國立臺灣大學電機資訊學院資訊工程學研究所

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

用 X 光影像三維重建

3D Reconstruction with X-Ray Images

陳弘毅

Hung-Yi Chen

指導教授：傅楸善 博士

Advisor: Chiou-Shann Fuh, Ph.D.

中華民國 106 年 6 月 5 日

June 5, 2017

誌謝

能順利完成本篇論文首先要感謝傅楸善教授的指導，他總是不遺餘力地在研究過程中支持我並給予建議，同時也要感謝德律科技公司提供合作的機會，支持我研究這個主題，公司同仁們專業的意見讓我受益良多，尤其要感謝巫宗昇先生、巖家和先生、鄭宇哲先生的指導。

另外也要感謝數位相機與電腦視覺實驗室的學長、學姊、同學和學弟們在研究期間的照顧。林奇賦學長、熊育萱學姐無論在任何問題總會給予我適當的幫助，讓我在研究過程中順利度過難關。同學翁丞世，在研究過程中互相扶持照顧，一同分享學業上的甘苦。學弟謝尊安、高咸培、楊至柔，實驗室因為你們的加入帶來不少歡樂，讓我忘卻課業上的煩悶。數位相機與電腦視覺實驗室的所有成員們對於幫助我完成這篇論文實在功不可沒。

最後要感謝我的家人、好友金娜和女友邱筠婕，總是毫無保留地支持我在學業上的任何決定，使我能專心投入課業，無後顧之憂。在此僅將這段時間的研究成果匯集成本論文，獻給所有曾經關心、照顧與幫助我的所有人。

國立臺灣大學碩士學位論文
口試委員會審定書

用 X 光影像三維重建

3D Reconstruction with X-Ray Images

本論文係陳弘毅君（學號 R04922099）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 106 年 6 月 5 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

傅楸善

（指導教授）

吳中昭

鄭文皇

系 主 任

中文摘要

本論文提出一個以 X 光影像三維重建印刷電路板上錫球的三維影像，以利檢測錫球內部瑕疵。在本論文中，我們採用同步代數重建技術與濾波背投影兩種方法來實作並加速執行時間，得出錫球在不同高度上的切面影像，最後將兩者的結果與現今市面上 Volume Graphics 的重建軟體之結果作比較。

關鍵字: X 光檢測、印刷電路板、同步代數重建技術、濾波背投影

Abstract

In our project, we attempt to reconstruct the 3D model of solder balls on the PCB (Printed Circuit Board) according to many projections of X-ray images of those solder balls. We mainly use two methods: SART (Simultaneous Algebraic Reconstruction Technique) and FBP (Filtered Back Projection). Consequentially, we finally obtain the 3D model of those solder balls, which can be used to inspect their inner structure. We also optimize our execution time and compare our result with that of Volume Graphics.

Key words: X-ray inspection, PCB, SART, FBP

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Overview

The PCB (Printed Circuit Board) is an essential part in many electronics products. Manufacturers need to ensure the quality for all their finished products. Therefore, PCB inspection plays an important role in industry. There are several kinds of defects, such as open solder joints, short circuit, solder bridges, component shifted, void, and so on. Many defects cannot be seen by human eyes, thus we need a precise way, such as Automated Optic Inspection (AOI), Solder Paste Inspection (SPI), and Automated X-ray Inspection (AXI). Even though there are some precise ways, the multi-layer structure of PCB where many electronic components overlap and make PCB defect inspection challenging.

We aim to implement the 3-dimensional reconstruction with X-ray images. The X-ray images are obtained by taking several photographs from different angles. With these X-ray images, we can reconstruct the images of PCB surface at different levels to inspect defect without breaking the PCB.

In this thesis, our goal is not only to produce high quality of the output images so that the void in solder balls are clear but also to make our program as fast as possible. We mainly focus on two parts: reconstruction and acceleration.

1.2 Reconstruction

Reconstruction plays a vital role in inspection because the better result of the reconstruction, the more effective we inspect the defects. The structure in our projection system is illustrated as follows:

