location, and removed the mutually-exclusive write operation. We will refer to this technique as Alg. 3 and is depicted in Figure 6.5. There is no explicit algorithm outlined as the technique is simplistic and only requires the radial detector channels to be reordered to decrease the probability of writing to the same pixel, as seen in Figure 6.5.

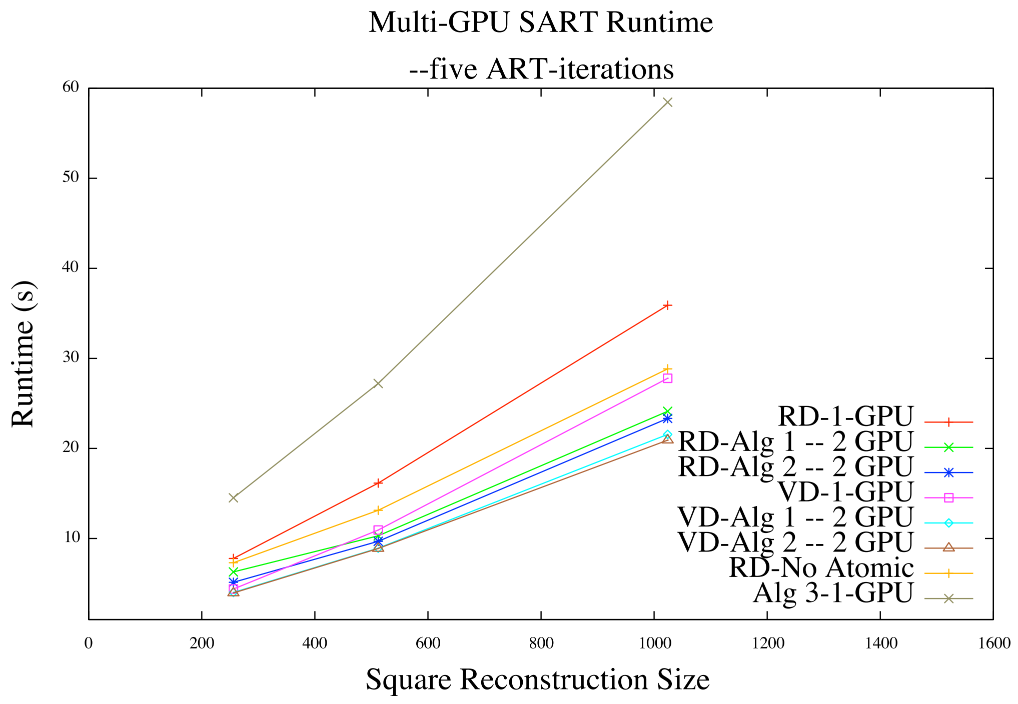


Figure 6.6: The basic GPU SART implementation is compared in Alg.1 and Alg.2. Alg.2, ROI division, is the leader under all reconstruction settings. Testing was conducted with five *ART*-iterations of SART using 1152 projection and 736 channels. The error-bars are not shown as the standard deviation is less than two for all methods.

6.1.5 Results

The basic analysis of all techniques consists of runtime performance and the qualitative metric estimated-NPS, as seen in Section 4.1.1. All techniques use a Ray-Driven Forward-Projection (RD-FP) and various Back-Projection methods. In summary, we have evaluated:

1. RD.Alg.1: Ray-Partitioning with a Ray-Driven Back-Projection (RD-BP) operation
2. VD.Alg.1: Ray-Partitioning with a Voxel-Driven Back-Projection (VD-BP) operation
3. RD.Alg.2: Volume-Partitioning with a Ray-Driven Back-Projection (RD-BP) operation
4. VD.Alg.2: Volume-Partitioning with a Voxel-Driven Back-Projection (VD-BP) operation
5. Alg3: Ray-Driven Back-Projection with no mutually-exclusive write operation
6. RD-BP-*No Atomic* is used to illustrate the cost of the mutually-exclusive operation in the RD-BP methods.

Table 6.1: Multi-GPU Techniques Compared
– Five *ART*-iterations –

Multi-GPU Reconstruction Techniques								
Recon Size	RD Single GPU	RD Alg.1	RD Alg.2	VD Single GPU	VD Alg.1	VD Alg.2	RD-No Atomic	Alg.3 Single GPU
256×256	7.78s	6.29s	5.13s	4.39s	4.01s	3.91s	7.31s	14.77s
512×512	16.15s	10.30s	9.70s	10.94s	8.93s	8.89s	13.13s	27.22s
1024×1024	35.90s	24.13s	23.33s	27.79s	21.56s	20.92s	28.84s	58.45s

RD Techniques

All RD techniques that utilized the mutually-exclusive atomic operation resulted in identical estimates to NPS and are consistent because all methods use standard GPU single-precision floating-point and the basic algorithm is identical. Figure 6.6 and Table 6.1 detail the runtime performance of all techniques. RD.Alg.1 resulted in a 1.49x speedup and RD.Alg.2 a 1.54x speedup when compared to the single GPU implementation for 1024×1024 reconstruction size. These results show a decrease in runtime and will provide convergence faster than a single GPU implementation. We believe under a GPU setting, with greater than two GPUs, RD.Alg.2 would produce substantially faster reconstructions, as it does not require the large volume of data transfer as required by RD.Alg.1, seen in stages 3-4 in Figure 6.1.

VD Techniques

Similar to the previously mentioned RD techniques, the VD methods all resulted in the same estimated-NPS as the methods all utilize the same algorithm but the processing is different. The runtime analysis results in a 1.29x speedup for VD.Alg.1 and a 1.33x speedup for VD.Alg.2. The justification for the VD methods performing better in terms of runtime is twofold. First, the RD methods do not require the mutually-exclusive constraint, which is shown to be costly in Table 6.1. Second, the VD technique relies on much faster pixel projections, rather than a more costly ray-tracing method. Specifically, the VD method has computational complexity of $0(n^2)$, rather than the RD method that requires approximately $O(k \times \sqrt{m^2 + n^2})$, where n is the dimension of the square reconstruction region. Likewise, the RD back-projection technique from an algorithmic perspective is more costly, as it requires an expensive ray-tracing technique as seen in Section 3.2.2. Whereas, the VD technique has a simpler algorithmic routine.

Alg. 3 Approximate RD Methods

The mutually-exclusive operations required for the RD methods were shown to be costly and, for this reason, we believed a well-ordered set of rays would provide a decreased probability of writing to a pixel location while in the process of being modified by another thread. The technique is only

Table 6.2: Estimated-NPS for ROIs

SART Reconstruction Techniques, results NPS in units$^2 10^{-4}$		
Acceleration Technique	RD-BP	VD-BP
ROI 1	13.8	17.7
ROI 2	1.02	8.0
ROI 3	7.70	11.8
ROI 4	8.80	17.1
ROI 5	8.40	14.0
ROI 6	8.23	15.6
ROI 7	0.51	5.2
ROI 8	0.37	8.3
ROI 9	0.49	8.2

applicable to the RD-BP method as the VD-BP methods does not require mutually-exclusive write operations. The technique, is theoretically sound as depicted in Figure 6.5, and result in L2-error of 0.013 difference in reconstruction image, versus that reconstructed by the RD technique that utilizes the atomic operation to guarantee mutual-exclusiveness during writing. The L2-error metric is similar to Root Mean Squared error analysis without the division by the number of samples. Figure 6.6 and Table 6.1 show the runtime for the technique and is substantially higher than all other methods; specifically requiring an average of 58.45s to reconstruct a 1024×1024 image. Similar results were observed for smaller images. We attribute the increased runtime performance, in the absence of the atomic operation, to the fact that the data is no longer well ordered for branches during execution. When the data is partitioned to decrease the need for mutually-exclusive operations it is not well ordered and increases the likelihood of not having full-*warps*, causing increased runtime. For more information refer to the discussion in Section 3.2.2 concerning fully active *warps*.

Estimated-NPS

As previously mentioned the estimated-NPS within VD and RD techniques was the same when considering Alg.1:ray-partitioning and Alg.2:volume-partitioning. However, comparing the two back-projection techniques we see from Table 6.2 that the VD method has significantly more noise in all ROIs than the RD method. This is likely the result of the RD method more accurately describing the area-based detector in the physical system by tracing the rays in the back-projection process. Nonetheless, it is interesting to see a noise comparison of the two techniques utilizing raw-attenuation values as previous literature shows the matched pair technique has less ringing artifacts and should be equivalent when using simulated data [112, 49].

6.1.6 Summary

The previous section has shown the analysis of three different algorithms for partitioning of workload to multiple GPUs in the aims to reduce the computation time for an unmanageable problem in the aims to make it tractable. Specifically, we have shown:

1. *the use of different workload partitioning schemes in a multi-core and multi-GPU environment for CT-reconstruction using a RD-FP, RD-BP, VD-BP, RD-BP-*No Atomic*, and a relaxed RD-BP technique in* AREMI. The use of multiple GPUs is complicated as the designer has to consider thread and resource management and merging of solutions. However, has been shown to result in significant speedups to the forward and back-projection process of the SART algorithm. Although the technique is not linear in speedup with the addition of GPUs, the techniques still reduce runtime significantly in both RD and VD methods and plays an important role in the facilitation of SART in clinical settings, where fast reconstructions are required.

2. *the relaxed RD-BP technique, which does not utilize a mutually-exclusive write operation in the back-projection process, did not realize a speedup. We also reinforced the likely importance of proper data ordering to facilitate full-warps.*

3. when considering estimated-NPS, *the use of raw-attenuation values the RD-BP method produced less noise than the VD-BP in our implementation, even though previous literature suggests they are equivalent when using simulated data.*

We have demonstrated that SART is distributable across multiple GPUs. This distribution can speedup reconstruction without increasing error. The analysis has provided the means to facilitate SART methods in clinical-CT where fast and accurate reconstruction are required.

6.2 Evaluation of the Use of a Complete System Matrix on the GPU for Iterative Techniques

The use of iterative techniques for reconstruction of CT images is important, as it has been shown to produce less noisy reconstructions and require less projection data. The decrease in noise is attributed to relaxation parameters and regularization during *ART*-iterations, while the trimming of projection count is attributed to the Fourier Slice Theorem which positions the data on a polar grid in frequency space when using FBP. For this reason, over-sampling the interior frequency region is required for reasonable sampling of the periphery [30, 65]. The fascination with ART [27] is interesting, as, with these methods, less projections are required and, in turn, less exposure to radiation is needed. There are complete methods that choose to compute the complete transformation matrix and store it for easy lookup, but these have been shown to be too expensive in terms of storage (Section 2.17). This thesis aims to investigate a current approach to the complete method through ray and ROI interaction by means of computing the ray-intersection when required in a multi-GPU environment. The technique is expensive, as it requires each pixel to be tested against the ray, but the method is important as varying degrees of ray-width can be used, as detailed in Section 2.14. These interpolation techniques do not require that a ray is infinitely thin, as in ROI traversal seen in Siddon's algorithm [94]. We do not show the results of varying the ray-width on reconstructions as this section focuses on the runtime analysis of a current implementation of a complete methods for ART.

There are various methods available for calculating the ray and ROI interaction. Techniques, such as Siddon's algorithm, provide a huge speedup and are currently widely used in the research community, because the method does not require a complete check of all pixels or voxel interaction [49, 42] and has approximate computational complexity of $O(k \times \sqrt{m^2 + n^2})$. However, the ability to use varying ray-width is difficult and is limited to varying the pixel interaction size. Alternative techniques are costly as they check all pixels against each ray, but varying ray width and other interpolation specifics can be built-in. The resulting cost is $O(n \times m)$ and $O(n \times m \times slices)$ for axial and helical reconstructions for each forward and back-projection of rays. Toft's dissertation provides a thorough review of the subject area but only implemented the techniques on a single-processor [97]. Currently, multi-processor technology, such as the GPU, is good at performing parallel computation in a cost-effective manner. We present a GPU implementation of the complete technique for the system matrix formulation that is required to solve the inverse Radon transform in linear algebraic form $\mathbf{b} = \mathbf{Af}$. The technique is important as it scales well with increased computational power and could, one day, out perform ray-driven techniques as they are difficult to parallelize as seen in Section 6.1

The ability to model different characteristics of the Radon transform can be beneficial [97, 102]. The results in Toft's thesis, that were also tested by Vlček, showed that the level L2-error was partially dictated by the interpolation scheme used [102]. Toft tested several different types of interpolation schemes and, overall, the nearest neighbour routine, seen in Section 2.14, performed best. As we are not evaluating the qualitative impact of the GPU-based techniques, we focus testing on the nearest neighbour technique on various reconstruction sizes. Xu *et al.* implemented various interpolation schemes on the CPU hoping to get guidance on a future GPU implementation [19]. However, their current publications focus mostly on Siddon's algorithm for the forward-projection process and voxel projection for the back-projection process.

We have found no current algebraic techniques that utilize a complete system matrix in recent literature. This is likely because the system matrix for a small reconstruction size of 512×512 is large, in the order of a terabyte as seen in Section 2.15 when using single-precision floating-point precision. This makes computation difficult and lookup tables costly as frequent memory lookups slow down GPU computation significantly. For these reasons, we believe the implementation of the complete method is important, as it is becoming more feasible on current hardware and could lead to results that outperform Siddon's algorithm. We will show, with the addition of computation on the GPU, a unmanageable problem is made more reasonable in terms of runtime with a 19x speedup versus our single-CPU implementation as seen in Table 6.3.

6.2.1 Proposed Implementation

Each GPU has many CUDA-cores and each of these cores is responsible to calculate the interaction between a pixel and the ray in question. This process is expensive as will results in 2^{16} pixel and ray interactions for the case of a 512×512 reconstruction. The process of creating a matrix row of the system matrix $\mathbf{A}$ is expensive, as this requires a test of all pixel elements against the ray. However, this is a parallel operation, and can easily be distributed in a blocked structure to a GPU, or GPUs, as seen in Figure 6.7.

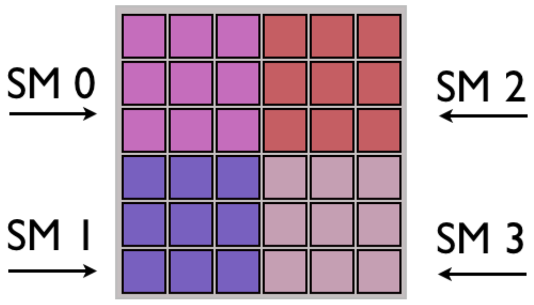


Figure 6.7: The blocking structure of the algorithm is displayed on a four Sequential Multiprocessor (SM) on the GPU where each block is distributed to a SM for computation. The blocking structure is dynamic and tunable.

6.2.2 GPU Workload Partitioning

The algorithmic details are similar for implementation on the GPU or CPU. Mainly, on the GPU we distribute half the volume to each GPU to ensure contiguous reads during processing. As the data is stored in an efficient vector representation this results in dividing the vector equally between computational devices. The only difficulty with the technique, and what prevents a linear speedup with the addition of GPUs, is the region when the results from each reduction need to be shared between computing devices.

