Design, Development and Implementation of a Parallel Algorithm for Computed Tomography Using Algebraic Reconstruction Technique

by

Cameron Melvin
Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, MB R3T 2N2, Canada

A thesis submitted in partial completion of requirements for
Master of Science in Computer Engineering

Abstract

This project implements a parallel algorithm for Computed Tomography based on the Algebraic Reconstruction Technique (ART) algorithm. This technique for reconstructing pictures from projections is useful for applications such as Computed Tomography (CT or CAT). The algorithm requires fewer views, and hence less radiation, to produce an image of comparable or better quality. However, the approach is not widely used because of its computationally intensive nature in comparison with rival technologies. A faster ART algorithm could reduce the amount of radiation needed for CT imaging by producing a better image with fewer projections.

A reconstruction from projections version of the ART algorithm for two dimensions was implemented in parallel using the Message Passing Interface (MPI) and OpenMP extensions for C. The message passing implementation did not result in faster reconstructions due to prohibitively long and variant communication latency. The shared memory implementation produced positive results, showing a clear computational advantage for multiple processors and measured efficiency ranging from 60-95%. Consistent with the literature, image quality proved to be significantly better compared to the industry standard Filtered Backprojection algorithm especially when reconstructing from fewer projection angles.

Acknowledgements

I would like to thank Dr. Parimala Thulasiriman for all of her support, technical and otherwise, as well as Dr. Bob McLeod for his constructive suggestions.. I'd also like to thank Debbie Rand and Bruno Chu at McKesson Medical Imaging group for their understanding and flexibility with my work contribution as I finished this project. Many thanks to friends and family for their encouraging words along the way.

Table of Contents

Table of Abbreviations

AIX	Advanced Interactive eXecutive
ART	Algebraic Reconstruction Technique
CAT	Computed Axial Tomography (AKA CT)
ccNUMA	Cache Coherent Non-Uniform Memory Access
CPU	Central Processing Unit
CT	Computed Tomography (AKA CAT)
DSM	Distributed Shared Memory
EM	Expectation Maximization
FBP	Filtered Backprojection (AKA Fourier Backprojection)
FIFO	First In First Out
GHz	Giga-Hertz
GTOMO	Computational Grid Parallel Tomography
HPC	High Powered Computing
I/O	Input/Output
IBM	International Business Machines
IEEE	Institute of Electrical and Electronics Engineers
MART	Multiplicative Algebraic Reconstruction Technique
MAS	Multi-level Access Scheme
MCM	Multiple Chip Module
MIMD	Multiple Instruction Multiple Data
ML	Maximum Likelihood
MPI	Message Passing Interface
MPICH	Message Passing Interface Chameleon
MSE	Mean Squared Error
NUMA	Non-Uniform Memory Access
OpenMP	Open Multi-Processing (shared memory language)
PART	Parallel Algebraic Reconstruction Technique
PBS	Portable Bach System
PVM	Parallel Virtual Machine
RAS	Random Access Scheme
SART	Simultaneous Algebraic Reconstruction Technique
SGI	Silicon Graphics Inc
SIMD	Single Instruction Single Data
SIRT	Simultaneous Iterative Reconstruction Technique
SMP	Symmetric Multi-Processor
SPARTAF	Streak Preventative Algebraic Reconstruction Technique with Adaptive Filtering
TD3	Torus (3-Dimensional)
TRIUMF	Tri-University Meson Facility
WestGrid	Western Canadian Research Grid

Table of Figures

Table of Tables

1 Introduction

1.1 Purpose

This paper examines possible parallelization schemes for the ART algorithm for Computed Tomography. As X-ray tube and detector technologies improve, Computed Tomography (CT) is increasingly used for diagnostic imaging small anatomical bodies. Cancerous lesions are of particular interest; however the carcinogenic nature of X-ray imaging deters the use of CT imaging in many cases, especially suspected cancers such as brain (head) and breast. The greatest advantage of the ART family of algorithms is the ability to produce better images with fewer projections, where each projection requires additional radiation dose.

The prohibitive aspect of the ART algorithm is the processing time required in comparison with Fourier Backprojection (FBP) techniques. In order to realize the radiation dose advantages, a faster form of the ART must be implemented. The purpose of this paper is to show that the proposed parallel ART algorithm exhibits an efficient speedup while maintaining the superior image quality of the ART algorithm. Timing results will be measured against a serial implementation as well as a FBP implementation. The Image quality of the parallel ART reconstructed images will be examined empirically and qualitatively in comparison with an FBP implementation

1.2 Scope

First, an introductory explanation of Computed Tomography will be given. Then the ART algorithm will be briefly developed, along with two of its variations MART and SPARTAF. An introduction to a competing algorithm family, the Fourier Backprojection, is also given.

The data dependencies present in the basic ART algorithm are assessed. An analysis of development options in pursuing parallel versions of this algorithm follows, leading to a conclusion regarding the best method for parallelization of the ART algorithm. 2-Dimensional, parallel beam geometry will be considered.

A brief description of the experimental platforms for message passing and shared memory implementations follows. A description of the shared memory implementation for two dimensional reconstruction is presented. Timing results of the message passing and shared memory implementations are presented. MSE (Mean Squared Error) and Entropy are used to quantitatively assess the image quality of the result images. In addition, result images are compared to a basic form of the Fourier Backprojection algorithm.

1.3 Prior Work

Published works in the field of Computed Tomography reconstruction fall into one of the following three categories: CT algorithm advancements, dedicated

reconstruction hardware, or parallel processing. The new work presented in this thesis is a new parallel processing technique. In the interests of completeness, previous works in all three areas will be discussed here.

Vector computer were used by Guerrini and Spaletta [12] for image reconstruction. The major limitation in that implementation was the speed and memory capacity of the hardware used.

3D reconstructions by Chen, Lee, and Cho [13] used convolution backprojection on an Intel iSPC/2 multiprocessor. Their incremental backprojection algorithm considers one ray at a time, as opposed to processing pixel-wise, and proved to be faster than the conventional backprojection algorithm. Convolution and backprojection functions were both parallelized using pipelining. Speedup ranged from 5 to 27, depending on problem size and number or processing elements.

Although the work does not appear in a peer reviewed publication, cone beam tomography in 2D was done in parallel by Rao, Kriz, Abbott, and Ribbens [14]. This work used both CM5 and Intel Paragon platforms using message passing

technology. Fourier operations were performed on the CM5 using Connection Machine Scientific Subroutine Library. This library allowed Rao et al. to easily map FFT operations to the CM5's massively parallel architecture. Up to 256 processors were required in experiments to achieve speedup on the CM5, while only 8 processors produced speedup on the Intel Paragon. Paradoxically, the message passing overhead begins dominate processing when increasing the number of processors in the CM5 implementation to the 200-500 processor range.

Based a an efficient 3D cone beam algorithm known as the Feldkamp algorithm [16], Reimann et al. [15] implemented this Fourier backprojection method on a shared memory architecture, as well as a message passing implementation on a cluster of workstations (COW). Using small scale machines, the authors discovered load balancing issues inherent in the algorithm and presented two methods to overcome. Their COW method increased utilization on the 6 machine cluster from 58.2% to 71.7%. The shared memory technique achieved a speedup of 1.92 with 2 processors, for a utilization of 95.1%.

Laurent et al. [17] examined three different algorithms for 3D cone beam reconstruction. The authors examined Feldkamp, block ART, and SIRT (Simultaneous Iterative Reconstruction Technique) analytically as well as empirically using five different MIMD (Multiple Instruction Multiple Data)

computers. These included a network of workstations (Sparc), a workstation network of 16 AXP processors, a 32-way Paragon (i860), a 128-way 3D Torus of AXP's (TD3), and a 32-way SP1 using RS6000 processors. Parallel programming used the Parallel Virtual Machine (PVM) library. The TD3 produced best speedup results.

Smallen et al. implemented 3D cone beam reconstruction using grid computing. The Computational Grid Parallel Tomography (GTOMO) method used both a network combining 7 workstations and 128-way SP2 supercomputers. The work explored scheduling strategies for queuing work for image reconstruction.

The work discussed thus far is primarily focused on Fourier techniques, with a lack of development of parallel approaches for ART techniques. Other iterative techniques such as Maximum Likelihood (ML), Expectation Maximization (EM), and Simultaneous ART (SART) [19], [20], [21]. EM work has been primarily focused on Positron Emission Tomography (PET) to compensate for high noise levels in that imaging modality. Transputer [22],[23], Butterfly network and Paragon[24], Beowulf cluster [25], and peer to peer approaches [21] have all been used to parallelize the EM algorithm.

Gordon et al. [38] first introduced the ART algorithm for use in image reconstruction in electron microscopy prior to the first commercial CT scanner.

Gordon [39], [41], [42], and others developed ART based and other iterative algorithms for CT. Continuing work by other researchers has yielded advances such as Rangayyan et al. [26][40], who improved streak formation for small numbers of views (angles). Deblurring methods for artifact reduction were introduced by Wang et al. [27],[28].

With iterative algorithms, faster convergence to a solution will reduce iterations and significantly speed processing time, resulting in a substantial body of work focused in this direction. Herman and Meyer [29] suggested non-uniform angles between projections, while Guan and Gordon [43] found that considering uniform angles in a different order improved convergence. Specifically, Guan and Gordon found that considering angles in an order such that consecutive angles are approximately orthogonal produced an optimal rate of convergence. The paper claimed that in addition to improved convergence rate over the ART method, their algorithm over all produced superior image quality when compared to Fourier backprojection, especially for limited views (few projections). Mueller et al. [44] advise a weighting scheme that according to their results produces better, reduced noise images.

A major advancement to ART came with SART. However, convergence for this technique was not theoretically established until recently by Jiang and Wang [30],[31].

Hardware focused work such as Lin and Jain in 1994 [1] and Shieh et al. [4] pursues a ground up hardware design. Coric et al. [5] presented an FPGA based parallel approach, discussing some of the major tradeoffs inherent in FPGA implementations. The most significant drawback is the use of fixed point arithmetic, leading to quantization errors not seen on CPU-based floating point implementations. In general, the hardware is normally algorithm dependant and any algorithm changes would require hardware changes.

Lattard et al. [6], [7] and Fitchett [3] used a hardware approach to reconstruction. Both projects used a SIMD array of processing elements, differing in the elements used. Lattard et al. proposed an array of cell processors, with a one-to-one mapping of processors to pixels. In addition to the inherent weakness of algorithm-specific hardware, the one-to-one mapping of processors to pixels imposes a further restriction to the data-set. The hardware is limited to a single image size. Although the architecture is theoretically scalable, advances in CT technology leading to larger reconstructions would still require either a larger scale processor array, or a different pixel-to-processor mapping. Lattard and Mazare suggest that many-to-one mappings are possible, but the effort required to change the pixel-to-processor mapping scheme is not clear.\

Lattard and Mazare present further possibilities in [7], admitting that, "The specificity of the processing part demands that we must redesign it for each application but leads to very fast computing and low complexity." Mueller and Yagel [32] proposed a more financially feasible solution by using PC graphics hardware and texture mapping to produce promising speedup for SART.

The approach presented in this thesis differs from other parallel iterative methods in that it is relatively hardware independent. One key advantage of the approach presented in this paper is its practicality and portability. Porting the solution to different platforms is relatively straight forward. While it is designed to take advantage of shared memory architectures, it can be run with equally valid results on a single processor machine, and can scale to various numbers of processors without re-compiling. CT manufacturers could potentially use not only off-the-shelf processors and peripherals, but whole systems off-the-shelf for the image reconstruction unit of a CT scanner. In addition, algorithm improvements could be implemented going forward with relative ease in comparison with the hardware changes required for improvements to any custom hardware configurations.

Specifically an 8-way IBM P-server (a scaled down version of the Cortex server used in experiments for this research) can be purchased off the shelf from IBM for cost in the range of $48,000 CDN. A scaled down 4 processor model is in the

range of $28,000 CDN. With newer 64-slice CT machines selling for prices in the order of $2,000,000 CDN, the incremental cost and simplicity of this solution make it feasible if dose and/or image quality can be improved.

While limited work on parallel iterative techniques was done by Laurent et al. [17] using a block-ART algorithm for 3-D cone beam geometries, the advantage of block-ART is only evident for noisy data sets as seen in Carvalho and Herman [32]. With single photon release x-ray emitters around the corner [34], performance in low noise situations will certainly be important in future work. In addition, the nature of the block-ART approach lends itself easily to the finer grained parallelism presented by Laurent et al. This finer grained parallelism presumably comes at a cost of more frequent communication in comparison with a courser grained parallelization of ART. Carvalho and Herman also showed that the ART family of algorithms shows a distinct advantage to Fourier Backprojection techniques in that the quality of ART reconstructions are not adversely affected by increased cone beam angle.

ART shows advantages over Fourier Backprojection for smaller numbers of views and a broader range of cone beam angles. ART shows advantage over similar algorithms such as block-ART for performance in low-noise data sets. While FPGA's are more flexible than ASIC-based approaches [5], platform independent software-based solutions offer the most adaptable approach for future advances.

With the fast paced progress and evolution of 3-D medical imaging technologies, flexibility is a key attribute for any lasting technology. Based on this body of literature, the potential advantages of a parallel ART algorithm are evident, and warrant exploration.

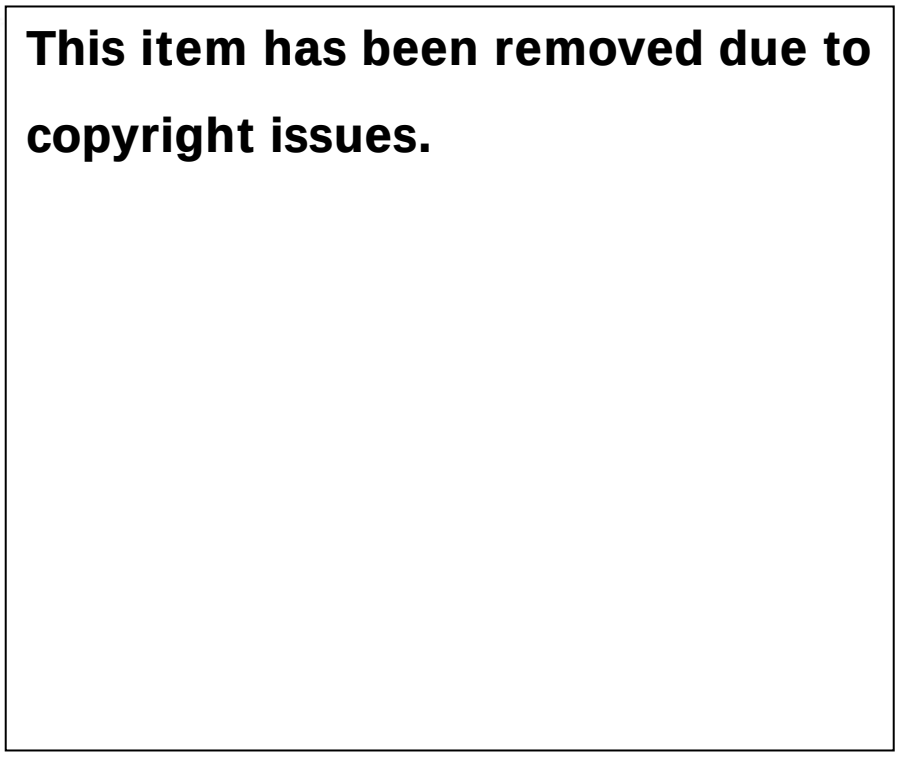
2 Introduction

2.1 Computed Tomography

Computed Tomography (CT) is an imaging modality which uses the technique of image reconstruction from projections. CT can be used to produce a 2D image from 1D projection data, although newer techniques produce 3D volumes from complex projection data sets. In order to simplify the discussion, reconstruction of 2D images will be considered. All results can be generalized to 3 dimensions, although a 3D problem would introduce another possibility for parallelization.

First, an X-ray projection of an object is taken, as shown below in Figure 1. The x-ray source emits radiation, and the detector collects the radiation that is not absorbed passing through the object. Computer Tomography estimates the absorption of radiation at each small section of the object, based on the total amount of radiation detected through each path. Each small section becomes a pixel in the image representation.

Figure 1: Schematic diagram of X-ray projection acquisition. In this simplified scheme, a row of X-ray sources (S_i) supply parallel beams passing through the object. Each beam is attenuated as it passes through the object, with the resultant attenuated beams measured by a row of detectors (D_i) [37].



Then several more projections are taken as the X-ray source and detector rotate around the object concentrically, with projections sampled at varying angles. In practice, the projection angles are usually equally spaced and taken in consecutive order, although this need not be the case (see Guan and Gordon [43]).

In practice, modern computer tomography machines collect data in one of two ways: concentric circles or spiral path.

Concentric circle data is collected by collecting data by rotating the X-ray tube and detector array around the subject in two dimensions, without any change in depth. Once a full set of angle data is collected, the depth is changed by one discrete step and another set of data is collected around the same axis but now at a different depth along the axis of rotation. In this case, data is collected in discrete sections – the projections for a single slice of the object are collected together in one pass. Two dimensional reconstruction algorithms such as the one presented in this paper are ideal for this type of data.

Spiral path data is collected by changing the depth of the X-Ray tube and detector array continuously as they rotate, forming a helical path around the object. Several commercial CT machines acquire data in this fashion but transform it into concentric circle pseudo-data for convenience in reconstruction and processing. Two dimensional reconstruction techniques then, are also applicable to this data collection method.

2.2 Algebraic Reconstruction Technique (ART)

One of the first reconstruction algorithms for CT was the Algebraic Reconstruction Technique (ART) developed by Gordon, et al. [1]. The ART

algorithm is the basis of many variations, developed to improve various flaws and non-idealities not addressed by the original ART.

The ART algorithm begins with some initial estimate of the image to be reconstructed (usually taken as a uniformly gray image). It modifies this estimate repeatedly until the pixel values appear to converge by some criterion. ART decides how to modify the image by summing the pixels along some straight path and comparing this sum to the measured ray sum (referred to earlier as an "X-ray projection"). The difference between projections calculated from the image estimate, and the measured ray is calculated, and the adjustment is divided among the pixels in the ray sum. Our new estimate for the picture, $P^{q+1}(i,j)$, is as follows:

$$P^{q+1}(i,j) = P^{q}(i,j) - \frac{\left(R(l,k) - S^{q}(l,k)\right)}{N(l,k)} \qquad \textbf{(1)}$$

Where:

$P^{q}(i,j)$ = the current estimate of the image (after the q^{th} iteration)
$P^{q+1}(i,j)$ = the updated ($q+1^{th}$ iteration) estimate of the image
$R(l,k)$ = choosing the projection at angle θ_l, selecting the ray k from that projection
$S^{q}(l,k)$ = the calculated ray sum at angle θ_l, selecting the ray k
$N(l,k)$ = the number of pixels in the ray (l,k)

A suitable criterion must be introduced for terminating this repetitive operation. One such criterion is based on the error in the calculated ray sums:

$$E_q = \frac{\sum_l \sum_k \left| R(l,k) - S_q(l,k) \right|}{M \bullet Numprojections} \qquad \textbf{(2)}$$

Where:

M = number of pixels in the image

N = number of pixels on one side of the image (M = nxn)

NumProjections = the number of projections used

$P^0(i,j)$ = constant
while q < 20
  for *l* = 1 to NumProjections
    for i = 1 to n
      for j = 1 to n
        if P(i,j) $\in$ R(*l*,*k*)
          $P^{q+1}(i,j) = P^q(i,j) + (R(l,k) - S^q(l,k))/ N(l,k)$
      next j
    next i
  next *l*
next while…

2.3 Multiplicative ART

The original ART algorithm, sometimes referred to as the Additive ART algorithm, adjusts discrepancies between measured and calculated ray sums by adding an error term to each pixel in a ray. This has the distinct disadvantage that negative values for $P^{q+1}(i,j)$ are possible. One possible approach is to set all

negative pixel values to zero. This, however, introduces additional "truncation" errors. Instead, a Multiplicative ART (MART) algorithm can be used:

$$P^{q+1}(i,j) = P^{q}(i,j) \bullet \frac{R(l,k)}{S^{q}(l,k) \bullet N(l,k)} \qquad \textbf{(3)}$$

(3 shows the approach employed in the Parallel ART algorithm presented in this paper.

