ABSTRACT

Title of Document: HIGH-SPEED RECONSTRUCTION OF LOW-DOSE CT USING ITERATIVE TECHNIQUES FOR IMAGE-GUIDED INTERVENTIONS

Venkatesh Bantwal Bhat,
Master of Science,
2008.

Directed By: Professor Raj Shekhar,
Dept of Diagnostic Radiology (University of Maryland Baltimore) and
Dept of Electrical and Computer Engineering

Minimally invasive image-guided interventions (IGIs) lead to improved treatment outcomes while significantly reducing patient trauma and recovery time. Ultrasound and fluoroscopy have been traditionally used for image guidance. But these imaging modalities do not provide a comprehensive three-dimensional (3D) view of the anatomy. Because of features such as fast scanning, high spatial resolution, 3D view and ease of operation, computed tomography (CT) is increasingly the choice of intra-procedural imaging technique during IGIs. The risk of radiation exposure, however, limits its current and future use.

We perform ultra low-dose scanning to overcome this limitation. To address the image quality problem with ultra low-dose CT, we reconstruct images using the

iterative Paraboloidal Surrogate (PS) algorithm. As iterative techniques are generally computationally intensive, we have accelerated the PS algorithm on a cluster of CPUs and also a GPU. Here, we first compare the quality of the low-dose images reconstructed using the PS algorithm and the standard filtered-back projection (FBP) algorithm. Using actual scanner data, we demonstrate visually acceptable improvement in the quality of reconstructed images using the iterative algorithm.

We further demonstrate a fast implementation of the Ordered Subsets version of the PS algorithm for axial scans on a cluster of 32 processors using the MPI (Message Passing Interface) and an NVIDIA 8800 GTX GPU using CUDA (Compute Unified Device Architecture). Several studies in the recent past have reported computing forward and back projection on GPU using the rasterization framework. However, the GP-GPU (General Purpose GPU) framework used in our implementation is more generic and accommodates a wider variety of penalty functions on the GPU as compared to the rasterization framework. This obviates the need to transfer data between the GPU and CPU during reconstruction.

We have compared the GPU and the cluster implementations using the ray-tracing method to the exact implementation using a pre-computed weight matrix on a single CPU. We demonstrate about 20 times speedup using a cluster of 32 processors and over two orders of improvement in speed using the GPU, while the image quality remains comparable to that of the exact implementation.

HIGH-SPEED RECONSTRUCTION OF LOW-DOSE CT USING ITERATIVE TECHNIQUES FOR IMAGE-GUIDED INTERVENTIONS

By

Venkatesh Bantwal Bhat

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2008

Advisory Committee:
Professor Raj Shekhar, Chair
Professor Shuvra Bhattacharya
Professor Peter Petrov

Dedication

To my Parents,
Uma, Rakesh and Rachna.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Raj Shekhar for introducing me to the amazing world of Medical Imaging. His guidance and advice was the key to all the work that has been reported in this thesis. The idea to make use of recent technological advances in both Signal Processing and Computer Engineering in the medical imaging field, no matter how interesting and inspiring, would not have borne fruits without his guidance and financial support.

I would like to thank Dr. Shuvra Bhattacharya and Dr. Peter Petrov for readily agreeing to serve on my thesis committee amidst their busy schedules. I greatly value their support and encouragement. I would like to thank Dr. Bulent Bayraktar from Philips for all the help in understanding the CT reconstruction process and extraction of real data from the Philips scanners. The results in this thesis would not have been as valuable or meaningful without his support.

I am thankful to Dr. William Plishker, Dr Omkar Dandekar and Peng Lei for their inputs and support in various aspects of my research during my entire stay at the Imaging Technologies Laboratory. In particular I would like to acknowledge all the support that I received from Dr. William Plishker with regard to the installation and maintenance of the CPU-GPU heterogeneous cluster. I also thank Vinay Gangadhar for taking the time to proofread this thesis. Finally, my family has always been a source of inspiration and encouragement in all my endeavors. I shall always be grateful to them for their sacrifices and their unalienable faith in me.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The current state-of-the-art laparoscopy uses the optical feed obtained from the camera on the laparoscope for surgical guidance and feedback. However, the main drawback of this system is the limited field of view offered by the laparoscope. The laparoscopic view limits the surgeon's visibility to a small region around the central axis of the scope. Apart from the small aperture, the view is also limited due to radial and axial optical distortions. Moreover, optical systems only enable the surgeon to view the surface of the anatomical structures and any internal structures such as the bile duct and arteries are not visible by the laparoscope. Visualization of internal structures, especially the vasculature, has been a long-standing need of minimally invasive surgeons. Therefore, there is a need to augment the laparoscopic view with images from other imaging modalities such as ultrasound, computed tomography (CT), positron emission tomography (PET), etc.

1.1 Augmenting laparoscopic views using CT

CT is a true three-dimensional (3D) imaging modality that is capable of providing a wide coverage of the area of interest in a short time. Apart from the coverage, CT is also attractive due to its high spatial resolution, ease of operation and low acquisition cost. Therefore, CT is a good option for augmenting the laparoscopic views. In the past, laparoscopic views have been augmented using pre-procedural contrast CT scans[28], a technique commonly referred to as Augmented Reality (AR). CT scans have also been used to aid in port (small skin incisions) placement for laparoscopy [35]. However, these techniques make use of outdated CT scans and do not provide

the surgeon with a current up-to-date view of the internal anatomy, which is essential for surgical procedures. Hence, we propose the concept of Live Augmented Reality (Live AR), where we propose to continuously scan the patient while the procedure is performed in the CT scanner.

The use of CT, as we know it today, for IGIs is a difficult proposition due to a number of reasons. The current CT scans are acquired using radiation doses of about 200-250 mA at 120-140KV. Such high radiation doses considerably increase the risk of cancer and other maladies in the patients [1][2]. Considering these facts, it would be nearly impossible to safely acquire a number of CT scans during the procedure without considerably increasing the risk of cancer in the patient. Hence the CT acquisition and reconstruction techniques need to be modified to better adapt CT for IGIs.

1.2 Live Augmented Reality

The advantages of using CT to enhance laparoscopic views by far outnumber the disadvantages. Once the modality for augmentation (CT in our case) is decided, the exact mode of augmentation has to be decided. In the past, peri-operative CT scans have been used to augment the laparoscopic data [28]. In these methods, a full body peri-operative scan is obtained under CT contrast before the procedure begins. This scan is then repetitively used to augment the laparoscopic data. During the procedure, the laparoscope is continuously tracked using an optical tracker system. The peri-operative contrast scan is then volume rendered from the same position as the laparoscopic camera. The view thus obtained is very similar to the laparoscopic view.

This image is then registered with the optical image from the camera to obtain the augmented reality images.

However, there are a number of drawbacks of this system. First of all, the CT data is not updated with time leading to outdated CT images. Since respiration and the surgical procedure will both cause considerable deformation of the internal organs, the structural alignment of the CT and the laparoscopic image will be highly suspect. Therefore, for accurate representation of the augmented data, it is essential to continuously acquire the CT scans during the surgical procedure. Hence we propose the "Live AR" using CT to enable complete 3D volumetric visualization of the anatomy during image-guided minimally invasive surgeries.

The "Live AR" procedure improves upon the AR procedure by continuously updating the CT data throughout the experiment. The laparoscopic data is now augmented using near real-time CT data, thus providing up-to-date information about the internal anatomy that is not visible in the laparoscopic optical images. There are a number of challenges in making "Live AR" a reality. In this thesis, we demonstrate solutions to a few of those challenges. Some of the other problems have also been overcome in [47]. The 3D Live Augmented Reality workflow can be summarized as in Figure 1.1.

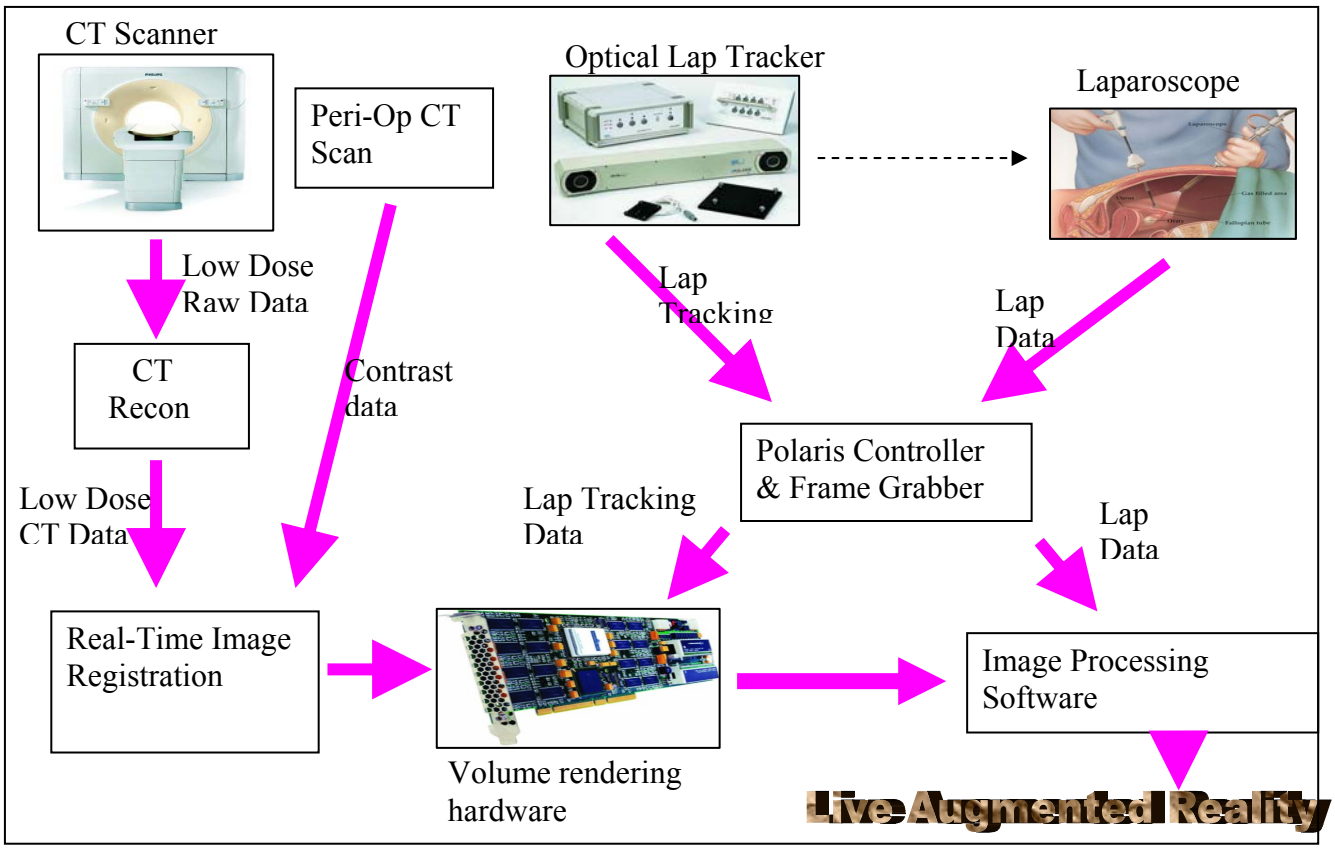

Figure 1.1 Proposed workflow for Live Augmented Reality

Before the procedure begins, a whole-body contrast scan of the patient is acquired and stored. This scan is henceforth referred to as the peri-operative scan. The peri-operative scan is acquired at the normal radiation dose. Since continuous update of CT data is essential for Live AR, the minimally invasive surgery is performed on the CT table such that the area of interest is within the CT scan region (field of view). During the surgery, frequent or continuous ultra-lose dose axial CT scans are acquired. The axial scans serve two purposes.

- Unlike the helical scans, they do not require translation of the CT table during the scanning procedure thereby not interfering with the procedure.

- They help to reduce the amount of radiation administered to the patient throughout the procedure.

The ultra-low dose scans obtained from the CT scanner are then reconstructed using high-speed reconstruction. Since the ultra-low dose scans will not essentially present all the arteries and veins to the surgeon, they are registered with the contrast-enhanced peri-operative scans. During the entire procedure, the surgical instruments are continuously tracked using an optical tracking system to ensure knowledge of their precise location.

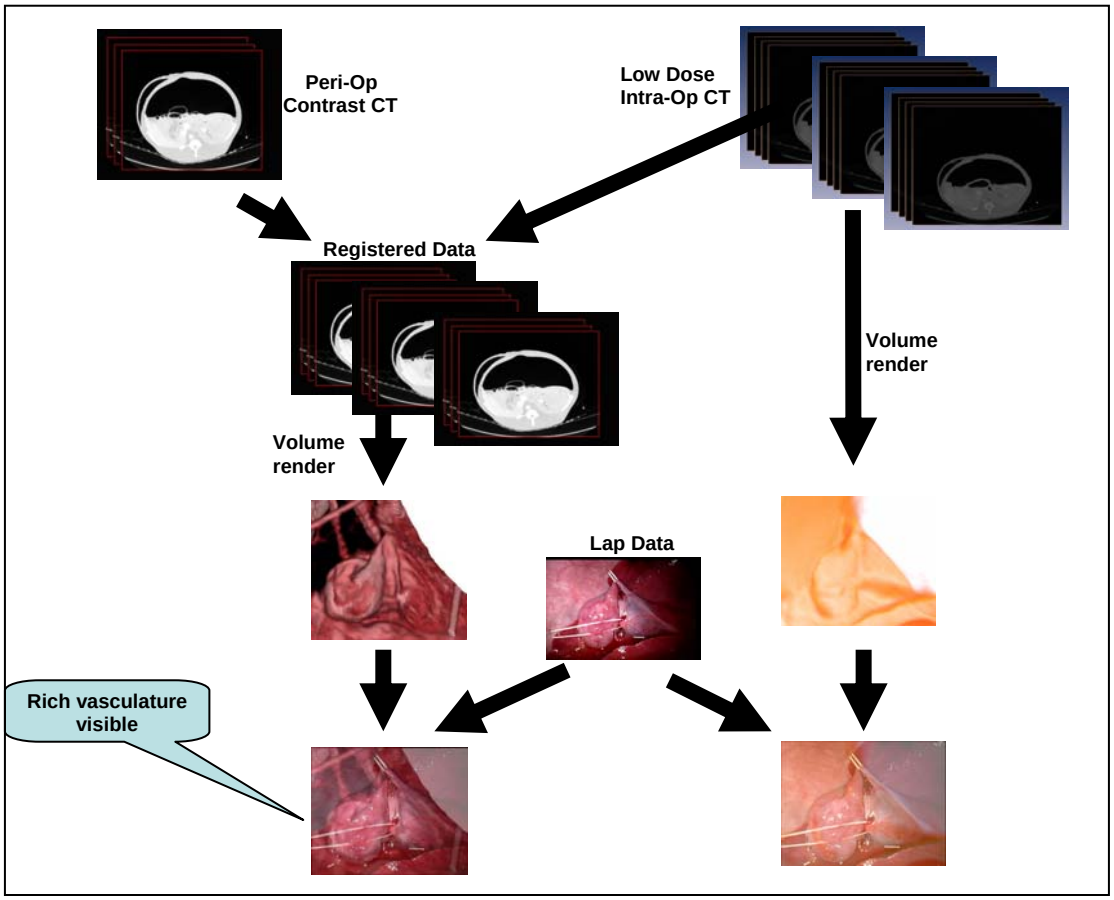

Figure 1.2 Intermediate outputs of Live AR Implementation

Using this information, the registered scan is volume-rendered to mimic the view from the laparoscope. The two images are then fused using specialized software to present the "Live Augmented Reality" feed to the surgeon. Figure 1.2 pictorially depicts the various intermediate outputs and the final results.

1.3 Contributions of this thesis

As explained in section 1.2, as the number of scans per patient increase, the need to lower the radiation doses becomes imperative. Also, high-speed reconstruction of these scans is extremely useful for real-time navigation during IGIs.

Since the low-dose scans are extremely noisy, they cannot be reconstructed using the standard procedures. In this thesis, we mainly explore iterative techniques for reconstruction of the ultra-low dose image data. We first demonstrate superior reconstruction of the low dose images using these techniques. Since iterative techniques are computationally intensive, we demonstrate ray-tracing modification to accelerate these algorithms. We also show equality in the quality of reconstructed images using these ray-tracing based techniques, which provide an acceleration of at least 30X. We further accelerate the algorithms using a cluster of computers and on the GPU using NVIDIA's CUDA architecture. We again compare the reconstructed images qualitatively as well as quantitatively and demonstrate equality of image quality while achieving over 2 orders of magnitude speedups. We finally propose a terminating condition to properly estimate the number of iterations required for convergence. We also experimentally validate the propriety of this terminating condition.

1.4 Outline of this thesis

The remainder of this thesis is arranged as follows. Chapter 2 gives an overview of CT reconstruction, the standard FBP algorithm and the iterative techniques. We use real scanner data to demonstrate the superiority of the iterative techniques over the standard FBP technique for low dose CT scans. Chapter 3 gives an overview of the most compute-intensive portions of the algorithm and our ray-tracing based solutions to accelerate these portions. We then demonstrate a speedup of at least 30X using the ray-tracing method over the standard pre-computed weight matrix method for iterative reconstruction while retaining the image quality.

Chapter 4 gives an overview of our multiprocessor heterogeneous cluster setup and an overview of NVIDIA's 8800 GTX GPUs. We explain our implementation of the ray-tracing based algorithms on the hardware with other improvements to make efficient use of the available resources. We also propose a terminating condition to ensure the right number of iterations for convergence of the image. Chapter 5 presents the results and compares the speedups achieved as well as the reconstructed image quality using each of the methods. We show 100-400X speedups using the GPU and about 6-22X speedups using the multiprocessor cluster while maintaining the quality of the reconstructed images. Using the terminating condition we demonstrate iterative reconstruction of 64-slice axial scans in under a minute. Finally, we present our conclusions and explain the scope for future work.

Chapter 2: Low-Dose CT Reconstruction

2.1 Need for radiation dose reduction

It has long been recognized that excessive exposure to X-ray radiations during CT scans can lead to an increase of the probability of cancer in patients. Studies conducted by Brenner et al. [2] lead to the conclusion that a single full-body CT scan at normal doses increases the cancer mortality risks by 0.08%. This factor increases to about 1.9% when about 30 CT scans are considered. Brenner et al. [1] show that the estimated risk of cancer mortality in infants is at least an order of magnitude higher than for adults. Berrington de Gonzalez et al. [54] further show that about 0·6% to 1.8% of the cumulative risk of cancer can be attributed to diagnostic X-rays.

Moreover, the risk posed by the radiation to the patient is somewhat insignificant when the situation of the surgeon is considered. Since a surgeon may be involved in a number of CT augmented IGIs in a single day, the risk due of secondary radiation from the scanner to the surgeon cannot be overlooked. These statistics reinforce the need to reduce radiation doses when CT is used to augment laparoscopy and for other IGIs.

2.2 The CT acquisition and reconstruction process

A typical CT scanner consists of a doughnut shaped gantry that consists of a set of X-ray sources and detectors on opposite sides. The sources emit X-rays that are attenuated as they pass through the object in the CT gantry. The attenuated X-rays are

then detected at the detectors. The amount of attenuation suffered at each detector is a measure of the cumulative attenuation/transmission coefficients of the materials intersected by the ray. These projections acquired at various locations at different angles are then used to reconstruct an intensity map of the transmission coefficients at various points in the object. This reconstructed image is a representation also of the density of the various objects in the body.

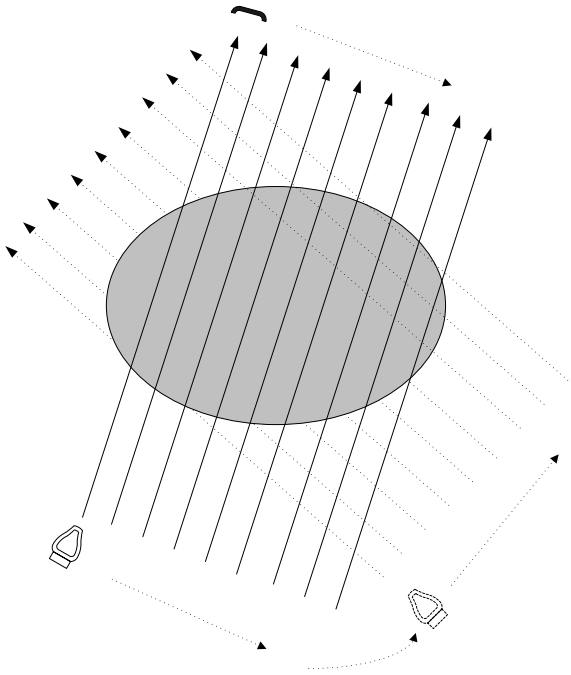

Figure 2.1 First generation parallel-beam CT configuration

The X-ray sources and detectors may be arranged in one of a number of different configurations. A few of the different configurations in the first-, second- and third-generation CT scanners are illustrated. While Figure 2.1 demonstrates the most basic parallel beam projections, Figure 2.2 illustrates the fan-bean projections in the rotate and translate format. Figure 2.3 illustrates the third-generation CT scanners in

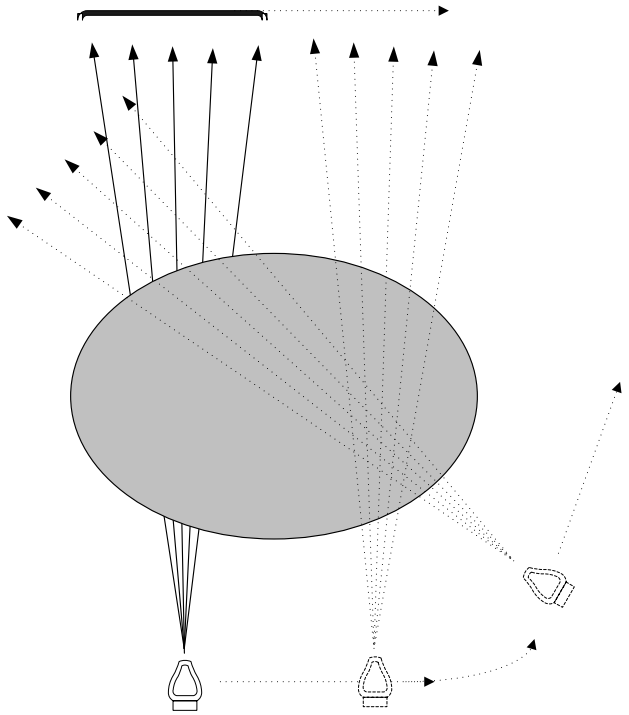

Figure 2.2 Second-generation translate and rotate fan beam CT configuration

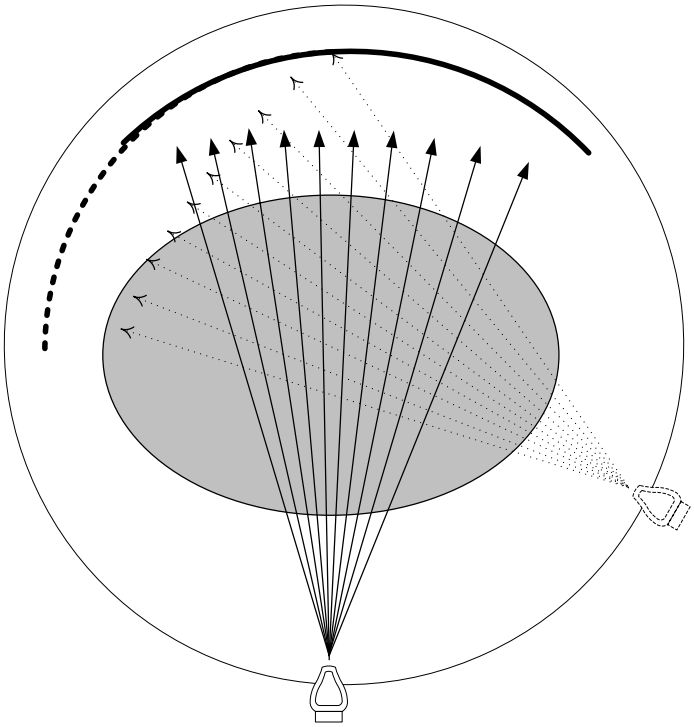

Figure 2.3 Third-generation rotate only fan beam configuration

the rotate only configuration. The current generation of scanners, however are mainly multi-slice in nature and often consist of more than one focal spot. This means that the detectors are arranged on a 2-Dimensional grid opposing the sources. The grid rotates around an axis in the gantry. The table and the patient are placed in the gantry along this axis of rotation. Depending on the motion of the table along the axis, helical or axial scans can be obtained. Figure 2.4 illustrates the concept. This enables fast acquisition of multiple projections across several planes during a single rotation. However, data collected from any of these configurations can be re-sorted to simulate the data obtained using a simple parallel-beam reconstruction method.

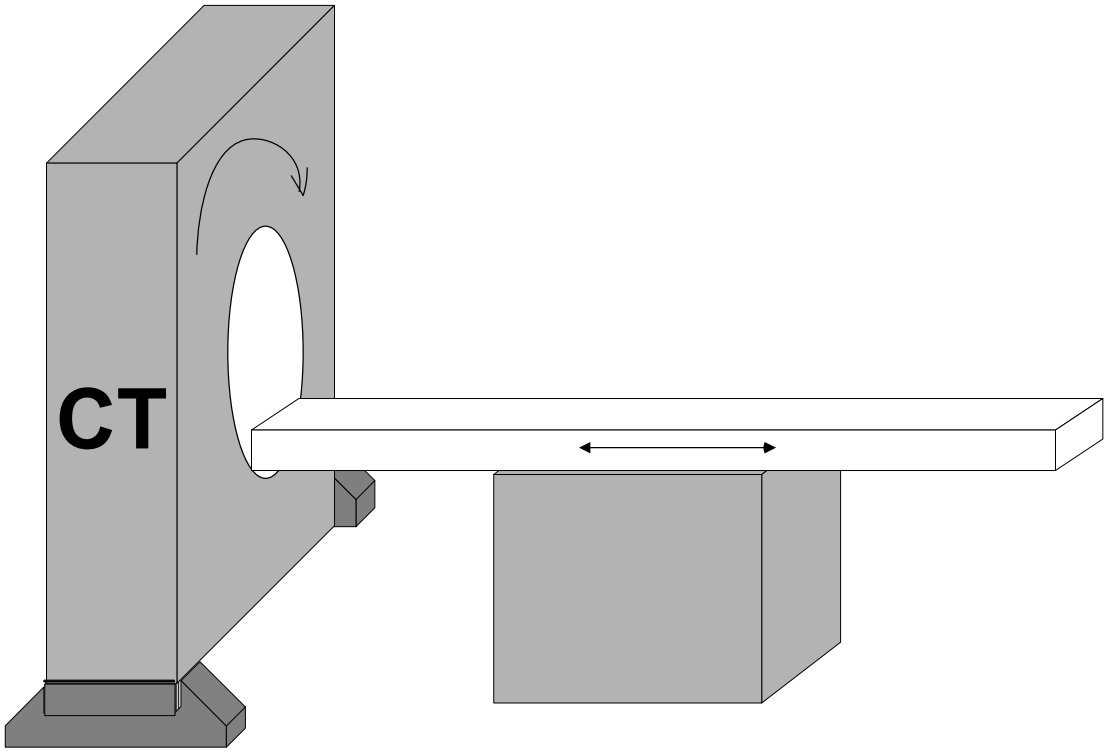

Figure 2.4 CT scanner setup that enables axial or helical scans

2.3 The Filtered Back Projection Algorithm

The Filtered Back Projection (FBP) algorithm is the most commonly used algorithm to reconstruct images from the scanned data. The algorithm makes use of the Fourier Slice theorem and the radon transform to reconstruct images from the scanned data[4][46].

The Fourier Slice theorem states that "The Fourier Transform of a parallel projection at a given angle θ gives a slice of 2-D Fourier Transform of the original image". The Fourier slice theorem makes use of this fact to obtain the Fourier Transform of the original image from the projection data. However there are drawbacks of using this scheme directly as the 2-D Fourier Transform slice of the original image obtained by this method is radial in orientation. Therefore interpolation would be essential to arrange the transform coefficients on a uniform 2-D grid before inverse transform can be obtained. To overcome this problem, the FBP method converts each of the slices into the spatial domain and performs back projection and summation in the spatial domain.

Since the Fourier slices obtained are linear in contrast to the wedge shaped slices required to accurately reconstruct the images, a filtering step is introduced before the back projection. Usually the Ram-Lak filter is used for reconstruction [46]. In case of noisy sinograms, the Ram-Lak filter (ramp filter) is first multiplied by a window that de-emphasizes certain high frequency components. Thus the FBP mainly consists of the following steps.

a) Fourier Transform.

b) Filtering

c) Inverse Fourier Transform

d) Back Projection

More details about the algorithm are explained in detail in Kak-Slaney [4].