6.2.3 Parallel Dot-product

In this section we will describe an efficient shared-memory implementation of ART that is designed for the the GPU. Equation 6.1, originally from Section 2.17, relies on a dot-product operation. This operation needs to be efficient in order to achieve fast runtime as it has complexity $O(n \times m)$ and is a frequency operation.

$$\mathbf{f}^{k+1} = \mathbf{f}^k + \lambda * \left(\frac{\mathbf{b}_i - \mathbf{a}_i^T \mathbf{f}^k}{\mathbf{a}_i^T \mathbf{a}_i} \right) \mathbf{a}_i \tag{6.1}$$

The technique implemented using shared-memory on the GPU is similar to the parallel reduction example, or Algorithm 7, in the techniques presented by Harris from Nvidia Corp [33]. The reduction method uses an efficient shared-memory implementation that relies on a blocking structure that uses a decreasing thread count to sum values. For example, take the dot-product of two vectors of 256 values. One could use 256 threads initially to compute pair values and store them in shared-memory. After that value is computed in shared-memory, the first 128 threads sum a pair of values and store the result in shared-memory. Then the next 64 sum a pair of values and store there result in the first 32 locations of shared-memory. This process continues until only two values are remaining and thread zero sums these two values up. This process is depicted in the illustration from M. Harris's web seminar, seen in Figure 6.8. All dot-products operations for ART, seen in Equation 6.1, are computed in this manner and the method has high throughput.

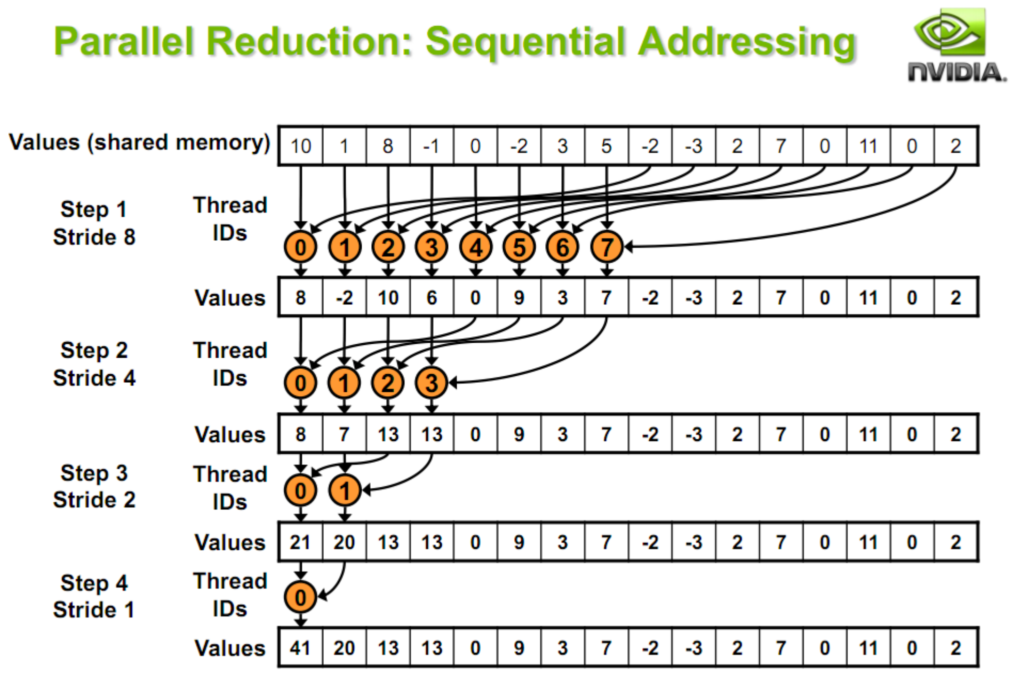


Figure 6.8: Shared memory is used to store the values. For every *ART*-iteration only half the threads are active and used to reduce the computation. The image is originally found in the web seminar from Harris [33].

6.2.4 Optimal Block-size Learning

Figure 6.7 shows the blocking structure, or workload per thread, is a dynamic parameter that can be tuned for the system in question. During testing, we realized this parameter had a severe impact on runtime and could equate to a 20% decrease in runtime if the most efficient parameter is chosen, as seen in Figure 6.9 when comparing the block size 32 to 512. Considering we often see runtime in the tens of minutes for ART or SART algorithms using the complete system matrix representation, this results in a substantial speedup. After testing on several GPUs we quickly realized this parameter was not optimal on all GPUs. Therefore, we opted for a simple learning phase where several test cases were formulated and the optimal blocking structure used. The learning phase is relatively simple and we found a correlation between small problem sizes or reconstruction and large problem sizes. This connection resulted in the ability to learn on small problem sizes that can be computed quickly. Those results were also applicable to the larger problem sizes, and the cost of learning was kept to a minimal.

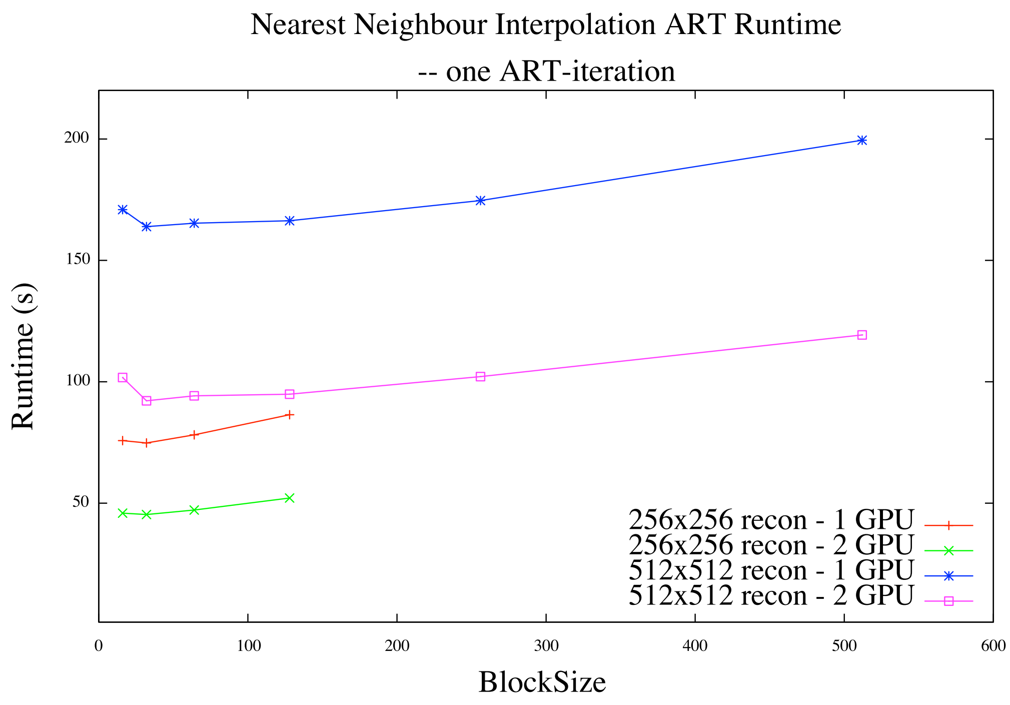


Figure 6.9: The number of GPUs, reconstruction time, and blocking strategy is shown. The results show a correlation of the blocking strategy used in smaller reconstructions and larger reconstructions. Testing showed consistently faster runtime was achievable when using a block size of 32 for the dot-product reduction. The error-bars are not shown as the standard deviation is less than two for all methods.

6.2.5 Results

The complete methods for the derivation of the system matrix $\mathbf{A}$ are expensive and often authors, such as Toft or Vlček, opted to include only *ART*-iterations or small problem sizes in their results. Runtime of over 1000s, for even simple single-slice problems, with smaller sinogram was common [97, 102]. Vlček dissertation showed interpolation techniques taking several hundred seconds

Table 6.3: Complete Techniques in SART Compared
– one *ART*-iteration –

Multi-GPU Reconstruction Complete Techniques				
Recon Size	Single CPU	Single GPU	Dual GPU	Toft Slow ART
512×512	1766.2s±2.21	163.9s	92.5s	1927.3s±2.45

and sometimes thousands. For this reason we developed a parallel algorithm to distribute the computationally difficult problem over GPUs in the aims to decrease runtime.

We have shown a method that constructs the system matrix on-the-fly, row-by-row, and on the dual-GPUs that is able to complete one complete ART-iteration of ART using 1152 projections, 736 channels, and a 512×512 *clinical reconstruction size in 92.15s and the learned blocking structure as seen in Figure 6.9.* Therefore, showing the on-the-fly methods are feasible on current HPC. Although we are able to obtain faster results of only $2.8s$ using dual GPUs, see Section 6.1.5, and a matched pair forward and back-projection ray tracer based on Siddon's algorithm, we still feel the results are important, as the technique is easily distributable to multiple GPUs.

A learning technique for optimal blocking strategy was presented and showed the correlation between small problem sizes and larger reconstructions. Figure 6.9 shows the technique resulted in an approximate 20% speedup in reconstruction when comparing the optimal 32 block to the slowest 512 block reconstruction. We found the technique to be system-dependent, as results on different GPUs have different optimal blocking strategies and, for that reason, the learning procedure should be carried out at least once when a new GPU is used.

Comparison to Alternative Complete Techniques

In order to establish a comparative evaluation of our methods in terms of runtime and to validate we have an efficient CPU-implementation of the complete technique we use Toft's source code for testing purposes. Toft's source code is approximately 16 years old and we modified it for current compilers and hardware to run some limited testing. Although we were only able to test on a simulated problem the computation time should be similar on actual raw attenuation data. Toft termed his complete method *slow-art*, and is comparable to our implemented method. The techniques were tested on a 512×512 reconstruction with 1152 projections, 736 channels, and both methods used nearest neighbour interpolation. The direct comparison of methods, in terms of noise, is difficult because Toft uses specialized output images. Likewise, the input only accepted simulated data. Nonetheless, the problem sizes are equal and in this evaluation we are interested in runtime for a given problem size. Therefore, we feel the comparison is valid and establishes we have implemented an efficient CPU-method for comparison purposes as the results show our CPU-implementation actually outperforms Toft's, as seen in Table 6.3.

Table 6.3 shows Toft's source-code resulted in an average runtime of 1927s$\pm$2.45 for one *ART*-iteration. We compared Toft's serial implementation to our serial version which resulted in an average runtime of 1766s$\pm$2.21 where we obtained a slight speedup. Likewise, *the dual-GPU implementation presented, with learned blocking structure, resulting in a runtime of 92.15s, and an approximate speedup of 21x over Toft's complete technique.* The justification for the performance increase is the learned blocking structure that optimizes the use of cache on the GPU, a multi-GPU algorithm that makes the problem more manageable, and the highly parallel nature of the GPU. We believe it is likely that Toft's code could be performance tuned for current hardware. This would likely result in a mild speedup, but not a 21x speedup as we have demonstrated. Although we were unable to perform noise analysis on Toft's reconstructions or use actual raw-attenuation values we believe the runtime analysis is fair as all techniques used the same problem size. The software package from Toft is large and rather complicated and for that reason further modifications are difficult.

The ability to use a complete method that tests all pixel locations against each ray is important as different interpolation techniques are possible and varying ray width is made simpler. We have shown a algorithm that aids in reducing the runtime of ART-methods. When the runtime is decreased the parameter space can be investigated easier and makes the technique more applicable to clinical-CT where timely reconstruction are important.

6.3 Comprehensive Analysis of HPC Methods for CWBP

The following section was jointly produced with Dr. C. Mendl, from Applied Mathematics at the Center for Mathematics and Analysis, Technische Universität München, Germany in the preparation of [17]. Mendl contributed the methods seen in Algorithm 1, which is based on a graphics pipeline implementation of Convoluted Weighted Back-Projection (CWBP).

While CUDA is the most popular software architecture for GPU programming, recent studies have proposed to exploit the various built-in, hardwired graphics pipeline components (as facilitated by the DirectX or OpenGL programming interfaces), instead of employing the graphics card as multi-processor only. In this section, we directly compare DirectX, CUDA and conventional shared-memory High-Performance Computing (HPC) CPU implementations of the CWBP algorithm with fan-beam geometry, as seen in Section 2.8. Our comparison concerns runtime as well as accuracy, and we estimate the NPS of the error with a double-precision serial implementation as reference.

Interactive CT imaging techniques, in the medical and physical sciences, have recently become the focus of active research. For example, they are used to enable assistance during surgery [67], for monitoring, or in (closed-loop) feedback experiments. The associated real-time data-processing tasks can profit considerably from the broad availability and exponential performance growth, of consumer graphics processing units (GPUs) and conventional HPC [107, 66]. However, the GPU programming paradigm is still rather intricate, when compared to traditional CPU implementations. To render this topic accessible to a broad audience, this paper provide a self-contained, pedagogical implementation of the prototypical forward and inverse Radon transform [79, 47] on the GPU shader pipeline. Likewise, we provides implementation details on an alternate GPU modality that treats the GPU as a conventional multi-processor through a CUDA implementation. Last, we provide implementation details of a highly parallelizable implementation on conventional HPC through a multi-threaded reconstruction algorithm.

The analysis is comprised of a runtime study of all methods under various hardware platforms, using single and double-precision floating-point when available. The qualitative analysis consists of comparison to a *base-line* double-precision serial implementation under double-precision where error is calculated. Likewise, through sampling of homogeneous, or uniform, areas of an ACR-accredited phantom, an estimated-NPS, originally presented Section 4.1.1, is calculated for each implementation.

In the following section, *we will show that all GPU-based implementations outperform a traditional shared-memory implementation in terms of runtime. We will also establish that single-precision floating-point is suitable for CWBP as minimal differences in quality is found when using double-precision.*

6.3.1 Analytic Framework: FBP

In what follows, we briefly recall the theoretical framework for the FBP algorithm [47, 55, 77], which traces back to the original work by Radon [79, 80].