2.4 SPARTAF – Streak Prevention

Images reconstructed using ART algorithms will display streaking artifacts along the angles θ_l used in the reconstruction. The streaks arise because of high contrast edges (such as metal and bone) in the projections. Since the adjustments in the ART algorithm are applied to all pixels in a projected ray, the brightness (or lack of brightness) is spread out over the pixels in the ray. The Streak Preventive ART with Adaptive Filtering (SPARTAF) is designed to reduce streaks before they are introduced by using the 8-neighborhood of each pixel to determine if a streak is forming. According to Rangayyan and Gordon [3], comparison testing indicated that the SPARTAF increased computation time from the ART algorithm by a factor of 2.6. This highlights the need for faster processing to produce better images.

Before showing the pseudo-code for the SPARTAF algorithm, we must first define a few measures of developing contrast, as defined by Rangayyan and Gordon:

Along the Streak:

$$c_1 = \frac{|p_-(i,j) - p(i,j)| + |p_+(i,j) - p(i,j)|}{p_-(i,j) + p_+(i,j) + 2 \bullet p(i,j)} \quad \textbf{(4)}$$

Across the Streak:

$$c_2 = \frac{|m(i,j) - p(i,j)| + |r(i,j) - p(i,j)|}{m(i,j) + r(i,j) + 2 \bullet p(i,j)} \quad \textbf{(5)}$$

On either side of the Streak:

$$c_3 = \frac{|m(i,j) - r(i,j)|}{m(i,j) + r(i,j)} \quad \textbf{(6)}$$

Where:

p(i,j) is in the ray (l,k)

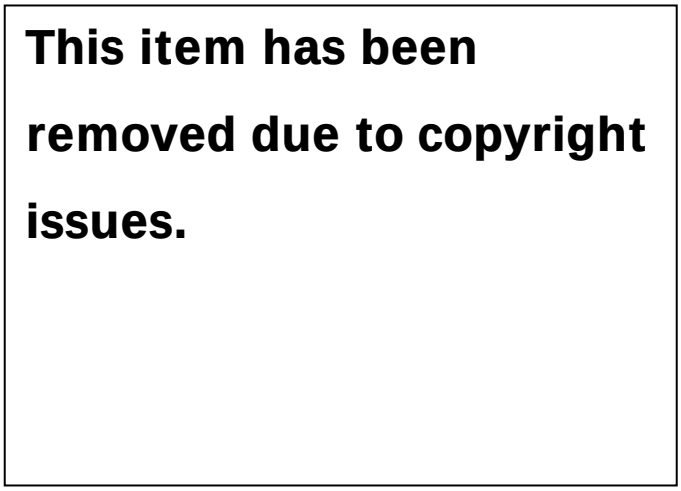
p_(i,j) and p_+(i,j) are the neighbors of p(i,j) along the ray (l,k) at the previous iteration q

m(i,j) = current average of the neighbors of p(i,j) in the ray R (l, k-1)

r(i,j) = current average of the neighbors of p(i,j) in the ray R (l, k+1)

(see Figure 2)

Figure 2: Reconstruction ray geometry [40].



We now use these defined measures of contrast, along with threshold parameters l_1, l_2, l_3 to produce the pseudo-code for SPARTAF:

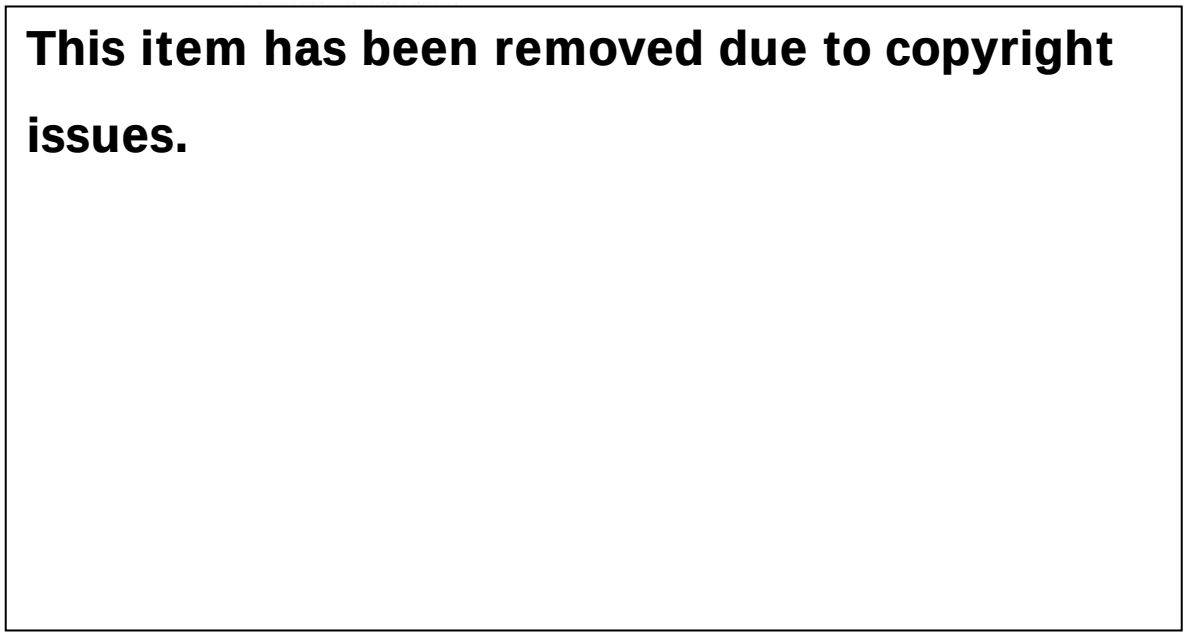
```
while q < 20
    for / = 1 to P
        for i = 1 to n
            for j = 1 to n
                if P(i,j) ∈ R(/,k)
                    P^{q+1}(i,j) = P^q(i,j) + (R(/,k) - S^q(/,k))/ N(/,k)
                        (c1 < l1) AND (c2 > l2) //then a streak!!
                        if c1 > l3
                            if | m(i,j) – p^{q+1}(i,j)| < | r(i,j) – p^q(i,j)|
                                a(i,j) = m(i,j)
                            else
                                a(i,j) = r(i,j)
                          else
                            a(i,j) = (m(i,j) + r(i,j))/2
                    else //no streak, proceed with ART
                        a(i,j) = P^{q+1}(i,j)
                next j
            next i
    next /
next while…
```

2.5 Fourier Backprojection

2.5.1 Fourier Slice Theorem

The current industry standard family of algorithms are commonly known as the *Fourier Backprojection* [1,11]. Based on the Fourier Slice Theorem, this algorithm takes advantage of the fact that the one dimensional Fourier transform of each projection is actually a slice of the two dimensional Fourier transform of original object (Figure 3). By using the 1D Fourier transform of projections from several angles, a rough estimate of the original object can be calculated.

Figure 3: The Fourier transform of an objects projection at a particular angle gives a slice of the 2D Fourier transform of the object [47].



If our object is represented by a function f(x,y), then the two dimensional Fourier transform is given by

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy \quad \textbf{(7)}$$

For a projection at angle θ, $P_\theta(t)$, the corresponding 1D Fourier transform is given by

$$S_\theta(w) = \int_{-\infty}^{\infty} P_\theta(t)e^{-j2\pi wt}dt \quad \textbf{(8)}$$

Consider the Fourier transform of the object along the line v = 0, shown here:

$$F(u,0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi ux}dxdy \quad \textbf{(9)}$$

This expression can be further manipulated to

$$F(u,0) = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(x,y)dy\right]e^{-j2\pi ux}dx \quad \textbf{(10)}$$

The term in the square brackets is in fact a parallel projection along a line where x is constant. In other words,

$$P_{\theta=0}(x) = \int_{-\infty}^{\infty} f(x,y)dy \quad \textbf{(11)}$$

Substituting this $P_{\theta=0}$ into Equation (10 yields

$$F(u,0) = \int_{-\infty}^{\infty} P_{\theta=0}(x)e^{-j2\pi ux}dx \quad \textbf{(12)}$$

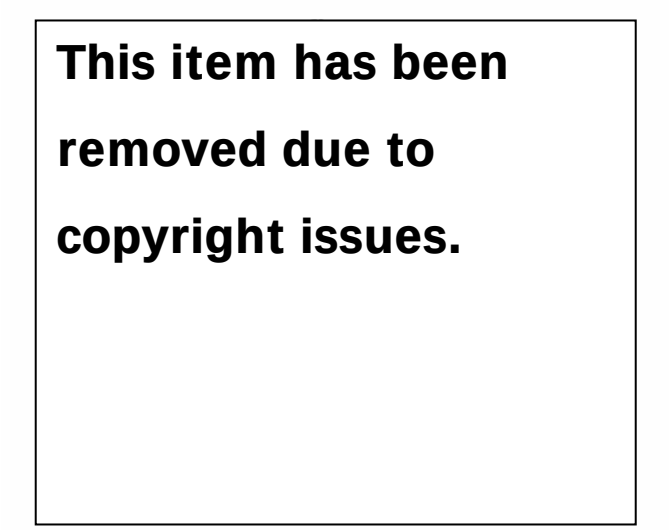
And we have shown that

$$F(u,0) = S_{\theta=0}(u) \quad \textbf{(13)}$$

Clearly, this result is equally as valid if the object were rotated. Thus, we can see that this result is valid for any orientation of the coordinate axis with respect to the object, and valid for all θ.

By measuring projections at several θ, we can calculate F(u,v) along several radial lines, as shown in Figure 4. These values calculated in this way can be used to estimate the rest of the 2D Fourier transform and "connect the dots". Section 3.3 develops the Fourier Backprojection method for reconstruction from projections. This algorithm takes advantage of the Fourier Slice theorem and a filtering/interpolation scheme to estimate the Fourier Transform of the object.

Figure 4: The radial lines represent points in the object's 2D Fourier transform [47].



2.5.2 Filtered Backprojection

From the Fourier Slice Theorem, we know that the 1D Fourier transform of a projection gives the values along a line in the 2D Fourier transform of the object.

In this sense, taking an object's projection can be seen as a filtering operation on the object's 2D Fourier transform.

Because we are limited to collecting a finite number of projections, we can only calculate the concentric radial lines of the object's 2D Fourier Transform (Figure 5). A simple method to estimate the rest of the 2D Fourier transform is to use the width of the wedge (Figure 5a) to weight the values along the line θ (Figure 5b), producing a weighted line shown in Figure 5c. For example, with K projections over 180^o, each wedge would have a width of $\frac{2\pi|w|}{K}$ at frequency *w*. In graphical terms, *w* is the distance from the center of the graph, or the point of the wedge. When many projections are taken and the wedge is narrow, we can approximate the wedge by simply weighting the line by a factor of $\frac{2\pi|w|}{K}$, as shown in Figure 5c.

Figure 5: Using the slice to estimate a section of the 2D Fourier Transform: (a) shows the wedge represented by a single line. (b) shows the actual line of elements calculated, and (c) shows a weighting scheme which, when applied to (b), approximates the wedge in (a) [47].

The 2D inverse Fourier transform of the estimate in Figure 5c produces a very fuzzy reconstruction of the original object. We can refine this estimate by summing it with the 2D inverse Fourier transforms of the weighted transforms of projections from other angles.

The greatest advantage of this method is that reconstruction can begin as soon as the first projection is acquired. The other advantage is that the interpolation is generally more accurate in the spatial domain than in the frequency domain.

3 Parallel ART

3.1 Introduction

The Algebraic Reconstruction Technique and Fourier Backprojection families of algorithms both have advantages and disadvantages. The main disadvantage of the FBP algorithms is the requirement of many projection angles, and hence more radiation dose required for patients. This is due to the interpolative nature of the FBP, as opposed to the iterative nature of the ART family. The primary disadvantage of the ART algorithms is the long processing time. In order to speed up an ART type algorithm, parallel processing can preserve the superior image quality while improving the processing time. The sections below discuss the nature of the ART algorithm and propose several redesigned algorithms for parallel processing. Both message passing and shared memory alternatives are explored.

3.2 Data Dependencies

The first consideration when parallelizing a sequential algorithm is determining the existence of data dependencies. CT reconstruction algorithms are solutions to the problem of "image reconstruction from projections". This problem is usually characterized by a set of underdetermined equations; in other words, the result is an approximation to the solution, as opposed to the actual solution. Consequently, the data dependencies inherent in iterative algorithms such as

ART are not necessarily inherent to the problem. Most importantly, violating an algorithm's data dependencies may change the particular solution, but not necessarily degrade the quality of the resulting solution.

An excellent example of this is Guan and Gordon's idea [6] of changing the order in which projections are considered in the reconstructions. Guan and Gordon suggested a multilevel access scheme (MAS), attempting to order projections so that consecutive projections are about 90° apart. They showed that such an MAS would dramatically increase the speed of the convergence of an ART algorithm. In fact, they also showed that a random access scheme (RAS) was nearly as fast in convergence as with an MAS.

Regardless, we must identify the data dependencies inherent in the ART algorithm before deciding whether or not they can be ignored. The 3-nested-loop structure of the algorithms implies that the data dependencies, when present, will generally exist between successive iterations of the loops. Thus, there will be three possible levels of data dependencies: between successive iterations (estimate at different values of q), between consideration of projections (value of $P^q(i,j)$ as different projection angles θ_l are considered), and between different rays (value of $P^q(i,j)$ as different rays, k, in projection at angle θ_l are considered)

The most obvious dependency is the relationship between iterations, i.e., image estimate $P^{q+1}(i,j)$ cannot be calculated before image estimate $P^{q}(i,j)$.

Next, there is the dependency of the new estimate of pixel value $P^{q+1}(i,j)$ on previous estimates at angles already considered *in the same iteration*. For example, for some angle θ_l, the newly calculated $P^{q+1}(i,j)$ becomes $P^{q}(i,j)$ in the calculation of $P^{q+1}(i,j)$ for θ_{l+1}. In addition, this new $P^{q}(i,j)$ is also used in the calculation of the sum $S^{q}(l,k+1)$. In this way the notation used above in the formula for $P^{q+1}(i,j)$ and $S^{q}(l,k+1)$ is somewhat misleading. However, since this notation is consistent with existing literature, this paper will preserve it. Note that it is the order of this dependency that Guan and Gordon found was actually *hindering* the convergence of the algorithm.

Finally, dependencies arise in the calculation of $P^{q+1}(i,j)$ as each ray in a particular projection is considered. The ray width is usually calculated such that each ray contains only one pixel in a particular row or column. In addition, pixels are often approximated as points such that each pixel is part of only one ray [6]. In this case, the calculation of the pixels in a particular ray is independent of the calculations of the pixels for all the other rays. This is not true, however, for the SPARTAF algorithm. Thus the SPARTAF introduces a dependency between the pixel $P^{q+1}(i,j)$ and all of it's 8-neighbors, which belong to adjacent rays.

3.3 Alternatives

3.3.1 Message Passing

In any imaging problem, the most logical starting place is to attempt to partition the image data using some scheme and assign one or more partitions to each processor. Due to data dependencies explained above, The ART algorithm is not suitable for image partitioning and options must be weighed carefully.

The three loops of the basic ART imply the possibility of loop unrolling:

WHILE loop (until the current estimate is very close to previous estimate)

FOR loop (over all projection angles)

FOR loop (over all rays of the current projection angle)

A logical place to begin is with the coarsest grain division - the outer-most loop. This WHILE loop iterates once for each successive estimate of the reconstructed picture. As revealed in the analysis of data dependencies, each estimate is based on the previous estimate. The process vaguely resembles a pipline type process. Unfortunately, the pipelining approach will not work because each stage of the pipeline will only see one unit of data in the two dimensional case considered here. (Three-dimensional slice CT may avail itself to this approach, but that consideration is beyond the scope of this paper.) Conversely, unrolling this loop (assuming ***n*** iterations) using ***n*** processors would give us ***n*** identical estimates, each equal to the first iteration of the sequential algorithm. This

tactic clearly breaks the unavoidable data dependency between iterations and accomplishes no advantage in comparison with the sequential algorithm.

The FOR loop which controls the consideration of projection angles might also be unrolled. The current image estimate adjusted to accommodate each projection, starting with the projection at angle θ_0 and ending with the projection at angle θ_l. The estimate at each angle is based on the estimate produced by the previous angle. As discussed above, the angles can be considered in a random order without deteriorating the quality of the estimate. Unfortunately, even with this random ordering, the estimate of the image after θ_x is still based on the estimate after θ_{x-1}.

Let us consider the ramifications of ignoring this data dependency. Looking at the very first iteration, and the very first projection considered in that iteration, the resulting image will be the projection "smeared" back across the rays, with all pixels of a particular ray having the same value. If the next angle θ_1 is considered based on the initial constant value instead of on $P^1(i,j)$ (after θ_0), the result will similarly be the projection smeared back across its rays. If all angles are considered this way, we can form a crude estimate by adding these "smeared" images together. A discussion found in Gordon, et al., [4] displays an image thus reconstructed. Presumably averaging would be more appropriate to achieve pixel values within the proper range. Successive iterations could use this

averaged image as $P^q(i,j)$, instead of the incrementally modified $P^q(i,j)$ resulting from the original algorithm. Such an alteration is easily parallelized by dividing the projections among the processors, collecting the averaged result, and continuing the next iteration (next ***q***).

A pseudo-code representation of this alteration is shown below, using communication routines similar to Message Passing Interface (MPI). The number of processors ***n*** is assumed equal to the number of projections.

