Local Independence in Computed Tomography as a Basis for Parallel Computing

A thesis presented
by

Dr. Daniel Morris Martin

to
The Department of Computer Science
in partial fulfillment of the requirements
for the degree of
Master of Science
in the subject of

Computer Science

The University of Manitoba
Winnipeg, Manitoba
August 2007

Thesis advisors

Dr. Parimala Thulasiraman

Dr. Richard Gordon

Author

Dr. Daniel Morris Martin

Local Independence in Computed Tomography as a Basis for Parallel Computing

Abstract

Iterative reconstruction algorithms are superior to the standard convolution backpropagation (CBP) methods when reconstructing from a small number of views (hence less radiation), but are computationally costly. To reduce the execution time, this work implements and tests a parallel approach to iterative algorithms using a cluster of workstations, which is a low cost system found in many offices and non-academic sites. A previous implementation showed little speedup because of the significant cost of inter-processor communication. In this thesis, several data partitioning methods are examined, including some image tiling methods that exploit the spatial locality demonstrated by *local CT*. Using these methods, computation can proceed locally, without the need for inter-processor communication during every iteration. A relative speedup of up to 17 times is obtained using 25 processors, demonstrating that good performance can be obtained running computationally intensive CT reconstruction algorithms on distributed memory hardware.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

First of all I would like to thank my wife Loretta and my daughter Rebecca for their understanding during the many hours I spent on this work. They have made many sacrifices to make this thesis possible.

Thanks to my advisors, Professor Parimala Thulasiraman and Professor Richard Gordon, for their patience and helpful suggestions during a rather long thesis project.

Thanks to Professor Jason Morrison for his help in editing this work and for giving me some well needed lessons on academic writing.

Thanks to Lynne Romuld for guiding me through the maze of prerequisites and deadlines for undergraduate courses.

Thanks to the NSERC for their generous financial support that made graduate studies possible.

Thanks to Professor Willi Kalender for the use of his excellent explanatory figures in my background chapter.

Thanks to Dr. Kevin Rosenberg for the use of his open source CT Simulator *CT Sim* used to generate the input data, saving me many hours of work.

Last, but certainly not least, I would like to thank the many excellent professors in the Computer Science Department who have taught me over several years. In particular, I thank Professor John Bate for sparking my interest in a more academic pursuit of computer science, and grounding me in the skills that I required.

This thesis is dedicated to my mother Margaret, who died from cancer in the prime of her life.

It is my hope that this work and similar efforts will spare someone of a similar fate from radiation exposure.

Chapter 1

Introduction

This chapter introduces the concept of Computed Tomography (CT), discusses its place relative to plain film X-ray technology, and the relevance of CT to modern medical imaging. It then discusses an important problem that motivates this thesis: exposure to ionizing radiation. The remainder of the chapter outlines the approach of the thesis in addressing this problem.

1.1 Introduction to Computed Tomography

The goal of medical imaging is to determine the internal structure of an organism with sufficient detail to yield diagnostic information. Imaging strives to achieve this in the least invasive manner possible, minimizing discomfort and harm to the patient.

Two dimensional plain film X-ray pictures have been the standard medical imaging technique for a century, and remain a common technique today. The patient is placed between an X-ray source and a photographic plate (film) or more recently a digital detector array. An X-ray exposure of sufficient intensity and duration is used to project shadows of body tissues onto the detecting surface. The X-ray ‘beam’ is attenuated by scattering and absorption of the intervening tissue proportional to the distance the beam must traverse the tissue, the density of the tissue, and the atomic numbers of the contained elements. A given detector in the detector plane records the intensity of the beam after this attenuation. The resulting two dimensional projection reveals structure from which a radiologist, using knowledge of the anatomy, and possibly other projections in different orientations, can infer detail of the three dimensional subject.

This section discusses the advantages of CT over the standard plain film technology from which it is derived. It then discusses the preeminent place of CT in X-ray diagnostics, and then finally discusses the correlation of CT images to the actual pathology.

1.1.1 Advantages of CT over Plain Film

X-ray projections have at least three limitations: limited projection angles from which the X-ray view can be taken; inability to localize the 3-D position of a structure; and, most importantly, a lack of detail due to lack of contrast. The limitation of viewing angles is imposed by the target object and the imaging equipment. Figure 1.1 (a) shows a computed tomography (CT) slice, while Figure 1.1 (b) shows a plain film lateral view. Plain film X-rays are the result of X-ray beams striking the film close to perpendicular to the plane of the image. In the case of Image (b) the X-ray source is on the right side of the patient’s head and projects X-rays onto a film on the left side. A craniocaudal (head to toe) projection similar to Image (a) on the left of Figure 1.1 is difficult if not impossible to achieve with standard X-ray films. To direct X-ray beams perpendicular to a cross section of the head, the patient might stand on a film with the source directed from the top of the patient's head, and would thus also include the torso and legs in the view. In the case of CT, the slices are obtained by X-ray beams projected at various angles *within* the plane of the image. If the X-ray source and image cannot be placed in the plane of the desired image, then obtainable slices from other orientations can be assembled into a 3-D structure. From this virtual 3-D representation, voxels from any given plane can be sampled and displayed. Thus *volumetric imaging* allows a subject to be viewed from any desired orientation. Alternately, virtual 3-D surfaces can be thresholded and displayed by assigning a virtual light source to illuminate these surfaces with chosen colours and textures in a 3-D *surface rendering* program. A useful example of this rendering is *virtual colonoscopy* [87], a CT examination in which views of the interior surface of the colon are reconstructed to show the colon much as it would appear in a colonoscope.

The second limitation of plain film projections is that a structure in a two dimensional image often cannot be located in three dimensions. An object may be visualized, but its location in the axis along the X-ray beam is not known. The radiologist may infer an object's position using other images projected from a different angle, assuming the object can be identified from other angles.

A common occurrence, which demonstrates this problem, is the "nipple shadow" in a PA (posteroanterior, back to front) or AP (front to back) chest X-ray. A circular

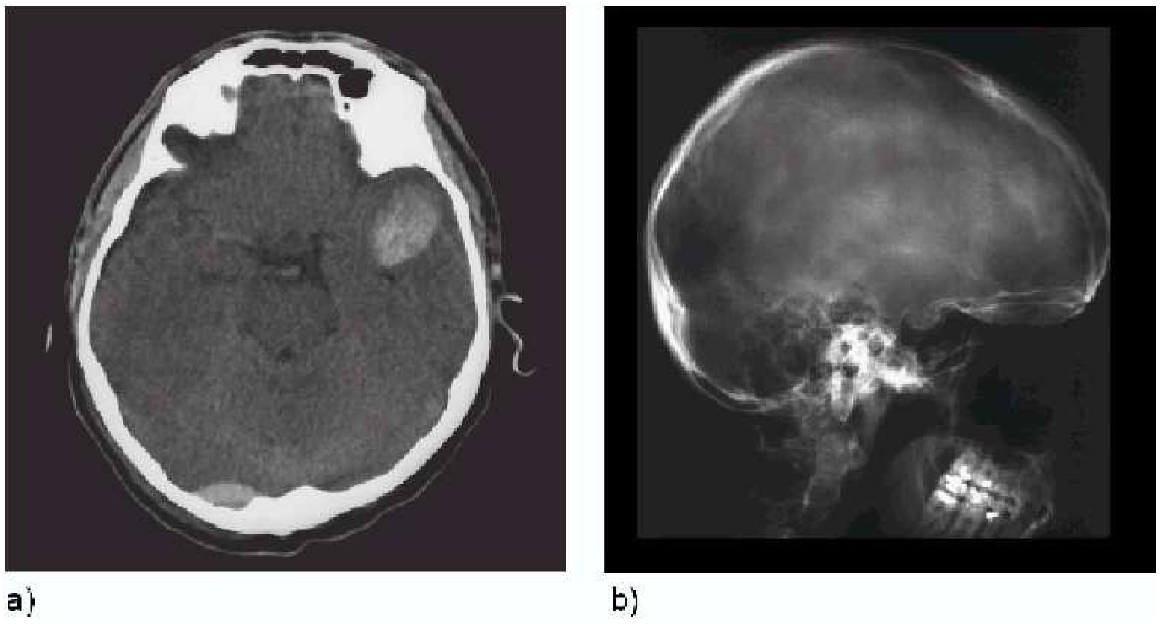


Figure 1.1: Improved contrast for CT compared to plain film. From Prof. W. Kalender [46, page 21], with permission June 18, 2007.

lesion may be visible on the X-ray, projected over the lung. The absence of such a lesion on a lateral view is somewhat reassuring, however a mass on this view could be obscured. CT is usually not needed in such an instance. A repeat X-ray with a paper-clip taped to the chest, its tip touching the periphery of the nipple, is sufficient to demonstrate that the round lesion is indeed a nipple (if the mass is *not* adjacent to the paper-clip then CT *is* needed).

The chief disadvantage of plain film is the lack of detail, resulting from the lack of contrast between adjacent tissues. An object that spans only a fraction of the distance along the axis of the beam changes the detector reading very little. The detector measures the sum of all tissue attenuation along the beam; a thin tissue or object contributes little to this overall sum, and may not be seen apart from the background in the image.

In the CT of Figure 1.1(a), a tumour is clearly visible, but this tumour is not seen on the plain film on Figure 1.1(b). The skull has a very high attenuation, hence a minor increase in skull thickness contributes more to a measured sum of attenuations than does the tumour. The end result is that minor variations in skull thickness obscure the tumour.

One technique to overcome this limitation is conventional tomography [89, page 226]. This form of tomography utilizes motion in the X-ray source and detector plane to blur parts of the subject that are not of interest. A plane of interest in the subject is relatively motionless with respect to the X-ray source and detectors, and detail in this region appears against a uniformly blurred background. Only a small number of planes can be visualized in one examination, and the image contrast is generally less than that of computed tomography. This technology has been largely replaced by CT.

1.1.2 The Advent of CT

CT is based on the same physical principle as plain film X-ray: an X-ray source sends a beam through the subject to a detector, which detects an intensity inversely related to the integral of attenuation of tissues along the beam. However, CT utilizes detector readings from X-rays taken at many projection angles to compute the attenuation of individual volume elements (voxels) inside the subject. Typically a number of parallel pencil beams or thin fan beams create a one-dimensional projection of each slice. A sufficient number of these projection measurements from a sufficient number of projection angles allows for calculation of sufficiently small areas (pixels) to render a two dimensional CT slice of the subject.

CT is based on the mathematical work of J. Radon in 1917, who demonstrated his assertion that “a 2-D or 3-D object could be reproduced from an infinite set of all projections” [68]. Radon’s work is a generalization of the work published by Abel [1] in 1826. Basically, CT attempts to calculate the contributing attenuations of individual voxels by utilizing projection data from numerous angles and applying an algorithm to the data (hence the ‘computed’ in ‘computed tomography’).

The problem of 3-D location of a structure is then easily solved by reconstructing a 3-D representation based on voxel attenuations. Views from ‘impossible’ angles are reconstructed based on projections from numerous ‘possible’ angles, most of these at right angles to the apparent axis of the final image. Most importantly, improved contrast from an estimate of individual pixel (or voxel) attenuation yields an image with superior detail. This contrast and detail is what makes CT a valuable diagnostic test.

1.1.3 Correlation of CT Images to Pathology

The contrast and detail of the anatomy demonstrated by CT suggest that CT may be an excellent diagnostic test. However, the images reconstructed should not be presumed to accurately reflect all pathologies. CT’s real worth as a diagnostic test must be proven in clinical situations.

A simple binary classification, the presence or absence of a pathological state, is often used to discuss the value of diagnostic tests. The test in question is compared to a gold standard; CT is compared to surgical or pathological findings or other gold standard tests. Test results fall into one of four categories: correct identification of the pathological state (true positives, TP), correct exclusion of the pathological state (true negatives, TN), incorrect identification of the pathological state (false positives, FP), and failure to identify the pathological state (false negatives, FN). Several measures of a diagnostic test can then be calculated [2].

$$sensitivity = TP/(TP + FN) \tag{1.1}$$

$$specificity = TN/(TN + FP) \tag{1.2}$$

$$positive\ predictive\ value = TP/(TP + FP) \tag{1.3}$$

$$negative\ predictive\ value = TN/(TN + FN) \tag{1.4}$$

Sensitivity is the ability of the test to detect the condition. High sensitivity of a test is useful to exclude a condition; if a sensitive test has not detected the condition, then we can be relatively certain that the patient truly does not have the condition. Specificity is the ability of the test to not 'over call' the condition. High specificity of a test is useful to confirm a condition; if a specific test is positive, then we can be relatively certain that the patient truly does have the condition. Positive predictive value is the likelihood that a positive test really is positive. Negative predictive value is the likelihood that a negative test really is negative.

Sensitivity and specificity are properties of the test, determined experimentally. The predictive values, however, depend not only on the diagnostic test, but also on the prevalence of the condition in the population studied. If the test is not a perfect predictor, then a decrease in prevalence results in more false positives and fewer true positives, resulting in a lower positive predictive value. Similarly, a decrease in prevalence results in a higher negative predictive value.

The value of a diagnostic test depends not only on these measures, but also on such features as financial cost and risk to the patient. The predictive values, cost, and risk change according the role that the diagnostic test plays. A screening test must exhibit high sensitivity, low finacial cost, and low risk to the patient. Although CT has some risk and significant cost, and is not useful for screening normal populations, it is becoming a more common means of risk stratification for coronary artery disease [8]. Coronary artery calcification can be observed without the use of contrast material, so in this case the risk and cost of CT may be less than the alternative tool, coronary angiography.

CT is now the standard diagnostic test in many clinical situations. Although it has been declared a gold standard preoperative diagnostic test for appendicitis [64; 63], CT continues to be reevaluated and compared to other diagnostic tests. A study as recent as 2007 [44] shows a sensitivity of 91% and a specificity of 94% for diagnosing appendicitis. This specificity, along with a high prevalence of the disease in the population studied (patients presenting to the emergency department with 'acute abdomen') leads to a high positive predictive value. The high sensitivity is valuable in excluding the disease.

Using a complex imaging tool such as CT, there is not a simple binary classification yielded by the test (*i.e.*, appendicitis or no appendicitis). There are a number of radiologic signs including fat stranding, enlarged unopacified appendix, adenopathy, paracolic gutter fluid, multiple signs involving various patterns of wall thickening of the caecum, ileum or sigmoid colon, and other signs. Each of these signs has its own sensitivity and specificity for acute appendicitis [71]. Noting the specific signs involved for a particular case may alter the certainty of the diagnosis for that particular case.

A useful feature of an imaging test such as CT is that it may lead to an alternate diagnosis. A CT performed to either confirm appendicitis or rule it out may be worthwhile for that diagnosis alone, given the high sensitivity and specificity of CT for appendicitis. However, in the scenario of the acute abdomen, CT has additional value because it may lead to other diagnoses that require emergency treatment [22].

Another consideration in the use of CT is the importance of knowing what is 'normal'. Many findings can exist in tissue that are either normal variants or somewhere in a continuous spectrum between health and disease. Dalal and Hansell [16] note that while there is considerable literature on the signs of disease for high-resolution computed tomography of the lungs, little is documented on the normal findings.

It is important to recognize the limits of CT. Many tumours, *e.g.* pancreatic cancer metastasized to the surface of the liver and peritoneum, may be too small to appreciate on CT [26]. Li *et al.* [53] review cases of lung cancer missed in a low-dose helical CT screening program. They separate these false negatives into detection errors (mainly smaller tumours obscured by other anatomical structures) and interpretation errors (the lesion is seen but interpreted as benign disease). It is noteworthy that a human observer/interpreter is an integral part of the diagnostic test in the case of CT. Studies such as this are important to evaluate CT in a screening role.

CT is often used to exclude pulmonary emboli. The literature review conducted by Krishnan and Segal [21] revealed that there were "a wide range of reported sensitivities, only a minority of which exceeded 90%." This is disconcerting to a clinician who wishes to rule out the condition, since a negative CT cannot completely exclude pulmonary emboli, and other tests used individually are less accurate than CT.

On the whole, CT is extremely useful in a wide variety of clinical situations. As with any diagnostic test, its value is increased when it is used appropriately and its limitations are understood.

1.2 Motivation

The advantages of CT come at a cost. Multiple exposures in quick succession are necessary, with the position of the source and detectors precisely known. Precise movements of heavy equipment and X-ray tubes that take multiple exposures present engineering challenges [47, page 43] that limit portability and increase financial cost. The concern of this thesis, however, is the increased X-ray dose to the patient created by these multiple exposures. One concern is the rising dose to the population from the increasing prevalence of CT. A recent survey in Japan [65] indicated an examination rate of about 290 per 1000 population. Another concern is the relatively large X-ray dose of plain film compared to CT, causing the X-ray doseage from CT to be disproportionate. Plain films continue to suffice for many diagnostic procedures, with CT accounting for only about 6% of X-ray investigations. In spite of this, CT accounted for 47% of X-ray exposure in Germany in 2003 [47, page 155]. The [estimated] figures are very similar for the UK in 2001/2002 [39], with CT again accounting for 47% of X-ray exposure. Mettler *et al.* [58] estimate that

> "In United States hospitals with over 100 beds, CT scanning probably represents about 10% of all radiology procedures and about 67% of the

> total effective dose. More CT scans are done on males than females. About 11% of all CT scans are done on children".

This exposure poses an increased risk of cancer development, which is especially significant in infants. Estimates are a "one in 2000 risk of induction of fatal carcinoma from CT of the abdomen" [41]. This risk is higher in infants; as noted by Don [20]

> "a 1-year old undergoing an abdominal CT has a risk of 1 in 550 of developing a fatal malignancy, and a 1-year old undergoing a head CT has a risk of 1 in 1500. The younger the age of exposure, the greater the risk because of the latency period needed to induce the malignancy and the greater the effective dose."

X-ray exposure from CT poses significant risk to the patient.

The challenge facing CT is to reduce the radiation exposure while maintaining a quality of image that yields good diagnostic information. The problem and some solutions were discussed by Gordon as early as 1976 [31]. Reducing the radiation exposure can be done by decreasing the duration or intensity of each exposure, or, decreasing the number of exposures in each examination. Rather than attempting to optimize radiation dosage per projection or improve the resolution or sensitivity of detector arrays, this thesis focuses on the consequences of decreasing the number of required X-ray views. Such a reduction is possible by using 3-D cone beam reconstruction (discussed in section 2.1.3), or by decreasing the number of exposures in each examination and compensating with a robust algorithm [31, page 508] (discussed in section 2.1.4).

The standard algorithms perform poorly at 3-D reconstruction and at limited view reconstruction, so the *iterative* algorithms are used. For either limited view reconstruction or true 3-D reconstruction, *iterative* algorithms are favoured over the standard algorithms.

Because these algorithms require more execution time, a parallel version of the Algebraic Reconstruction Technique (ART), the historical and logical forerunner of the iterative algorithms, is implemented on a distributed memory system. Such machines, which include the common cluster of workstations (COW), are ubiquitous, and their hardware components can be as low cost and commonplace as desktop personal computers (PCs) connected by Ethernet. Implementation on a distributed memory system requires software to pass messages between processors, and the Message Passing Interface (MPI) software to do this is easily available.

Others implement parallel versions of ART or similar algorithms on distributed memory systems, and Melvin [56; 55] implements a parallel version of ART on a cluster very similar to ours. A recurring problem in these works is that improvements in speed over the sequential versions are very limited due to the cost of communication between processors. The processing of data on a given processor may also require data from a different processor. Such data dependencies are common to many problems, and the data must be partitioned among the processors in a way that minimizes data dependencies between processors.

This thesis examines six different data partitioning techniques including four unique methods that exploit spatial locality to reduce communication overhead. Communication between processors is limited to every second iteration or less, and two of the six methods maintain reasonable image quality while improving speed over the sequential algorithm.

The motivating factor for this thesis is to limit the radiation exposure from CT. The problem addressed is the communication overhead of parallel iterative CT reconstruction algorithms implemented on distributed memory systems. This thesis shows that given appropriate partitioning, the data in given locale is relatively independent of other data.

1.3 Thesis Overview

Computed Tomography has a vital role in medical diagnostics as an imaging method that yields detailed information. As an X-ray technology, however, it exposes the patient to ionizing radiation that is known to be harmful. A challenge for this technology is to obtain the high quality images that have come to be expected from it, while limiting this harmful radiation.

CT uses multiple X-ray views of the target for image reconstruction. Each view is associated with a dose of radiation, hence limiting the number of views will reduce the radiation. Using a true 3-D reconstruction from 2-D views, rather than assembling from 2-D reconstructions of 1-D views, should theoretically reduce the number of views required. A second approach is to limit the number of views outright.

Modifications of the commonly used convolution algorithms allow for some limited reconstruction in the third dimension, but these algorithms are not ideal for a full 3-D reconstruction. Limiting the number of views outright causes the standard reconstruction algorithms to fail. The *Algebraic Reconstruction Technique* (ART) and similar *iterative* algorithms yield better quality reconstructions using limited views or a true 3-D reconstruction, but these algorithms are much more costly in execution time and memory.

We can ameliorate this cost in execution time and memory by running the algorithm in parallel. In this thesis, a parallel version of ART is implemented on a Cluster of Workstations (COW). To implement this successfully and gain performance, we must overcome the problem of data dependence; data utilized in one local memory is dependent upon data in other nodes, and the work required to communicate this data between nodes nullifies the gains from parallelization in many implementations.

This thesis confronts the problem of data dependence and communication cost by examining partitioning methods. By separating the data and mapping it to processing entities (PEs) in such a way that data within a given PE is more interdependent than data between PEs, two of the six partitioning methods examined contribute to improved performance. The result is that interprocessor communication is kept to a minimum and doesn't detract significantly from the benefits of parallelization. Significant speed benefits are obtained compared to the sequential version of the ART algorithm. One of the partitioning methods also shows good scalability: as more processors are added, the speed continues to improve.

The iterative algorithms may have a role to play in limited view CT reconstruction or in 3-D CT reconstruction, and should not be rejected out of hand because of speed or memory limitations. These limitations are overcome significantly by implementing the algorithms in parallel, if communication is limited and an appropriate partitioning scheme is used.

The remainder of this thesis is organized into four chapters. Chapter 2 introduces the technology of CT and the reconstruction algorithms upon which CT is based.

Chapter 2 also introduces two methods to reduce radiation exposure. It then discusses some of the literature related to the iterative algorithms, to implementing CT reconstruction in parallel, and introduces the problem of communication overhead in CT reconstruction in parallel. Chapter 3 discusses the materials and methods used in the experimental work, including programming languages, simulator software and phantoms, hardware, the parallel algorithm, and the data partitioning models used. Chapter 4 presents the experimental results, comparing the partitioning methods by image quality, speed, scalability to the number of processors, and scalability to problem size. Finally, Chapter 5 presents some conclusions and discusses future work arising out of this thesis.

Chapter 2

Background

This chapter gives some background regarding CT: the hardware and its evolution, the reconstruction algorithms, and two modalities that may limit X-ray exposure from CT. It then presents some related work regarding improvements to iterative reconstruction algorithms, parallel approaches to CT reconstruction, and the problem of communication overhead for the parallel approaches.

2.1 Computed Tomography

CT is a two step process of collecting the projection data, then calculating the attenuation values that could have generated these projection values (reconstruction). This thesis first discusses the geometry of the scanners (the source and detector arrangements and the beam geometries) then the reconstruction algorithms. Two modalities that limit the radiation from CT are then presented: 3-D cone beam reconstruction, and limited view CT.

2.1.1 Collecting the Data: CT Scanner Geometry

CT has evolved through five generations, each having different geometries for projecting the X-ray beam onto the detector. The 1^{st} generation machines scan using a single pencil beam, projected in a line from the source to a single detector (Figure 2.1, 1970). The source and detector are translated in the same direction, with exposures taken at regular intervals, to produce a set of measurements taken from a set of parallel beams. These beams or *rays*, projected from a single angle, form one *view*. The entire apparatus is rotated about the object and the translation process repeated, to get another set of parallel projections (another view) as in Figure 2.2 (1^{st} generation). Many views are used to compute a 2-D slice in the plane common to all of the views. The apparatus or the target object is then moved, and the entire process repeated to obtain further slices. This is obviously time consuming, but has the advantage that all rays in a given projection are parallel, which somewhat simplifies the image reconstruction.

Second generation scanners employ narrow fan beams (Figure 2.1, 1978) so that a single source can project onto a small linear array of detectors simultaneously. As in the 1^{st} generation machines, the source and the set of detectors are translated a number of times before each rotation. Fewer translations allow for a significant improvement in speed, while the slight variation from truly parallel beams is insignificant in reconstruction (Figure 2.2, 2^{nd} generation).

Modern day third generation scanners employ a longer array of detectors (usually in an arc). A wide angle fan beam strikes the detectors in the arc simultaneously, and the source and detector array are rotated continuously about the target object.

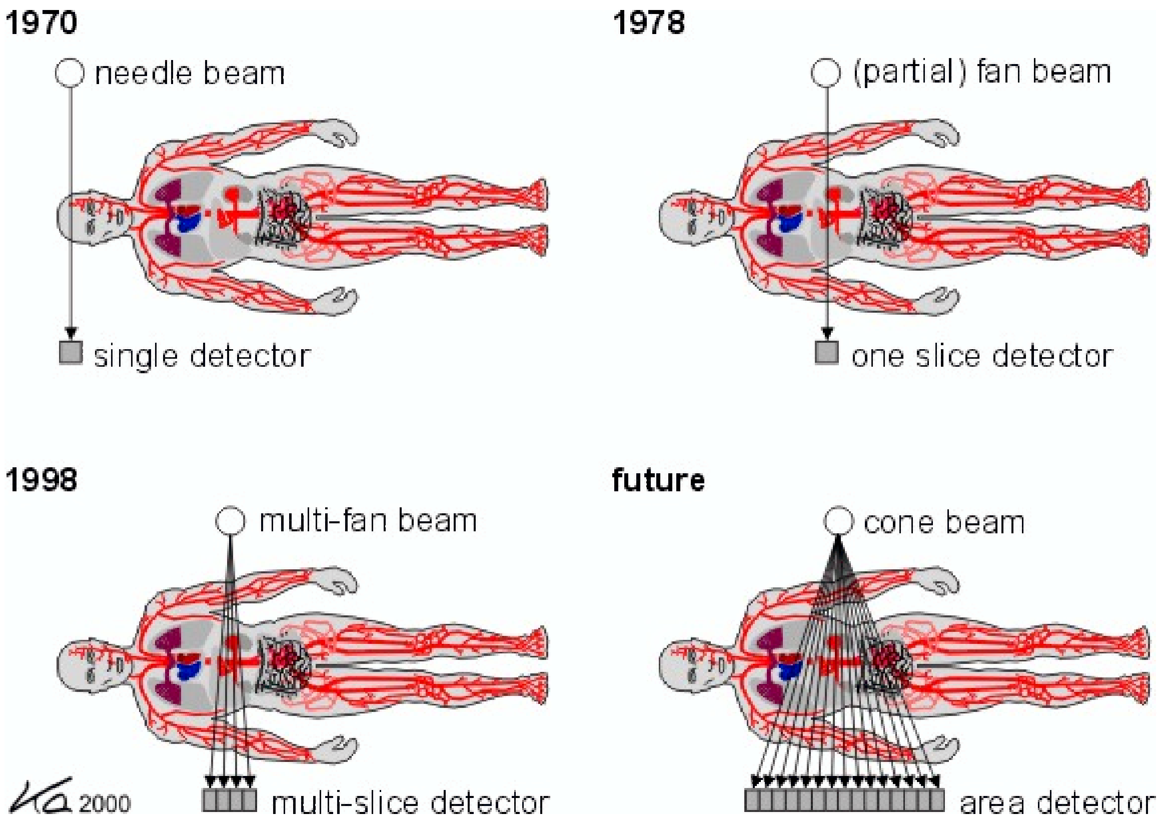


Figure 2.1: Beam geometries. From Prof. W. Kalender [46, page 177], with permission June 18, 2007.

If the fan beam is sufficiently widened, as in Figure 2.2 (3^{rd} generation), translation is no longer necessary at all.

Fourth generation scanners (Figure 2.2, 4^{th} generation) are similar to third generation scanners, but have a continuous stationary circle of detectors. These have not gained favour over the third generation scanners.

Original rotary systems have to be rotated back after each rotation to unwind power cables. Subsequent *slip ring* technology allows for continuous rotation for third and fourth generation scanners, with further improvement in speed.

Fifth generation *electron beam* CT scanners [89, page 329] have a stationary X-ray

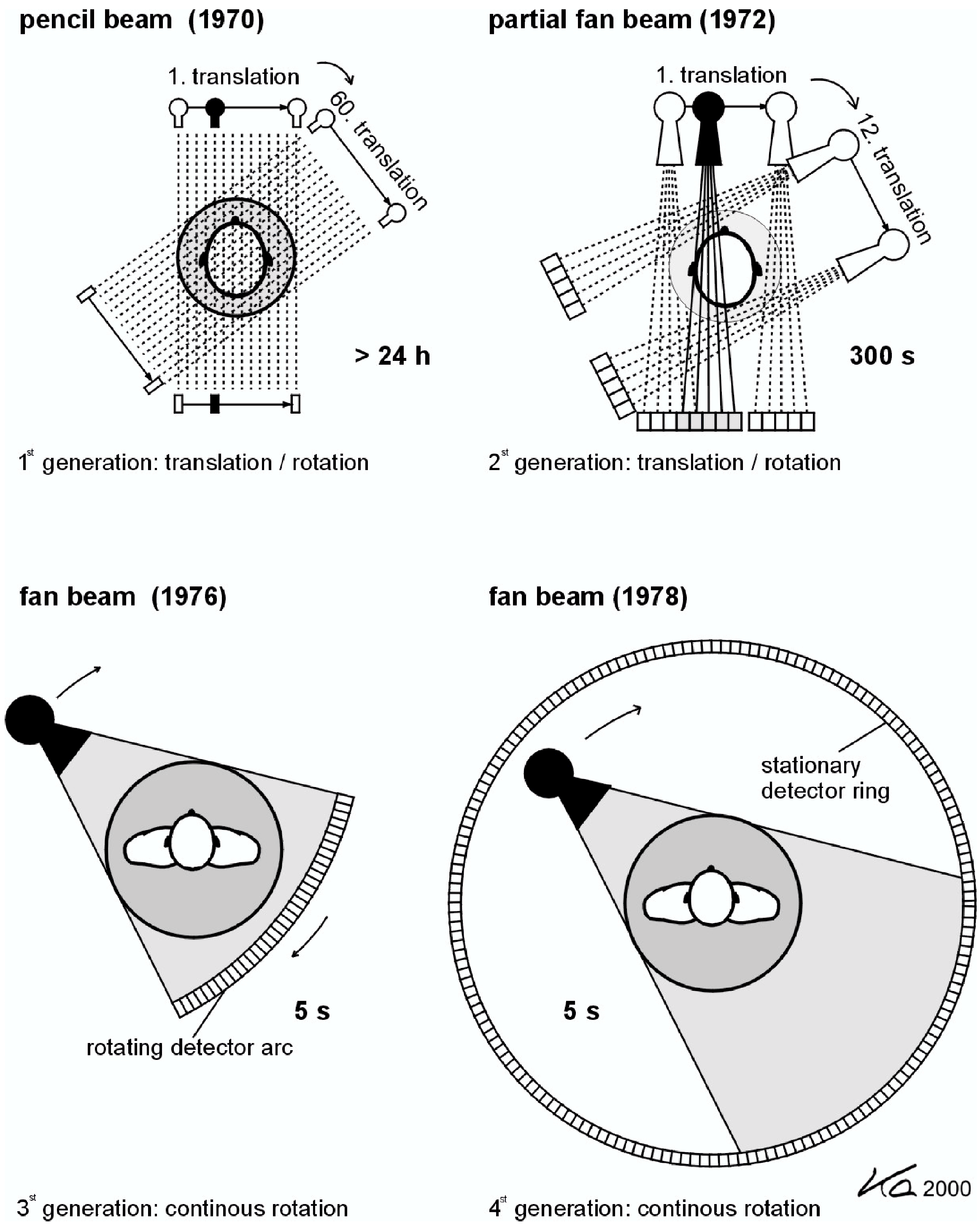


Figure 2.2: Generations of CT Scanner. From Prof. W. Kalender [46, page 36], with permission June 18, 2007.