The fan-beam geometrical setup is illustrated in Figure 3.5. A point-like radiation source S and a corresponding array of detectors rotate around the subject (rotation angle β), with the detectors arranged on a circular arc (fan angle γ). D denotes the fixed distance between S and the origin (rotation center). The detectors measure each fan projection $p_\beta(\gamma)$. According to the derivation in [47], we may write the reconstruction formula as:

$$f(x,y) = \frac{1}{2}\int_{0}^{2\pi} W_\beta\big(\mathrm{Rot}(\beta)\,(x,y)^T\big)\,\mathrm{d}\beta, \tag{6.2}$$

with the rotation matrix $\mathrm{Rot}(\beta) := \left(\begin{smallmatrix}\cos\beta & \sin\beta \\ -\sin\beta & \cos\beta\end{smallmatrix}\right)$ and

$$W_\beta(x,y) := \frac{1}{L(x,y)^2}\,\tilde{Q}_\beta(\gamma'(x,y)),$$

where

$$L(x,y) := \left\|(x, D-y)^T\right\| = \sqrt{x^2 + (D-y)^2},$$
$$\gamma'(x,y) := \arctan\left(\frac{x}{D-y}\right).$$

These formulas for L and γ' can directly be obtained from Figure 3.5 when setting $\beta = 0$. The function $\tilde{Q}_\beta$ is defined as a convolution, seen by:

$$\tilde{Q}_\beta(\gamma') := (\tilde{p}_\beta \star c_f)(\gamma') = \int_{-\gamma_m}^{\gamma_m} \tilde{p}_\beta(\gamma)\, c_f(\gamma' - \gamma)\, \mathrm{d}\gamma, \tag{6.3}$$

with

$$\tilde{p}_\beta(\gamma) := p_\beta(\gamma) \cdot D \cos\gamma,$$
$$c_f(\gamma) := \left(\frac{\gamma}{\sin\gamma}\right)^2 c(\gamma),$$

as well as, formally,

$$c(\gamma) := \int_{-\infty}^{\infty} |\rho|\, \mathrm{e}^{2\pi j\, \rho\, \gamma}\, \mathrm{d}\rho = \left(\mathcal{F}^{-1}\, |\cdot|\right)(\gamma).$$

In the last equation, $\mathcal{F}^{-1}$ denotes the inverse Fourier transform. If we ignore the factor $\gamma/\sin\gamma$ in the filter c_f to first approximation, and express the convolution (6.3) via Fourier Transforms, we arrive at:

$$\tilde{Q}_\beta(\gamma') = \Big(\mathcal{F}_1\, \tilde{G}(\cdot, \beta)\, |\cdot|\Big)(\gamma') \equiv \int_{-\infty}^{\infty} \tilde{G}(\rho, \beta)\, |\rho|\, \mathrm{e}^{2\pi j\, \rho\, \gamma'}\, \mathrm{d}\rho, \tag{6.4}$$

with

$$\tilde{G}(\rho, \beta) := (\mathcal{F}_1\, \tilde{p}_\beta)(\rho) = \int_{-\gamma_m}^{\gamma_m} \tilde{p}_\beta(\gamma)\, \mathrm{e}^{-2\pi j\, \rho\, \gamma}\, \mathrm{d}\gamma. \tag{6.5}$$

Summarizing, the FBP algorithm consists of two steps: first, the filtering operation (6.3), which we implement via the Fourier transformation in (6.4) and (6.5), and second, the actual back-projection (6.2), as an integral over β.

6.3.2 Algorithm 1: DirectX Graphics Pipeline

Figure 6.10 illustrates how the back-projection operation (6.2) can be mapped to the graphics pipeline. The central idea of our implementation is as follows: we affect the rotation by β in equation (6.2) by rendering a rotated full-screen rectangle for each β. The four rotated vertices have to be set up just once. The input texture has the following structure: each row corresponds to a rotation angle β (see Figure 3.5), and each column to a detector at fan-angle γ. Back-projecting can be visualized as "smearing" the highlighted blue bar in Figure 6.10 over the reconstruction plane, which is schematically indicated by the grayscale pixel lines.

Algorithm 6.1 shows the pixel shader code of the back-projection rendering pass. The code is quite concise since the rotation by β has already been accomplished implicitly by the screen quads.

Convolution

Concerning the filtering operation, the graphics pipeline has to carry out the Fourier transforms, seen in Figures (6.4) and (6.5), for each row of the input texture (i.e., projection β). The entries of

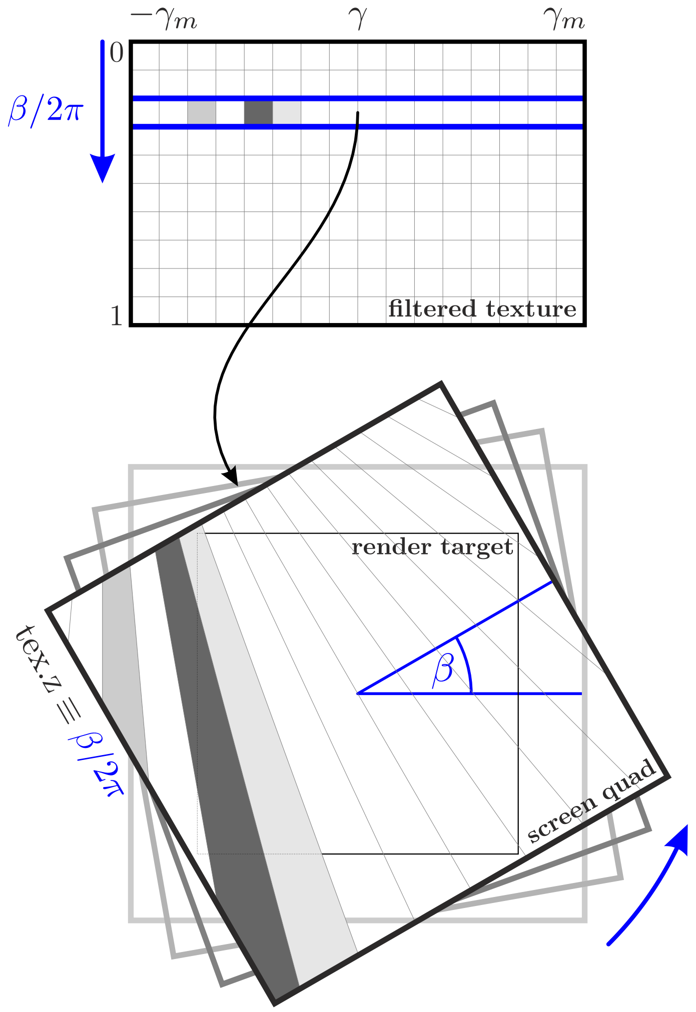


Figure 6.10: Implementing the back-projection step in a single draw call by employing rotated squares ("screen quads"). Each column of the filtered input texture corresponds to one detector on the fan (with γ between $-\gamma_m$ and γ_m), and each row to a rotation angle β (defined in Figure 3.5). The 4 vertices of each screen quad are rotated by β in the setup stage of our implementation. Sampling the rows in the input texture is affected via the tex.z texture coordinate, which has the same value for all four vertices within a quad.

the to-be-filtered input vector correspond to discretized values of γ. We zero-pad the input vector to a power-of-two, in order to prevent spatial domain aliasing, and also to speed up the FFT. In our implementation, we employ the well-known Cooley-Tukey FFT algorithm [11].

Interpolation

We use the available DirectX hardware accelerated linear interpolation functions. These techniques prevent multiple reads from the sinogram as in the CPU techniques that will be discussed in Algorithm 3. The DirectX kernel uses a single memory read that is hardware accelerated to provide linear interpolation of the sinogram values in the radial direction.

Algorithm 6.1 DirectX Back-Projection Pixel Shader

```
Require: $x, y$: coordinates (from texture coordinate register); $z$: projection index; $\mathrm{tex}$: input texture. Constants: $D$: distance between rotation center and radiation source; $C$: texture coordinate factor; $A$: fan arc length; $N$: number of projections
1: $(x', y') \leftarrow (x, D - y)$
2: $\gamma' \leftarrow \arctan(x'/y')$
3: $\mathrm{Lsq} \leftarrow x'^2 + y'^2$
4: $t \leftarrow \begin{pmatrix} C \cdot (\gamma'/A + 0.5) \\ z \end{pmatrix}$ return $\mathrm{tex.sample}(t)/(N \cdot \mathrm{Lsq})$
```

6.3.3 Algorithm 2: CUDA Implementation

The CWBP algorithm previously described is employed where each pixel location is projection onto a curved sensor array. This is equivalent to projecting a given pixel location onto a sinogram and linear interpolating the results. The technique is efficient as an inner-loop in the CUDA-kernel can be used to iterate through the different projection values and prevents some repetitious computation as seen in line 5 of Algorithm 6.2. We also provide testing on the use of *fastmath*-intrinsic functions that are faster but less accurate [13]. The intrinsic functions can be activated by the NVCC-compiler flag (*−use_fast_math*).

Algorithm 6.2 CUDA Back-Projection Kernel

```
Require: $x, y$: coordinates (CUDA blockDim, blockIdx, and threadIdx can be used to calculate); Constants: $D$: distance between rotation center and radiation source; $isoCenterChannel$: defines isocenter channel location; $Beta\_angle$: projection start angle; $N$: number of projections; $beta\_angle\_increment$: angle increment value; $fan\_angle\_grid$: angle between radial fan channels
1: $\theta \leftarrow \arctan(x/y)$ :arctan2 also a good choice
2: $r \leftarrow \sqrt{x^2 + y^2}$
3: $IV \leftarrow D^2 + x^2 + y^2$
4: $IV2 \leftarrow 2 \times D \times r$
5: for $i < N$ do
6:     $L2 \leftarrow IV - IV2 \times \cos(Beta\_angle - \theta)$
7:     $\gamma \leftarrow \arcsin\left(\frac{(r \times \sin(\theta - Beta\_angle))}{\sqrt{L2}}\right)$
8:     if $\gamma < max\_fan\_angle$ then
9:         $channel\_index \leftarrow isoCenterChannel + (\gamma \times fan\_angle\_grid)$
10:        $integralSummation \leftarrow integralSummation + \frac{texLookup(channel\_index)}{L2}$
11:    end if
12:    $Beta\_angle \leftarrow Beta\_angle + angle\_increment$
13: end for
14: $reconstruction[calculatedIndex] \leftarrow intergralSummation$
```

Convolution

Through experimentation, one quickly learns that the filtration process, through typical convolution, is much too expensive, having complexity $O(k^2)$. However, employing a transform to frequency space through efficient CuFFT libraries [68] results in a efficient point-wise multiplication, which can be carried out to obtain the same results but with complexity $O(k \log k)$. As the FFT libraries are optimized, this is the method of choice, basic filtering in the frequency space. We obtained the qualitative values by using a zero-padded transform to frequency space to the second-next-higher power-of-two, given 736 detector channels. This resulted in 2048 values, with the final values being zeros. In order to not add any high frequency drop-off at the transition from detector channel 736 to 737, we employed a linear decrease in values to entry 800. Finally, template-methods were used for easy comparison of numerical accuracy, given the level of precision.

Interpolation

Algorithm 6.3 CUDA texture setup for linear interpolation and single-precision floating-point

```
1: channelDesc := cudaCreateChannelDesc(sizeof(T), 0, 0, 0, cudaChannelFormatKindFloat)
2: cudaMallocArray( cuArray, channelDesc, detectorChannels, projectionCount)
3: cudaMemcpyToArray( cuArray, 0, 0, hostProjections, detectorChannels * projectionCount *
   sizeof(T))
4: $\rightarrow$ set texture parameters
5: $tex.addressMode[0] \leftarrow cudaAddressModeWrap$
6: $tex.addressMode[1] \leftarrow cudaAddressModeWrap$
7: $tex.filterMode \leftarrow cudaFilterModeLinear$
8: $tex.normalized \leftarrow true$
9: $\rightarrow$ Bind the array to texture
10: cudaBindTextureToArray(tex,cuArray,channelDesc)                ▷ Internal CUDA Call
```

The CUDA implementation details are a slight bit simpler as the projection data, or sinogram, is loaded into texture memory that supports linear interpolation, as seen in Algorithm 6.3. However, although the texture-based approach supports hardware based linear interpolation and caching, the method cannot use double-precision. Therefore, an alternative method uses standard global memory storage on the GPU. Last, the technique uses an inner *for*-loop to accomplish the *beta* projection-based rotations.

Multi-GPU Implementation

The use of multiple GPUs for CWBP is accomplished through a volume-partitioning technique that distributed the reconstruction region evenly over each GPU. The method effectively reduced the number of pixel-projection computations to $O(\frac{n \times m}{GPUs})$ per-GPU from $O(n \times m)$ in the single-GPU methods.

6.3.4 Algorithm 3: CPU Serial and Parallel Implementations

The implementation of CWBP is similar to the CUDA pseudocode seen in Algorithm 6.2. We investigated many different workload policies and found a volume partitioning strategy worked well as the data is laid out in an efficient contiguous vector representation. The technique distributes the reconstruction region evenly over each GPU. The method reduces the number of pixel-projection

Table 6.4: Techniques Summarized

CWBP Reconstruction Techniques			
Method	Alg1	Alg2	Alg3
Convolution	Cooley-Tukey FFT	cuFFT	FFTW3
Back-Projection	CWBP DirectX Graphics Pipeline	CWBP CUDA	CWBP Shared-Memory
Interpolation Technique	Linear	Linear	Linear

computations to $O(\frac{n \times m}{CPUs})$ per-CPU from $O(n \times m)$ in the single-CPU method. The data structure used is a multi-slice *valarray* from the Standard Template Library (STL). The structure was originally created for efficient mathematical operations on values and works well for our implementation because we require many vector based operations on values.

Convolution

The main difference in this stage, versus the CUDA implementation, is the use of the FFTW3 library for the forward and inverse transform during filtering [22]. This library was chosen because it is efficient and, like the other implementations, a standard ramp-filter was used.

Interpolation

As the CPU does not have hardware accelerated interpolation techniques we are required to lookup multiple values in the final stage of CWBP. This is a major shortfall of the technique and results in double the memory reads when compared to those GPU techniques that have hardware accelerated linear interpolation.

6.3.5 Summary

We have presented three different methods for CWBP, summarized in Table 6.4. All techniques will be analyzed in terms of runtime and the effects of numerical precision on the resulting estimated-NPS, as originally described in Section 4.1.1.

6.3.6 Results

The ability to test several implementations of CWBP, under various HPC environments, provides data for a clear and thorough investigation. Specifically, analysis was performed on 1152 projections, 736 channels, and a 1024×1024 reconstruction, as it represents the future of denser sensor arrays and more computationally demanding integrations. Figure 6.11 shows the reconstructed image.

We will investigate one qualitative analysis where all algorithms are compared against a *base-line* double-precision implementation on the CPU, where L2-error is calculated when summing the NPS-values. *The estimated-NPS is analyzed to show each implementation is similar and validates the fact each implementation is identical besides numerical precision. We also show the effects of numerical precision on the reconstruction and can explicitly state the use of single-precision floating-point only moderately effects noise in the reconstruction. Last, the runtime of all techniques is compared using the* base-line *implementation as a baseline and graphed according to the speedup obtained.*

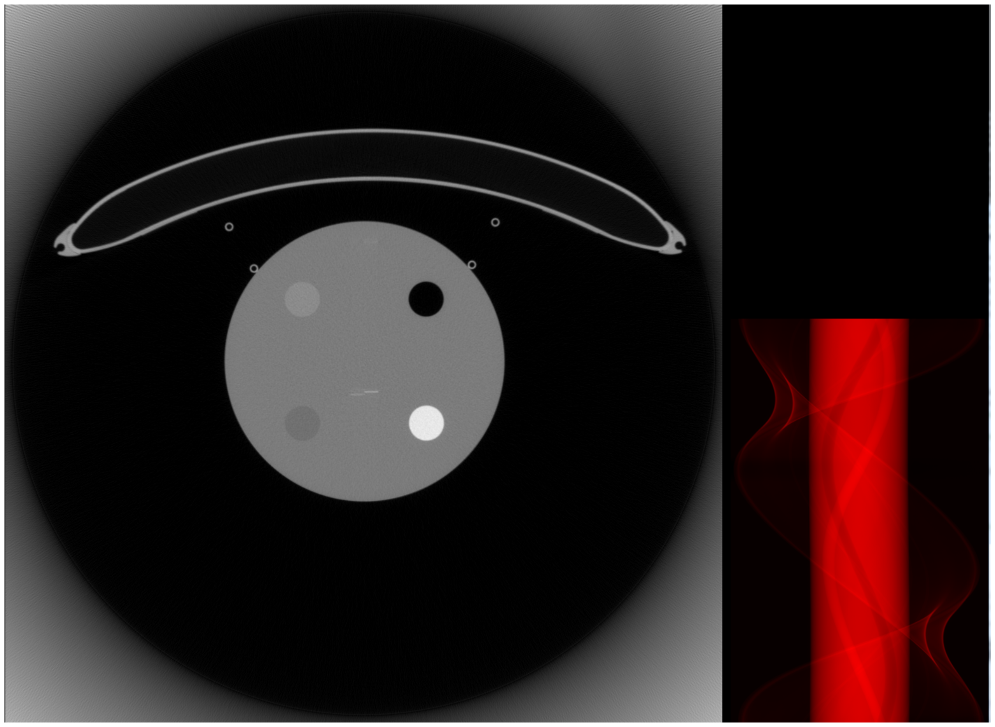


Figure 6.11: Reconstruction of a 1024×1024 image (grey) via the FBP algorithm. The red inset image is the input texture of the algorithm (containing the actual measurement data), with 736 individual detectors (texture width) as well as 1152 projections (texture height).