ALGORITHM 1m

```
E = threshold
if (my_rank == root)
   P⁰(i,j) = constant
// send projection for angle n to processor n
Scatter( R(l,k), Projections(k), root)
while q < 20
   Bcast(Pᵠ(i,j), root)
   for i = 1 to n
      for j = 1 to n
         if P(i,j) ∈ R(n,k)
            Pᵠ⁺¹(i,j) = Pᵠ(i,j) + (Projections(k) -
         Sᵠ(n,k))/ N(n,k)
      next j
   next i
   Reduce(Pᵠ⁺¹(i,j), Pᵠ(i,j), AVERAGE)
   Bcast(Pᵠ⁺¹, root)
next while…
```

A generalization to cases where ***n*** is not equal to the number of projections is straight forward from this implementation, using a mapping such as cyclic or block cyclic assignment of projections to processors.

This alteration will most likely cause a slower convergence, requiring more iterations and resulting in a *sub-optimal* parallelization. However, if the work is divided on 10 processors, and takes twice as many iterations to converge, a speed up of 5 times is still achieved (in an ideal situation ignoring communication latency). This is still a significant improvement and well worth investigating by experiment. This Parallel ART (PART) algorithm does not violate the data dependency between neighboring pixels introduced by the SPARTAF, and so could be used to implement each of the ART, MART, or SPARTAF algorithms.

The inner FOR loop controls the consideration of rays within a projection. For the ART and MART, dependencies do not exist between pixels of adjoining arrays, so this loop could be unrolled, similar to the PART algorithm above. Each processor would consider a particular ray ***k*** of each projection.

A pseudo-code representation of this alteration is shown below, again using communication routines similar to Message Passing Interface (MPI). The number of processors ***n*** is assumed equal to the number of rays in a projection.

ALGORITHM 2m

```
if (my_rank == root)
   P^0(i,j) = ETC

while q < 20
   for l = 0 to NumProjections-1
      Bcast(P^q(i,j), root)
      P^q+1(i,j) = 0
```

```
      // send ray sum for angle l, ray my_rank to
   processor my_rank
      Scatter( R(l,k), RaySum, root)
      if P(i,j) ∈ R(l,my_rank)
         P^(q+1)(i,j) = P^q(i,j) + RaySum - S^q(l, my_rank))/
      N(l,my_rank)
      Reduce(P^(q+1)(i,j), P^q(i,j), ADD)
   next l
   if (my_rank == root)
      calculateE
   Bcast(E, root)
next while…
```

This parallelization is somewhat more complex than the PART algorithm above. Its finer grain division requires significantly more communication between processors. The Broadcast of $P^q(i,j)$, the Scatter of projection data, and the Reduce operation are all being executed inside the FOR ***l*** loop. This results is an increase in the number of communication calls by a factor of *NumProjections*. For the Scatter operation the amount of data sent each time is less, but the total amount of data sent is equal. The Broadcast and Reduce operations, however, are still communicating the entire $P^q(i,j)$ image array each time through the loop. The amount of data sent could be decreased by only sending the needed pixels, but this requires a significant amount of processing to determine which pixels fall in which rays. With enough memory, these could be calculated ahead of time, but communicating this information to all the nodes would be a bottleneck.

Another complication to this latest implementation is that the workload cannot easily be distributed evenly among processors. The number of rays per projection is not constant. The ray width, however, is usually constant (see [1]),

so that projections at some angles (e.g. 45°) are *wider* and require more rays. The minimum number of rays occurs at angles of 0° and 90°, so that at these angles, several processors with low ranks and high ranks will have no pixels in the rays assigned to them. They will not be able to do work until the angle has changed such that they do have some pixels in their assigned rays. Even at the maximum projection width at an angle of 45°, these "end" processors will have very few pixels compared to processors near the center of the projection. Thus, these processors at the "ends" will always have a light workload. A possible solution to this is some sort of non-linear assignment scheme to give the "end" processors more rays than processors near the middle. For the case where the number of processors is much less than the number of rays, a cyclic assignment scheme is ideal to even out the workloads.

The flaw of this algorithm is that it violates the data dependencies of the SPARTAF algorithm. However, as SPARTAF is designed to suppress the growth of *nascent* (newly forming) streaks, this may not matter.

3.3.2 Shared Memory

The message passing approach reveals disappointing experimental results (see Section 4.2). Closer examination reveals that the MPI alternatives are communication laden. A shared memory approach offers an attractive alternative, eliminating the inefficiency of this communication.

Both Algorithm 1m and Algorithm 2m presented above can be more efficiently implemented with a shared memory architecture. The following describe modified algorithms, 1s and 2s, using the shared memory directives of OpenMP.

As in Algorithm 1m, Algorithm 1s unrolls the first for loop (*for l*). As with the serial algorithm, averaging is presumably an appropriate operation to reduce the images calculated by each thread. This makes sense for the simple case where each thread processes the angles of a single projection (one angle). In general though, this may not be the case. Allocating several angles to each thread may in fact prove to be more efficient. As discussed above, Guan and Gordon [6] showed that a Multilevel Access Scheme produced faster convergence of the MART algorithm. A round robin allocation of projection angles would approximate this approach while maximizing the benefit of parallel threads. However, as the individual threads attempt to modify the shared image estimate data, collisions become possible. If two different threads attempt to apply a multiplicative adjustment to the same pixel at the same time, a collision occurs. Assuming a FIFO queue waiting to access a single picture element, and a large number of image elements (relative to the number of threads), the waiting time will be negligible. With this approach, the pixel value used to calculate the multiplier may be changed by another thread before the multiplier can be applied. Such a value, which is modified by another process between when it is read and when it is processed, is said to be a "stale". Using stale pixel values may delay

convergence, but any convergent solution is equally as valid. As discussed earlier, the ART algorithm is an approximation of the image solution.

The OpenMP specification document ([8]) states, "The correctness of a program must not depend on which thread executes a particular iteration." Further, "An OpenMP-compliant program should not rely on … a schedule *kind* conforming precisely to the description … because it is possible to have variations in the implementations of the same schedule *kind* across different compilers." We are proposing a new variation of the MART and we've noted that ART an MART are approximate solutions, with other equally correct solutions possible. Because of this we are not concerned with strictly adhering to order in which processing takes place in ART or MART, or "correctness" in relation to the classical sequential ART or MART. We are more concerned with a convergent solution that is correct in the sense that it approaches the original object. However, it should be noted that this algorithm may produce slightly variant results on a different platform.

A pseudo-code representation of this alteration is shown below, using directives similar to the OpenMP application programming interface for shared memory programming. The number of processors ***n*** is assumed equal to the number of projections.

ALGORITHM 1s

```
#parallel shared P^0(i,j) = constant

//share projection matrix
parallel shared projections(  R(l,k), Projections(k))

// do 20 iterations
while q < 20
   #parallel shared P^(q+1)(i,j) = 0
   #parallel for schedule (static scheduling, blocksize 1)
      //each thread gets one angle's projection
      for l = 1 to numProjections
         for k = 1 to numRays
           for i = 1 to n
              for j = 1 to n
                if P(i,j) ∈ R(n,k)
                   P^(q+1)(i,j)  =  P^q(i,j)  +  (Projections(k)  -
                                                 S^q(n,k))/ N(n,k)
               next j
            next i
          next k
       next l
next while…
```

The inner FOR loop controls the consideration of rays within a projection. For the ART and MART, dependencies do not exist between pixels of adjoining rays, so this loop could be unrolled, similar to the PART algorithm above. Each processor would consider a particular ray ***k*** of each projection, as shown below in Algorithm 2s.

This finer grained parallelism requires more overhead because threads will be initialized and rejoined for every single ray of every projection of every iteration. In other words, each thread does less work, requiring more threads to accomplish the same amount of work. The advantage of this approach is that it does not require any modification of the original and proven MART algorithm.

A pseudo-code representation of this direct parallelization is shown below, again using shared memory directives similar to OpenMP. The number of processors ***n*** is assumed equal to the number of rays in a projection.

ALGORITHM 2s

```
#parallel shared P^0(i,j) = constant

//share projection matrix
parallel shared projections(  R(l,k), Projections(k))

// do 20 iterations
while q < 20
   #parallel shared P^(q+1)(i,j) = 0
   for l = 1 to numProjections// for each angle
      #parallel section
      #parallel for schedule (static scheduling, blocksize
                                                        1)
         //each thread gets one ray of the projection
         for k = 1 to numRays
            for i = 1 to n
               for j = 1 to n
                  if P(i,j) ∈ R(l,k)
                     P^(q+1)(i,j) = P^q(i,j) + (Projections(l,k)
                                    - S^q(l,k))/ N(n,k)
               next j
            next i
         next k
  next l
next while…
```

Similar to the contrast between the message passing algorithms, Algorithm 1s utilizes a finer grained parallelism than Algorithm 2s. In practical terms, finer grained parallelism in the shared memory realm requires more overhead in creation of threads. In addition, finer grained parallelism implies the need to divide labour in calculating which pixels fall into which rays, as well as sharing

the resulting information. And as with Algorithm 2m, Algorithm 1m introduces a load balancing issue because of the variation in the number of rays per projection and the number of pixels per ray as well as violating the data dependencies of the SPARTAF algorithm.

It would seem more logical then to use a courser grained parallelism, with a simpler division of labour and less overhead in spite of the potential for collisions in accessing a particular pixel value.

4 Implementation

4.1 Experimental Platform

4.1.1 Message Passing

The Parallel ART algorithm was coded in C language, using Message Passing Interface (MPI) for communication. The program was run on the UNIX network of the Electrical and Computer Engineering Department at the University of Manitoba. This network uses the SunOS 5.9 operating system and employs the MPICH implementation of MPI on a Beowulf cluster.

4.1.2 Shared Memory

4.1.2.1 Mercury

The shared memory PART was coded in C language using OpenMP directives. The program was initially run on the Mercury shared memory machine of the Computer Science Department at the University of Manitoba. The machine has 8 processors capable of running 72 threads each, for a total of 576 threads possible. The Linux Red Hat 7.1 operating system is the base for the Omni OpenMP library, currently running Omni compiler version 1.3s. Code was initially compiled and run on this machine.

4.1.2.2 WestGrid

PART was subsequently ported to the Western Canadian Research Grid (WestGrid). WestGrid is a high powered computing (HPC) consortium offering

HPC and networking facilities to several member institutions in Western Canada, listed in Table 1.

Table 1: WestGrid member facilities.

Facility Name
Simon Frasier University
The Banff Centre
TRIUMF
University of Alberta
University of Calgary
University of Lethbridge

Researchers from other affiliated institutions, such as the University of Manitoba, can apply to access the 10% of resource time earmarked for outside projects. The $50 million dollar initiative offers several grid-enabled clusters. Nexus and Cortex are the only groups to feature architectures favorable for OpenMP shared memory applications.

4.1.2.2.1 Nexus

Nexus is a group of Silicon Graphics Inc (SGI) symmetric multiprocessing (SMP) machines. According to Flynn's Taxonomy, these machines are classified as Multiple Instruction, Multiple Data (MIMD). Each machine in this group is an SGI Origin, with varying models but similar architectures. Table 2 shows a summary of hardware specifications for this group.

Table 2: Nexus group hardware summary.

Machine Name	Model	Number of processors	CPU
Nexus	Origin 350	8	MIPS R16000 rev2.1 @ 700 MHz
Arcturus	Origin 3900	256	MIPS R16000 rev2.1 @ 700 MHz
Aurora	Origin 2000	36	MIPS R10000 rev2.6 @ 195 MHz
Borealis	Origin 2400	64	MIPS R12000 rev2.1 @ 400 MHz
Australis	Origin 3800	64	MIPS R12000 rev3.5 @ 400 MHz
Helios	Origin 300	32	MIPS R14000 rev1.4 @ 500 MHz
Corona	Origin 300	32	MIPS R14000 rev1.4 @ 500 MHz

Specific experiments on the Nexus group were run on Australis (64 processors) and Helios (32 processors).

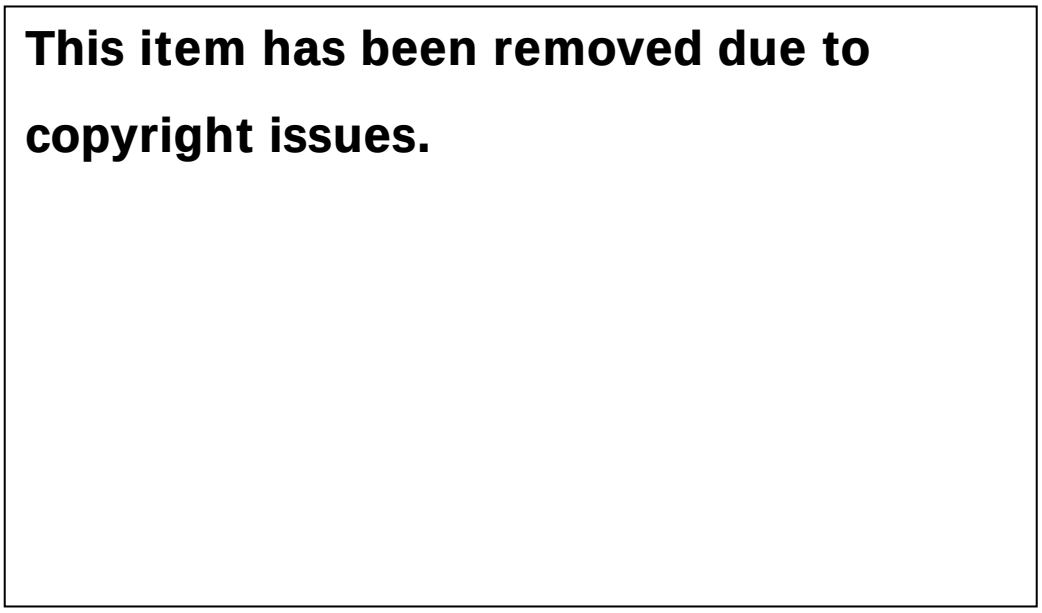
Because the Origin has memory scattered through the system with all memory accessible globally by all other processors, it is known as a distributed shared memory (DSM) system. In such architectures, memory access latency varies depending on the physical location of the data. For example, access time for memory on a processor's local memory section is faster than access time for memory stored on another processor's local memory section. This is known as Non-Uniform Memory Access, or NUMA.

One design feature of Australis that reduces memory access latency is the cached memory local (and private) to each processor. However, cached copies of shared memory pose a risk of that cached memory becoming out of date when another processor modifies the global master copy of those values. The out of date values in cached memory are known as *stale values*. The Origin

addresses this with a memory architecture named Cache Coherent Non-uniform Memory Access, or ccNUMA. Cache coherency is maintained by a memory directory system that tracks cached copies of memory locations at the block level (128 bytes). Before processor A modifies its cached memory, the directory calls a method to purge all unmodified (stale) copies of the memory and grants processor A exclusive rights to that block, or blocks of memory.

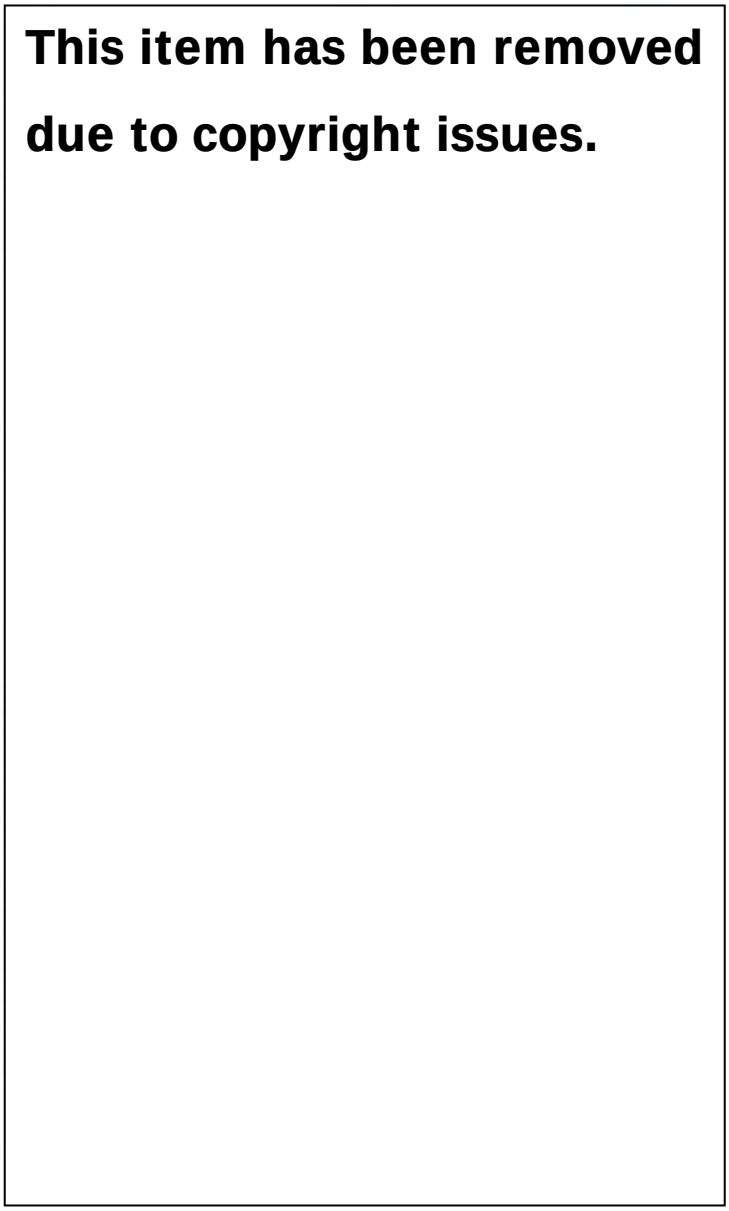
Processors are grouped together in modules called "C-Bricks" in SGI terminology. A C-brick in an Origin 3800 such as Australis contains 4 processors and a block of memory, as shown in Figure 6 below [51].

Figure 6: Block diagram of an SGI Origin 3000 series "c-brick" [51].



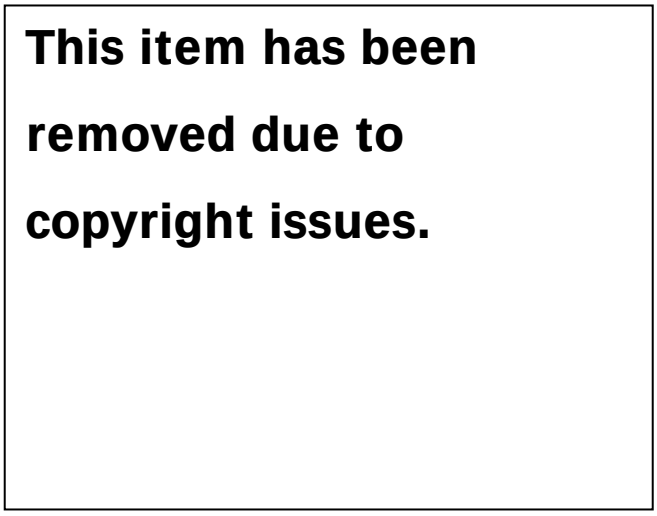
The four processors share the local main memory block, and are connected to other c-bricks via an "r-brick" or router module. The interconnection of the processor modules (c-bricks) can be seen in Figure 7.