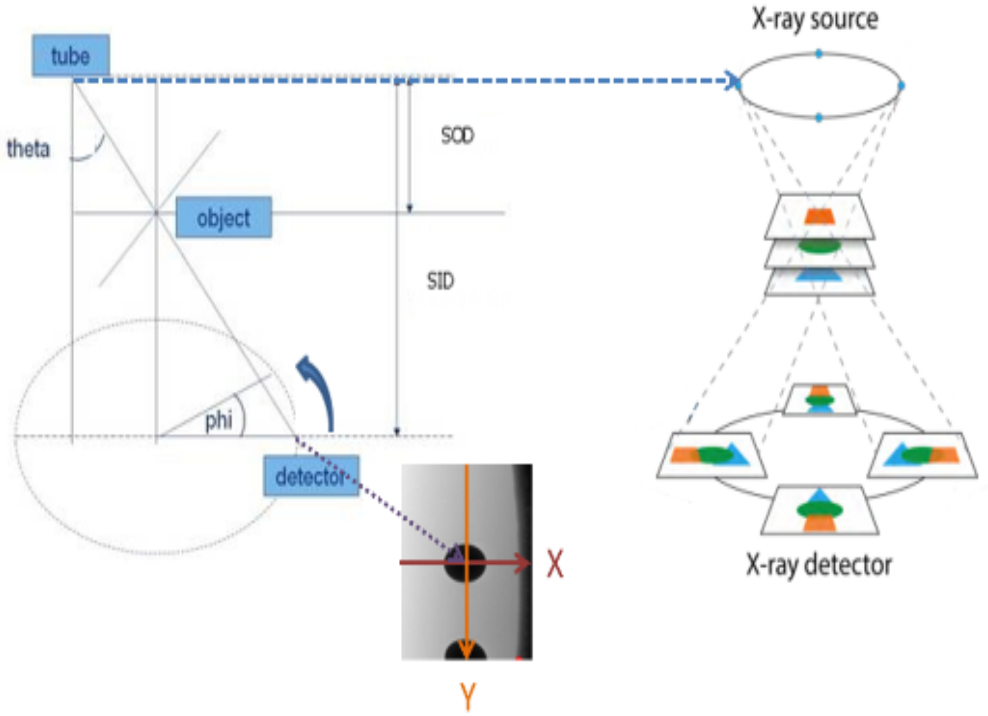


Figure 1-1: System setup. Theta is the angle between tube to detector and vertical line. SOD is Source to Object Distance (=18.432mm). SID is Source to Image Distance (=201.541mm), and phi is the angle of rotation.

By the relative motion between the source and detector, the system can obtain an image including tilt object after rotating phi degrees $\left(=\frac{360°}{\text{the number of projection images}}\right)$. When the source and detector go through 360 degrees, we can use those images to implement our reconstruction.

Many algorithms were proposed for reconstruction such as Algebraic Reconstruction Technique (ART) [13]. ART is the simplest method for reconstruction based on Radon transform, however, the quality of output images and computation efficiency are not as satisfactory [15].

Thus, we find another way to reach our goal. We decide to implement two different common approaches: Simultaneous Algebraic Reconstruction Technique (SART) and Filtered Back Projection (FBP) [10]. For FBP, we also compare the results with different kinds of filters including ramp filter, Hann filter, Hamming filter, cosine filter,

and Shepp-Logan filter.

1.3 Acceleration

After obtaining the correct reconstruction result, we will further pursue the shortest execution time in order to come up to the expected throughput. We aim at adjusting the process of the reconstruction algorithm to make program executing more effective. Furthermore, Intel® Threading Building Blocks library is used to help the parallel loop execution to be accelerated.

1.4 Thesis Organization

Chapter 2 briefly introduces related works about Algebraic Reconstruction Technique and the method to enhance contrast. Some important knowledge used in our approach will be introduced in Chapter 3. The detailed procedure of our approach will be described in Chapter 4. Chapter 5 demonstrates the experimental results. Chapter 6 contains not only the summary but also the future work, and the final chapter is the list of references.

Chapter 2 Related Work

2.1 Algebraic Reconstruction Technique

The Algebraic Reconstruction Technique is a classic of iterative algorithm used in the field of computed tomography [13] and be widely used to reconstruct the cross-section of object. We can assume that the cross-section of object is formed by an array and all the values in the array are all unknown. Thus, the reconstruction problem can be regarded as a system of linear equations, and we aim to solve every variable. The values of the pixels are considered as variables collected in a vector *f*, and the image processing is described by a matrix *A*. The measured angular projections are collected in a vector *p*. Given a real or complex $m \times n$ matrix *A* and a real or complex vector *p*, respectively, ART computes an approximation of the solution of the linear systems of equations as follows [4],

$$\boldsymbol{f_{ij}^{q+1}} = \boldsymbol{f_{ij}^{q}} + \frac{\boldsymbol{p_j} - \sum_{i=1}^{N} \boldsymbol{f_{ij}^{q}}}{\boldsymbol{N}}$$

where q is iteration; p_j is the measured data for a projection *j*; $\sum_{i=1}^{N} f_{ij}^{q}$ is the sum of the reconstructed elements along the ray; f_{ij} is an *i*-th element of the matrix *A* along the projection ray p_j; and N is the number of the reconstruction elements.

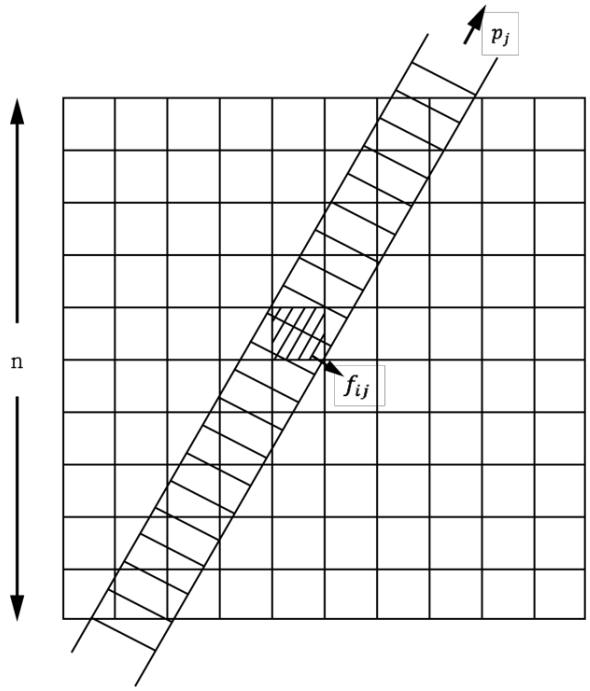


Figure 2-1: Geometric construction (using one-dimensional rays) of elements of the measurement $n*n$ matrix A in the pixel basis, where p_j is the measured data for a projection j, and f_{ij} is an i-th row element of the matrix A along the j-th projection ray [[9].

In experientmental stage, we try to recontruct 2D image as follows:

Figure 2-2: Shapes to be reconstructed.