The FBP algorithm is widely used due to its speed of reconstruction and simplicity. It also produces good quality images at normal radiation doses (120-140 kVp, 200-250 mAs). The FBP reconstruction can commence immediately after a set of projections are obtained and hence data acquisition and reconstruction can overlap, thereby reducing the time for reconstruction. Moreover, being an analytical technique, FBP gives a closed loop solution to the reconstruction problem and requires no iterations.

All of these techniques make the FBP extremely favorable for CT reconstruction. However, the FBP has a number of drawbacks when we consider scans acquired at low radiation doses. Since low-dose corresponds to significantly lowering the number of incident photons, with the body further attenuating the same, the number of photons reaching the detectors is extremely small. Under such circumstances, effects due to beam hardening, reflections and scatter become significant as compared to the attenuation effects. These lead to corruption of the data obtained at the detectors. The FBP being an analytical algorithm cannot accommodate such corruption of data and leads to artifacts in the final images, either streaking or speckled in nature [5-10].

The metal artifact is yet another common concern with commercial CT reconstruction. Since metals have high attenuation coefficients, the diagnostic X-ray beams are severely attenuated by the presence of metals. This results in an insufficient number of photons reaching the detectors [51][52]. The problem is further compounded when scans are obtained at extremely low doses. This leads to streaking artifacts in images reconstructed using the FBP algorithm. The artifacts are often reduced using filters during post-processing or by using interpolation to approximate the data lost due to metallic attenuation. However, these methods have not met with much success and metal artifacts are an area of concern even in commercial scanners today [51-53].

The problem of metal artifacts does not usually occur in patient X-ray diagnosis as there are very few metallic objects in the area of interest. Except for metallic filling in dentures and metallic prosthetics, the scanned objects are usually free of any metals. However, this can be a major problem in Live AR since the laparoscope and other surgical instruments which are metallic in nature can cause severe artifacts in the reconstructed images. Thus FBP is not a good choice for image reconstruction for Live AR.

2.4 Iterative Statistical Reconstruction Algorithms

Statistical reconstruction algorithms take into consideration the exact processes behind the X-ray generation and attenuation. They assign mathematical models and distributions to the photon generation at the X-ray source and the attenuation at the object. It is widely acknowledged that photon generation is a Poisson process [9][10].

The iterative expectation maximization (EM) algorithm is explained by Lange and Carson [10], and is repeated here for convenience.

Suppose Y is a random vector that is observed during a process, with a density function of g(Y,θ), where θ is the parameter to be estimated. Also suppose that it is difficult to maximize g(Y,θ) with respect to θ. The EM algorithm proceeds by imagining a vector X in a space that encompasses the space of Y, such that h(X) = Y. if f(X,θ) is the density of X with respect to some measure μ(X), then g can be expressed as,

$$g(Y,\theta) = \int f(X,\theta) d\mu(X) \tag{2.1}$$

Maximizing 'g' now involves two steps.

- The E step that involves forming the conditional expectation

$$E(\ln(f(X,\theta)|Y,\theta^n) \tag{2.2}$$

- The M step that involves maximization of (2.2) with respect to a new 'θ' called $\theta^{(n+1)}$.

Thus 'θ' converges to its true solution after a series of steps.

For the emission tomography, where the system detects pairs of gamma rays emitted indirectly by a positron-emitting radionuclide (tracer) that is introduced into the body on a biologically active molecule, an easy closed loop solution for the EM algorithm has been reported by Lange and Carson [10]. This is widely used in emission modalities such as PET, SPECT etc. However, the solution suggested for the transmission tomography is not easy to implement and parallelize. Equation (2.3)

gives the solution to the EM algorithm for the transmission tomography as reported in [10].

$$\theta_k^{n+1} = \frac{\sum_i (M_{ik} - N_{ik})}{1/2 \sum_i (M_{ik} + N_{ik}) l_{ik}} \tag{2.3}$$

Here, l_{ik} gives the length of intersection of the i^{th} ray with the j^{th} pixel. And M_{ik} represents the number of pixels of the i^{th} ray entering the j^{th} pixel, while N represents the number of photons leaving the pixel. Assuming a Poisson distribution for the number of photons leaving the source and the probability of a photon reaching the detector given by

$$p = e^{\sum_j l_{ij} \theta_j} \tag{2.4}$$

Where 'θ' is the attenuation constant of pixels and 'j' is the pixel counter and 'i' is the ray counter, the number of photons entering and leaving a pixel can be estimated. This is the iterative statistical expectation maximization (EM) algorithm.

The main drawback of the EM transmission algorithm is its difficulty in implementation. For any iteration, the exact number of photons entering and leaving every pixel needs to be calculated. Since this is dependent on the previous estimate of the attenuation constant and the exact path traced by the ray, it cannot be pre-calculated. Also, parallelization of the algorithm is extremely hard and calculation of the pixel updates for the various pixels is not independent. Hence, not only is the algorithm computationally intensive, it is also not amenable to parallelization. Yet another problem with the transmission EM algorithm is its slow convergence rate.

Due to all of these reasons, we decide against using the EM algorithm or any of its variants in our implementation.

2.5 The Paraboloidal Surrogates Algorithm

Due to the various concerns associated with the EM algorithm as listed in section 2.4, Erdogen and Fessler [5] proposed a set of Monotonic algorithms for transmission tomography. These algorithms involved a class of algorithms called the Paraboloidal Surrogate algorithms (PS). These algorithms made use of the optimization transfer principles to maximize the Log Likelihood function. We present a brief overview of the same as described in [5] for the purposes of completion. A more detailed account can be found in the references [5-7].

For the optimization transfer principle, Erdogen and Fesslar [5] use a surrogate parabola to construct a paraboloidal surrogate function to the log likelihood function. However in order to be able to easily maximize the surrogate function, they use a separable paraboloidal surrogate. Moreover, a penalty function is introduced to make use of any a priori information about the images. The penalty function can also be designed as a separable function to enable easy maximization. The final Separable Paraboloidal Surrogates (SPS) algorithm can be defined as,

$$\mu_j^{n+1} = \mu_j^n + \frac{\sum_{i=1}^{Ny} a_{ij} h_i^n - \beta \sum_{k=1}^{K} c_{kj} w_k [C\mu^n]_k}{\sum_{i=1}^{Ny} a_{ij} a_i y_i} \qquad (2.5)$$

$$h_i^n = b_i e^{-\sum_{j=1}^{Nx} a_{ij} u_j} - y_i \tag{2.6}$$

$$a_i = \sum_j a_{ij} \tag{2.7}$$

Here 'i' is a ray counter varying from 1 to 'N_y'. 'j' is a pixel counter varying from 1 to 'N_x'. 'b_i' is an estimate of the initial number of photons from the air scan. 'a_{ij}' is a measure of the length of intersection of the i^{th} ray with the j^{th} pixel. 'y_i' is the value of the sinogram for the i^{th} ray. 'μ_j' represents the value of the j^{th} pixel. And a superscript gives the iteration number for any of the variables. 'β' is the scaling factor for the contribution from the penalty function and the contribution is given by

$$\sum_{k=1}^{K} c_{kj} w_k [C\mu^n]_k \tag{2.8}$$

Here, we assume a quadratic penalty function of the form

$$R(\mu) = \sum_{k=1}^{K} w_k \frac{1}{2} ([C\mu]_k)^2 \tag{2.9}$$

where C is the penalty matrix, and $[C\mu]_k = \sum_j c_{kj} \mu_j$. w_k is the scaling factor.

Hudson and Larkin [15] first proposed the Ordered Subsets technique to accelerate iterative reconstruction algorithms. The technique proposes to use a subset of the projective rays to update the image before moving on to the next subset. This leads to a number of image updates per iteration instead of the usual single update, thereby leading to faster convergence. This technique has also been used for the PS algorithm

by Erdogen and Fesslar[5]. However, the OS version of the algorithm is not guaranteed to be monotonic, though we have found that for a reasonable number of subsets, the OS algorithm performs extremely well and is monotonic in nature. The updated algorithm for OS subsets can be written as,

$$\mu_j^{n+1} = \mu_j^n + \frac{M\sum_{i\in S} a_{ij} h_i^n - \beta \sum_{k=1}^{K} c_{kj} w_k [C\mu^n]_k}{\sum_{i=1}^{Ny} a_{ij} a_i y_i} \qquad (2.10)$$

Here all the symbols are similar to those in equations (2.5) and (2.6). 'M' represents the number of Ordered Subsets, and 'S' is the current subset.

For the implementation of the algorithm, the values of all the a_i can be pre-computed. These values remain constant for a given scanner geometry and an image size. Moreover, the size of these values is only as large as the sinogram size.

2.6 Metrics for Image Comparison

In medical imaging, the quality of the reconstructed images needs to be judged on the basis of the ability to discriminate between the various parts of the anatomy in the images rather than their visual appeal. Similarly, comparison of images generated using two different reconstruction algorithms would require comparisons based on the amount of discernable information in the images irrespective of the average contrast levels or exact pixel values. We use both the Minkowski metric based Peak Signal-to-

Noise Ratio (PSNR), which is a pixel by pixel comparison technique as well as the covariance-based Q index proposed by Wang and Bovik [55]. While the PSNR is a pixel by pixel difference based metric, the Q index tries to compare the images based on the variation of the pixel intensities between the corresponding blocks of the two images.

2.6.1 The Peak Signal to Noise Ratio (PSNR)

The PSNR of an image with respect to a benchmark image was calculated as follows. The two images f1(i,j) and f2(i,j) were first normalized to the range of 0 to 1 as follows.

$$\overline{f} = \frac{f - \min(f)}{\max(f - \min(f))} \tag{2.11}$$

The Mean Square Error (MSE) between the two images with N × N pixels each was calculated as,

$$MSE = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \overline{f1}(i,j) - \overline{f2}(i,j)}{N \times N} \tag{2.12}$$

Finally, the PSNR was calculated as 10 times the inverse logarithm to the base 10 of the Mean Square Error.

$$PSNR = 10 \times \log_{10}\left(\frac{1}{MSE}\right) \tag{2.13}$$

2.6.2 The Q index

The Q index was calculated as explained in [55]. The two images f1(i,j) and f2(i,j) were first normalized to the range of 0 to 1 as follows.

$$\overline{f} = \frac{f - \min(f)}{\max(f - \min(f))} \tag{2.14}$$

The Q index is calculated on a block by block basis and then averaged across all the blocks in the image to obtain the final value. We chose a block size of 16 × 16 pixels for our calculations to get a good trade-off between the number of blocks per image and the contribution of each pixel to the overall index.

The variance for a block of size N×N, the mean and variance were calculated as,

$$\mu(f) = \frac{1}{(N \times N)} \sum_{i=1}^{N} \sum_{j=1}^{N} \overline{f}(i, j) \tag{2.15}$$

$$\sigma^2(f1, f2) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\overline{f1}(i, j) - \mu(f1))(\overline{f2}(i, j) - \mu(f2))}{(N \times N) - 1} \tag{2.16}$$

The Q index for the k^{th} block of images f1 and f2 can be calculated as,

$$Q(k) = \frac{4\sigma^2(f1, f2)\mu(f1)\mu(f2)}{(\sigma^2(f1, f1) + \sigma^2(f2, f2))(\mu(f1)^2 + \mu(f2)^2)} \tag{2.17}$$

Finally, the Q index for the images is calculated as the average of the Q indices of all the blocks in the two images. Suppose there are K overlapping blocks in the 2 images the sliding window Q index can be calculated as,

$$Q = \frac{1}{K} \sum_{k=1}^{K} Q(k) \tag{2.18}$$

The Q index, thus varies from -1 to 1. A Q Index of 1 suggests that the images that are being compared are the same, while -1 suggests that the images are highly uncorrelated.

2.7 Low-Dose Reconstruction using PS Algorithm

The PS algorithm was developed for attenuation correction in emission tomography [10]. Often, a few transmission scans are required before the emission scans to calculate the Attenuation Correction Factors (ACF) in emission tomography. Since these scans are extremely noisy, Erdogan et al. [10] first proposed the PS algorithm for this kind of transmission scans. Since low-dose CT scans are also marred by extremely high noise variance, we propose to use the PS algorithm for low-dose reconstruction.

2.7.1 Methods and Setup

To compare the reconstruction quality of low dose images reconstructed using PS algorithm and FBP algorithm, two specimens were scanned using Philips Brilliance 64 slice CT scanner. The specimens were scanned at 120 kVp with the tube current varying from 200 mAs to 15 mAs. All the scans were axial in nature. The raw data for these axial scans was extracted from the scanner and re-binned to obtain the parallel beam projections. The projections were then normalized using the air scan and corrected for faulty or missing detectors. The preprocessed sinograms finally consisted of 580 views of 1498 projections each. These were used to reconstruct the images using both FBP algorithm as well as PS algorithm. The FBP images were not post-processed in any manner. For fair comparison, no penalty function was used in

the PS algorithm. 80 iterations of the PS algorithm were used to ensure convergence of the reconstruction to the optimal solution. The first specimen, a dead chicken, was reconstructed to a resolution of 512 × 512 while the second, a live swine was reconstructed to a resolution of 1024 × 1024 pixels.

2.7.2 Results and Conclusion

The figures 2.5 to 2.8 display the reconstructed images at varying radiation doses. For displaying purposes, the images have been normalized and cropped/resized. At high radiation doses, both FBP as well as PS algorithms give good quality images. However, as the radiation dose is lowered, the FBP images start showing speckled noise, which is absent in the PS reconstructed images.

Figure 2.5 (top) demonstrates the chicken model images scanned at 120 kVp and at a tube current of 200 mAs. It can be seen that FBP as well as the PS reconstructed images do not have any perceptible noise in those. Figure 2.5 (bottom) displays the reconstructed images for the same specimen scanned at 120 kVp, but at a tube current of 15 mAs. While the FBP image has undergone considerable degradation with respect to the earlier image, the PS reconstructed image remains virtually unchanged.

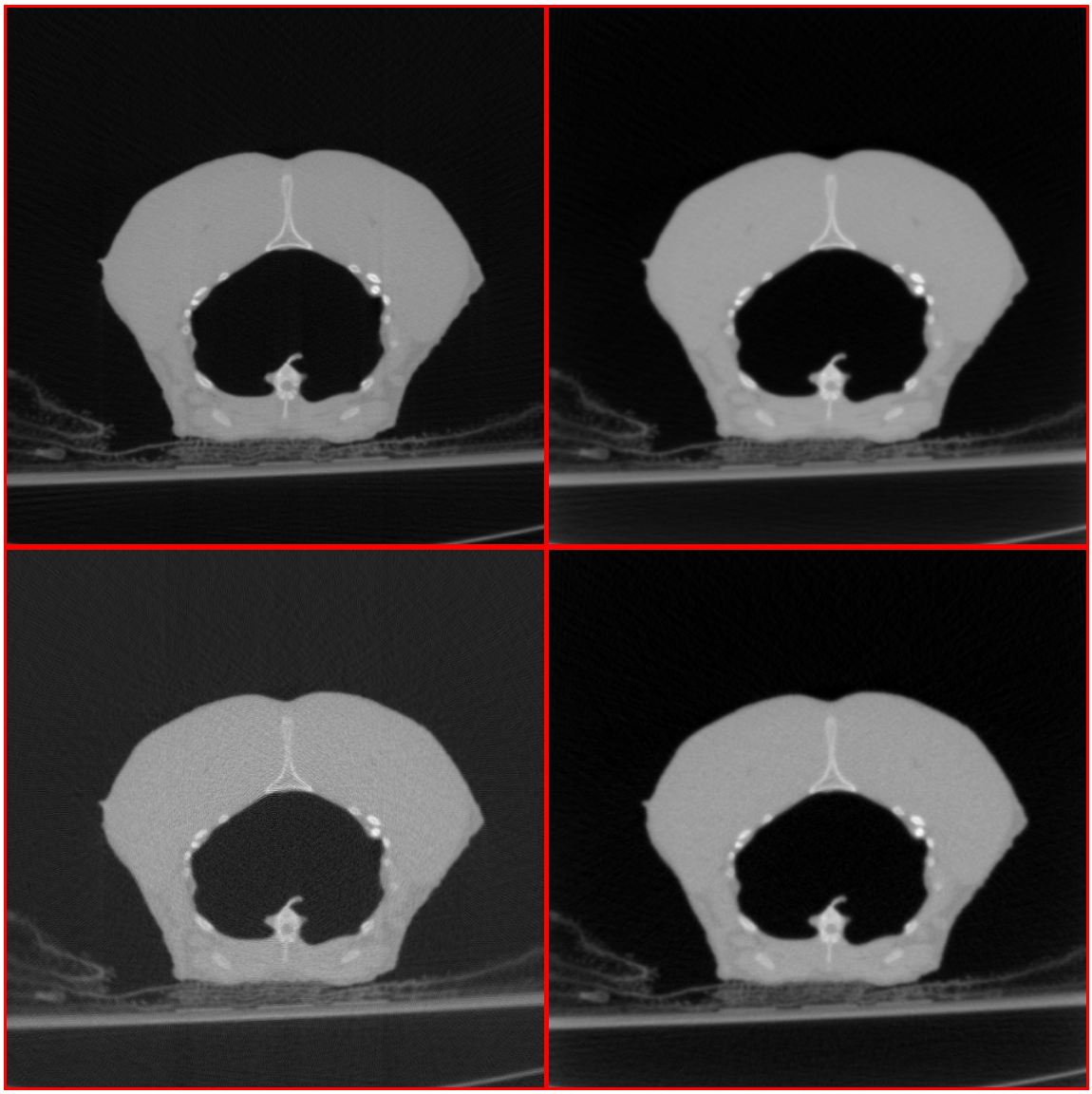

Figure 2.5 Comparison of reconstructed image quality at varying doses for FBP and PS images using 1498 X 580 sinograms at 120kVp and reconstructed at 512 X 512 pixels

Top-L: FBP reconstructed image at 200mAs . Top-R: PS reconstructed at 200mAs.
B-L: FBP reconstructed image at 15mAs . B-R: PS reconstructed at 15mAs.

Figures 2.6 and 2.7 demonstrate the noise in the in-vivo porcine model images reconstructed at extremely low tube current of 25 mAs with 120-kVp tube voltage. Again, the PS reconstructed image remains largely clear of any noise, while the FBP image is severely degraded. To demonstrate the progressive degradation of FBP images with reduction in tube current, Figure 2.8 demonstrates the reconstructed images for tube currents of 100 mAs, 75 mAs and 50 mAs, respectively, at their original resolution. The image reconstructed at 1024 × 1024 pixels has been cropped and scaled to fit in the document. Again we notice increase in the noise levels of FBP reconstructed images with reduction in the radiation dosage while the PS images are by and large unaltered.

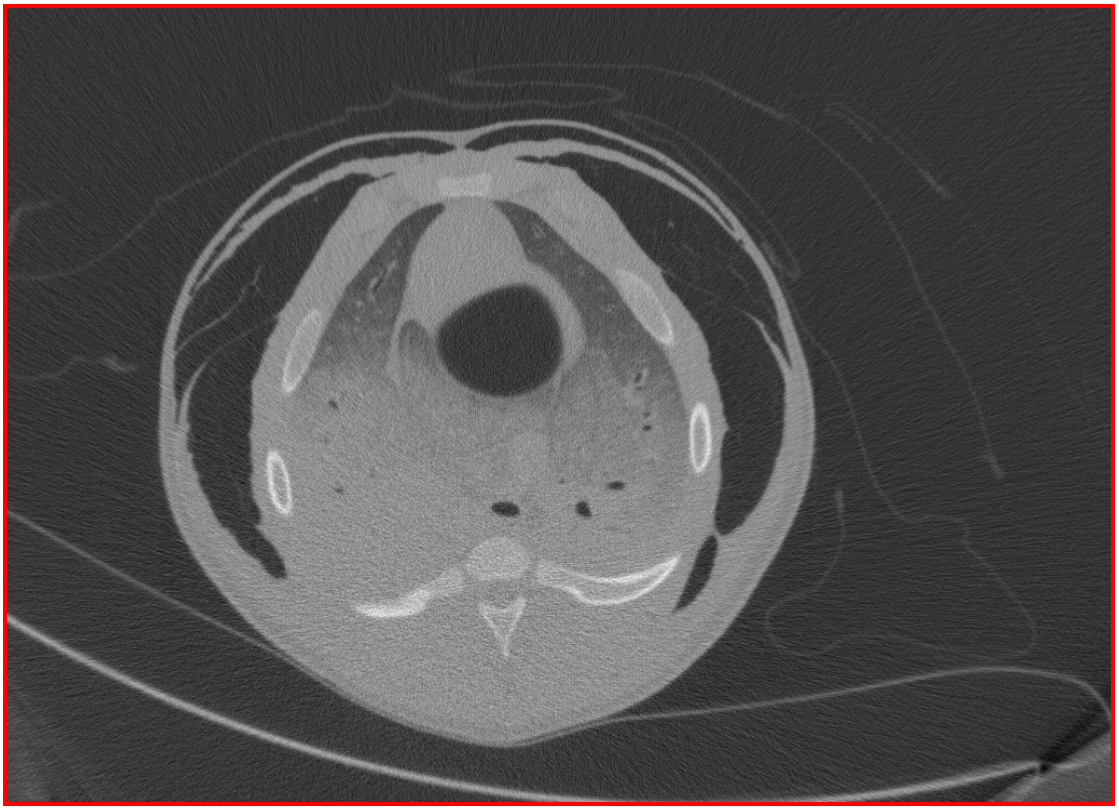

Figure 2.6 FBP reconstructed image using 1498 × 580 sinograms at 1024 × 1024 pixels at 120kVp and 25mAs

Figure 2.7 PS reconstructed image using 1498 × 580 sinograms at 1024 × 1024 pixels at 120kVp and 25mAs

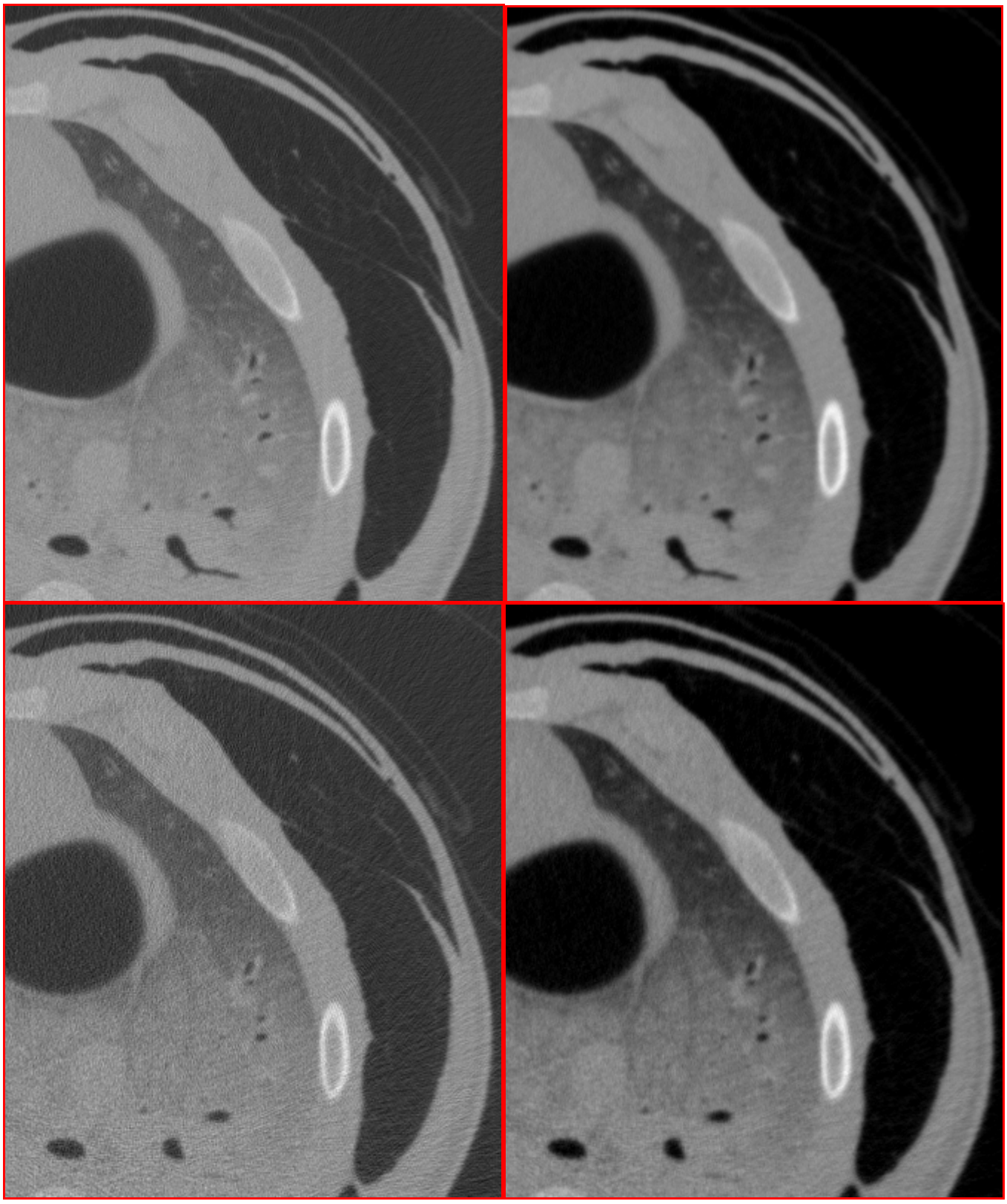

Figure 2.8 Comparison of reconstructed image quality at varying doses for FBP and PS images using 1498 × 580 sinograms at 120kVp and reconstructed at 1024 × 1024 pixels

Top-L: FBP image at 100mAs. Top-R: PS image at 100mAs.
B-L: FBP at 25mAs. B-R: PS image at 25mAs.

To quantify the amount of degradation in the image quality with the reduction of the dose, we used the PSNR as well as the Q index as comparative measures. Since there is no reference image, we use the FBP reconstructed image at 210 mAs tube current at 120 kV peak voltage as the benchmark to compare the degradation of image quality with decrease in radiation dose.

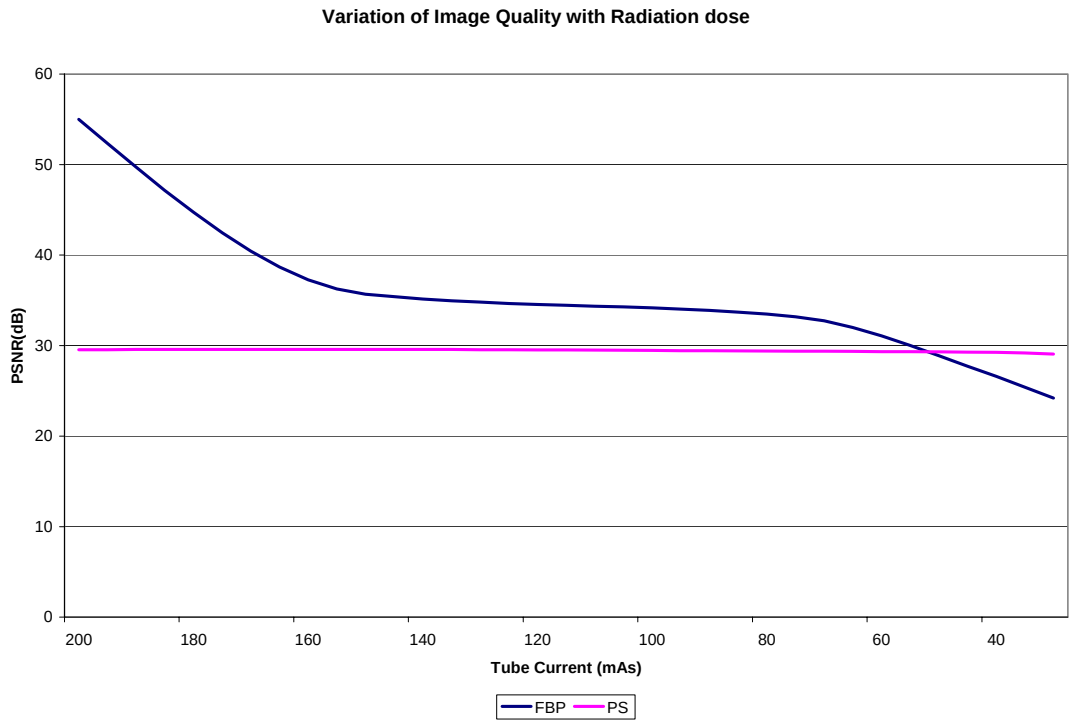

Figure 2.9 Quantitative comparison of FBP and PS reconstructed images at varying radiation doses (PSNR comparison).

The plots in figure 2.9 demonstrate the rapid decrease in the image quality for the FBP images with decrease in radiation dose, while the quality of the PS images remain almost the same even at extremely low radiation doses. Since the PSNR is extremely sensitive to any changes in image contrast, we also use the Q index to compare the image quality at low radiation doses. We can again see that the Q index for the FBP images falls drastically with decrease in radiation dose, while the Q index for the PS images suffers only a slight decrease even at extremely low radiation doses. Thus we prove that the PS algorithm can be effectively used for reconstruction of CT transmission images at low radiation doses.