Conventional HPC – Shared-Memory

The typical modality for HPC computation is a multi-threaded application running on a shared-memory environment. We tested an efficient CWBP implementation from a single serial execution to 128 threads on two shared-memory systems and obtained significant speedups. Specifically, on the SGI system with 32 dual-core Intel Itanium 9150M processors, an almost linear speedup of 64x was achieved and, on the 12-core MacPro5.1 system a 12.64x speedup was seen in Figure 6.12, versus the standard serial execution of the algorithm on each machine. This resulted in 7.27s and 9.09s reconstruction runtimes respectively, as seen in Table 6.5. We believe greater than linear speedup is attributed to the operating system (Mac OS X version 10.7.3) and use of the Hyper-Threading ability of the Intel Xeon X5650 processor, as greater than linear speedup should not be

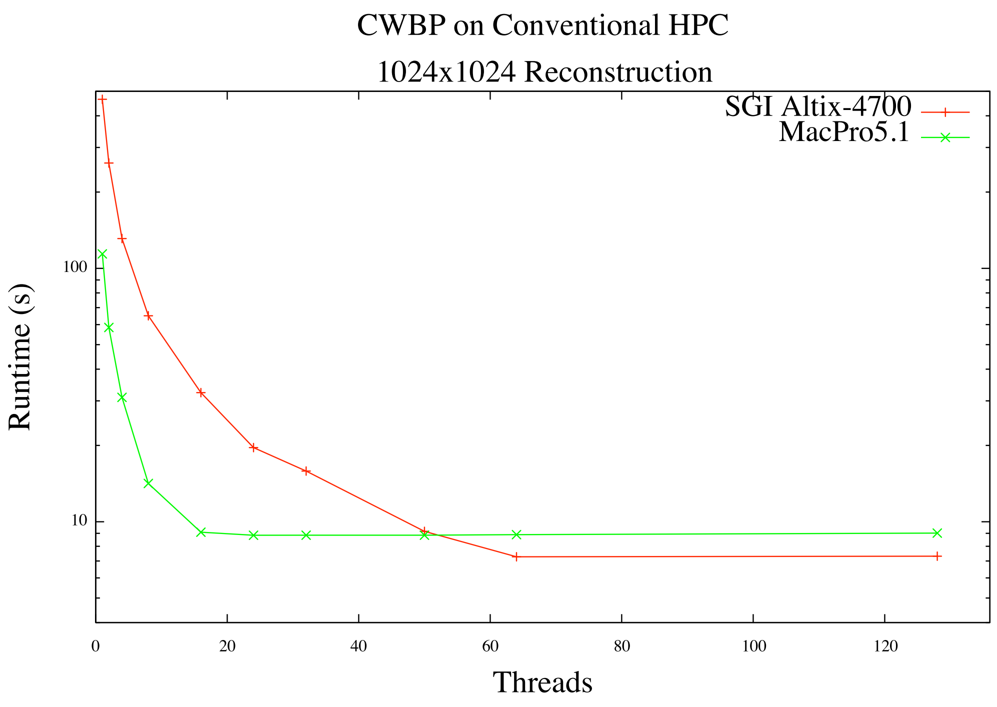


Figure 6.12: Conventional HPC is used to reconstruct a single 1024×1024 image using CWBP, an obvious performance boost is seen when using more threads. Once the system reaches full utilization the runtime stabilizes with no decrease in reconstruction time. The error-bars are not shown as the standard deviation is less than two for all methods.

possible.

Zeng *et al.* serial implementation for a 512×512 reconstruction using 1160 projections, and 672 radial detector channels, resulted in a runtime of 52s when using a Intel Xeon 3.2GHz processor [113]. We were only able to test a similar problem size, but do so to validate we have a reasonably efficient *base-line* implementation. Our technique achieved approximately a 2x speedup when compared to Zeng *et al.* when using MacPro5.1 with Intel Xeon X5650 2.66GHz processors. The CWBP parallel techniques from Zeng *et.al* only tested up to two cores, and no scalability results past two cores were ever published [113].

In terms of double-precision, the conventional HPC environment is not severely impacted and obtains nearly the same runtime results as single-precision, as seen when reviewing Table 6.5. We do show the technique does scale well past two cores, for both double and single-precision floating-point implementations, and conventional HPC will decrease the runtime substantially, representing a good medium to reduce runtime. *We also concluded that given the nature of the integration process in CWBP, by simply applying addition operations, that the use of double-precision did not enhance the quality of reconstruction and estimated-NPS was near identical to the standard single-precision floating-point implementation.*

system	CPU 1-core SGI	CPU 64-core SGI	CPU 12-core MacPro	CUDA 1-GPU MacPro	CUDA 2-GPU MacPro	CUDA 1-GPU MacPro fastmath	CUDA 2-GPU MacPro fastmath	DirectX 1-GPU MacPro
single-precision (32-bit) floating-point								
time [s]	464.74	7.27	9.09	1.08	0.65	0.56	0.32	0.35
speedup	1×	64×	51×	430×	715×	830x	1452x	1332×
double-precision (64-bit) floating-point								
time [s]	460.66	7.35	9.16	1.82	1.06	–	–	–
speedup	1×	63×	50×	253×	435×	–	–	–

Table 6.5: Runtime comparisons of our FBP reconstruction implementations.

Table 6.6: Estimated-NPS for ROIs

CWBP Reconstruction Techniques, results NPS units$^2 10^{-5}$						
Acceleration Technique	CPU Float	CPU Double	CUDA Float	CUDA Double	CUDA Float fastmath	DirectX Float
ROI 1	8.8	8.8	8.7	8.8	8.7	9.5
ROI 2	3.9	4.0	4.1	3.9	4.1	4.7
ROI 3	5.9	5.9	5.9	5.9	5.9	7.1
ROI 4	8.6	8.6	8.6	8.6	8.6	9.3
ROI 5	6.9	6.9	6.9	6.9	6.9	8.8
ROI 6	7.8	7.8	7.8	7.8	7.8	6.9
ROI 7	7.3	7.3	7.1	7.3	7.1	7.5
ROI 8	4.2	4.2	4.1	4.2	4.1	3.5
ROI 9	4.2	4.2	4.0	4.2	4.0	5.2

Non-Conventional HPC – GPU

We have shown three GPU implementations that all outperformed the conventional HPC environment, in terms of runtime as seen in Table 6.5. Specifically, the DirectX implementation was able to consistently obtain an approximate speedup 1332x, the CUDA implementation was able to achieve an approximate speedup of 430x for one GPU and 715x for two GPUs, under single-precision floating-point precision and cached textures for storage of all projection data. The use of CUDA *fast-math* resulted in a significant speedup in runtime, specifically 830x and 1452x for single and dual-GPU implementations respectively. Double-precision cached textures are not supported under CUDA and, for this reason, we opted for conventional global-memory on the GPU which allows for double-precision storage. The double-precision implementations were notably slower, only obtaining a 253x speedup for one-GPU and 435x speedup for two-GPUs, as seen in Table 6.5. The impact of double-precision on runtime is substantial, but not as large as expected because the GPUs used for testing are Nvidia Fermi class, which are optimized for double-precision computation.

Error Analysis

As all techniques are based on the same CWBP, found in [77, 24], and linear interpolation, we predict that the NPS is similar, with only small differences based on GPU based single-precision floating-point IEEE-754 standards or truncation.

We opted to use the estimated-NPS to validate the standardized algorithm used in all implementations and the results show the techniques are similar. The estimated-NPS also showed that the use of double-precision does not substantially change the estimated-NPS or L2-error. Therefore, one can conclude the use of single-precision floating-point to be more than adequate when using CWBP with linear-interpolation and a standard ramp-filter. Table 6.5 and 6.6 show the use of intrinsic functions on the GPU, as in the *fastmath*-mode, results in significant runtime savings and no difference in estimated-NPS. Therefore, under CUDA, the use of *fastmath*-mode does not increase the noise in the reconstruction. However, if more computationally demanding filters are used, that are in themselves sensitive to double-precision or *fastmath*, then one must use double-precision or disable *fastmath*. A careful numerical and error analysis of the filter and implementation would determine stability with single-precision floating-point and the use of *fastmath*-modes.

Table 6.6 show our DirectX implementation does suffer from a greater error than the alternate implementations. The estimated-NPS is greater in all ROIs when comparing to the CPU and CUDA implementations. We believe this is caused by *fastmath*-mode that DirectX employs to speed up computation [59]. These hardware accelerated operations are faster then conventional operators but are less accurate. Although the increase in error is constrained to the fifth decimal point we believe this should be noted when considering the modality for reconstruction and the requirements for display purposes. For instance, if a standard PAC display of 8-bit gray-scale is used, this numerical difference could not be viewable on the display because of the limited gray-scale. Although we obtained identical results using CUDA, with no intrinsics and *fastmath*-mode, we can not guarantee DirectX employs the same functions. Future research will look at understanding why the DirectX-method resulted in more error.

6.4 Summary

In the previous sections we analyzed various multi-GPU reconstruction algorithm, we have shown:

1. in Section 6.1, a detailed multi-GPU algorithm for SART that resulted in 1.54x speedup for RD-BP using Alg2:volume-partitioning when compared to the single-GPU technique. We also showed how ray-ordering can significantly effect runtime and detailed how different back-projection operations can effect the estimated-NPS in a reconstruction.

2. in Section 6.2, the ability to reduce runtime substantially using a multi-GPU algorithm for the derivation of the complete system matrix in ART. The technique resulted in a 19x speedup when compared to our serial-CPU implementation and 21x faster than Toft's implementation.

3. in Section 6.3, the analysis of three different CWBP techniques in terms of runtime, numerical-precision, and estimated-NPS. The analysis concluded that single-precision floating-point is adequate for the methods and the DirectX-method provided the fastest runtime at the expense of slightly higher noise in the reconstruction.

Chapter 7

Conclusion

The volumetric representation of internal human anatomy was revolutionary to patient care, as specialists could more easily diagnose ailments. The modality of choice for this document was fan-beam medical CT, because of its widespread utility and great number of challenges. Currently, the greatest concern regarding medical CT is the use of ionizing radiation which has been linked to the development of cancer. Through the use of computationally demanding algorithms, we hope to eventually reduce radiation exposure, thereby reducing the potential cancer causing effects of CT.

Reconstruction in diagnostic imaging represents a difficult computational problem for even current HPC modalities. The complexity of present clinical fan-beam CT requires special attention to scanner specifics and knowledge of the unit used. *A complete document has been compiled on the derivation of the mathematical framework behind the Radon transform and various reconstruction techniques based on the forward and backward-projection processes in fan-beam layout that is typically not available for researchers when using raw data.* We presented a number of challenges and solutions to these problems under fan-beam geometry that are lacking in the existing literature.

Using raw data presented a number of challenges for reconstruction, mainly the alignment, noise, FFS modes, and qualitative and quantitative analysis of results. *We provided implementation details concerning these difficulties that should aid any researcher wishing to use medical-CT in a research setting. We showed that alternative methods to the maximally-orthogonal technique of projection ordering, such as our hybrid-technique, can result in faster convergence, seen in Section 4.1.3. We showed that a typical projection ordering technique like maxOrthogonal would require close to 500 more ART-iterations to achieve similar results to the hybrid30 technique after only 100 ART-iterations.* The new projection ordering techniques, in AREMI, can save a considerable amount of computation time and provide quicker results under our implementation.

Through the use of AREMI and HPC, in terms of shared-memory parallel implementations on the CPU and highly optimized GPU implementations, we have produced a methods that can provide speedy reconstructions under traditional CWBP and the computationally expensive iterative techniques like ART or SART. *We have supplied clear implementation details for multi-GPU iterative algorithms based on our implementation of Jacob's et al. algorithm when using GPUs and real-world data with FFS modes enabled.*

7.0.1 Analysis of a Multi-GPU Environment for SART

We have illustrated the use of a multi-core and multi-GPU environment for CT-reconstruction, namely AREMI. *We show through the use of iterative ART algorithms, designed for GPUs, a computationally difficult problem is made manageable.* We recognized a 1.48x and 1.53x speedup using different workload policies using a RD-BP algorithm that scales over two GPUs versus our RD-BP algorithm on one GPU. We also presented a VD-BP technique with different workload policies that does not require a mutually-exclusive write operation. This method recognized a 1.29x and 1.33x speedup over a single GPU VD-BP algorithm. Last, we showed that well-ordered data is important to branching in our version of Siddon's algorithm when we attempted to further speedup SART. This approximation reduced the likelihood of threads writing to the same location in memory. However, the technique results in almost double the runtime compared to the conventional technique that required a mutually-exclusive write operations.

The use of multiple GPUs is complicated in nature as it requires careful algorithm design that is aware of hardware, but has been shown to result in significant speedups to the forward and back-projection process of the SART algorithm. We have shown that although the technique is not linear in speedup with the addition of GPUs, the techniques still reduce runtime and play an important role in the facilitation of SART in clinical settings, as fast reconstructions are required. As the technique is only available for shared-memory systems, and a maximally available eight-GPU environment is currently supported by Nvidia under shared-memory, the techniques presented should perform well past two GPUs as the data shared among GPUs is constant. The ray-partitioning algorithm presented, that distributes projection values over GPUs, would not scale well when there is not enough work to completely utilize all CUDA-cores: this setting is only true under axial scans, as in helical scans a larger workload would be available.

7.0.2 Complete System Matrix Derivation using a Multi-GPU Environment

The ability to use complete derivations of the system matrix in a timely fashion could eventually lead to algorithms that out-perform traditional techniques, as these methods scale to multiple GPUs easily because the blocking structure is relatively simple, as shown in Figure 6.7. Likewise, this blocking structure for the dot-product reduction was found to be GPU-type dependent and we reverted to a learning scheme that tested small problem sizes to gain knowledge of the optimal blocking structure.

The complete system matrix derivation is computationally expensive as the technique relies on a $O(n \times m)$ pixel interactions. *We presented a method, based on using the GPU in a multi or single GPU environment, that can significantly speed up runtime in* AREMI. *The technique is based on an optimal blocking strategy that is learned which results in, approximately, a 20% speedup versus a naive implementation.* Considering runtime lasts for several minutes, and possibly hours, for 20 *ART*-iterations, these results are important.