First, we obtain projection values in different angles by generating projections of this image, 1° for each projection, from 1° to 179°. According to the approach mentioned above, we put value of each projection on a corresponding row of output image shown in Figure 2-3.

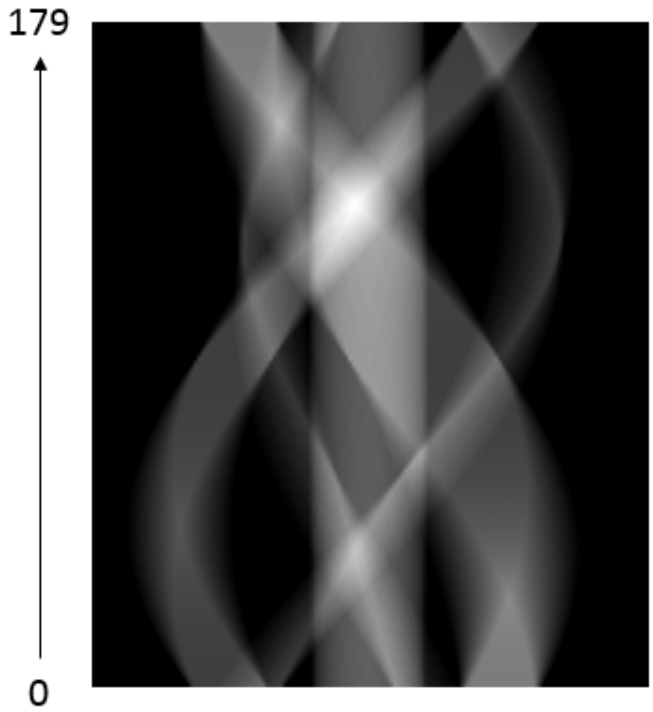
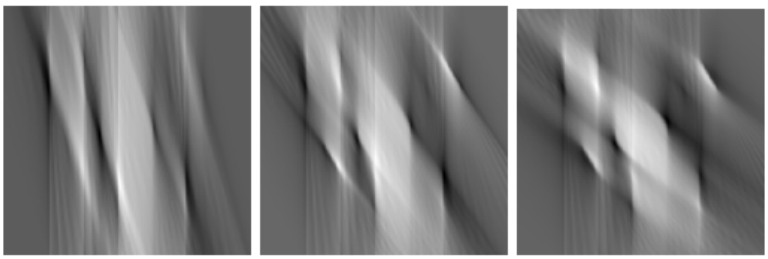
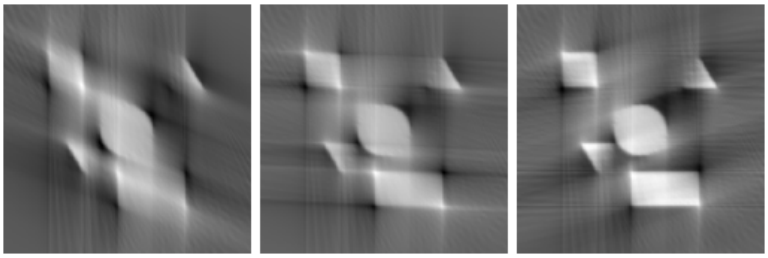


Figure 2-3: Projections of the image. Every row of this figure is one projection of a certain angle of that image (1 degree/projection).

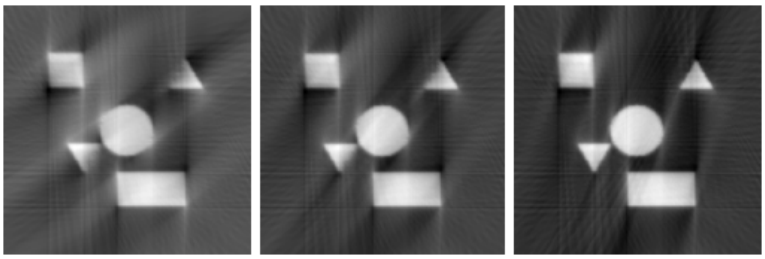
After obtaining the image above, we apply ART by a Matlab toolbox [2] to reconstruct the original image shown in Figure 2-4. However, ART seems to take too many iterations for the reconstuction image. As a result, We find other way to reach our goal.



(a) 1100 iterations. (b) 2200 iterations. (c) 3300 iterations.



(d) 4400 iterations. (e) 5500 iterations. (f) 6600 iterations.



(g) 7700 iterations. (h) 8800 iterations. (i) 9900 iterations.

Figure 2-4: The results after increasing number of iterations (1100 iterations/image).

2.2 Histogram Specification

The void is unclear even by our eyes after reconstruction in some samples. The main reason is manufacturers hope to get throughput as fast as possible. This makes the projection system generate fewer projection images. However, the number of projection images is not enough for reconstruction to get high quality reconstructed images. As a result, we need to apply some enhancements additionally.