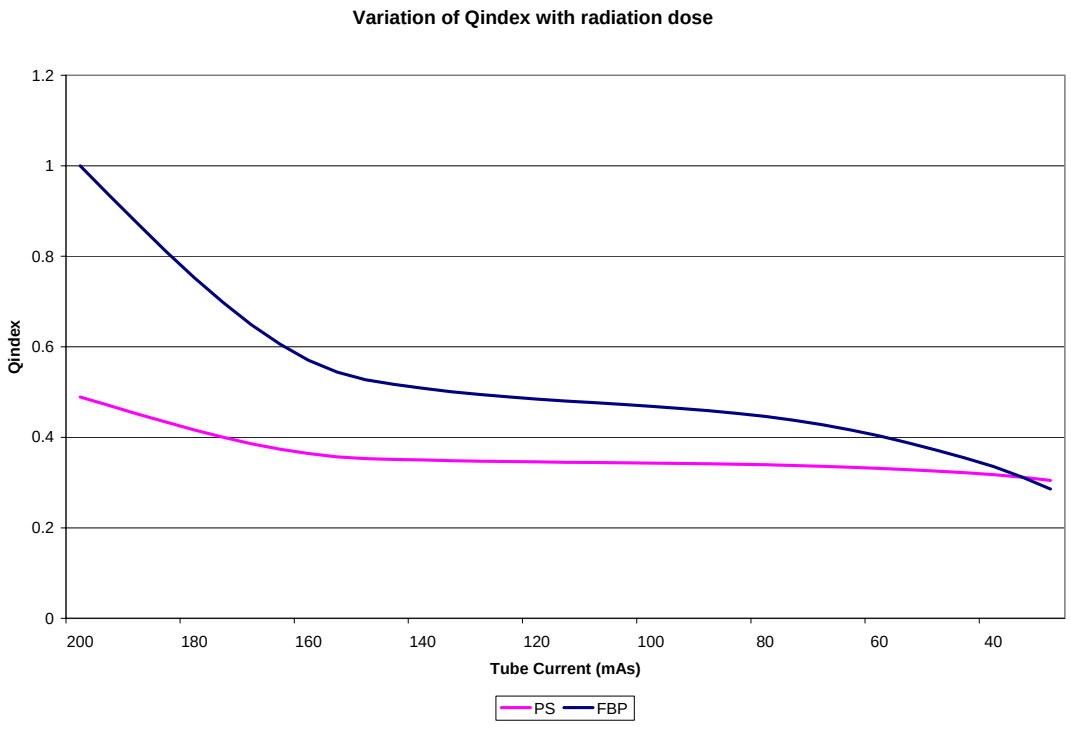

Figure 2.10 Quantitative comparison of FBP and PS reconstructed images at varying radiation doses (Qindex comparison).

The PS algorithm due to its iterative nature also presents other advantages. It can be used for metal artifact reduction to minimize the streaking effects caused due to

scatter and extreme attenuation of photons by metallic objects. Since IGIs involve use of metallic surgical instruments in the field of view of the CT scanner, metal artifacts can be expected in FBP reconstruction. A number of investigations [52][53] have shown that iterative algorithms perform better in reconstructing images when there is attenuation from metallic objects in the image scan.

Thus it is clear that reconstruction of CT transmission data using iterative statistical techniques, such as the PS algorithm, results in improved quality of the reconstructed images when the scans are acquired at extremely low doses. Moreover other benefits such as better control over metal artifact reduction also suggest that the iterative techniques are better suited for Image Guided Interventions.

Chapter 3: High-Speed Reconstruction Using Ray-Tracing Methods

3.1 Acceleration of PS algorithm

The PS algorithm is computationally very intensive as compared to the FBP algorithm. Each iteration of the PS algorithm consists of one back projection and one forward projection apart from other operations such as computing exponentials, subtractions and divisions. The FBP algorithm, on the other hand, has only one back projection and is not iterative in nature. The forward and back projection operations are computationally very intensive as they require the calculation of the length of intersection of each ray with every pixel in the image. Though these values remain constant for a particular geometry and can be pre-calculated, the size of the pre-computed values becomes extremely large as the image size increases. For example, for a typical geometry with 1498 detectors and 600 views, if the image is reconstructed at a resolution of 1024 × 1024, the weight matrix will have 942 billion entries. Though most of these will be zeros and the matrix can be stored in lean matrix format [57][58], managing the weight matrix for different configurations and different reconstruction resolutions becomes unwieldy.

In the past, ray-tracing [17-19] has been effectively used to perform the forward and back projection operations for 3D cone beam reconstruction algorithms. We propose to leverage the ray-tracing mechanism to digitalize the implementation of the PS algorithm for axial slices. This version of the algorithm gives an approximately 30

times speedup over the analog pre-computed weights implementation while drastically reducing the memory requirements.

3.2 Forward and Back Projection using Ray Tracing

3.2.1 The Forward Projection Process

The forward projection is an operation common to most iterative or algebraic reconstruction methods. The forward projection mimics the scanner and creates projections from the image estimates.

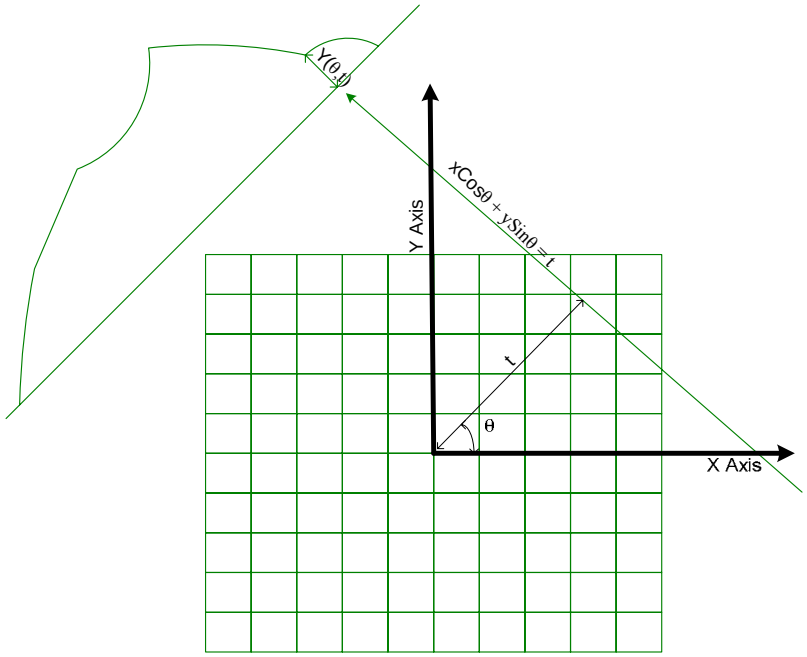

Figure 3.1 Analytical view of the CT reconstruction process

Given an image (map of attenuation coefficients) - μ(x,y) , the projection at angle 'θ' and at a distance 't' from the center of the image will be the projection along the line given by the equation.

$$xCos(\theta) + ySin(\theta) = t \tag{3.1}$$

The projection, Y(θ,t) can be given as line intergral at (θ,t) calculated as

$$Y(\theta, \mathrm{t}) = \int_{(\theta, \mathrm{t})} \mu(\mathrm{x}, \mathrm{y})\, \mathrm{ds} \tag{3.2}$$

This can be rewritten as

$$Y(\theta,t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu(x,y)\delta(xCos\theta + ySin\theta - t)dxdy \tag{3.3}$$

Since the value of μ is known only at discrete points (i,j) at the center of each pixel, we can rewrite the above equation as

$$\mathrm{Y(k)} = \sum_i \sum_j \mu(i, j)a(i, j, k) \tag{3.4}$$

where a(i,j,k) is the length of intersection of pixel (i,j) with the k^{th} ray. Figure 3.2 below shows the forward projection calculation.

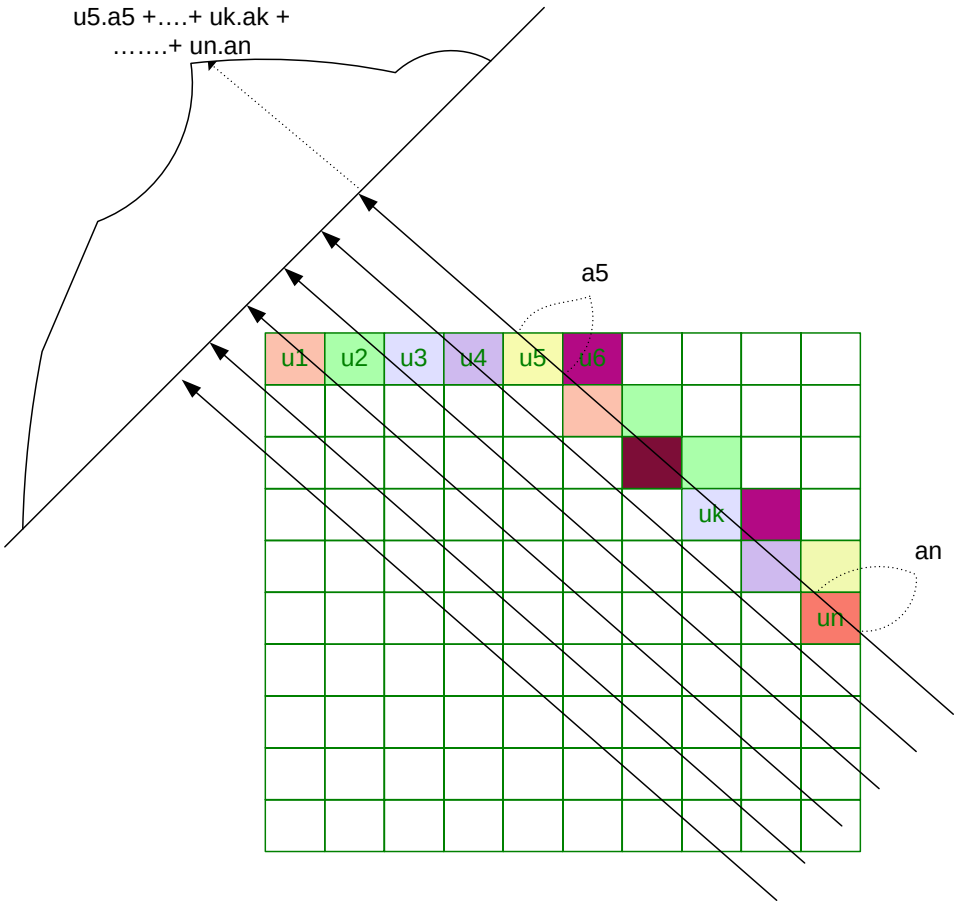

Figure 3.2 The forward projection process

3.2.2 The Ray-Tracing approach to Forward Projection

Various kinds of interpolation-based discrete methods have been used in the past for volume rendering and inverse volume rendering (Reconstruction) of CT data [17-19]. Since we are mainly interested in axial reconstruction, we consider the reconstruction of one axial slice at a time using a bilinear interpolation based ray-tracing approach. The traditional weight matrix based forward reconstruction method assumes fixed regions of uniform attenuation coefficients, defined by the pixels in the image. This kind of discrete representation of the image is natural considering that the image is finally displayed as a 2D grid of intensity values on display devices.

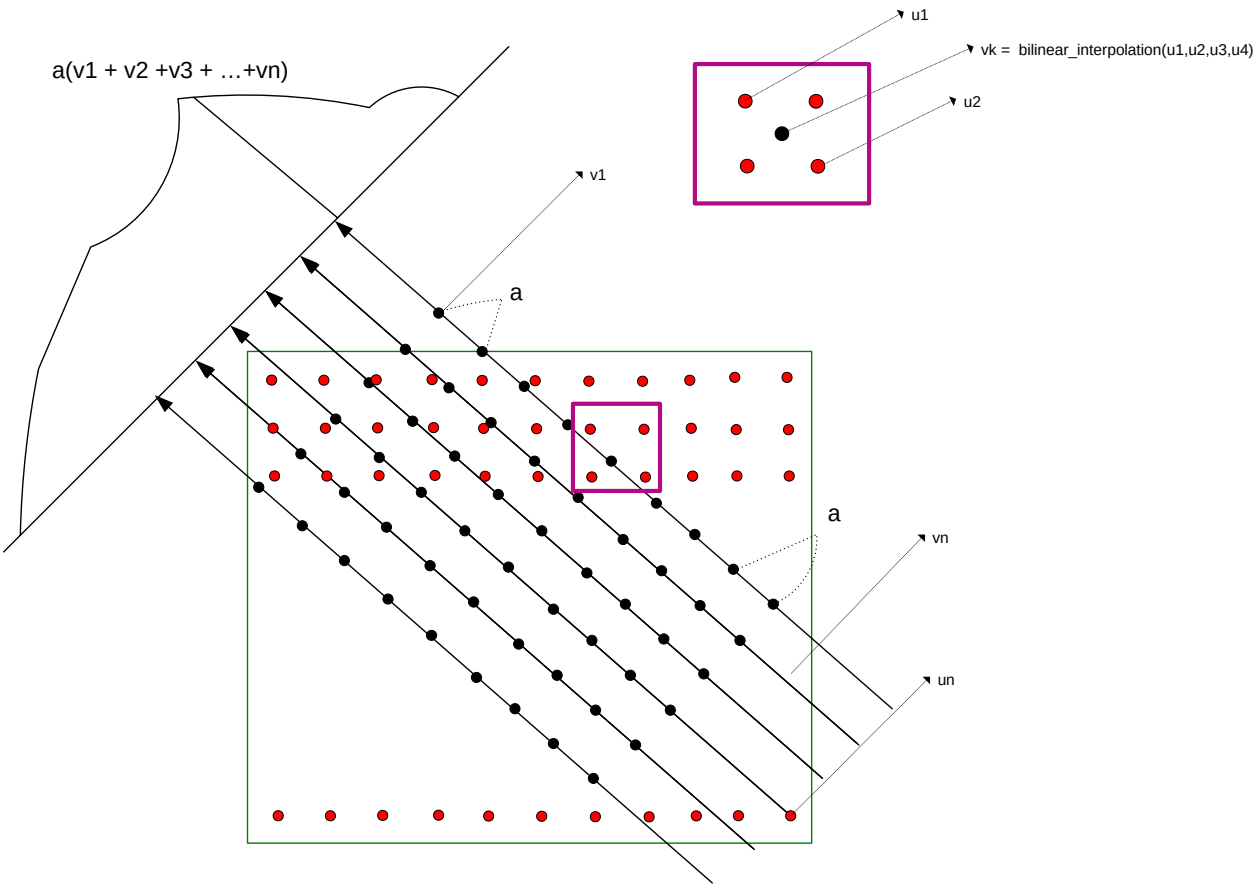

Figure 3.3 Bilinear interpolation based ray tracing approach to forward projection

However, the assumption of fixed regions of uniform attenuation coefficients makes the calculation of the projections extremely compute intensive. In the ray-tracing approach, we assume that the pixel value represents the attenuation coefficient at the center of each square pixel. Thus we only know the attenuation coefficients on the grid points of a uniform 2D grid that falls on the center of the pixels. The values at all other locations can be calculated by bilinear interpolation of the values at the 4 surrounding grid points.

Given the pixel values at grid points, the value at any location (x,y) can be given by,

$$\mu(x,y) = \frac{\mu(\mathrm{i},\mathrm{j})}{(1-\mathrm{x})(1-\mathrm{y})} + \frac{\mu(\mathrm{i}+1,\mathrm{j})}{\mathrm{x}(1-\mathrm{y})} + \frac{\mu(\mathrm{i},\mathrm{j}+1)}{(1-\mathrm{x})\mathrm{y}} + \frac{\mu(\mathrm{i}+1,\mathrm{j}+1)}{\mathrm{xy}} \quad (3.5)$$

where i,j are integers such that $i<x<i+1, j<y<j+1$.

The line integral can now be calculated as the sum of the attenuation coefficients at points unit distance apart along the ray.

$$\mathrm{Y}(\theta,t) = \sum_{j} \mu(tCos\theta - jSin\theta, tCos\theta + jSin\theta) \quad (3.6)$$

where,

$-0.5*(\text{Number of Projections per View}) < j < 0.5*(\text{Number of Projections per View})$

and $j \in$ Integers

Figure 3.3 demonstrates the ray-tracing approach to forward projection using bilinear interpolation.

3.2.3 The Back Projection Process

Back projection is the inverse of forward projection mechanism. It is mainly used to smear back the projection values from the sinograms onto the pixels in the image. Each sinogram value is distributed among the pixels proportional to the length of the ray intersecting the pixel. Suppose Y(θ,t) is the projection value at an angle 'θ' and at a distance 't' from the center of the image, then the back-projected value of any pixel at location (x,y) can be given by,

$$\mu(x,y) = \int_{-\infty}^{\infty}\int_{0}^{\pi} Y(\theta,t)\delta(xCos\theta + ySin\theta - t)d\theta dt \quad (3.7)$$

Since the value of Y(θ,t) is known only at discrete points (θ,t), we can rewrite the above equation as

$$\mu(i,j) = \sum_{k} a(i,j,k)\mathrm{Y(k)} \tag{3.8}$$

Where Y(k) is the discrete value of Y(θ,t) at a particular point (θ,t) and a(i,j,k) is the length of intersection of the ray Y(k) with the pixel at (i,j).

Figure 3.4 demonstrates the back projection procedure.

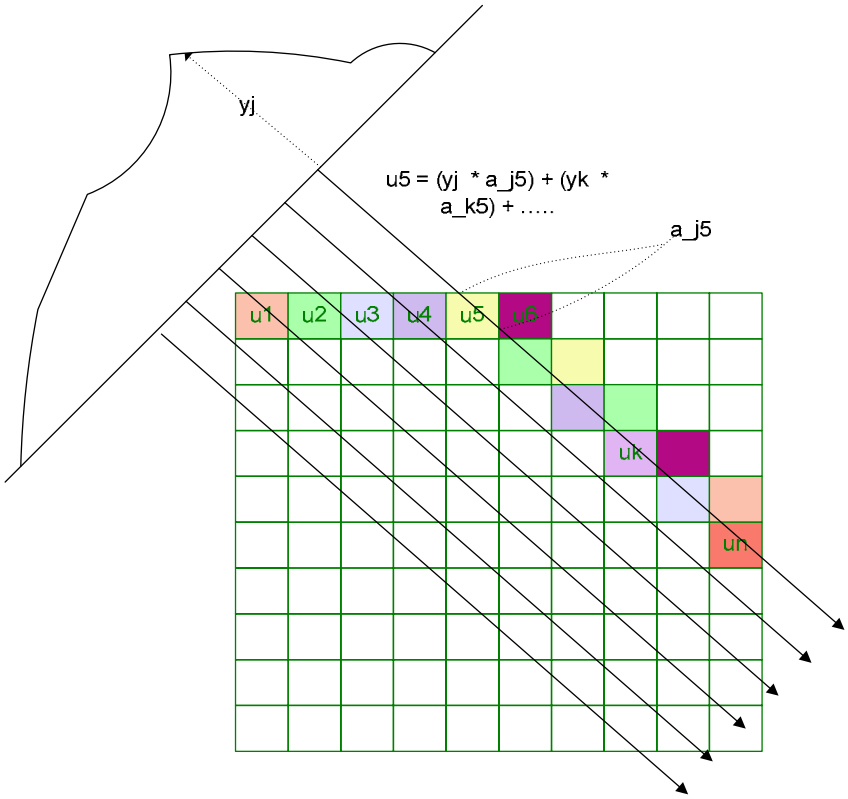

Figure 3.4 The back projection process

3.2.4 The Ray-Tracing approach to Back Projection

The back projection can also be implemented using ray-tracing and bilinear interpolation in a manner similar to forward projection. Here, we replicate the value of the projection all along the projective ray. The values are replicated at unit

distances along the ray. The value at each of the pixel centers due to one particular view can now be obtained by bilinear interpolation of these back-projected values. Given the back projected values, v(x,y) at points along the ray for view θ, the contribution of the view at pixel center, $\mu_\theta(i,j)$ can be calculated using the equation 3.5.

The total value at each pixel due to all the views can now be calculated as the sum of the values due to each of the views.

$$\mu(i, j) = \sum_{\theta} \mu_\theta(i, j) \tag{3.9}$$

Where,

θ varies from 0 to 180 or 0 to 360 depending on the number of views in the sinogram and i,j ∈ Integers.

The idea is demonstrated in figure 3.5.

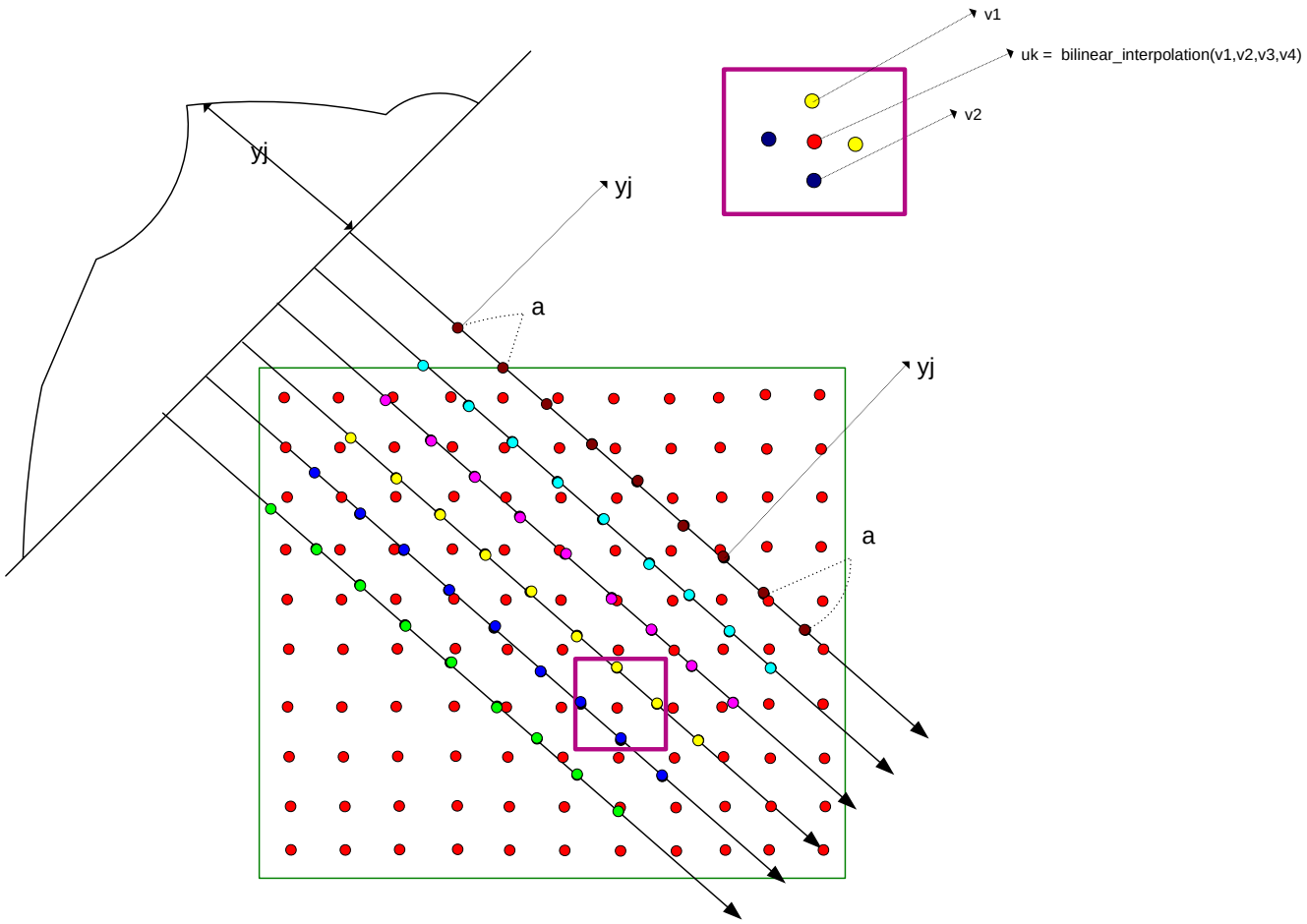

Figure 3.5 The bilinear interpolation based ray tracing approach for back projection

3.3 Implementation and results

3.3.1 Methods and setup

The PS algorithm was implemented as suggested in Erdogen and Fesslar [5] as follows:

$$\mu_j^{n+1} = \mu_j^n + \frac{\sum_{i=1}^{Ny} a_{ij} h_i^n - \beta \sum_{k=1}^{K} c_{kj} w_k [C\mu^n]_k}{\sum_{i=1}^{Ny} a_{ij} a_i y_i} \qquad (3.10)$$

$$h_i^n = b_i e^{-\sum_{j=1}^{Nx} a_{ij} u_j} - y_i \qquad (3.11)$$

Here 'i' is a ray counter varying from 1 to 'N_y'. 'j' is a pixel counter varying from 1 to 'N_x'. 'b_i' is an estimate of the initial number of photons from the air scan. 'a_{ij}' is a measure of the length of intersection of the i^{th} ray with the j^{th} pixel. 'y_i' is the value of the sinogram for the i^{th} ray. 'μ_j' represents the value of the j^{th} pixel. And a superscript gives the iteration number for any of the variables.'β' is the scaling factor for the penalty function and the penalty function is as described in [5].

As demonstrated in equations 3.10 and 3.11, each iteration consists of two back projection operations as well as one forward projection. However, since the denominator does not vary across iterations, it can be calculated once. Hence every iteration consists of one forward projection and one back projection.

The sinograms were first obtained from the scanner for the axial scans. The sinograms so obtained were first pre-processed for bad detectors. They were then normalized using an air scan to ensure uniformity among the detector readings. The

Beer's law was used to estimate the photon counts from the sinograms obtained. This states that the number of photons that reach a detector 'y_i', can be given by

$$y_i = b_i e^{-\sum_{j=1}^{Nx} a_{ij} u_j} \tag{3.12}$$

where the notation is as described above and the sinogram from the scanner represents the values,

$$\sum_{j=1}^{Nx} a_{ij} u_j \tag{3.13}$$

The sinogram so obtained is then rebinned to obtain the parallel beam projections that are used in the reconstruction process.

The PS algorithm was implemented using pre-computed weights as well as ray-tracing algorithm. For the pre-computed weights method, the exact weights matrix (system matrix) was pre-calculated based on the system geometry. The matrix contained the length of intersection of each of the rays with every pixel in the image ('a_{ij}'). This was then used to implement the algorithm as represented in equation 3.10. The images reconstructed using this method were taken as the baseline for comparison. Though the weight matrix was not optimized for size, for small images, for which the entire weight matrix can be loaded into the main memory, the execution time can be considered as being representative of the actual implementation time.

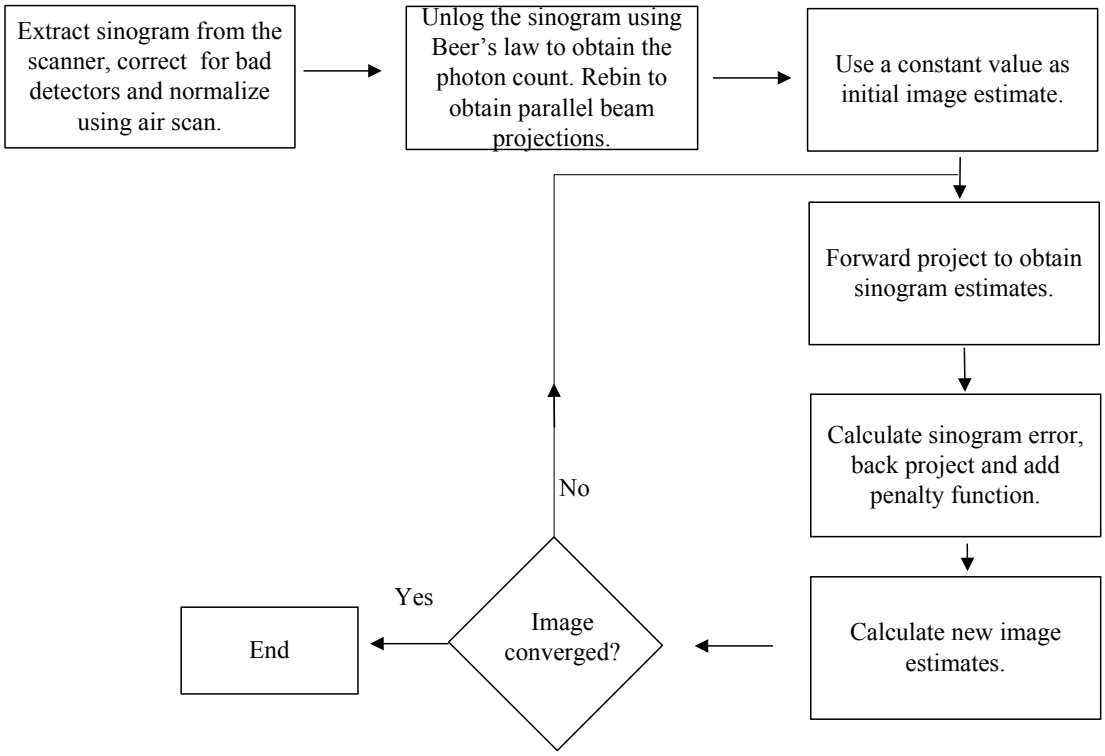

Figure 3.6 The ray tracing algorithm

For the ray-tracing method, ray-tracing and bilinear interpolation operations were used as described in sections 3.2.2 and 3.2.4 for the forward projection and back projection operations. The algorithm was implemented as depicted in figure 3.6.