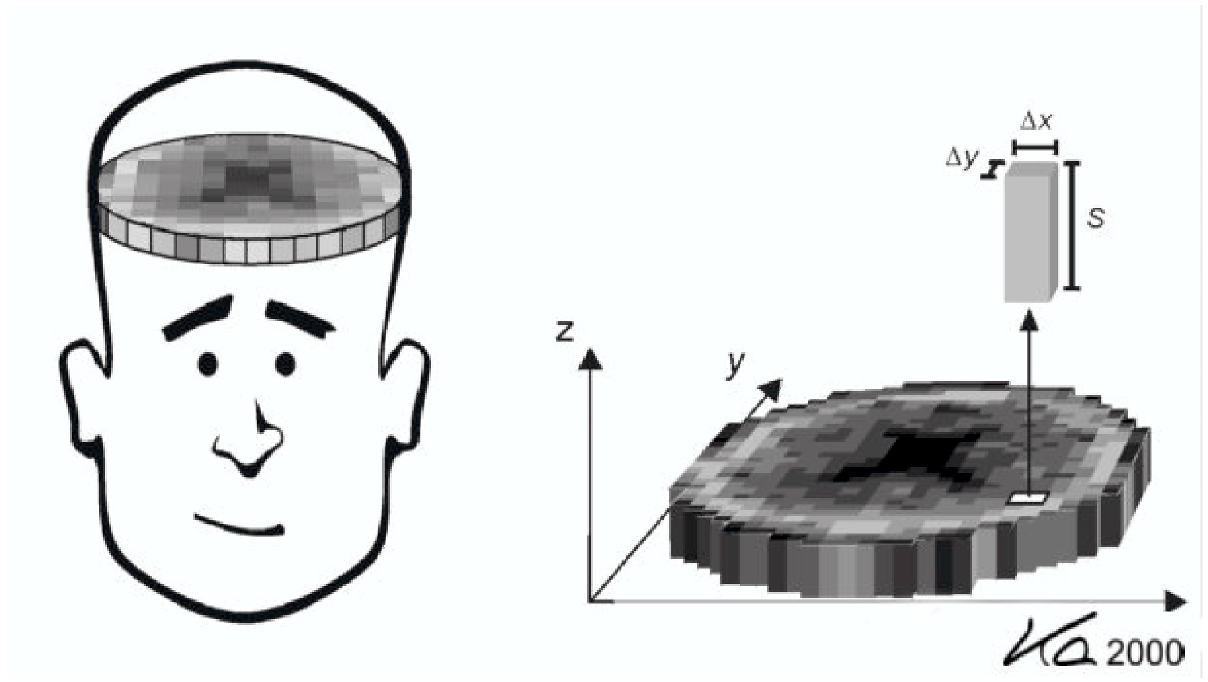


Figure 2.3: A 2-D slice of a 3-D volume. From Prof. W. Kalender [46, page 18], with permission June 18, 2007.

source as well as stationary detectors. An electron beam is deflected onto an X-ray tube anode in the form of a large semicircular ring surrounding the gantry. The location of the source of X-rays in a given instant is the result of where the electron beam is deflected.

All of these systems produce an image of a 'slice' perpendicular to the axis of rotation. Consecutive slices can then be stacked to produce a representation of a 3-D volume (Figure 2.3).

The table has been translated a small distance along the axis of rotation (the z axis) between each slice, but another innovation has allowed continuous motion of the table, so that scanning occurs in a *spiral* or *helical* pattern. Each slice continues into the next, much like a twist of lemon. Without the need to repeatedly start and stop the motion of the table, there is a further improvement in speed. For such spiral CT scanners, various 'z interpolation' algorithms exist to determine the attenuation value of a voxel, given its angle of rotation [47, page 81].

Another innovation is the addition of multiple rows of detectors to produce a 'thick' fan beam (Figure 2.1, 1998). This allows for thicker slices or overlapping slices to be used.

Extending these multiple row detectors to a true 2-D array of detectors yields *cone beam* CT (Figure 2.1, future). The diverging rays project in the shape of a cone or pyramid, the 3-D equivalent of the 2-D fan beam. Introducing the effects of a third dimension into the reconstruction has implications for the choice of reconstruction algorithm.

2.1.2 Reconstruction Algorithms

There are two major families of reconstruction methods: filtered or Fourier backprojection (FBP), or convolution backprojection (CBP) methods, and iterative, or algebraic techniques. An outline of both major methods is presented here.

Convolution Backprojection

An image can be obtained by adding the detection value to every contributing voxel in the projection. If the target object has sharply defined contrasting regions, this *summation* method will cause these to be *blurred* much like a photograph out of focus (see the left hand side of Figure 2.4). This summation is termed *backprojection* because it involves placing the projections back into the image. This thesis uses the terms *straight* backprojection or *unfiltered* backprojection to refer to this process when no other operations are performed on the backprojection image.

The result can be focused using a *filtered* backprojection (FBP) algorithm. Typically, a convolution filter is utilized that is based on the Fourier Slice Theorem [45]. Because the convolution in the CBP methods is typically done in Fourier space, we use the terms *Filtered backprojection or Fourier backprojection (FBP)* and *convolution backprojection (CBP)* collectively to refer to the whole group of filtered backprojection methods. Although in principle the backprojected image could be deconvoluted using a 2-D filter, an equivalent transformation can be obtained by passing a 1D filter over the projection data *before* the backprojection in the case of parallel projections.

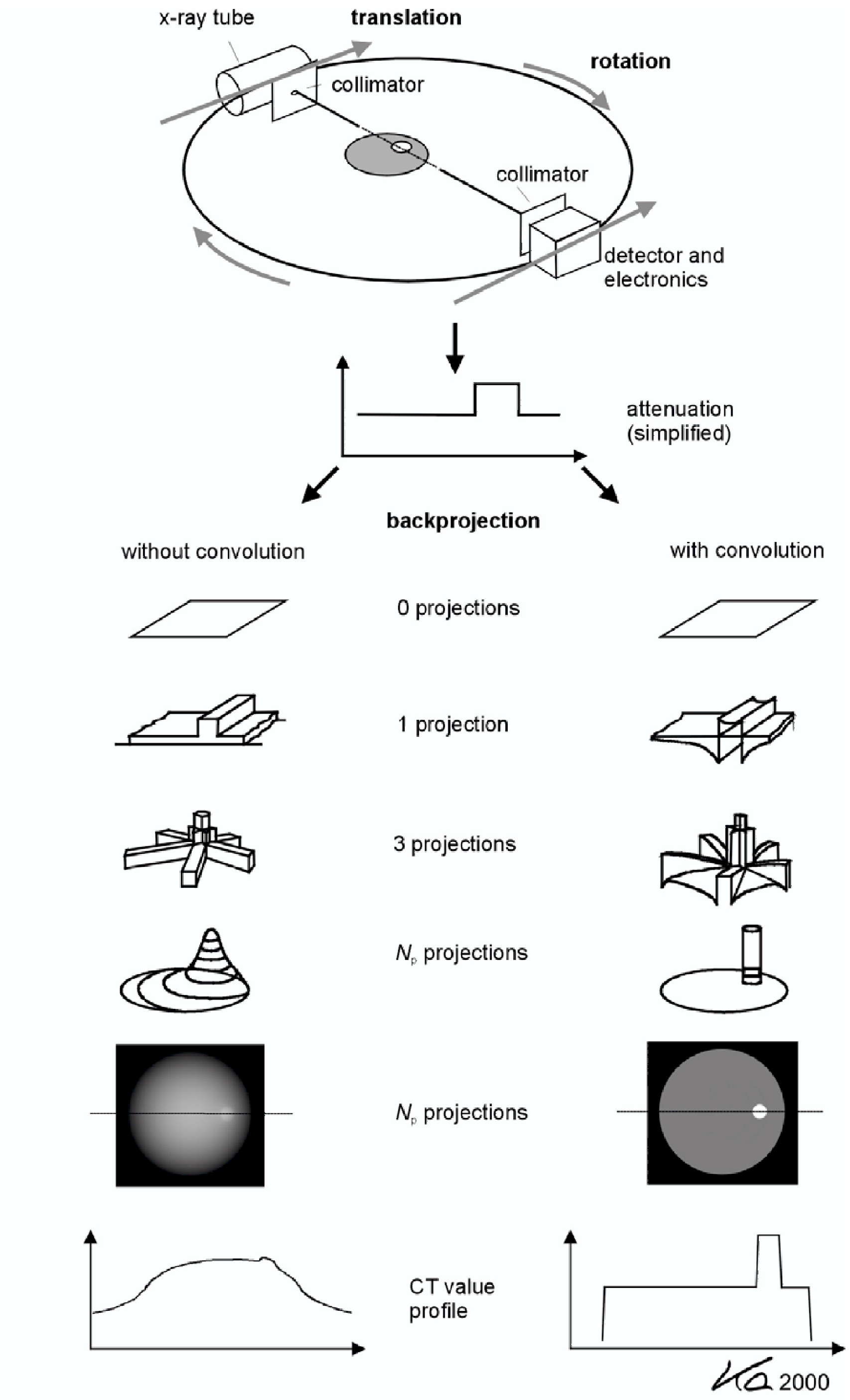


Figure 2.4: Summation *vs.* convolution and backprojection. From Prof. W. Kalender [46, page 28], with permission June 18, 2007.

An overview of CBP is given in Kalender's book [46]. The CBP algorithm is shown as Algorithm 2.1, and is comprised of four steps for each view angle [72]. The Q_{θ_i}'s are views taken at various angles θ_i at which the projections were taken, K is the number of angles, and $f(x, y)$ are the pixel values of the resulting image.

```
1: for each projection $Q_{\theta_i}$ at angle $\theta_i$ do
2:     Find the 1-D Fourier Transform (FT)
3:     Convolve the result of step 2 by a response function in the spatial domain, or
       multiply the result of step 2 by a response function in the frequency domain
4:     Find the inverse FT of the results of step 3
5:     Backproject by computing $f(x, y) = \sum_{i=1}^{K}(x\ cos(\theta_i) + y\ sin(\theta_i))Q_{\theta_i}$
```

Algorithm 2.1: Convolution Backprojection Algorithm.

This method has some advantages that were important in the first and second generations of CT. The Fast Fourier Transform (FFT) algorithm developed by Cooley and Tukey in 1965 [15] greatly increase the speed of the Fourier Transform. The FBP algorithm can be hard wired for further speed increases. Also, given that the projections at each angle can be made to approximate a parallel set, the *Central Slice Theorem* allows each set of projections to be convolved separately. This can occur in parallel with the scanning process, so that calculations can be made on some projections before others have even been collected. Due to the minimal computing power in the early decades of CT, this ability to interleave data collection with computing was of great importance.

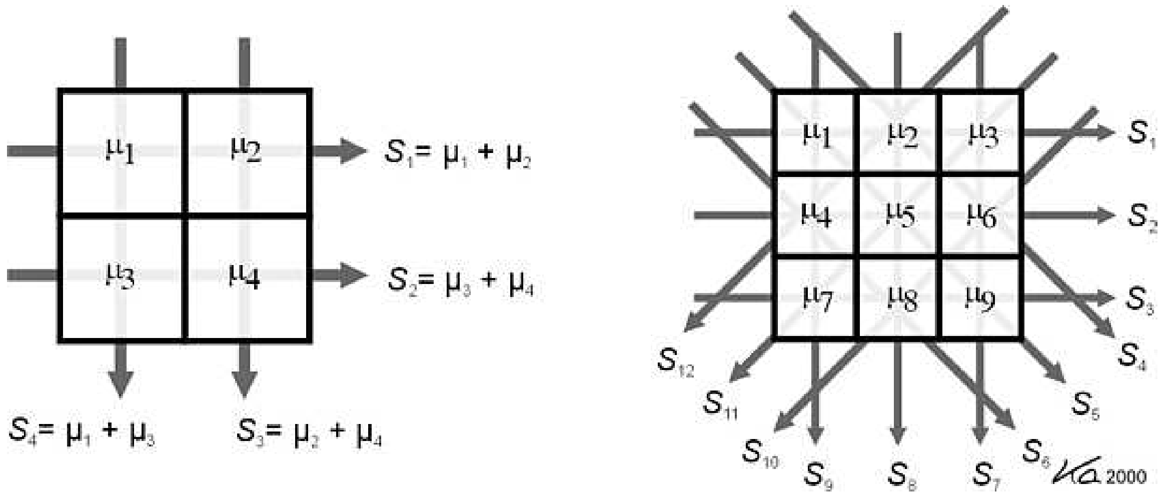


Figure 2.5: The reconstruction problem as a system of linear equations. From Prof. W. Kalender [46, page 27], with permission June 18, 2007.

Iterative Methods

The problem of CT reconstruction can be viewed as a system of linear equations. In this model, each pixel (voxel) j is assumed to have a homogenous attenuation μ_j, an unknown value to be solved. The measured projection data is a set of attenuation sums S_i. Each S_i is the weighted sum of the attenuations of pixels along a given ray, also known as a ray integral or raysum (as seen in the left hand illustration of Figure 2.5).

Different variations of the model can be used to determine the weight w_{ij} that each pixel j contributes to the ith weighted attenuation sum S_i. Let us use a model where each weight w_{ij} is the product of the pixel's attenuation μ_j and the length of the ray's intersection with the pixel (expressed in pixel widths). The weights can then be determined geometrically from the angle and position of the ray (these are determined from the geometry of the scanner) and the chosen pixel dimensions.

Generally, the scanner has a linear array of D equally spaced detectors, and the entire array is rotated through Q angles. The collection of D rays at a given angle is called a 'view' (the term used in this thesis) or a 'projection'. There is a total of $M = DQ$ raysum measurements. Let us assume that there are N pixels in the image; then there are MN weights that each pixel contributes to each raysum.

Using the left hand side of Figure 2.5 as an example, we have an image of $N = 4$ pixels. There are 2 detectors in the detector array, and the array is rotated through 2 views (horizontal and vertical) to produce $M = 4$ raysums.

We therefor have $MN = 16$ weights. The weights for raysum S_1 are calculated easily in this case. The ray traverses the width of pixel 1, so the weight of contribution of pixel 1 to the raysum is $w_{11} = 1$. Likewise, $w_{12} = 1$. Pixels 3 and 4 do not intersect ray 1, so $w_{13} = w_{14} = 0$. Similarly for the other rays in this example, all weights are 0 or 1, and the raysum equations are as follows:

$$S_1 = \mu_1 w_{11} + \mu_2 w_{12} + \mu_3 w_{13} + \mu_4 w_{14} = \mu_1 + \mu_2$$

$$S_2 = \mu_1 w_{21} + \mu_2 w_{22} + \mu_3 w_{23} + \mu_4 w_{24} = \mu_3 + \mu_4$$

$$S_3 = \mu_1 w_{31} + \mu_2 w_{32} + \mu_3 w_{33} + \mu_4 w_{34} = \mu_2 + \mu_4$$

$$S_4 = \mu_1 w_{41} + \mu_2 w_{42} + \mu_3 w_{43} + \mu_4 w_{44} = \mu_1 + \mu_3$$

In general, each ray S_i can be represented as:

$$S_i = \sum_{j=1}^{N} w_{ij}\mu_j, i = 1, 2, ..., M \tag{2.1}$$

Half of the NM weights w_{ij} are zero. For the case of the 9 pixel image on the right hand side of Figure 2.5, approximately two thirds of the weights are zero. In general, for large images, a substantial portion of the weights are zero, because many of the pixels make no contribution to a particular raysum.

As an example, we try to solve for the attenuation values given: $S_1 = 5$, $S_2 = 9$, $S_3 = 8$, $S_4 = 6$.

There is no unique solution, but rather an infinite number of solutions. Among the M equations, there must be N *independent* equations and all of the equations must form a *consistent* system in order for us to find a unique solution. By inspection, we can see that $S_1 + S_2 = S_3 + S_4$, hence the equations are not independent.

For demonstration purposes, we can remedy the situation by adding a single ray, running diagonally through the centres of pixels 1 and 4. The distance that this new ray, (labelled ray 5), travels through pixel 1 is $\sqrt{2}$ pixel widths, so the weight of contribution of pixel 1 to the raysum is $w_{51} = \sqrt{2}$. Likewise, $w_{54} = \sqrt{2}$. Pixels 2 and 3 do not intersect ray 5, so $w_{52} = w_{53} = 0$. We add the equation

$$S_5 = \mu_1 w_{51} + \mu_2 w_{52} + \mu_3 w_{53} + \mu_4 w_{54} = \sqrt{2}\mu_1 + \sqrt{2}\mu_4$$

Given the additional information that $S_5 = 7\sqrt{2} \approx 9.89949493661167$, we can now obtain a unique solution: $\mu_1 = 2$, $\mu_2 = 3$, $\mu_3 = 4$, $\mu_4 = 5$.

Now, suppose that there is a 1% measurement error in S_1, so that $S_1 = 5.05$ instead of $S_1 = 5$. There is no solution to satisfy all of the equations, since the system is inconsistent.

We may choose the number of detectors and the number of views such that their product is N. In practice, some equations may be linear combinations of others, causing the system to have an infinite number of solutions. Very often, detector noise creates inconsistencies, so no exact solution can be found. Hence, it is not practical to solve such systems of equations using a method such as Gauss-Jordan.

One approach to solving large systems of equations, iterative approximations, forms the basis of the *iterative* or *algebraic* methods. Successive adjustments are made to the attenuation values until a solution is reached that is consistent with the projection values by some criterion. Iterative methods compare the computed ray sums of an estimated image with the original projection measurements and use the error obtained from this comparison to correct the estimated image. Though there is unlikely to be an exact solution because of inconsistencies, this method yields an approximate solution to the attenuation values.

In general, M and N are quite large. For example, when reconstructing an image size of 256×256 pixels, from 256 detector measurements in each of 256 views, N and M are both 65,536. In such cases the weight matrix size is $65,536 \times 65,536 = 4,294,967,296$. We require algorithms that are efficient in terms of both time and memory requirements to solve this on a computer without increasing turnaround time in the CT suite.

A variety of algebraic techniques exist for CT reconstruction. One such algorithm is ART (Algebraic Reconstruction Technique). Developed in 1970 by Gordon *et al.* [32] for work in electron microscopy, it is historically the first algebraic (iterative) technique. A pseudo-code synopsis of the (sequential) ART algorithm is given in Algorithm 2.2. The meanings of the variables are found in Table 2.1.

The iterations of the outer loop ($k = 1..K$) are called ‘cycles’ by some authors, who then use the term ‘iteration’ to apply to loops within the views. The term ‘view’ is synonymous to the term ‘projection’, and refers to the collection of rays

```
Require: a seed image consisting of pixels $S^0_{ij}$
 1: for each iteration $k = 1$ to $K$ (or until convergence) do
 2:   for each view $q = 1$ to $Q$ at angle $\theta$ do
 3:     for each ray $d = 1$ to $D$ do
 4:       $P_{calc} \Leftarrow 0$; $w_{total} \Leftarrow 0$; $h \Leftarrow 0$;
 5:       for each column in the pixel array $i = 1$ to $n$ do
 6:         for each row in the pixel array $j = 1$ to $n$ do
 7:           if $w_{ij\theta d} > 0$ then
 8:             $P_{calc} \Leftarrow P_{calc} + S^{q-1}_{ij} w_{ij\theta d}$
 9:             $w_{total} \Leftarrow w_{total} + w_{ij\theta d}$
10:             $h \Leftarrow h + 1$
11:       for each column in the pixel array $i = 1$ to $n$ do
12:         for each row in the pixel array $j = 1$ to $n$ do
13:           $U_{ij} \Leftarrow U_{ij} + \frac{P_{\theta d} - P_{calc}}{denominator}$
14:     for each column in the pixel array $i = 1$ to $n$ do
15:       for each row in the pixel array $j = 1$ to $n$ do
16:         $S^q_{ij} \Leftarrow S^{q-1}_{ij} + \alpha U_{ij}$
17: end
```

Algorithm 2.2: ART Algorithm.

projected in a given direction. The detector index specifies a given ray within the view. The total number of iterations K is specified in advance for some variations of the algorithm, but can be determined by a measure of convergence (such as the correction factors being below a given threshold). Note that the angles may be

ART Algorithm Variables	
Variable	Description
k	iteration index
K	total number of iterations
q	view index
Q	total number of views
θ	angle of view
d	detector (or ray) index
D	total number of detectors
S_{ij}^{q}	current pixel value at row i, column j during the q'th view
U_{ij}	array to store updates to pixel value at row i, column j
$P_{\theta d}$	intensity detected by detector d at view angle θ
P_{calc}	calculated raysum
α	relaxation factor
$w_{ij\theta d}$	weight of the contribution of pixel S_{ij} to $P_{\theta d}$
w_{total}	sum of weights of all pixels contributing to ray
h	count of pixels contributing to ray
denominator	EITHER w_{total} OR h OR path length of ray

Table 2.1: Variables used to describe the ART algorithm.

processed in any order, depending on the specific variation of ART, so that θ for $q = 1$ may be greater than θ for $q = 2$. The variable α is the *relaxation factor*, and, if set to less than 1, decreases the magnitude of corrections made, possibly generating a smoother image or preventing oscillation. The *denominator* is typically h, the count of pixels contributing to the particular raysum, but variations allow for the denominator to be the sum of the weights of all pixels contributing to the raysum w_{total}, or the length of the portion of the ray contained within the image (expressed in pixel widths) [45].

ART, as well as its successor the Multiplicative Algebraic Reconstruction Technique (MART) [30, page 85], have been shown to converge on a least squares solution by Byrne and Graham-Eagle [9]. The Simultaneous Algebraic Reconstruction Technique (SART) [4], was proposed in 1984 as a refinement of ART. This has now become popular, at least, in academia. Recently, Jiang and Wang [43] proved that SART also converges to a weighted least squares solution.

Historically, the Fourier (convolution backprojection) methods have been favoured over iterative methods in production CT scanners, and it is these faster "analytic methods that are currently used today in CT machines" [89, page 324]. The computational speed and the analytical nature of the CBP methods may be among the factors responsible for this.

In fact, both families of methods, iterative and CBP, accomplish the same task, shown to have a theoretical common ground noted by Older and Johns [66]. The approaches to creating the image, however, are quite different. In many situations, the iterative methods provide better image quality than CBP methods, particularly with noisy data or a limited number of views, as described in the sections that follow. This thesis uses ART, and iterative algorithms that are minor variations of ART, exclusively.

2.1.3 Cone Beam Reconstruction

To date, CT has been done by accumulating a large number of one dimensional projections and reconstructing a two dimensional slice, usually by a convolution method. Three dimensional reconstructions are made by stacking these slices. In this conventional configuration, the X-ray beam from source to detectors is in the shape of a fan (see Figure 2.1). Cone beam technology, on the other hand, uses a beam directed at a 2-D detector surface array, and allows for direct reconstruction in 3-D.

The problem of reconstruction from cone beam data is not one of simply combining 2-D views. In any 2-D view, a voxel's point of rotation about the axis as well as its distance and direction from the axis impacts on the z location of its representation, as seen in Figure 2.6. This increases the difficulty of performing such 3-D reconstructions.

Cone beam technology makes more efficient use of the X-ray source. The total X-ray dose for a typical CT examination is the product of a large number of 1-D views per slice and the number of slices. Cone beam offers the potential to make a 3-D reconstruction from a much smaller total number of 2-D views, since separate slices are not required. However, an array of $d \times d$ detectors used to take a 2-D view requires a larger X-ray dose than does a linear array of d similarly sized detectors used to take a 1-D view. Mueller *et al.* [62] estimate the number of projections required to reconstruct a spherical region of interest as $0.67n$, where n is the length in pixels of the $n \times n \times n$ cubic reconstruction grid. In the more practical case of reconstructing a cylindrical area of interest, which could be done with modification of a helical scanner,

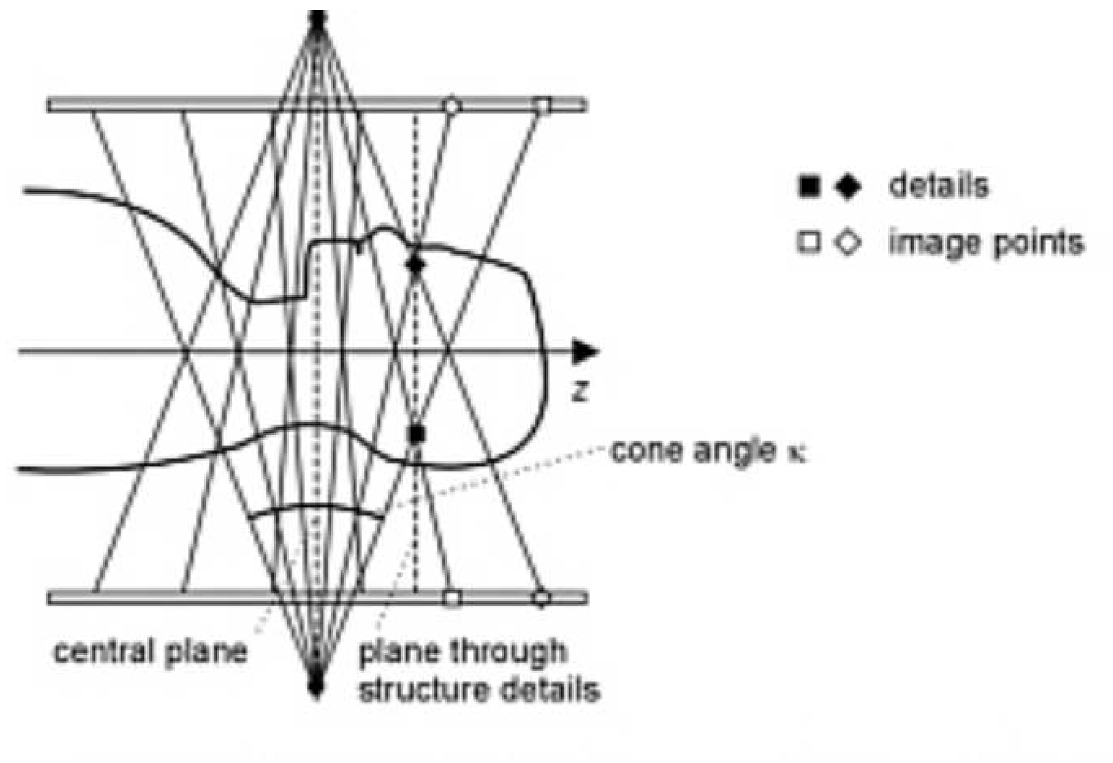


Figure 2.6: Challenge of cone beam reconstruction. From Prof. W. Kalender [46, page 64], with permission June 18, 2007.

the number of projections required is $0.78n$ [62, page 539]. This suggests a theoretical basis to expect improved efficiency with cone beam reconstruction.

Defrise [19, page 115] notes that the improved speed and efficiency comes at a computational cost:

> "By increasing the number of photons detected, this 3-D approach allows faster imaging, but image reconstruction becomes much more complex since the 3-D object can no longer be separated into a stack of independent slices."

Basu and Bresler [6] discuss a CBP algorithm that runs in time $O(N^2 \log N)$, and adapt this for 3-D reconstruction [7] in $O(N^3 \log N)$. Grass *et al.* [35] develop an algorithm which created a 3-D reconstruction from 2-D projections, using an extra 'rebinning' step to convert the diverging cone beam rays into parallel rays striking a "virtual detector plane". With the addition of this step, an overall decrease in complexity is claimed. De Man and Basu [18] discuss a time and memory efficient algorithm for projection and backprojection which is easily extensible to 3-D.

The developments in cone beam reconstruction discussed thus far deal with CBP algorithms, some of the studies using the Feldkamp algorithm (FDK), an adaptation of CBP for cone beam CT [23]. However, the FDK algorithm is primarily used with wide fan beam, rather than full 3-D cone beam reconstruction.

Progress has also been made with 3-D cone beam iterative algorithms. Mueller *et al.* [62] examine methods to speed up iterative algorithms with the goal of making 3-D cone beam iterative reconstruction practical. Chlewicki *et al.* [12] examines cone based 3-D reconstruction, comparing the Feldkamp algorithm to SART, an iterative algorithm. Chlewicki *et al.* [13] also propose a fast SART implementation for 3-D use. There is a significant role to fill for fast iterative algorithms in full 3-D reconstruction.

2.1.4 Limited View Reconstruction

Another approach to limiting radiation exposure is to reduce the number of views from the number currently used in most CT scanners, using algorithms that optimize use of the information present and minimize artifacts when presented with small numbers of projections. The examination and modification of such reconstruction algorithms is a problem that clearly falls within the scope of computer science.

The Central Slice Theorem applies only to parallel projection rays. To utilize convolution backprojection with any Fourier method, (2-D) fan beam or (3-D) cone beam projections must be sorted into parallel projections. As the number of views becomes more limited, it becomes more difficult to 're-bin' individual detections to parallel rays, and true 3-D reconstruction algorithms must be considered. An iterative method does not have this limitation; while geometric distortion can occur if this is done naively, there is no inherent requirement for rays of any view to be parallel. Mueller [61] notes the degradation that may occur with convolution backprojection methods when the number of projections is limited, and proposes the use of iterative algorithms for limited view cone beam reconstruction. Thibault *et al.* [83] find improved image quality using iterative methods compared to CBP methods (the Feldkamp algorithm) when working with 3-D cone beam reconstruction.

In general, Fourier convolution backprojection methods tend to degrade the image badly when the number of views is reduced [36, page 25]. It is therefore likely that any limited view algorithm will use the iterative approach, in spite of the increased computational cost.

Kak and Slaney [45, page 275] note:

> there are situations where it is not possible to measure a large number of projections, or the projections are not uniformly distributed over 180 or 360^0, both these conditions being necessary requirements for the transform-based techniques to produce results with the accuracy desired in medical imaging.