We compared our complete matrix derivation technique to Toft's original source code and the new method based on the GPU resulted in a 21x speedup when reconstruction a 512×512 *reconstruction.* Although we were unable to modify Toft's source code to use our complex scanner geometry and raw-attenuation values, we believe the reconstruction time for raw-data versus simulated data to be similar. We hope to extend the implementation to more than two GPUs, and test more computationally expensive interpolation techniques, formally comparing them with our ray-driven forward and back-projection techniques and also the ray-driven forward-projection and voxel based back-projection technique in a qualitative and quantitative analysis. As the computational power of the GPU is increasing, we believe, in the near-future, the complete matrix derivation or on-the-fly ray intersection techniques may become viable as the basic operations are less costly than ray-tracing.

7.0.3 Analysis of HPC for CWBP

There are several methods of HPC available now, and the choice of which environment is most suitable is often difficult. There are cost effective methods, such as the current GPU which performs well when compared to typical HPC, if an implementation can be constructed that does not require frequent memory transfers from the GPU to the CPU. *We have shown that all CWBP implementations are nearly equivalent and only minor differences, based on numerical precision or hardware accelerated operations, are found, as illustrated by the estimated-NPS and L2-error metrics.* The ability to compare various implementations shows the current GPU is powerful and the DirectX pipeline is efficient, producing results all most as fast as the dual-GPU implementations under CUDA. Therefore, if possible, construction under this modality can be beneficial. However, the use of the graphics pipeline is more complex and experience with DirectX and OpenGL is essential. Likewise, the complexity of iterative-type filtering can be incredibly difficult to factor for the graphics pipeline, even for experienced graphics programmers. We also showed that the *fastmath*-mode in CUDA that employs intrinsic functions, which are faster but less accurate, results in substantial runtime savings with no effect on estimated-NPS.

We have shown the double-precision based implementations are superior, but are slower on the GPU, though generally do not decrease the noise introduced during reconstruction substantially. Therefore, one must decide if the qualitative impact is realized by experienced radiologists, and justifies the increased runtime. For future work, we will conduct user studies, as they will provide the answer to these questions as the estimated-NPS and L2-error metrics show subtle differences, but realistically these small improvements would be difficult to see on current display hardware. The DirectX pipeline does have the ability to use double-precision, but only on select cards. Obviously this will change in the future with the natural progression of technology. *We have shown, although conventional HPC is suitable for many problem domains, the high cost associated with this modality and less than impressive runtime performance for CWBP show that the modality of choice is that of the GPU in a multi-GPU environment, as it is the cheapest, simplest, and performs the best in terms of runtime and precision.*

7.0.4 Summary

We have investigated many issues with the reconstruction of raw-attenuation values for fan-beam CT. There are many specifics that are thoroughly explored in the development of AREMI. *We have shown various implementations of CWBP using raw fan-beam data from medical-CT in order to evaluate those techniques in terms of runtime, noise analysis, and the effects of numerical precision on the result.* The methods presented are beneficial to the research community as they establish a framework for reconstruction under the noted modality.

In this thesis we have explored many different reconstruction algorithms and investigated some the problems involved with reconstruction under specific methods. The techniques outlined have shown to be valid in taking a problem that is intractable and making it manageable.

7.1 Future Work

The ability to control the number of projections in a shutter-like modality would be interesting, as it would prevent a constant X-ray exposure to patients and represent the discrete nature of the detector sampling more closely. The development of this hardware is interesting and complicated. As noted by Wang *et al.* on the analysis of noise at various mAs values, from 100 down to 17 mAs, the conclusion showed that, through highly attenuated regions, the noise spikes on lower mAs value scans [104]. For this reason, decreasing the dose in terms of mAs value has dire effects on the quality of reconstruction. However, carrying out less noise prone mAs levels and a shutter like system, or possible limited rotation angle, would greatly reduce the exposure to X-ray radiation and; therefore, further investigation into this hardware would be beneficial.

Techniques such as CWBP were shown to require a greater number of projections, but iterative techniques, such as ART, have been shown to require far less. For these reasons, further analysis into the reduction of noise, and speedier reconstructions for the computationally-intense iterative methods, shows promise. Using limited projection angles results in the system matrix $\mathbf{A}$ being even more underdetermined. Therefore, more reconstruction time is needed to obtain a good solution. We believe through testing and further optimizations that AREMI, in computational environments that have greater than two GPUs, would significantly decrease runtime.

We have provided solutions to many of the difficulties found in using raw attenuation data to reconstruct images. Therefore, we have the ability to investigate the parameter space and attempt to optimize the solutions and reconstructions. For this reason we believe investigation in the following to be very rewarding. First, convergence is often subjective and research into compressive sensing and the sparse layout is interesting, as convergence to an exact solution is possible. Second, using AREMI and compressive sensing technique presents a number of challenges and research using HPC methods would likely be rewarding, as the HPC environment supplied by Servier, SGI and Nvidia are powerful and rare. Last, an investigation into the use of computationally-expensive iterative filters in the goal of better image quality and timely reconstructions for clinical-CT. These filters actually work on the sub-iterative level and some are, in fact, iterative in their nature and converge upon an exact solution. These iterative filters are complex and computationally-expensive and; therefore, are a perfect fit for AREMI and the hardware resources available.

We also hope to further validate the use of a 14-bit monochrome 10MP 30inch PAC display. The display offers excellent pixel density, as it is only 30inch with 10 MPixel and an supports HDR. Coupled with the raw reconstruction environment AREMI and the ability to extend the output range to 14-bits, we hope to quantify the use of the display and extended dynamic range available in AREMI in a clinical setting through numerous user studies.

We have found comparison against other implementations very difficult when using raw fan-beam data and believe the development of a database of raw-attenuation values for fan-beam CT would be beneficial to the research community. Likewise, we would like to establish specific qualitative and quantitative metrics to enable benchmarking of similar techniques in the aims to eventually use these reduced radiation techniques in clinical-CT.

Bibliography

[1] A. H. Andersen and A. C. Kak. Simultaneous algebraic reconstruction technique (sart): a superior implementation of the art algorithm. *Ultrasonic Imaging*, 6(1):81–94, 1984.

[2] A. Berrington de Gonzalez, M. Mahesh, K. Kim, R. Bhargavan, M.and Lewis, F. Mettler, and C. Land. Projected cancer risks from computed tomographic scans performed in the united states in 2007. *Archives of Internal Medicine*, 169(22):2071–2077, 12 2009.

[3] D. Bharkhada, H. Yu, K. Zeng, E. Bai, and G. Wang. A comparative study on interpolation methods for controlled cardiac ct. *International Journal of Imaging Systems and Technology*, 17(2):91–98, 2007.

[4] R. N. Bracewell. *The Fourier transform and its applications / Ronald N. Bracewell*. McGraw-Hill, New York :, 1978.

[5] R.N. Bracewell and A.C. Riddle. Inversion of fan-beam scans in radio astronomy. *Astrophysical Journal*, 150:427–+, November 1967.

[6] D.J. Brenner, C.D. Elliston, E.J. Hall, and W.E. Berdon. Estimates of the cancer risks from pediatric ct radiation are not merely theoretical: comment on "point/counterpoint: in x-ray computed tomography, technique factors should be selected appropriate to patient size. against the proposition". *Medical physics*, 28(11), 11 2001.

[7] D.J. Brenner and E. J. Hall. Computed tomography —an increasing source of radiation exposure. *New England Journal of Medicine*, 357(22):2277–2284, 11 2007.

[8] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[9] E. Cardis, M. Vrijheid, M. Blettner, E. Gilbert, M. Hakama, C. Hill, G. Howe, J. Kaldor, C. R. Muirhead, M. Schubauer-Berigan, T. Yoshimura, F. Bermann, G. Cowper, J. Fix, C. Hacker, B. Heinmiller, M. Marshall, I. Thierry-Chef, D. Utterback, Y-O Ahn, E. Amoros, P. Ashmore, J-M Auvinen, A .and Bae, J. B. Solano, A. Biau, E. Combalot, P. Deboodt, A. D. Sacristan, M. Eklof, H. Engels, G. Engholm, G. Gulis, R. Habib, K. Holan, H. Hyvonen, A. Kerekes, J. Kurtinaitis, H. Malker, M. Martuzzi, A. Mastauskas, A. Monnet, M. Moser, M. S. Pearce, D. B. Richardson, F. Rodriguez-Artalejo, H. Rogel, A .and Tardy, M. Telle-Lamberton, I. Turai, M. Usel, and K. Veress. Risk of cancer after low doses of ionising radiation: retrospective cohort study in 15 countries. *BMJ Journal*, 331(7508), 07 2005.

[10] Y. Censor. Finite series-expansion reconstruction methods. *Proceedings of the IEEE*, 71(3):409–419, March 1983.

[11] J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19:297 – 301, 1965.

[12] S. Coric, E. Leeser, M.and Miller, and M. Trepanier. Parallel-beam backprojection: an fpga implementation optimized for medical imaging. In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, FPGA '02, pages 217–226, New York, NY, USA, 2002. ACM.

[13] NVIDIA Corp. Nvidia cuda compute unified device architecture - programming guide 4.0, 2011.

[14] J. Dean, S. Ghemawat, and Google Inc. Mapreduce: simplified data processing on large clusters. In *In OSDI 2004: Proceedings of the 6th conference on Symposium on Operating Systems Design and Implementation*. USENIX Association, 2004.

[15] P. Després, F. Lacroix, and J. Carrier. Fast drr and cbct reconstruction on gpu. In *Medical Physics*, volume 35, page 2915. American Association of Physicists in Medicine, 2008.

[16] C. A. Diaz Leon, S. Eliuk, and H. T. Gomez. Simulating soft tissues using a gpu approach of the mass-spring model. *IEEE Virtual Reality*, 2009. Poster Publication.

[17] S. Eliuk and Mendl C. Comprehensive analysis of hpc methods for filtered back-projection. Submitted to the Journal of Medical Imaging and Physics, 2011. http://christian.mendl.net/software/radon_gpu_manuscript.pdf.

[18] L. D. Enochson and R. K. Otnes. Programming and analysis for digital time series data. *U.S. Deptartment of Defense, Shock and Vibration info*, page 142, 1968.

[19] X. Fang and K. Mueller. A comparative study of popular interpolation and integration methods for use in computed tomography. In *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*, pages 1252 –1255, April 2006.

[20] L. A. Feldkamp, L. C. Davis, and J. W. Kress. Practical cone-beam algorithm. *J. Opt. Soc. Am. A*, 1(6):612–619, 06 1984.

[21] T.G. Flohr, K. Stierstorfer, S. Ulzheimer, H. Bruder, A.N. Primak, and C.H. McCollough. Image reconstruction and image quality evaluation for a 64-slice ct scanner with z-flying focal spot. *Medical Physics*, 32(8):2536–2537, August 2005.

[22] M Frigo and SG Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 02 2005.

[23] S. Ghemawat and S. Gobioff, H.and Leung. The google file system. *SIGOPS Oper. Syst. Rev.*, 37:29–43, October 2003.

[24] R. C. Gonzalez and R. E. Woords. *Digital Image Processing*. Pearson Prentice Hal, Upper Saddle River, New Jersey 07458, iii edition, 2010.

[25] R. Gordon. A tutorial on art(algebraic reconstruction techniques). *IEEE Transactions on Nuclear Science*, NS-21:78–93, 1974.

[26] R. Gordon. Stop breast cancer now! imagining imaging pathways toward search, destroy, cure, and watchful waiting of premetastasis breast cancer. In Tibor Tot, editor, *Breast Cancer*, pages 167–203. Springer London, 2011.

[27] R. Gordon, R. Bender, and G. Herman. Algebraic reconstruction techniques(art) for three dimensional electron microscopy and x-ray photography. *Journal of Theoritical Biology*, 36:105–117, 1970.

[28] K. Grant and T.Flohr. Iterative reconstruction in image space (iris). White Paper - Siemens Technical Report, 2010.

[29] H. Guan and R. Gordon. A projection access order for speedy convergence of art (algebraic reconstruction technique): a multilevel scheme for computed tomography. *Physics in Medicine and Biology*, 39(11):2005, 1994.

[30] H. Guan and R. Gordon. Computed tomography using algebraic reconstruction techniques (art's) with different projection access schemes: A comparison study under practical situations. *Physics Medicine Biology.*, 41:1727–1743, 1996.

[31] H. Guan, R. Gordon, and Y. Zhu. Combining various projection access schemes with the algebraic reconstruction technique for low-contrast detection in computed tomography. *Phys Med Biol*, 43(8):2413–2421, August 1998.

[32] M. Harris. Rendering technqiues. http://www.markmark.net/misc/rendertexture.html.

[33] M. Harris. Efficient reduction on the gpu, 2011. http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/reduction/doc/reduction.pdfnb07-niwaiQZ8jZxtsQ.

[34] G. Herman and A. Lent. Iterative reconstruction algorithms,. *Computing Biology Medicine*, 6:273–294, 1976.

[35] G. Herman and A. Naparstek. Fast image reconstruction based on a radon inversion formula approppriate for rapidly collected data. *SIAM Journal of Applied Mathematics*, 33:511–533, 1976.

[36] G. T. Herman and H. Lung. Reconstruction from divergent beams: A comparison of algorithms with and without rebinning. *Computers in Biology and Medicine*, 10(3):131–139, 1980.

[37] G.T. Herman. The fundamentals of computerized tomography. *Academic Press*, 1980.

[38] G.T. Herman and L.B. Meyer. Algebraic reconstruction techniques can be made computationally efficient. *IEEE Transactions on Medical Imaging*, 12(3):600–609, September 1993.

[39] H.P. Hiriyannaiah. X-ray computed tomography for medical imaging. *Signal Processing Magazine, IEEE*, 14(2):42 –59, mar 1997.

[40] J. E. Hopcroft and J. D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1990.

[41] F. Jacobs, E. Sundermann, B. D. Sutter, M. Christiaens, and I. Lemahieu. A fast algorithm to calculate the exact radiological path through a pixel or voxel space. *CIT. Journal of computing and information technology*, 6(1):89–94, 1998.

[42] B. Jang, D. Kaeli, S. Do, and H. Pien. Multi gpu implementation of iterative tomographic reconstruction algorithms. In *Proceedings of the Sixth IEEE international conference on Symposium on Biomedical Imaging: From Nano to Macro*, ISBI'09, pages 185–188, Piscataway, NJ, USA, 2009. IEEE Press.

[43] M. Jiang and G. Wan. Convergence of the simultaneous algebraic reconstruction technique (sart). *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 12(8), August 2003.

[44] P. M. Joseph. An improved algorithm for reprojecting rays through pixel. *IEEE Transactions on Medical Imaging*, 1:193–196, 1982.

[45] M. Kachelrieß, M. Knaup, C. Penßel, and W. A. Kalender. Flying focal spot (ffs) in cone-beam ct. *IEEE TRANSACTIONS ON NUCLEAR SCIENCE*, 53(3):1238–1247, june 2006.

[46] S Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bulletin International de l'Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques*, 35(355-357), 1937.

[47] A. C. Kak and M. Slaney. *Principles of computerized tomographic imaging*. Society for Industrial and Applied Mathematics, philidelphia, 2001.

[48] B. Keck. Medical Image Reconstruction using Graphics Hardware (CUDA). Technical Talk, 2009.

[49] B. Keck, H. Hofmann, H. Scherl, M. Kowarschik, and J. Hornegger. GPU-accelerated SART reconstruction using the CUDA programming environment. In Ehsan Samei and Jiang Hsieh, editors, *Proceedings of SPIE*, volume 7258, Lake Buena Vista, 2009.