Figure 7: Interconnection of modules in an SGI Origin 3800 [51].



Each router block in this configuration has 8 connections, forming an eight-port crossbar switch as shown in Figure 8.

Figure 8: Origin3000 Router interconnect schematic [51].



The interconnection of processors and memory in the Origin 3000 forms a mesh topology with several paths between each processor and between processors and memory banks. Each C-brick contains a single block of memory. Looking at the connections of one router brick, you can see that each memory block is connected by one hop to 3 other memory blocks, and connected by at most 2 hops to every other memory block in the system. Thus the system is more connected than a hypercube, but not quite fully connected.

The operating system for all machines in the Nexus group is Irix 6.5 UNIX, using SGI's MIPSPro 7.4 C compiler, with support for OpenMP 2.0.

4.1.2.2.2 Cortex

The Cortex group is a new collection of IBM p5 systems with OpenMP capability. Cortex is the head node, with Dendrite and Synapse being strictly batch processing machines. Table 3 shows a hardware summary of this group.

Table 3: Hardware summary of Cortex group of WestGrid.

Machine Name	Model	Number of processors	CPU
Cortex	IBM 550	4	Power5 @ 1.5 GHz
Dendrite	IBM 595	64	Power5 @ 1.9 GHz
Synapse	IBM 595	64	Power5 @ 1.9 GHz

The p5 chip encapsulates two identical processor cores, each capable of 2 logical threads. In this sense, it appears as a 4-way symmetric multiprocessor to the operating system. IBM considers each core of the chip one processor, so that a multiple chip module (MCM) containing 4 p5 chips is actually 8 processors. Processors are physically and logically organized into processor books. Each book contains 2 MCM's for a total of 16 processors in a processor book. Figure 9 shows the memory interconnections within a processor book. Within the processor book, each chip is directly connected to 3 other chips, forming a cubic topology.

Figure 9: IBM p595 memory interconnect diagram (16 processors) [54].

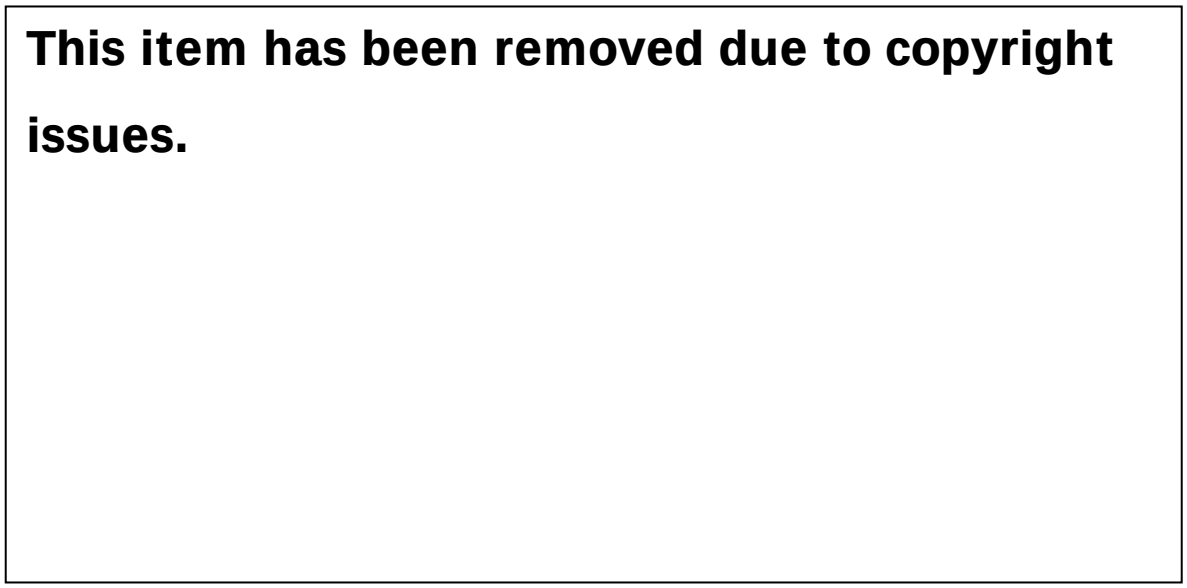
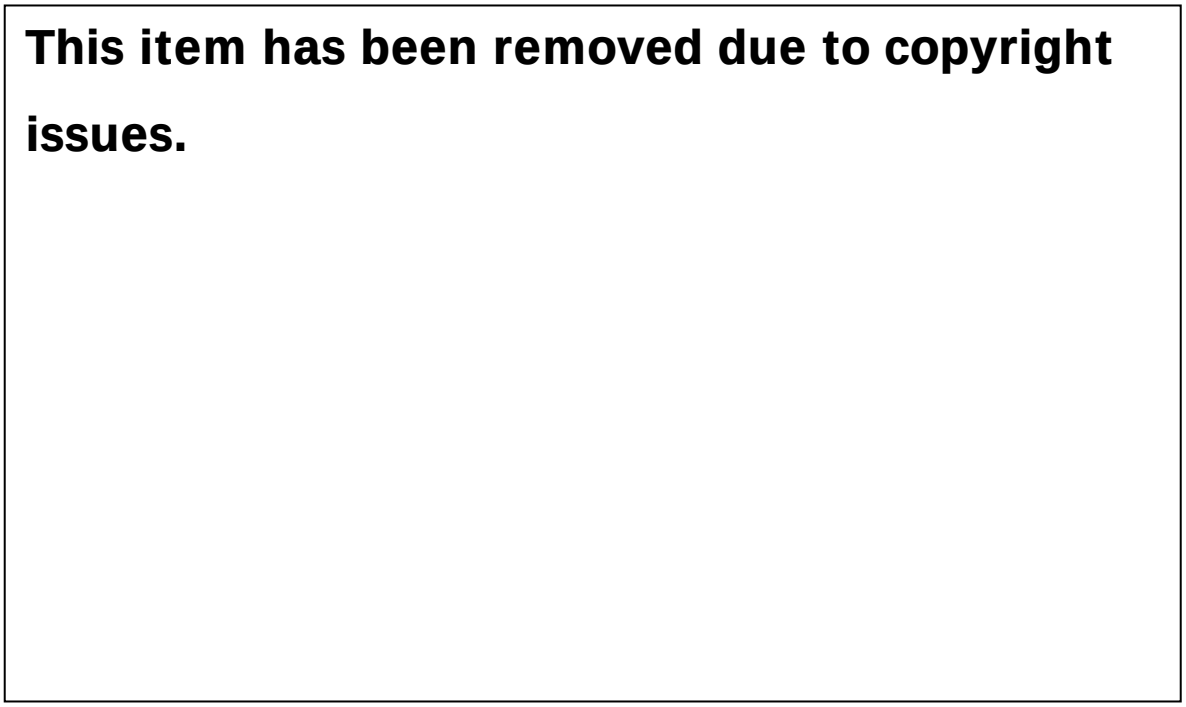


Figure 10 shows the connections between the 4 processor books of a 64 processor p595 such as Dendrite or Synapse. Here, each processor is also connected to a processor in another book. Because Book0 and Book3 in the diagram are not connected, the 64 processor interconnection is not quite a hypercube topology, though it still qualifies as a mesh topology.

All three machines of the Cortex group run IBM's AIX v5.3 operating system and ***xlc*** v8.0 compiler.

Figure 10: IBM p595 memory interconnect diagram (64 processors) [54].



4.1.2.3 Portability

Changing platforms from Mercury to the WestGrid group was initially undertaken to prove the portability of the code and explore the technology offerings of WestGrid. The movement to the Nexus and Cortex platforms resulted in 3 technological advantages. First, with images larger than 512x512 pixels, the output data for 50 iterations surpasses the storage available under a student computing account on Mercury. A second advantage of the Nexus and Cortex machines is the additional stack space available. Initial code versions used statically declared arrays for input data and image estimates data structures. While theoretically faster, statically declared data breached the limits for stack space for images larger than 1024x1024 pixels. Finally, WestGrid provides a job queuing structure so that jobs can run with exclusive control over the processors

they use. This provides an additional dimension of reliability and accuracy to timing measurements. Minor compiler differences were encountered but easily overcome.

The P5 processors used on the Cortex machines feature 64 bit computing capability. Taking advantage of this offering is not entirely straight forward. When compiling applications for 64 bit capability, a few key differences can lead to problems. One prime example is the difference in storage size between pointers and integers on 64-bit platforms. On 32-bit platforms, both integers and pointers require 4 bytes of storage, whereas the expanded memory space on 64-bit platforms require 8 bytes of storage for pointers, with integers still requiring 4 bytes.

Pointers can be implicitly cast to integers and visa versa on 32-bit platforms without penalty. However, applications which use this implicit casting will experience truncation of pointers which are implicitly cast to integers. One example of this is seen below, provided by IBM's ***xlc*** programming guide:

```
a=(char*) calloc(25);
```

If *calloc* is not prototyped, the compiler assumes that it returns an integer. Compilers run in 64-bit mode will truncate the 8byte pointer returned in order to implicitly cast it to a 4 byte integer, even though it will be subsequently re-cast

to an 8 byte char pointer. This particular problem can be solved by including `stdllib.h`, which includes a function prototype for *`calloc`*.

The example presented is in itself not a major difficulty, although it could be extremely hard to diagnose without prior knowledge. The example does, however, underline the need for adherence to explicit programming techniques to maximize the portability of an application.

The major difficulty with WestGrid was the waiting time for experiments. Interactive use is limited to at most 2 processors, and is only allowed on the head node NEXUS (not allowed at all on Cortex). This is sufficient for simple code debugging, but does not provide the ability of the code to execute for many threads.

Experiments and more advanced debugging have to be submitted for queuing using Portable Batch System. Using a priority scheme, this system allocates exclusive use of the required number of processors for the duration of each job. Unfortunately, most other jobs in the queue require runtimes on the order of 24 hours or longer, so waiting for results of a single experiment can take several days even though the queues are relatively short and the experiment itself might take less than one hour. Any small errors in the batch script or PART code can result in substantial delays in development and experimentation. Cortex was

favored over Nexus in the end because it's newer, more advanced technology led to shorter queuing times.

While the code itself could be debugged on Nexus before being submitted on Cortex, the different operating system and compilers on the IBM machines (Cortex) required significant changes to batch scripts used for debugging and testing on Nexus. Because Cortex does not allow interactive use, batch scripts developed for Cortex could not be tested before being submitted to its queuing system. This resulted in a steep curve for development of batch scripts.

Another consideration is the problem of monitoring the progress of the software as it runs. Because standard out is be default piped to a file instead of the terminal, monitoring progress and debugging requires a few extra steps. In order to ensure the timely writing of messages from the standard out buffer to the output file, the **fflush()** function had to be called after each message call.

Timing of WestGrid executed programs can be done with the SpeedShop utilities. After code is compiled normally, it is executed using ***ssrun***, producing a data file which is subsequently processed using ***prof***. The utility counts actual clock cycles used for segments of the program, showing percentage and actual time used. This utility is valuable for analyzing CPU resource consumption within a

program, but the required overhead significantly skews the total runtime measurements.

Timing results presented in this paper were measured using the UNIX ***time*** command. These simple results measure the total execution time without any profiling overhead to distort the subject's performance. An additional advantage of this utility is universally available on almost all UNIX systems

4.2 Load Balancing and Threads

The pseudocode presented in Chapter 3 assumes an equal number of angles and processors. This will most likely not be the case in clinical applications, since many projections can be taken and manufacturers will not likely implement reconstruction hardware as expensive and complex as a 64-way symmetric multiprocessor for such applications.

Mercury, Nexus, and Cortex behave similarly in the way that they associate threads with processors. If the number of processors available is greater or equal to the number of threads requested in an OpenMP program, then each thread gets a dedicated processor. An error results when the number of OpenMP threads is greater than the number of processors, so that each processor will have at most a single thread running on it.

The PART reconstruction software, and even other reconstruction software, require little or no I/O in parallel regions outside of memory access. Theoretically, the only possible wasted clock cycles would occur when a memory is requested which is not stored in the local memory cache, resulting in a "cache miss". Cache misses would include cache purged by the cache coherency mechanisms. Assuming few cache misses, multithreading on a single processor is approximately as efficient as running a single thread on one processors. In other words, running PART using 8 threads on 8 processors would be about as fast as using 16 threads on 8 processors

The PART code was designed to request one thread per angle of projection data. A single thread executes the serial section, including loading input data files into memory and initializing variables. During the main body of each iteration, the software instantiates a thread for each row of the projection data. The threads rejoin at the end of each iteration to do some cleanup operations. In this way, each iteration is only as fast as its slowest thread. On a symmetric multi-processor machine like those on WestGrid, one expects each thread to complete its processing in approximately the same amount of time whenever the processors in question are dedicated exclusively to the application.

When fewer processors are available than data items to process, a method is required to best group together and assign the work associated with projection

data. Guan and Gordon [6], and further by Mueller, et al. [7], showed that ART-type algorithms produce better images with faster convergence with non-sequential schemes for considering projections. A random order was found to produce good results, while the best results were achieved when the order that projections are considered is such that the angles between the next projection and previous projections are as close as possible to 90°. Therefore a load balancing scheme can be implemented which attempts to assign a set of projections for each process which are as orthogonal as possible.

A round robin allocation of work could approximate orthogonal angles depending on the number of angles, angle spacing and the number of processors. However, the number of rows (angles) of data to process and the angle spacing are highly variable, even on the same CT machine from one data acquisition to the next. The number of processors available for reconstruction would also vary between CT machines. The code presented here could be modified to accommodate a dynamic allocation scheme to maximize the orthogonality of sequentially considered data units. Results for this modification are not presented.

5 Implementation

5.1 Analytical Results

The problem size for this algorithm is best represented by the size of the "reconstruction grid" - in other words the size of the image to be reconstructed. Hence a problem size N will refer to an image with N pixels on an edge – an NxN image. In addition, two other parameters affect the computation time for CT algorithms. These are the number of projections in the input data (l), and the number of elements in each projection (k). Both of these parameters are usually of the same order as the image size n, so in this analysis we will consider l~=k~=n for simplicity. One last parameter is the number of iterations required to reconstruct the image (q).

5.1.1 The Serial Algorithm

The first action of the serial algorithm is the reading in of input data in the order $O(l \cdot k) \sim = O(n^2)$. Then for each iteration, each of the following operations are executed:

- initialize reconstruction projections $O(q \cdot l \cdot k) \sim = O(q \cdot n^2)$
- for the l projections scattered to each processor
    - calculate reconstruction projections of seed image $O(q \cdot l \cdot n^2) \sim = O(q \cdot n^3)$
    - calculate the adjustment vector $O(q \cdot l \cdot k) \sim = O(q \cdot n^2)$
    - apply adjustments to image $O(q \cdot l \cdot n^2) \sim = O(q \cdot n^3)$

- normalize the image values $O(q \cdot n^2)$

Thus, it is clear that the dominating section of this serial algorithm is of order $O(q \cdot n^3)$.

By comparison, the standard serial FBP algorithm is $O(n^3)$ according to Bassu and Bresler [53].

5.1.2 The Parallel Algorithm

The serial portion of the parallel algorithm includes reading in of input data, scaling the current image estimation after each iteration, and writing the final output to file. Reading the input file is $O(l \cdot k) \sim = O(n^2)$, scaling the input image is $O(n^2)$, and writing the final reconstruction image to file is $O(n^2)$. Thus the serial portion of this algorithm is $O(n^2)$.

Before processing begins, projection data is scattered to all processors, which is of the order $O(l \cdot k) \sim = O(n^2)$. Then, each of the following operations are executed for each iteration:

- current image estimate is broadcast to all processors, $O(q \cdot n^2)$
- initialize local reconstruction projections $O(q \cdot l \cdot k) \sim = O(q \cdot n^2)$
- for the $\frac{l}{p}$ projections scattered to each processor
    - calculate reconstruction projections of seed image or previous iteration reconstruction $O(q \cdot \frac{l}{p} n^2) \sim = O(q \cdot \frac{n^3}{p})$

- calculate the adjustment vector $O(q \cdot \frac{l}{p} \cdot k) \sim = O(q \cdot \frac{n^2}{p})$
- apply adjustments to image $O(q \cdot \frac{l}{p} n^2) \sim = O(q \cdot \frac{n^3}{p})$

- reduce the image from each processor (sum) $O(q \cdot n^2)$
- scale the image values $O(q \cdot n^2)$

Thus, it is clear that the dominating processing of this parallel program will be of order $O(q \cdot \frac{n^3}{p})$.

5.1.3 Speedup

According to Amdahl speedup of the PART algorithm, defined as $\frac{T_{serial}}{T_{parallel}}$, is given by $S = \frac{q \bullet n^3}{q \bullet \frac{n^3}{p}} = p$. This is clearly a theoretical value, which will be reduced by barriers and reduction operations. Both the MPI and OpenMP implementations exploited the same loop to employ parallelism, so this analytical result applies to both scenarios.

5.2 Experimental Results

5.2.1 Convergence

Iterative algorithms require some criteria to term terminate the iterative process. The concept of an iterative approach to image reconstruction from projection

uses an algebraic approximation to iteratively approach a solution. The image is gradually bettered with each iteration, with the improvements lessening with each additional iteration. As the improvement dwindles with higher numbers of iterations, the solution is said to be converging.

As discussed earlier, one common measure of the quality and accuracy of a reconstruction technique is the mean squared error of the resulting image in comparison with the original phantom. In analogy, a common convergence criterion for ART based algorithms is a measurement of the error in of the calculated projections of the reconstructed image, with respect to the actual projection data. The results presented in this paper use the formula offered by Gordon, Bender, and Herman [38] , and revisited by Fitchett [3]. The definition of this quantity is shown below.

$$E = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\frac{\left(p_j - q_j\right)^2}{N_j}} \qquad \textbf{(14)}$$

In this equation:

M = the total number of rays for a set of projections

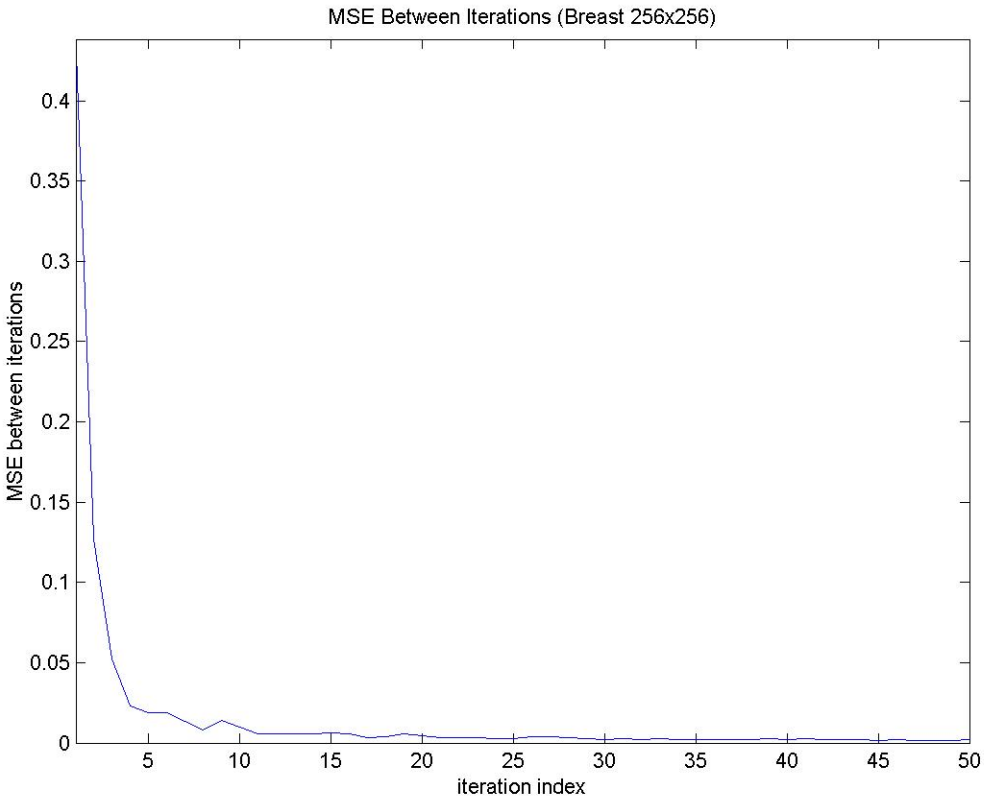
N = total number of pixels

P_j = actual projection values

q_j = calculated projection values from the current iteration

Figure 11 shows the Mean Squared Error between iterations for a reconstruction of the breast phantom (256x256).