3.3.2 Results and conclusion

To exactly estimate the difference between the images reconstructed using the analog (pre-computed weights) methods and the ray-tracing algorithm, we used a scanner image as the benchmark image. Projections were then created from this image at 580 views with 672 detectors per view. The sinograms so obtained were then reconstructed using both the analog (pre-computed weights) as well as the ray-tracing based methods. The PSNR with respect to the initial scanner image was used as a measure to compare the quality of reconstruction from the two methods.

Figure 3.7 compares the quality of the reconstructed images using the analog and the ray-tracing methods using the PSNR as a metric while figure 3.8 compares the same using the Q index as a metric. It is clear that the ray-tracing method increases monotonically at a rate similar to the analog method. It can also be observed that the quality of the images after the same number of iterations remains similar.

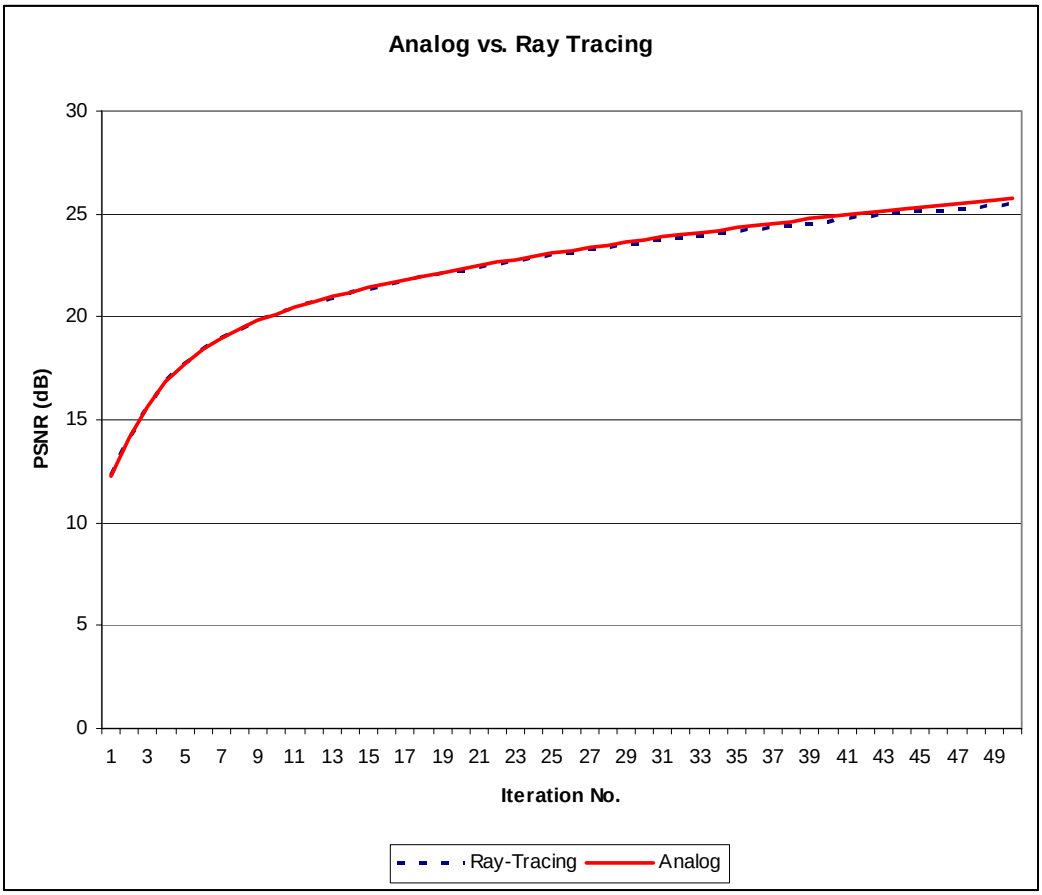

Figure 3.7 Quantitative comparison of reconstructed image quality using PSNR for Analog vs. Ray Tracing methods for images reconstructed to 512 × 512 pixels using 580 × 672 sinogram.

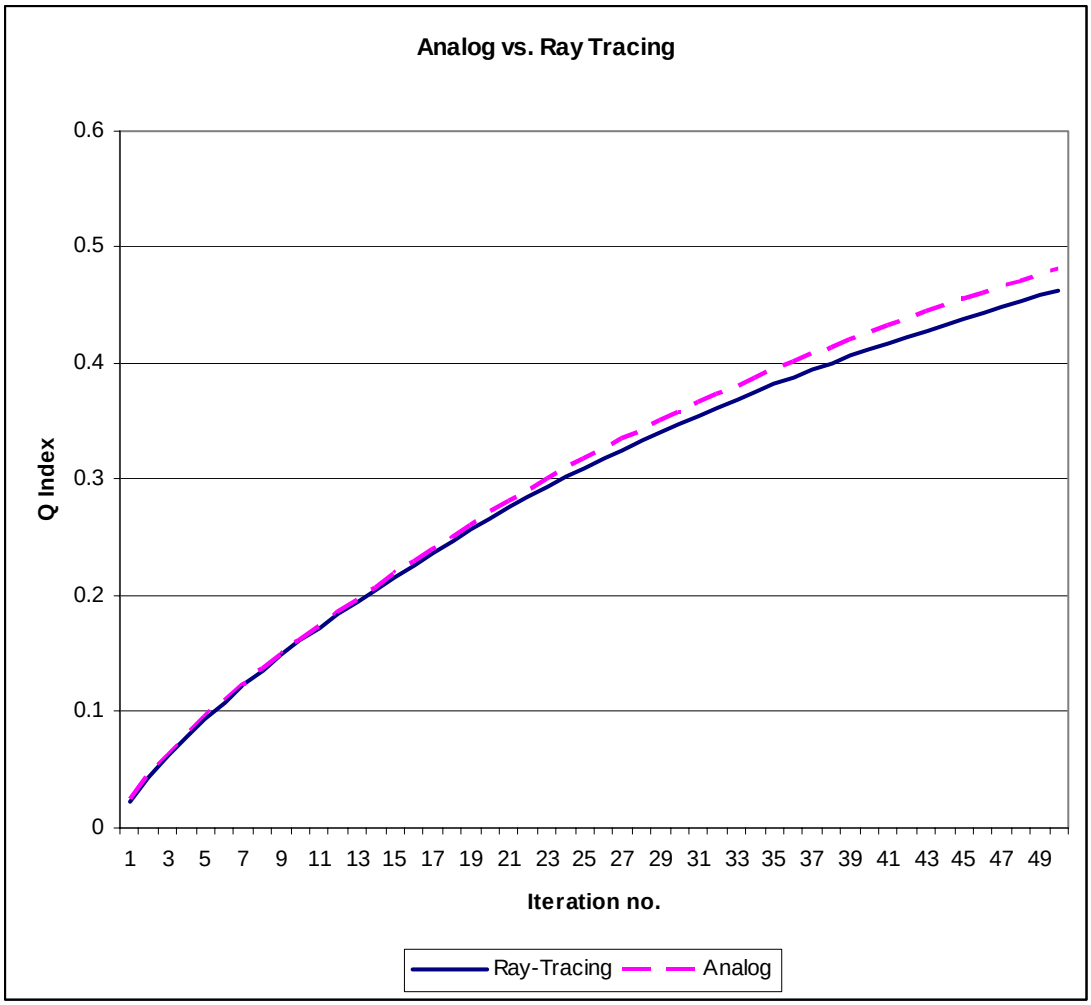

Figure 3.8 Quantitative comparison of reconstructed image quality using Qindex for Analog vs. Ray Tracing methods for images reconstructed to 512 × 512 pixels using 580 × 672 sinogram.

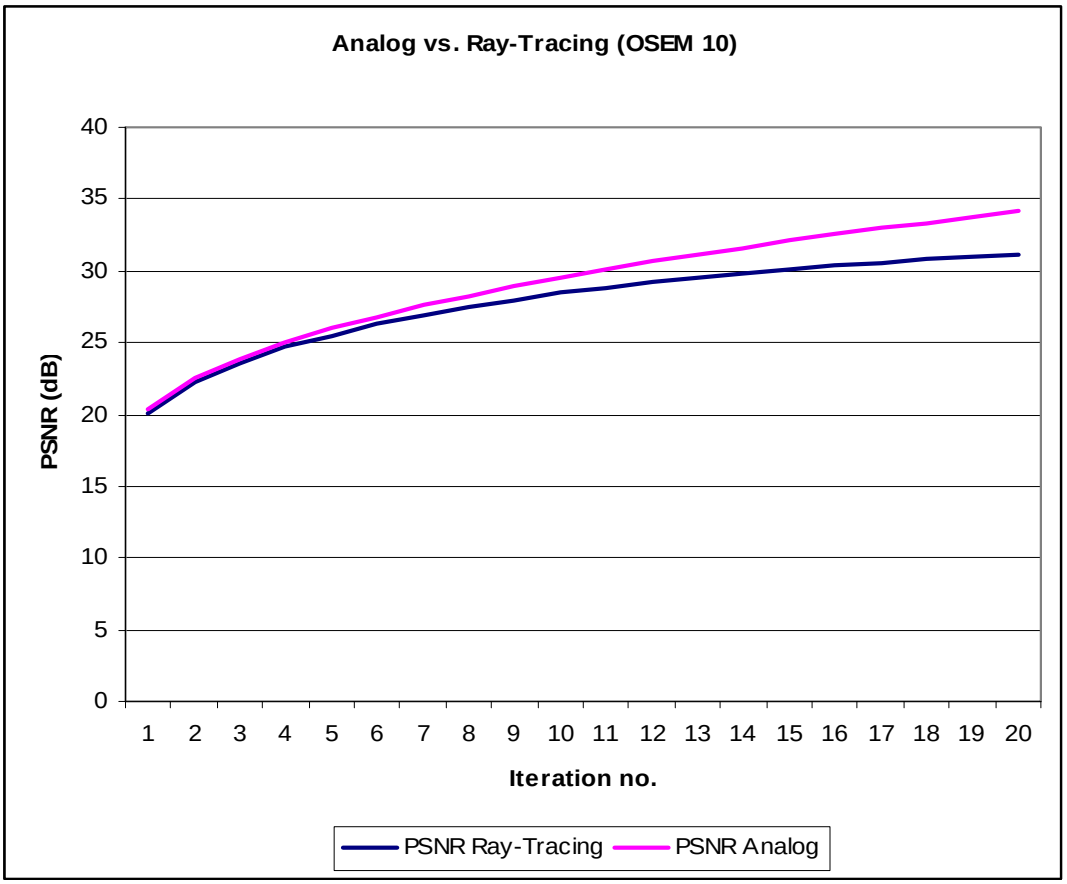

Figure 3.9 Quantitative comparison of reconstructed image quality using PSNR for Analog vs. Ray Tracing methods for images reconstructed to 512 × 512 pixels using 580 × 672 sinogram and 10 subsets per iteration.

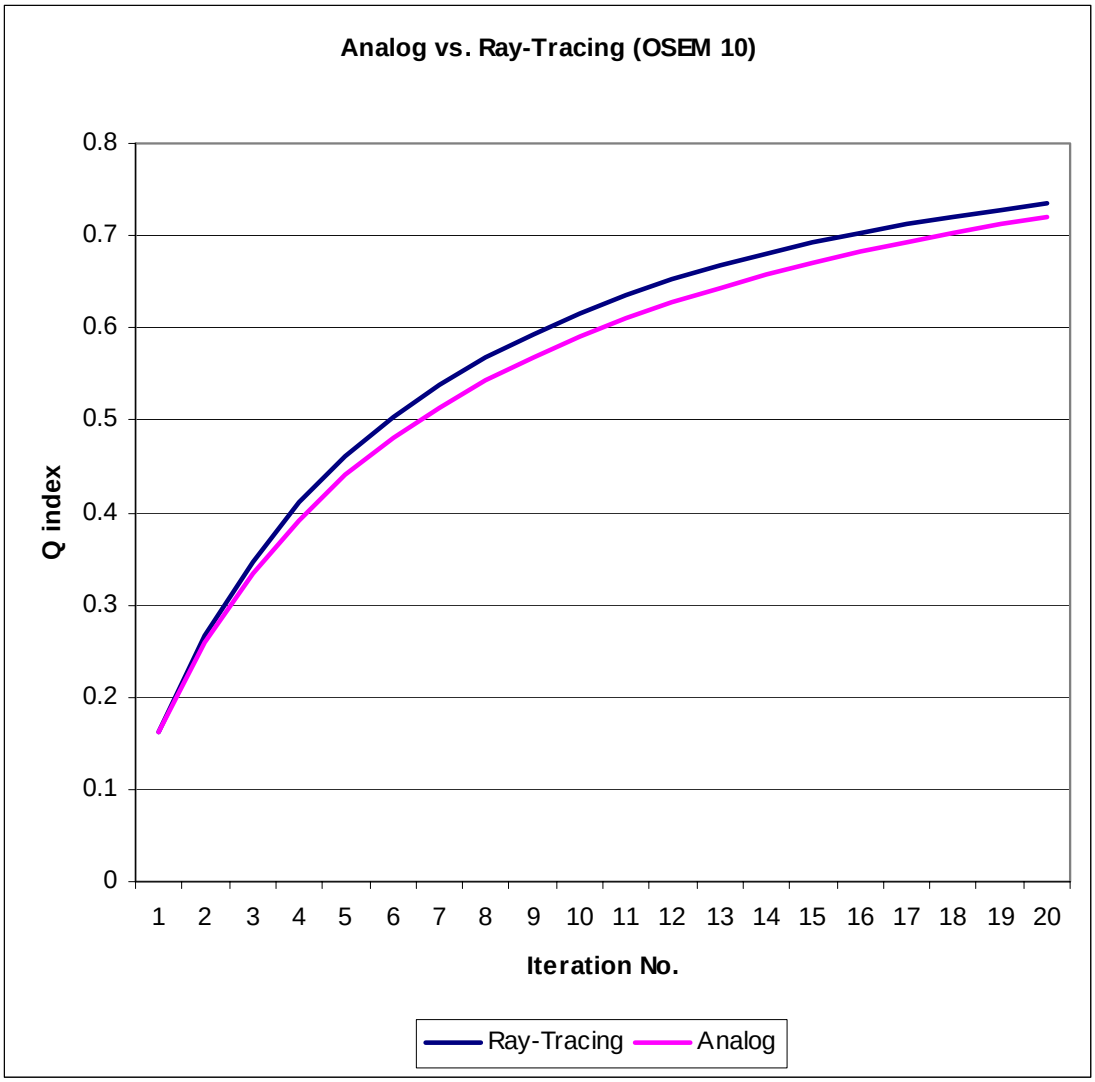

Figure 3.10 Quantitative comparison of reconstructed image quality using Qindex for Analog vs. Ray Tracing methods for images reconstructed to 512 × 512 pixels using 580 × 672 sinogram and 10 subsets per iteration.

In Figures 3.9 and 3.10, the PSNR and the Q index are again used to compare the quality of the reconstructed images using the two methods. The images, however, are reconstructed using the OSEM version of the two algorithms with 10 subsets per iteration. It can be noticed that while the PSNR reports the quality of the analog (pre-computed weights) image as slightly better than the ray-traced image, the Q index reports vice versa. However, from the absolute values of the PSNR and the Q index, it

is clear that both methods are monotonic in nature and give good quality images. Finally figures 3.11 and 3.12 display the images reconstructed via the two methods for qualitative comparison. It is clear that the ray-tracing method gives image quality that is comparable to the analog method (images cropped to fit document).

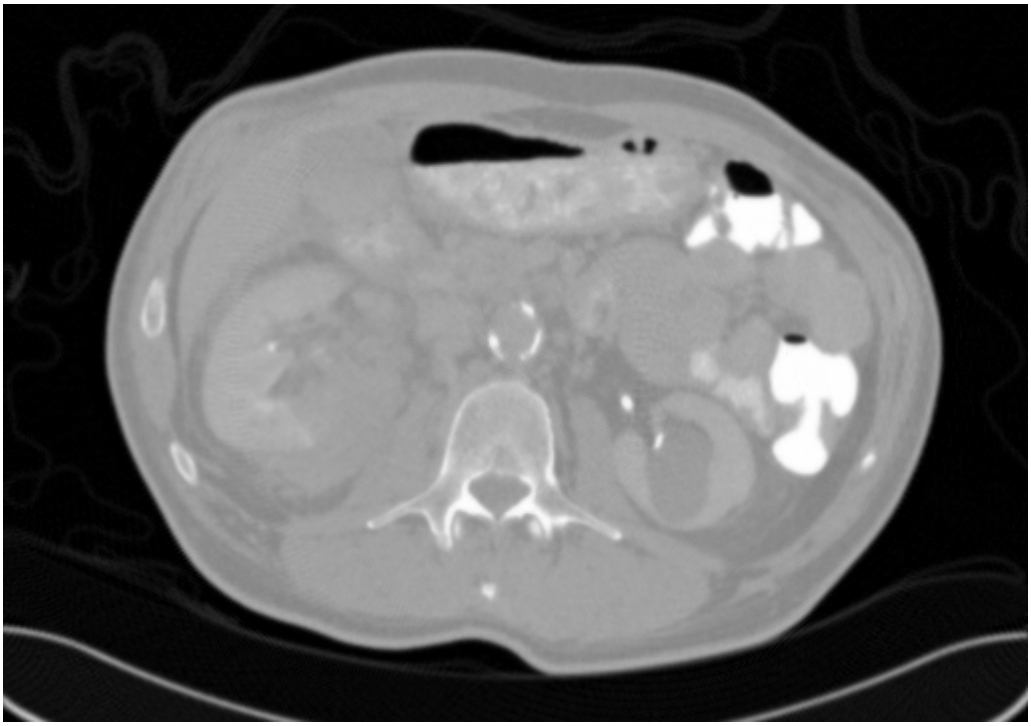

Figure 3.11 PS reconstructed image after 20 iterations of Analog algorithm (OS-10)

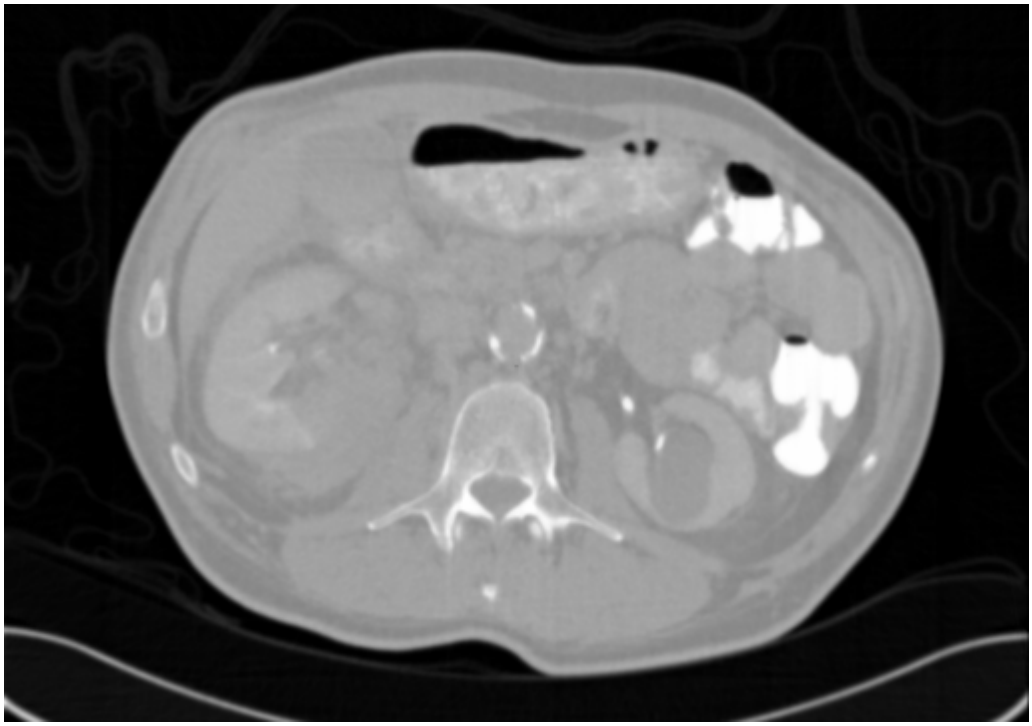

Figure 3.12 PS reconstructed image after 20 iterations of Ray-Tracing algorithm (OS-10)

The table 3.1 below gives the time taken for reconstruction of the images using the Analog and Ray-Tracing algorithms. The analog version of the algorithm was a direct implementation of the equation 3.10 using a non-sparse representation of the pre-computed weight matrix. All algorithms were run on single core of the dual-core Intel Xeon at 2.33GHz with 4 GB of memory. The pre-computed weight matrix was such that it completely fit into the main memory. The time per iteration is inclusive of all the memory operations performed in between two iterations.

Sinogram Size	*OSEM Subsets*	*Reconstructed Image Size*	*Time per Iteration*		*Speedup*
			Analog	*Ray Tracing*	
367 × 300	1	256 × 256	207	6.68	30.98
367 × 300	10	256 × 256	244	6.68	36.52
672 × 580	1	512 × 512	3122	39.58	78.87
672 × 580	10	512 × 512	3220	39.59	81.33

Table 3.1 Speed-ups achieved by using the ray-tracing methods for reconstruction

Thus it is clear that the suggested ray-tracing algorithm gives speedups of at least 30X over the analog version of the same algorithm. The speedups achieved increase with the increase in the sinogram and image sizes. This is mainly due to the increased memory access delays due to a larger number of cache misses. The ray-tracing algorithm also does not result in undue degradation of image quality and gives good quality reconstructed images.

Chapter 4: Hardware-based Acceleration of PS Algorithm for Low-Dose CT Reconstruction

4.1Cluster-based acceleration scheme

4.1.1 Introduction and previous work

With the recent advances in VLSI technologies, the total number of transistors per chip continues to increase. The increase in the number of transistors along with the decrease in the average half-pitch and feature size has led to ever faster computers with large amounts of main memory. However, better, faster and extremely sensitive data acquisition techniques have led to almost explosive amounts of data that are collected and need to be processed. Since most applications require similar operations to be performed on these large data sets, clusters of processors working in parallel have emerged to be the most preferred form of hardware accelerators [3] [20-25].

In the past, many groups have used clusters of generic computers to accelerate reconstruction of CT [22][25]. However, most of these approaches have been directed towards the widely used 3D FBP algorithms and reconstruction of PET and SPECT images. Supercomputers as well as mainframe parallel computers have also been used for CT reconstruction. In 1989, Guerrini et al. [20] used the Vector computer, while Chen et al. [24] used the hypercube to accelerate CT reconstruction. Other mainframe parallel computing approaches include mesh-parallel approach by McCarty et al. [22] in 1991 and the transputers by Atkins et al. [23] in 1991. We use a cluster of

computers to accelerate the PS algorithm for reconstruction of low-dose CT scans. Since the iterative algorithms are extremely compute intensive we use a cluster of machines, each having 2 dual-core processors for accelerating the algorithm.

4.1.2 The Cluster Setup

Our cluster has 8 nodes each consisting of 2 dual-core Intel Xeon processors running at 2.33 GHz. Each of the nodes has a 4GB main memory that is shared by the processors. The nodes are connected together using a 1Gbps Ethernet switch. The individual nodes run Red Hat Linux. The Portable Batch System (PBS) is installed on the cluster for efficient management of resources and jobs. Each of the nodes can run 4 independent threads on the 4 cores to give effectively 32 independent processing cores. Message Passing Interface (MPI) is used for communication between the individual cores. MPI is a library specification standard for message passing [26][42]. We used the MPICH2 [43] implementation of MPI that is freely available and extremely efficient for our purposes. Figure 4.1 gives an overview of the cluster setup.

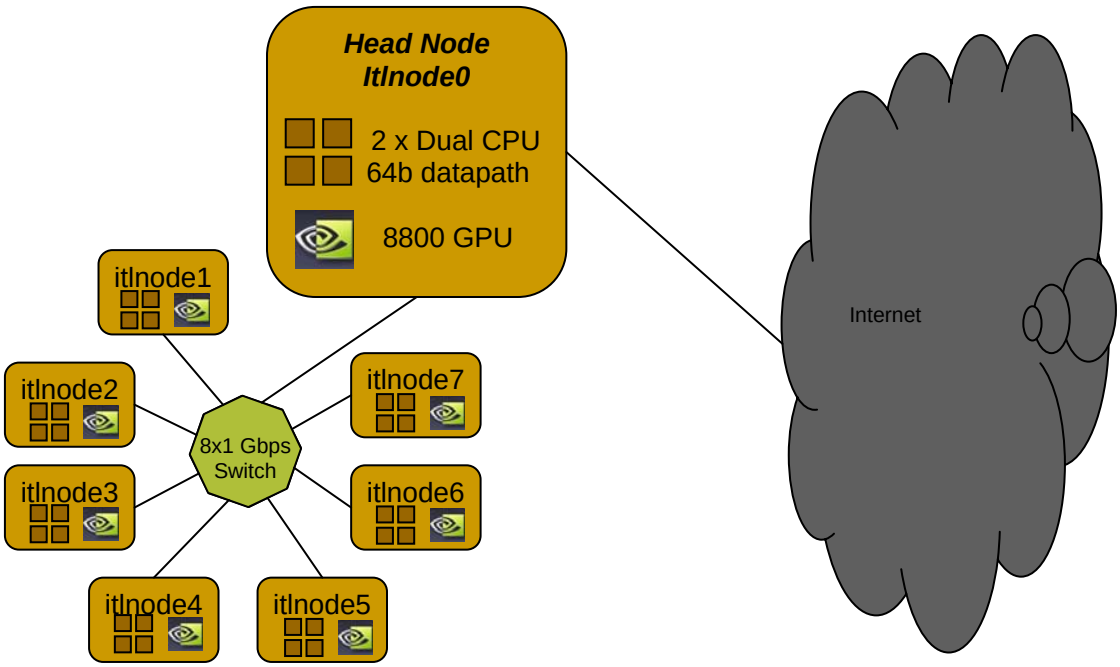

Figure 4.1 Overview of the cluster setup at ITL

Each machine on the cluster also has an NVIDIA GeForce 8800 GTX GPU card installed. The GPU card can be programmed using NVIDIA's Compute Unified Device Architecture (CUDA). CUDA is an extension to the generic C programming language and gives the user the ability to decide the actual placement of the data on the GPU and the distribution of the computations among the GPU stream processors.

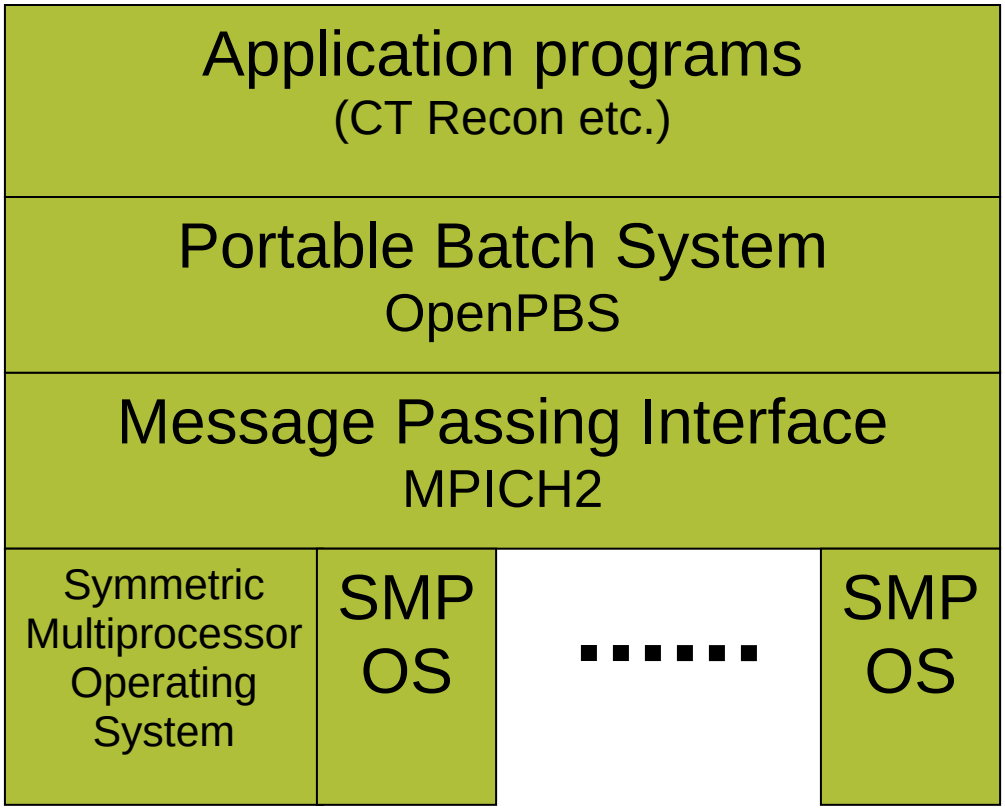

Figure 4.2 The Software stack overview for the cluster

4.1.3 Implementation

The ray-tracing based algorithm was implemented on the cluster in a manner similar to the single CPU implementation as described in section 3.3.1. The 3 main operations of the algorithm are

a) Forward Projection.

b) Back Projection.

c) Pixel Update.