[50] B. Keck, H. Hofmann, H. Scherl, M. Kowarschik, and J. Hornegger. High Resolution Iterative CT Reconstruction using Graphics Hardware. In Bo Yu, editor, *2009 IEEE Nuclear Science Symposium Conference Record*, pages 4035–4040, N/A, 2009.

[51] M. F. Kijewski and P. F. Judy. The noise power spectrum of ct images. *Physics in Medicine and Biology*, 32(5):565–575, 1987.

[52] A.V. Lakshminarayanan. Reconstruction from divergent ray data. Technical report, State University of New York at Buffalo, 1975.

[53] B. Lee, H. Lee, and YG. Shin. Fast hybrid cpu and gpu based ct reconstruction algorithm using air skipping technique. *Journal of X-Ray Science Technology*, 18(3):221–234, 01 2010.

[54] Otha W. Linton and Fred A. Mettler. National conference on dose reduction in ct, with an emphasis on pediatric patients. *American Journal of Roentgenology*, 181(2):321–329, 08 2003.

[55] C.L. Lueng-Hendriks, M. Van Ginkel, P.W. Verbeek, and L.J. Van Vliet. The generalized Radon transform: sampling, accuracy and memory considerations. *Pattern Recognition*, 38:2494 – 2505, 2005.

[56] C. McCollough, M. Bruesewitz, M. McNitt-Gray, K. Bush, T. Ruckdeschel, J. Payne, J. Brink, and R. Zeman. The phantom portion of the american college of radiology (acr) computed tomography (ct) accreditation program: Practical tips, artifact examples, and pitfalls to avoid. *Journal Medical Physics*, 31(9), 2004.

[57] C. Melvin, P. Thulasiraman, and R. Gordon. Parallel algebraic reconstruction technique for computed tomography. *The 2003 International Conference on Parallel and Distributed Processing Techniques and Applications*, 2003.

[58] C. Melvin, M. Xu, and P. Thulasiraman. Hpc for iterative image reconstruction in ct. In *Proceedings of the 2008 C3S2E conference*, C3S2E '08, pages 61–68, New York, NY, USA, 2008. ACM.

[59] Microsoft. Windows direct3d compiler reference, 2012. http://msdn.microsoft.com/en-us/library/windows/desktop/gg615083%28v=vs.85%29.aspx.

[60] G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 38(8), April 1965.

[61] K. Mueller. Ultra-fast 3d filtered backprojection on commodity graphics hardware. *IEEE International Symposium on Biomedical Imaging: Macro to Nano (IEEE Cat No. 04EX821)*, 2:571–574, 2004.

[62] K. Mueller, F. Xu, and N. Neophytou. Why do commodity graphics hardware boards (gpus) work so well for acceleration of computed tomography? *SPIE Electronic Imaging*, page 64980N, January 2007.

[63] K. Mueller, R. Yagel, and J. J. Wheller. Accurate low-contrast 3d cone-beam reconstruction with algebraic methods. *IEEE Transactions on Medical Imaging,*, September 1999.

[64] K. Mueller, R. Yagel, and J. J. Wheller. Anti-aliased three-dimensional cone-beam reconstruction of low-contrast objects with algebraic methods. *IEEE Transactions on Medical Imaging*, 18:519–537, 1999.

[65] K. Mueller, R. Yagel, and J. J. Wheller. Fast implementations of algebraic methods for three-dimensional reconstruction from cone-beam data. *IEEE Transactions on Medical Imaging*, 18:538–548, 1999.

[66] N. Neophytou, F. Xu, and K. Mueller. Hardware acceleration vs. algorithmic acceleration: Can gpu-based processing beat complexity optimization for ct? *SPIE Medical Imaging*, 6510:65105F, February 2007.

[67] Paul M. Novotny, Jeff A. Stoll, Nikolay V. Vasilyev, Pedro J. del Nido, Pierre E. Dupont, Todd E. Zickler, and Robert D. Howe. GPU based real-time instrument tracking with three-dimensional ultrasound. *Medical Image Analysis*, 11(5):458 – 464, 2007.

[68] NVIDIA. CuFFT Library, August 2010.

[69] H. Nyquist. Reprint: Certain topics in telegraph transmission theory. *Proceedings of the IEEE*, 90(2):280–305, 2002.

[70] Y. Okitsu, F. Ino, and K. Hagihara. High-performance cone beam reconstruction using cuda compatible gpus. *Parallel Computing*, 36:129–141, February 2010.

[71] X. Pan, E. Y. Sidky, and M. Vannier. Why do commercial ct scanners still employ traditional, filtered back-projection for image reconstruction? *Inverse Problems*, 25(12):123009, 2009.

[72] Y. Pan and R. Whitaker. Iterative ct reconstruction integrating sart and conjugate gradient. *Proc. SPIE*, 7961, 2011.

[73] Y. Pan, R. Whitaker, A. Cheryauka, and D. Ferguson. Regularized 3d iterative reconstruction on mobile c-arm ct. In *In Proceedings of The First CT Meeting*, Salt Lake City, UT, 2010.

[74] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa. The design and implementation of a first-generation cell processor, 2005.

[75] D. L. Preston, D. A. Pierce, Y. Shimizu, H. M. Cullings, S. Fujita, S. Funamoto, and K. Kodama. Effect of recent changes in atomic bomb survivor dosimetry on cancer mortality risk estimates. *Radiation Research*, 162(4):377–389, 2011/04/07 2004.

[76] D.L. Preston, E. Ron, S. Tokuoka, S. Funamoto, N. Nishi, M. Soda, K. Mabuchi, and K. Kodama. Solid cancer incidence in atomic bomb survivors: 1958-1998. *Radiation research*, 168(1), 07 2007.

[77] J. L. Prince and J. M. Link. *Medical Imaging Signals and Systems*. Pearson Prentice Hal, Upper Saddle River, New Jersey 07458, 2007.

[78] J. Radiol. Studies of mortality of atomic bomb survivors. report 13: solid cancer and non-cancer disease mortality: 1950-1997. *Journal of Radiological Protection*, 23(4), 2003.

[79] J. Radon. Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten. *Sächsische Akademie der Wissenschaften, Leipzig*, 69:262 – 277, 1917.

[80] J. Radon and P.C. Parks. On the Determination of Functions from Their Integral Values along Certain Manifolds. *IEEE Transactions on Medical Imaging*, 5:170 – 176, 1986.

[81] G. N. Ramachandran and A. V. Lakshminarayanan. Three-dimensional reconstruction from radiographs and electron micrographs: Application of convolutions instead of fourier transforms. *Proceedings of the National Academy of Sciences*, 68(9):2236–2240, 1971.

[82] P. Raman, R. D. Kriz, A. L. Abbott, and C. J. Ribbens. Parallel implementation of the filtered back projection algorithm for tomographic imaging. Online Reference Last Verified Sept 28, 2011.

[83] H. Scherl, S. Hoppe, M. Kowarschik, and J. Hornegger. Design and implementation of the software architecture for a 3-d reconstruction system in medical imaging. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 661–668, New York, NY, USA, 2008. ACM.

[84] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger. Fast gpu-based ct reconstruction using the common unified device architecture (cuda). In *Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE*, volume 6, pages 4464 –4466, 26 2007-nov. 3 2007.

[85] H. Scherl, M. Koerner, H. Hofmann, W. Eckert, M. Kowarschik, and J. Hornegger. Implementation of the fdk algorithm for cone-beam ct on the cell broadband engine architecture. *Proceedings of SPIE Medical Imaging 2007*, 6510, Feb 2007.

[86] Hsi-Yu Schive, Chia-Hung Chien, Shing-Kwong Wong, Yu-Chih Tsai, and Tzihong Chiueh. Graphic-card cluster for astrophysics (gracca) performance tests. Jan 2008.

[87] Hsi-Yu Schive, Yu-Chih Tsai, and Tzihong Chiueh. Gamer: a gpu-accelerated adaptive mesh refinement code for astrophysics. Jul 2009.

[88] M. Schlindwein. Iterative three-dimensional reconstruction from twincone beam projections. *IEEE Trans. Nucl. Sci.*, 25:1135–1143, May 1978.

[89] T. Schroeder. Nvidia cuda unified virtual address space – peer-to-peer and unified virtual addressing. Web Seminar, 2011.

[90] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5:3–55, January 2001.

[91] L.A. Shepp and B.F. Logan. The fourier reconstruction of a head section. *IEEE Trans Nuclear Sciences*, 21(21-43), 1974.

[92] J. Shtok, M. Elad, and M. Zibulevsky. Adaptive filtered-back-projection for computed tomography. In *IEEE 25-Th Convention Of Electrical And Electronics Engineers In Israel*, Eilat, Israel, December 2008.

[93] C. Shuai, L. Jie, J.W. Sheaffer, K. Skadron, and J. Lach. Accelerating compute-intensive applications with gpus and fpgas. In *Application Specific Processors, 2008. SASP 2008. Symposium on*, pages 101 –107, june 2008.

[94] R. L. Siddon. Fast calculation of the exact radiological path for a three-dimensional ct array. *Medical Physics*, 12(2):252–255, 1985.

[95] G. Stantchev, W. Dorland, and N. Gumerov. Fast parallel particle-to-grid interpolation for plasma pic simulations on the gpu. *Journal Parallel Distributed Computing*, 68(10):1339–1349, 2008.

[96] D. B. Thomas, L. Howes, and W. Luk. A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation. In *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '09, pages 63–72, New York, NY, USA, 2009. ACM.

[97] P. Toft. *The Radon Transform Theory and Implementation*. PhD thesis, Department of Mathematical Modelling Section for Digital Processing Technical University of Denmark, 1996.

[98] Rice Univesity. Compressive sensing, October 2011. http://dsp.rice.edu/cs.

[99] National Research Council (U.S.). *Health risks from exposure to low levels of ionizing radiation*. 2006.

[100] V. V. VLČEK. Computation of inverse radon transform on graphics cards. *INTERNATIONAL JOURNAL OF SIGNAL PROCESSING*, 1(1):1–12, 2004.

[101] V. V. VLČEK. Computation of filtered back projection on graphics cards. *WSEAS Transactions of Computers*, 4(9):1216–1221, September 2005.

[102] V. V. VLČEK. *1D and 2D Digital Data Filtering by the Inverse Radon Transform*. PhD thesis, Department of Mathematics Pilsen University, 2007.

[103] V. V. VLČEK. Comparison of reconstruction methods for computerized tomography. In *The 1st Young Researchers Conference on Applied Sciences Pilsen*, ISBN 978-80-7043-5748, pages 86–91. ZCU, 2007.

[104] J. Wang, H. Lu, Z. Liang, D. Eremina, G. Zhang, S. Wang, J. Chen, and J. Manzione. An experimental study on the noise properties of x-ray ct sinogram data in radon space. *Physics Medicine Biology.*, 53(12):3327–3341, June 21 2008.

[105] Y. Wang, T. Feng, L. Shen, and Y. Xing. Research on ati-cal for accelerating fbp reconstruction. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, pages 4126 –4129, 24 2009-nov. 1 2009.

[106] A. Weinlich, B. Keck, H. Scherl, M. Kowarschik, and J. Hornegger. Comparison of High-Speed Ray Casting on GPU using CUDA and OpenGL. In Rainer Buchty and Jan-Philipp Weiß, editors, *Proceedings of the First International Workshop on New Frontiers in High-performance and Hardware-aware Computing (HipHaC'08)*, volume 1, pages 25–30, Karlsruh, 2008.

[107] F. Xu and K. Müller. Real-Time 3D Computed Tomographic Reconstruction Using Commodity Graphics Hardware. *Physics in Medicine and Biology*, 52:3405 – 3419, 2007.

[108] Fang Xu. Fast implementation of iterative reconstruction with exact ray-driven projector on gpus. *Tsinghua Science and Technology*, 15(1):30 –35, feb. 2010.

[109] W. Xu and K. Mueller. A performance-driven study of regularization methods for gpu-accelerated iterative ct. Workshop on High Performance Image Reconstruction, 2009.

[110] W. Xu and K. Mueller. Evaluating popular non-linear image processing filters for their use in regularized iterative c. *Conference Record IEEE Medical Imaging Conference*, 2010.

[111] H. Yang, M. Li, K. Koizumi, and H. Kudo. Accelerating backprojections via cuda architecture. In *9th Int. Meeting on Fully 3D Imaging Reconstructiom Radiation and Nuclear Medicine.*, pages 52–55, 2007.

[112] G. L. Zeng and G. T. Gullberg. Unmatched projector/backprojector pairs in an iterative reconstruction algorithm. *IEEE Transactions on Medical Imaging*, 19:548–555, 2000.

[113] K. Zeng, E. Bai, and G. Wang. A fast ct reconstruction scheme for a general multi-core pc. *Journal of Biomedical Imaging*, 2007:1–1, January 2007.

Appendix A

Appendix A: GPU Computing Related-Work

Recent advances in high performance computing have provided super-computing environments for the masses. These recent advances were facilitated by the computer gaming industry and the vast demand for realism in video games. The Graphics Processing Unit (GPU) is a parallel super computer packed nicely into a commodity hardware device, which is often less expensive than the computer in which it resides. The peak single-precision floating-point operations of a Nvidia GTX 580 GPU is approaching 1.5-Teraflops as seen in Equation A.1. Whereas, the peak single-precision floating-point computational power of the latest intel Westmere 5600 6-core processor is approximately 150 GFlops, as depicted in Figure A.2. Figure A.1 shows that the single-precision floating-point computational power of a single GPU is far superior than a CPU. The GPU computational power is increasing at a rate equal to the cube of Moore's law [60].

The following sections will review basic hardware on the GPU, the Compute Unified Device Architecture (CUDA) programming and computational models, and General Purpose Graphics Processing Unit (GPGPU) based computation. This primer on GPUs is useful as some of the implemented techniques require knowledge of the GPU and the basic techniques behind programming the GPU.

A.1 Insight Into the Current GPU

As general purpose computing on the GPU became more popular, many researchers quickly realized some of the apparent problems. These included the complexity of programming the graphics pipeline and the inability to directly access shared-memory on the GPU for reading and writing and many more features unavailable for direct programmable access. Second, many disciplines of science wanted an easy method to use the GPU to accelerate their respective simulations/applications. Researchers that were not familiar with computer graphics programming found using the graphics pipeline required an in-depth knowledge of the field and wanted an easier method to access the highly parallel, multi-threaded, and high memory bandwidth available on the GPU. Nvidia Corp. created the Compute Unified Device Architecture (CUDA) to facilitate the use of the GPU. CUDA was well received in the research community and provided access to some of the features on the GPU previously not directly accessible through the graphics pipeline.The following sections will give a short review of CUDA-based programming. However, the CUDA-Guide [13] from Nvidia Corp. reviews the techniques in their entirety.

Hardware-specific components, and most importantly the understanding of those specifics components, leads to efficient algorithm generation. In the following section, we will review the computational units and the various memory types available on the current GPU, under the CUDA framework.