Figure 11: MSE between iterations for breast image, 256x256.



This quantity essentially measures the difference between successive iterations, and is calculated according to the following formula:

$$MSE_i = \frac{\sqrt{\sum_i \sum_j (P_{i,j} - C_{i,j})^2}}{N^2} \qquad \textbf{(15)}$$

Where:

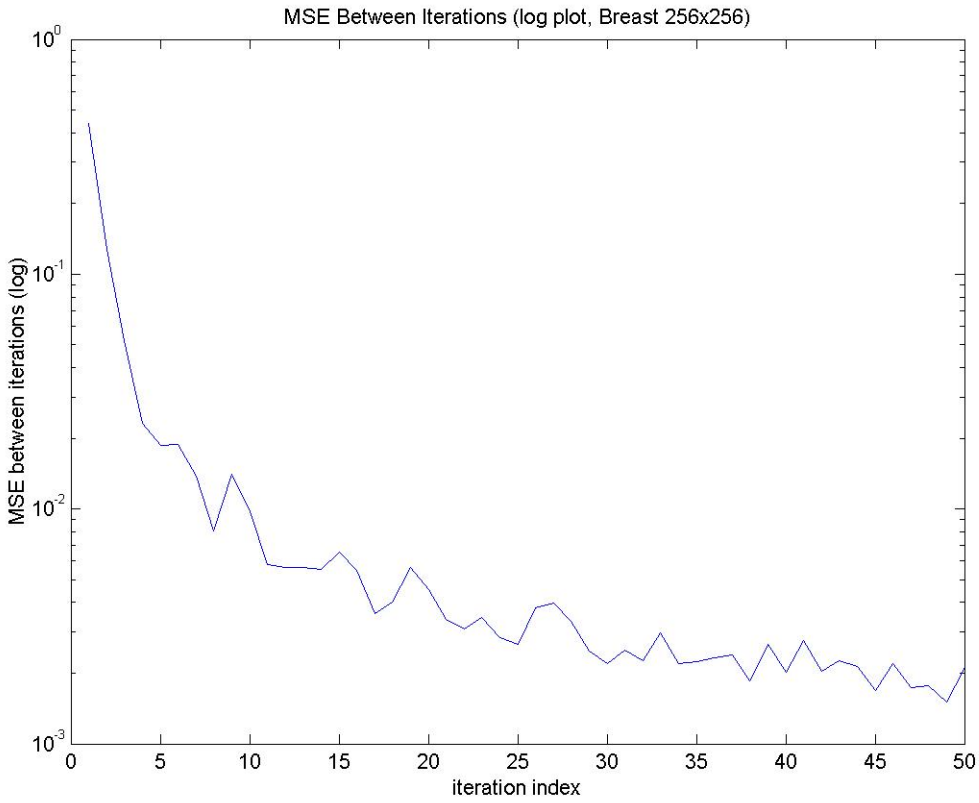
P = previous image estimate

C = current image estimate

N^2 = number of pixels in the image

The same data is plotted on a log scale in Figure 12.

Figure 12: Log plot of MSE between iterations for breast image 256x256.

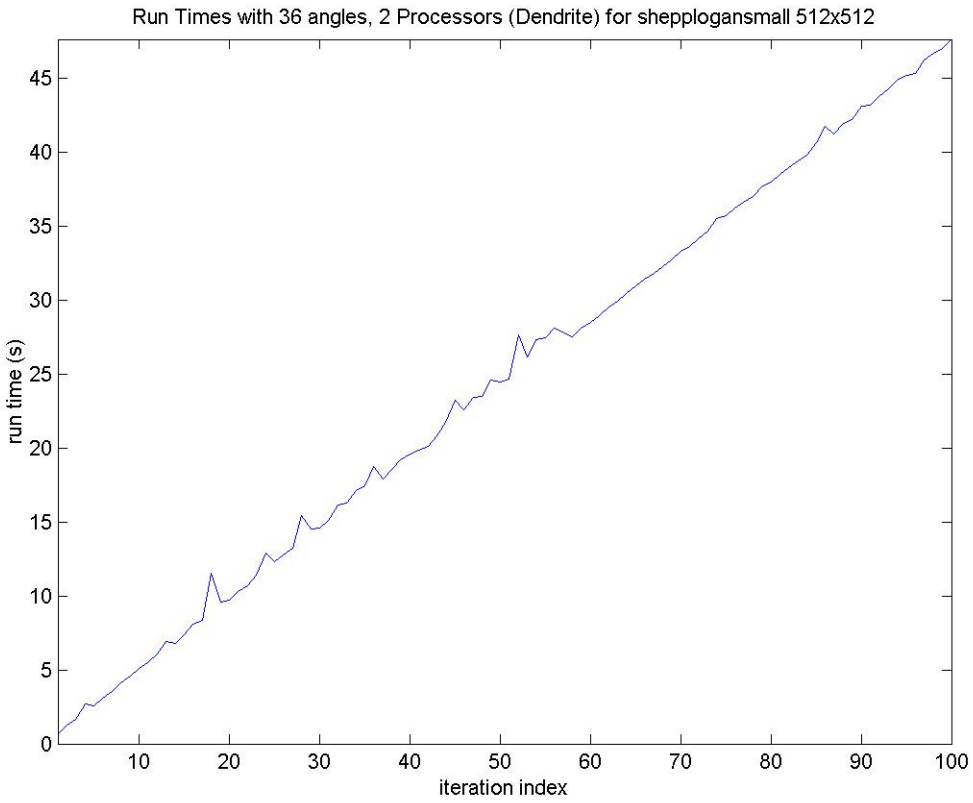


The graph clearly shows the mean squared error approaching zero. By inspecting the actual values, it is determined that the error oscillates about a

value of approximately 0.0020 after 30 iterations. Clearly each successive image is approaching a single solution.

Regardless of the image size or number of projections, the time to process each iteration is relatively consistent within the scope of a single reconstruction. Figure 13 below shows the runtimes for numbers of iterations varying from 1 to 100 for 2 processors. The graph displays the linearity of the run time from one iteration to the next, with similar results for other numbers of processors. From this linearity, we can easily talk about the amount of time to reach convergence as a ratio of the time to complete the total number of iterations. For the 256 x 256 breast image reconstruction, the solution converges at approximately 30 iterations, so we can reliably say the run time to finish is 30% of the time to complete 100 iterations.

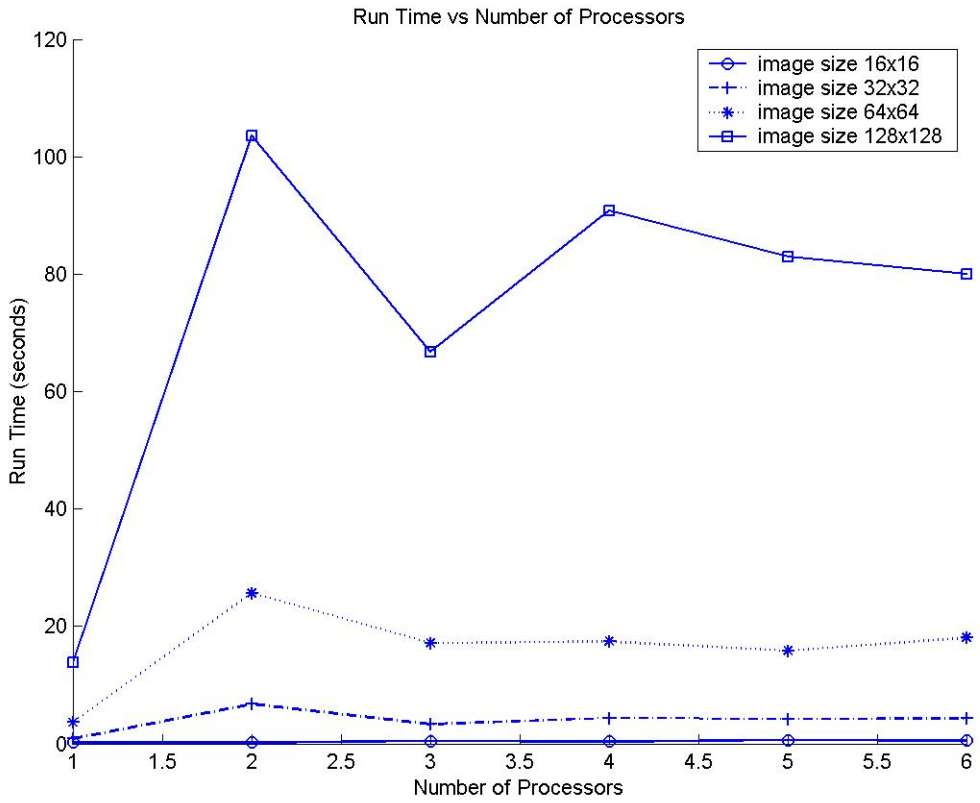
Figure 13: Run time in seconds versus number of iterations.



5.2.2 Computation Time

Empirical results for message passing results exhibited a great deal of variation. In an attempt to smooth the resulting curves, all run times were averaged over a minimum of 4 trials. Several images were reconstructed, varying in size from 16x16 pixels to 128x128 pixels. All data sets included 32 projections, with the number of detector elements in each projection equal to the image size – i.e., for 16x16 image, 16 detector elements.

Figure 14: MPI Run times for various inputs and numbers of processors

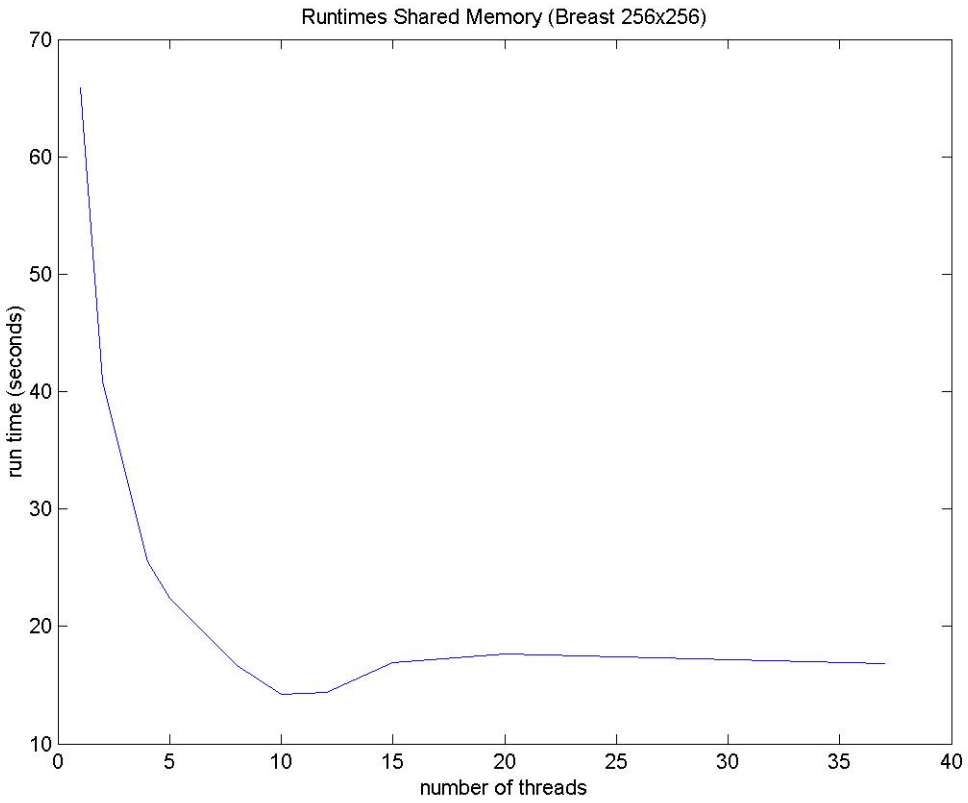


The theoretical speedup of p is clearly not supported by the experimental results shown in Figure 14. As illustrated above, the run times showed a great deal of variance. The two factors which could cause this inconsistent performance are network (communication) latency and shared processing time. Both of these factors were uncontrollable on this experimental platform. Performance gains in processing times were lost due to high communication latencies. This is not surprising when surveying other message passing work, such as Rao, Kriz, Abbott, and Ribbens [14], who also experienced a communication dominant result. The problem of communication latency gets even more significant for

larger problem sizes. The author discusses MPI implementation and results more thoroughly in Melvin et al. [35].

Initial shared memory experiments were run on the Mercury machine of University of Manitoba. Execution times for these experiments are shown in Figure 15 for numbers of threads ranging from 1 to 37. The parallelization scheme implemented requires that the number of threads be less than or equal to the number of projection angles, which in this case is 37. A clear advantage is seen for increasing numbers of threads. However we also note an apparent optimum at approximately 8-10 threads. By inspecting the timing profiles for these trials, we note that comparatively little time is spent in the OpenMP barrier and lock functions. Presumably, the ratio of threads to data chunks is particularly favorable for this particular situation, perhaps indicating that the implementation is not an optimal one for greater numbers of threads. The probable root cause is the hardware limitation on Mercury of having 8 processors, thereby forcing multiple threads to run on each processor for higher numbers of threads.

Figure 15: Run time versus number of threads for PART reconstruction of the breast phantom 256x256 on Mercury.



In order to overcome this limitation of the Mercury experiments, the PART code was ported to Nexus and Cortex of the WestGrid high powered computing consortium. Experiments run on the Nexus group showed positive results, however prohibitively long queuing time made extensive experimentation impractical. Results hereafter presented are run on the Cortex group, specifically the 64 processor machine Dendrite (see section 4.1.2.2.2).

According to Amdahl speedup is given by of the PART algorithm, defined as $\frac{T_{serial}}{T_{parallel}}$, is given by $\boldsymbol{S} = \frac{q \bullet n^3}{q \bullet \frac{n^3}{p}} = p$. This theoretical value will be reduced by memory access latency, communication time, and other overhead. A measure of the practical cost when implementing a parallel algorithm is the Efficiency, given by $\frac{Speedup}{p}$.

Table 4 shows runtimes, speedup, and efficiency for several numbers of processors on Dendrite, reconstructing to 30 iterations.

Table 4: Run times for PART to reconstruct Shepp and Logan phantom to 30 iterations on various numbers of processors. Experiments were run on Dentrite (Cortex group of WestGrid)

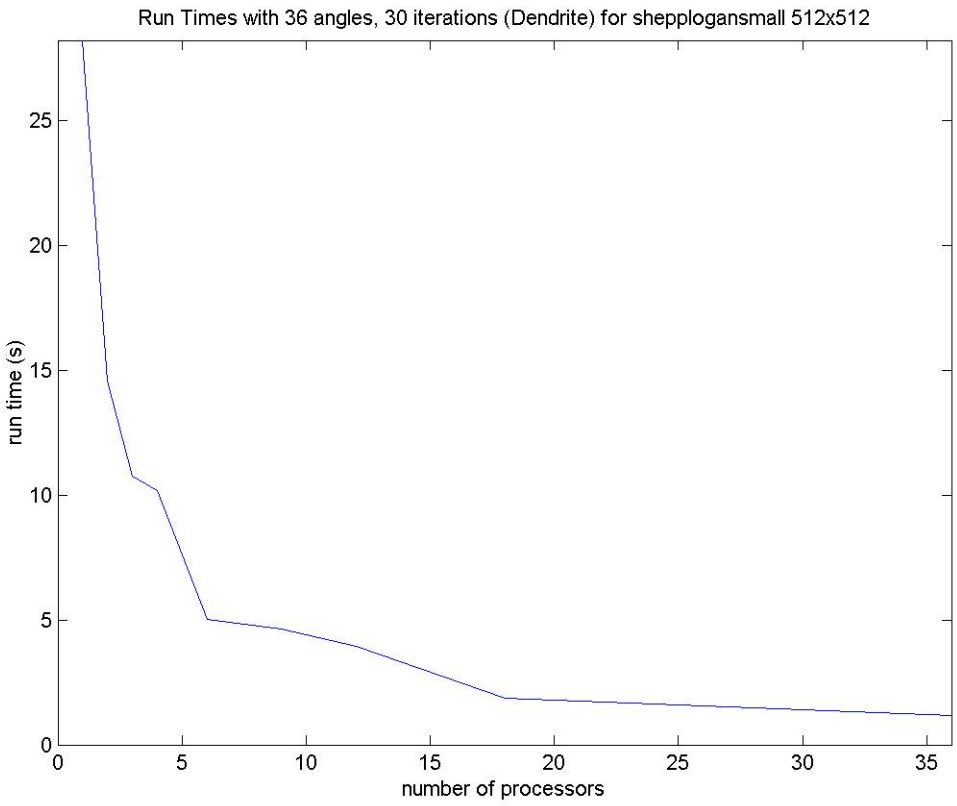
Run times for PART to 30 iterations

Number of Processors	Run time (seconds)	Speedup ($\frac{T_{serial}}{T_{parallel}}$)	Efficiency % (Speedup/Num Processors)
1 (serial)	28.218	1	100
2	14.583	1.935	96.750
3	10.775	2.6188	87.295
4	10.191	2.7689	69.223
6	5.038	5.601	93.351
9	4.657	6.0593	67.325
12	3.951	7.142	59.517
18	1.885	14.97	83.165
36	1.183	23.853	66.258

Run times are shown graphically in Figure 16, with speedup graphed in Figure 17. Images were reconstructed using 36 angle projection data. With a block-

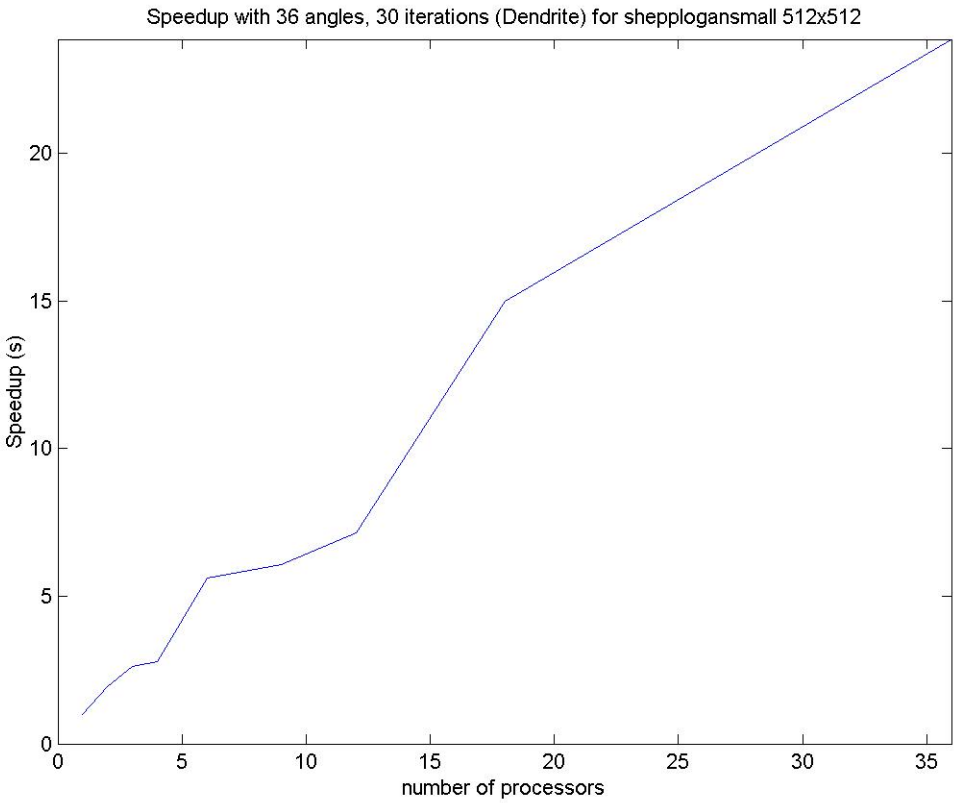
cyclic data distribution pattern, it makes sense to use numbers of processors that are factors of 36.