In such situations, algebraic techniques are essential. Iterative methods also suffer from streak artifacts with limited views, but there are some solutions to this problem [69; 70]. An example of a limited view application is CT imaging of the heart, where the beating motion limits the number of views that can be acquired for a given position of the heart. For such an application, iterative methods are preferred.

2.2 Related Work

This chapter discusses the relevant literature on CT with regard to improvements in iterative CT reconstruction algorithms, parallel approaches to CT reconstruction, and in particular with regard to communication overhead in parallel approaches.

2.2.1 Improvement in Iterative Algorithms

This thesis uses iterative algorithms, ART and minor variations of ART, hence some related work regarding iterative algorithms is reviewed. Prior to the invention of the first commercial CT, Gordon at al. [32] developed ART, the forerunner of the iterative algorithms, to use for electron microscopy. The application to radiology became immediately clear, and Gordon [34; 33; 30] along with others describe early iterative algorithms primarily for use with CT (then called 'reconstruction from projections' or 'computerized axial tomography'). The ART algorithm has been fine tuned by various researchers. For example, Rangayyan *et al.* [69; 70] refine the algorithm to prevent streaks with limited views. Wang *et al.* [88; 74] use iterative deblurring techniques to reduce artifacts in local region CT.

The rate at which the algorithm converges on a solution is a major factor in the overall speed of the algorithm. The rate of convergence is affected by the order in which views are accessed, an issue studied by several researchers. Herman and Meyer [40] propose a scheme that uses variable angles between successive projections as they are accessed. Guan and Gordon [37] implement a Multilevel Access Scheme (MLS) to maximize orthogonality between consecutive projections, and demonstrate a dramatic improvement in the speed of the convergence of ART. They claim that their algorithm speeds the convergence rate of ART and has superior image quality compared to the Fourier backprojection algorithm for a small number of projections. Their algorithm works best when the number of projections is a power of two. Mueller *et al.* [60] propose a Weighted Distance Scheme (WDS) that speeds up the convergence of ART algorithms. This scheme heuristically determines the angular distances of the newly selected projections. They claim that their scheme produces better images with less noise than other ordering schemes, but it is sub optimal compared to a global optimum selected from all permutations of the projection angles.

There are variations of iterative algorithms including the Multiplicative algebraic reconstruction technique (MART) [30, page 85], the Maximum Likelihood (ML), Expectation Maximization (EM) [75; 80; 49], Simultaneous Iterative Reconstruction Technique (SIRT) [17; 48; 86], and Simultaneous Algebraic Reconstruction Technique (SART) [3; 4], that is recently proven to converge [43; 42]. The EM algorithm is mainly applied to Positron Emission Tomography (PET) imaging techniques where the noise level in projection data is relatively high. The EM algorithm is efficiently parallelized on the transputer [5; 10], BBN butterfly and Intel Paragon [14], Beowulf cluster [78] and P2P environments [54].

There are a wide variety of iterative algorithms, many of which demonstrate adaptability to 3-D and robustness to input of a limited number of views. Iterative algorithms are necessary in those situations, where radiation exposure is reduced, and are the algorithms of choice in this thesis. Many enhancements to improve speed are listed in this section, but performance remains an issue. The section that follows discusses a means of improving performance: parallelization.

2.2.2 Parallel Approaches to CT Reconstruction

CT reconstruction algorithms are implemented on a wide variety of hardware. Lattard *et al.* [51; 50] use a SIMD data-flow machine (a fine-grained array of processing elements) to run parallel ART implementations. Fitchett [24] also implemented ART on a SIMD machine, using a different type of processing element (PE). One objection to the use of such specialized hardware is cost. Mueller and Yagel [59] use a much lower cost hardware: the modern PC's graphics hardware. They exploit the parallelism of texture mapping hardware to achieve significant speedup with the SART algorithm.

A common and hence less expensive form of parallel hardware is a computer network, which can be anything from a homogeneous *Cluster of Workstations* (COW) to a loosely connected heterogeneous network. Software such as the Message Passing Interface (MPI) [25, Chapter 8] or Parallel Virtual Machine (PVM) [27] is often used in such systems to handle the communication between processors. Typically under MPI, all processors execute the same program in a *single program multiple data* (SPMD) model, but it is possible for the processors to execute different programs, creating a *multiple program multiple data* (MPMD) model [25]. PVM runs on MPI and provides additional functionality for heterogeneous networks, and is especially suited for running programs under the MPMD model.

Guerrini and Spaletta [38] implemented reconstruction algorithms on vector computers. These computers were quite limited in terms of power and memory for large 3-D reconstruction problems.

Chen *et al.* [11] implemented the convolution backprojection algorithm for 3-D parallel beam geometries on the Intel hypercube, iSPC/2 multiprocessor. They use the 3-D incremental backprojection algorithm [11] that performs backprojection on a ray-by-ray (beam-by-beam) basis as opposed to a pixel-by-pixel approach. This algorithm is faster than the conventional backprojection algorithm. They exploit functional parallelism on two functions, convolution and backprojection, by a two stage pipelining approach. Task partitioning in the convolution and backprojection stages is discussed extensively. Processing an image containing 31×31×31 voxels, a speedup of 5 is obtained using 6 processing elements (PEs): 5 for convolution and 1 for backprojection. However, a speedup of 26 is obtained on a 63×63×63 voxel image using 32 PEs, 30 for convolution and 2 for backprojection.

Rao *et al.* [72] implemented the filtered back projection algorithm for 2-D cone beam tomography on the CM5 and Intel Paragon parallel platforms. On the CM5, they use the Connection Machine Scientific Subroutine Library (CMSSL) for finding the FFT and Inverse FFTs. Their results indicate that the Intel Paragon produces much more efficient results than the CM5. The CM5 does not produce speedup until it reaches 256 processors, while the Intel Paragon produces speedup at 8 processors. As the number of processors is increased in the CM5 (from 200 to 500 processors) the communication overhead takes over the whole computation. While the communication overhead dominates at larger numbers of processors on both types of machine, the communication overhead is more marked on the CM5 due to the overall higher number of processors used and "the greater communication requirements between the processors of the CM5 as compared to the Intel Paragon" [72].

Reimann *et al.* [73] extend the idea of 2-D CT to 3-D cone beam tomography. They implemented the most efficient filtered backprojection algorithm by Feldkamp, Davis and Kress [23] (also called FDK or Feldkamp algorithm) for cone beam tomography on a shared memory machine and cluster of workstations (COW) using the message passing interface (MPI). They notice that the backprojection step of the Feldkamp algorithm produces a load imbalance. They tackle this problem by providing two efficient techniques that increase the processor utilization. The authors use very small machine sizes to implement their algorithm and conduct experiments. On the COW, with only 6 processors, they use load balancing obtain a processor utilization of 71.7% compared to 58.2% when the algorithm is unbalanced. Implemented on the shared memory machine with two processors, They obtain a speedup of 1.94 with processor utilization of 95.1%.

Laurent *et al.* [52] study 3-D cone beam tomography using three different algorithms: Feldkamp, block ART and SIRT. They provide both theoretical complexities and experimental results for these algorithms. They also propose local and global solutions for the parallelization of projection and backprojection algorithms. The algorithms are implemented on five different MIMD computers: a network of workstations (Sparc 2 and 5 Sparc 3 machines), a farm of processors (16 AXP processors, Giga-switch network topology), a Paragon (32 i860 processors), a T3D (128 AXP processors, 3-D torus network topology), and an SP1 (32 RS6000 processors). All experiments were conducted using the Parallel Virtual Machine (PVM) [27] library. The Feldkamp algorithm yields the best performance on all machines. The T3D machine, in general, produces better speedup than the other machines, this being attributed to efficient communication handling on the T3D.

Recently there has been added interest in Computational Grids. Smallen *et al.* [81]

studies implementation of the 3-D cone beam reconstruction algorithm on Grids. By combining workstations and supercomputers, they run the GTOMO (Computational Grid Parallel Tomography) application implemented with a work queue scheduling strategy used in production at NCMIR (National Center for Microscopy and Imaging Research). The experiments were conducted on a cluster of 7 workstations (available at University of California San Diego) and an SP2 supercomputer (available at San Diego Supercomputer Center with 128 nodes). This work is focused primarily on scheduling strategies for applications running on a Grid, rather than their experiences with the reconstruction problem.

Melvin *et al.* [56] use MPI on a Beowulf cluster to reconstruct a 2-D image in parallel using iterative methods. Data partitioning is done according to projections, each of a limited number of views (projections) being assigned to one processor in a cluster. The speedup is minimal, presumably because the work of communication between the processors detracts from any computational speed gains. Data dependencies are noted between successive projections, and pixel values from each projection have to be communicated to every other processor during each iteration.

The issue of communication overhead is a recurring theme in the studies mentioned thus far that use distributed memory architectures. Melvin's solution was to reimplement on a shared memory architecture [55]. This thesis reexamines the issue of communication overhead by exploiting some evidence of locality in the data, discussed in the next section.

2.2.3 Communication Overhead in Parallel CT Reconstruction

A recurring theme in the previous section dealing with the literature on CT reconstruction in parallel, particularly on distributed memory architectures, is limited speedup due to communication overhead. A problem with parallel CT reconstruction is that of data dependence: an estimate of the value for any pixel may affect the value of any other in a subsequent iteration. This data dependence can be conceptualized as follows: choose any two pixels on a CT image slice; if there is a sufficiently large number of views, there is at least one projection ray that includes both pixels. Hence, any estimate of one pixel affects the estimate for the other, since both contribute to the detected sum. Because calculations involving data on one processor's local memory may depend on data in a different processor's local memory, significant communication overhead may be required to move the data between processors, needed to parallelize a CT reconstruction algorithm. Three possible solutions to this problem are: to use a shared memory implementation to avoid communication overhead, to find new algorithms which are more inherently parallel, or, to find and exploit some locality to minimize communication overhead.

The shared memory solution may introduce synchronization overhead, and involves hardware that is somewhat less common and more expensive than the ubiquitous distributed memory hardware utilized in this thesis.

An example of the second solution is to exploit optimization algorithms. It is possible to conceive, for example, of an iterative algorithm that is influenced by a 'guess' for a starting pixel value. Each processor could work on its own image initiated from a different set of starting values, working in parallel until the images were compared by some criterion to select the preferred image.

The third possible solution, used in this thesis, is to find and exploit some locality in the data. In spite of the apparent data dependence, there is evidence of spatial locality in the reconstructed image. Evidence for this comes from: work done with image 'errors', and the success of a technique known as *Local CT*. Gordon [29, page 276] observes that a reconstruction may be locally manipulated with little effect on other regions [29]. He examines the effects of feature addition or removal during reconstruction of an image with limited views (hence highly undetermined equations) to look at the possibility of detecting features not truly present and missing features which could be present in the limited view reconstruction. Of note is that a feature could be significantly altered but manipulations made on a localized region of a reconstruction have only minor consequences to other regions. This somewhat unexpected result indicates that there is some data locality related to spatial position in the image.

Local CT may serve as a model to demonstrate spatial locality in a CT image. Describing the mathematical foundations of CT, Smith and Keinert [82, Figure 3 on page 3957] propose a reconstruction of δf, differences in attenuation, rather than f, the actual attenuation. They note that such a reconstruction is highly local, requiring only rays that traverse the region of interest or are very close to it to provide an accurate reconstruction [82]. They propose coning the X-ray beam in to the region of interest, using small detectors to get a high resolution scan of this area. This is done using data from scans of phantom test images, with surprisingly good results in spite of limited views and coarse detector sampling.

This is the underlying principle of local CT. As described by Daatselaar [85, page 361],

> “Local CT is a variant of CT that aims to reduce the radiation dose to the patient by using a narrow X-ray beam that only covers the area of interest.”

Local CT involves the use of limited radiation projected at a particular organ or body region of interest to minimize radiation exposure to regions which are not of interest. Some loss of image quality occurs due to the incorrect assumption that parts of the object outside the region of interest do not contribute to the projection values. In spite of this, an image of reasonable quality can be obtained.

Gordon’s work with image ‘errors’ and the success of local CT demonstrate some evidence of spatial locality: data dependence is highest between image pixels which are spatially close to one another, but this dependence is much less between widely separated pixels. This thesis makes use of such locality for data partitioning (Section 3.6) in order to limit the communication overhead on a distributed memory implementation of parallel ART.

Chapter 3

Materials and Methods

This chapter describes the tools used and the methods applied in order to run the experiments of the following chapter. First there is a discussion of the choice of programming language. A brief outline of the hardware, operating system, and programming environment are then presented. A section on CT simulators and virtual phantoms follows, containing an introduction to CT simulators, the available types of virtual phantoms and CT simulators, and the CT simulator and virtual phantoms used in this thesis. The following two sections discuss how the implementation is used to specify the algorithm and its parameters: the first describes the choice of algorithm and parameters for the underlying iterative algorithm, and the second describes the choice of parameters specific to the parallel version. The final section describes the data partitioning methods that have been tested in this thesis.

3.1 Programming Language

An imperative programming language such as C or Fortran allows for convenient and detailed specification of the reconstruction algorithm. Both C and Fortran have libraries available to allow multiprocessing. Matlab's facility to handle matrices and its graphic capabilities make it an obvious choice, and this is used by Melvin [56; 55] in a previous work with parallel ART.

However, because of the potential complexities of the data structures such as weight tables and scanner geometries, an object oriented language might be more suitable. Smalltalk is designed to allow objects to interact by message passing, but requires a very specific operating environment and may have some performance issues [77, Chapter 15].

The best overall choice would be one of two languages which combine imperative and object oriented paradigms: Java and C++. Both of these languages are widely available on a large number of platforms, and both have libraries to facilitate GUI development and multiprocessing.

Java has two main advantages over C++. The first is the built-in libraries: it has the same standard GUI library regardless of the platform, and has built-in multi-threaded capabilities. This saves development time when migrating to an unfamiliar platform. The second advantage is the ease of memory allocation, saving time in coding and saving even more time removing subtle bugs from C++ constructor methods and correcting memory leaks.

Java does have one drawback compared to C++, however, and that is performance. Java is usually interpreted at run time from byte codes, whereas C++ is (often) compiled into machine code for faster execution. The actual overall execution time may not be as important for a theoretical prototype scanner as it would for a production machine, since we want to demonstrate *relative* performance gains. However, given a large enough problem size the execution times could be large enough to impede the progress of the thesis.

C++ is chosen because of its object oriented benefits, efficient compilation, and the fact that a CT simulator is available in C++. Perl and Bash scripts fill a supporting role.

3.2 Experimental Platform

Development was carried out on a Pentium 3 containing 256 MB of RAM, running Linux Fedora Core 4, kernel 2.6.11-1.1369_FC4. The Gnu compiler used was gcc version 4.0.2 20051125 (Red Hat 4.0.2-8). For parallel programming, LAM/MPI 7.1.1 was utilized.

After initial development and some testing of multiprocessing code on the single processor, the programs were ported to the 25 "Bird" machines, a cluster of x86_64 machines each containing 1 GB of RAM, and running Linux Fedora Core 5, kernel 2.6.20-1.2316.fc5. The Gnu compiler gcc version 4.1.1 20070105 (Red Hat 4.1.1-51) and LAM/MPI 7.1.2 were used to continue development.

The *Boost* libraries were used, primarily for directory and file handling code.

3.3 CT Simulators and Virtual Phantoms

Projection data is required as input for the reconstruction algorithms that are tested in this thesis. To generate this data, a CT scanner and test objects to scan are required; the scanner scans the objects to create the projection data. A *CT simulator* or *virtual scanner* is software that simulates the physical process of scanning an object to a collection of projections. Many CT simulators have associated test objects, or phantoms, to scan. Just like physical CT scanners, most CT simulators come bundled with reconstruction software to create an image from the projections. This thesis makes use of a CT simulator and some virtual phantoms to generate projection data, but the reconstruction software of the scanner is not used. Instead, newly created reconstruction software that implements parallel iterative algorithms is used.

This section looks at the reasons for using a CT simulator (virtual scanner), discusses two major types of virtual phantom, and then introduces the specific CT simulator and virtual phantoms used in this thesis.

3.3.1 Advantages of a CT Simulator

Experimentation with CT reconstruction algorithms requires the use of clean projection data and detailed knowledge of the object being scanned. To test or calibrate a physical CT scanner, one may scan a physical *phantom*, an object whose measurements and attenuation characteristics are known precisely. One could obtain scans using a physical scanner on a physical phantom to experiment with reconstruction algorithms, but there are several problems in doing this. Most scanners are closed systems, with no access to the data directly; only the final product is accessible. The manufacturer's target market is the end user, so there is little justification to add features (however minimal they might be) to extract data from early stages of the scan. Moreover, such data could be used to improve reconstruction, in competition with the manufacturer.

Assuming raw projection data from a modern CT scanner were available, the financial cost of the technician and run time must be considered. Several standard physical phantoms exist, but these also accrue financial cost.

Instead, a CT simulator is utilized to scan at least one *virtual phantom*, an image or mathematical description of an object. The reconstruction software can then be tested on the resulting projections.

There are several advantages to using a virtual scanner. A major advantage is that one can choose any desired geometry for a virtual CT scanner, varying the distance from source to detector. Current hardware is not a limitation. For example, to experiment with wide cone beam CT, where few physical units exist, geometric parameters are simply adjusted to widen the beam.

When constructing and evaluating new algorithms, it is helpful to be unencumbered by the limitations of precision in the engineering and construction of the scanner, the imperfections of physical phantoms, and noise introduced in the measuring process. A virtual phantom allows complete mathematical knowledge of the phantom, so that reconstruction results can be compared to the original (phantom) subject. If so desired, we can add specific types and amounts of noise to the projections to evaluate the robustness of reconstruction algorithms. In the early stages of development, as in this thesis, the problems of instrument noise are avoided as much as possible.

A subtle problem to recognize in the use of a virtual scanner is the possibility of systemic error. If the programming code used to project the phantom is the same code (or similar code) to that used to backproject and reconstruct, then the possibility exists that serious errors in the projection will be cancelled by the same errors in the backprojection, hiding the errors.

A CT simulator scans certain types of objects, known as virtual phantoms, that are designed specifically to work with that particular scanner. General types of virtual phantoms and scanners are described in the following sections, as well as a description of the CT simulator used to generate projections from virtual phantoms in this thesis.

3.3.2 Two Types of Virtual Phantom

There are two major types of virtual phantoms to scan using a CT simulator, and hence two major types of CT simulators to scan these phantoms. One may scan a rasterized image phantom (Figure 3.1), or scan the geometric figures which comprise the phantom (Figure 3.2).

In order to convert the image or other mathematical construct into projection data, the CT simulator must determine which areas of homogeneous attenuation from the image are intersected by the beam of the detector in question. In the case of a rasterized image, a grid of image pixels represents the uniform areas of attenuation. These pixels are generally assumed to be square in shape (an exception using hexagons is noted [84]). The raysum measured by the detector will be the sum of the products of the fraction of pixel area intersected by the beam multiplied by the corresponding attenuation value. One could calculate directly the area of intersection between the

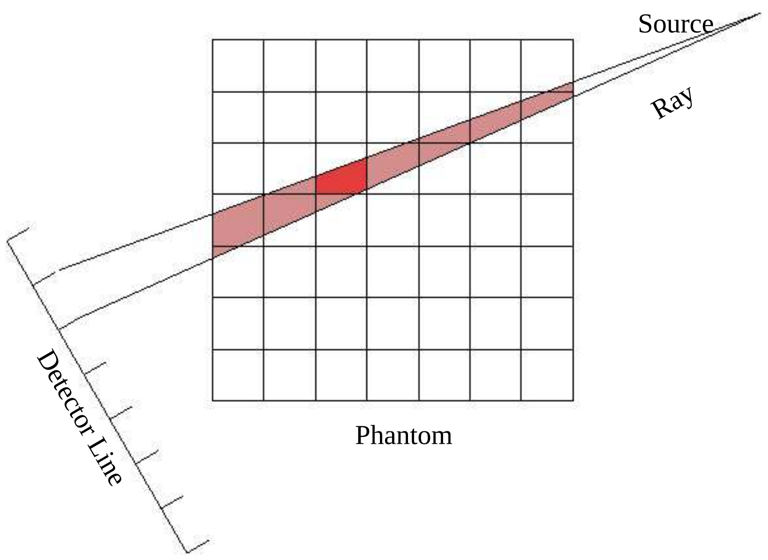


Figure 3.1: Scanning rasterized phantom.

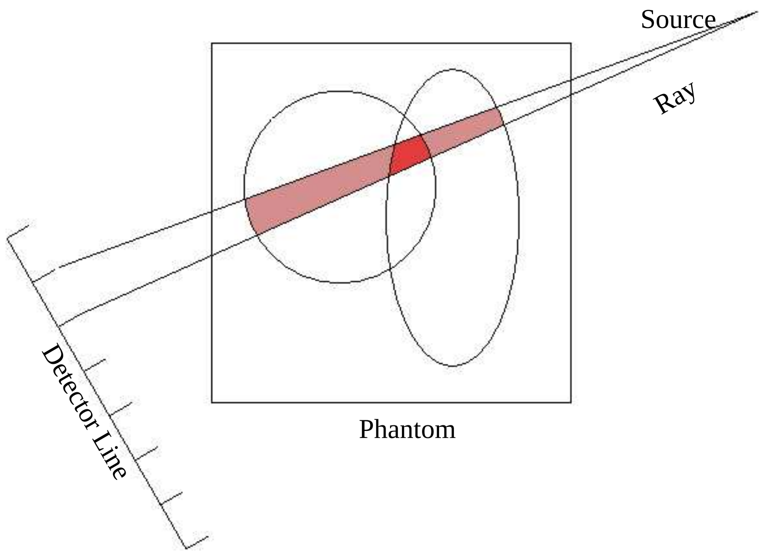


Figure 3.2: Scanning phantom of geometric curves.

areas of uniform attenuation and the beam, but this quickly becomes difficult for anything but the simplest shapes of attenuation areas and beams. Generally, an approximation to the area of intersection is calculated.

One way to approximate this is to construct a grid of equally spaced points in the image. Each point can be tested to determine on which side of a gridline or beam line the point resides, and then, by counting the points determine which reside inside both the beam and the cell, approximate the intersecting area (Figure 3.3). The algorithm nests the test for location inside several loops, potentially using much execution time (Algorithm 3.1). The same approach can be used for the phantom of geometric curves, counting the points which are inside the figure as well as the cell (Figure 3.4).

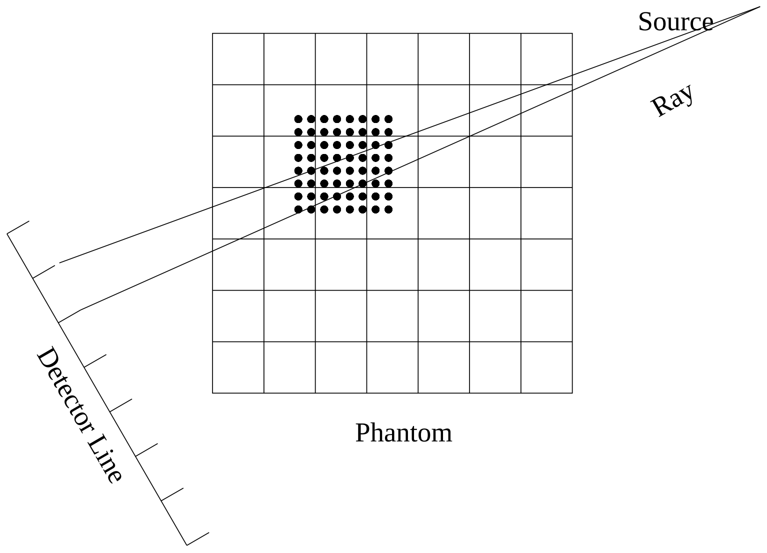


Figure 3.3: Counting points to approximate area of pixel in beam.

```
for all views do
  for $x = 0$ to $x_{max}$ do
    for $y = 0$ to $y_{max}$ do
      for $z = 0$ to $z_{max}$ do
        for each beam $b_i$ $i = 1$ to beams in view do
          for each figure in the image $g_j$ $j = 1$ to $n$ do
            if $p(x, y, z)$ is in the beam and in the figure then
              add density for figure $\mu_j$ to detector reading $d_i$
for each detector $d_i$ $i = 1$ to beams in view do
  divide total by density per grid point (normalize)
```

Algorithm 3.1: Algorithm to approximate intersection areas.

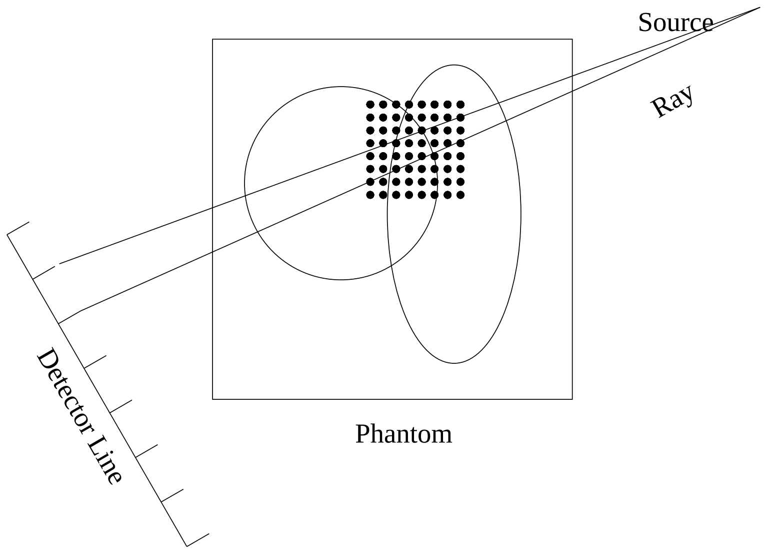


Figure 3.4: Counting points to approximate area of intersection of beam and figures.

A less computationally intensive, though less accurate approach, is to examine a ray from the source to the centre of the detector in question, and measure the length of the ray segment which traverses the region (Figure 3.5). A similar approach can be used with the geometric figures, using the ray segment length in the specified region as a proxy for the area. (Figure 3.6).

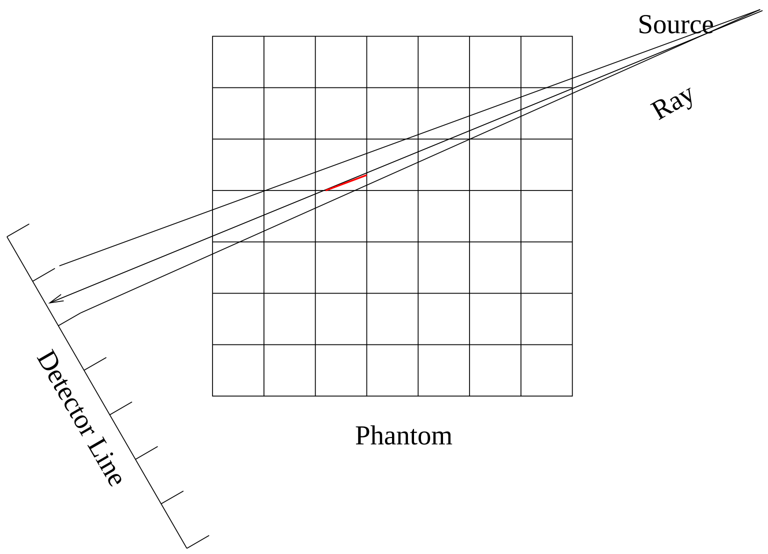


Figure 3.5: Ray segment length as a proxy for area of pixel intersected by beam.

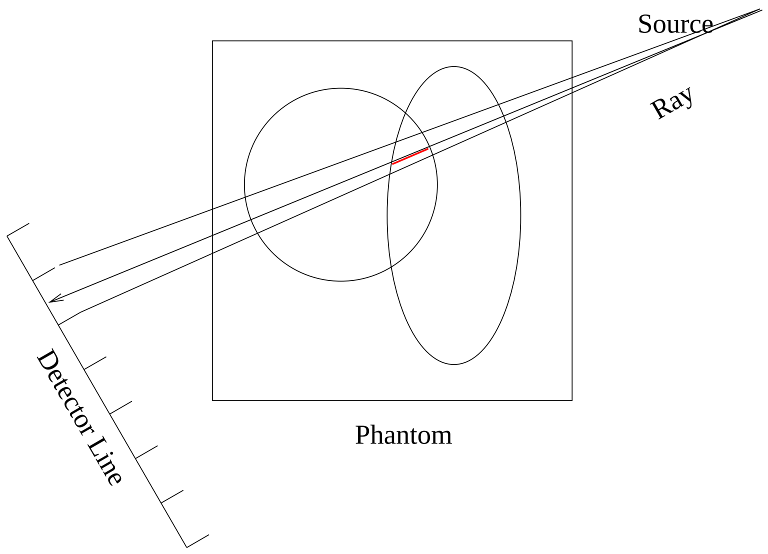


Figure 3.6: Ray segment length as a proxy for area of geometric curves intersected by beam.

To improve accuracy, several ray segments can be measured. (Figure 3.7). The same approach can be done using geometric figures and also yields a more accurate estimate than a single ray segment. (Figure 3.8).

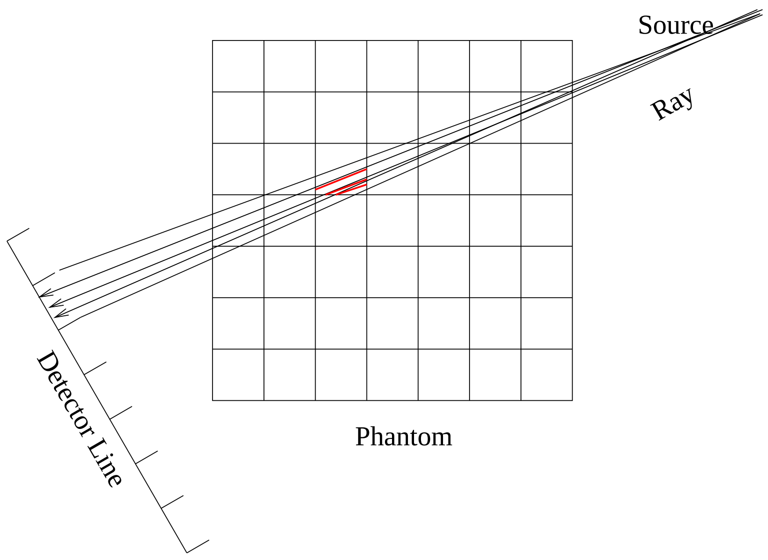


Figure 3.7: Multiple ray segment lengths as a proxy for area of intersection between beam and pixels.

Scanning the images, either rasterized or geometric, using discrete detector arrays results in a discrete data set. Reconstruction results in a rasterized image. Note that in order to compare the reconstructed result to the original phantom image, we must first rasterize the geometric phantom (Figure 3.9).