Because our output is in low contrast, we choose histogram specification or histogram matching for enhancement. The principal idea is to convert a histogram of input image into another specified histogram [12]. A mapping function can be found below:

First, equalize the histogram p_x of the input image x:

$$y = f(x) = \int_0^x p_x(u)du$$

Then, equalize the desired histogram p_z of the output image:

$$y' = g(z) = \int_0^z p_z(u)du$$

Because y and y' have the same equalized histogram, we get

$$y = y'$$

The transform from the given image x to the desired image z can be:

$$z = g^{-1}(y') = g^{-1}(y) = g^{-1}(f(x))$$

However, as the image in grey levels are discrete, the continuous mapping above is just an approximation. The discrete histograms are not necessary in the same form. Therefore, we have to find the mapping between them with lowest error so that the mapping from y to y' can be tenable. The steps of histogram matching algorithm are as follows:

Step 1: Generate histogram h_x of input image x, and find its cumulative distribution function C_x:

$$C_x[j] = \sum_{i=0}^{j} h_x[i]$$

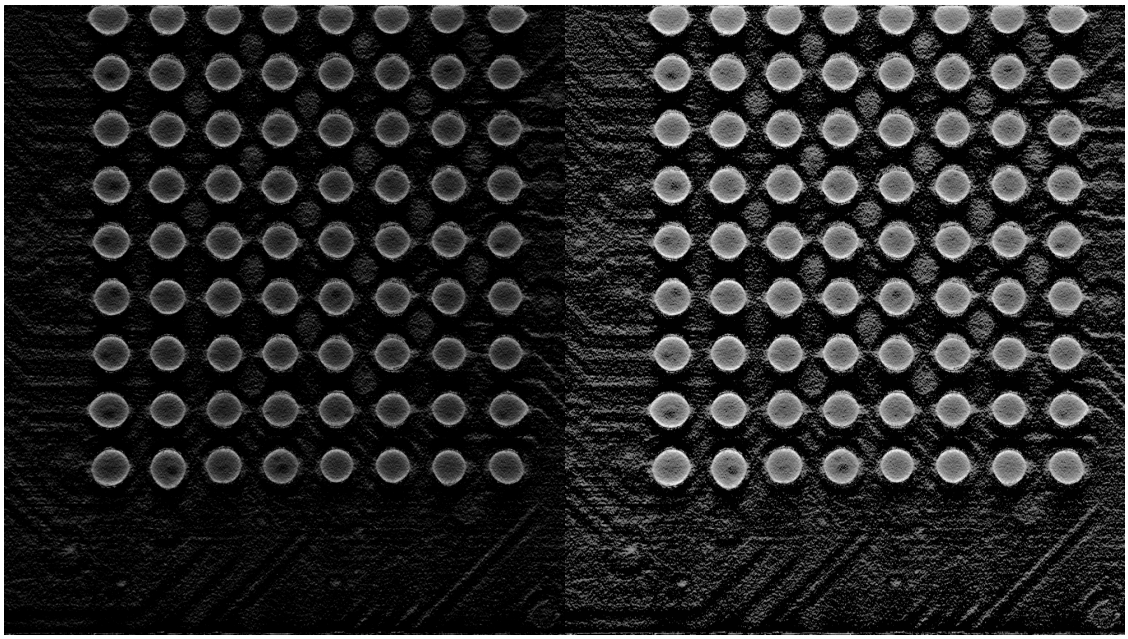
Step 2: Generate histogram h_z of desired image z, and find its cumulative distribution function C_z:

$$C_z[j] = \sum_{i=0}^{j} h_z[i]$$

Step 3: Build a lookup table $lookup[i] = j$. For each input level i, find an output level j so that $H_z[j]$ best matches $H_x[i]$:

$$|H_z[j] - H_x[i]| = \min_k |H_z[i] - H_x[k]|$$

The following example shows the result of histogram specification:



(a) (b)

Figure 2-5: (a) Original reconstructed image without histogram specification. (b) Reconstructed image after histogram specification.

Chapter 3 Background

3.1 Radon Transform

In mathematics, the Radon transform was introduced in 1917 by Johann Radon, who also provided a formula for the inverse transform [15]. The transform is the integral transform which takes a function *f* defined on the plane to a function $g(\phi, s)$ defined on the (two-dimensional) space of lines in the plane, whose value at a particular line is equal to the line integral of the function over that line. The Radon transform is widely applicable to tomography, the creation of an image from the projection data associated with cross-sectional scans of an object [15].

A function $g(\varphi, s)$ is the line integral of the image intensity $f(x, y)$, along a line l that is distance s from the origin and at angle φ off the x-axis [7].

$$g(\phi, s) = \int_l f(x, y) dl$$

All points on this line satisfy the equation:

$$x\sin(\phi) - y\cos(\phi) = s$$

Therefore, the forward projection function $g(\phi, s)$ can be rewritten as:

$$g(\phi, s) = \int \int f(x, y)\delta(xsin\phi - ycos\phi - s)dxdy$$

where the $\delta(n)$ is a delta function, if $n = 0$, $\delta(n) = 1$; else, $\delta(n) = 0$.

The collection of these $g(\phi, s)$ at all ϕ is called the Radon transform of Image. An illustrated is as follows:

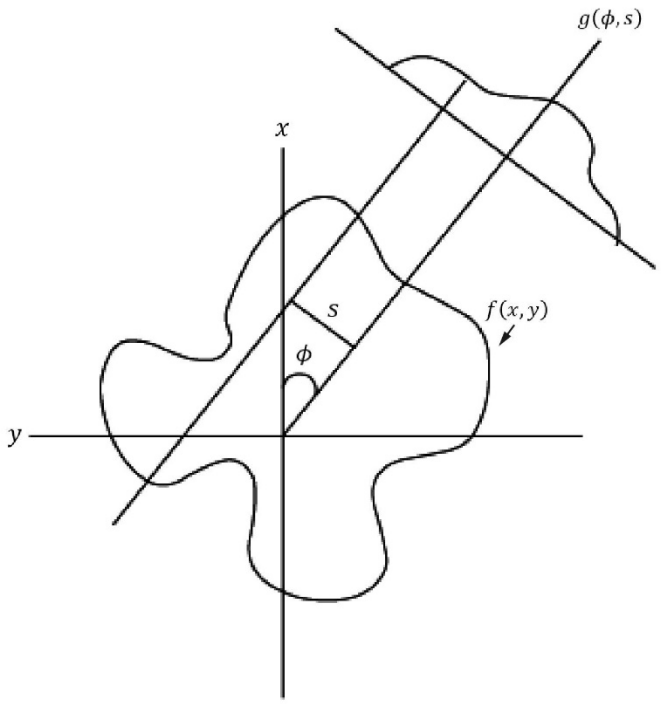


Figure 3-1: Illustration of Radon transform, where *x, y* are axes; *s* is the shifting constant; ϕ is rotation angle; $f(x, y)$ is the intensity of object in (x, y); and $g(\phi, s)$ is the forward projection value after Radon transform.

If a function f represents an unknown density, then the Radon transform represents the projection data obtained as the output of a tomographic scan. Hence the inverse of the Radon transform can be used to reconstruct the original density from the projection data, and thus it forms the mathematical underpinning for tomographic reconstruction, also known as image reconstruction [15].

3.2 Fourier Slice Transform

In mathematics, applying the Fourier Slice Transform, central slice theorem, or projection-slice theorem [[14] to two-dimensional image that the results of the following two computations are equal:

(i) Take a two-dimensional function, project it onto a (one-dimensional) line, and do a Fourier transform of that projection.

(ii) Take the same function above, but do a two-dimensional Fourier transform first, and then slice it through its origin, which is parallel to the projection line.

In operator terms, if F_1 and F_2 are the 1- and 2-dimensional Fourier transform operators mentioned above, P_1 is the projection operator (which projects a 2-D function onto a 1-D line) and S_1 is a slice operator (which extracts a 1-D central slice from a function), then:

$$F_1P_1 = S_1F_2$$

The projection-slice theorem is well proven for the case of two dimensions. Without loss of generality, we can take the projection line to be the *x*-axis and the law still applies if we use a shifted and rotated line. Using a shifted line (in *y*-axis) gives the same projection and therefore the same 1D Fourier transform results. The rotated function is the Fourier pair of the rotated Fourier transform, for which the theorem again holds.

If $f(x,y)$ is a two-dimensional function, then the projection of $f(x,y)$ onto the *x*-axis is $p(x)$ where

$$p(x) = \int_{-\infty}^{\infty} f(x,y)dy$$

The Fourier transform of $f(x,y)$ is

$$F(k_x,k_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(xk_x+yk_y)}dxdy$$

The slice is then $s(k_x)$

$$s(k_x) = F(k_x,0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi ixk_x}dxdy$$

$$= \int_{-\infty}^{\infty}[f(x,y)dy]\int_{-\infty}^{\infty} e^{-2\pi ixk_x}dx$$

$$= \int_{-\infty}^{\infty} p(x)e^{-2\pi ixk_x}dx$$

According to the equation above, the slice is not associated with *y*-axis and the rotation is correspondent, so using a shifted line (in *y*-axis) gives the same Fourier transform results and the rotated function is the Fourier pair of the rotated Fourier transform.

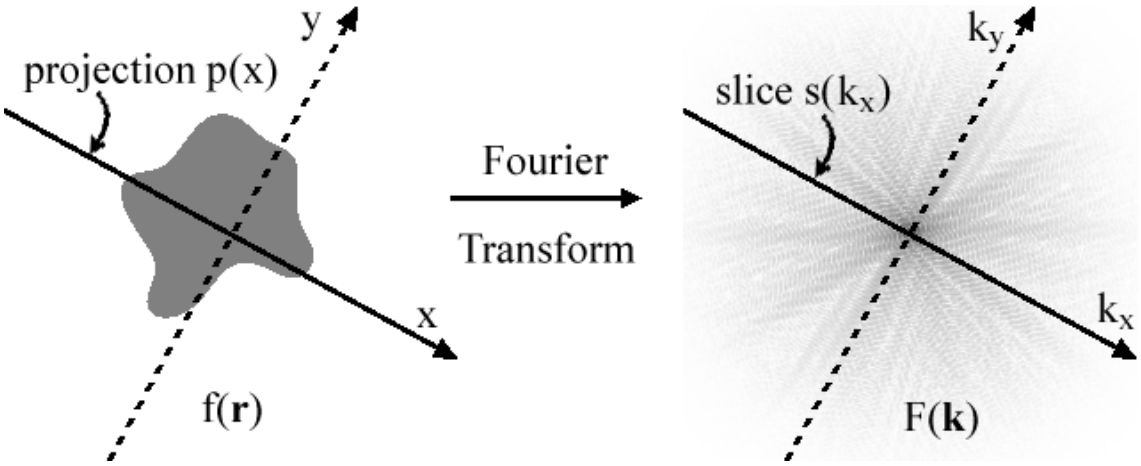


Figure 3-2: $f(r)$ and $F(k)$ are 2-dimensional Fourier transform pairs. The projection of $f(r)$ onto the *x*-axis is the integral of $f(r)$ along lines of sight parallel to the *y*-axis and is labelled *p(x)*. The slice through $F(k)$ is on the k_x-axis, which is parallel to the x-axis and labelled $s(k_x)$. The projection-slice theorem states that $p(x)$ and $s(k_x)$ are 1-dimensional Fourier transform pairs [14].