Figure 16: Run times for PART reconstructions on Dendrite. Shepp and Logan phantom 512x512 was reconstructed from 36 angles (30 iterations).



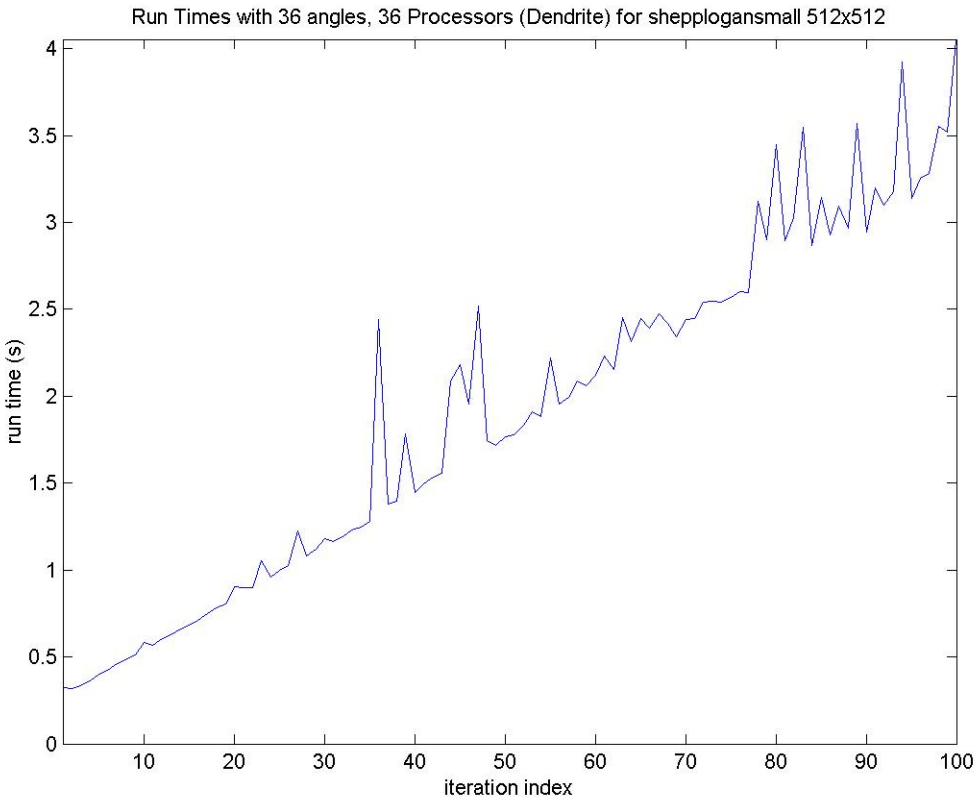
The speedup shows positive results, with efficiency ranging from 96.75% to 66.26%. Note that the efficiency for 6 processors is 93.35%. Since the most practical hardware for CT manufacturers will likely be 4 to 12 processors, 6 processors may prove to be a good choice in commercial applications, depending on the most common number of angles used when collecting data.

Figure 17: Speedup for PART reconstructions on Dendrite. Shepp and Logan phantom 512x512 was reconstructed from 36 angles (30 iterations).



For 36 processors, it takes 1.183 seconds to reach 30 iterations. The parallel technique presented here need only be on the same order of magnitude as the FBP to prove worthwhile, considering the image quality improvements evident in the literature and empirically shown in Section 5.2.3.

Figure 18: Run times to reach different numbers of iterations for PART algorithm reconstruction Shepp and Logon phantom on 36 processors.



In order to compare run times directly, a serial Fourier Backprojection implementation was coded in C for the Cortex P-servers. Table 5 shows the results for these experiments. The implementation required 2.0700 seconds to reconstruct the 512x512 Shepp and Logan image from 36 projection angles, averaged over 4 trials.

Table 5: Run times for a serial Fourier Backprojection image reconstruction from various angles. Data is averaged over 4 trials, compiled and run in 32 bit mode on Dendrite machine of WestGrid.

Run times for Fourier Backprojection	
Number of Angles	Average Run time (seconds)
36	2.0700
90	4.6107
180	8.9645

Using 36 processors, the parallel part took only 1.183 seconds to reach convergence on the same 36-angle data set that took the FBP implementation 2.0700 seconds to reconstruct. Considering that FBP reconstructions usually require on the order of 180 angles, it might be more apt to compare the FBP time for 180 angles to the PART reconstruction for 36 angles. In this case, the FBP takes 8.9645 seconds compared to the PART reconstruction at 1.183 seconds. The parallel ART implementation presented here speeds up computation of ART reconstructions to the same order of magnitude as the FBP, if not better.

5.2.3 Image Quality

In gauging the usefulness of any speedup to a reconstruction algorithm, it is necessary to quantitatively evaluate the image quality to ensure accurate image results. Mean Square Error (MSE) provides a general feel for the algebraic distance of each pixel to the ideal goal value, when comparing an output image to the original phantom. MSE is also useful in measuring convergence when

comparing output images from successive iterations. To avoid some of the variation seen for smaller images and fewer angles, the following image quality discussions refer to results using breast phantom of size 256 x 256 pixels, reconstructed from 37-angle projection data.

Figure 19 shows the MSE between the current estimate and the original phantom for the shared memory PART implementation. This quantity is calculated according to the following formula:

$$MSE = \frac{\sqrt{\sum_i \sum_j (O_{i,j} - C_{i,j})^2}}{N^2} \quad \textbf{(16)}$$

Where:

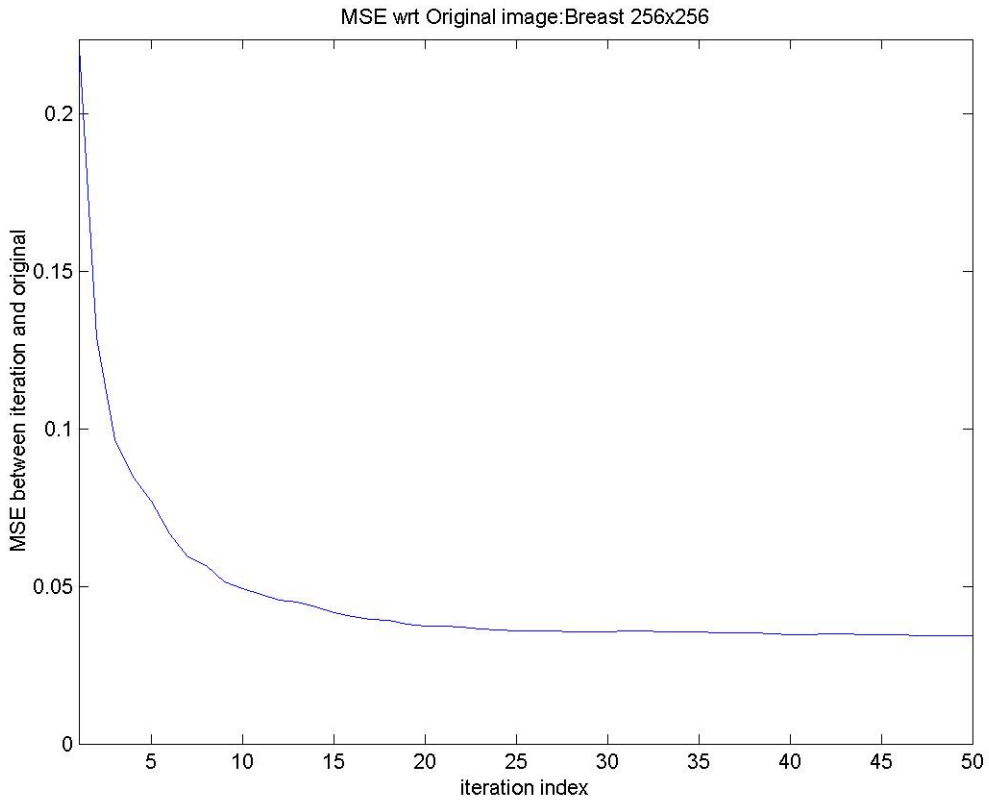
O = original image phantom

C = current image estimate

$\boldsymbol{N^2}$ = number of pixels in the image

The graph shows this mean squared error also approaching zero. The individual values show the MSE converging to 0.0344. The convergent solution then, is approaching the original phantom (the ideal solution) with acceptably small error.

Figure 19: MSE between iteration image and original phantom.



Another way to gauge the validity of the given solution is to compare the entropy of that solution to the entropy of the known ideal solution (i.e. the original breast phantom). This quantity is commonly measured by the following formula:

$$Entropy = \frac{\sqrt{\sum_i \sum_j \left(C_{i,j} \cdot \ln(C_{i,j})\right)}}{\sum_i \sum_j \left(C_{i,j}\right)} \qquad \textbf{(17)}$$

Where:

$\boldsymbol{C}$ = current image estimate

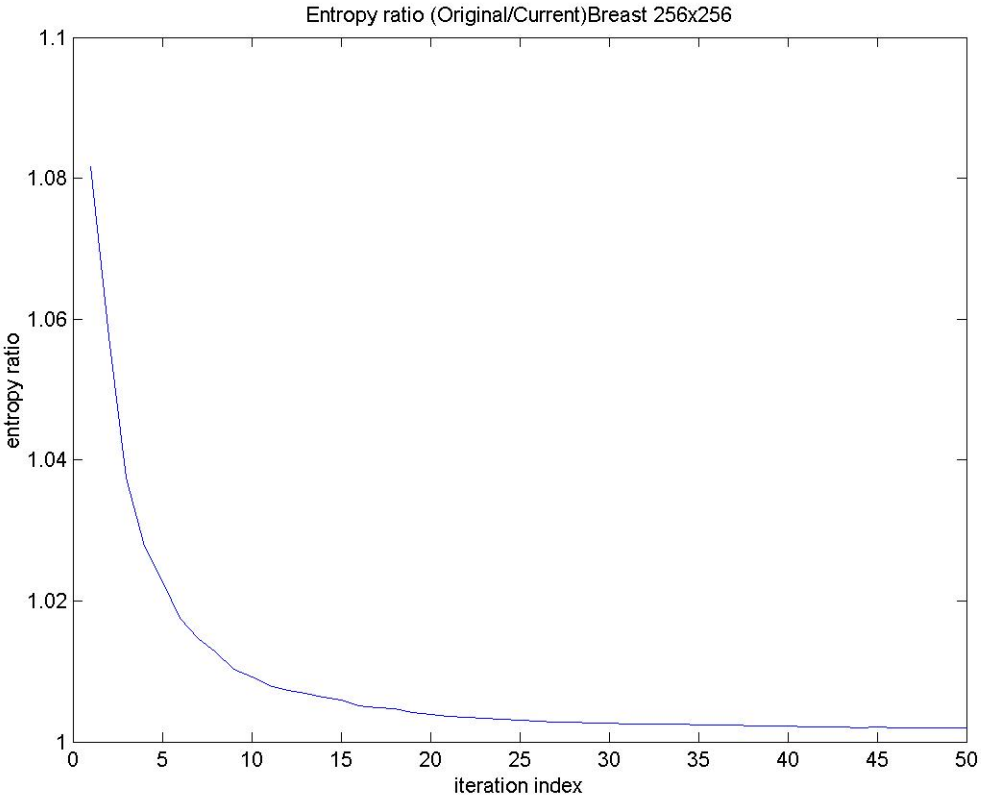
Entropy is usually interpreted as the amount of information present in a signal, or alternatively a measure of the randomness of a signal. In the context of image reconstructions, we can interpret entropy as one way to measure the valuable information contained in the image.

In order to effectively compare this measure with the original image, a ratio of the entropy of the original phantom to the reconstruction is taken, calculated according to the following formula:

$$EntropyRatio = \frac{Entropy_{original}}{Entropy_{reconstruction}} \quad \textbf{(18)}$$

Figure 20 shows a graph of the ratio of the entropy of the original image to that of the current estimate, vs. iterations. The ratio approaches the ideal 1, achieving a ratio of 1.0019 after 50 iterations.

Figure 20: Entropy ratio of original to iteration image.



The following images (Figure 22, Figure 23, and Figure 24) were reconstructed from the famous Shepp and Logan phantom, shown in Figure 21.

Figure 21: Original image: Shepp and Logan Phantom 512x512.

Figure 22: PART reconstruction after 5 iterations.

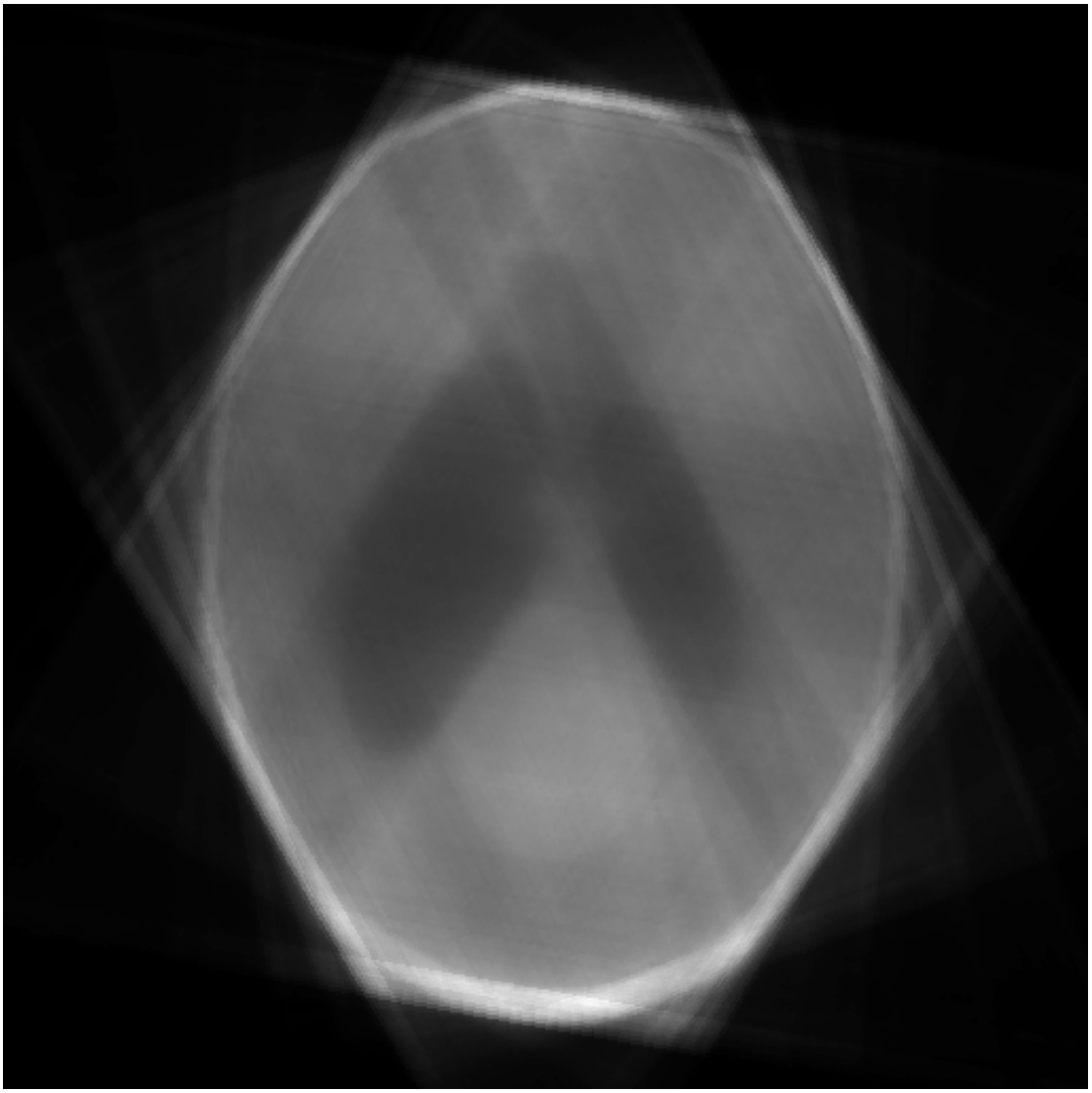


Figure 23: PART reconstruction after 10 iterations.