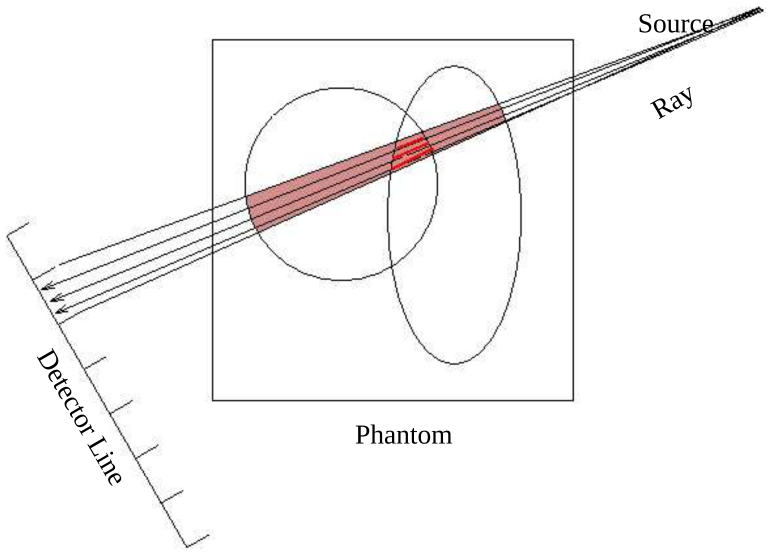


Figure 3.8: Multiple ray segment lengths as a proxy for area of intersection between beam and geometric curves.

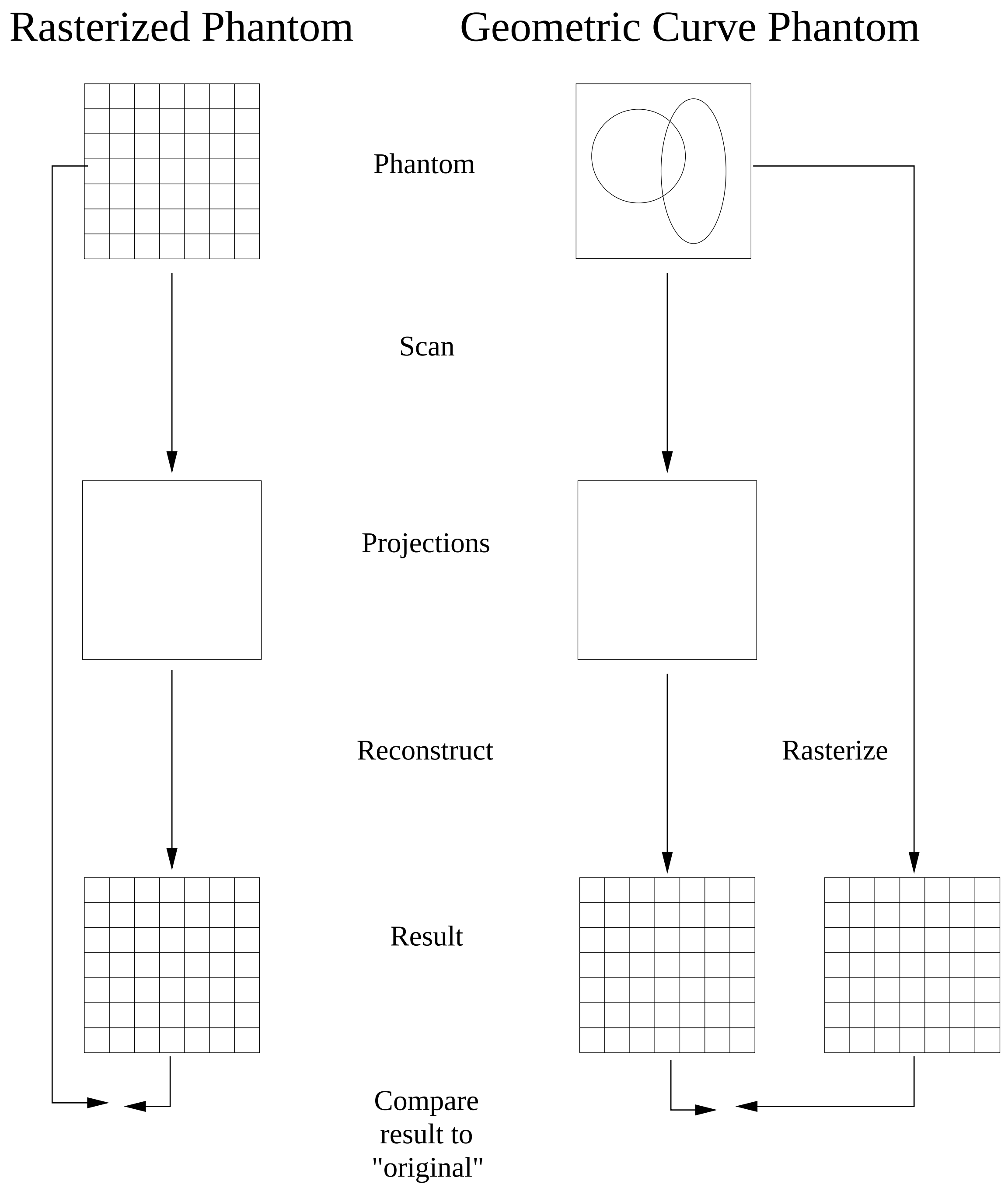


Figure 3.9: The scan, reconstruct, compare cycle for rasterized and geometric scans.

3.3.3 CT Sim

CT Sim [76] is an open source CT simulator written by Kevin Rosenberg. Using CT Sim provides a convenient way to scan images. Because it is open source, the internal structure of image files and other structures are open to inspection and modification, so that compatible code to reconstruct the projections resulting from CT Sim can be written. CT Sim was written in C and subsequently ported to C++. It is described in Table 3.1.

Features of CT Sim	
Feature	Description
phantom images	geometric curves: ellipses, rectangles, sectors
backprojection	centre of pixel projected
reconstruction	filtered backprojection
language	C++
libraries used	Wx Widgets
	MPI
	libpng
	FFT
installation	multi-platform
	automated

Table 3.1: Features of CT Sim

The scanner from CT Sim scans geometric phantoms, using the length of ray segment within a geometric region as a proxy for the area of intersection between the beam and the region. Multiple ray segment lengths can be used to improve this estimate as in Figure 3.8. The default is two rays, but a higher number of rays can be specified.

Many tools included in the suite were also used, and these are listed in Table 3.2.

CT Sim Programs Used	
Program	Purpose
ifinfo	image information
if2	binary functions on two input IF image files
ifexport	export images in viewable form (png)
phm2pj	scan phantom
phm2if	rasterize phantom
pj2if	projections in image form
pjinfo	projection information
pjrec	reconstruct image from projections
CT Sim	most of above with GUI interface

Table 3.2: Programs in CT Sim suite

3.3.4 Virtual Phantoms

CT Sim provides built-in phantoms: the standard Herman and Shepp-Logan phantoms, and the unit pulse phantom which is not a phantom to be scanned, but rather a specific set of detector readings. With the exception of the unit pulse phantom, all CT Sim phantoms are geometric phantoms. CT Sim also provides the capability to create user defined geometric phantoms from a number of shapes.

Unit Pulse Phantom

The unit pulse 'phantom' is not an object or image to be scanned, but rather a condition where the centre detector is turned on, seeing a raysum of 1, while all other detectors are 0. If there are an odd number of detectors, this is the same as a phantom having a narrow spike at the centre of the image; using an even number of detectors makes the resulting projections somewhat discordant. This phantom was used during development to provide a small (7×7 pixels) rasterized image having a centre pixel with value 1 and all others 0 (Figure 3.10).

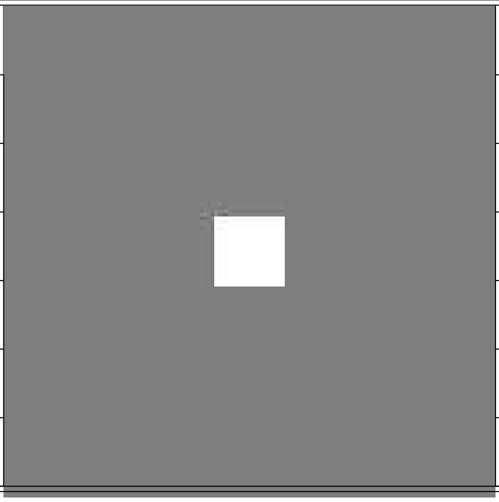
	0	1	2	3	4	5	6		0	1	2	3	4	5	6
0								0	0	0	0	0	0	0	0
1								1	0	0	0	0	0	0	0
2								2	0	0	0	0	0	0	0
3								3	0	0	0	1	0	0	0
4								4	0	0	0	0	0	0	0
5								5	0	0	0	0	0	0	0
6								6	0	0	0	0	0	0	0

Figure 3.10: Unit Pulse Phantom, rasterized to 7×7.

Phantom 'Phm01'

The phantom 'phm01' was created to CT Sim specifications (Figure 3.11). It consists of two circles, and its description is given in Table 3.3. Phm01 was created to have a single non-centre pixel turned on in a 7×7 pixel rasterized image (Figure 3.12).

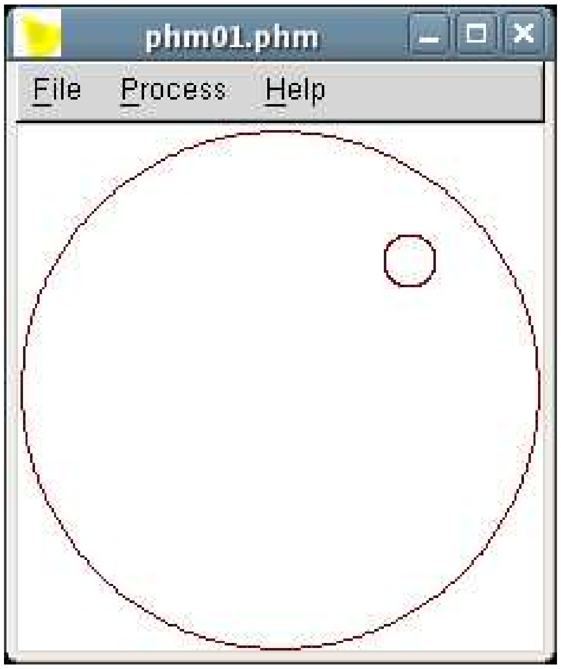
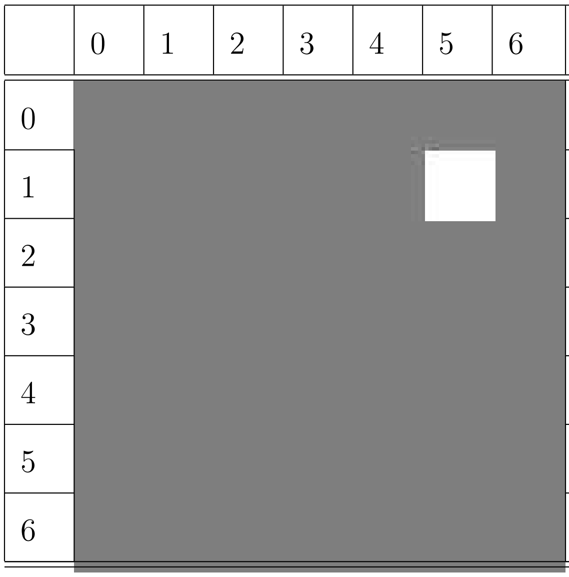


Figure 3.11: Design of Phm01.

From this point forward, all rasterized phantom images or reconstructed images which are 7×7 pixels or smaller will be shown in their rotated form, as in Figure 3.13. CT Sim has followed the usual algebraic convention of listing the x-coordinate of a Cartesian pair before the y-coordinate. Unfortunately, it has also followed the C language convention that the leftmost subscript represents rows (vertical displacement) and the right subscript represents columns (horizontal). This results in the unhappy situation that x is displayed vertically, and y is displayed horizontally in the images.



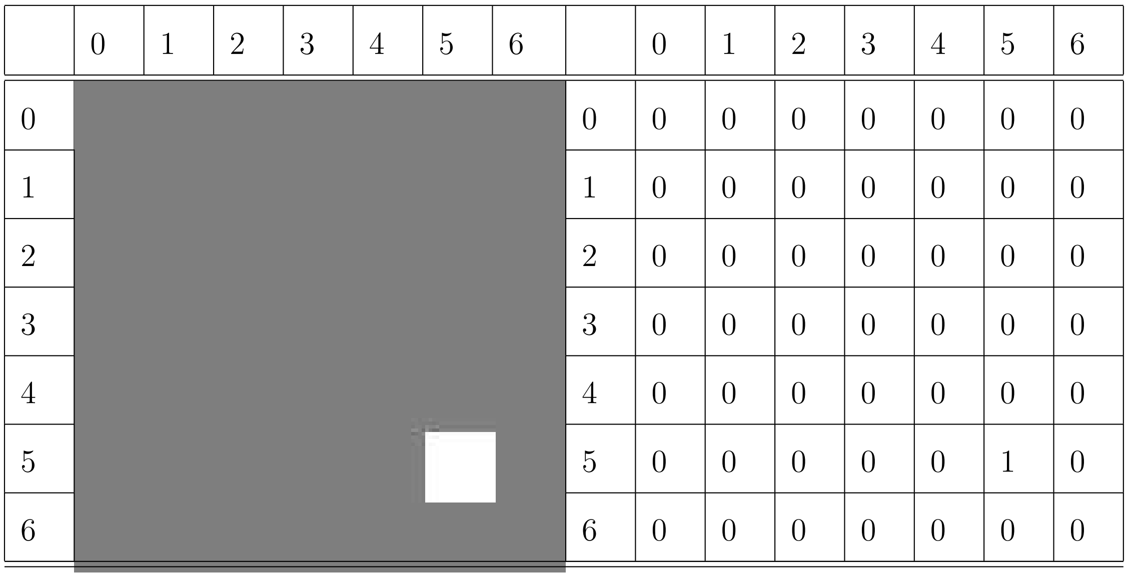
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0

Figure 3.12: Phm01, rasterized to 7×7 as produced by CT Sim.

Hence, some of the CT Sim backprojection code has the *sin* function in an expression where one would normally expect *cos*. This thesis rotates the small test images that have labelled coordinates ninety degrees clockwise to correct for this and restore the axes to their customary Cartesian positions. Built-in phantoms such as the Shepp-Logan have this rotation already taken into account, so these images are not rotated from their expected orientation.

Shepp-Logan Phantom

The standard Shepp-Logan phantom (Figure 3.14) is comprised of a number of ellipses, listed in Table 3.3. A rasterized version is seen in Figure 3.15.



	0	1	2	3	4	5	6		0	1	2	3	4	5	6
0								0	0	0	0	0	0	0	0
1								1	0	0	0	0	0	0	0
2								2	0	0	0	0	0	0	0
3								3	0	0	0	0	0	0	0
4								4	0	0	0	0	0	0	0
5								5	0	0	0	0	0	1	0
6								6	0	0	0	0	0	0	0

Figure 3.13: Phm01, rotated to match Cartesian axes.

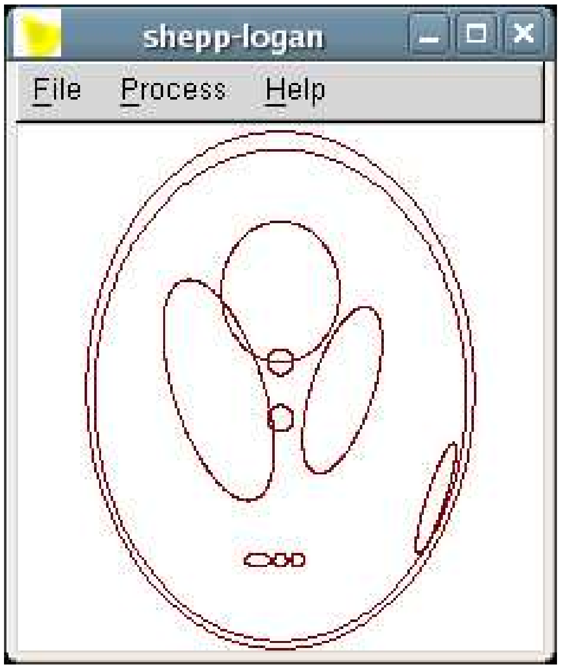


Figure 3.14: Elements of Shepp-Logan.

CT Sim Phantom Elements						
element	cx	cy	dx	dy	r	a
ellipse	centre coordinates (x,y)		major axis length	minor	rotation deg c.c.	*attenuation* μ
Phantom 01						
ellipse	0	0	1	1	0	0
ellipse	0.5	0.5	0.1	0.1	0	4
Shepp-Logan						
ellipse	0.5538	-0.3858	0.033	0.206	-18	0.03
ellipse	0.06	-0.605	0.023	0.023	0	0.01
ellipse	0	-0.605	0.023	0.023	0	0.01
ellipse	-0.08	-0.605	0.046	0.023	0	0.01
ellipse	0	-0.1	0.046	0.046	0	0.01
ellipse	0	0.1	0.046	0.046	0	0.01
ellipse	0	0.35	0.21	0.25	0	0.01
ellipse	-0.22	0	0.16	0.41	18	-0.02
ellipse	0.22	0	0.11	0.31	-18	-0.02
ellipse	0	-0.0184	0.6624	0.874	0	-0.98
ellipse	0	0	0.69	0.92	0	1

Table 3.3: The geometric composition of phantoms used.

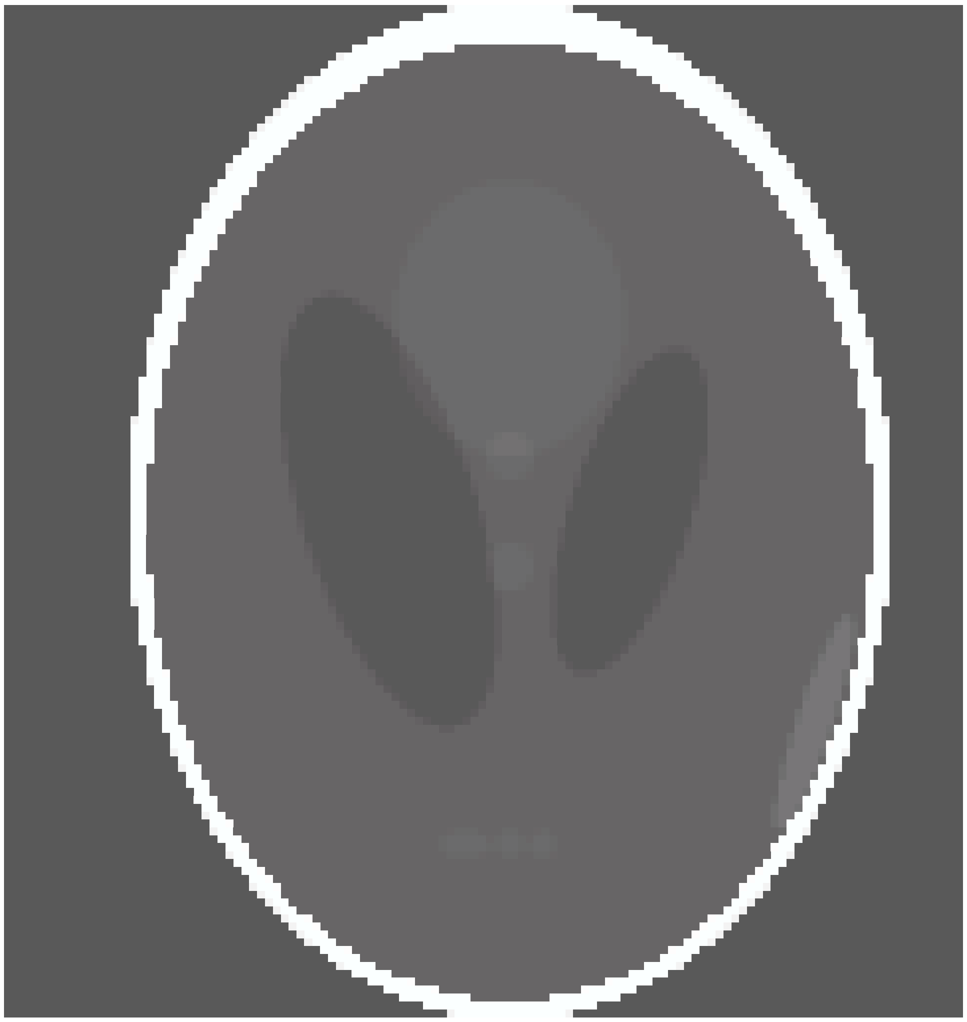


Figure 3.15: Shepp-Logan Phantom, rasterized to 128×128 pixels.

3.4 Specification of Sequential Algorithm

There are many variations of the original iterative algorithm, the *Algebraic Reconstruction Technique* (see Section 2.1.2, page 28). The basic ART algorithm is presented, followed by some variations. All of the discussed variations will be implemented, to allow flexibility during trial runs, and to avoid the pitfalls of poor performance due to a poorly chosen specific algorithm.

ART is our basic choice because it is the forerunner of the iterative methods. It serves as a starting point to discuss related algorithms. A pseudo-code synopsis of the (sequential) ART algorithm is given in Algorithm 2.2. The pseudo-code algorithm iterates over a sparse matrix of weights in the loops specified at steps 3, 5, and 6. The algorithm that is implemented in this thesis represents the rays of the weight table as a data structure containing three arrays: two arrays to store the row and column indices of the pixels having non-zero weight contributions to the ray, and one array to store the weights themselves. The implemented weight table is much smaller, since the many zero weights are not stored. Complexity is reduced when iterating over the weight table, because rather than iterating over n^2 pixels for a $n \times n$ pixel image for each *ray*, we iterate over n^2 pixels for each *view* (there are an *average* of n pixels with non-zero weight per ray and n rays per view).

The ART algorithm is implemented with a number of parameters to be specified, allowing flexible choices for a variety of features of algorithm. Such features as the basic updating mechanism (ART, MART, or WART), the frequency of updating, the termination conditions, the seed image, clipping of values, and various parallel options can be specified. These features are discussed so that the results of the thesis project can be viewed in a proper context.

The correction term to be applied can be modified from the basic ART algorithm to yield other algorithms. In ART (Algorithm 2.2) the correction term $\frac{P_{\theta d} - P_{calc}}{denominator}$ is added to the update array, and subsequently added to the pixel attenuation estimate. When the correction is made, the same term is applied to each pixel that contributes to the raysum. A novel modification to this scheme that we will call *Weighted* ART (WART) first weights this correction factor according to the proportion of the pixel's weight compared to the total weight of all contributing pixels in the raysum, then applies this weighted correction factor to the pixel. *Multiplicative* ART (MART) is a well known refinement to ART using multiplied, rather than added, correction terms. This allows pixels in the periphery which are 'zeroed' to remain zero, which speeds convergence of the remaining pixels.

The timing of updates in the algorithm must also be specified. From Algorithm 2.2, the q superscript of S_{ij}^q in the updating equation indicates that S_{ij}^q is updated every *view*. This is standard for ART, and hence this is the updating schedule shown in the pseudocode. Rather than update the pixel directly, however, the corrections to pixels are added to a separate 'update' image array. The time that the updates are added to the actual image pixels is specified by a parameter. This updating can take place after each ray (though instability may result), after each view, or 'simultaneously' after each cycle as in the *Simultaneous* Iterative Reconstruction Technique (SIRT). Exactly one of these schedules is chosen. After the update array element is added to the pixel estimate (or multiplied, in the case of MART) the update array element is set to zero (or to one, in the case of MART).

The denominator for the correction factor, identified in Algorithm 2.2 as '*denominator*', is also specified. The denominator may be specified as: the total number of relevant pixels; the total *weight* of relevant pixels; or the constant value 1. If the latter is chosen, then a low relaxation factor should be chosen to avoid excessive oscillations. The option to specify a ray's length within the image as the denominator is not implemented at the time of this writing.

The termination conditions are to be specified by the user. The ART example iterates K times. This maximum number of iterations is specified. Error is calculated as the sum of the magnitudes of the difference between the calculated error and the measured error, $|P_{\theta d} - P_{calc}|$, for the entire iteration. If updating takes place more frequently than once every iteration, the total error changes as the calculation is taking place. The option is available to do a separate iteration specifically for error calculation. The implementation allows for specification of an error tolerance, below which iterations cease, and also specification of a maximum number of 'non-converging iterations'. This number could be specified as 1, meaning that termination occurs as soon as one iteration has a total error greater than or equal to the previous iteration's error. However, it is possible that after several iterations without convergence the program might 'back out' of a local minimum and resume convergence toward a new, more global error minimum. For this reason a higher maximum number of non-converging iterations may be specified.

One must be able to specify the starting 'seed' image. This can be specified as a number, to which every pixel in the image will be set (0 is a good choice for non-multiplicative algorithms), or to random numbers between 0 and 1, or to a specified image file. The original seed image is destroyed as the algorithm modifies the image with new estimates.

Guan and Gordon [37] show that the order in which views are accessed in an iterative algorithm can have a dramatic impact on the rate of convergence. *Multi-level Sequencing* (MLS),as described in their paper, may be specified in place of sequential ordering. An algorithm to carry out this sequencing is created in the work of this thesis.

Other choices include an option to clip values below a certain bound and a choice of relaxation factor (α). If clipping is specified, a lower bound is selected and any attenuation estimate less than that bound is set to the value of the bound. Typically, zero is chosen in order to eliminate negative attenuation estimates. A relaxation factor (α) less than 1.0, such as 0.8 or 0.6, may be chosen to dampen oscillations that might be present.

A flexible implementation has been created that allows selection from a number of options. These options are not only parameters that affect some aspects of the algorithm's behaviour, but also specify the actual algorithm to be used.

3.5 Specification of Parallel Algorithm

The architecture on which we implement the algorithm is a cluster of workstations, an example of a distributed memory system, classified as a MIMD (Multiple Instruction stream Multiple Data stream) system in Flynn's classification [25, section 1.2]. The approach taken in this thesis is that of a *Single Program Multiple Data* (SPMD) model [25, section 2.1]. Each processor executes (approximately) the same program simultaneously. This model is also known as *data parallelism* because a separate partition of the data flows to each of the processors in parallel.

```
Require: Distribution of the projection data and seed image to each of $p$ processors
1: Each processor $P_i$ reads its own portion of the weight table, or reads the entire
   table, filtering it to keep what is relevant. Each $P_i$ will be assigned a weight table
   which is approximately $w = \frac{W}{p}$ (in cases of non-overlapping weight tables)
2: repeat
3:    repeat
4:       Each processor $P_i$ performs the CT reconstruction algorithm on its own image
         but iterating over weight table $w$
5:    until $x$ iterations or until local convergence criteria are met (for example,
      further iterations produce no change within a specified tolerance)
6:    barrier
7:    Each processor $P_i$ communicates its image data to the root processor
8:    The root processor combines the image solutions, and communicates the com-
      bined image data to each processor $P_i$
9: until a global convergence criterion is met
```

Algorithm 3.2: Parallel Iterative Algorithm.

Algorithm 3.2 is a pseudo-code synopsis of the parallel iterative algorithms for a 2-D image. Iterations for any iterative CT reconstruction could be used in step 4 of Algorithm 3.2. In general, this algorithm is *synchronous*, meaning the processors are executing the same code at the same time. This is ensured by the barrier at step 6 (processes must block until they all reach the barrier). The number of iterations of the global outside loop (step 2) is the same for each processor, and is dictated by a decision within the root process at step 9. The inner loop (step 3) however, may be executed a different number of times from one processor to the next (asynchronous). Since all communication between processors takes place in the global loop, the algorithm is synchronous.

If the step 4 algorithm is updated at the end of the iteration (simultaneously) and the image from each processor is combined by adding, then the same results are obtained as if the algorithm were done sequentially on a single processor. In general, however, this is *not* the case and the sequential algorithm yields different results from the parallel algorithm. It cannot be emphasized strongly enough that the parallel algorithm is not the same algorithm as the sequential algorithm, and results differ depending on the number of processors used.

The implementation allows specification of several parameters for termination of the parallel algorithm (Algorithm 3.2). The exit conditions for step 3 of the parallel algorithm are exactly as specified for termination of the sequential algorithm in section 3.4. The termination conditions for the outer loop, step 2 of the parallel algorithm, are specified separately but are analogous to the termination conditions for step 3. An error tolerance, below which iterations cease, may be specified for the outer loop of the parallel algorithm. Also similar to the sequential algorithm is the specification of a maximum number of iterations, and a maximum number of 'non-converging iterations'. The algorithm iterates through the loop starting at step 3 until a convergence criterion in step 5 is met. Synchronization then takes place, and the processes communicate their image data to the root and receive back an updated global image. Because this communication updates the image, the variables determining the loop exit conditions for step 3 may have changed. If the program is not terminated by the outer loop at step 9, then the inner loop is entered again at step 3 and the exit criteria is reevaluated. The specification of global criteria as well as local criteria allows for a more exact manipulation of the algorithm.

3.6 Data Partitioning

Using the SPMD approach, a major design task is deciding how to *partition* the data. In this case, the data to consider is the large series of tables of weights, the weighting factor for each image pixel on the final raysum. In general, efficiency is maximized by dividing the data equally among p (identical) processors. Each processor is labelled P_i, where $i = [0..p-1]$. The weight table, containing W weights, is divided among each of p processors so that each processor has a weight table of approximately $w = \frac{W}{p}$ weights in its local memory. If it takes time T for a processor to iterate through the weight table of W weights, then the same processor will iterate through the table of w weights in time $t = \frac{T}{p}$.

This thesis also investigates some strategies of overlapping data among the processors so that $w > \frac{W}{p}$ and $t > \frac{T}{p}$. The immediate increase in t is not desirable, but in some situations this is balanced by a decreased work of communication of data between processors.

The main differences between this and Melvin's work [56] are that a number of iterations is done on the data within a processor's memory so that data is not communicated between processors at every iteration; and the weight tables are partitioned by image pixels (tiling) in addition to by projection. Four partitioning methods are studied in this thesis:

view by sequence: V views are distributed to p processors so that each processor receives $\frac{V}{p}$ consecutive views in its local weight table;

view by round robin: Views are distributed to each processor in turn until all are distributed, so that each processor receives $\frac{V}{p}$ widely separated views in its local weight table;

local tile: Individual rays are assigned to the weight tables of each tile, and these rays are modified to refer only to pixels within the tile;

ray by tile: Individual rays are assigned to the weight tables of each tile, and these rays may access pixels outside of the tile.

The design decision of how to *partition* the data is crucial. This thesis examines more than one scheme to partition the weight tables. The simplest and most direct method is to partition according to view. Since iterative algorithms typically iterate through views, rays within views, and finally contributing pixels within rays, the simplest way to structure the weight table is to create an array of views, each containing an array of rays, each ray being a structure that contains the pixels and their weights for that raysum. The views, then, are the root of the weight table structure, and distributing the weights by assigning views to specific processors is straight forward.