Figure 4.3 gives the relative time taken for the execution of each of these operations on a single CPU.

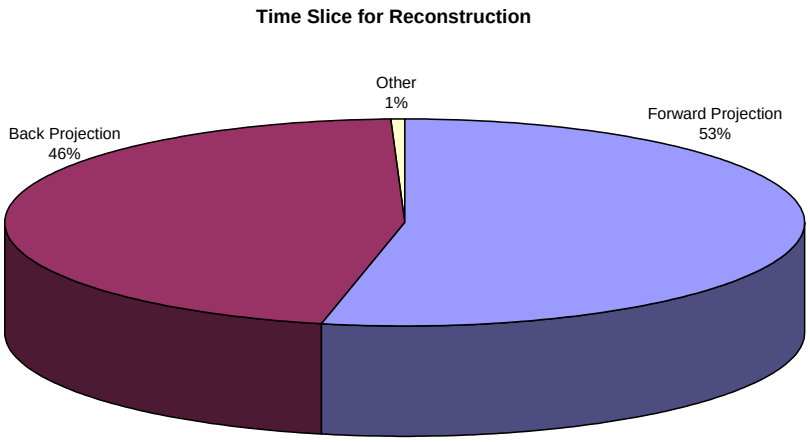

Figure 4.3 Distribution of total execution time across various operations for the ray-tracing based implementation of the PS algorithm

It is clear that the forward and back projection operations consume almost 99% of the total reconstruction time. Hence these operations were targeted for acceleration on multiple nodes. One of the main concerns with multi-processor implementation is the time taken for inter-processor communications. Hence the forward and back projection operations were not implemented separately, but combined, as explained below, to ensure that there will be only one inter-processor communication per iteration.

The work flow is as shown in figure 4.4. The flow can be explained as below.

1) The sinogram is first read by the head node and distributed to all the nodes. Each of the nodes starts with the same initial image.
2) Every node creates a subset of the forward projection.

   If the view ‘V’is such theat ‘V’ mod k = 0, then node k generates the forward projection for view ‘k’.

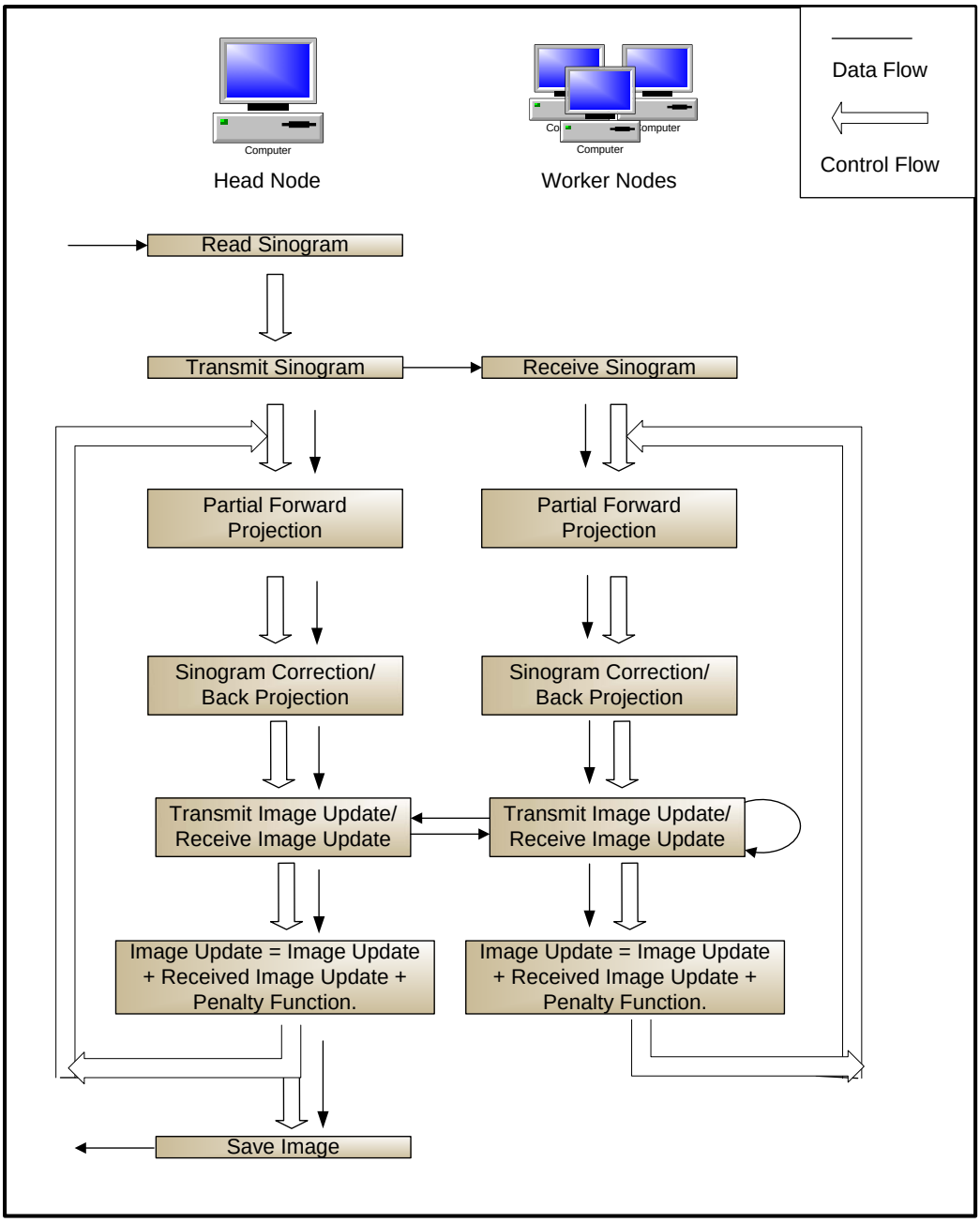

Figure 4.4 Flowchart for the cluster based implementation of the ray-tracing algorithm

3) Each node compares the available sinogram views thus created with the corresponding views of the original sinogram to create corresponding views of the 'difference sinogram'. These 'difference sinogram' views are then back-projected to get partial increments for all the image pixels.
4) All the nodes broadcast their partial image updates to all other nodes. Every node sums up all the partial updates to get the true image update for that iteration.
5) All the nodes calculate the penalty values, and update the image using the updates calculated in step 4.

This completes the iteration. Thus we can see that there is only one inter-processor communication per iteration thus decreasing the I/O overhead substantially.

4.2 GPU-based acceleration scheme

4.2.1 Introduction and previous work

With the rise in the number of graphics intensive applications, the Graphics Processing Unit has been continuously improving in its computational performance. The exponential increase in the computational power and the number of transistors in the GPU has made it extremely attractive to other computationally intensive applications beyond graphics. With multiple cores per processor, the general-purpose computers are also moving towards parallel processing models. In such a scenario, the dozens of stream processors of the GPU present a very attractive model for computationally intensive applications.

In recent years, the GPU is becoming more programmable for non-graphics applications and is being increasingly used as a co-processor for various applications in medical imaging, image processing, molecular chemistry, seismology, databases etc [56]. With the advent of NVIDIA's Compute Unified Device Architecture (CUDA), non-graphics programmers are also able to program commercially available GPU's using simple extensions to the 'C' programming language.

Cabral et al. [17] used the texture mapping hardware for CT reconstruction. This was followed by many works ([16][18][19] etc.) that made use of the texture mapping hardware for CT reconstruction. In [33] a general framework for the use of GPU in reconstruction algorithms was presented. This was mainly based on the use of the graphics pipeline for acceleration of forward and back projection steps. In [48] the GPU was used to accelerate these steps of the convex algorithm using the framework suggested in [33]. Other algorithms such as SART and OSEM have also been accelerated using similar frameworks. All of these implementations relied on languages such as OpenGL and other shading languages that prevented direct programming of the GPU for various kinds of mathematical operations. In [40], the FDK algorithm was accelerated for 3D cone beam geometry using CUDA. CUDA offers many advantages over the traditional shading languages. CUDA gives the developer the complete control over the stream processors. There is no fixed pipeline and the developer is free to exploit the various memory and computational resources to his liking. Hence we chose to use CUDA for the acceleration of our ray-tracing based PS reconstruction algorithm.

4.2.2 The NVIDIA CUDA architecture

CUDA is a software and hardware architecture that enables a programmer to efficiently implement single instruction multiple data program samples on the CUDA enabled NVIDIA GPUs. We made use of the NVIDIA 8800GTX GPU for our implementation. The GPU contains 128 programmable stream processors arranged as 16 SIMD multiprocessors with 8 processors per multiprocessor. The GPU has 768MB of on board memory and a peak theoretical performance of 518Gflops. It has a core clock of 575MHz, a stream processor clock of 1.35GHz, and 900MHz memory.

For ease of execution, CUDA classifies the code to be run on the device as a kernel. Each kernel is essentially a SIMD instruction set. The kernel can be executed on thousands of threads. The CUDA architecture allows the programmer to arrange the threads in the form of a grid as shown in figure 4.5.

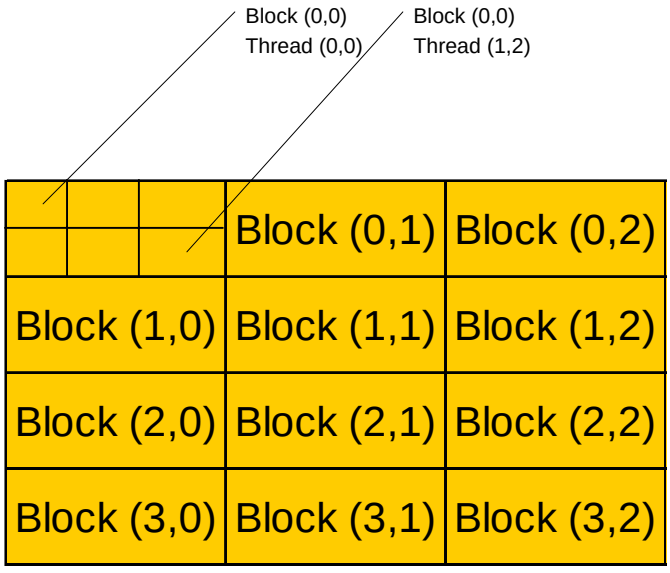

Figure 4.5 The arrangement of blocks of threads into a grid in CUDA

The threads are first grouped together in Blocks. Threads within a block are more tightly tied together and can synchronize amongst themselves. They share a common high-speed memory block and thus can share data faster with other threads within the same block. Each kernel is executed as a grid of blocks. The grid consists of similar blocks that have the same number of threads.

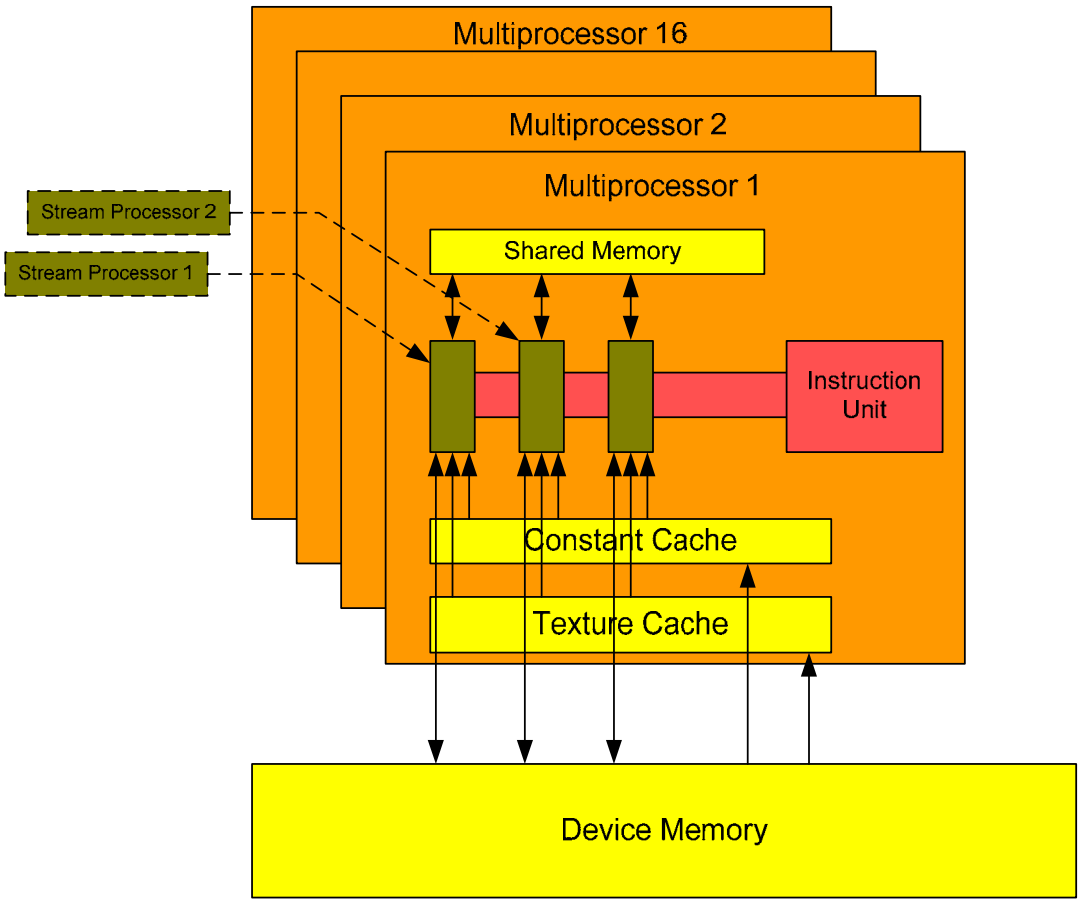

Figure 4.6 The multiprocessor hardware and memory architecture in CUDA

The GPU consists of a number of multiprocessors. Each multiprocessor has a number of stream processors that share the 'shared memory'. Each stream processor has its own set of registers. The stream processors within a multiprocessor have a common texture and constant cache. Figure 4.6 demonstrates the hardware architecture. During execution, all threads within a block are assigned to the same multiprocessor. A multiprocessor may, at a given time, have a number of active blocks that are executed on a time sharing basis.

The CUDA software API's are extensions to the generic 'C' programming language. The software development model is as shown in figure 4.7. The source code that consists of basic C statements as well as CUDA extensions is first given to the CUDA compiler. The compiler processes all CUDA kernel calls and outputs a generic CPU specific C code that is then compiled using a standard C compiler. The GPU specific instructions are output in CUDA Assembly and are further processed by the 'CUDA runtime' and 'CUDA drivers' before being executed on the GPU. The generic C Code compiled for the CPU is executed on the CPU. More details of the architecture can be obtained in [44][45].

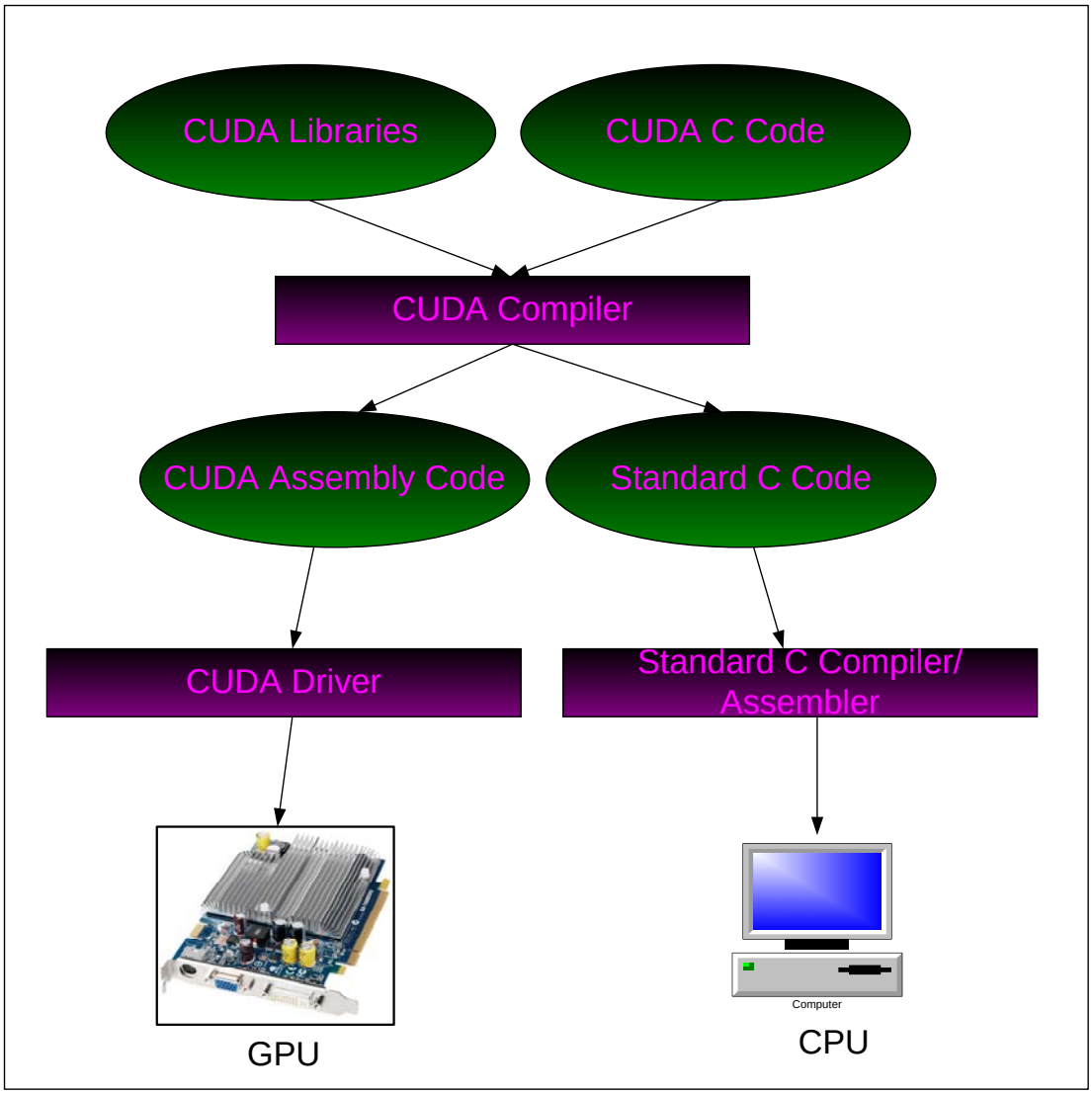

Figure 4.7 The CUDA software development process

4.2.3 Implementation

The NVIDIA 8800GTX was used as the hardware platform for the implementation. The CUDA architecture was used to program the GPU. The entire reconstruction process was implemented on the GPU. The CPU was used to preprocess the sinograms. The sinograms were preprocessed as described in section 3.3.1. Since the texture memory in the GPU has 4 channels for RGB and Alpha, 4 slices of the axial scans were simultaneously processed at any given time.

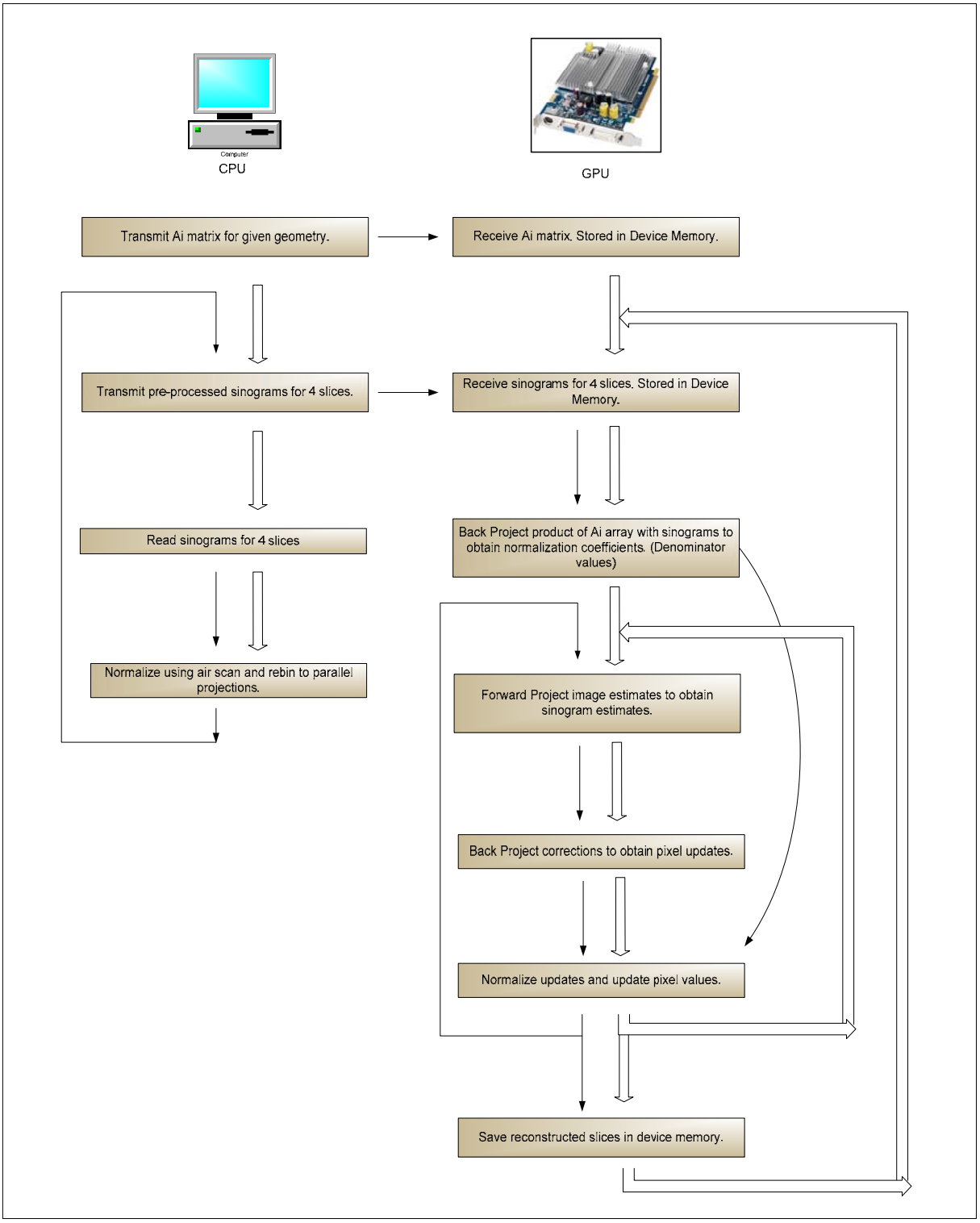

Figure 4.8 CT reconstruction using the PS algorithm on CUDA enabled GPU

Figure 4.8 gives an overview of the overall reconstruction process. The PS algorithm explained in section 2.5 is repeated here for convenience.

$$\mu_j^{n+1} = \mu_j^n + \frac{\sum_{i=1}^{Ny} a_{ij} h_i^n - \beta \sum_{k=1}^{K} c_{kj} w_k [C\mu^n]_k}{\sum_{i=1}^{Ny} a_{ij} a_i y_i} \tag{4.1}$$

$$h_i^n = b_i e^{-\sum_{j=1}^{Nx} a_{ij} u_j} - y_i \tag{4.2}$$

Here 'i' is a ray counter varying from 1 to 'N_y'. 'j' is a pixel counter varying from 1 to 'N_x'. 'b_i' is an estimate of the initial number of photons from the air scan. 'a_{ij}' is a measure of the length of intersection of the i^{th} ray with the j^{th} pixel. 'y_i' is the value of the sinogram for the i^{th} ray. 'μ_j' represents the value of the j^{th} pixel. And a superscript gives the iteration number for any of the variables. 'β' is the scaling factor for the penalty function and the penalty function is as described in [5].

The algorithm consists of 4 main parts:

- One time calculation of the normalizing factor (denominator of equation)
- One forward projection per iteration. (i.e calculation of $\sum_{j=1}^{Nx} a_{ij} u_j$)
- One back projection per iteration. (i.e calculation of $\sum_{i=1}^{Ny} a_{ij} h_i^n$)
- Calculation of penalty function and image update.

Each of these operations is performed in a separate GPU kernel. For each operation, the individual threads always perform the gather operation, i.e., each of the results is assigned to an individual thread. Hence, for the forward projection each thread calculates the contribution of a single ray (i.e. one value in each of the stack of 4 sinograms). For back projection, each thread calculates the image update value for a single pixel in each of the 4 images. In the following sections, we explore some of the salient features of each of the 4 operations.

4.2.3.1 Forward Projection:

For the forward projection operation, each of the threads calculates the value of h_i^n (equation 4.2) for one particular value of 'i'. For the forward projection operation, the sinograms are stored in the device memory. The 'image estimate', from which the 'sinogram estimate' is to be calculated, is loaded into the texture memory. The 4 images are loaded into the 4 channels of the texture memory. The sinogram is first divided into a grid as shown in figure 4.9.

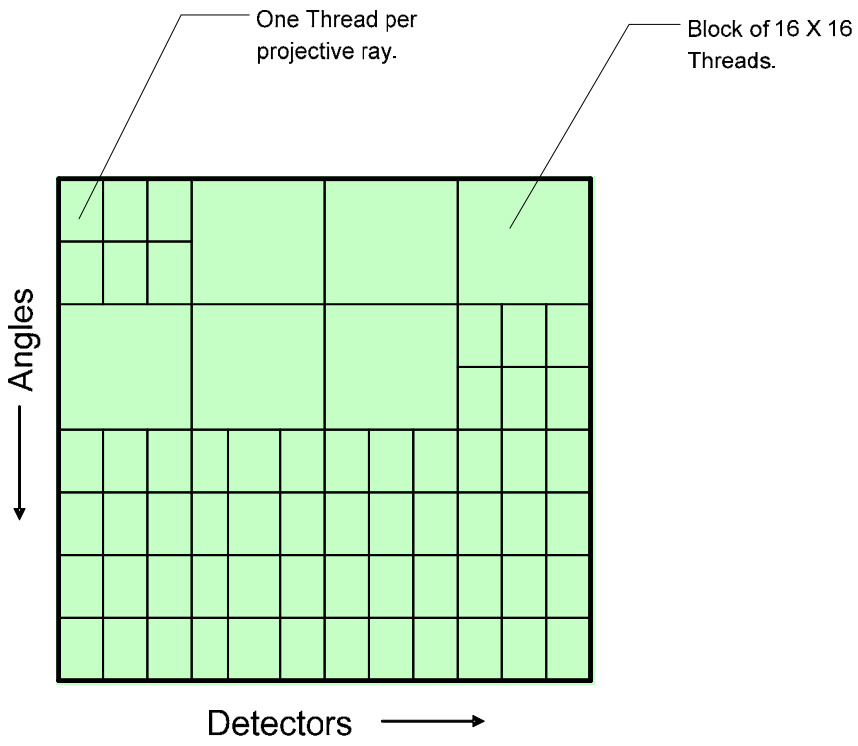

Figure 4.9 The arrangement of threads in a grid for forward projection on the GPU

Each block in the grid consists of 16 × 16 rays and corresponds to a block of threads in CUDA. Each thread first calculates the position of each of the sample points along its ray. The value of the image at the sample point is obtained by using the hardware bilinear interpolation in the texture memory. These values are then summed together to obtain the forward projection values. Each thread then reads the corresponding ray value in the original sinogram(y_i) from the device memory, calculates h_i^n and stores it back in the device memory. Figure 4.10 demonstrates the forward projection implementation.