3.3 Radon Transform with Fourier Slice Transform

The Fourier slice transform shows that the 1D Fourier transform of the projection function $g(\phi, s)$ mentioned in Radon transform is equal to the 2D Fourier transform of the image evaluated on the line that the projection was taken on (the line that $g(\phi, 0)$ was calculated from) [7]. Thus, we know what the 2D Fourier transform of the image looks like (or at least what it looks like on certain lines and then interpolate), and we can simply take the 2D inverse Fourier transform and obtain the original image. We can show the Fourier slice theorem in the following way:

The 1D Fourier Transform of g is given by:

$$G(\phi, \omega) = \int e^{-j\omega s} g(\phi, s) ds$$

Substitute the expression for $g(\phi, s)$ into the Radon transform to get:

$$G(\phi, \omega) = \int \int \int f(x, y) e^{-j\omega s} \delta(x sin\phi - y cos\phi - s) dx dy ds$$

Use the sifting property of the Dirac delta function to simplify to get:

$$G(\phi, \omega) = \int \int f(x, y) e^{-j\omega(x sin\phi - y cos\phi - s)} dx dy$$

The definition of the 2D Fourier transform of f:

$$F(u, v) = \int \int f(x, y) e^{-j(ux+vy)} dx dy$$

It shows that Radon transform with Fourier slice transform is just $F(u,v)$ evaluated at $u = w * sin(\phi)$ and $v = -w * cos(\phi)$, which is the line corresponding to the projection $g(\phi, s)$. Therefore, we can apply some image operation in Fourier domain instead of spatial domain to get some acceleration and inverse to spatial domain to get our output.

3.4 Filter

For Filtered Back Projection, filtering plays a significant role by changing the views in two ways. First, the top of the pulse is made flat, resulting in the final back projection creating a uniform signal level within the circle. Second, negative spikes have been introduced at the sides of the pulse. When back projected, these negative regions counteract the blur near the target object [10].

There are different methods to filter the reconstructed image. We compare with different kinds of filters including Ramp filter, Hann filter, Hamming filter, cosine filter, and Shepp-Logan filter.

- Ramp filter

We expect that a high-pass filter would eliminate the artifact by amplifying high-frequency components in the back projection. Ramp filter is generally used in practice.

- Hann filter

The statistical noise in the data manifests in Fourier space as high-frequency components. Thus the process of filtered back projection amplifies noise in the image. In order to reduce this effect, a range of modifications to the ramp filter can be used. The most commonly used of these is the Hann filter [1].

$$w(n) = \frac{1}{2}\left[1 + cos\left(\frac{2\pi n}{N-1}\right)\right], \qquad 0 \le n \le N-1$$

where N represents the width.

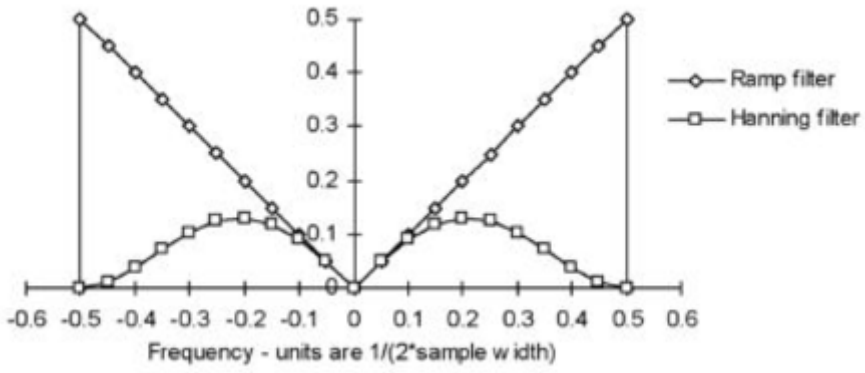


Figure 3-3: Comparison with Ramp filter and Hann filter [1].

- Hamming filter

The window with these particular coefficients was proposed by Richard W. Hamming [17]. The window is optimized to minimize the maximum (nearest) side lobe, giving it a height of about one-fifth that of the Hann window.

$$w(n) = 0.54 - 0.46 * cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \leq n \leq N-1$$

where N represents the width.

- Cosine filter

$$w(n) = \cos\left(\frac{\pi n}{N-1}\right), \qquad 0 \leq n \leq N-1$$

where N represents the width.

- Shepp-Logan filter

$$w(n) = \frac{\sin\left(\frac{\pi n}{N-1}\right)}{\frac{\pi n}{N-1}}, \qquad 0 \leq n \leq N-1$$

where N represents the width.