Evidence of spatial locality in the final image (discussed on page 43), indicates a possible method of partitioning according to spatial location in the image. An entire image is divided into a grid, each part of the grid being computed as a separate 'region of interest'. There are a variety of issues to consider with this *tiling* approach. Hexagonal tiling or overlapping circular tiling might provide improved locality. However, the scope of this paper is limited to a particular rectangular tiling, the rectangles being made as square as the image geometry and number of processors allow.

The underlying assumption is that two pixels within a given tile are more closely related to each other than two pixels chosen from two different tiles. This is less true of pixels at the boundaries between tiles. Overlapping tiles might provide a way to deal with border pixels, so I also examine overlapping tiles.

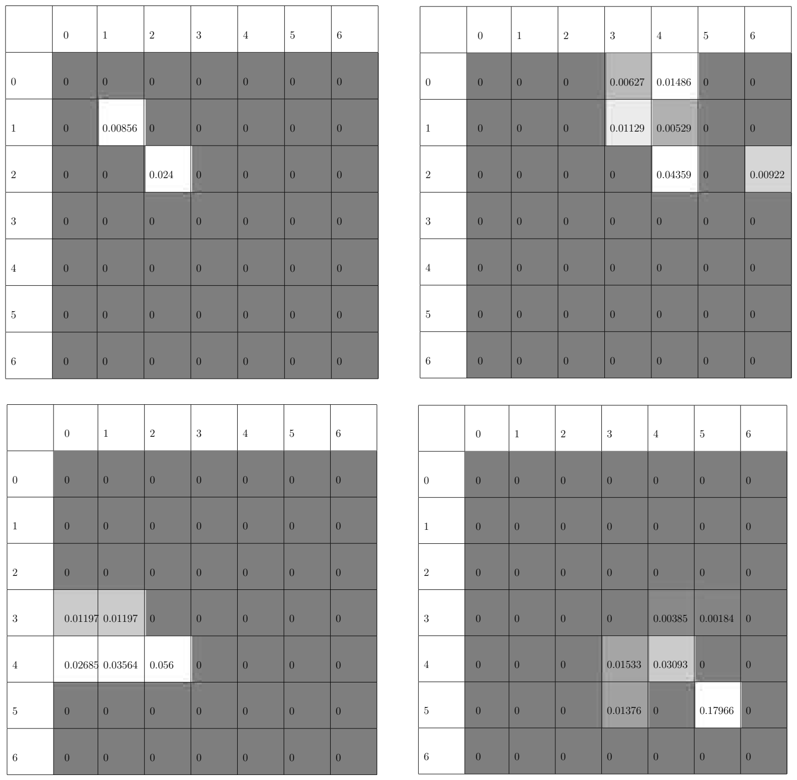
The input data is the collection of projection data. This data can be partitioned, but it is not necessary given the small size of the data relative to the weight tables; the entire projection data can be distributed to each processor. The data that needs to be partitioned is the weight table. Every view from the weight table includes a given tile, hence all views are represented in a given processor's weight table. Within each view, rays that do not intersect the processor's tile are discarded. Rays that intersect the tile are handled differently, depending on the specific tiling method.

Two main methods of tiling are implemented. Using the first method, rays that intersect the tile are 'broken'. The piece of the ray that has pixels only inside the tile is retained. The remaining piece, containing references to pixels outside of the tile, is discarded.

An obvious question arises about what to do with the projection data. Even if the pixel values from the retained piece of a ray exactly match the corresponding pixels of a rasterized phantom, the calculated raysum may be very different from the measured raysum due to missing contributions from pixels in other tiles. An adjustment could be made to either the measured or calculated raysum based on the proportion of the length, number of pixels, or sum of pixel weights compared to the complete ray. Further adjustments could be made depending on the average pixel values of the contributing tiles, which could be calculated prior to partitioning. There are assumptions about all of these adjustments that may not hold true, and so this method makes no adjustments to the raysum. This first method of tiling makes a strong assumption regarding spatial locality. and is referred to in this thesis as *local tiling.*

An example of this method using small sized 7×7 pixel test images is shown in Figure 3.16. All of the small images presented in this chapter are reconstructions of the phantom Phm01 (Figure 3.13 on page 65), taken from projection data that contains 60 evenly spaced views 3 degrees apart in one half of a rotation, using a line of 7 detectors and parallel geometry. Reconstructions were done using ART and a seed image of all zeroes. Figure 3.16 shows that all modifications made to the image, *i.e.* all non-zero pixels, are local to the tile. For example, tile (0, 0) in the top left corner of the figure includes pixels having (x, y) coordinates (0–2, 0–2). The only pixels changed from the zero background are pixel (1, 1) and pixel (2, 2), pixels within the tile. Likewise, tile (0, 1) (coordinates (0–2, 3–6)), tile (1, 0) (coordinates (3–6, 0–2)), and tile (1, 1) (coordinates (3–6, 3–6)) modify only pixels within their respective tiles, using image pixel values from only within their tiles.

The final result is achieved by simply 'stitching together' the tiles, shown in Figure 3.17. It is similar to the sequential solution shown in Figure 3.18, but contains greater error.



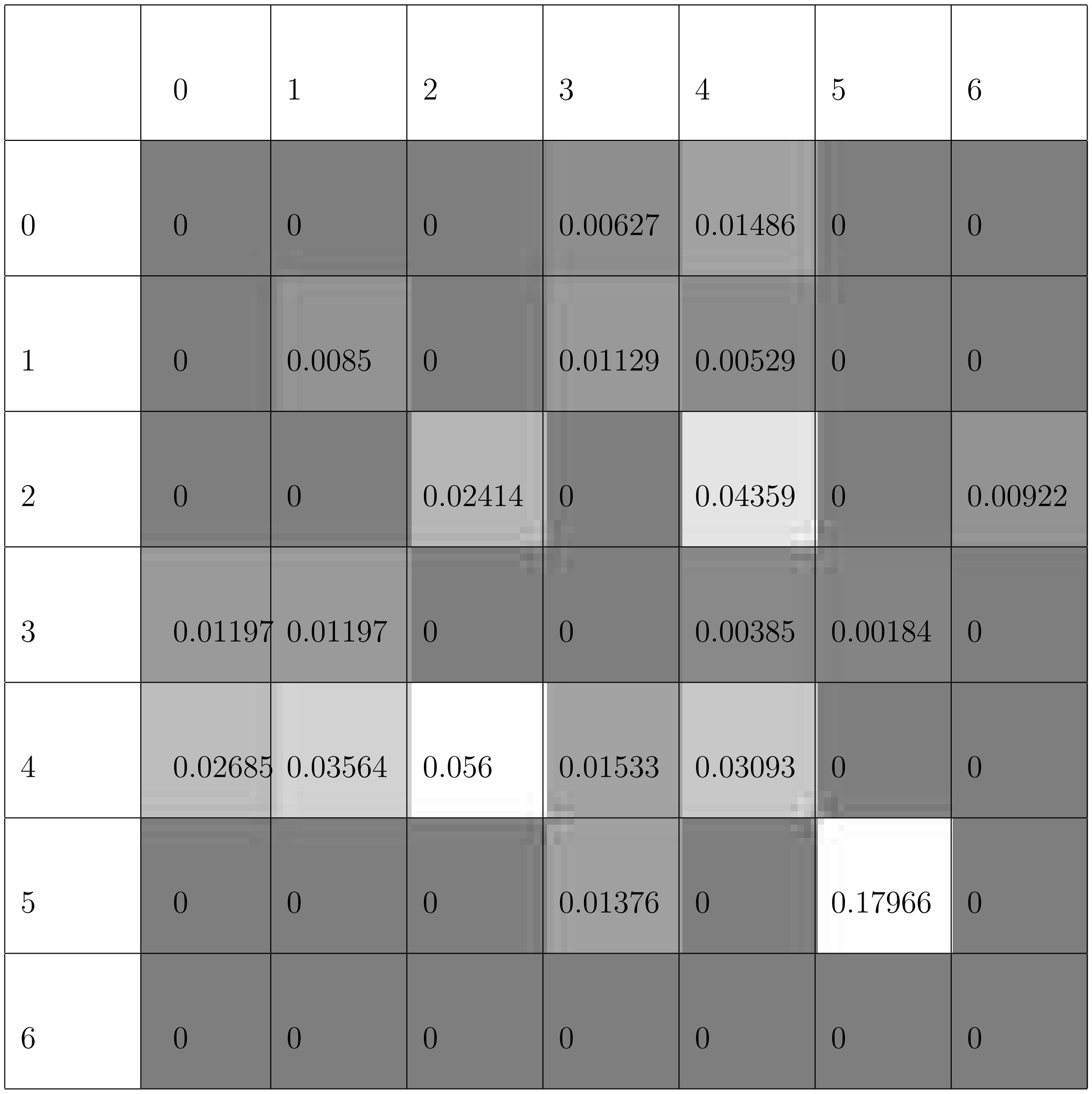
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0.00856	0	0	0	0	0
2	0	0	0.024	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0	0	0	0	0.00627	0.01486	0	0
1	0	0	0	0.01129	0.00529	0	0
2	0	0	0	0	0.04359	0	0.00922
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0.01197	0.01197	0	0	0	0	0
4	0.02685	0.03564	0.056	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

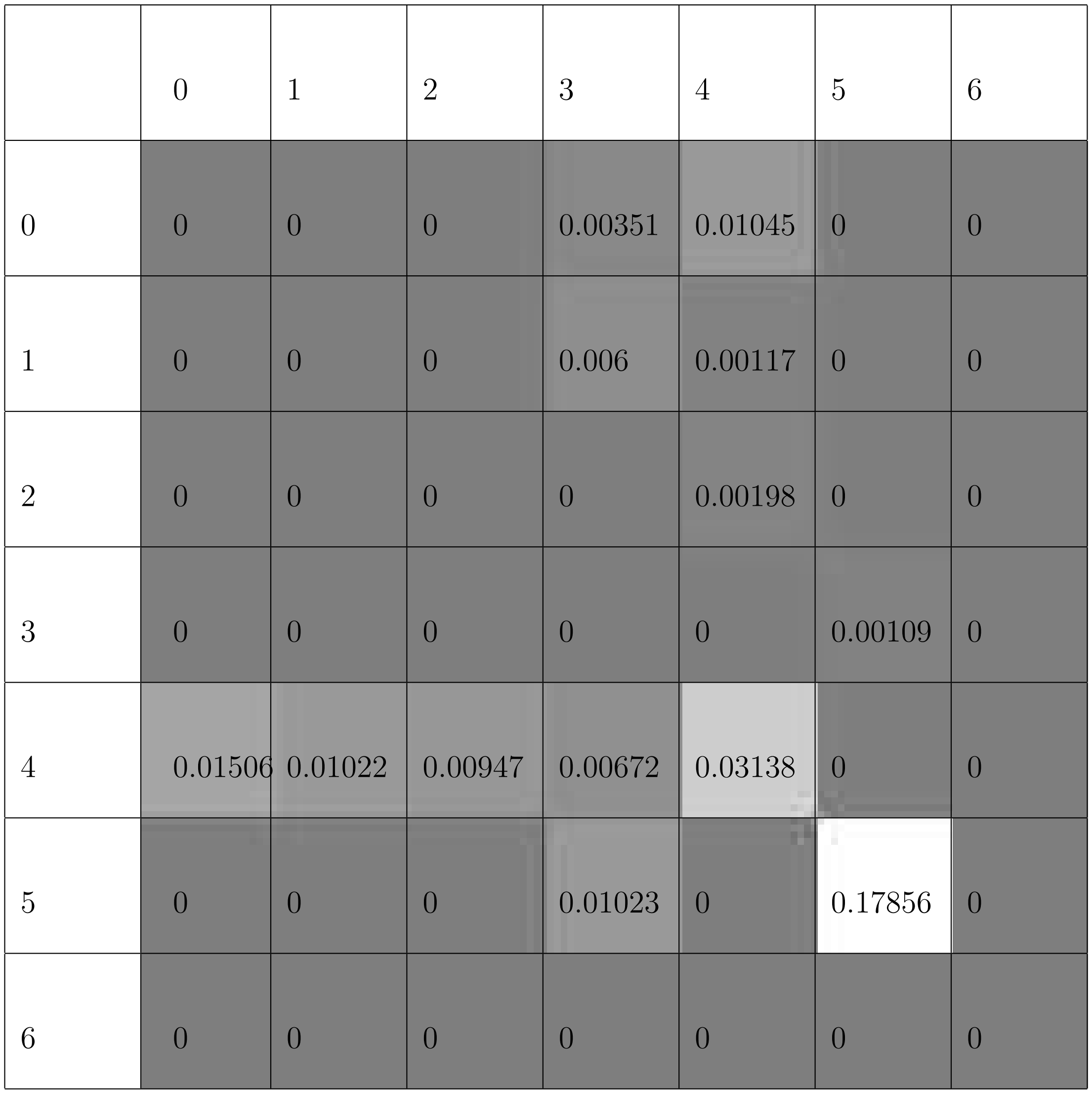
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0.00385	0.00184	0
4	0	0	0	0.01533	0.03093	0	0
5	0	0	0	0.01376	0	0.17966	0
6	0	0	0	0	0	0	0

Figure 3.16: Local images produced by each partition, local tile method.



	0	1	2	3	4	5	6
0	0	0	0	0.00627	0.01486	0	0
1	0	0.0085	0	0.01129	0.00529	0	0
2	0	0	0.02414	0	0.04359	0	0.00922
3	0.01197	0.01197	0	0	0.00385	0.00184	0
4	0.02685	0.03564	0.056	0.01533	0.03093	0	0
5	0	0	0	0.01376	0	0.17966	0
6	0	0	0	0	0	0	0

Figure 3.17: Final image, local tile method, 4 processors.



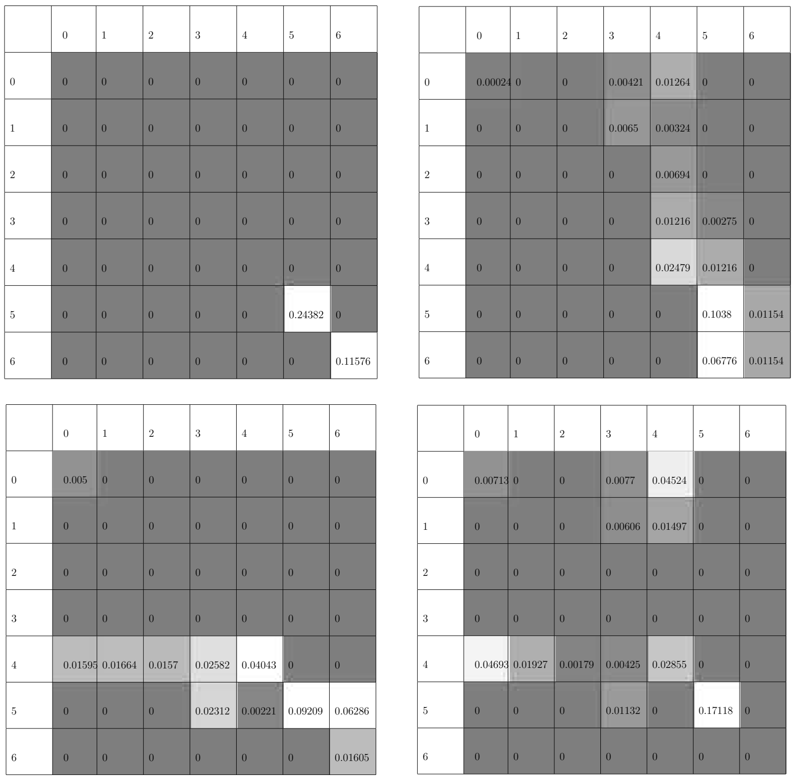
	0	1	2	3	4	5	6
0	0	0	0	0.00351	0.01045	0	0
1	0	0	0	0.006	0.00117	0	0
2	0	0	0	0	0.00198	0	0
3	0	0	0	0	0	0.00109	0
4	0.01506	0.01022	0.00947	0.00672	0.03138	0	0
5	0	0	0	0.01023	0	0.17856	0
6	0	0	0	0	0	0	0

Figure 3.18: Final image, sequential version.

A second method of tiling is examined in which rays are not ‘broken’ during the weight table partitioning. If a ray intersects t tiles, then the ray is represented in its entirety in the weight table of each of the t tiles. This ‘softens’ the assumption of spatial locality, as each tile now references data (pixels) outside of itself. This second tiling method, in which entire rays are distributed this thesis refers to as *ray by tile*.

There are two concerns with regard to this partitioning method. The first is that a ray that intersects t tiles is represented t times, creating duplication from a global perspective of the reconstruction. Such rays may be overweighted in the final reconstruction. A second concern is that this duplication of information increases the size of the weight tables for each processor (and hence the time to iterate through them).

An example using small sized test images is shown in Figure 3.19. Estimates for pixel values of pixels outside of the tile occur in all partitions, and this is where the error tends to be maximal, *i.e.*, the estimate made for pixel (6,6) by the partition containing tile (0,0), the tile on the top left of the figure. The error within the tile may be lower than for the local tile method. The pixel value estimates for the 9 pixels of tile (0,0) are 100% accurate in this case.



	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0.24382	0
6	0	0	0	0	0	0	0.11576

	0	1	2	3	4	5	6
0	0.00024	0	0	0.00421	0.01264	0	0
1	0	0	0	0.0065	0.00324	0	0
2	0	0	0	0	0.00694	0	0
3	0	0	0	0	0.01216	0.00275	0
4	0	0	0	0	0.02479	0.01216	0
5	0	0	0	0	0	0.1038	0.01154
6	0	0	0	0	0	0.06776	0.01154

	0	1	2	3	4	5	6
0	0.005	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0.01595	0.01664	0.0157	0.02582	0.04043	0	0
5	0	0	0	0.02312	0.00221	0.09209	0.06286
6	0	0	0	0	0	0	0.01605

	0	1	2	3	4	5	6
0	0.00713	0	0	0.0077	0.04524	0	0
1	0	0	0	0.00606	0.01497	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0.04693	0.01927	0.00179	0.00425	0.02855	0	0
5	0	0	0	0.01132	0	0.17118	0
6	0	0	0	0	0	0	0

Figure 3.19: Local images produced by each partition, ray by tile method.

Because the assumption of locality is not as valid at tile borders, increased error along the borders between tiles is anticipated. This is confirmed in some test images. By overlapping the tiles, the area for pixel modification is extended by at least one pixel, and the error accumulates in the new extended border region. Only the original area of the tile contributes to the reconstruction, and the less accurate pixels in the overlapped area are discarded. The overall error rate is thus improved.

To specify overlapping, an 'overlap' parameter ω from 0 to 100 is specified for the tiling methods, to indicate an overlap of between 0% and 100%. The tile is expanded $\omega/2$ % of its width in each direction that does not border the boundary of the image. Hence, a square tile which is not on the image boundary has its area expanded to 4 times the previous area. Iterations and pixel updates (local to the processor) occur over the larger area, but only the original tile's non-overlapping pixels are communicated to the root.

Four main partitioning methods are implemented, a single method being chosen at run time. The four basic methods to choose from are view by sequence, view by round robin, local tile, and ray by tile. In addition, the previously described overlap parameter is specified for the two tiling methods.

Chapter 4

Results

Projections of the Shepp-Logan phantom are taken using the default settings of CT Sim's *phm2pj* tool. All projection datasets are comprised of 180 evenly spaced views in a half rotation. Each view consists of parallel rays striking a line of evenly spaced detectors. Attenuation values are clipped at zero, updates are applied after each view, the denominator of the correction factor is the sum of weights of all pixels on the ray, and the relaxation factor is 1. Runs are performed to reconstruct the Shepp-Logan phantom using sequential ART, and parallel ART using 1, 4, 8, 9, 16, and 25 processors. Some images produced by MART and WART are also viewed.

Partitioning methods included: view by sequence; view by round robin; local tile with no overlap; local tile with 50% overlap; ray by tile with no overlap; and ray by tile with 50% overlap.

Images of size 64×64 pixels, 128×128 pixels, and 256×256 pixels were reconstructed using projections taken from 64, 128, and 256 detectors respectively. Results for reconstructions of size 64×64 pixels are shown in Appendix A and results for reconstructions of size 256×256 pixels are shown in Appendix B. The results of runs shown in this chapter, unless otherwise indicated, are performed on 128×128 pixel images.

Results are shown using a maximum of 2 iterations between interprocess communications, and an overall maximum of 8 iterations of the parallel cycle (hence an overall maximum of 16 iterations of the inner loop). The exception to this is Figure 4.5, that shows 10 iterations between communications. Iterations cease when there is no longer convergence (the sum of errors for the entire iteration does not decrease from the previous iteration).

The results, shown in the remainder of this chapter, are organized by image quality, speed and scalability to the number of processors, and scalability to the problem size. A summary follows, to indicate which partitioning methods show best overall performance.

4.1 Image Quality

Reconstructed images are compared to a rasterized version of the original phantom as in Figure 3.9. A visual inspection of the reconstructed images is important to evaluate the reconstruction algorithms, and perhaps to gain insights into the failings of certain algorithms. However, we also need a more objective measure to compare reconstruction results from various algorithms and parameter choices.

In the case of production CT, we want to replicate even some of the more subtle structures. A number of trained examiners could evaluate various reconstructions of phantoms or actual clinical scans for the presence or absence of diagnostic features, and aggregate scores could be utilized to reduce the effect of observer bias. Another objective means of evaluating the structure would involve the use of intelligent computer programs to detect the preservation of structures in the reconstructed image. Both of these methods are resource intensive, and beyond the scope of this thesis.

In general, the reconstructed images from this thesis lack the quality to be used in a production scanner. Production quality, however, was not the intent of this thesis. The intent of this thesis is to demonstrate that the algorithms are actually reconstructing the phantom, and to show that the parallel algorithm reconstructions are of similar quality to the sequential reconstruction.

Distance measures between corresponding pixels of two images (in this case the reconstruction of a phantom and the rasterized image of the same phantom) are sensitive to changes in scaling, translation, or rotation, though these transformations should not occur in reconstruction.

Some of these measurements are sensitive to differences in the scale of image intensity or image size, *e.g.* a Euclidean distance is larger for non-identical images if the range of intensity values is larger. These measures do not provide a consistent scale to compare images intuitively. To eliminate these two problems, this thesis computes the *image correlation coefficient* between the reconstruction and the rasterized phantom images. The image correlation coefficient for an image $f(x, y)$ and a sub-image $w(x, y)$ is given by Gonzalez and Woods [28] as:

$$\gamma(x,y) = \frac{\sum_s \sum_t [f(s,t) - \overline{f}(s,t)][w(x+s,y+t) - \overline{w}]}{\sqrt{\sum_s \sum_t [f(s,t) - \overline{f}(s,t)]^2 \sum_s \sum_t [w(x+s,y+t) - \overline{w}]^2}} \tag{4.1}$$

This formula is used to find the best match for sub-image w within the image f.

The sub-image w is placed with its origin over position (x, y) in the image, then each pixel $(x+s, y+t)$ of the sub-image is correlated to the underlying pixel in the image. The resulting correlation coefficient γ at point (x, y) is normalized to range between -1 and +1 so that the actual scale of the image intensity is not important, only the corresponding *differences* in value. Coefficients are found for each possible position of the sub-image within the image, and the maximum of the resulting $\gamma(x, y)$ indicates the best match.

This thesis compares two identically sized images (the reconstruction and an equally sized rasterized phantom), iterating over x and y once only, producing a single correlation coefficient:

$$\gamma = \frac{\sum_x \sum_y [f(x,y) - \overline{f}(x,y)][w(x,y) - \overline{w}]}{\sqrt{\sum_x \sum_y [f(x,y) - \overline{f}(x,y)]^2 \sum_x \sum_y [w(x,y) - \overline{w}]^2}} \tag{4.2}$$

This coefficient is a value between 0 (indicating no correlation) and ± 1 indicating a perfect (positive or negative) correlation. The coefficient γ is equivalent to r, the *Pearson Product Moment Coefficient of Correlation*[57, page 433]. The Pearson correlation is the primary measurement of reconstruction quality used in this thesis. When referring to the 'Pearson correlation' or 'Pearson coefficient' as a measure of a reconstructed image's quality, it should be understood that we are referring to the reconstructed image's correlation with the corresponding rasterized phantom.

As seen in Figure 4.1, image quality, by the primary measure of Pearson correlation between the reconstruction and the phantom rasterized to the same granularity, declines with increasing numbers of processors. This is to be expected when the problem solution is initiated without communication. In the extreme case, when a processor can iterate over only one view (in the case of view partitioning) or one pixel (in the case of tile partitioning), no reasonable solution can be found.

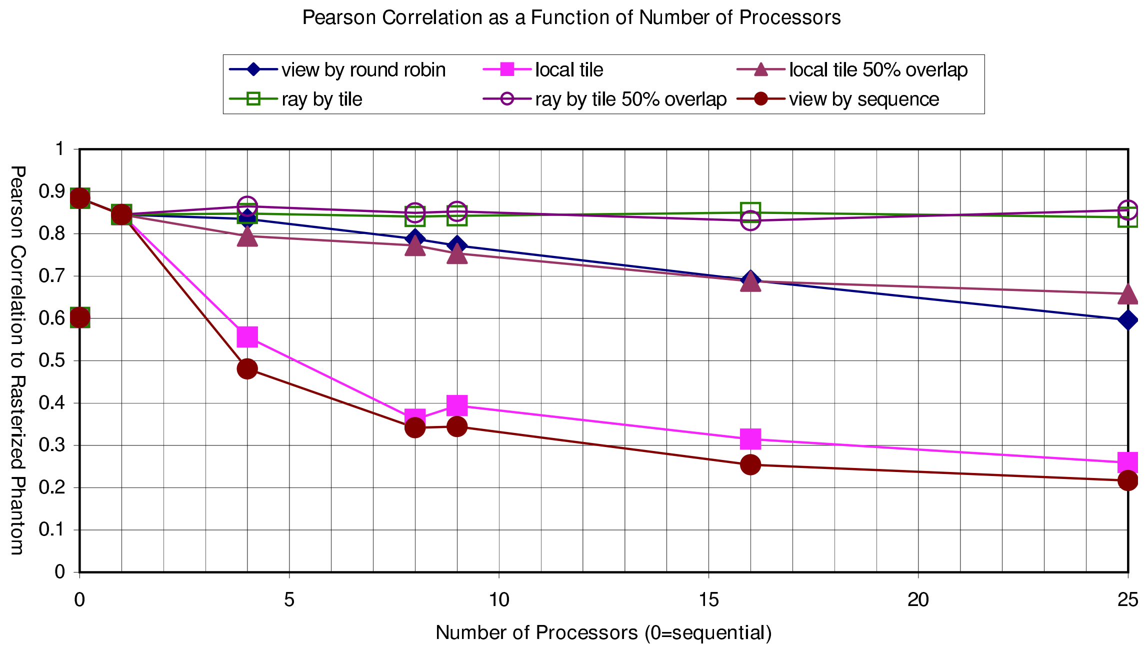


Figure 4.1: Image quality *vs.* number of processors.

The Pearson coefficient between an unfiltered backprojection, shown in Figure 4.2, and the rasterized Shepp-Logan phantom, shown in Figure 3.15 of Section 3.3.4, is approximately 0.6 and is marked on the Y axis of Figure 4.1. Let us agree that any reconstructions that achieve a Pearson correlation less than this do not demonstrate a functioning reconstruction algorithm.

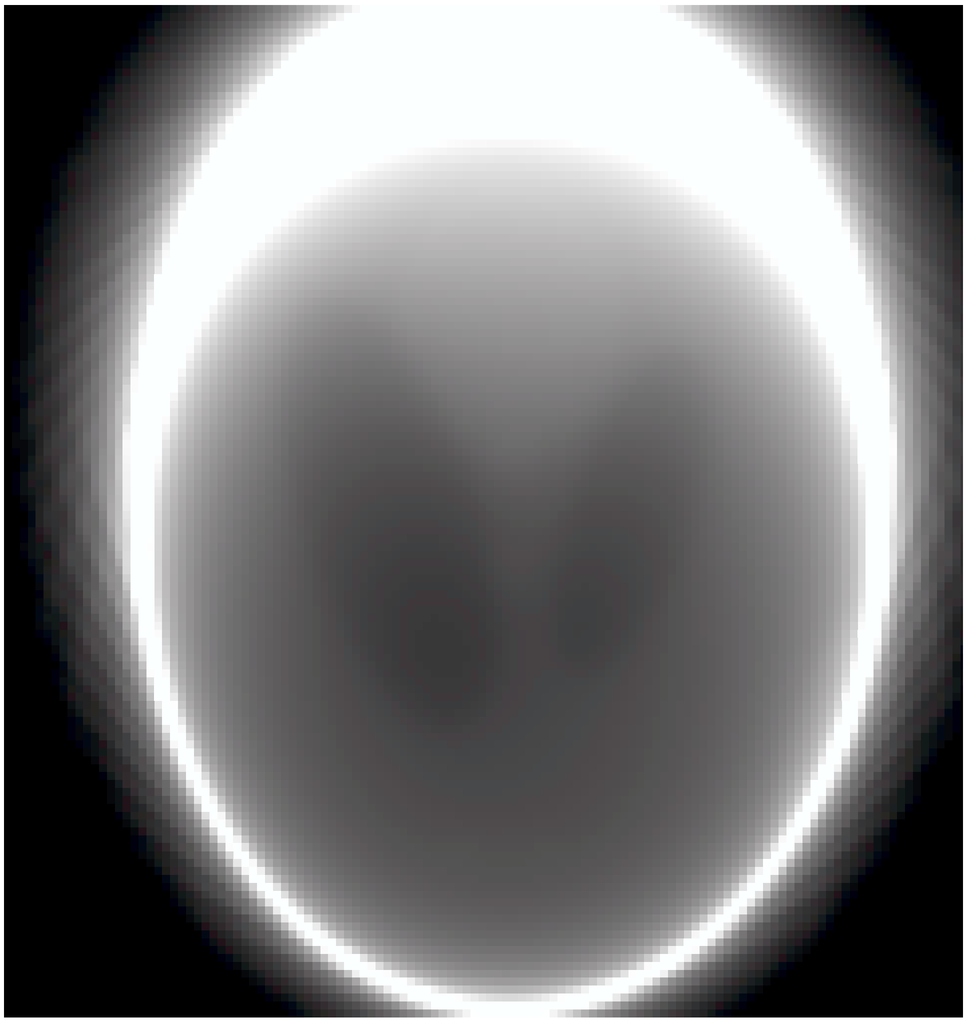


Figure 4.2: Unfiltered backprojection, 128×128 pixels.

In the cases of view partitioning by sequence and local tiling without overlap, image quality deteriorates quickly as processors are added. Using 4 processors, the Pearson coefficient between these images and the phantom is less than that of the unfiltered backprojection. Most observers would agree that the quality of images for view by sequence partitioning (Figure 4.3) and for local tiling without overlap (Figure 4.4) is unacceptable.