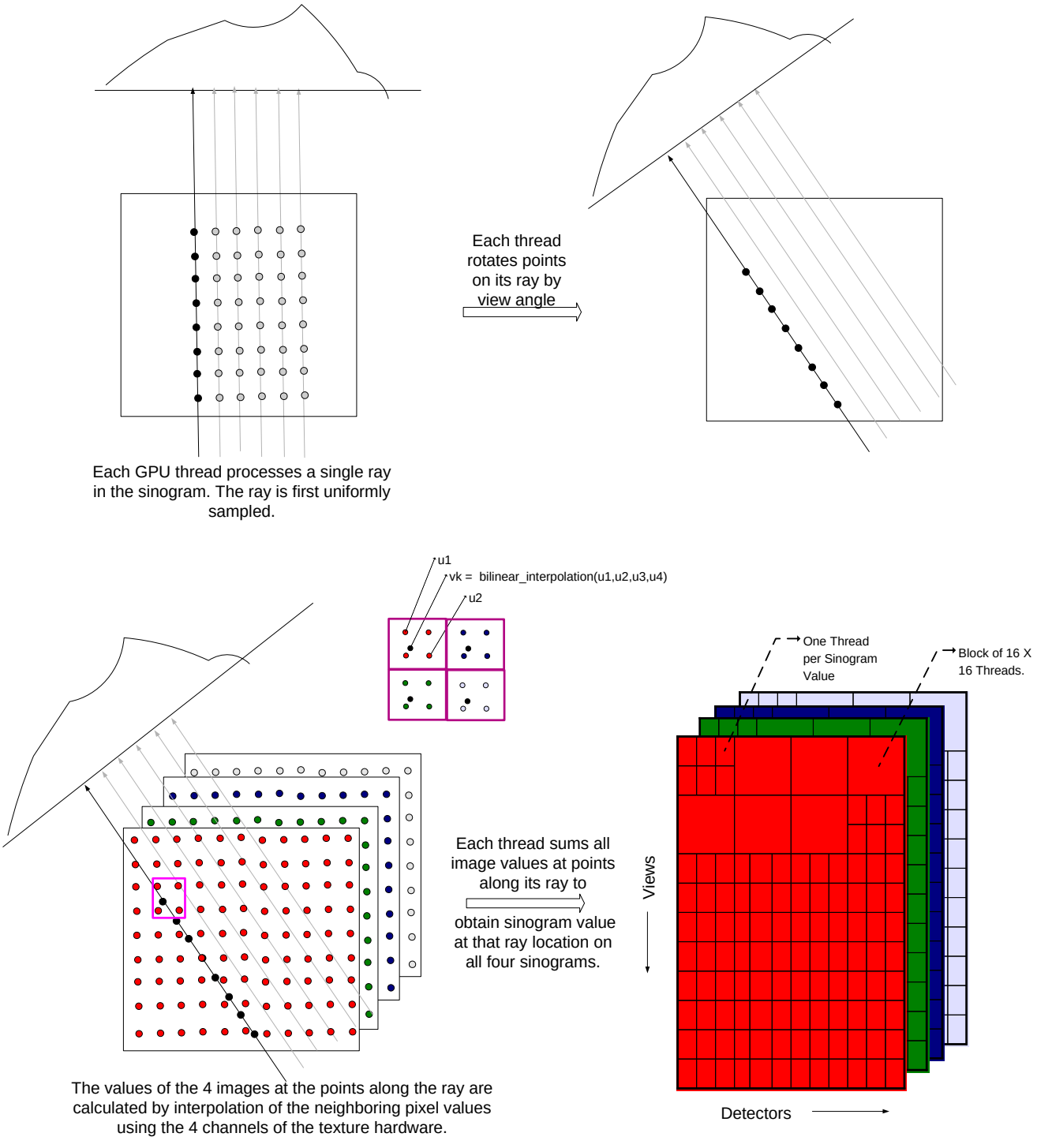

Figure 4.10 The forward projection mechanism on the GPU

4.2.3.2 Back Projection:

For the back projection operation, each of the threads calculates the value of $\sum_{i=1}^{Ny} a_{ij} h_i^n$ (equation 4.1) for one particular pixel, 'j' in all 4 image slices. The values of h_i^n calculated by the forward projection operation are stored in the device memory. For each of the views, it is now required to smudge back the h_is over the entire image. For each pixel, the contribution from each of the views is to be summed to obtain the 'image update' estimate. As explained in section 3.2.4, the back projection operation would require the following steps,

1) Load one row of the sinogram.(Each row corresponds to a view)
2) Back Project (Smudge back) the row onto the entire image.
3) Rotate these values by the view angle 'θ'.
4) Load these rotated values into the texture memory.
5) Estimate the contribution of this view to every pixel by bilinear interpolation.
6) Add the resulting pixel-updates to the pixel updates from the previous views.
7) Repeat steps '1' to '6' for every view in the sinogram.

From the aforesaid procedure, it is clear that the back projection operation is extremely complicated as compared to the forward projection operation. The total number of memory copy and texture memory load operations equal to the number of views in the sinogram.

We used a more sophisticated approach to solve this problem. There is a lot of data redundancy in the back projection operation. One can easily notice that when each

view of the sinogram is back projected (smudged back) onto points that are uniformly distributed along the rays, the value of each row of points equals the value of the original view. Thus, there is plenty of data redundancy.

For views at varying projection angles θ, we can either rotate the view by an angle θ or rotate the image by an angle of '$2\pi - \theta$'. Considering we rotate the image, we can observe the following:

1) The value at any pixel center is obtained by interpolating between the immediately upper and lower rows of back projected values.
2) All the back projected rows have the same value.
3) For correct interpolation, the relative position of the pixel center with respect to the back projected rows immediately above and below the pixel center position is important. The absolute position of the pixel center however is immaterial.

Hence instead of actual back projection, we propose to just replicate the value of each row of the sinogram once. Then, all the pixel centers in the image are moved along the view direction to a new position in between these two rows such that the relative position between the rows and the pixel center remains unchanged. Simple bilinear interpolation now gives the back projected value at every pixel center for this particular view.

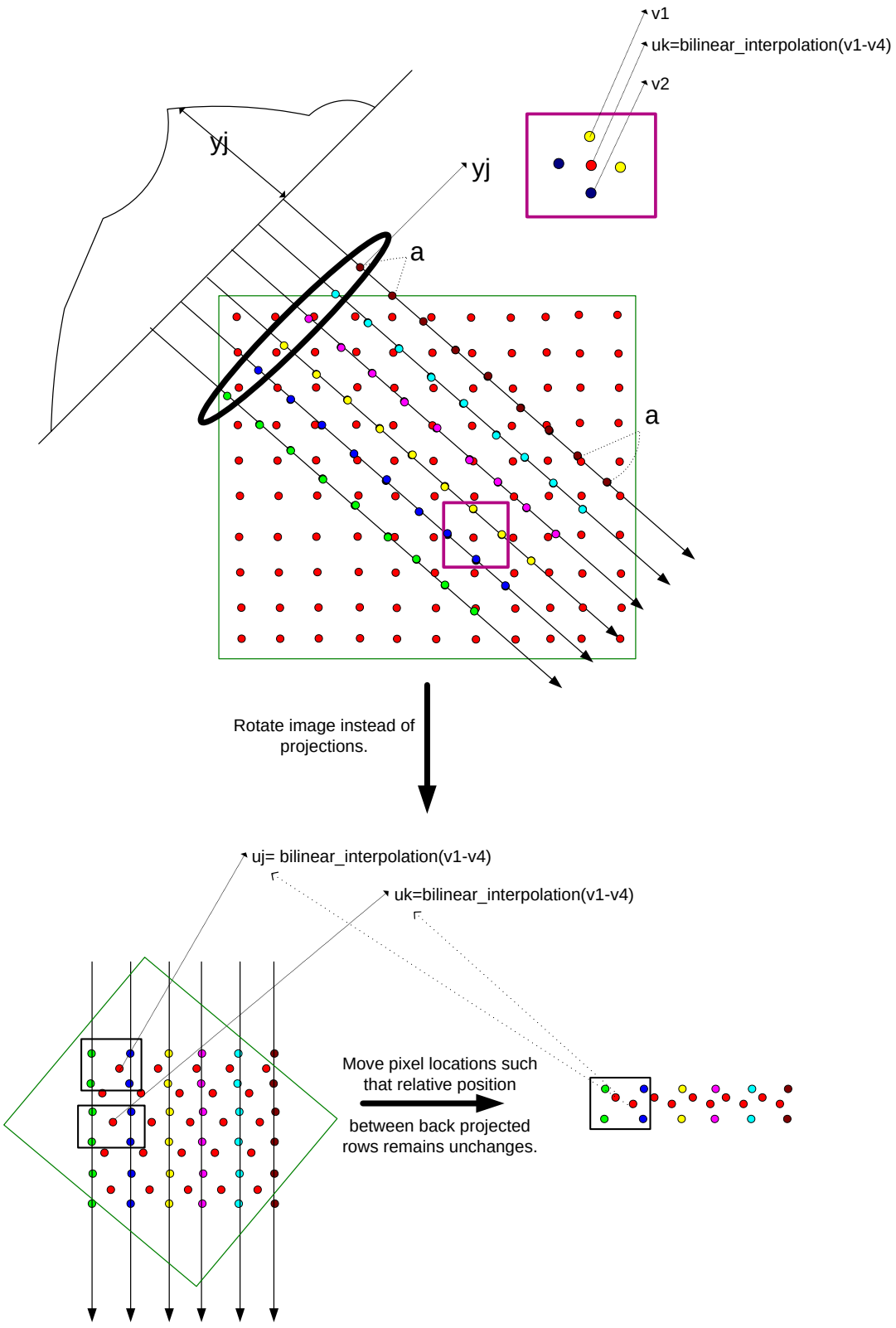

Figure 4.11 The back projection mechanism on the GPU

This method gives us two specific advantages,

1) Only 1 replication of any view is required for back projection. This is equivalent to removal of all redundancy in the back projection operation.
2) The back projected sinogram views are not rotated. Instead the image is rotated in the opposite direction for each back projection.

These improvements translate to considerable savings in terms of memory bandwidth. Suppose a sinogram consists of 'N' views with 'D' detectors, a back projected view will consist of 'D' × 'D' values. 'N' similar views are typically required to complete the back projection operation. However, by replicating each row once, we only need 'N' × 'D' × 2 values to complete the entire back projection. Thus, the total memory required is only '2/D' times the original memory. The algorithm is now implemented as follows.

1) Replicate each row of the sinogram (The new sinogram is called the 'Extended Sinogram').
2) Load the extended sinogram into the texture memory.
3) For each view θ, rotate the image pixel centers by '2π-θ'.
4) Move each of the pixel centers along the view direction (vertically) such that they lie within the two rows corresponding to that view of the extended sinogram.
5) Estimate the contribution of this view to every pixel by bilinear interpolation.
6) Add the resulting pixel-updates to the pixel updates from the previous views.
7) Repeat steps '3' to '6' for every view in the sinogram.

The updated algorithm now reduces the number of memory copy operations for back projection of one view of the sinogram from 'Nd-1' to '1', where 'Nd' is the number of detectors. It also reduces the number of texture memory loads from 'Θ' to '1', where 'Θ' is the total number of views. Figure 4.11 depicts the algorithm.

For implementation of the algorithm, the image was divided into 16 × 16 blocks. Each CUDA thread block corresponded with a block in the image. Figure 4.12 gives the grid scheme for the back projection operation. Since 4 sinograms for each of the 4 slices were transferred to the texture memory, the back projection operation gave the image updates to all 4 image slices in the GPU. The bilinear interpolations for the 4 slices were again performed using the 4 channels of the texture memory.

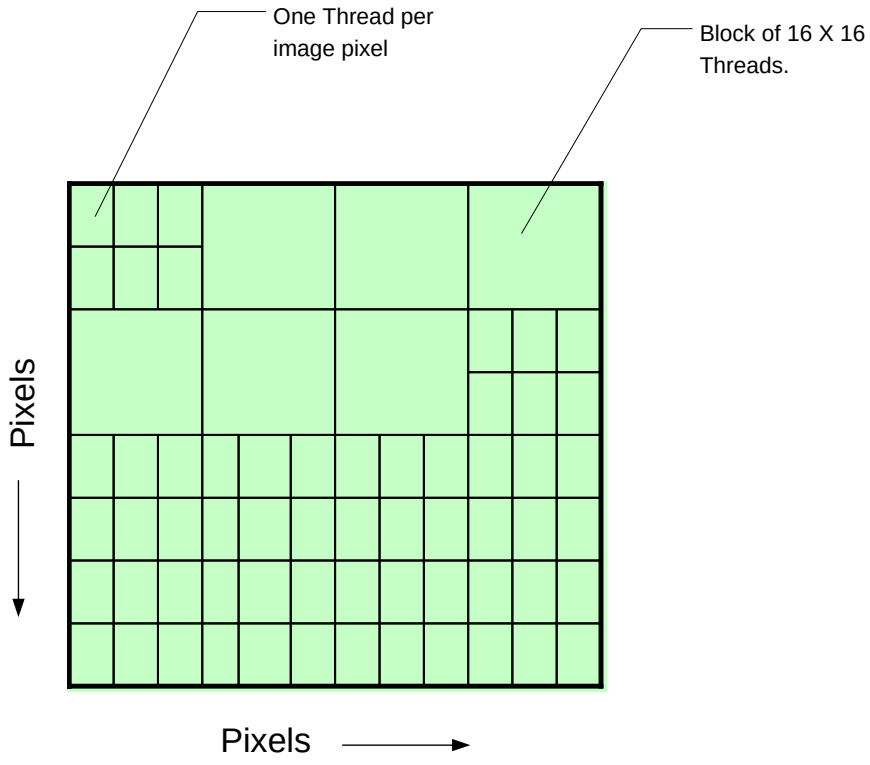

Figure 4.12 The arrangement of threads in a grid for back projection on the GPU

4.2.3.3 Penalty Function and Image update:

The image slice was again divided into 16 × 16 blocks and each thread in a CUDA block processed a pixel of the image block. Each thread computed the penalty value for that particular pixel. The back projected values were available in the device memory. The threads used these values along with the pre-computed denominator values to update the image as required in equation 4.1 and thus complete the iteration.

4.3 Termination condition

For any iterative algorithm to be used in practice, it is important for the algorithm to be monotonic and converge to a unique solution. However, it is also important to have a well defined termination condition to ensure that enough number of iterations is completed and unnecessary iterations are avoided. The termination condition also should not be computationally expensive since it does not contribute directly to the reconstruction process.

We compare the image update after every iteration, to the image value at the end of the previous iteration to check the rate of image update. Since CT value of air is -1000 and that of bone is 1000; thereby giving a minimal resolution of 1/2000=0.0005; we decide to terminate iterations when the average image pixel update is less than 0.05% of the original value. Hence, if the rate of change of the image is less than 0.05%, then we know that the image has converged and any future iteration will not considerable benefit the quality of the image. The termination condition may therefore be defined as follows:

$$\frac{\sum_j abs(\mu_j^{n+1} - \mu_j^n)}{\sum_j \mu_j^n} 100 < 0.05 \tag{4.3}$$

where μ_j^n is the value of the j^{th} pixel after the n^{th} iteration.

We can note here that the subtraction is not required during the actual reconstruction as μ_j^{n+1} is calculated as,

$$\mu_j^{n+1} = \mu_j^n + \mu_{update} \tag{4.4}$$

Hence the computation of the termination condition only involves two reduction operations and one division operation.

For the ordered subsets version of the algorithm, it is important to note that the image converges faster per iteration and the rate of change is accelerated by a factor almost equal to the number of subsets per iteration. Hence we still maintain the 0.05% rate of change in image for the termination condition, but we ensure that this rate of change holds for every subset in the iteration. So the modified termination condition becomes,

$$\frac{\sum_j \mu_j^{n+1} - \mu_j^n}{M \sum_j \mu_j^n} 100 < 0.05 \tag{4.5}$$

where M represents the number of subsets per iteration. The other symbols are as described in equation (4.3). Here we ensure that the average rate of change per subset

over all the subsets of the iteration is less than 0.05%. This termination condition is calculated only once per iteration after all the subsets are completed. This ensures that the algorithm does not terminate mid-way through an iteration.

Chapter 5: Results and Conclusions

5.1 Reconstructed Image Quality for various hardware platforms

We have introduced two hardware platforms for acceleration in this thesis. One is the multi-processor cluster and the other is the GPU. The ray-tracing based implementation of the PS algorithm was mapped to both the multi-processor cluster as well as the GPU as explained in sections 4.1 and 4.2. Projections were created from a scanner image to obtain 672 × 580 sinograms. The sinograms thus obtained were reconstructed on both the platforms.

The figure 5.1 below compares the quality of the reconstructed images after each iteration of the PS algorithm using the PSNR as a metric. While the blue continuous curve represents the single processor implementation, the pink broken curve represents the multi-processor implementation. The red curve gives the GPU based implementation. We can clearly see that the multi-processor implementation gives exactly the same results as the single processor implementation. This is expected as all the nodes in the cluster have the same specifications and every processor is exactly the same. Further, there is no modification of the algorithm itself for parallelization across the nodes. Interestingly the GPU, despite its single precision accuracy, gives a slightly higher PSNR for the reconstruction images as compared to the general purpose computer based implementations. However, since PSNR alone is not a very reliable metric of actual image quality, we also use the Q index to compare the images.

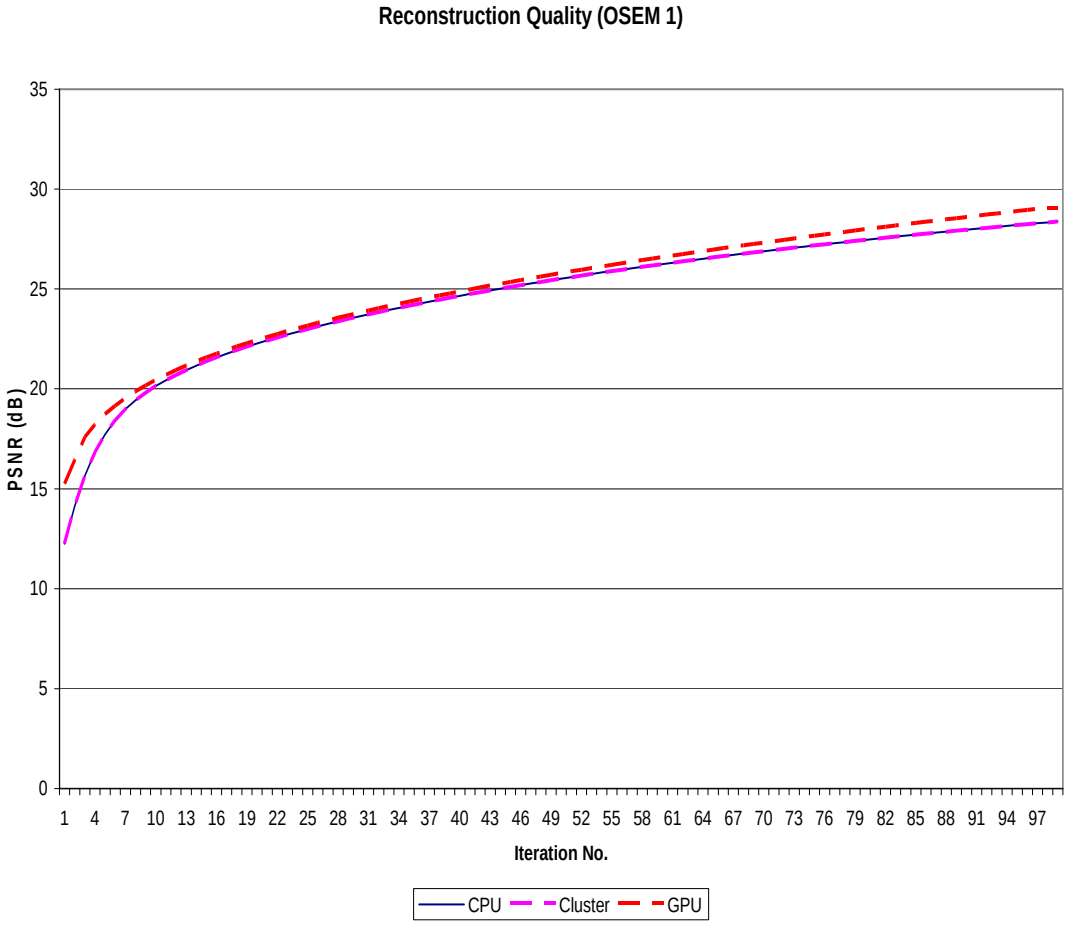

Figure 5.1 Quantitative comparison of the images reconstructed using the CPU, Cluster and the GPU with sinograms of 672 × 580 using PSNR as the metric

Figure 5.2 compares the quality of the reconstructed images using the Q index as the metric. Here again it is obvious that the GPU and the cluster based solutions are comparable to, if not better than, the single processor implementations.

We can notice that the single subset per iteration implementation of the PS algorithm, though monotonically increasing is slow in convergence. To enable faster convergence, multiple subsets were used per iteration as explained in [5]. Figures 5.3 and 5.4 compare the quality of the reconstructed images when 10 Ordered Subsets are

used per iteration. We again use both the PSNR as well as the Q index as the metrics to compare the quality of the reconstructed images.

Figure 5.2 Quantitative comparison of the images reconstructed using the CPU, Cluster and the GPU with sinograms of 672 × 580 using Qindex as the metric

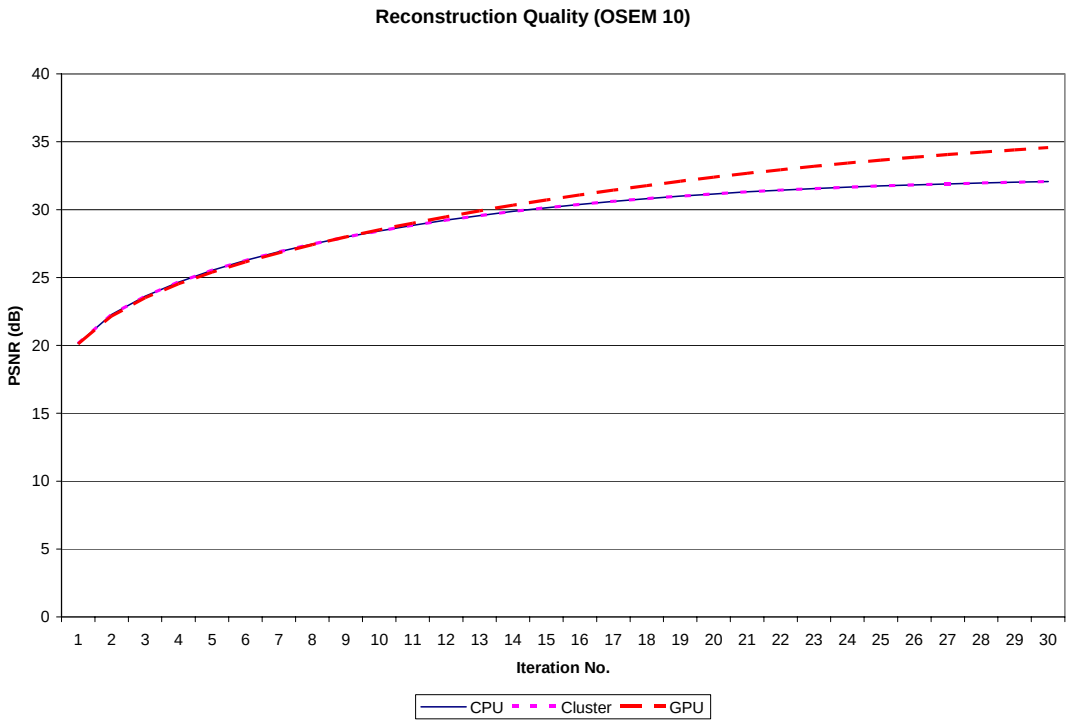

Figure 5.3 Quantitative comparison of the images reconstructed using the CPU, Cluster and the GPU using PSNR as the metric and 10 Ordered Subsets

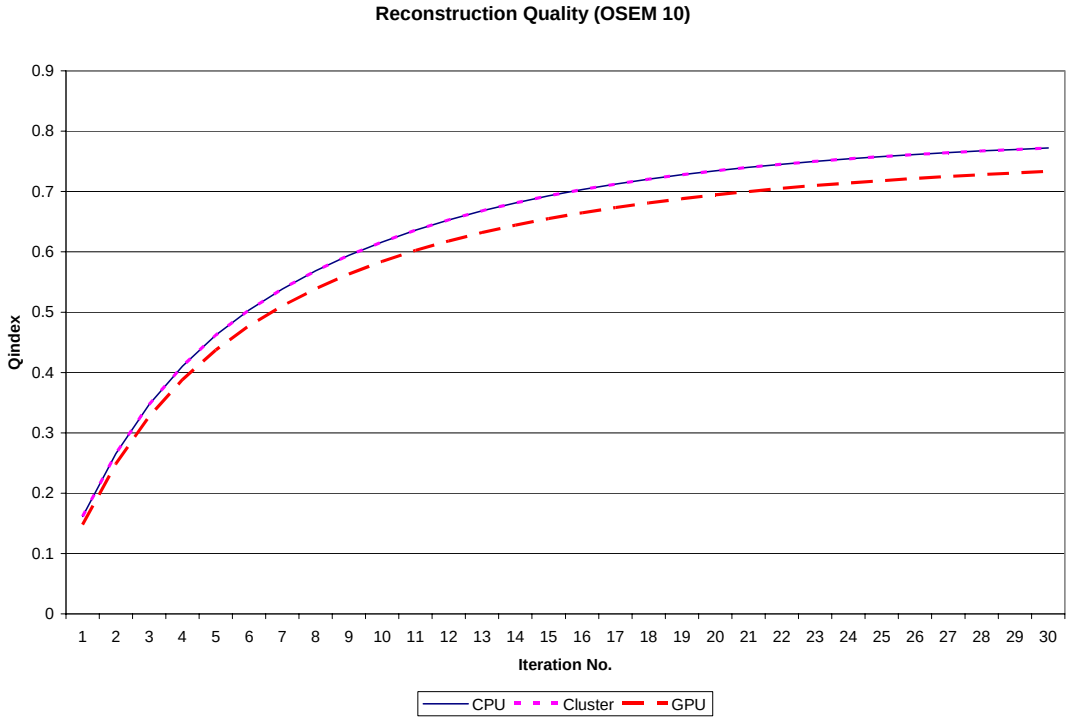

Figure 5.4 Quantitative comparison of the images reconstructed using the CPU, Cluster and the GPU using Qindex as the metric and 10 Ordered Subsets

It is clear that the quality of the images reconstructed using the cluster is the same as the single CPU. However, the PSNR metric suggests a slightly improved image quality for GPU-based reconstruction, while the Q index suggests a slight decrease in image quality. However, the reconstructed image quality does not vary much from the single CPU implementation.

For qualitative evaluation, figures 5.5 and 5.6 display the resulting images for the cluster and the GPU based reconstruction after 100 iterations of the OSEM1 algorithm and 30 iterations of the OSEM10 algorithm when the images have stabilized. The original CPU based reconstruction results are exactly the same as the cluster based results and are hence not displayed.

From the images we can clearly notice that the GPU-based reconstruction converges to the same solution as the Cluster and the software based approaches. A slight difference in the sharpness of the reconstructed images gives the slight difference in the resulting PSNR. This can be corrected by running a few more iterations of the algorithm or using a simple edge preserving penalty function as described in [5][12][14] etc.