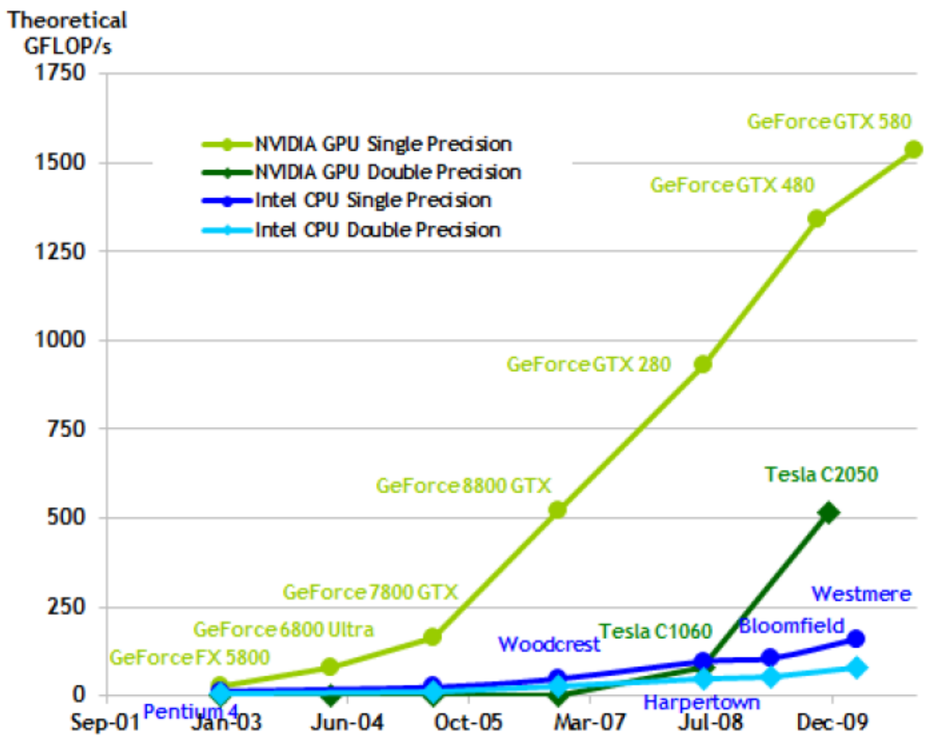


Figure A.1: Peak Single-Precision Floating-point Operations GPU vs CPU [13].

$$MaxFlop/sec = Flop/sec * Processorcores * processorclock \tag{A.1}$$

A.2 Computational Units of the current GPU

The twenty-first century GPU is undoubtedly impressive, providing a quantum leap in computational power. Specifically, the GPU is encompassed by Streaming Multiprocessors (SMs), seen in Figure A.3. Each SM commonly contains 32 CUDA cores running at a lower frequency than

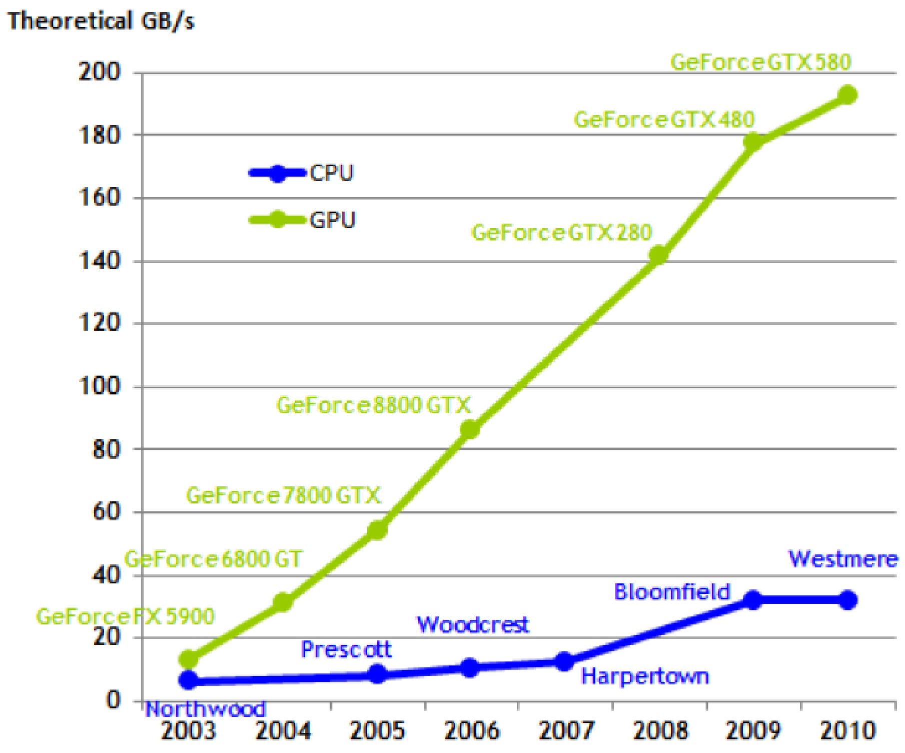


Figure A.2: Memory Bandwidth GPU vs CPU [13].

Table A.1: Example: Nvidia GTX 580

Gflops Calculation for Nvidia GTX 580	
Processor Cores	512
Processor Clock Rate	1544MHz
Flop/sec	2 flops max per cycle
Max Flop/sec	$2 \times 512 \times 1544 \approx 1.5$ TFlop/sec

CPUs, generally less-than 2000 MHz. The cores run at lower frequencies to insure reliability, less energy consumption and, in turn, less heat is generated. There are various models of GPUs from Nvidia where the number of SMs differ in terms of quantity and frequency for the SP cores contained. The Nvidia GTX 285 contains 30 SMs, resulting in 240 SP, cores each running 1.476GHz and providing $\approx$ one-TFlop of single-precision floating-point performance, as calculated in Equation A.1. Where as the Nvidia Quadro 4000 Fermie based GPU has 8 SMs, and 32 CUDA cores running at 0.95Ghz and provides approximately 0.5TFlops and 0.25TFlops of single-precision and double-precision floating-point performance respectively.

SMs are responsible for mapping threads to an individual SP core, with one thread running on each core of the SM. The design of the SM gives them the ability to run different programs on different SMs. The SM SIMT (single-instruction, multiple thread) creates, manages, schedules, and executes kernels in groups of 32 threads, called *warps* [13].

A.3 Computational Units of the current GPU-*Warp*

A *warp* is a grouping of threads from a kernel. Specifically, 32 threads makeup a *warp*. These threads start with the same execution path, but are free to diverge through conditional statements. The threads are then designated to one of the eight-cores on the SM. An instruction is loaded and each thread in a *warp* is deemed *active* or *inactive*, depending if their execution is active given the loaded instruction. Those deemed *active* have computation based on the loaded instruction, while those threads loaded but *inactive* have no computation performed, because their respective execution path is different than the currently loaded path. Therefore, conditional statements are understood to have a significant performance decrease because there are *inactive* threads where cycles are simply wasted. In general, a *warp* executes one common instruction across all threads, so full efficiency (no *inactive* threads) is seen when all 32 threads follow the same execution path (no divergent execution paths). In the case where threads diverge through conditional statements, which would be conditionally based on some loaded data, the *warp* serially executes each branch of the path, disabling threads that are not on the currently executing path. When all the execution paths are complete, the threads naturally converge to the same execution path; however, because there is no guarantee of ordering, *serialization* is used to ensure a common place for threads to wait until all threads are ready to continue, ensuring a common execution path and realization of maximum performance because of a fully active *warp*. In general, *serialization* is used to ensure convergence and execution of the same instruction set (see Example A.1). Although the code, described previously, is based on CUDA, the language is simply used to illustrate the serialization needed to verify all shared memory data is loaded.

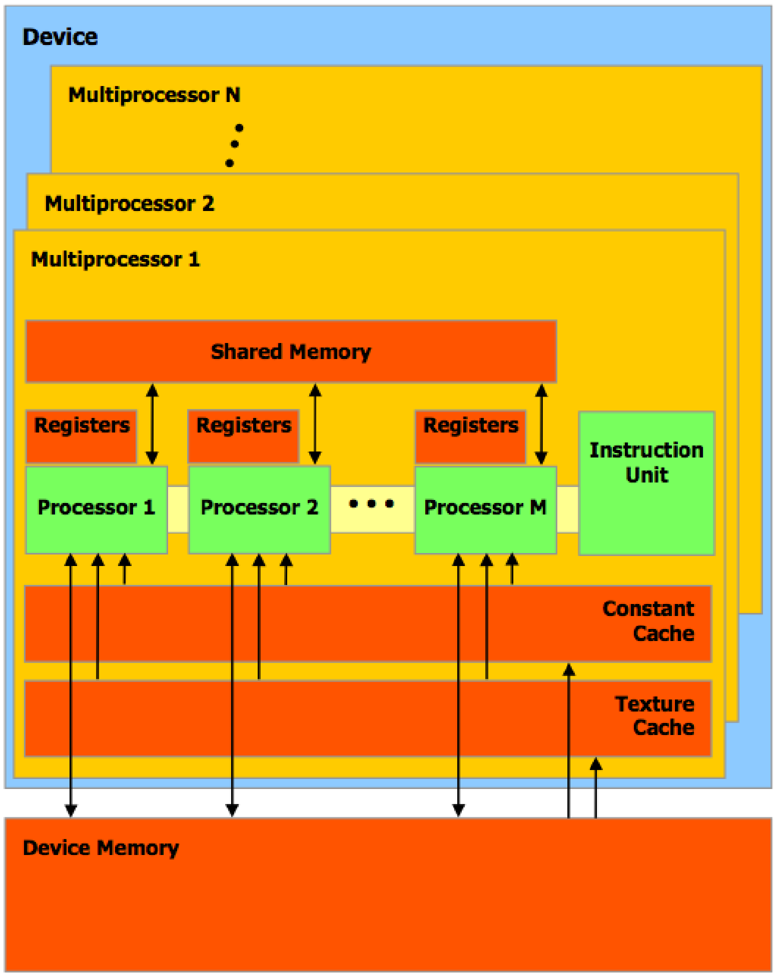


Figure A.3: SIMT multiprocessors, various memory available. [13].

Listing A.1: CUDA Serialize Example [13]

```
    // each thread is responsible for loading
    // two values from global memory, global
    // memory A and B.

    // Declaration of the shared memory array.
    __shared__ float A_shared[BLOCK_SIZE];

    // Declaration of the shared memory array.
    __shared__ float B_shared[BLOCK_SIZE];

    // Load the matrices from device memory
    // to shared memory; each thread loads
    // one element of each matrix.
    int index = threadIdx.x;
    A_shared(index) = A[index];
    B_shared(index) = B[index];

    // Synchronize is needed to make sure
    // all threads have loaded the data into
    // shared memory.
     __syncthreads();

    // carry out some computation on the
    // data loaded into shared memory.
    // We are assured all values have been
    // loaded into shared memory because
    // of the synchronization call.
    ...

    }
```

A.4 Computational Units of the current GPU-Thread-blocks

Although 32 threads make up a *warp*, and are executed on a SM, the actual grouping of warps is specific to thread-blocks. A thread-block is a collection of threads from a kernel. The size of the thread-block can be controlled in CUDA by specifying the dimensionality needed [one-dimentional, two-dimentional, or three-dimentional], making blocks easy to understand in terms of a arrays or matrices. In total, 512 threads designate the maximum number of threads allowable in a thread-block. This constraint requires programmers to develop specialized code to breakup large fields (3D-case) into small subfields, in order to fit into the 512 threads constraint.

A.5 Memory on the current GPU

The contemporary GPU typically had two different types of memory available to programmers. We will separate the memory into two sections, SM memory and globally accessible memory and a final section on the SM and the caches available. However, most recently Nvidia developed the Unified

Virtual Addressing (UVA) for Fermie GPUs with compute capability 2.0. The UVA provided an easy means to share memory between GPUs and access specific locations of memory from host process or GPU kernel through fast DMA transfers.

A.6 SM Memory and Accessibility Constraints

SMs have three types of memory available to them: registers, local memory and shared memory. Registers are a small amount of space used to store an instruction set; they are fast to access, but limited in size. The second local memory type is available to store locally created variables. The third, and most important memory type, is the 16 KB of shared-memory available or 48 KB on Fermie GPUs. This memory is important because different threads can share information between each other, as long as they are contained within the same processing block. Remember that a block, and the threads which encompass that block, are always contained on the same SM. The shared memory is fast (4 clock cycles) compared to the, already quick, global memory (400-600 clock cycles) [95], which will be discussed in the next section. This gives threads an efficient way to share data between each other, and possibly load data from global memory to shared memory, if the given thread block will be accessing a subset of the global memory frequently. By loading the data from global to shared memory, assuming there is frequent access from a limited subset of global memory, memory access and thread wait for memory fetches will be greatly reduced. All of these memory types are readable and writable.

A.7 GPU Memory - Global Memory

Transferring data from main memory to the GPU is a time consuming processes and should be limited because of the slow PCI-bus that the data must be transmitted through. As one can see in Figure A.2, the G200 GPU has a memory bandwidth of approximately 150 GB/s vs. the Nehalen CPU of approximately 30 GB/s. This shows that we have memory access times of at least three times greater when accessing main memory vs. GPU based global memory. There is another constraint that must be considered, namely that the maximum bandwidth of the current PCI-Express (PCI-E) bus cannot exceed 16 GB/s, in the best case. PCI-E is currently limited to 32x and approximately 8 GB/s and 16 GB/s bidirectionally. This shows, that, although one has a major speedup in the memory bandwidth of the current intel i7 CPU with close to 30 GB/s, three times faster than previous intel CPUs, one still has a limiting factor of the PCI-E bus when one needs to transfer data to, or from, the GPU. When considering memory access times and the constraints of the PCI-E bus, the G200 GPU has 10x faster access times when using the GPU's global memory. Global memory is readable and writable from threads on the GPU and facilitates sharing of data between thread blocks, but is much slower than GPU-based shared memory. Also global memory is not cached and programming parallel GPU programs should take this into account.

A.8 GPU Memory - Texture and Constant Memory

One can see in Figure A.2 that, besides global memory, there is also constant and texture memory. Texture-based memory is specifically based on texturing, and facilitates fast reading of coalesced data. Likewise, constant-based memory is quick and can be used. However, both texture and constant memory is read-only, which is why the data can be accessed so quickly. Furthermore texture and constant memory are cached on the GPU, as seen in Figure A.3. For example, say a given thread requests data from texture memory from a specific position. First the texture cache is checked to see if the data is already cached. If not, the data is loaded and extra data is transferred. The likelihood of another thread on the same SM requesting this data or data close to the original loaded data is high; therefore, caching of texture and constant memory can provide a substantial increase in performance and justifies loading the extra data into respective caches.

A.9 Caches Available on the GPU

Texture and constant caches are available and this justifies the use of texture and constant memory when designing efficient parallel GPU programs. The caches provide a substantial speedup because reading from SM caches, is must faster than reading from texture or constant memory. Also, global memory is not cached, and if writing data is not required, it is better to use texture or constant memory because of the speedup provided by the fast on-chip cache available to the SMs.

A.10 Unified Virtual Address Space (UVA)

The UVA space on the current GPU enables simpler programming by developers because one address space is used for all CPU and GPU memory. The physical location of the memory is determined by a pointer value and removes the need to specify the location of shared resources. Many of the most frequent memory copy functions, cudaMemcpy{*HostToHost, HostToDevice, DeviceToHost, DeviceToDevice*} have been replaced with cudaMemcpy*Default*. Using the UVA mode much higher PCI bandwidth can be realized [89].