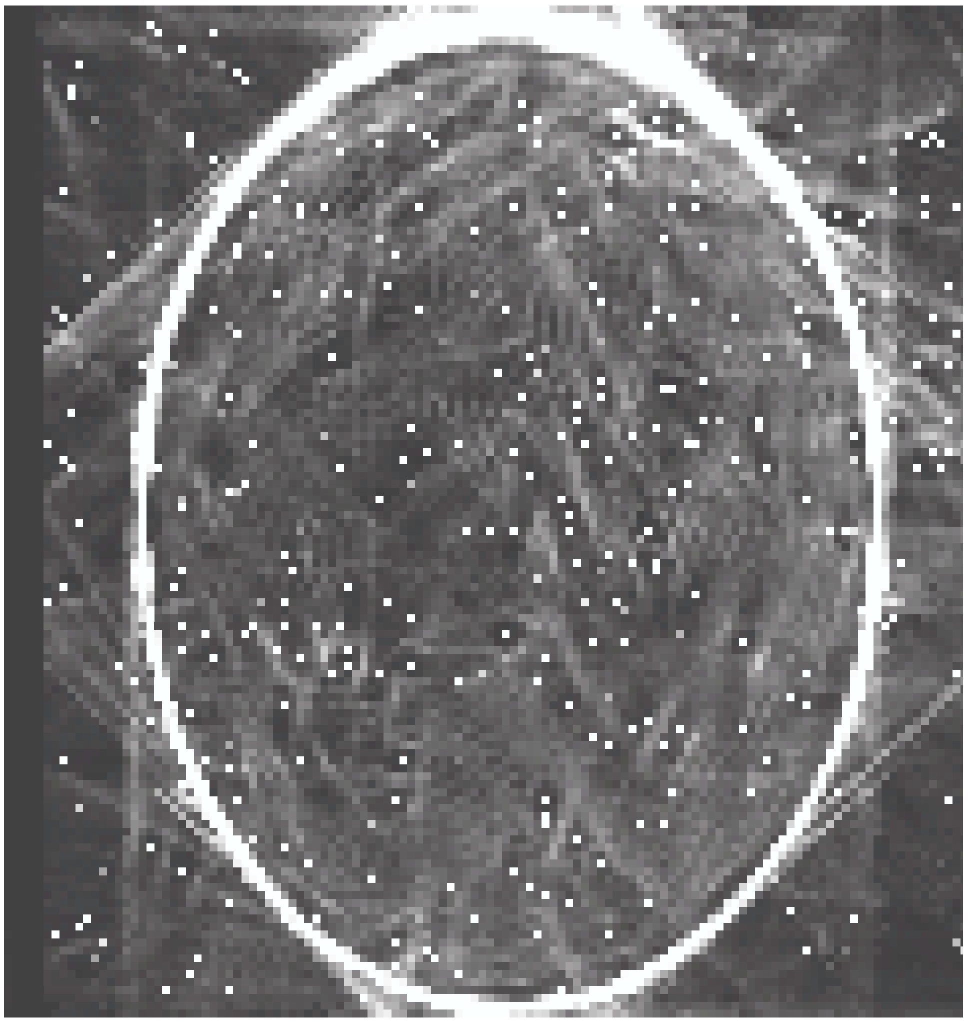


Figure 4.3: Reconstruction by sequential views, 4 processes.

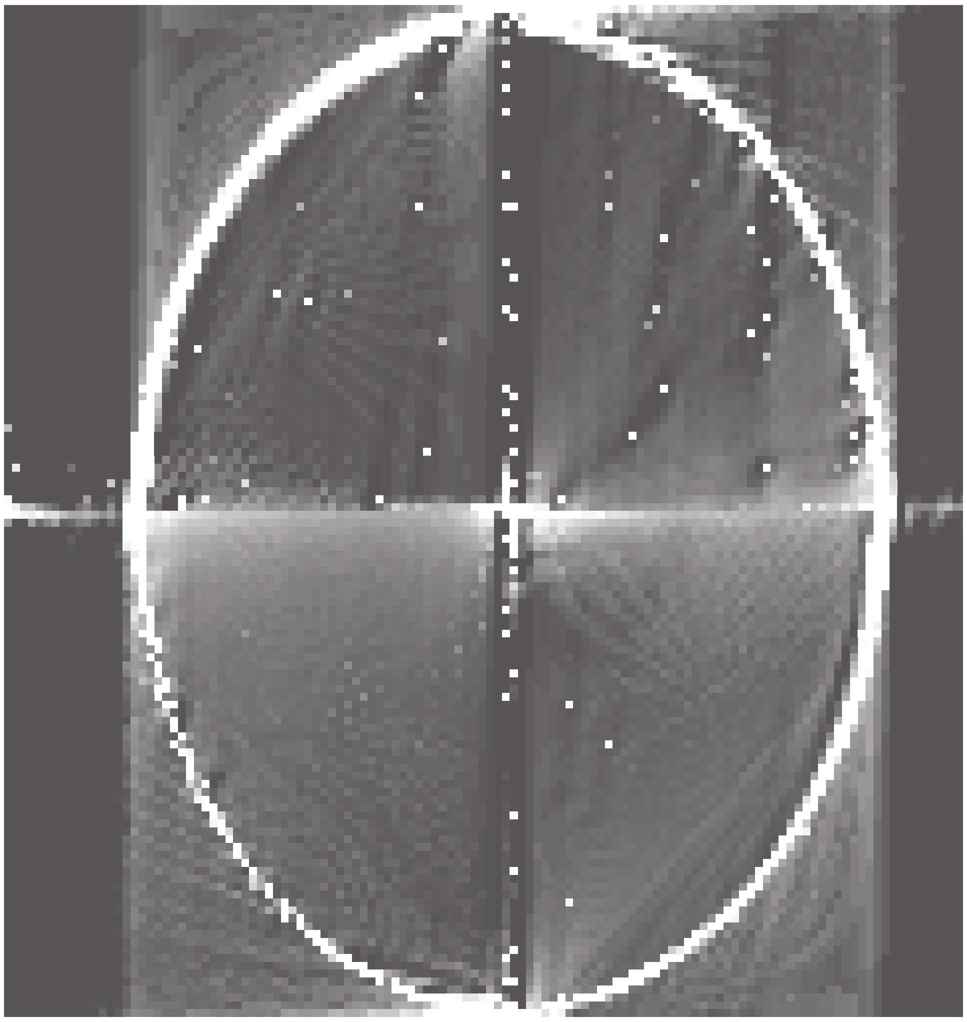


Figure 4.4: Reconstruction by local tile, no overlap, 4 processes.

One might expect that a partitioning method such as local tiling, designed to function without interprocess communication, would perform better than other methods when communication is less frequent. Measurements were taken using a maximum of 10 iterations between interprocess communications, and an overall maximum of 6 iterations of the parallel cycle (generally there was convergence after 1 parallel cycle). Surprisingly (see Figure 4.5), the view by round robin partitioning method showed the best improvement and best overall performance.

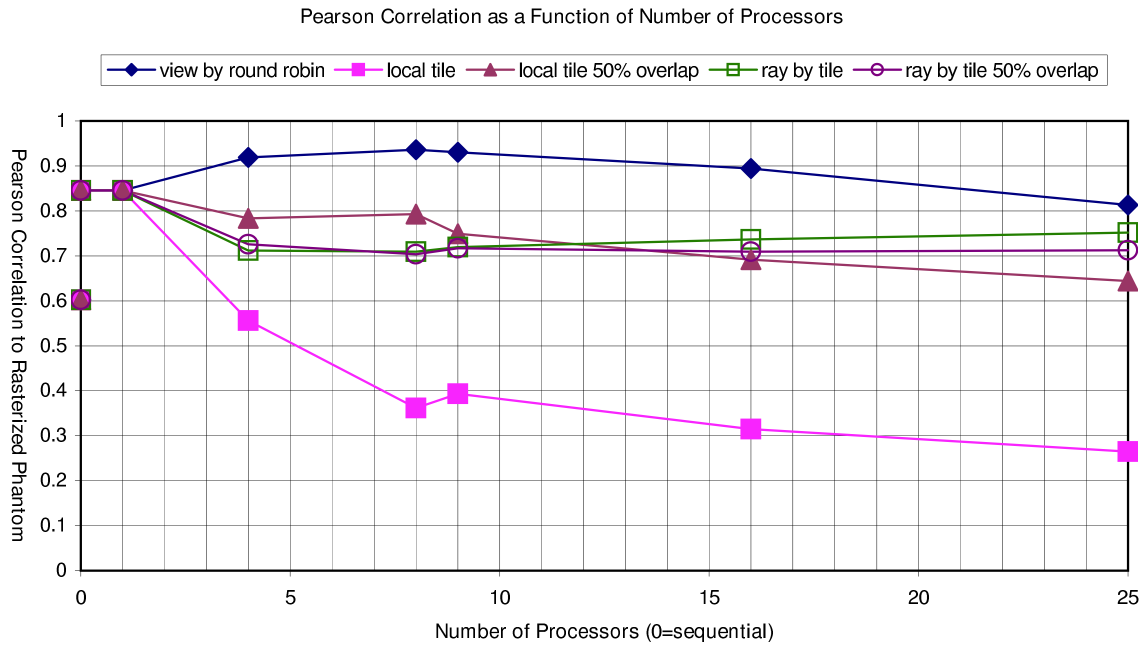


Figure 4.5: Image quality *vs.* number of processors, minimal communication.

Although the sequential view partitioning does not produce suitable images, view by round robin partitioning produces more reasonable, though not ideal, images. An example of reconstruction using view by round robin partitioning, using parallel MART at a resolution of 256×256 pixels, is shown in Figure 4.6.

In general, the ray by tile partitioning methods, both without explicit overlap (result shown in Figure 4.7) and with 50% overlap produce images of reasonable quality. This quality remained constant as more processors were added.

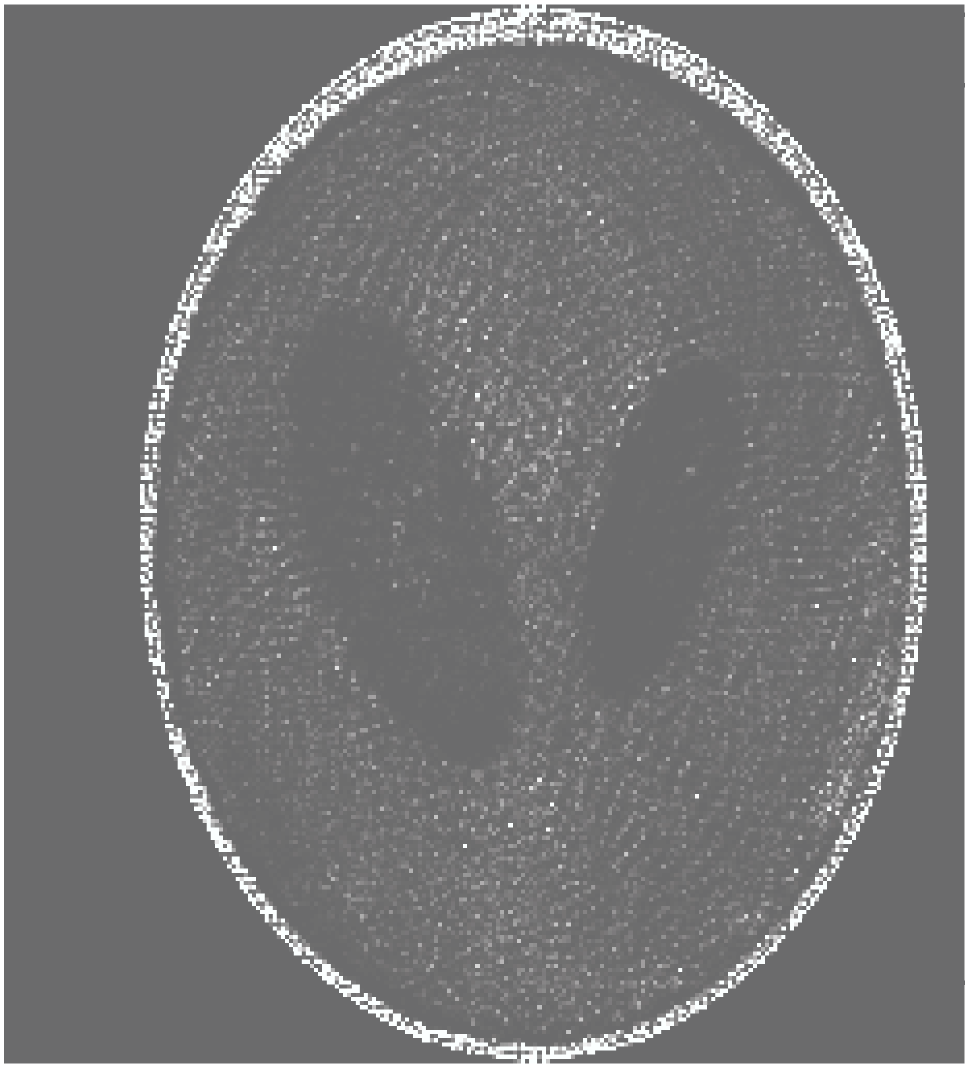


Figure 4.6: Reconstruction using view by round robin, MART at 256×256 pixels, 4 processes.

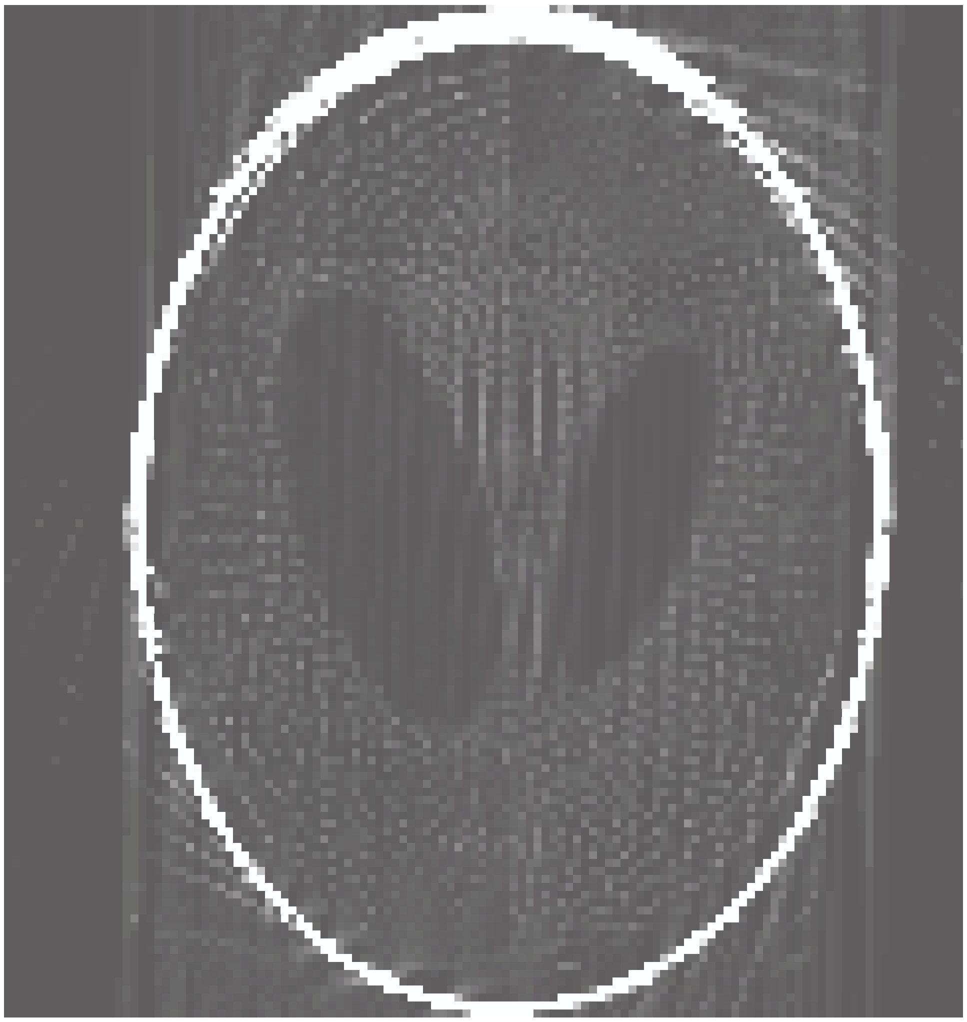


Figure 4.7: Reconstruction using ray by tile, no overlap, 16 processes.

In addition to the subjective impression of preservation of structure, the more objective measurement of Pearson correlation between the reconstruction and its associated rasterized phantom is used in this thesis to measure image quality. Applying this measurement, it can be seen that some deterioration in image quality may occur with increasing numbers of processors. The ray by tile partitioning methods appear to provide good quality reconstructions. The local tile with 50% overlap method and the view by round robin methods may provide acceptable quality. The local tile without overlap method and the view by sequence method are unacceptable in their present forms, due to poor performance as judged by the primary image quality measure as well as by the subjective appearance of the images.

4.2 Speed and Scalability to the Number of Processors

Many iterative algorithms, including the ones examined in this thesis, run in two steps: the first step calculates the weight tables, and the second step uses these tables repeatedly while iterating through successive approximations. In the first step, the weight tables are calculated from the scanner's geometric configuration and the size of the image to be reconstructed,using backprojection. This implementation stores these tables for use in the future for any reconstruction of the same size, made from projections using the same scanner geometry.

The second step, applying the tables during iterations, must be done at the time of each individual reconstruction. This second step affects the turnaround time for the scan results, and is therefor time critical. Only this second step is made parallel in this thesis. At the beginning of this second step, the weight tables must be read into main memory from disk storage, a process that could be eliminated in a commercial application by having weights stored in read-only memory (ROM) on each processor. The 'critical section' for the purpose of this thesis starts immediately after the root process has read its local weight table, and ends with completion of the reconstructed image.

Execution times shown in the results are for this critical section of the algorithm. The timings for this thesis are done by calling the system clock before and after the critical section. Timings are done for at least 3 trials for each number of processors selected. There is some small variation between trials, as expected due to varying processor loads. Trials were done late at night or early in the morning when system loads were presumably minimal. Rather than the usual practice of taking a median or average measurement from a number of trials, the best (minimum) time is chosen since it most accurately reflects the performance of the algorithm. Times in excess of the minimum are assumed to be due to system loads that are unrelated to the algorithm.

The reason for doing the reconstruction in parallel is to gain performance in speed as more processors are used. A parallel algorithm might provide an adequate quality of result, but if performance is not increased by using a number of processors, then there is no advantage over a corresponding sequential algorithm.

Speedup of an algorithm ($S_p(N)$) is defined as follows:

$$S_p(N) = T_s(N)/T_p(N) \tag{4.3}$$

Where:

$T_s(N)$ = time required by best sequential algorithm to solve a problem of size N

$T_p(N)$ = time required by parallel algorithm using p processors to solve a problem of size N

Linear speedup is ideal: for p processors, we get a speedup of p for various partitioning sizes. This indicates that a suitable problem has been appropriately implemented in parallel. Superlinear speedup is possible as the result of cache effects.

Using a parallel iterative method on a distributed memory system, Melvin [56] achieved a maximum speedup of 20% going from 2 processors to 6 to reconstruct a 128×128 pixel image. Direct comparisons of execution time would not be valid given the different implementations and possibly different platforms. However, speedup, being relative, is a reasonable indicator of the success of parallelization.

Scalability indicates that the same speedup occurs with larger values of p. In practical terms, this scalability is often limited by the increased work of communication between processors as their number is increased.

As noted in the previous section, the ray by tile methods provide good quality reconstruction results. Unfortunately, this image quality comes with a significant performance cost. As seen in Figure 4.8, neither of the ray by tile methods match single processor performance as the number of processors is increased. There is no speedup.

Other methods improve on the single processor ‘parallel’ performance, but the local tile methods scale poorly after 8 processors. The view partitioning methods scale quite well, as demonstrated in Figure 4.9.

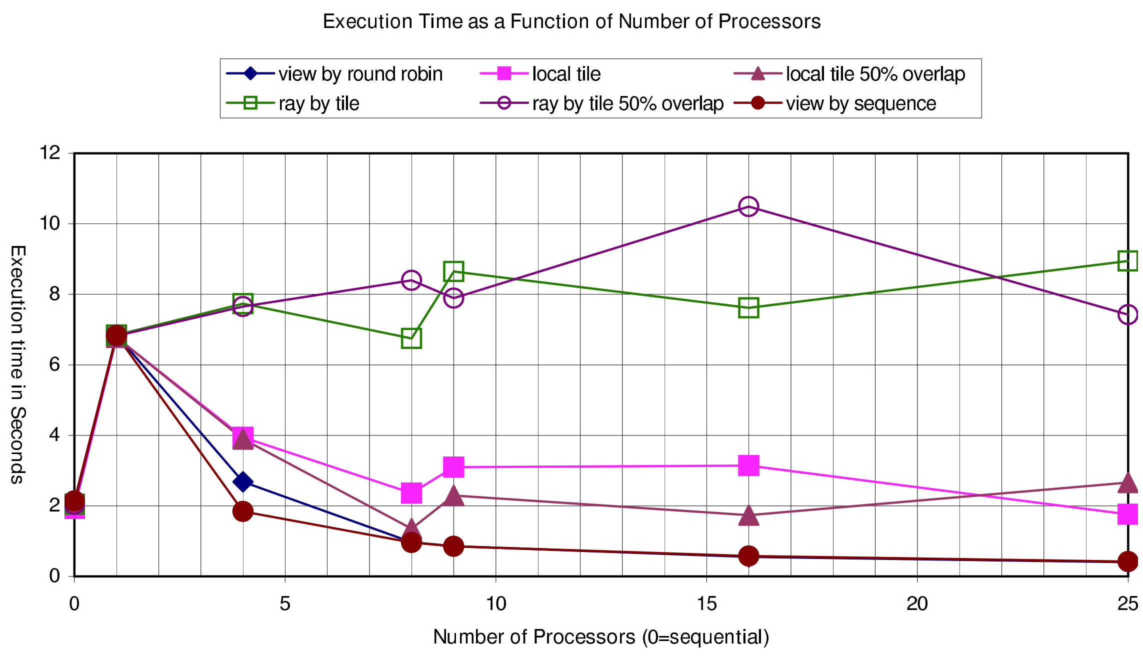


Figure 4.8: Execution time *vs.* number of processors.

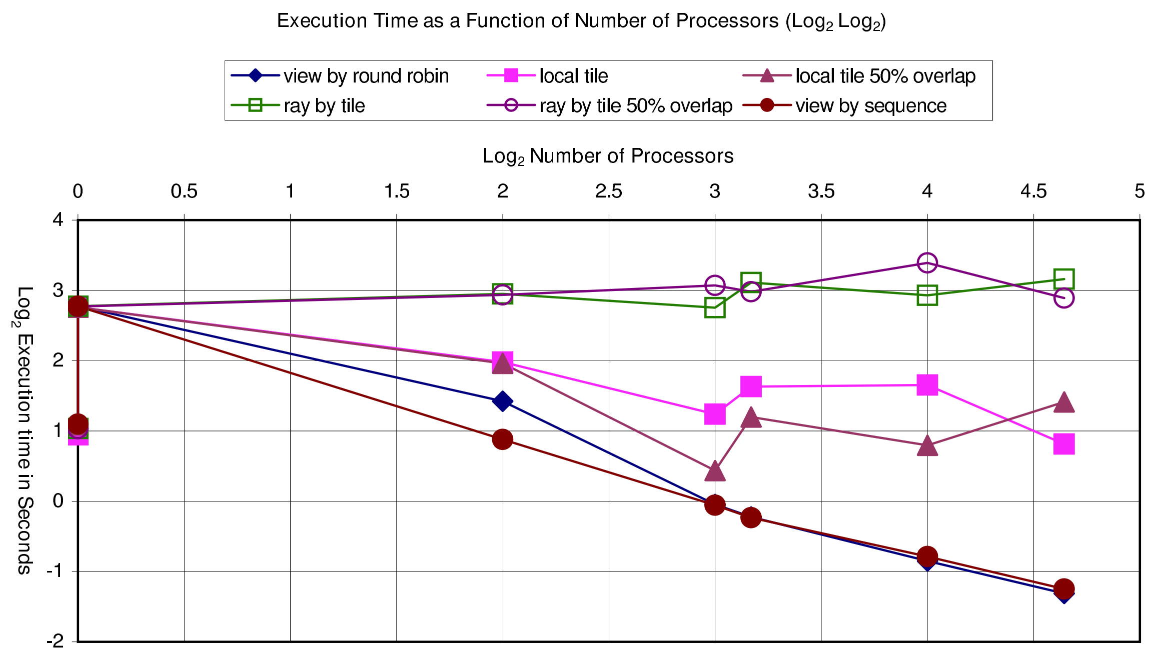


Figure 4.9: Execution time *vs.* number of processors (log-log).

The execution time of the sequential method is shown on the Y axis as a reference only. It is not valid to compare other timings to the sequential method, because it completes the inner iterations only to the specified maximum, and does not iterate through the outer 'parallel' loop to repeat these iterations again. In spite of this, the view partitioning methods improve on the execution time of the sequential method for 4 processors and greater.

The reason for the poor performance of the ray by tile methods is easily seen from Figure 4.10 and from the log plot of Figure 4.11. The local weight tables for the ray by tile methods are much larger than the local weight tables for other partitioning methods. This is because the pixel weights for a single ray, if it intersects multiple tiles, are distributed to multiple processors. This duplication causes the local weight table for a partition of the ray by tile method without any *explicit* overlap to be larger than the local tile tables have an explicit 50% overlap. Since an individual process must iterate over a local weight table that is close to the size of the global weight table, processing times will be only slightly smaller than for the sequential process. Any additional iterations that might be required or any significant work of communication could then cause poorer performance of the parallel algorithm compared to the sequential.

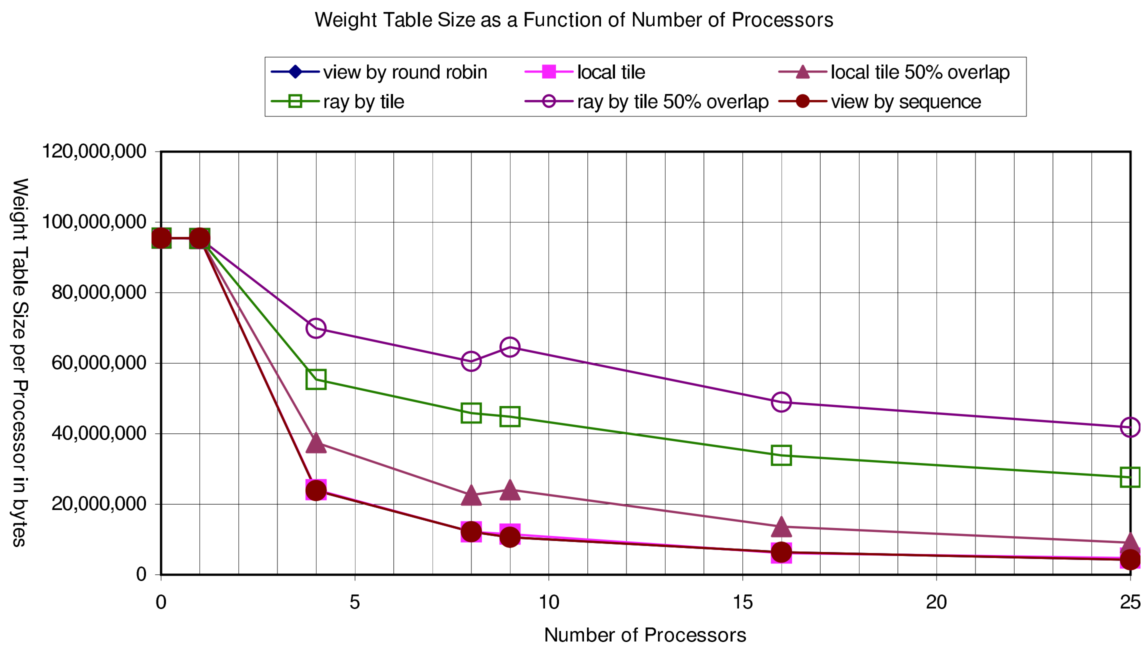


Figure 4.10: Weight table size *vs.* number of processors.

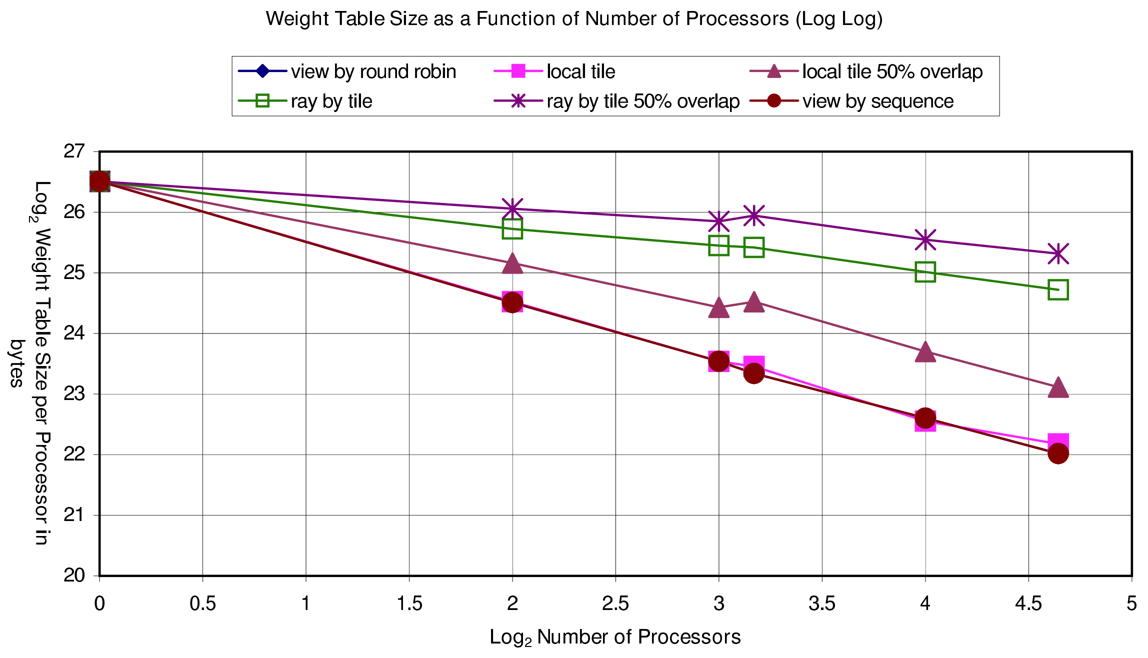


Figure 4.11: Weight table size *vs.* number of processors (log-log).

The only partitioning methods that yield reasonable quality images and show some speedup are the local tile with 50% overlap method and the view by round robin method. Relative speedup and efficiency are plotted for these two methods.

Relative speedup of an algorithm ($S_{relative}(N)$) is defined as follows:

$$S_{relative}(N) = T_1(N)/T_p(N) \tag{4.4}$$

Where:

$T_1(N)$ = time required by single processor parallel algorithm to solve a problem of size N

$T_p(N)$ = time required by parallel algorithm using p processors to solve a problem of size N

This does not indicate the value of parallelization (a high value does not necessarily translate to better performance than the sequential algorithm) but it is useful to gain some understanding of how performance varies as a function of the number of processors. This is plotted for the two best candidate methods in Figure 4.12.