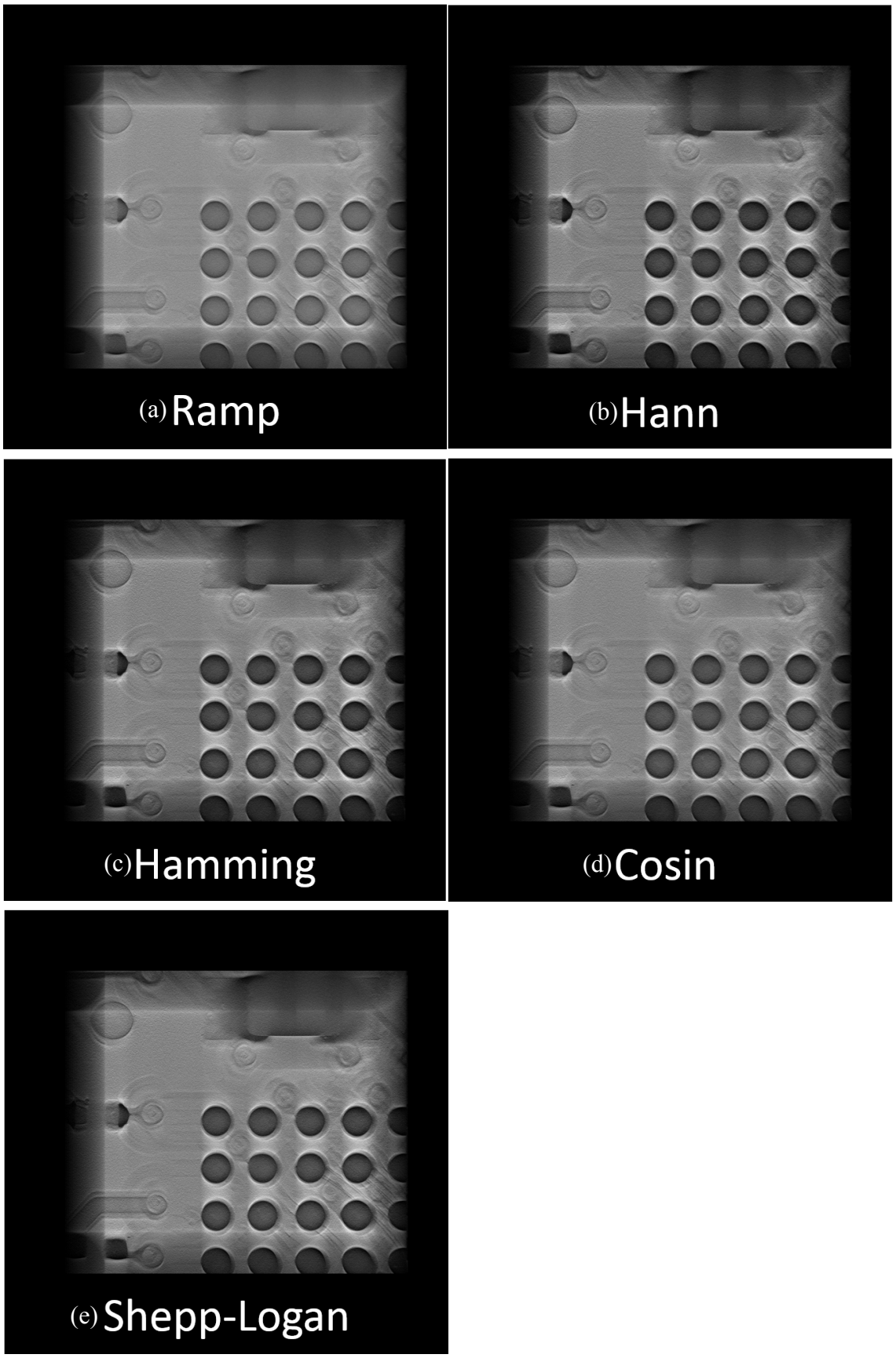


Figure 3-4: Comparison with each filter on reconstructed images at height 200.