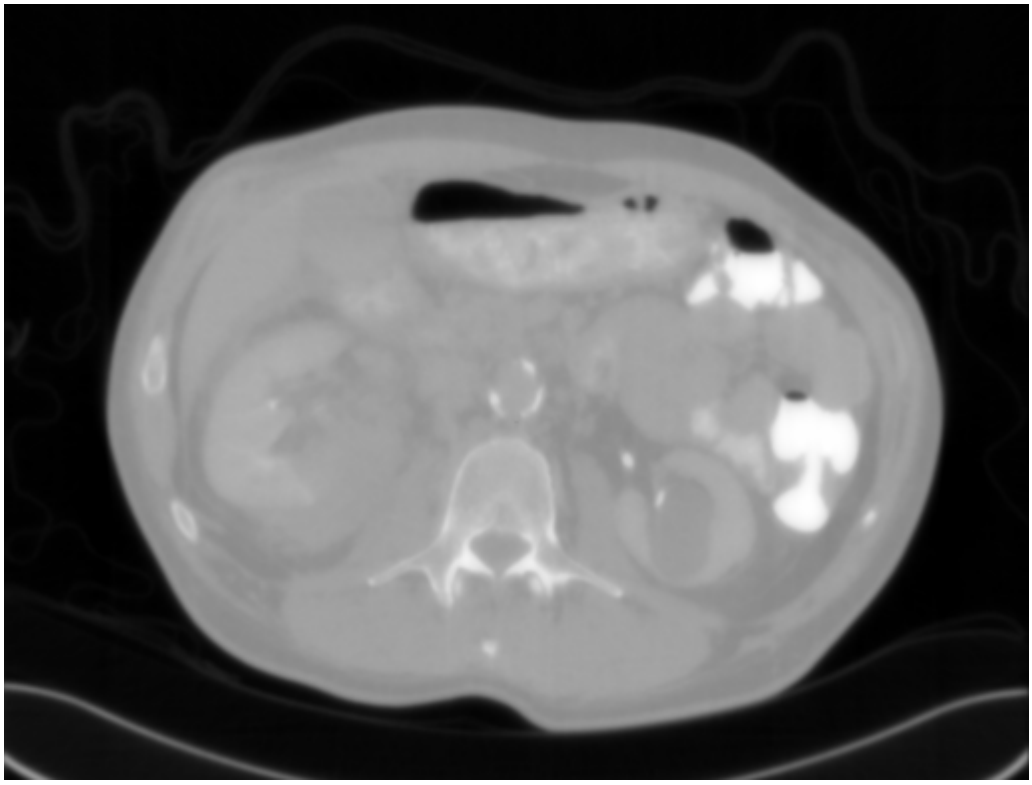

Figure 5.5 Image reconstructed after 100 iterations of PS (OS-1) on CPU/Cluster

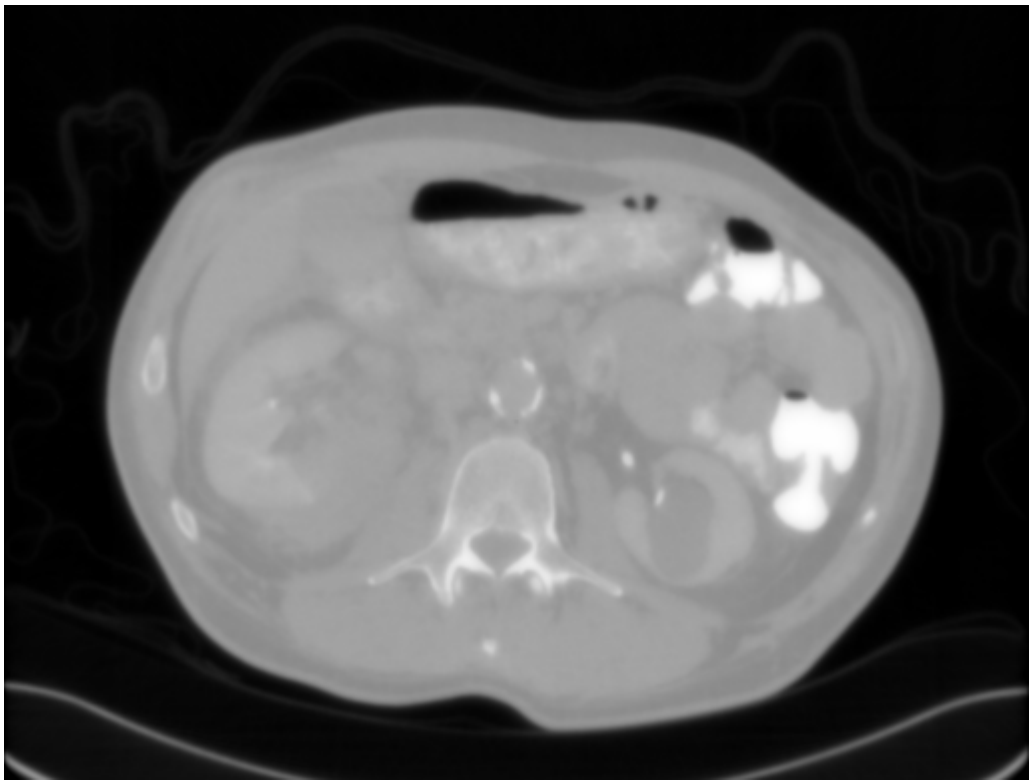

Figure 5.6 Image reconstructed after 100 iterations of PS (OS-1) on GPU

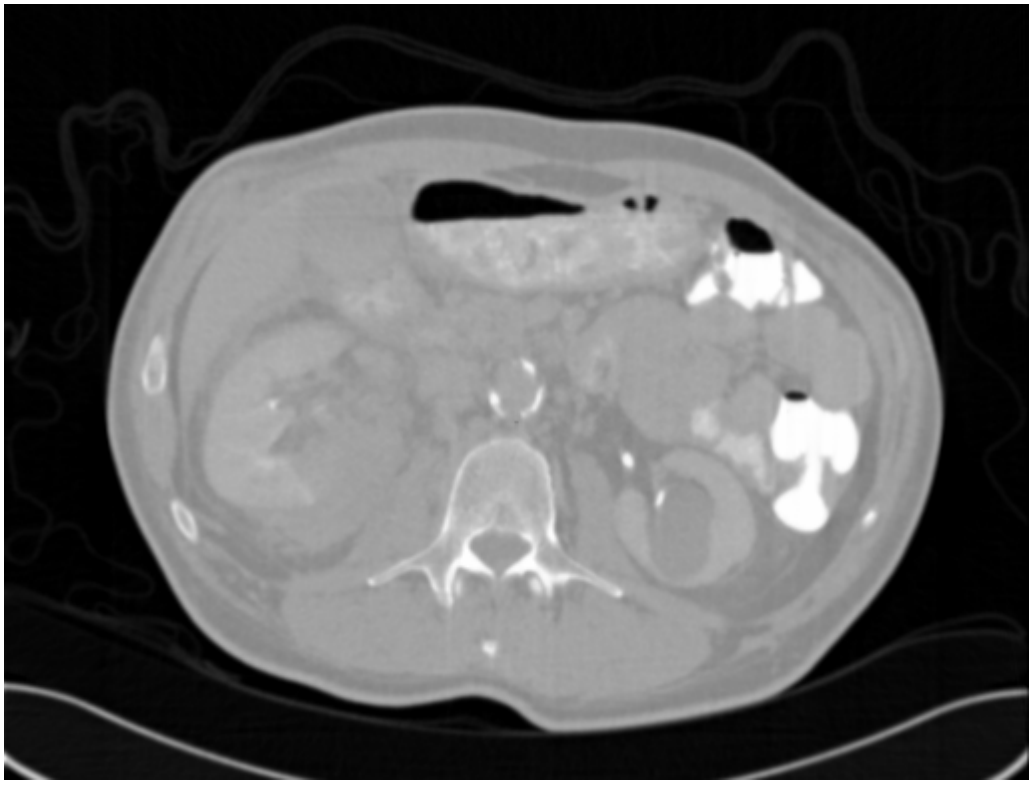

Figure 5.7 Image reconstructed after 30 iterations of PS (OS-10) on CPU/Cluster

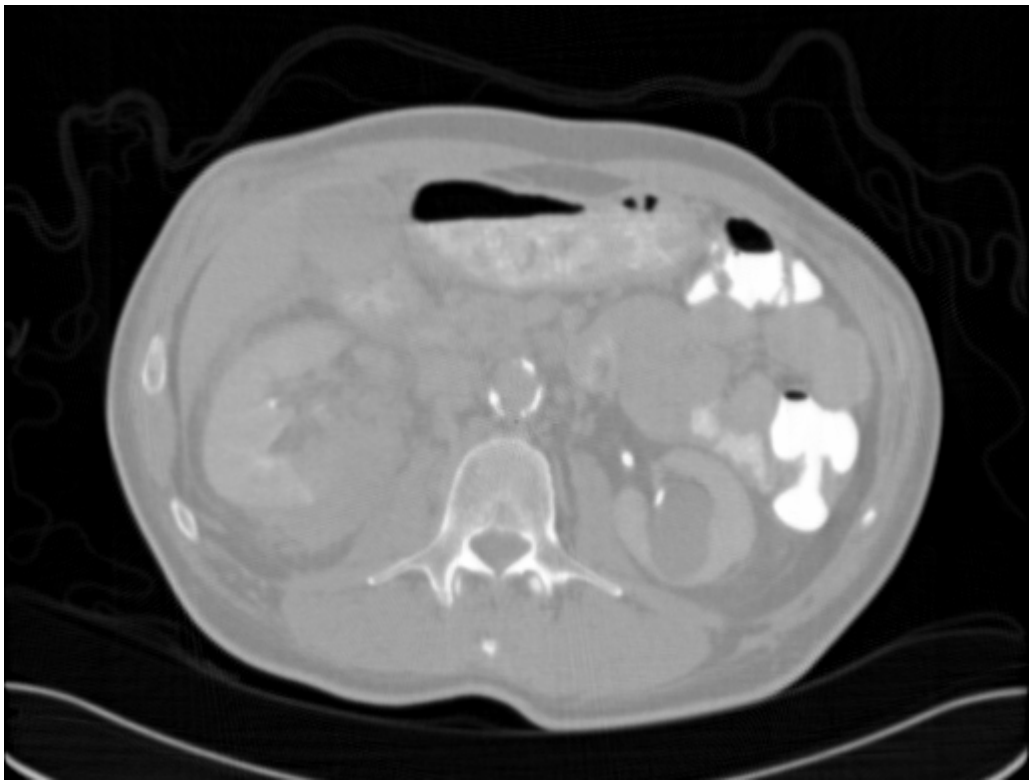

Figure 5.8 Image reconstructed after 30 iterations of PS (OS-10) on GPU

Figure 5.9 Difference image between 100 iterations of PS (OS-1) on CPU/Cluster and GPU

Figure 5.10 Difference image between 30 iterations of PS (OS-10) on CPU/Cluster and GPU

5.2 Speed-up for hardware based solutions

It is clear from section 5.1 that both the GPU as well as the cluster based solutions provide good quality reconstructed images for transmission CT. We compared the reconstruction time for each of the techniques as follows. The CPU, cluster and GPU based implementations were used to run 100 iterations of the PS algorithm on sinograms of various sizes. In each case, the total time taken for execution (except the time taken to load the sinogram) was recorded. The time included the time taken for inter-processor communication as well as the time to write the images to disk. The time taken to load the sinograms always remained constant at about 1 second.

The time per iteration was calculated as the Total time/ No. of iterations. Table 5.1 gives the time per iteration, the speedup in comparison to the CPU implementation with one sub-set per iteration and the total throughput. It is clear that the larger the sinogram and the reconstructed image, the better the speed-ups achieved. It can also be seen that the total throughput achieved is 4 times the speedup. This is because the GPU is capable of processing 4 slices at any given time utilizing the 4 channels of the texture memory. Even at extremely large sinogram sizes of 1498 × 580 and reconstruction image size of 1024 × 1024 pixels, the GPU gives a throughput of 9 iterations per second. For normal reconstructed image sizes of 512 × 512, the GPU gives a throughput of about 37 iterations per second.

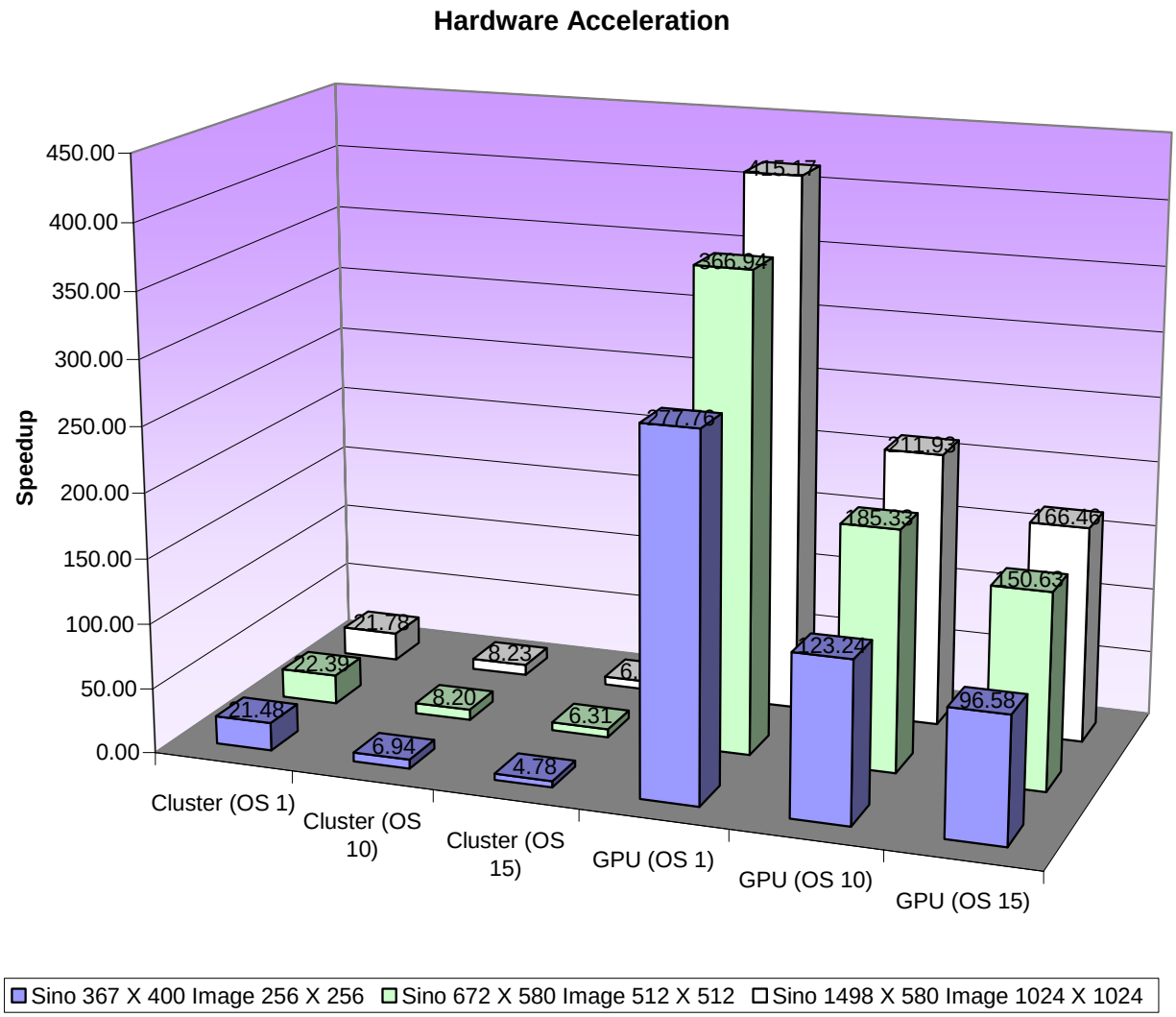

Figure 5.11 Hardware acceleration achieved for various reconstruction geometries using the Cluster and the GPU

We can notice that as the number of subsets per iteration increases, the speed-up achieved reduces drastically in the case of the Cluster. For 1 subset per iteration, the Cluster of 32 processors manages to give a speedup of about 22X over the single

Sinogram Size (Detectors x Views)	*Reconstructed Image Size*	*Platform*	*No of slices reconstructed*	*Subsets /Iteration*	*Time/ Iteration (s)*	*Speedup*
367 X 400	256 X 256	CPU	1	1	6.68	1.00
		CPU		10	6.68	1.00
		CPU		15	6.68	1.00
		Cluster(32)	1	1	0.31	21.48
		Cluster(32)		10	0.96	6.94
		Cluster(32)		15	1.40	4.78
		GPU	4	1	0.02	277.76
		GPU		10	0.05	123.24
		GPU		15	0.07	96.58
672 X 580	512 X 512	CPU	1	1	39.06	1.00
		CPU		10	39.08	1.00
		CPU		15	39.08	1.00
		Cluster(32)	1	1	1.74	22.39
		Cluster(32)		10	4.76	8.20
		Cluster(32)		15	6.19	6.31
		GPU	4	1	0.11	366.94
		GPU		10	0.21	185.33
		GPU		15	0.26	150.63
1498 X 580	1024 X 1024	CPU	1	1	184.38	1.00
		CPU		10	196.01	0.94
		CPU		15	193.30	0.95
		Cluster(32)	1	1	8.47	21.78
		Cluster(32)		10	22.40	8.23
		Cluster(32)		15	28.64	6.44
		GPU	4	1	0.44	415.17
		GPU		10	0.87	211.93
		GPU		15	1.11	166.46

Table 5.1 Hardware acceleration (Speedup and throughput) achieved for various reconstruction geometries using the Cluster and the GPU

CPU implementation. However, as the number of subsets per iteration increase, the contribution of the accelerated forward and back projection operations decreases as compared to the inter-processor communication. Therefore we can see a large hit in the speed-ups achieved at higher orders of subsets. It is hence clear that the amount of

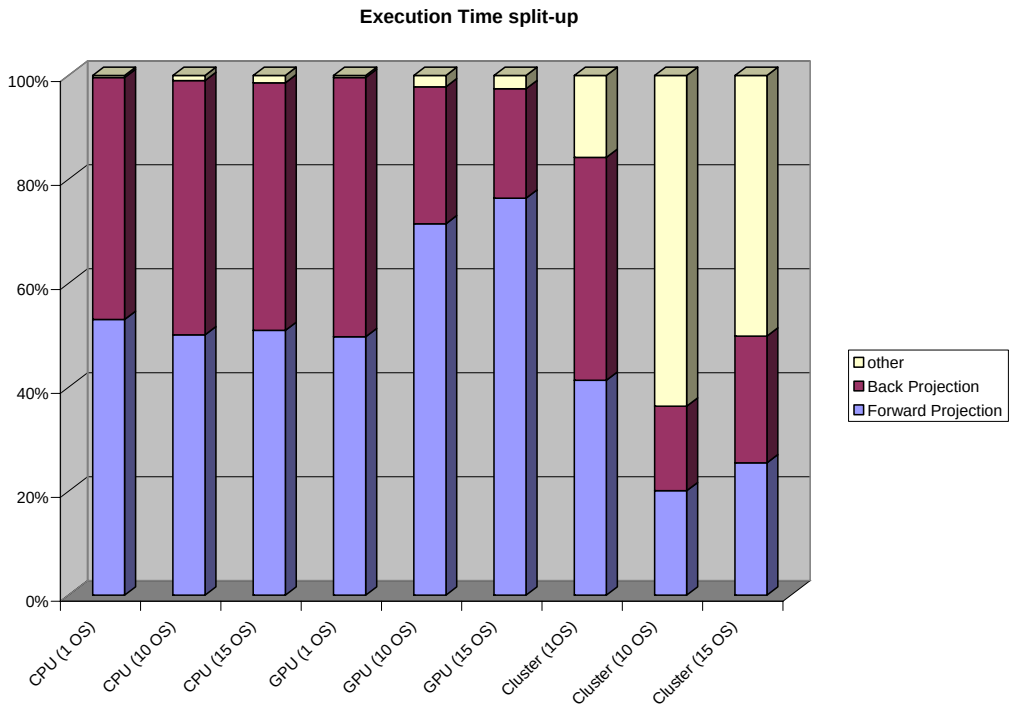

Figure 5.12 Split-up of execution time for various reconstruction geometries on the CPU, Cluster and GPU

acceleration that can be achieved by using groups of processors in parallel is severely limited by the inter-processor communication latencies. Figure 5.12 displays the percentage of total time spent in each of the operations for various numbers of Ordered Subsets.

However, the reduction in speed-ups is not so drastic in the case of the GPU. This is due to two main reasons:

1) Unlike the cluster based implementation, the GPU implementation parallelizes all portions of the algorithm including the image update and penalty function calculation.

2) The inter-thread communication in the GPU is achieved simply via a global read from the device memory. This is much faster than the inter-processor communication over external networks in the case of the cluster.

Yet another observation is that though the time taken for forward projection increases with increase in the number of subsets in the GPU, the percent of time taken for the back-projection operation actually decreases. This is because the back-projection operation is implemented using the concept of the 'extended sinogram' as explained in section 4.2.3.2. The bandwidth and the computation necessary for this implementation are directly proportional to the number of views used for back-projection. This lack of additional overhead results in the total time for back-projection to remain the same in spite of varying number of subsets per iteration.

Table 5.2 gives the exact time taken for the various operations across the platforms and using varying number of subsets per iteration.

		Total	Forward Projection (s)	Forward Projection (%)	Back Projection (s)	Back Projection (%)	other (s)	other (%)
CPU	OSEM1	184.38	97.81	53.05	85.82	46.55	0.75	0.41
	OSEM10	196.00	98.19	50.10	95.84	48.90	1.97	1.01
	OSEM15	193.30	98.45	50.93	92.12	47.66	2.73	1.41
GPU	OSEM1	0.44	0.22	49.72	0.22	49.85	0.00	0.44
	OSEM10	0.87	0.62	71.46	0.23	26.34	0.02	2.20
	OSEM15	1.11	0.85	76.36	0.23	21.05	0.03	2.59
Cluster	OSEM1	8.47	3.50	41.35	3.63	42.87	1.34	15.78
	OSEM10	22.41	4.50	20.09	3.65	16.28	14.26	63.63
	OSEM15	28.64	7.29	25.45	6.98	24.39	14.37	50.16

Table 5.2 Distribution of time across various operations for reconstruction of PS algorithm on various platforms

Thus it is clear that the ray-tracing algorithm implemented on the GPU using the 'extended sinogram' is a inexpensive and an excellent platform for acceleration of iterative algorithms for CT reconstruction.

Figure 5.13 gives a comparison of the speed-ups achieved by various groups using GPU's. The speed-ups are all for images reconstructed at 256×256×256, reconstructed using different algorithms and different GPU hardware.

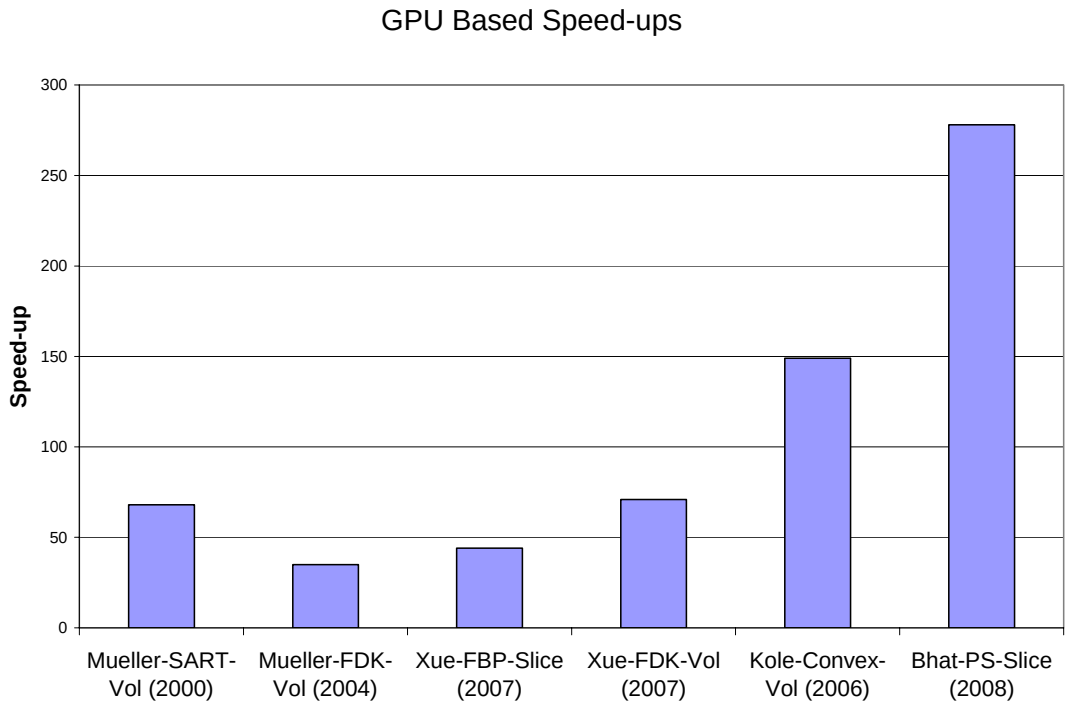

Figure 5.13 Comparison of speed-ups achieved using various algorithms on GPUs.

5.3 Speed-up with variation in number of ordered subsets

From the results above, it is clear that the GPU is an excellent platform for acceleration of the PS algorithm. However, it can be noticed that the images converge faster with the increase in the number of subsets per iteration. It is obvious that

arbitrarily large number of subsets cannot be used to get improved quality as the number of projections per subset decreases with the increase in the number of subsets.

Figure 5.14 demonstrates the variation of the reconstructed image PSNR after just 1 iteration with varying number of subsets for a sinogram with 672 detectors and 580 view angles. It is clear that quality of the reconstructed images improves monotonically till about 8 views/subset or 70 subsets per iteration. Beyond that, the quality of the reconstructed image after 1 iteration is not monotonic and varies widely depending on the distribution of the subsets.

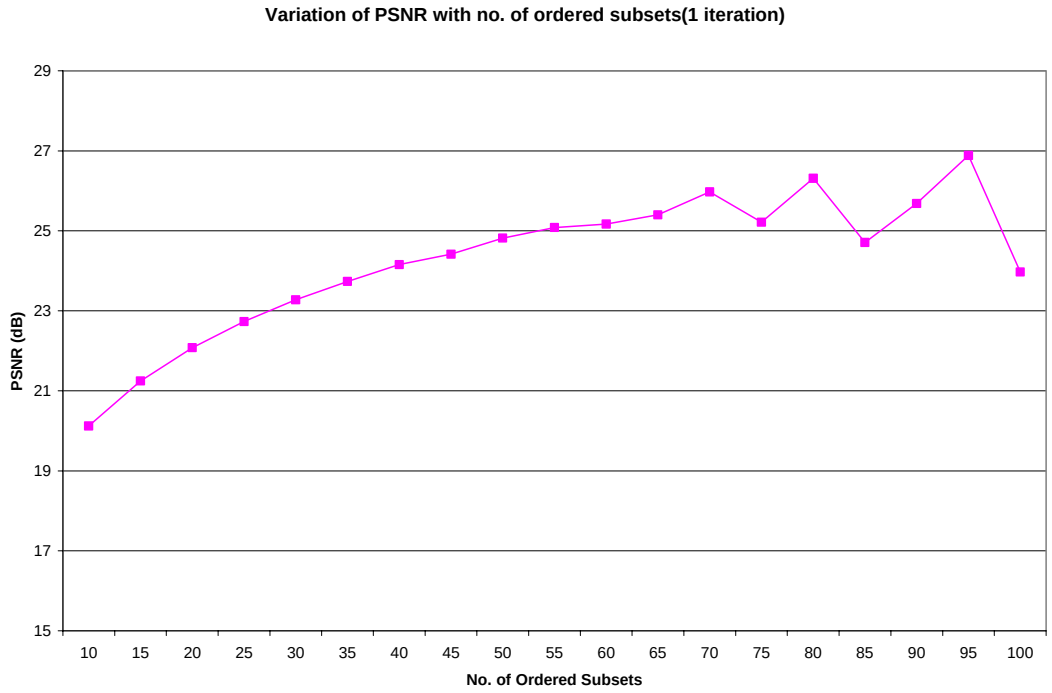

Figure 5.14 Variation of PSNR with the number of subsets after 1 iteration of the PS algorithm using 672 × 580 sinogram.

Figure 5.15 demonstrates the time for reconstruction of a single iteration of the image at a 512 × 512 resolution with varying subsets. The reconstruction time linearly increases with the increase in the number of subsets. Thus, increasing the number of ordered subsets does not give any benefit beyond a certain point.

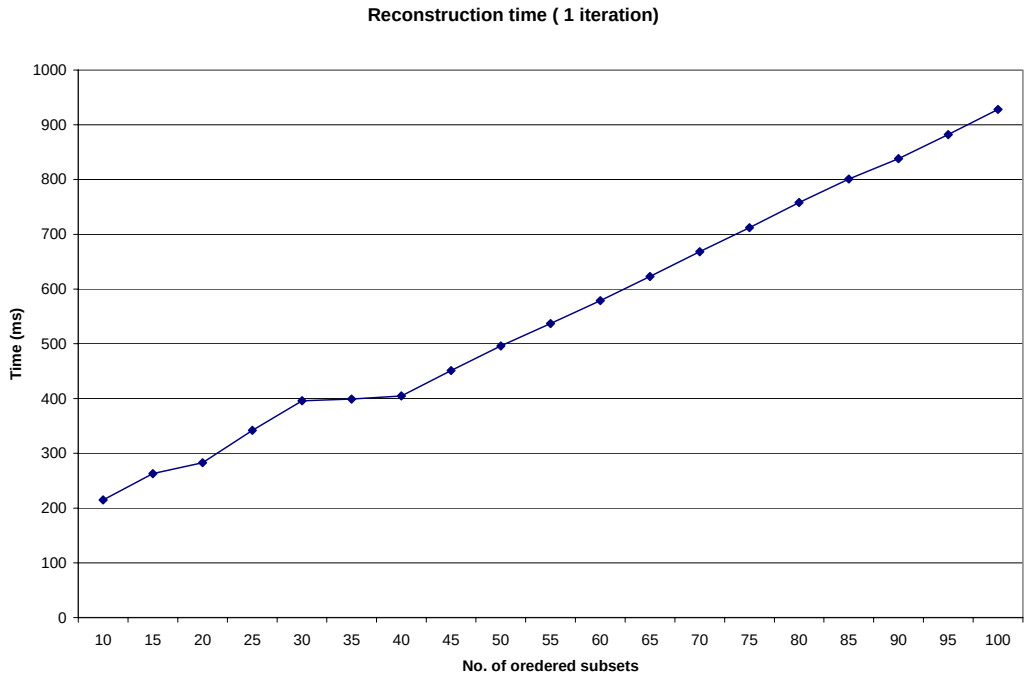

Figure 5.15 Variation of reconstruction time per iteration with the number of ordered subsets for the PS algorithm on the NVIDIA 8800GTX GPU

Table 5.3 gives the time for reconstruction as well as the PSNR for a 512 × 512 image reconstructed from the 672 × 580 sinogram. It is clear that with about 50-70 subsets per iteration, reasonable quality images can be obtained after just a single iteration on the GPU.