Relative efficiency of an algorithm ($E_{relative}(N)$) is defined as follows:

$$E_{relative}(N) = T_1(N)/(pT_p(N)) \tag{4.5}$$

This is plotted for the two best candidate methods in Figure 4.13. These plots show some scalability for number of processors for the view by round robin method, and efficiency which drops off very slowly.

Using problem sizes shown, a reconstruction that takes several seconds can be reduced to a fraction of a second using parallel ART with view by round robin partitioning. The view by round robin method of partitioning is quite scalable, and it

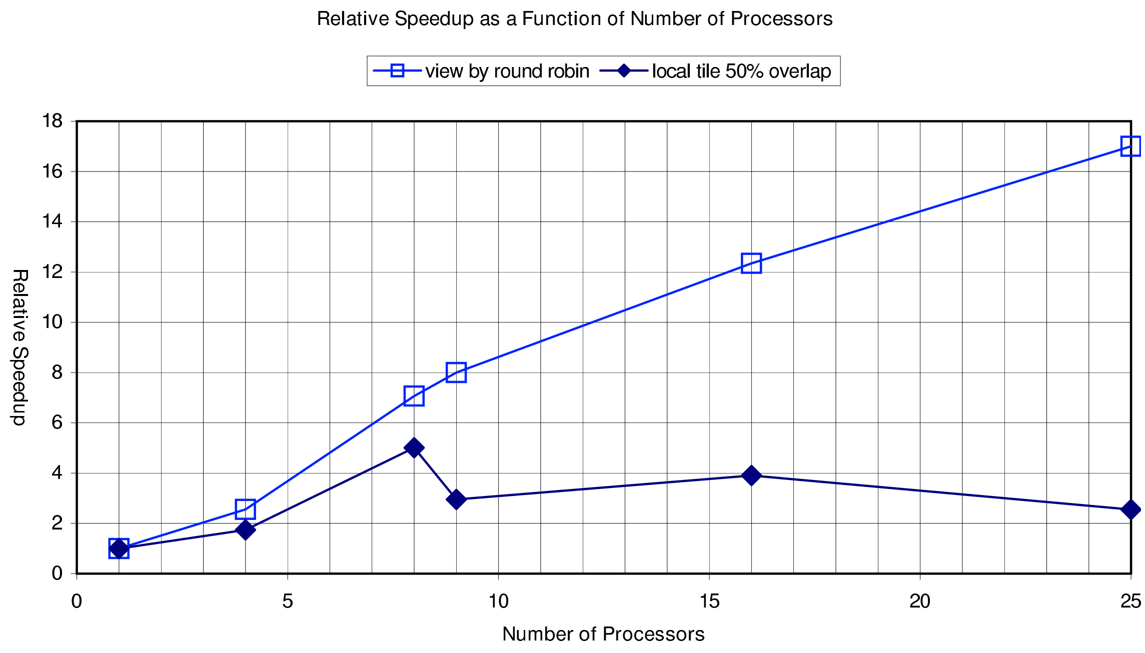


Figure 4.12: Relative speedup *vs.* number of processors.

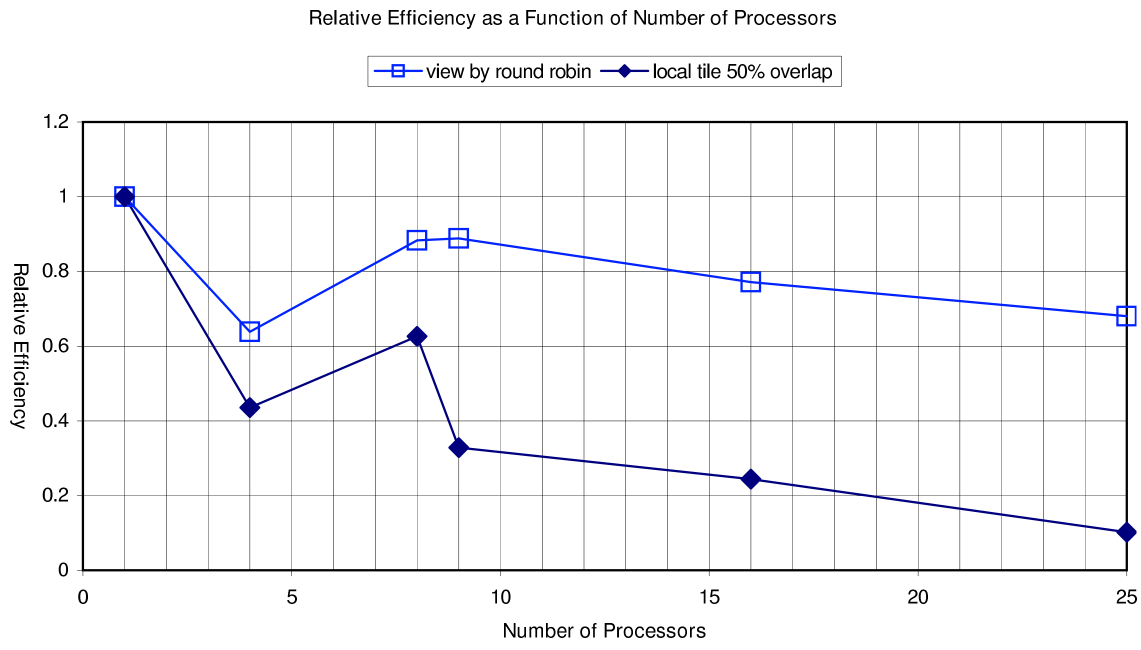


Figure 4.13: Relative efficiency *vs.* number of processors.

appears possible to reduce the execution time even further by adding more than 25 processors, an option not available to the experimenter at present.

4.3 Scalability to Problem Size

All of the methods scale with increasing problem size, as seen in Figures 4.14 and 4.15.

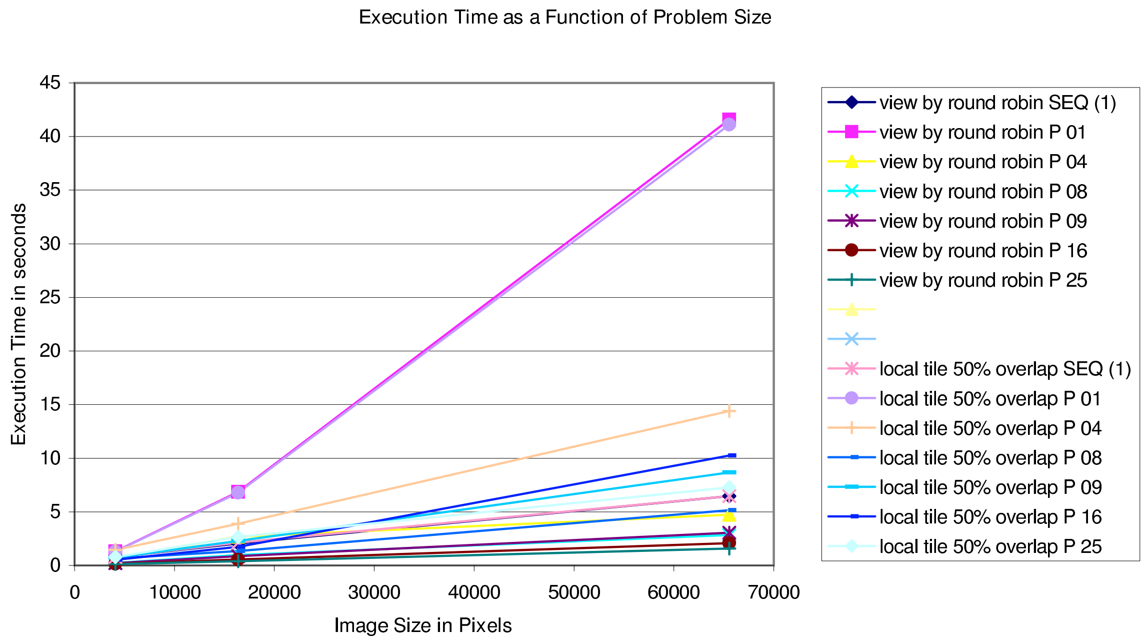


Figure 4.14: Scalability with increasing problem size.

Unfortunately, this only holds true up to a 256×256 pixel image when projections from 180 views are reconstructed. The weight tables are constructed by a sequential process, since this does not belong to the time-critical part of the reconstruction. At a resolution of 512×512 pixels, the weight tables cannot be entirely contained in the 1 gigabyte memory of the processors, so thrashing occurs during weight table

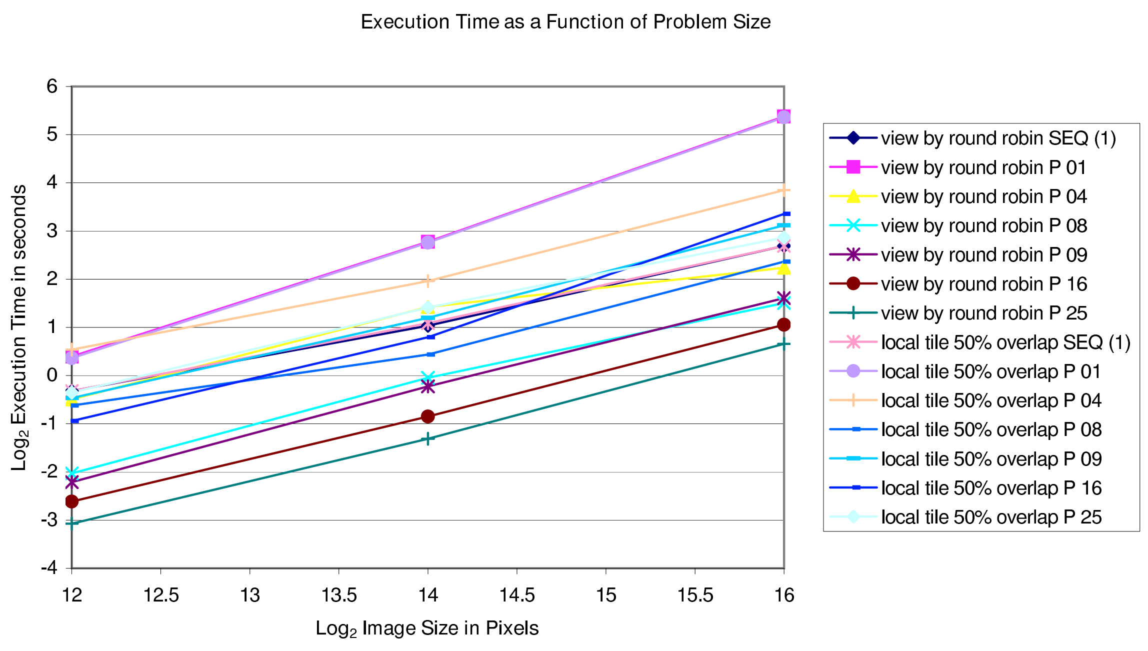


Figure 4.15: Scalability with increasing problem size (log-log).

construction. A similar problem is noted by Mueller [62, page 547]: CPU speeds increase faster as technology progresses than do memory speeds. The result is that memory limitations may form more of a bottleneck than do the processor speeds.

This problem could be easily solved. In fact, a parallel approach offers one solution: using a partitioning method that allows minimal overlap, and a large enough number of processors, the local data set can be small enough to fit local memory.

We do in fact want to work with much larger problem sizes. If the number of views is kept constant, then a weight table for a 512×512 pixel 2-D image would approximate the size of a weight table for a 64×64×64 voxel 3-D image. Such a large problem could serve as a proxy for work in 3-D. Aside from the memory limitation, there is no reason to doubt that the algorithm should continue to scale well with larger problem sizes.

Chapter 5

Conclusions and Future Work

This thesis has demonstrated significant speedup for iterative algorithms by a parallel approach which limits interprocessor communication among distributed memory processors. One method of partitioning the data by view demonstrated that the data dependence between various sets of views can be reduced, obviating the need for communication at every iteration. A novel approach to partitioning the data by image tiling shows some promise for the future.

Successful methods for data partitioning allow us to keep communication overhead a minimum, and therefor gain performance for an iterative CT reconstruction algorithm implemented in parallel on a distributed memory system.

Of the six partitioning methods discussed, two of them yield poor quality images (view by sequence and local tile without overlap, Figure 4.1), and two show no speedup (ray by tile with and without overlap, Figures 4.8 and 4.9). The two remaining methods, local tile with 50% overlap and view by round robin, show both reasonable preservation of image quality (Figure 4.1), and speedup with parallelization (Figure 4.8). Of the two, the view by round robin method offers more visually appealing reconstructions and is very scalable (Figure 4.12).

The success of the partitioning suggests some data locality of individual views, that may not have been previously recognized. Local CT provides a model to suggest possible success for tiling; limited view CT (using only a fraction of the views that would typically be used) may provide a model for view partitioning.

Improvements can be made to both the view and the tiling partitioning methods. The ray by tile methods produced good image quality, but with no speedup. The local tile methods had some degree of speedup, but image quality was not as good as for the ray by tile methods. The local processes using local tile partitioning could not make use of updated values for pixels outside of the tile. The ray by tile methods could make use of these updated values, but at the expense of iterating over these out-of-tile pixels, even when they hadn't been updated. A method can be conceived that combines the best features of both of these tiling methods, achieving the ability of ray by tile methods to update images based on global values, while creating a local weight table only slightly larger than that for local tiling.

Some variations of the view partitioning methods could be attempted. While the view by round robin method (views in equally spaced intervals) give reasonable results, image quality could be improved. Guan and Gordon [37] showed benefits to ordering views to maximize orthogonality between pairs. It might be worthwhile to maximize orthogonality of views within a partition.

At first glance, the algorithm as implemented in this thesis appears to scale well to the problem size. Relative speed up for a 128×128 pixel image (Figure 4.12) is higher than for a 64×64 pixel image (Figure A.6) and speed up for a 256×256 pixel image (Figure B.6) is higher still. In fact, using view by round robin partitioning, the speedup for a 256×256 pixel image is superlinear.

This superlinear speedup suggests cache effect. Inspection of the relative speedup (Figure B.6) and relative efficiency (Figure B.7) of the view by round robin method show a marked increase in performance going from 1 to 4 processors. *Relative* performance then decreases slowly as more processors are added. It seems likely that this impressive performance is a *decrease* in the single processor performance, creating a large relative speedup in comparison for trials using higher numbers of processors. The size of the local weight table approaches 0.4 GB when using a single processor (Figure B.4), a significant portion of memory for a processor that has a total memory of 1 GB. This may have caused some slowdown, and perhaps even some swapfile use.

The limitation of scalability to problem size caused by the large size of the weight tables could be managed in one of at least three ways. Care could be taken to partition the data set over enough processors so that any individual process can contain its entire partition of the set in memory. Another solution is to find and exploit repeating patterns within the weight tables in order to compress them. A more straightforward method is to trade memory use for processing speed, and recalculate the weights every time they are needed, *i.e.*, eliminate actual tables entirely. This is reasonable given that CPU speeds increase faster than do memory speeds as technology progresses, creating a bottleneck in memory size. A similar problem is noted by Mueller [62, page 547].

Potential applications for a rapid iterative reconstruction are abundant. As reconstruction times for an entire volume approach fractions of a second, CT fluoroscopy using limited views might be possible, as just one example. Combined with other technologies, these algorithms may allow a reduction in the X-ray dose for CT.

Appendix A

Results – 64×64 Pixel Resolution

The view by sequence method was not tested at the 64×64 pixel or 256×256 pixel resolutions.

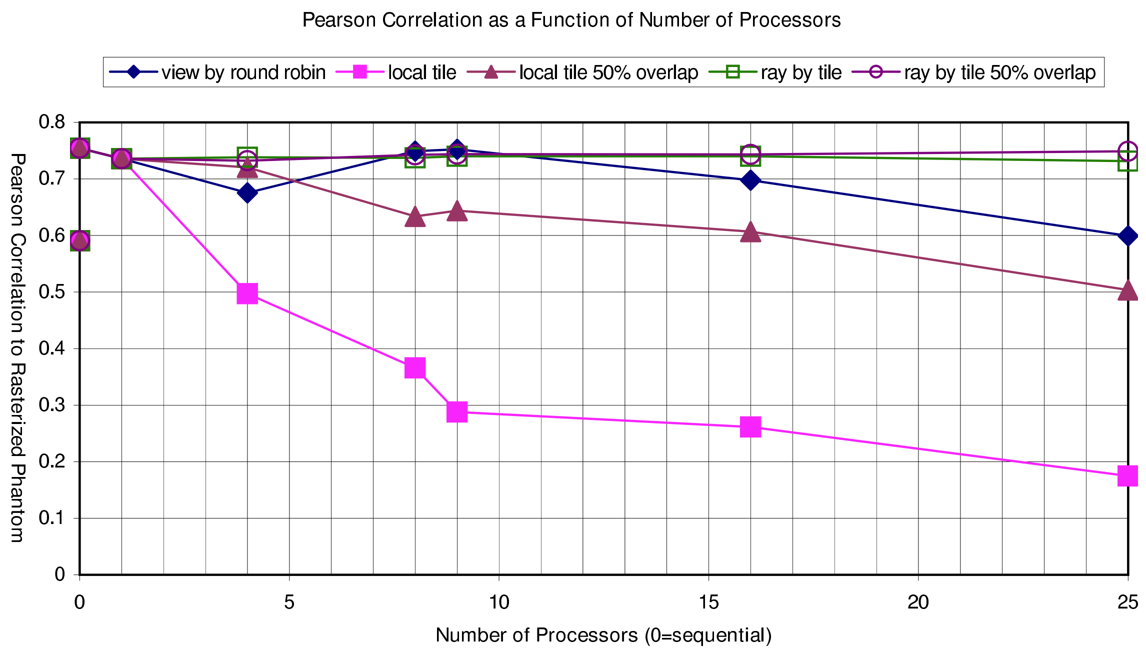


Figure A.1: Image quality *vs.* number of processors.

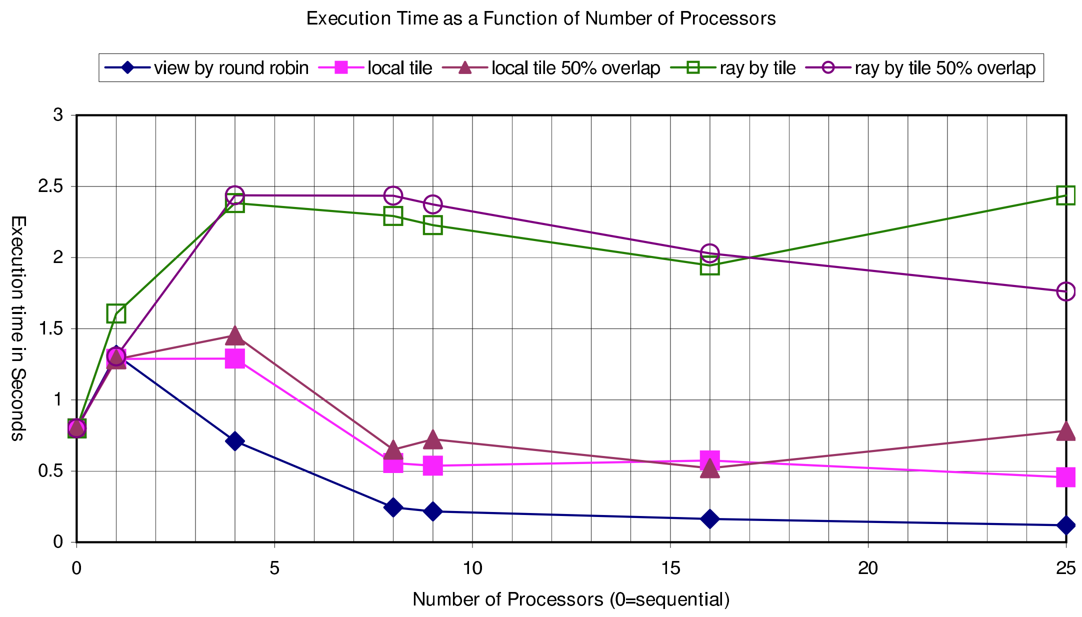


Figure A.2: Execution time *vs.* number of processors.

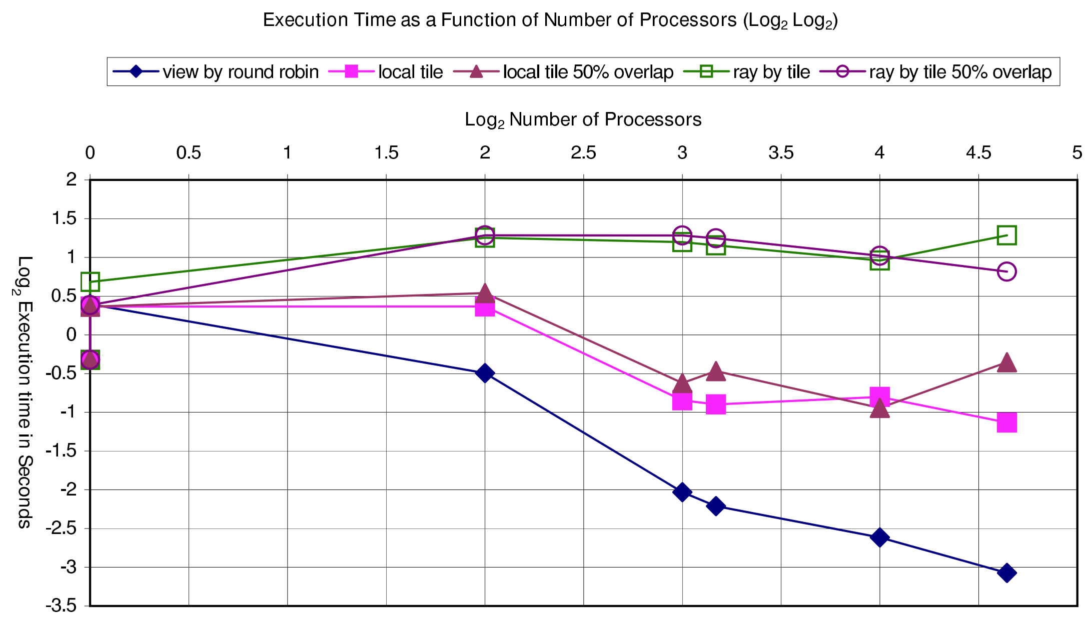


Figure A.3: Execution time *vs.* number of processors (log-log).

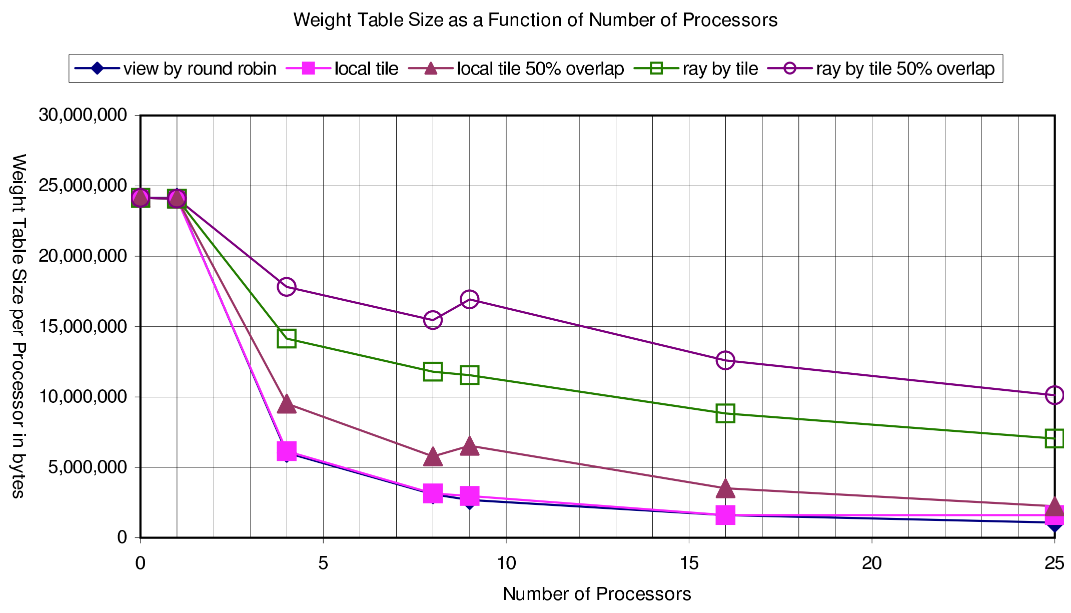


Figure A.4: Weight table size *vs.* number of processors.

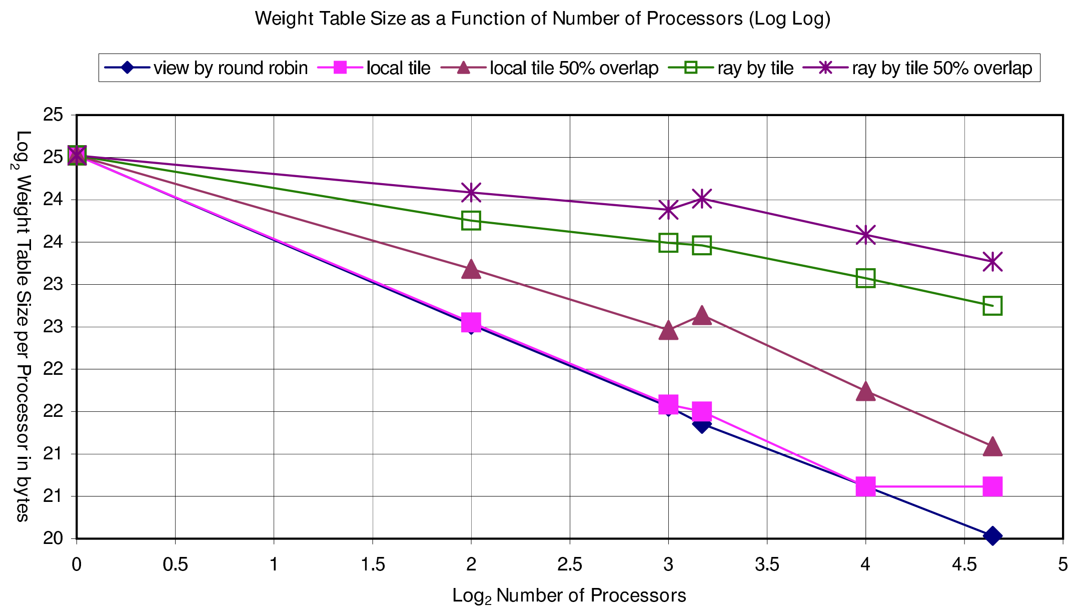


Figure A.5: Weight table size *vs.* number of processors (log-log).

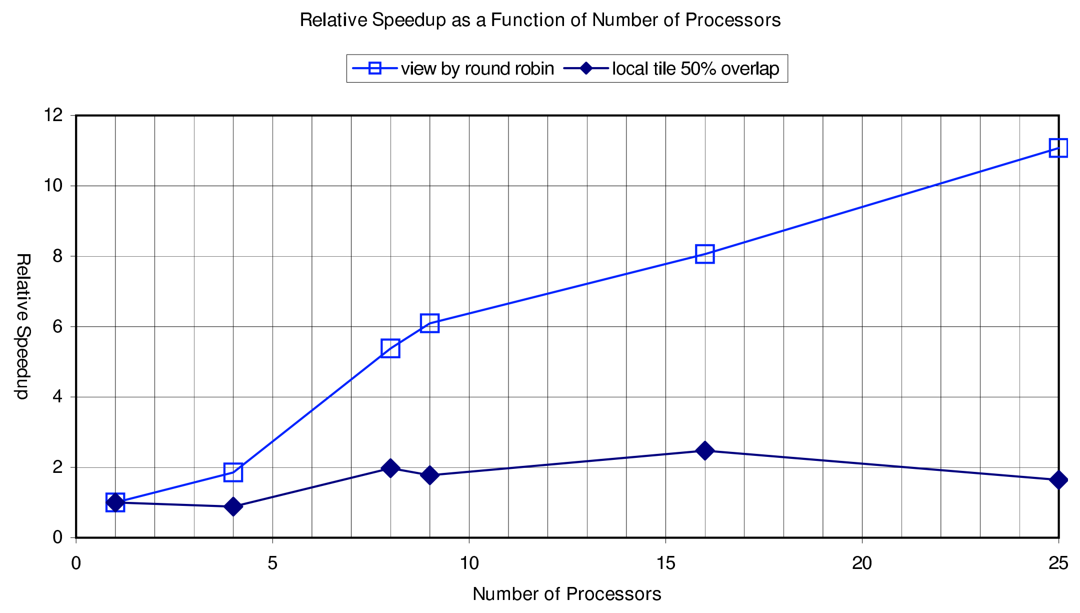


Figure A.6: Relative speedup *vs.* number of processors.

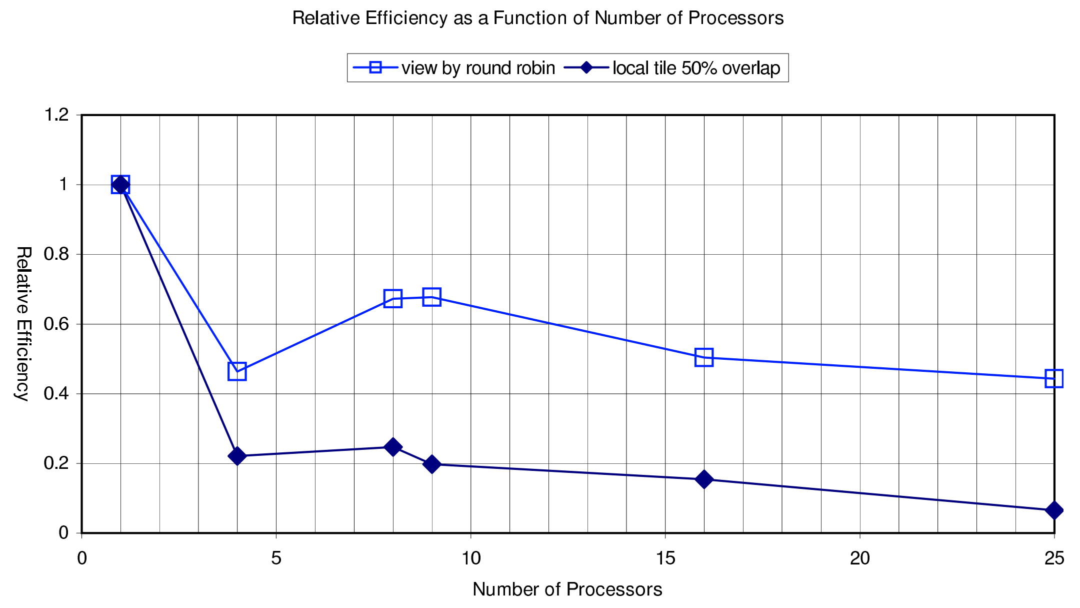


Figure A.7: Relative efficiency *vs.* number of processors.