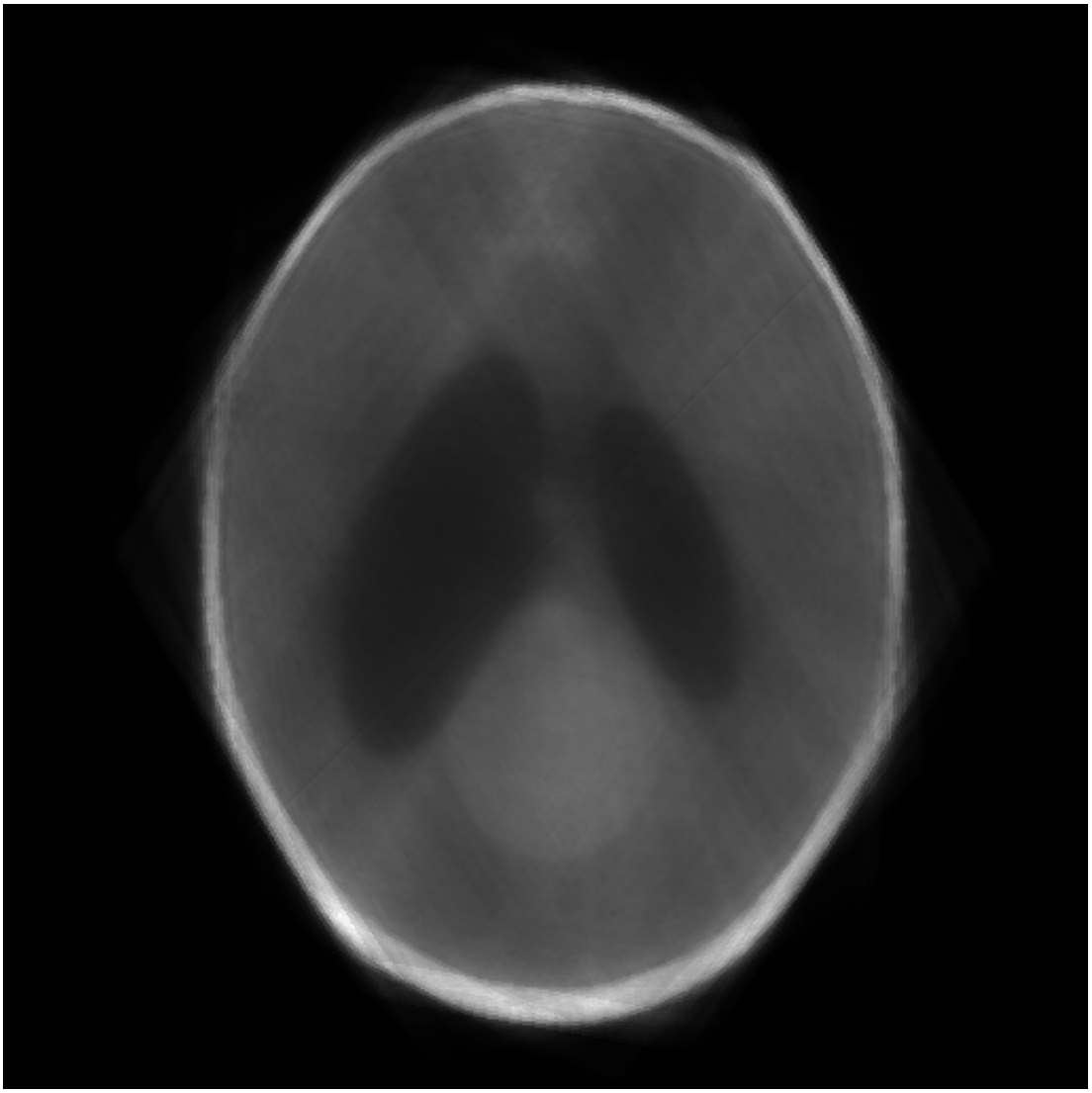
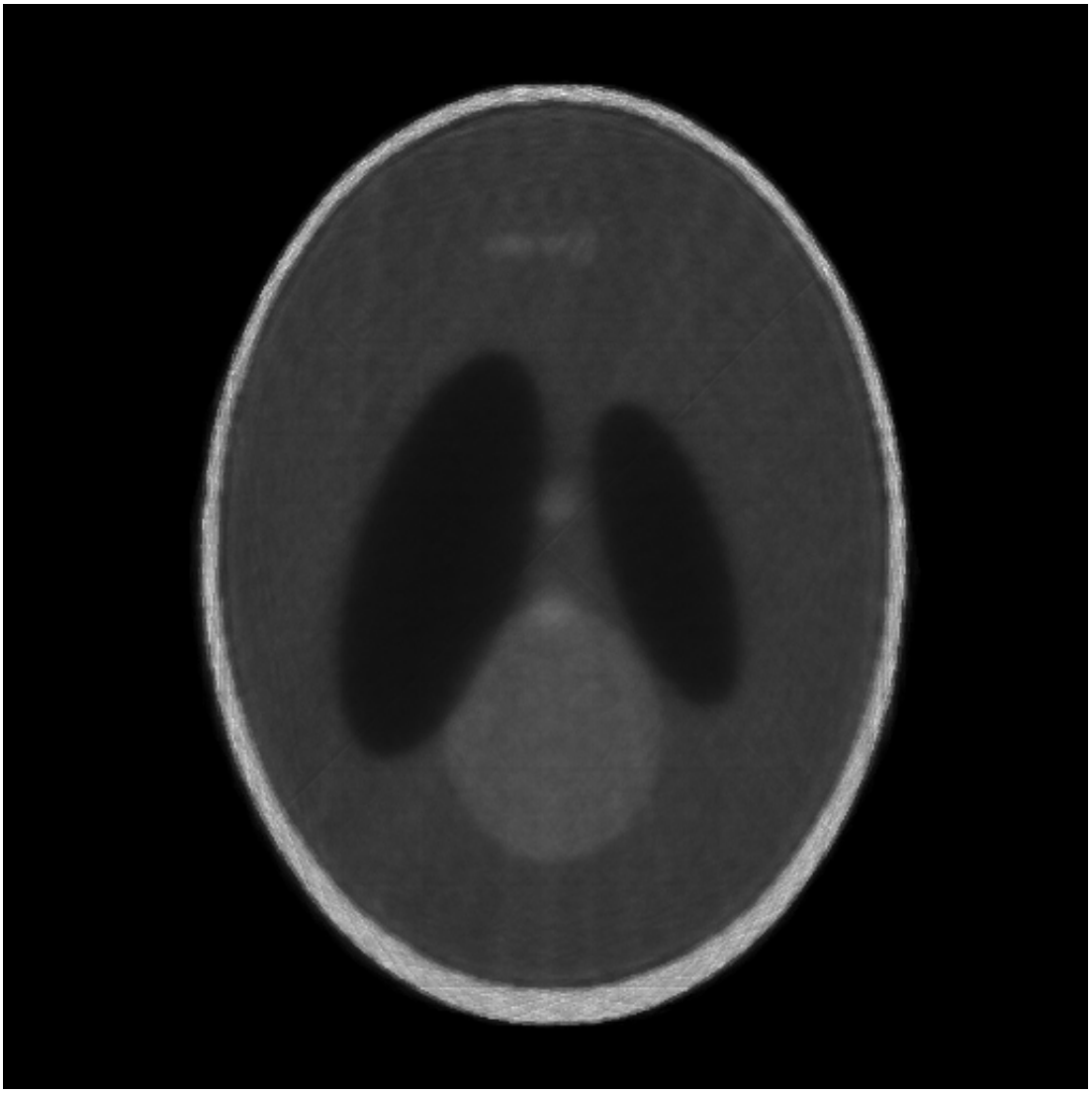


Figure 24: PART reconstruction after 50 iterations.



The streaks visible in the images are characteristic of CT images. These streaks are especially apparent in images reconstructed from fewer numbers of angles, such as these, reconstructed from 19 angles.

The MPI ART implementation is very similar to the shared memory implementation in the level of parallelism employed, and nearly identical in

image quality results. Because of the lack of computational advantage found with the MPI implementation [35], specific image quality results for that work will not be presented here.

Wei et al. [52] compared Fourier Backprojection to simple Backprojection techniques using the common error measurement, shown in Equation (19 below.

$$error = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\left(x_{i,j}' - x_{i,j}\right)^2}{\sum_{i=1}^{M}\sum_{j=1}^{M} x_{i,j}^{\ 2}}} \qquad \textbf{(19)}$$

where $x_{i,j}$ = original image phantom

$x_{i,j}'$ = current image estimate

M^2 = number of pixels in the image

This formula is slightly different than the error calculation used earlier in this paper. The problem with comparing to this value is that it is not adjusted for image size. The images used in the work of Wei et al. were 256x256, so we cannot compare directly to error for reconstructions of images of other sizes. However, if we use the modified Equation (20 below, we can adjust Wei et al.'s error values for comparison with other image sizes,

such as the 512x512 Shepp and Logan images used for some of the experiments presented in this thesis.

$$error = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\left(x_{i,j}'-x_{i,j}\right)^2}{M^2\sum_{i=1}^{M}\sum_{j=1}^{M}x_{i,j}^{\ 2}}} \qquad \textbf{(20)}$$

$$= \frac{1}{M}\sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\left(x_{i,j}'-x_{i,j}\right)^2}{\sum_{i=1}^{M}\sum_{j=1}^{M}x_{i,j}^{\ 2}}}$$

Wei et al., found that on average, FBP had an error of 0.1005 when reconstructing from 256 angles. Adjusting for the image size, this would be 0.3925 x 10^3. The recalculated error for PART reconstruction of the Shepp and Logan phantom using Equation (20 is 0.399 x 10^3. The error for these two reconstructions varies by only 1.63%, with the PART reconstruction using only 36 angles compared with the FBP's 256 angles. For similar MSE values, the PART required 85.9% fewer angles, or more generally 85.9% less dose.

Consistent with the literature, FBP reconstructed images showed poor image quality for low numbers of projections. Qualitatively, the PART reconstructed images are clearly higher in quality seen in Figure 25 and Figure 26.

Figure 25: PART Reconstruction of Shepp & Logan 512x512 from 37 angles.

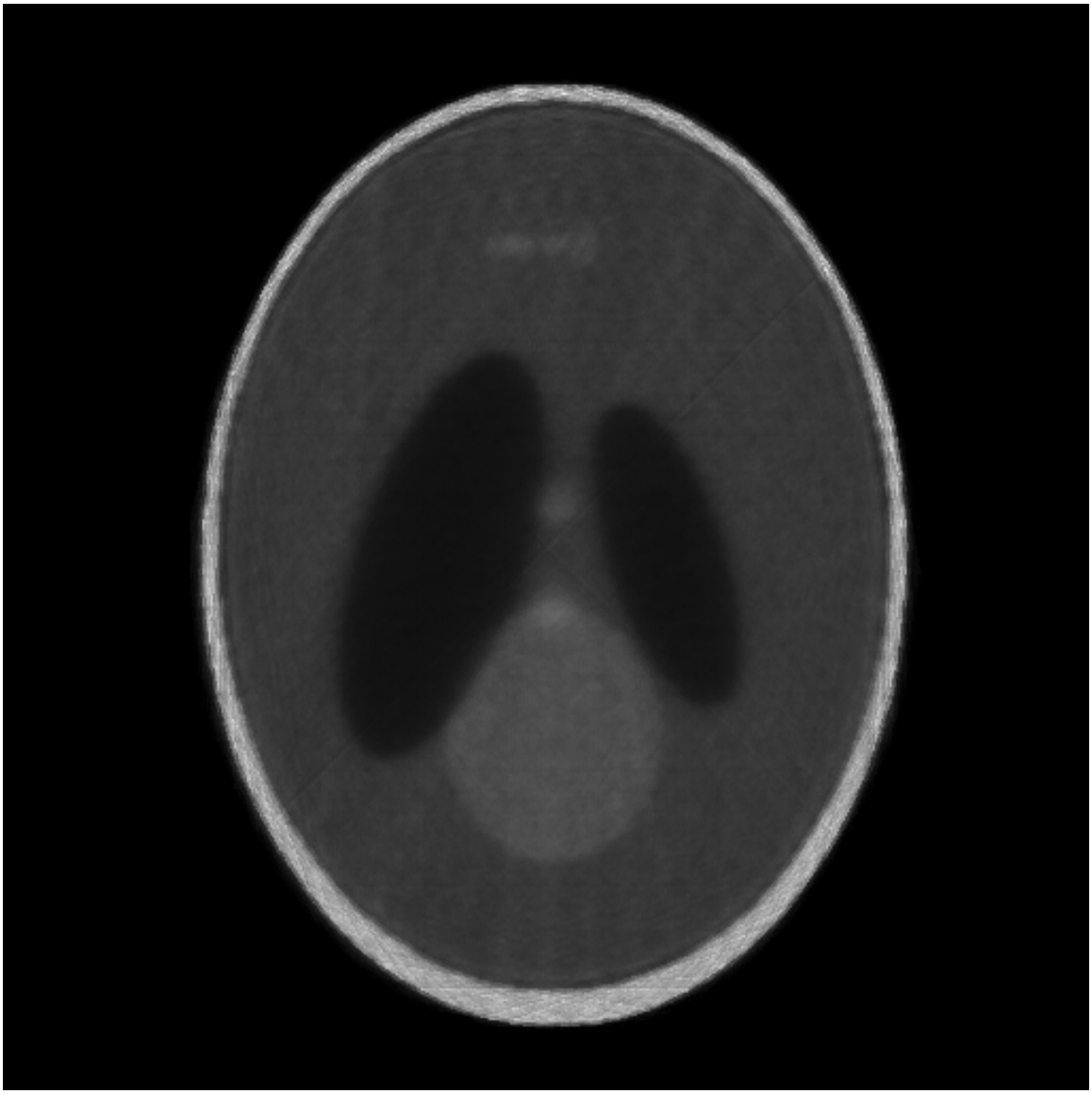
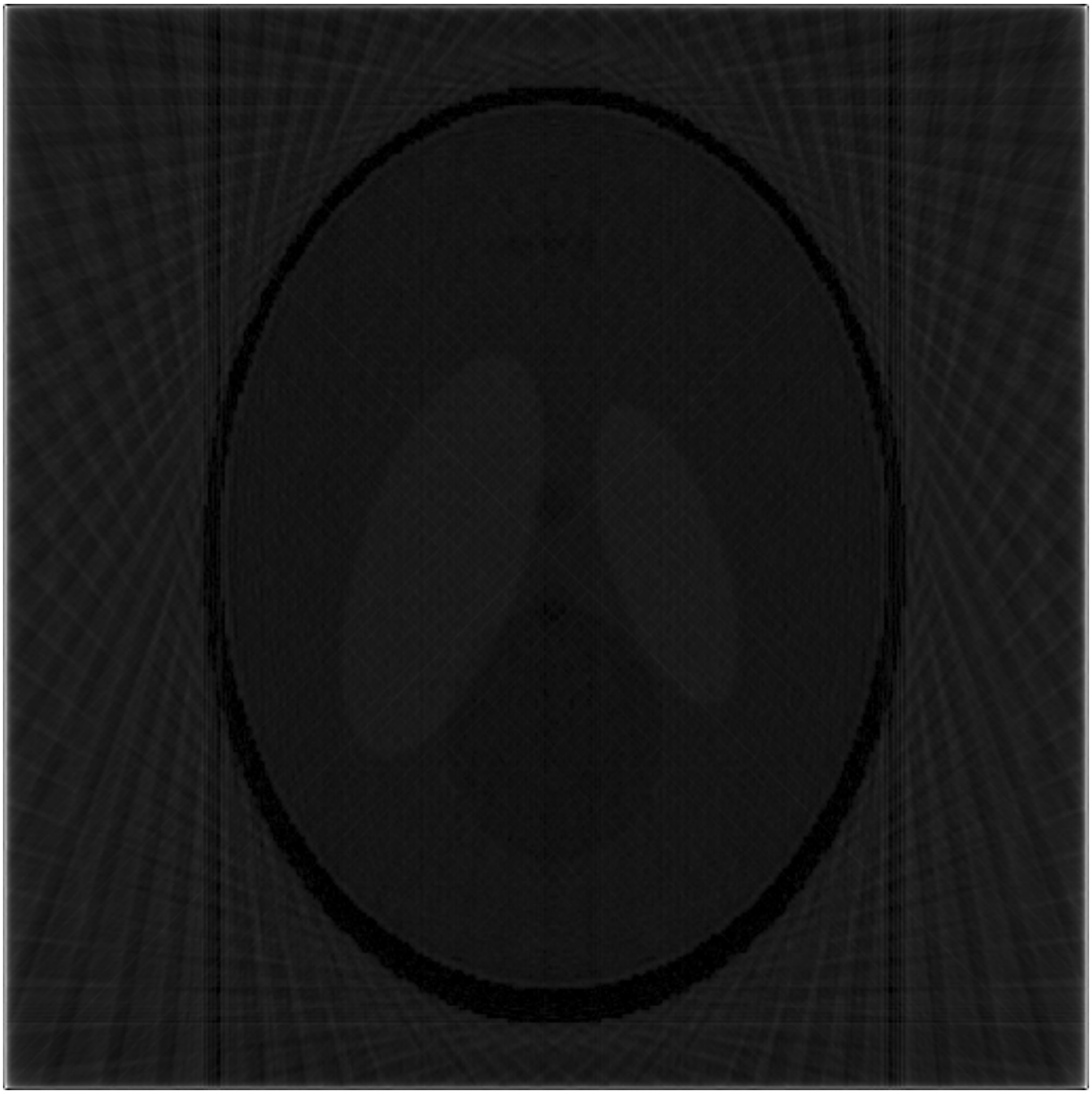


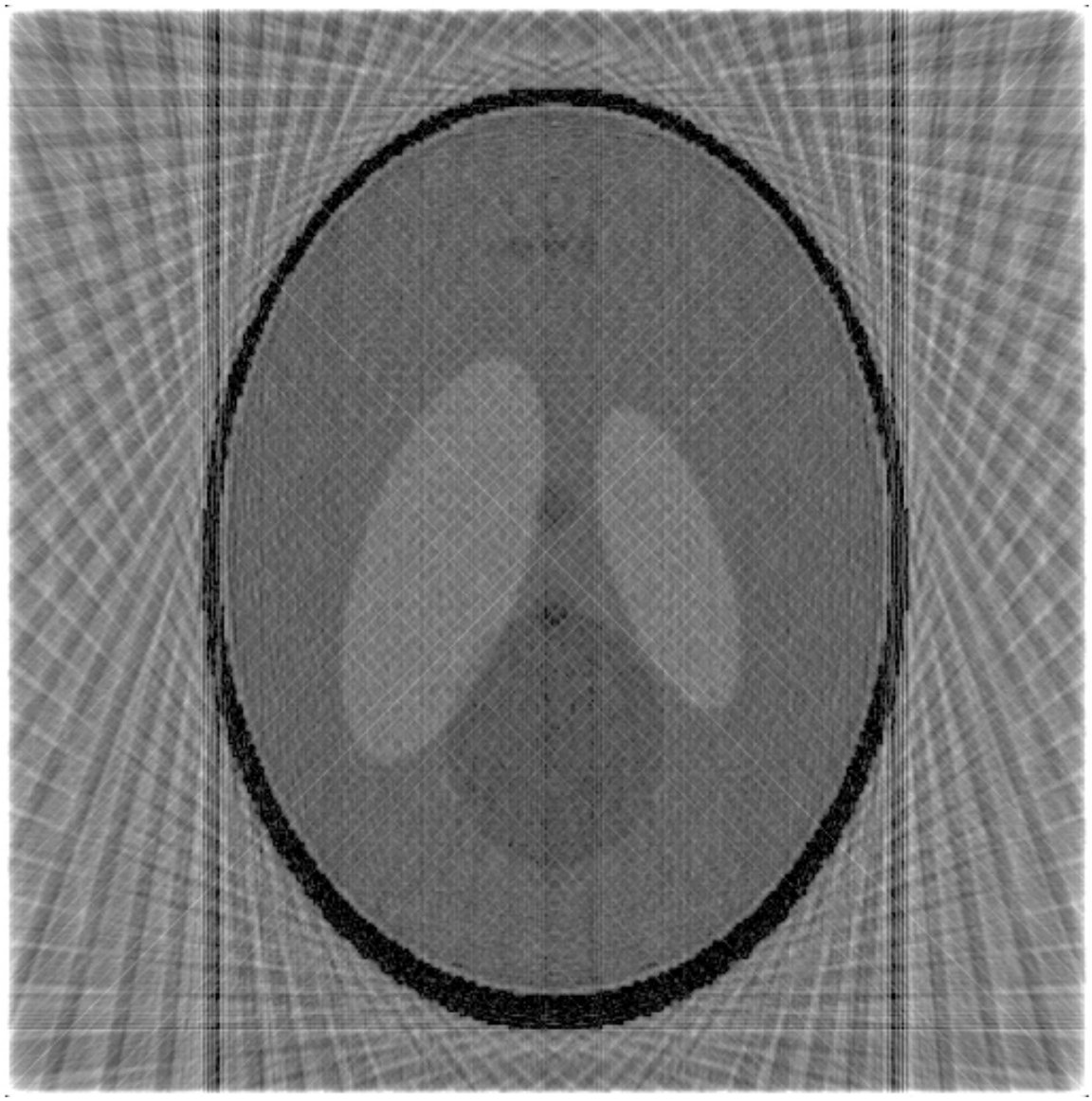
Figure 26: FBP reconstruction of Shepp & Logan 512x512 from 37 angles



With no post processing, the PART reconstructed image in Figure 25 is clearly superior. Although post processing is not within the scope of this paper, a simple window and level (brightness and contrast) adjustment is appropriate to compare these images on a more level playing field. Finding an optimal window and level adjustment is user specific, but a general improvement is achieved with

simple histogram equalization. This was achieved by using the image adjustment features of Adobe Photoshop 7.0. After using Photoshop's "auto levels" adjustment, the FBP image was not significantly improved and manual adjustment was required to see the "anatomical" structures of the phantom. The resulting FBP image is shown in Figure 27.