A.11 GPU Discussion

The current day GPU has a high computational rate when considering peak single-precision floating-point operations. There are many considerations when programming a current day GPU, such as the use of different GPU-memory, memory caches, sharing of data between threads in the same block, and the sharing of memory between threads of different blocks. Therefore, one must take note of the considerations and limitations of the various memories in order to implement efficient parallel programs on the GPU.

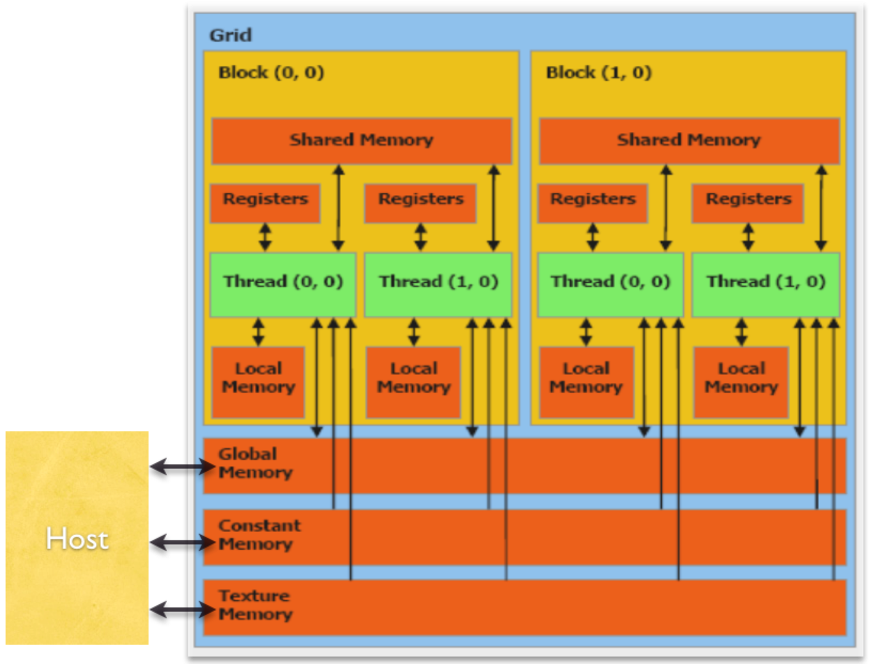


Figure A.4: Memory current GPU, modified diagram from [13].

A.12 General Purpose Computing on the Graphics Processing Unit (GPGPU)

GPU based computing is generally based on two forms: GPGPU and CUDA development. GPGPU organizes the problem into a special computational model where problems are solved by the graphics pipeline, as illustrated in Figure A.5. The most widely used GPGPU based method would be the Ping-Pong technique for off-screen rendering [32] and is reviewed in the following section. The graphics pipeline approach have been shown to perform well as seen in Table 6.5 where a several reconstruction techniques were compared in medical imaging and the GPGPU method outperformed a multi-GPU CUDA based implementation.

A.13 GPGPU OpenGL Frame Buffer Object and RW Textures

As the most basic example, we will use one Frame Buffer Object (FBO) for off-screen rendering with two textures attached to it. One of the textures is used for reading previously stored data and the other texture is used for writing/updating of results. The initialization routine for an FBO is read and write textures can be seen in Listing A.2 and A.3. The source code shows the process of creating an off-screen rendering target, but the specifics should not be overlooked. The Ping-Pong Technique's name comes from the fact that one can swap reading and writing textures after every rendering pass, like a Ping-Pong ball bounces back and forth. This technique reads values from the readable-texture, computes an update/computation based upon those values, and writes the output to

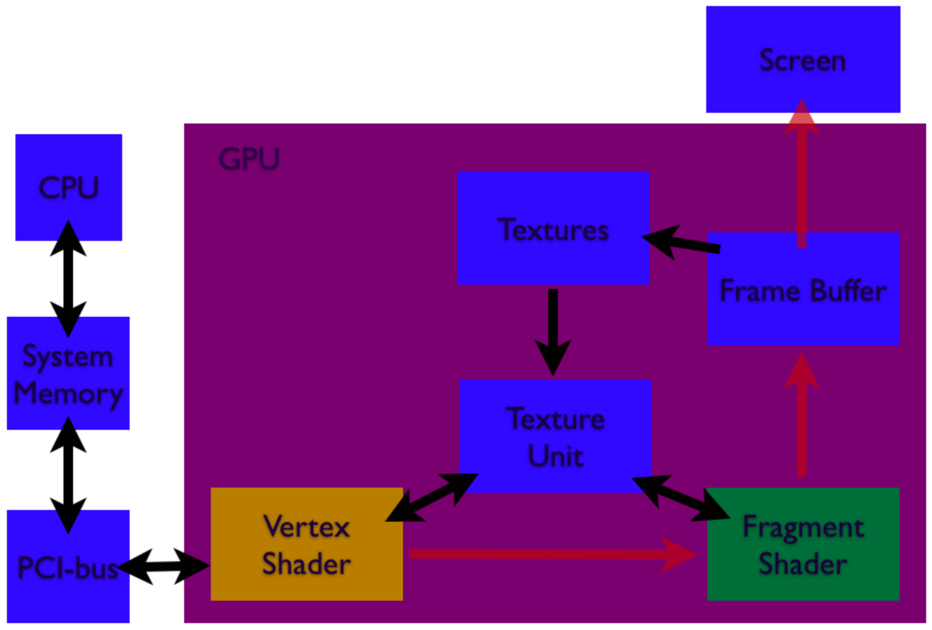


Figure A.5: Simplified graphics pipeline.

an off-screen buffer; in our example, a texture. However, instead of writing the output to the screen, as in on-screen rendering, the output is rendered off-screen to the write-texture.

Listing A.2: Read and Write Texture Initialization

```
void initReadWriteTextures(){
  GLuint writeTex, readTex;
  glGenTextures( 1, writeTex );

  //need two textures for reading and writing
  //create writeTex for initial writing, this is
  //the texture we will write to during offscreen
  //rendering
  glBindTexture(GL_TEXTURE_RECTANGLE_ARB, writeTex);
  glTexParameteri(
    GL_TEXTURE_RECTANGLE_ARB,
    GL_TEXTURE_MIN_FILTER,
    GL_NEAREST);

  glTexParameteri(
    GL_TEXTURE_RECTANGLE_ARB,
    GL_TEXTURE_MAG_FILTER,
    GL_NEAREST);

  //define the texture as rectangle_arb,
  //values not clamped between [0,1]
  //_width and _height should be the
  //size of your required texture.
  glTexImage2D(
    GL_TEXTURE_RECTANGLE_ARB,
    0, GL_RGBA32F_ARB, _width, _height, 0,
    GL_RGBA, GL_FLOAT,  0);

  glGenTextures( 1, readTex );
  //create readTex for initial reading
  glBindTexture(GL_TEXTURE_RECTANGLE_ARB, readTex);
  glTexParameteri(
    GL_TEXTURE_RECTANGLE_ARB,
    GL_TEXTURE_MIN_FILTER,
    GL_NEAREST);

  glTexParameteri(
    GL_TEXTURE_RECTANGLE_ARB,
    GL_TEXTURE_MAG_FILTER,
    GL_NEAREST);

  //define the texture as rectangle_arb,
  //values not clamped between [0,1]
  //_width and _height should be the
  //size of your required texture.
  glTexImage2D(
    GL_TEXTURE_RECTANGLE_ARB,
    0, GL_RGBA32F_ARB, _width, _height, 0,
    GL_RGBA, GL_FLOAT,  0);

}
```

Listing A.3: Frame Buffer Object Initialization

```
void initFBO(){
  GLuint _fbPosition;

  //generates one frame buffer object
  glGenFramebuffersEXT(1, &_fbPosition);

  //bind the frame for offscreen rendering
  glBindFramebufferEXT(
    GL_FRAMEBUFFER_EXT,
    _fbPosition);

  const GLenum _attachmentpoints[] =
  { GL_COLOR_ATTACHMENT0_EXT,
    GL_COLOR_ATTACHMENT1_EXT};


  glFramebufferTexture2DEXT(
    GL_FRAMEBUFFER_EXT,
    _attachmentpoints[writeTex],
     GL_TEXTURE_RECTANGLE_ARB,
      _textureHolder[writeTex], 0);

  glFramebufferTexture2DEXT(
    GL_FRAMEBUFFER_EXT,
    _attachmentpoints[readTex],
    GL_TEXTURE_RECTANGLE_ARB,
    _textureHolder[readTex], 0);

  //reset to default frame buffer for onscreen rendering
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
}
```

Listing A.4: Activating Framebuffer for Rendering.

```
void activateFBO(){
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, _fbPosition);

  glFramebufferTexture2DEXT(
  GL_FRAMEBUFFER_EXT,
  GL_COLOR_ATTACHMENT0_EXT,
  GL_TEXTURE_RECTANGLE_ARB,
  writeTexture, 0);

  glFramebufferTexture2DEXT(
  GL_FRAMEBUFFER_EXT,
  GL_COLOR_ATTACHMENT1_EXT,
  GL_TEXTURE_RECTANGLE_ARB,
  readTexture, 0);

  glDrawBuffers(1, writeTexture);
}
```

Listing A.5: Ortho2D Mapping for Assured, Unclipped Rendering.

```
void adjustViewPort(){
  //grab current view port settings,
  //will need to restore after ortho run
  //for correct viewport
  GLint viewport[4];
  glGetIntegerv(GL_VIEWPORT, viewport);
  glMatrixMode(GL_PROJECTION);
  glPushMatrix();{

    glLoadIdentity();
    glOrtho(0, _width, 0, _height,-1,1);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    //_width and _height is equal to width
    //and height values of textures used.
    glViewport(0,0,_width,_height);

    //enable rectangle textures
    glEnable(GL_TEXTURE_RECTANGLE_ARB);
    //call shaders that will work on textures
    //enabling appropriate textures for
    //reading.
    ...
    //draw a quad to activate shaders
    //quad should fill screen width and height
    glBegin(GL_QUADS);{
        glTexCoord2f(0, 0);
        glVertex2f(0, 0);
        glTexCoord2f(_width,0);
        glVertex2f(_width, 0);
        glTexCoord2f(_width,_height);
        glVertex2f(_width, _height);
        glTexCoord2f(0, _height);
        glVertex2f(0, _height);
    }glEnd();

  }glPopMatrix();

  //restore previous viewport values
  glMatrixMode(GL_MODELVIEW);
  glViewport(
   viewport[0],
   viewport[1],
   viewport[2],
   viewport[3]);

  //restore rendering to onscreen buffer
  glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);

  //swap the reading and writing textures
  swap(readTex, writeTex);
  }
```

A.14 Adjusting Viewpoint for One-to-One Mapping

The viewport needs to be adjusted to ensure correct and complete computation. A call to *glOrtho(left, right, bottom, top, near, far)* is used in order to setup a two-dimensional orthographic viewing region. This is important because the viewport will not clip the drawn quad, seen in Listing A.5, and creates a correct rendering for a one-to-one mapping between the fragment shader and textures used.

A.15 GPGPU Example

For an example, take the development of a quick particle-engine based on this technique. The particle-engine would use two readable textures and two writable textures. Read-texture1 would contain information about the initial position of the particles and read-texture2 would contain the initial velocity information of the particles. For the initial time-step the readable textures would be used in the fragment shader as illustrated at Figure 2.5 by the green box, while the position and velocity read-textures would be used to update the position and velocity of the particles. The updated position and velocities would be written to the off-screen buffer, specifically the two writable textures. After every update the read and write textures would be swapped so that the next integration step, the pass through the shader, would contain the updated position and velocity from the previous time step in the readable texture. The process would continue, swapping readable and writable textures until the simulation/particle-engine is no longer needed.

A.16 GPGPU Final Thoughts

The Ping-Pong Technique uses the graphics pipeline and, specifically, the fragment shader to perform computation. The computation performed is generally some update to a simulation, or some incremental technique that depends on previous values. Difficulties arise, in GPGPU-based programming that uses the graphics pipeline, because shared memory is not available through shaders and direct writing is not available, as in CUDA. These limitations helped facilitate the development of CUDA and, in turn, helped with the evolution of AREMI.

A.17 Summary

GPU programming has provided exceptional speedups for many problem domains. There are a number of underlining justifications that dictate if a problem will have a large speedup on a GPU, or a moderate speedup. Specifically, those problems that are considered coarse-grained, large amount of computation between communication, will perform very well. Those coarse-grained problems that read memory in a coalesced fashion will perform even better. Those that randomly access memory will not perform as well. Problems that are considered fine-grained, small amount of computation between communication, can still benefit from the GPU; however, they will only see moderate speedups. By paying special attention to the use of shared memory, refer to Section A.6 , and the performance benefits of it, one can speedup computation [16, 86, 87].

The previous sections identify many areas of research that must be utilized in order to create an efficient reconstruction environment for CT. Through the use of specialized algorithms in the multi-GPU environment, coupled with the streamlined resources available on a shared memory computer, one can create a hardware modality that fashions an environment for real-time CT reconstruction, optimization, and enhancement. The research benefits all scientists wishing to use larger HPC hardware, coupled with multiple GPUs.

Appendix B

Appendix B: Hardware and Software Environment

B.1 Hardware and Software Environment Description

The hardware and software development needs to be accurately defined and a consistent environment is needed on all systems. Therefore, no specialized system calls were used in Mac OSX Lion, SUSE. The implementation of CWBP, in the DirectX shader pipeline, used specialized system calls in Windows 7x64 Professional.

B.2 Hardware Environment

The hardware environment consists of three systems: a Mac Pro with dual 6-core Intel Xeon X5650 Westmere 2.66GHz 12MB L3 Cache processors, 64GB of 1333Mhz ECC memory, and dual Nvidia Quadro 4000 Fermie GPUs with 2GB each of frame-buffer, a Dell 7500 dual 4-core Intel Xeon X5550 Nehalem 2.66GHz 4 x 256KB L2 Cache 8MB L3 Cache processors, 24GB of 1333Mhz ECC memory, and dual Nvidia Tesla c2070 Fermie GPUs with 6GB-ECC each of frame buffer memory, and lastly, an SGI Altix 4700 32$\times$dual-core Intel Itanium 9150M 1.66GHz 24MB L3 Cache processors, 256GB of 1333Mhz ECC shared-memory. The visualizations are displayed on two monitors: namely a 27” 3.5MP Apple Display with 7-bit grayscale, and a 30” 10MP WIDE Display with 14-bit grayscale.

B.3 Software Layout and Description

The software environment consists of either Apple OSX Lion-10.7-64bit, Windows 7 Professional-64-bit, or Suse Enterprise linux. All systems use the same C++, C, OpenGL, and CUDA-4.0 calls for cross-system comparison. No specialized or optimized compilers were used, just g++, gcc, and nvcc CUDA compiler.

B.4 Scanner Specifics

The CT-scanner used for all raw data purposes is a Siemens Definition Flash+ 64 or 128 slice scanner. The raw data file is parsed according to Siemens raw data format specifications, and appropriate attributes corresponding to FFS modes are specified for reconstruction.

B.5 Phantom Specifics

The phantom used for scanning is the Gammex 464 CT phantom, which is an American College of Radiology CT accredited phantom. The phantom has various homogenous regions and contains features of known geometric size and composition. For more information regarding ACR-accredited phantoms, and the complexity of their design, please see [56].