Appendix B

Results – 256×256 Pixel Resolution

The view by sequence method was not tested at the 64×64 pixel or 256×256 pixel resolutions.

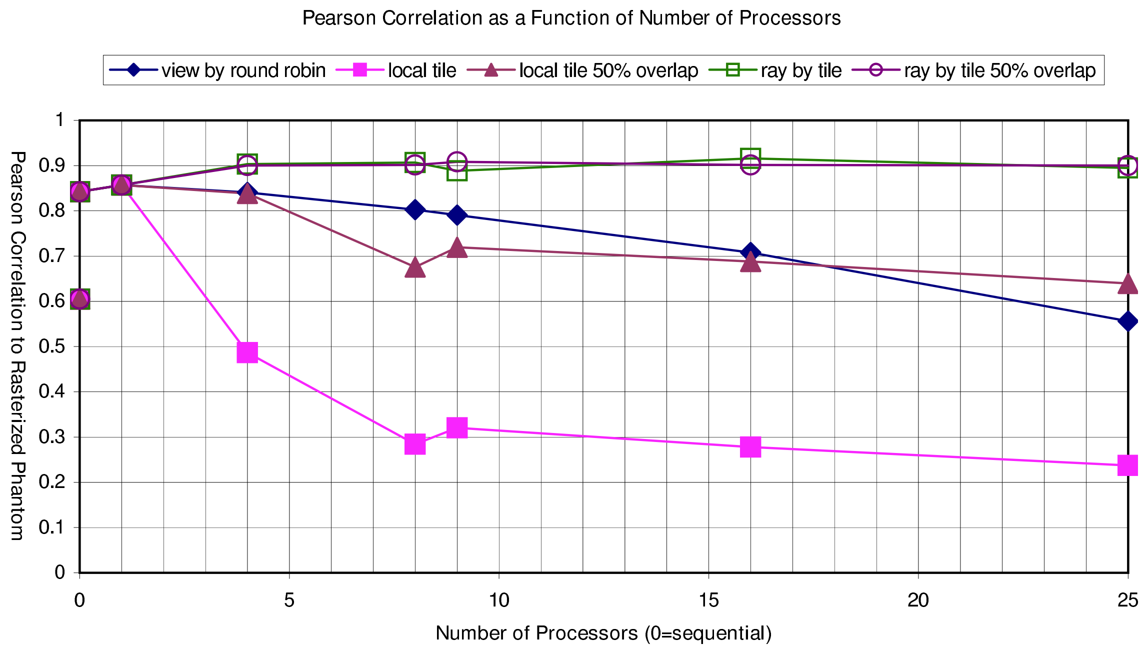


Figure B.1: Image quality *vs.* number of processors.

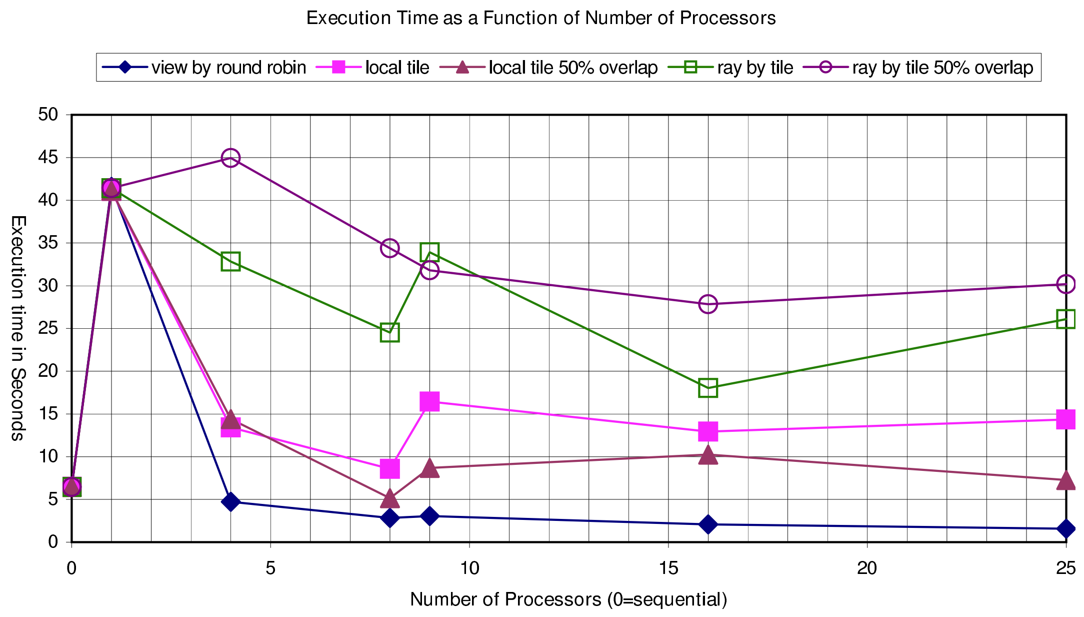


Figure B.2: Execution time *vs.* number of processors.

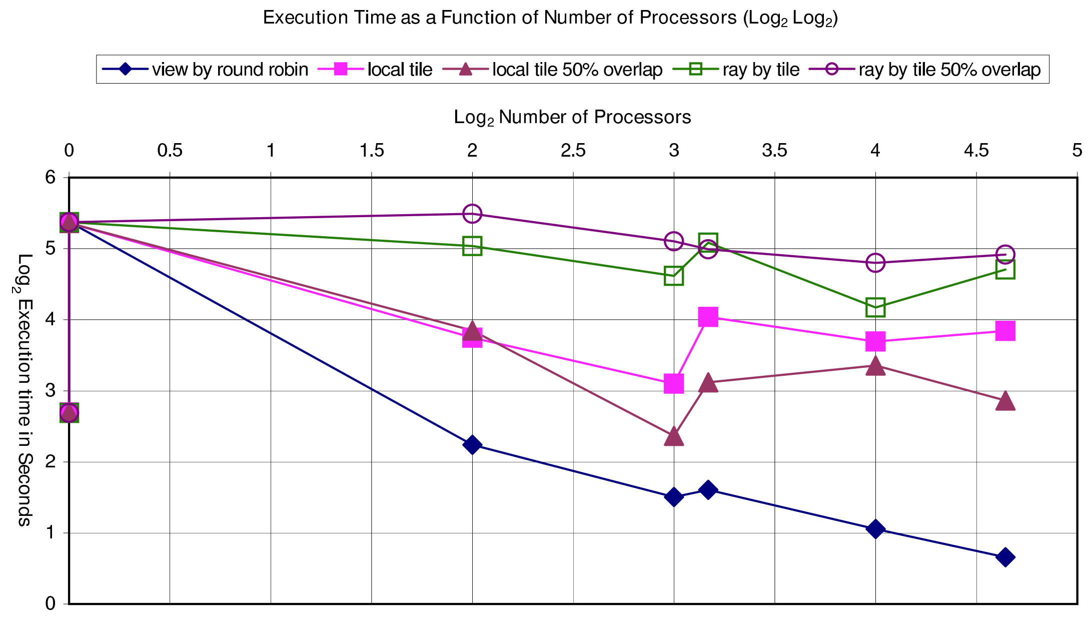


Figure B.3: Execution time *vs.* number of processors (log-log).

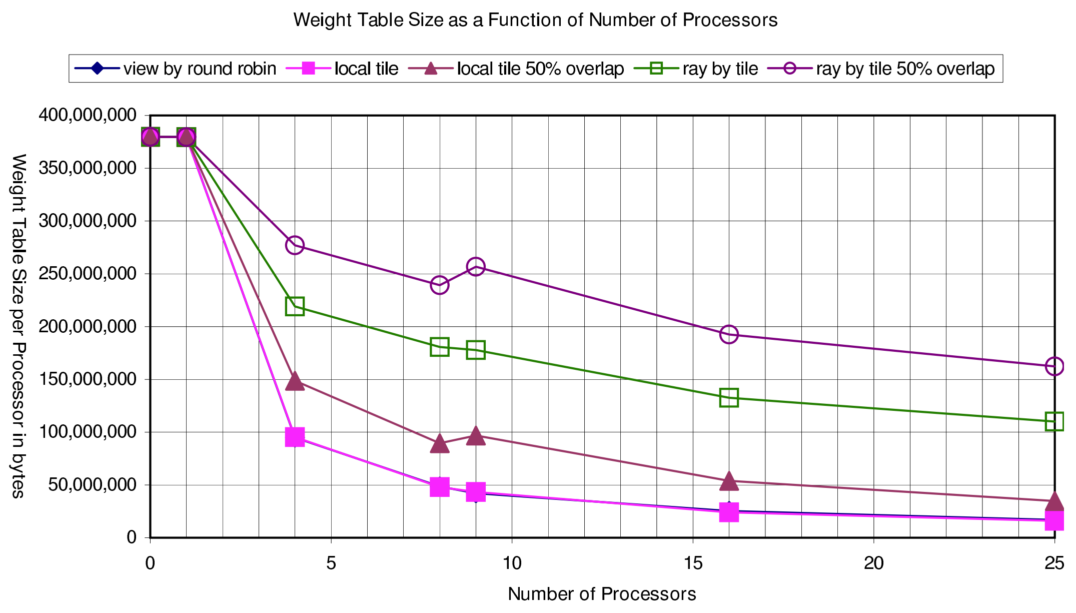


Figure B.4: Weight table size *vs.* number of processors.

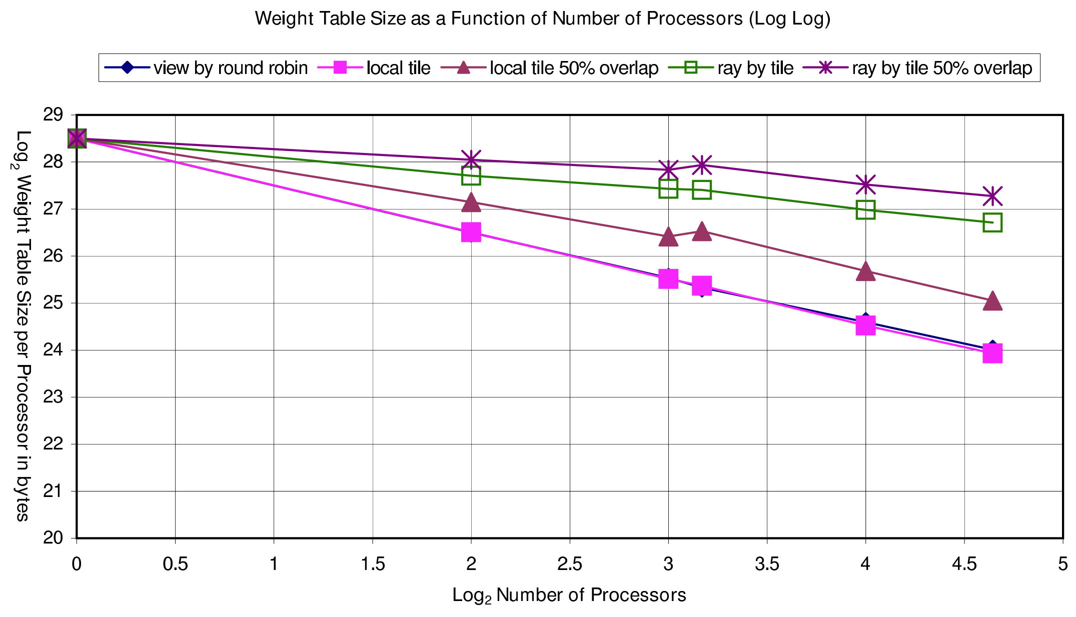


Figure B.5: Weight table size *vs.* number of processors (log-log).

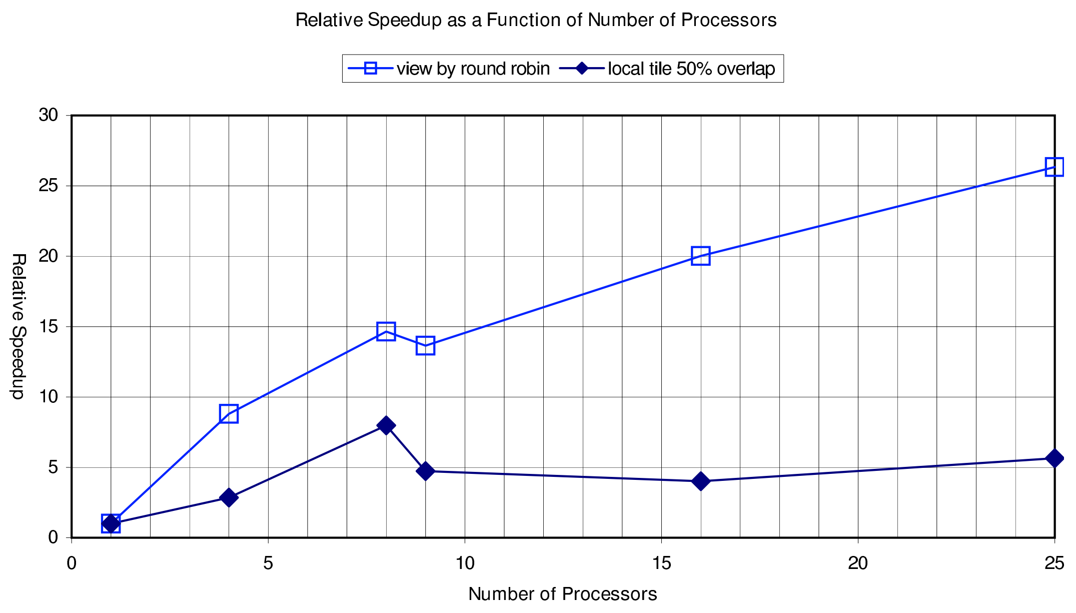


Figure B.6: Relative speedup *vs.* number of processors.

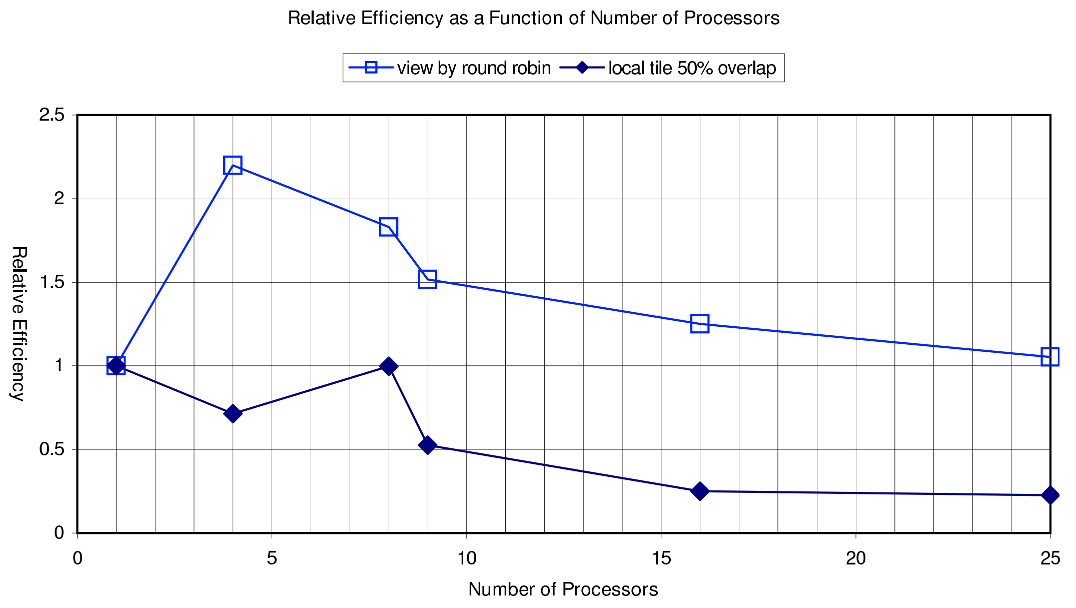


Figure B.7: Relative efficiency *vs.* number of processors.

Bibliography

[1] N.H. Abel. Résolution d'un problème de mécanique. *Journal für die Reine und Angew andte Mathematik*, 1:153–157, 1826.

[2] D.G. Altman and J.M. Bland. Statistics notes: Diagnostic tests 1: sensitivity and specificity. *British Medical Journal*, 308(11):1552, 1994.

[3] A.H. Andersen. Algebraic reconstruction in CT for limited views. *IEEE Transactions on Medical Imaging*, 8:50–55, 1989.

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

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

[6] S. Basu and Y. Bresler. O($N^2 logN$) filtered backprojection reconstruction algorithm for tomography. *IEEE Transactions on Image Processing*, 9(10):1760–1773, 2000.

[7] S. Basu and Y. Bresler. O($N^3 logN$) backprojection algorithm for the 3-D Radon transform. *IEEE Transactions on Medical Imaging*, 21(2):76–88, 2002.

[8] M.J. Budoff, A. Gopal, and D. Gopalakrishnan. Cardiac computed tomography: Diagnostic utility and integration in clinical practice. *Clinical Cardiology*, 29(9 Suppl 1):I4–14, 2006.

[9] C.L. Byrne and J. Graham-Eagle. Convergence properties of the Algebraic Reconstruction Technique (ART). In *IEEE Medical Imaging Conference*, pages 51–2, Orlando, 1992.

[10] C.M. Chen. An efficient four-connected parallel system for PET image reconstruction. *Parallel Computing*, 24:1499–1522, 1998.

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

[12] W. Chlewicki, C. Badea, and N. Pallikarakis. Cone based 3D reconstruction: a FDK-SART comparison for limited number of projections, 2001. Unpublished.

[13] W. Chlewicki, C. Badea, and N. Pallikarakis. Performances of fast SART implementation for reconstruction using cone beam projections, 2001. Unpublished.

[14] Z.H. Cho, C.M. Chen, and S.-Y. Lee. Incremental algorithm - a new fast backprojection scheme for parallel beam geometries. *IEEE Transactions on Medical Imaging*, 9(2):207–17, 1990.

[15] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[16] D.M. Dalal, P.U. amd Hansell. High-resolution computed tomography of the lungs: The borderlands of normality. *European Radiology*, 16(4):771–80, 2006.

[17] A.N. Datta and B. Bandyopadhyay. An improved SIRT-style reconstruction algorithm for microwave tomography. *IEEE Transactions on Biomedical Engineering*, 32(9):719–723, 1985.

[18] B. De Man and S. Basu. Distance-driven projection and backprojection in three dimensions. *Physics in Medicine and Biology*, 49(11):2463–75, 2004.

[19] M. Defrise. A short reader's guide to 3D tomographic reconstruction. *Computerized Medical Imaging and Graphics*, 25(2):113–6, 2001.

[20] S. Don. Radiosensitivity of children: potential for overexposure in CR and DR and magnitude of doses in ordinary radiographic examinations. *Pediatric Radiology*, 34 Suppl 3:S167–72; discussion S234–41, 2004.

[21] J. Eng, J.A. Krishnan, J.B. Segal, D.T. Bolger, L.J. Tamariz, M.B. Streiff, M.W. Jenckes, and E.B. Bass. Accuracy of CT in the diagnosis of pulmonary embolism: A systematic literature review. *American Journal of Roentgenology*, 183(6):1819–27, 2004.

[22] M.P. Federle. CT of the acute (emergency) abdomen. *European Radiology*, 15 (Suppl 4D100-4), 2005.

[23] L.A. Feldkamp, L.C. Davis, and J.W. Kress. Practical cone beam algorithm. *Journal of the Optical Society of America*, 1(6):612–619, 1984.

[24] J. Fitchett. *Locally synchronous globally asynchronous vertex-8 processing element for image reconstruction on a mesh.* Masters, University of Manitoba, 1993.

[25] I. Foster. *Designing and Building Parallel Programs.* Addison-Wesley, 1995.

[26] P.C. Freeny. Pancreatic carcinoma: Imaging update 2001. *Digestive Diseases*, 19(1):37–46, 2001.

[27] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM 3 user's guide and reference manual. Technical report, Oak Ridge National Laboratories, 1994.

[28] R.C. Gonzalez and R.E. Woods. *Digital Image Processing.* Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 2002.

[29] R. Gordon. Artifacts in reconstructions made from a few projections. In K. S. Fu, editor, *The First International Joint Conference on Pattern Recognition*, pages 275–85, Washington, D. C., Northridge, California, 1973. IEEE Computer Society.

[30] R. Gordon. A tutorial on ART (algebraic reconstruction technique). *IEEE Transactions on Nuclear Science*, NS-21:78–93, 95, 1974.

[31] R. Gordon. Dose reduction in computerized tomography. *Investigative Radiology*, 11(6):508–17, 1976.

[32] R. Gordon, R. Bender, and G.T. Herman. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *Journal of Theoretical Biology*, 29(3):471–81, 1970.

[33] R. Gordon and G.T. Herman. Reconstruction of pictures from their projections. *Communications of the ACM*, 14(12):759–768, 1971.

[34] R. Gordon, G.T. Herman, and S.A. Johnson. Image reconstruction from projections. *Scientific American*, 233(4):56–61, 64–8, 1975.

[35] M. Grass, T. Kohler, and R. Proksa. 3D cone-beam CT reconstruction for circular trajectories. *Physics in Medicine and Biology*, 45(2):329–47, 2000.

[36] H. Guan. *Multilevel algebraic reconstruction techniques for X-ray computed tomography.* Ph D, University of Manitoba, 1996.

[37] H. Guan and R. Gordon. A projection access order for speedy convergence of ART (algebraic reconstruction technique): a multilevel scheme for computed tomography. *Physics in Medicine and Biology*, 39(11):2005–2022, 1994.

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

[39] D. Hart and B.F. Wall. UK population dose from medical X-ray examinations. *European Journal of Radiology*, 50(3):285–91, 2004.

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

[41] K. Jacob, G. Vivian, and J.R. Steel. X-ray dose training: are we exposed to enough? *Clinical Radiology*, 59(10):928–34; discussion 926–7, 2004.

[42] M. Jiang and G. Wang. Convergence of the simultaneous algebraic reconstruction technique (SART). *IEEE Transactions on Image Processing*, 12(8):957–961, 2003.

[43] M. Jiang and G. Wang. Convergence studies on iterative algorithms for image reconstruction. *IEEE Transactions on Medical Imaging*, 22(5):569–79, 2003.

[44] E.P. Johansson, A. Rydh, and K.Å. Riklund. Ultrasound, computed tomography, and laboratory findings in the diagnosis of appendicitis. *Acta Radiologica*, 48(3): 267–73, 2007.

[45] A.C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[46] W.A. Kalender. *Computed Tomography: Fundamentals, System Technology, Image Quality, Applications*. Wiley-VCH, 2000.

[47] W.A. Kalender. *Computed Tomography: Fundamentals, System Technology, Image Quality, Applications*. Publicis Corporate Publishing, Erlangen, second revised edition, 2005.

[48] A.V. Lakshminarayanan and A. Lent. Methods of least squares and SIRT in reconstruction. *Journal of Theoretical Biology*, 76(3):267–295, 1979.

[49] K. Lange and R. Carson. EM reconstruction algorithms for emission and trans-

mission tomography. *Journal of Computer Assisted Tomography*, 8(2):302–316, 1984.

[50] D. Lattard and G. Mazare. Image reconstruction using an original asynchronous cellular array. In *Proceedings of the International Symposium on Circuits and Systems*, pages 13–16, 1989.

[51] D. Lattard and G. Mazare. Parallel image reconstruction by using a dedicated asynchronous cellular array. *Parallel Processing for Computer Vision and Display*, pages 489–498, 1989.

[52] C. Laurent, F. Peyrin, J.-M. Chassery, and M. Amiel. Parallel image reconstruction on MIMD computers for 3D cone beam tomography. *Parallel Computing*, 24:1461–1479, 1998.

[53] F. Li, S. Sone, H. Abe, H. MacMahon, S.G. Armato, 3rd, and K. Doi. Lung cancers missed at low-dose helical CT screening in a general population: comparison of clinical, histopathologic, and imaging findings. *Radiology*, 225(3):673–83, 2002.

[54] X. Li, T. He, S. Wang, G. Wang, and J. Ni. P2P-enhanced distributed computing in medical image EM reconstruction. In H. R. Arabnia, editor, *2004 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04)*, volume 2, pages 822–828, Las Vegas, Nevada, USA, 2004. CSREA Press.

[55] C. Melvin. *Parallelization of the ART algorithm for Computed Tomography.* Masters, University of Manitoba, 2006.

[56] C. Melvin, P. Thulasiraman, and R. Gordon. Parallel algebraic reconstruction technique for computed tomography. In *The 2003 International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 535–536, Las Vegas, Nevada, 2003.

[57] W. Mendenhall. *Introduction to Probability and Statistics.* PWS Publishers, Boston, sixth edition, 1983.

[58] F.A. Mettler, Jr., Wiest P.W., Locken J.A., and Kelsey C.A. CT scanning: patterns of use and dose. *Journal of Radiological Protection*, 20(4):353–9, 2000.

[59] K. Mueller and 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*, 19(12):1227–37, 2000.

[60] K. Mueller, R. Yagel, and J.F. Cornhill. The weighted-distance scheme: a globally optimizing projection ordering method for ART. *IEEE Transactions on Medical Imaging*, 16(2):223–30, 1997.

[61] K. Mueller, R. Yagel, and J.J. Wheller. Anti-aliased three-dimensional cone-beam reconstruction of low-contrast objects with algebraic methods. *IEEE Transactions on Medical Imaging*, 18(6):519–37, 1999.

[62] K. Mueller, R. Yagel, and J.J. Wheller. Fast implementations of algebraic methods for three-dimensional reconstruction from cone-beam data. *IEEE Transactions on Medical Imaging*, 18(6):538–48, 1999.

[63] L. Neumayer and A. Kennedy. Imaging in appendicitis: a review with special

emphasis on the treatment of women. *Obstetrics and Gynecology*, 102(6):1404–9, 2003.

[64] M.L. Nipper and L.K. Jacobson. Expanded applications of CT. Helical scanning in five common acute conditions. *Postgraduate Medicine*, 109(6):68–70, 73–7, 2001.

[65] K. Nishizawa, M. Matsumoto, K. Iwai, and T. Maruyama. Survey of CT practice in Japan and collective effective dose estimation. *Nippon Igaku Hoshasen Gakkai Zasshi*, 64(3):151–8, 2004.

[66] J.K. Older and P.C. Johns. Matrix formulation of computed tomogram reconstruction. *Physics in Medicine and Biology*, 38(8):1051–64, 1993.

[67] M.B. Plunkett, M.S. Peterson, R.J. Landreneau, P.F. Ferson, and M.C. Posner. Peripheral pulmonary nodules: preoperative percutaneous needle localization with CT guidance. *Radiology*, 185(1):274–276, 1992.

[68] J. Radon. On the determination of functions from their integrals along certain manifolds. *Mathematisch-Physiche Klasse*, 69:262–77, 1917.

[69] R.M. Rangayyan, A.P. Dhawan, and R. Gordon. Algorithms for limited view computed tomography: A survey. In *IEEE International Conference on Computers, Systems and Signal Processing*, pages 1540–1545, Bangalore, India, 1984. IEEE Press.

[70] R.M. Rangayyan and R. Gordon. Streak preventive image reconstruction with

ART and adaptive filtering(SPARTAF). *IEEE Transactions on Medical Imaging*, MI-1(3):173–178, 1982.

[71] P.M. Rao, J.T. Rhea, and R.A. Novelline. Sensitivity and specificity of the individual CT signs of appendicitis: Experience with 200 helical appendiceal CT examinations. *Journal of Computer Assisted Tomography*, 21(5):686–92, 1997.

[72] R.P.V. Rao, R.D. Kriz, A.L. Abbott, and C.J. Ribbens. Parallel implementation of the filtered back projection algorithm for tomographic imaging, 1995. (revised April 5, 1995) Retrieved January 6, 2005, from http://www.sv.vt.edu/xray_ct/parallel/Parallel_CT.html.

[73] D.A. Reimann, V. Chaudhary, M.J. Flynn, and I.K. Sethi. Parallel implementation of cone beam tomography. In *International Conference on Parallel Processing*, pages 1–4, Indian Lakes Resort, Bloomingdale, IL, 1996.

[74] D.D. Robertson, J. Yuan, G. Wang, and M.W. Vannier. Total hip prosthesis metal-artifact suppression using iterative deblurring reconstruction. *Journal of Computer Assisted Tomography*, 21(2):293–8, 1997.

[75] J. Rockmore and A. Macovski. A maximum likelihood approach to image reconstruction. In *Proceedings of the Joint Automatic Control Conference*, pages 782–786, 1977.

[76] K.M. Rosenberg. CT Sim - computed tomography simulator, 2003. (November 2, 2003) Retrieved February 20, 2005, from http://directory.fsf.org/science/biology/CTSim.html.

[77] R.W. Sebesta. *Concepts of Programming Languages*. Benjamin/Cummings, Redwood City, second edition, 1993.

[78] D.W. Shattuck, J. Rapela, E. Asma, A. Chatzioannou, J. Qi, and R.M. Leahy. Internet2-based 3D PET image reconstruction using a PC cluster. *Physics in Medicine and Biology*, 47:2785–2795, 2002.

[79] D.H. Sheafor, E.K. Paulson, M.A. Kliewer, D.M. DeLong, and R.C. Nelson. Comparison of sonographic and CT guidance techniques: does CT fluoroscopy decrease procedure time? *American Journal of Roentgenology*, 174(4):939–942, 2000.

[80] L.A. Shepp and Y. Valdi. Maximum likelihood reconstruction for emission tomography. *IEEE Transactions on Medical Imaging*, MI-1:113–122, 1982.

[81] 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. In *Heterogeneous Computing Workshop*, pages 241–252, Cancun, Mexico, 2000.

[82] K.T. Smith and F. Keinert. Mathematical foundations of computed tomography. *Applied Optics*, 24(23):3950–7, 1985.

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

[84] G.G. Tirunelveli. *Comparison of square and hexagonal pixels for image processing*

and alternative solutions to linear equations for Computed Tomography. Masters, University of Manitoba, 2003.

[85] A.N. van Daatselaar, P.F. van der Stelt, and J. Weenen. Effect of number of projections on image quality of local CT. *Dentomaxillofacial Radiology*, 33(6): 361–9, 2004.

[86] A. van der Sluis and A. van der Vorst. SIRT- and CG-type methods for the iterative solution of sparse linear least-squares problem. *Linear Algebra and its Applications*, 130:257–303, 1990.

[87] D.J. Vining. Virtual colonoscopy. *Gastrointestinal Endoscopy Clinics of North America*, 7(2):285–91, 1997.

[88] G. Wang, D.L. Snyder, J.A. O' Sullivan, and M.W. Vannier. Iterative deblurring for CT metal artifact reduction. *IEEE Transactions on Medical Imaging*, 15: 657–664, 1996.

[89] A.B. Wolbarst. *Physics of Radiology.* Appleton & Lange, Norwalk, Connecticut, sixth edition, 1993.

[90] X.L. Xu, J.S. Liow, and S.C. Strother. Iterative algebraic reconstruction algorithms for emission computed tomography: a unified framework and its application to positron emission tomography. *Medical Physics*, 20(6):1675–84, 1993.