No. of subsets	Reconstruction Time/iteration - 4 slices (ms)	Reconstruction Time/iteration - 64 slices (s)	PSNR after 1 iteration (dB)
10	215	3.44	20.12
15	263	4.21	21.25
20	283	4.53	22.08
25	342	5.47	22.73
30	396	6.34	23.27
35	399	6.38	23.73
40	405	6.48	24.16
45	451	7.22	24.42
50	496	7.94	24.82
55	537	8.59	25.08
60	579	9.26	25.17
65	623	9.97	25.35
70	668	10.69	25.97
75	712	11.39	25.22
80	758	12.13	26.31
85	801	12.82	24.71
90	838	13.41	25.68
95	882	14.11	26.89
100	928	14.85	23.97

Table 5.3 Reconstruction time and image quality after 1 iteration on 672 × 580 sinogram with image resolution of 512 × 512 pixels

Figure 5.16 shows the image after a single iteration of the PS algorithm using 70 subsets. It is clear that images of reasonable quality can be obtained using a high number of subsets per iteration after just a single iteration of the PS algorithm.

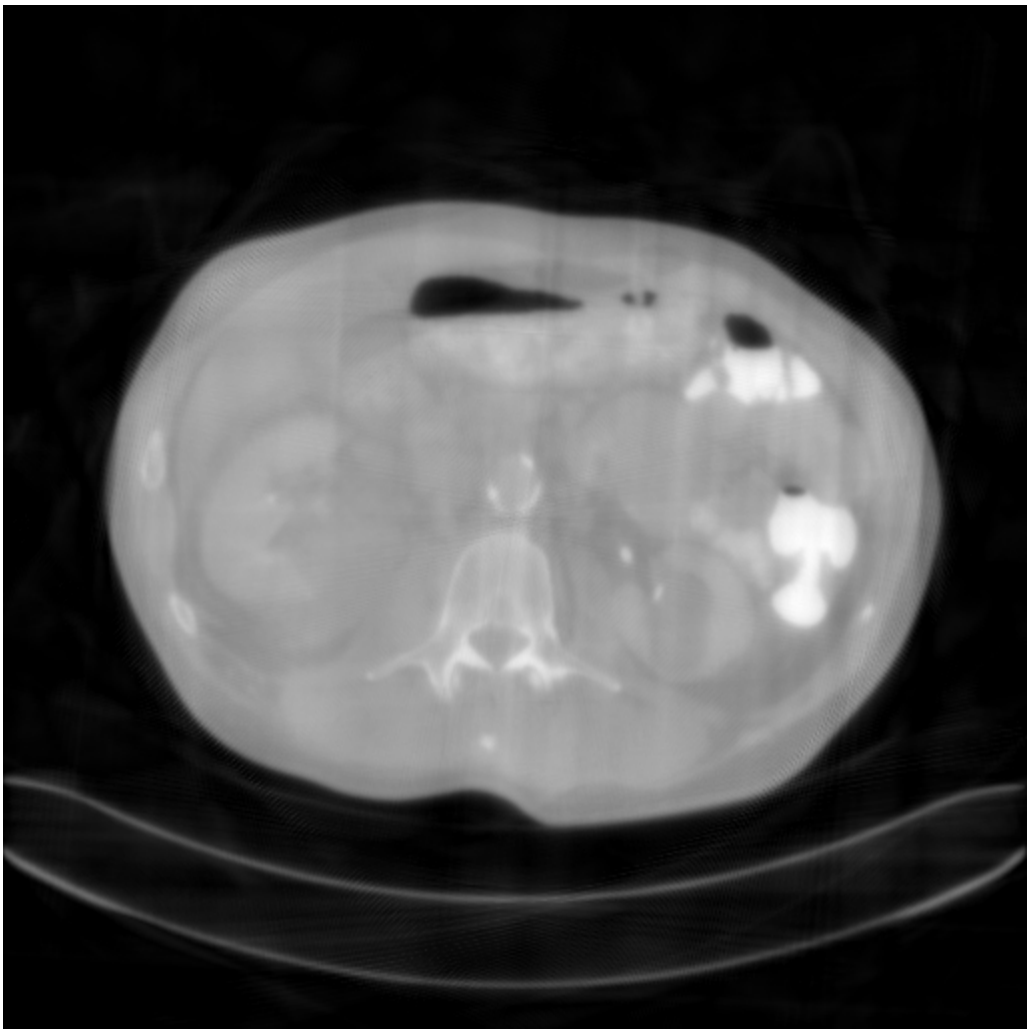

Figure 5.16 Reconstructed image after 1 iteration of PS (OS-70) on GPU using a sinogram of 672 detectors and 580 views.

5.4 Termination condition

To verify the effectiveness of the termination condition proposed in section 4.3, we create sinograms from a known image. The proposed implementation of the PS algorithm is them run on these sinograms for about 75 iterations using various numbers of subsets per iteration. The PSNR of the images reconstructed after every iteration is then calculated using the known image as the benchmark. The number of iterations required for reconstruction as proposed by our termination condition in equations 4.3 and 4.5 is then computed. The condition is then checked against the

PSNR to ensure that the images have converged as predicted by the termination condition.

Figures 5.17 and 5.18 give the termination condition as predicted by our method along with the PSNR curve. We can see that the termination condition correctly predicts the number of iterations required to ensure that the image has converged to a stable solution.

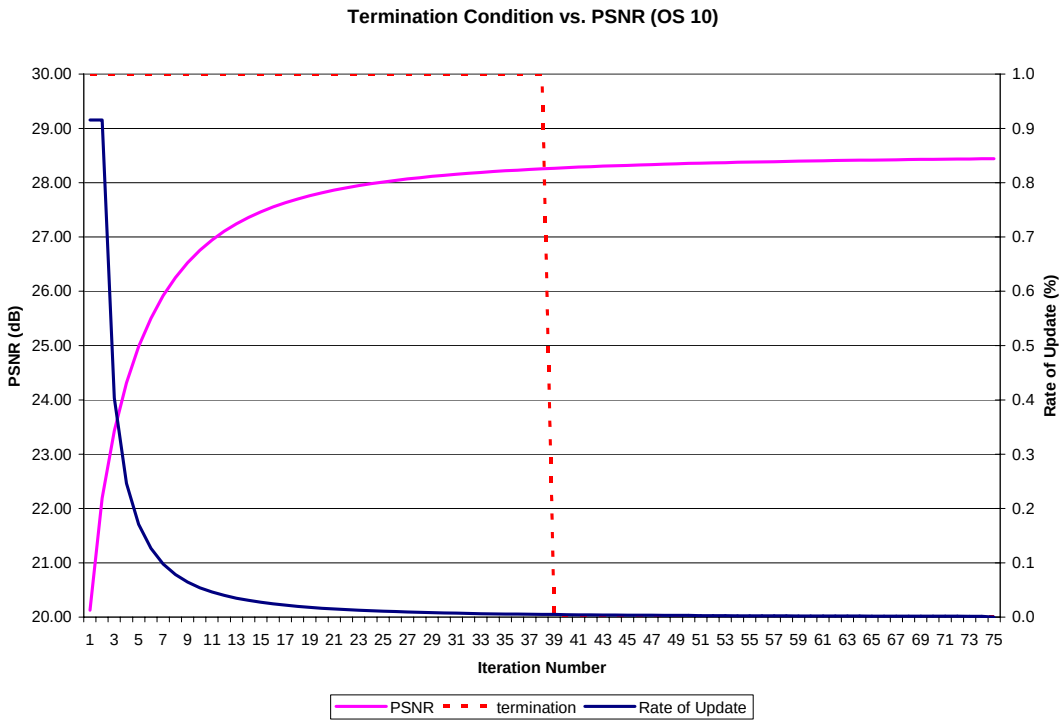

Figure 5.17 Plot displaying the termination condition, rate of image update and the PSNR for an image reconstructed using a sinogram of 672×580 and reconstructed at 512×512.

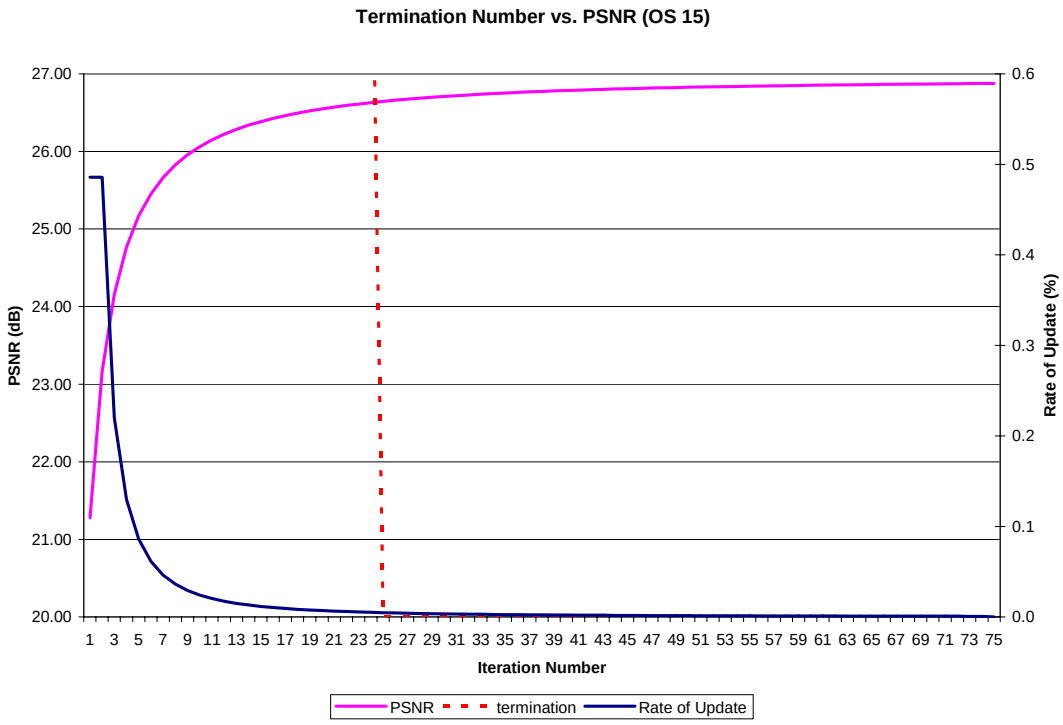

Figure 5.18 Plot displaying the termination condition, rate of image update and the PSNR for an image reconstructed using a sinogram of 672×580 and reconstructed at 512×512.

Sinogram	*Image*	*Subsets/Iteration*	*Iterations to converge*	*Time per slice (s)*
1498 x 580	1024 x 1024	1	170	12.34
		10	18	2.99
		15	12	2.89
		20	9	2.00
		25	***7***	***1.89***
		30	6	1.90
		35	6	2.15
		50	4	2.02
1498 x 580	512 x 512	10	23	1.44
		15	16	1.44
		20	12	1.04
		25	10	1.05
		30	***8***	***1.00***
		35	7	1.01
		40	6	1.00
		45	6	1.07

Table 5.4 Reconstruction time for 1024x1024 image slice using the proposed termination condition on the GPU.

The table 5.4 gives the time taken for reconstruction using the termination conditions described in equations (4.3)-(4.5) and real sinogram data from the scanner. It must be noted that the time indicated includes the time taken to load the sinogram as well as store the image. From the table it is clear that as the number of ordered subsets increases per iteration, the time per iteration also increases. However, the number of iterations required for reconstruction decreases. From the table it is also clear that using 25 subsets per iteration gives the most efficient reconstruction time of about 1.9 sec per slice for a 1024 × 1024 image. Similarly 8 iterations of OS-30 give an optimal trade-off for the image reconstructed at 256 × 256. Figure 5.19 pictorially depicts the optimal reconstruction configuration.

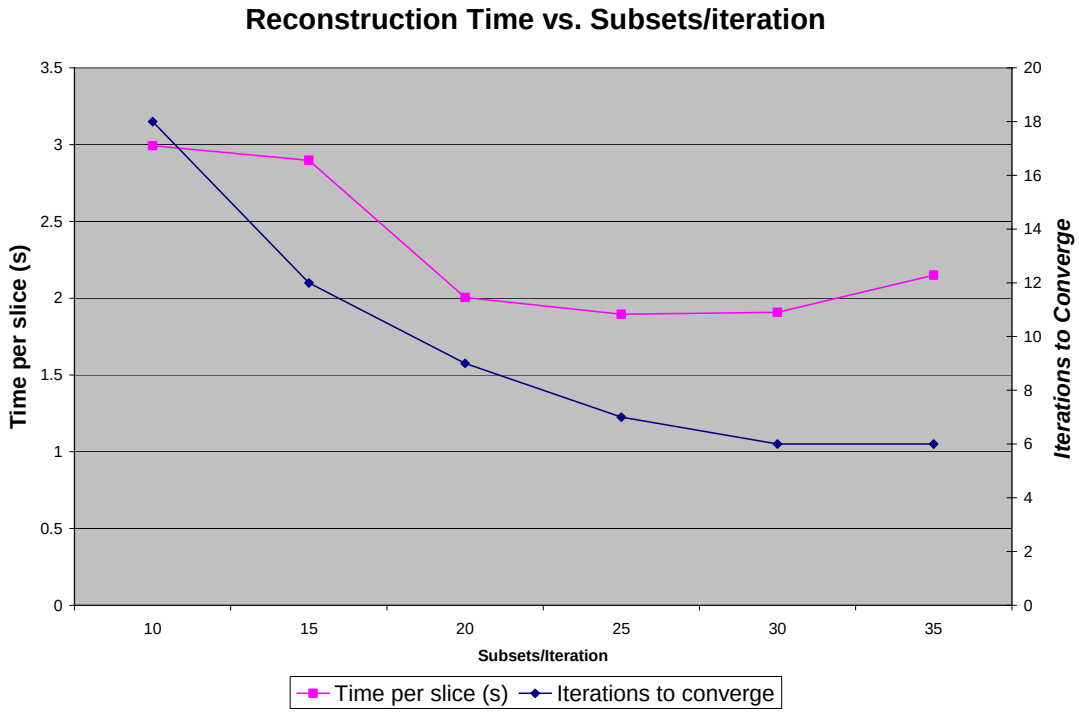

Figure 5.19 Optimal number of ordered subsets for reconstruction of 1024 × 1024 image.

Figures 5.18 displays the reconstructed image using 25 subsets per iteration and after 7 iterations as predicted by the termination condition.

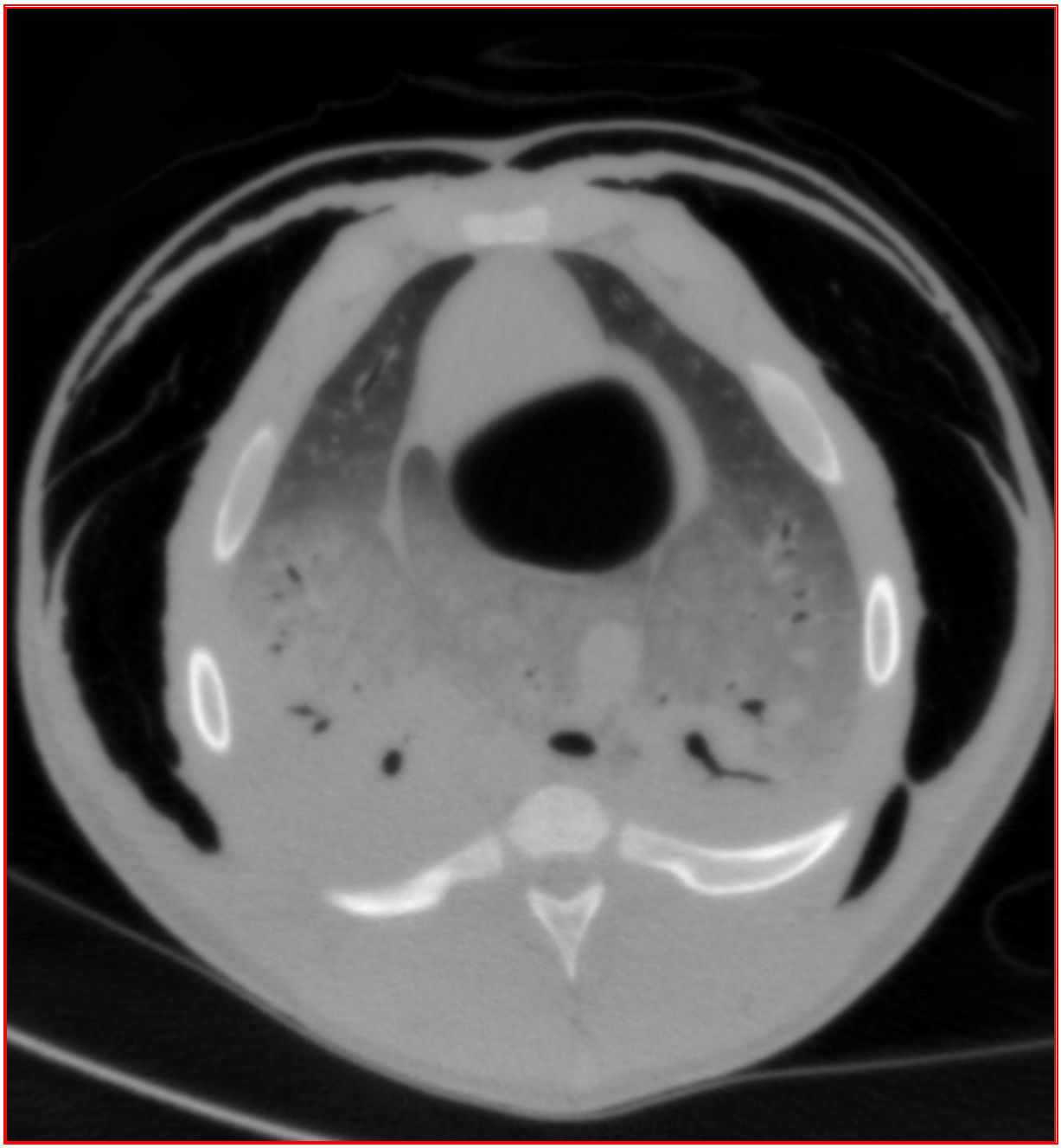

Figure 5.20 Image reconstructed using 7 iterations of OS-25 using 1498 × 580 sinograms at 1024 × 1024 pixels

5.5 Conclusions and future work

In this thesis, we have demonstrated that the GPU is an excellent, yet inexpensive platform for fast reconstruction of low-dose scans that can be used for navigation and guidance in image-guided interventions. This method of reconstruction is also extremely useful for evolving techniques such as Live Augmented Reality. We have also demonstrated that though a limited number of CPU nodes working in parallel give excellent speed-ups, the inter processor communication becomes the bottleneck as the number of ordered subsets increases. The same is the case with the increase of the number of nodes in the cluster. Also, the GPU gives a better performance at a comparable quality and is economically more viable than a cluster of computers.

Future work would involve partnering with one of the CT scanner vendors for transferring the GPU based iterative reconstruction technology proposed here on the scanners for low-dose reconstruction. More work would also be required to ensure complete removal of artifacts from metal objects in the scanned images. The tracking information that is collected during laparoscopic procedures such as Live Augmented Reality can be effectively used for removal of metal artifacts from the reconstructed images. Multiple GPU's can also be used in the scanner to reconstruct various slices to ensure yet faster reconstruction without significantly increasing the cost of the system. A thorough study using a large number of actual clinical cases on human subjects would be required before the system can qualify for use during actual procedures.

Bibliography

1. Brenner DJ, Elliston CD, Hall EJ, et al. Estimated risks of radiation-induced fatal cancer from pediatric CT. AJR 2001;176:289-296.
2. Brenner D J, Elliston CD, Estimated Radiation Risks Potentially Associated with Full-Body CT Screening, Radiology v.232, n.3, Sept 2004.
3. Ni, Jun; Li, Xiang; He, Tao; Wang, Ge , Review of Parallel Computing Techniques for Computed Tomography Image Reconstruction, Current Medical Imaging Reviews, Volume 2, Number 4, November 2006 , pp. 405-414(10)
4. Kak A C, Slaney M., Principles of computerized tomographic imaging. SIAM 2001.
5. Erdogan H, Fessler J, Monotonic Algorithms for Transmission Tomography, IEEE trans. Med. Imag., vol 18, No. 9, Sept 1999.
6. H Erdogan, G. Gualtieri, and J. A. Fessler. An ordered subsets algorithm for transmission tomography. In Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf., 1998.
7. J. A. Fessler, Grouped coordinate descent algorithms for robust edge-preserving image restoration. In Proc. SPIE 3170, Im. Recon. and Restor. II, pages 184–94, 1997.
8. Rockmore AJ, Macovski A. A maximum likelihood approach to image reconstruction. IEEE Trans. Nucl. Sci. 1976; NS-23: 1428-1432.
9. Shepp LA, Valdi Y. Maximum likelihood reconstruction for emission tomography. IEEE, Trans. Med. Imag. 1982; MI-1: 113-122.
10. Lange K, Carson R. EM reconstruction algorithms for emission and transmission tomography. J. Comput. Assist. Tomog. April 1984; 8(2): 302-316.

11. Andersen AH, Kak AC. Simultaneous algebraic reconstruction technique (SART): A superior implementation of the ART algorithm. Ultrasonic Imaging. 1984; 6: 81-94.
12. Lange K, Fessler JA. Globally convergent algorithms for maximum a posteriori transmission tomography. IEEE Trans. Image Processing. 1995; 4 (10): 1430-1438.
13. Kamphuis C, Beekman FJ. Accelerated iterative transmission CT reconstruction using an ordered subsets convex algorithm. IEEE Trans. Med. Imaging. December 1998; 17 (6).
14. Erdogan H, Fessler JA. Ordered subsets algorithms for transmission tomography. Phys. Med. Biol. 1999; 44(11).
15. Hudson HM, Larkin RS. Accelerated image reconstruction using ordered subsets of projection data. IEEE Trans. Med. Imag. 1994; 13: 601-609.
16. Xu F. Tomographic Reconstruction using graphics hardware. Nov. 2003.
17. Cabral B, Cam N, Foran J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In Proceedings of the 1994 symposium on Volume visualization. Tysons Corner. 1994; 91-98.
18. Chidlow K, Möller T. Rapid emission volume reconstruction. Volume Graphics Workshop. 2003.
19. Mueller K, Yagel R. Rapid 3-D cone-beam reconstruction with the simultaneous algebraic reconstruction technique using 2-D texture mapping hardware. IEEE Trans. Med. Imag. 2000; 19: 1227-1237.

20. Guerrini C, Spaletta G. An image reconstruction algorithm in tomography: A version for the CRAY X-MP vector computer. Computers and Graphics. 1989; 13: 367-372.
21. Miller M, Butler C. 3-D maximum a posteriori estimation for single photon emission computed tomography on massively-parallel computers. IEEE Trans. Med. Imag. 1993; 12: 560-565.
22. McCarty A, Miller M. Maximum likelihood SPECT in clinical computation times using mesh-connected parallel computers. IEEE Trans, Med. Imag. 1991; 10: 426-436.
23. Atkins MS, Murray D, Harrop R. Use of transputers in a 3-D positron emission tomography. IEEE Trans. Med. Imag. 1991; 10 (3): 276-283.
24. Chen CM, Lee SY, and Cho ZH. A parallel implementation of 3-D CT image reconstruction on hypercube multiprocessor. IEEE Trans. Nucl. Sci. 1990; 37 (3): 1333-1346.
25. Backfrieder W, Benkner S, Engelbrecht G. Web-based parallel ML_EM reconstruction for SPECT on SMP clusters. In Proceeding of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Science, Las Vegas, Nevada, CSREA Press. 2001.
26. The message passing interface (MPI) standard. http://www-unix.mcs.anl.gov/mpi
27. Li X, Ni J, Wang G. Parallel iterative cone-beam CT image reconstruction on a PC cluster. Journal of X-Ray Science and Technology. 2005; 13: 1-10.

28. Shahidi et. al., Implementation, calibration and accuracy testing of an image-enhanced endoscopy system, IEEE Trans. Med. Imag., Dec 2002,Volume: 21, Issue: 12, p1524- 1535
29. F. Xu and K. Mueller, Real-time 3D computed tomographic reconstruction using commodity graphics hardware Physics in Medicine and Biology, 2007.
30. F. Xu and K. Mueller, Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware, IEEE Trans. Nucl. Sci., vol. 52, no. 3, pp. 654-663, 2005.
31. N. Neophytou, F. Xu and K. Mueller, Hardware acceleration vs. algorithmic acceleration: can GPU-based processing beat complexity optimization for CT?, SPIE Medical Imaging'07, 2007.
32. F. Xu and K. Mueller, A comparative study of popular interpolation and integration methods for use in computed tomography, IEEE International Symposium on Biomedical Imaging, 2006.
33. F. Xu and K. Mueller, Towards a unified framework for rapid 3D computed tomography on commodity GPUs, IEEE Medical Imaging Conference, 2003.
34. Benson T.M.,Gregor J., Framework for Iterative Cone Beam MicroCT Reconstruction, IEEE trans. Nucl. Sci.,Vol. 52, No. 5, Oct 2005.
35. Feuerstein M. et al., Intraoperative Laparoscope Augmentation for Port Placement and Resection Planning in Minimally Invasive Liver Resection, IEEE trans. Med. Imag.,Vol 27.,No. 3, Mar 2008.
36. Rajan K,Patnaik L.M. et al., Linear Array Implementation of the EM Algorithm for PET Image Reconstruction., IEEE trans. Nucl. Sci.,vol 42,No 4, Aug 1995.

37. Zeng G.L. et al., A MAP Algorithm for Transmission Computed Tomography, Nuclear Science Symposium and Medical Imaging Conference, 1993., p1202-1204.
38. P Toft,A very fast Implementation of 2D Iterative Reconstruction Algorithms, Nuclear Science Symposium, 1996. Conference Record., 1996 IEEE, Vol 3, p1742-1746.
39. Mueller K, Xu F, Neophytou N., Why do Commodity Graphics Hardware Boards (GPUs) work so well for Acceleration of Computed Tomography?, SPIE Electronic Imaging 2007, Computational Imaging V Keynote.
40. Scherl H. et al., Fast GPU-Based CT Reconstruction using the Common Unified Device Architecture (CUDA), Medical Imaging Conference, Honolulu, November 2007.
41. H. Scherl, M. Koerner, H. Hofmann, W. Eckert, M. Kowarschik, and J. Hornegger. Implementation of the FDK algorithm for cone-beam CT on the Cell Broadband Engine Architecture. In J. Hsieh and M. Flynn, editors, Proceedings of SPIE Medical Imaging 2007: Physics of Medical Imaging, volume 6510, page 651058, San Diego, February 2007.
42. Message Passing Interface (MPI),SP Parallel Programming Workshop,http://www.mhpcc.edu/training/workshop/mpi/main.html
43. MPICH2 User's Guide, ver 1.0.5, http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-doc-user.pdf

44. NVIDIA CUDA Programming Guide, ver 1.1, http://developer.download.nvidia.com/compute/cuda/1_1/NVIDIA_CUDA_Programming_Guide_1.1.pdf
45. http://www.nvidia.com/object/cuda_home.html
46. Jain A. K., Fundamentals of Digital Image Processing, Prentice Hall Information and System Sciences Series, 2004.
47. O Dandekar, High Performance 3D Image Processing Architectures for Image-Guided Interventions, PhD Dissertation, Dept. of Elec. And Comp. Eng., Univ. of Maryland College Park, May 2008.
48. Kole J S, Beekman F J, Evaluation of accelerated iterative x-ray CT image reconstruction using floating point graphics hardware, Phys. Med. Biol. 51 (2006) 875–889.
49. G C Sharp, N Kandasamy, H Singh and M Folkert, GPU-based streaming architectures for fast cone-beam CT image reconstruction and demons deformable registration, Phys. Med. Biol. 52 (2007) 5771–5783.
50. 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 trans. Med. Imag., vol. 19, No. 12, Dec 2000.
51. S Zhao, D D. Robertson, G Wang, B Whiting, and K T. Bae, X-Ray CT Metal Artifact Reduction Using Wavelets:An Application for Imaging Total Hip Prostheses, IEEE trans. Med. Imag., vol. 19, No. 12, Dec 2000.

52. G Wang, D L. Snyder, J. A. O'Sullivan and M W. Vannier, Iterative Deblurring for CT Metal Artifact Reduction, IEEE trans. Med. Imag., vol. 15, No. 5, Oct 1996.

53. Robertson, D D.; Yuan, J; Wang, G; Vannier, M W., Total Hip Prosthesis Metal-Artifact Suppression Using Iterative Deblurring Reconstruction, Jour. Comp. Asst. Tomo., Volume 21(2), March/April 1997, pp 293-298.

54. Berrington de Gonzalez A, Darby S. Risk of cancer from diagnostic X-rays: estimates for the UK and 14 other countries. Lancet. 2004; 363:345-51.

55. Wang Z, Bovic AC. A universal image quality index, IEEE Sig. Proc. Lett 3:81Y84, 2002.

56. www.gpgpu.org

57. Rajan K. et al, Linear Array Implementation of the EM Algorithm for PET Image Reconstruction, IEEE Trans. Nucl. Sc., vol. 42, No 4, Aug.,1995.

58. Toft P.,A very fast Implementation of 2D Iterative Reconstruction Algorithms, IEEE NSS, vol. 3,pages 1742-1746, 1996.