Figure 27: FBP reconstruction of Shepp & Logan 512x512 from 37 angles with manual window and level adjustment.



A similar adjustment was performed on the corresponding PART reconstruction, with a small amount of improvement (shown in Figure 28).

Figure 28: PART reconstruction of Shepp & Logan 512x512 from 37 angles with manual window and level adjustment.

Aside from the general aesthetic differences in the image, two key clinical problems with the FBP reconstruction of Figure 28 should be noted. First, the

outer rim of the major anatomical body, in this case representing the outside of the skull, is white in the original phantom and the PART reconstruction, whereas it is black in the FBP reconstruction. Another significant discrepancy is the gray level of the two larger elliptical bodies. In the original phantom and the PART reconstruction, these bodies are darker, specifically in relation to the large circular body. In the FBP reconstruction, these elliptical regions are a lighter shade of gray than the circle. This problem cannot be remedied with post processing and poses an important clinical problem, which is the reliable determination of the density (or more specifically X-ray opacity) of anatomical structures.

In quantitative terms, the MSE of the FBP image at 37 angles is 0.1352, while the MSE of the PART image is 0.057076.

The entropy ratio measured for the FBP image was 1.3923, while the PART reconstructed image was measured at 1.1138. As previously discussed, the ideal value for this measure is 1, which would denote that the reconstructed image contains approximately the same amount of information as the original. This result reinforces the already existing body of literature, showing better image quality for the PART reconstruction for limited projections when compared to FBP.

5.2.4 Problem Size

Images commonly reconstructed in a clinical setting are on the order of 128 by 128 pixels to 512 by 512 pixels. Images of size up to 1024 by 1024 pixels were reconstructed successfully using the PART technique. Attempts to reconstruct larger images, specifically 2048 by 2048 pixel images, were initially unsuccessful. Images of this size require 32 megabytes to store as an array of data type ***double***. Original code versions used several arrays of this size, allocated statically. For images of this size, the PART program with static memory allocation requires stack space on the order of 128 megabytes.

In C, statically allocated arrays are defined in the declaration section, for example:

```
double    current[IMAGEWIDTH][IMAGEWIDTH];
```

The memory allocated for variables declared in this way is located on the stack space of the program. This limited memory space (8 megabytes on Mecury by default) cannot handle variables of the size required for large images. Dynamic memory allocation circumvents this issue.

Dynamically allocated variables are stored on the heap instead of the stack. In C, these variables are declared as pointers and physically allocated by calling a memory allocation function such as ***malloc(memsize)***.

The code below shows how this might be done for an image array:

```
double     *seed;
seed = (double *)malloc(sizeof(double)*IMAGEWIDTH*IMAGEWIDTH);
```

Declaring, allocating, and manipulating dynamic variables is more difficult programmatically, but provides performance advantages in addition to the memory management advantages for large images.

6 Conclusion

The proposed message passing parallel algorithm presents encouraging theoretical results in terms of speedup. However, empirical data was inconclusive. Erratic data collected implies that the communication latency in the message passing implementation presented in this paper is far too variable to achieve reliable results.

In addition to the prohibitive communication latency of the message passing approach, most commercial CT machines are equipped with dedicated computational power, and occasionally custom architectures. As a result, a shared memory implementation would be more practical than a message passing implementation in industry. Timing results of the shared memory implementation show a clear advantage with greater numbers of threads. Speedup increased steadily for increasing numbers of processors, with efficiency ranging from 59.52% to 96.75%. The number of processors should be a factor of the total number of angles for optimal speedup for maximum efficiency.

In more practical terms, a 6 processor IBM P-server could reconstruct the same image from 36 angles in approximately 5.038 seconds, with an efficiency of 93.35%. In other words, a PART reconstruction could be done in about the same amount of time as a 180 angle serial FBP reconstruction, yielding

approximately equivalent image quality, but at an 80% reduction in dose. Alternatively, for the same dose, the 5.038 second PART reconstruction produces a higher quality image in approximately the same amount of time. Such an off-the-the shelf IBM system is valued in the range of $30,000-$50,000. When replacing a manufacturer's existing reconstruction hardware, this would account for about 1 to 2% of the $2M cost of a modern CT system.

7 Future Work

Before being commercially used, the PART algorithm should go through some additional experimentation using real CT data instead of the simulated data used in this thesis. Such experiments would further validate the results presented, and might expose some yet unforeseen flaws in the approach. In addition, the code would need to be ported to whatever hardware platform the reconstructions would be done on. With the strict adherence to ANSI standards, porting the code to any platform should be relatively straight forward.

Commercial CT machines vary in the method by which they gather data. Several newer machines in the market, such as Toshiba's newest Aquillion models, use a helical cone beam acquisition scheme. As discussed earlier, preprocessing is often done in commercial CT machines to conform the machine's native acquisition data to a data form suitable for the reconstruction algorithm being employed. For example, helical cone beam data is sometimes interpolated or re-projected to produce sequential slice data before reconstruction. The most common native algorithm in use for this type of data is the Fourier backprojection-based code introduced by Feldkamp et al. [16]. Rival cone beam algorithms include block-ART and ART-based approaches. The parallel ART introduced in this thesis could be extended to a cone beam case, to process native helical cone-beam data without any preprocessing.

Further in the future, single photon X-ray and CT machines may be available [34]. Current CT machines use X-ray tubes producing X-ray photons in a random fashion, characterized by a Poisson distribution. When single photon X-ray emitters are available, a new class of CT algorithms will be needed. An ART based algorithm for this application might easily be parallelized based on the work in this thesis.

8 References

[1] Cho, Z. H. "General View on 3-D Reconstruction and Computerized Transverse Axial Tomograpny", Special Issue on Physical and Computational Aspects of 3-Dimensional Image Reconstruction, IEEE Transactions on Nuclear Science, V. 21 no. 3. June 1974. pp 44-71.

[2] L. Lin, V. K. Jain. "Parallel Architectures for computing the Hough Transform and CT Image Reconstruction", *International Conference on Application Specific Array Processors, 1994 Proceedings*. San Francisco CA.

[3] J. Fitchett. "A Locally Synchronous Globally Asynchronous Vertex-8 Processing Element for Image Reconstruction on a Mesh". M. Sc. Thesis, University of Manitoba. 1993.

[4] E. Shieh, K. Current, P. Hurst, I. Agi. "High Speed Computation of the Radon Transform and Backprojection Using an Expandable Multiprocessor Architecture". IEEE Transactions on Circuits and Systems for Video Technology 6, Vol. 2 no. 4. 1992.

[5] S. Coric, M. Leeser, E. Miller, M. Trespanier. "Parallel Beam Backprojection: An FPGA Implementation Optimized for Medical Imaging". Tenth ACM International Symposium on Field-Programable Gate Arrayes (FPGA02), February, 2002. pp 217-226.

[6] D. Lattard, G. Mazare. "Image Reconstruction Using an Original Asynchronous Cellular Array", IEEE International Symposium on Circuits and Systems, 1989. pp 13-16.

[7] D. Lattard, B. Faure, G. Mazare. "Massively Parallel Architecture: Application to Neural Net Emulation and Image Reconstruction". Proceedings of the International Conference on Application Specific Array Processors, 1990. pp 214-225.

[8] E. Artzy. 1979. "Display of three-dimensional information in computed tomography". Computer Graphics and Image Processing 9, pgs 196-198.

[9] E. Artzy, G. Frieder, G. T. Herman, H. K. Liu. "A system for three-dimensional dynamic display of organs from computed tomograms". Proceedings of the 6th Conference on Computer Applications in Radiology & Computer Aided Analysis of Radiological Images. Newport Beach, CA. June, 1979. pgs. 285-290.

[10] E. Artzy, G. Frieder, G. T. Herman. "The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm". Computer Graphics and Image Processing. January, 1981. pgs 1-24.

[11] M. R. Stytz, G. Frieder, O. Frieder. "Three-Dimensional Medical Imaging: Algorithms and Computer Systems". ACM Computing Surveys, Vol. 23 No. 4. 1991.

[12] C. Guerrini, G. Spaletta. "An image reconstruction algorithm in tomography: A version of the CRAY X-MP vector computer". Computers and Graphics 13:367-372, 1989

[13] C. Chen, S. Lee, Z. Cho. A parallel implementation of 3D CT image reconstruction on a hypercube multiprocessor. IEEE Transactions on Nuclear Science. 37(3):1333 – 1346, 1990.

[14] R. Rao, D. Kriz, A. Abbott, Ribbens, C. Parallel Implementation of the Filtered Back Projection Algorithm for Tomographic Imaging. 1995, revised April 5, 1995. Retrieved July 11, 2006 from http://www.sv.vt.edu/xray_ct/parallel/Parallel_CT.html. 1995

[15] D. Reimann, V. Chaudhary, M. Flynn, I. Sethi. Parallel Implementation of cone beam tomography. International Converence on Parallel Processing. Bloomingdale, IL. 1996. pp 1-4.

[16] L. Feldkamp, L. Davis, J. Kress. Practical cone beam algorithm. Journal of the Optical Society of America, 1984. 1(6). pp 612-619.

[17] C. Laurent, F. Peyrin, J-M. Chassery, M. Amiel. Parallel image reconstruction on MIMD coputers for 3D cone beam tomography. Parallel Computing. 1998. 24. pp 1461-1479.

[18] S. Smallen, W. Cirne, J. Frey, F. Berman, R. Wolski, M-H Su, C. Kesselman, S. Young, and M. Ellisman. Combining workstations and

supercomputers to support grid applications: The parallel tomography experience. Heterogenous Computing Workshop, Cancun, Mexico. 2000. pp. 241-252.

[19] K. Lange, R. Carson. EM reconstruction algorithms for emission and transmission tomography. Journal of Computer Assisted Tomography. 8(2). 1984. pp 302-316.

[20] J. Rockmore, A. Macovski. A maximum likelihood approach to image reconstruction. Proceedings of the Joint Automatic Control Conference. 1977. pp 782-786.

[21] L. Shepp, Y. Valdi. Maximum likelihood reconstruction for emission tomography. IEEE Transactions on Medical Imaging. 1982. MI-1 pp 113-122.

[22] M. Attkins, D. Murray, and R. Harrop. Use of transputers in 3-D positron emission tomography. IEEE Transactions on Medical Imaging. 1991. 10(3).

[23] C. Chen. An efficient four-connected parallel system for PET image reconstruction. Parallel Computing. 1998. 24. pp 1499-1522.

[24] Z. Cho, C. Chen, S-Y. Lee. Incremental algorithm – a new fast backprojection scheme for parallel beam geometries. IEEE Transaction on Medical Imaging. 1990. 9(2). pp 207-217.

[25] X. Li, T. He, S. Wang, G. Wang, J. Ni. P2P-enhanced distributed computing in medical image EM reconstruction. H. Arabnia, editor, 2004 International Conference on Parallel and Distribute Processing Techniques and Applications (PDPTA '04), volume 2. Las Vegas, Nevada. 2004. pp 822-828.

[26] R. Rangayyan, A. Dhawan, R. Gordon. Algorithms for limited view computed tomography: A survey. IEEE Transactions on Medical Imaging. MI-1(3). 1982. pp 173-178.

[27] G. Wang, D. Snyder. J. O'Sullivan, M. Vannier. Iterative debluring for CT metal artifact reduction. IEEE Transactions on Medical Imaging. 1996. 15. pp 657-664.

[28] D. Robertson. J. Yuan, G. Wang, M. Vannier. Total hip prosthesis metal artifact suppression using iterative deblurring reconstruction. Journal of computer Assisted Tomography. 1997. 21(2). pp 293-298.

[29] G. Herman, L. Meyer. Algebraic reconstruction can be made computationally efficient. IEEE Transactions on Medical Imaging. 1993. 12(3). Pp 600-609.

[30] M. Jiang, G. Wang. Convergence of the simultaneous algebraic reconstruction technique (SART). IEEE Transactions on Image Processing. 2003. 12(8). pp 957-961.

[31] M. Jiang, G. Wang. Convergence studies on iterative algorithms for image reconstruction. IEEE Transactions on Medical Imaging. 2003. 22(5). Pp 569-579.

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

[33] B. Carvalho, G. Herman. Helical CT Reconstruction from Wide Cone-Beam Angle Data Using ART. IEEE Proceedings of the XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'03). 2003.

[34] C. Melvin, K. Abdel-Hadi, S. Cenzano, R. Gordon. A simulated comparison of turnstile and Poisson photons for X-ay imaging. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'02). Winnipeg, Manitoba. 2002. pp 1165-1170.

[35] C. Melvin, P. Thulasiriman, R. Gordon. Parallel Algebraic Reconstruction Technique for Computed Tomography. Proceedings of Parallel and Distributed Processing Techniques and Applications 2003 (PDPTA'03). Las Vegas, Nevada.

[36] R. Leahy C. Byrne. "Recent developments in iterative image reconstruction in PET and SPECT" IEEE Transactions in Medical Imaging. 2000. 19257–60.

[37] W. Kalender. "Computed Tomography: Fundamentals, System Technology, Image Quality, Applications"

[38] R. Gordon, R. Bender, G. T. Herman, "Algebraic reconstruction techniques (ART) for three dimensional electron microscopy and x-ray photography," in Journal of Theoretical Biology, 29(3), 471-481, 1970.

[39] R. Gordon, "A Tutorial on ART (Algebraic Reconstruction Technique)," in IEEE Transactions on Nuclear Science, Vol. NS-21, June 1974.

[40] R. M. Rangayyan and R. Gordon, "Streak Preventive Image Reconstruction with ART and Adaptive Filtering," in IEEE Transactions on Medical Imaging, Vol. MI-1, No. 3, November 1982.

[41] R. Gordon, G. T. Herman, S. A. Johnson, "Image Reconstruction From Projections," in Scientific American, October 1975.

[42] R. Gordon and G.T. Herman, "Reconstruction of Pictures from Their Projections," Graphics and Image Processing, Vol.14, No. 12, December 1971.

[43] H. Guan and R. Gordon, "A projection access order for speedy convergence of ART: a multilevel scheme for computed tomography," Physics in Medicine and Biology, Vol. 39, 1994.

[44] K. Mueller, R. Yagel, J. F. Cornhill "The Weighted Distance Scheme: A Globally Optimized projection Ordering Method for ART," Transactions on Medical Imaging, Vol. 16, No. 2, April 1997.

[45] K. Mueller, R. Yagel, J. F. Cornhill "OpenMP C and C++ Application Programming Interface," OpenMP Architecture Review Board, Version 2.0, March 2002.

[46] M. Misiti, Y. Misiti, G Oppenheim, J. Poggi. "Wavelet Toolbox for Matlab (User's Guide)," Mathworks Inc., Natick MA http://www.mathworks.com/access/helpdesk/help/pdf_doc/wavelet/wavelet_ug.pdf, pg 550

[47] A. Kak, M. Slaney "Principals of Computerized Tomographic Imaging". IEEE Press, New York NY. 1988.

[48] "Education: Digital X-ray: Image Quality Parameters for Digital Detector" General Electric Company, New York NY. 2003. http://www.gehealthcare.com/inen/rad/xr/education/dig_xray_iq.html

[49] Matteo Frigo and Steven G. Johnson, "The Design and Implementation of FFTW3," *Proceedings of the IEEE* **93** (2), 216–231 (2005). Invited paper, Special Issue on Program Generation, Optimization, and Platform Adaptatio

[50] H. Stark, J. W. Woods, I. Paul, and R. Hingorani, "Direct Fourier reconstruction in computer tomography," IEEE Trans. Acoust. Speech Signal Processing, vol. ASSP-29, pp. 237-244, 1981

[51] C. Curtis, F Razo, L Rae Sande. *SGI Origin 3000 Series Owner's Guide*. Silicon Graphics Inc. Mountain View CA. 2001.

[52] Y. Wei, G. Wang, J. Hsieh. "Relation between Filtered Backprojection Algorithm and the Backprojection Algorithm in CT," IEEE Signal Processing Letters, vol 12, NO 9. pp. 633-636. 2005

[53] S. Basu, Y. Bresler. "$O(N^2 \log_2 N)$ Filtered Back Projection Reconstruction Algorithm for Tomography". IEEE Transactions on Image Processing, vol 9 No 10. pp 1760-1773. 2000.

[54] C. Costantini, J. Wood. "IBM Eserver p5 590 and 595 Technical Overview and Introduction". International business Machines. Austin TX. 2005.