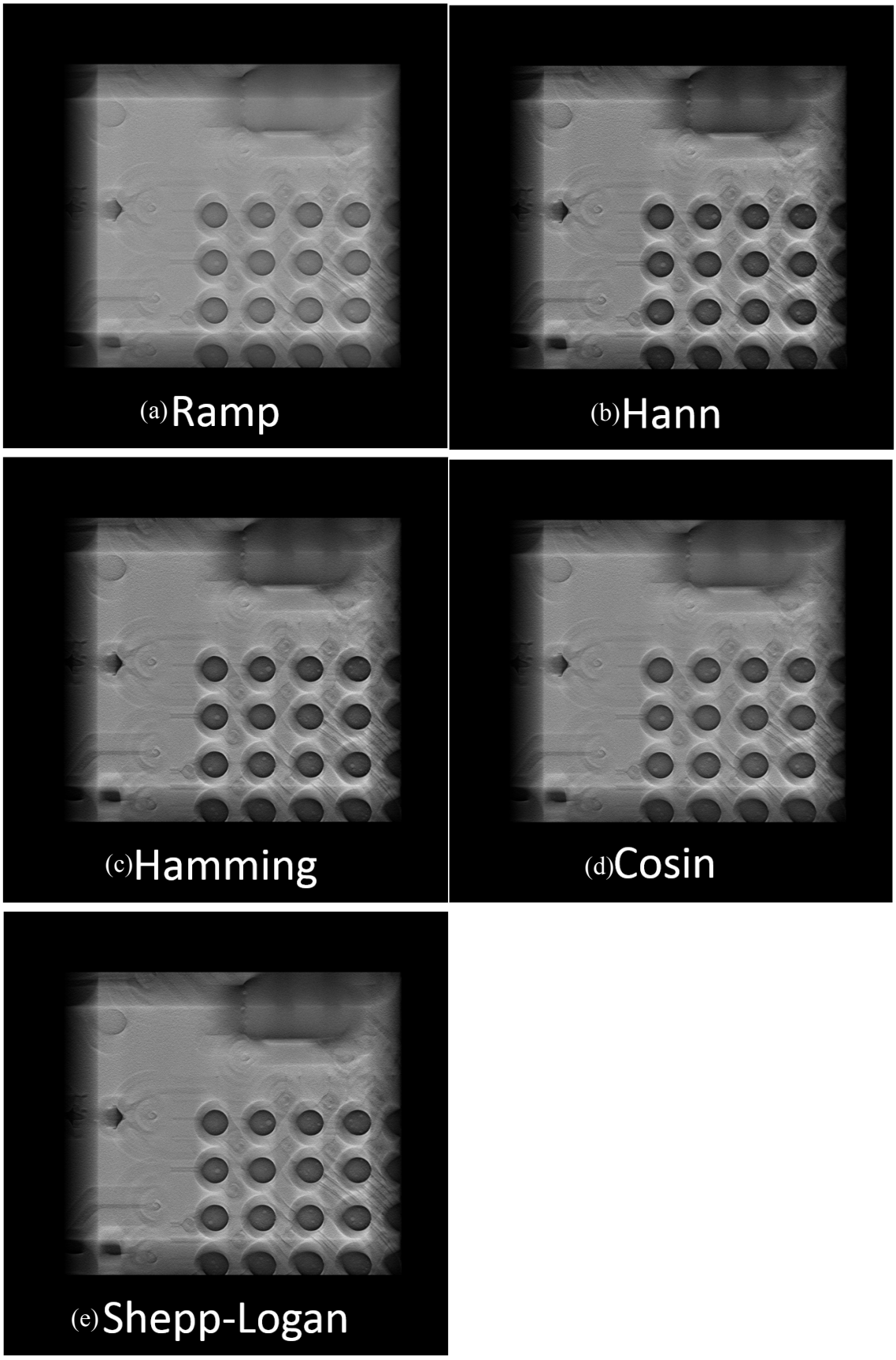


Figure 3-5: Comparison with each filter on reconstructed images at height 260 of object.

Chapter 4 Methodology

4.1 Overview

In this chapter, detailed methodology is completely discussed, including how to implement the two reconstruction methods: SART and FBP. In order to get higher throughput, we make our program executes in parallel by Threading Building Blocks (Intel ® TBB) by Intel. For FBP, we additionally generate a filter and enhance the reconstructed images contrast. More details will be discussed in next sub-sections. The flow of our proposed algorithm is as follows:

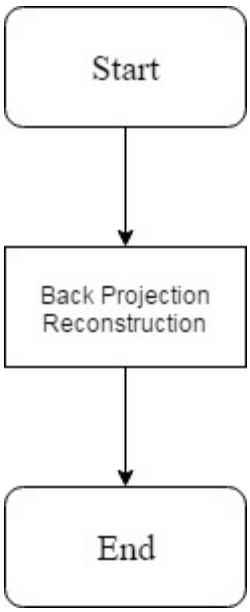


Figure 4-1: The flowchart of SART.

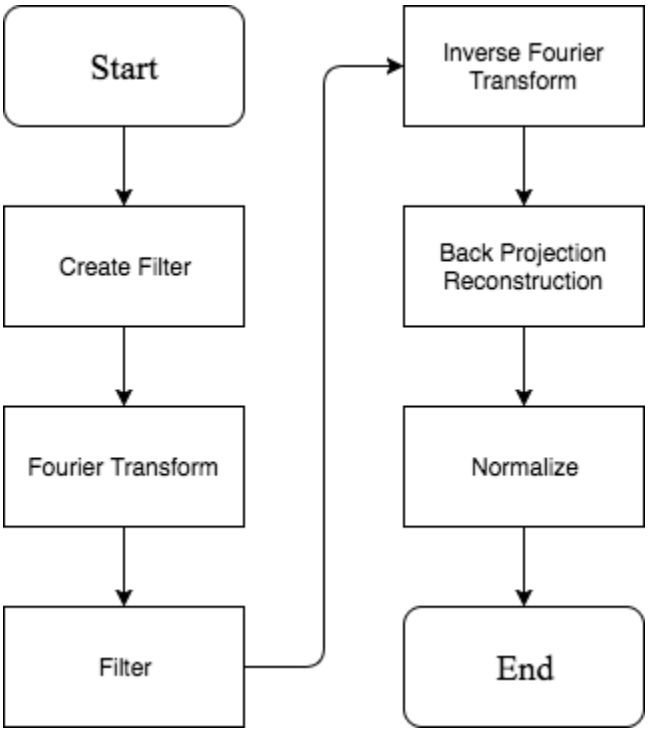


Figure 4-2: The flowchart of FBP.

4.2 Simultaneous Algebraic Reconstruction Technique (SART)

The Simultaneous Algebraic Reconstruction Technique [16], proposed by Anders Andersen and Avinash Kak in 1984, has had an improvement in Computed Tomography (CT). Although the projection data which the system possesses is limited, the SART still generates a good reconstruction in just one iteration, and it is superior to standard Algebraic Reconstruction Technique.

The ART computes an approximation of the solution of the linear systems of equations as follows [4],

$$\boldsymbol{f_{ij}^{q+1} = f_{ij}^{q} + \frac{p_j - \sum_{i=1}^{N} f_{ij}^{q}}{N}}$$

where q is iteration; p_j is the measured data for a projection j; $\sum_{i=1}^{N} f_{ij}^{q}$ is the sum of the reconstructed elements along the ray; f_{ij} is an i-th element of the matrix A along the projection ray p_j; and N is the number of the reconstruction elements.

In computation, the ART focuses on pixels along a ray in each projection. Conversely, the idea of SART is based on that a pixel must be through by not just one ray in one iteration. It makes a great difference between SART and ART that ART only updates all the pixels along one ray in one iteration, but SART updates a pixel by a value considering all the ray through it in one iteration [[5]. Consequently, SART is much faster with solving equation than ART.

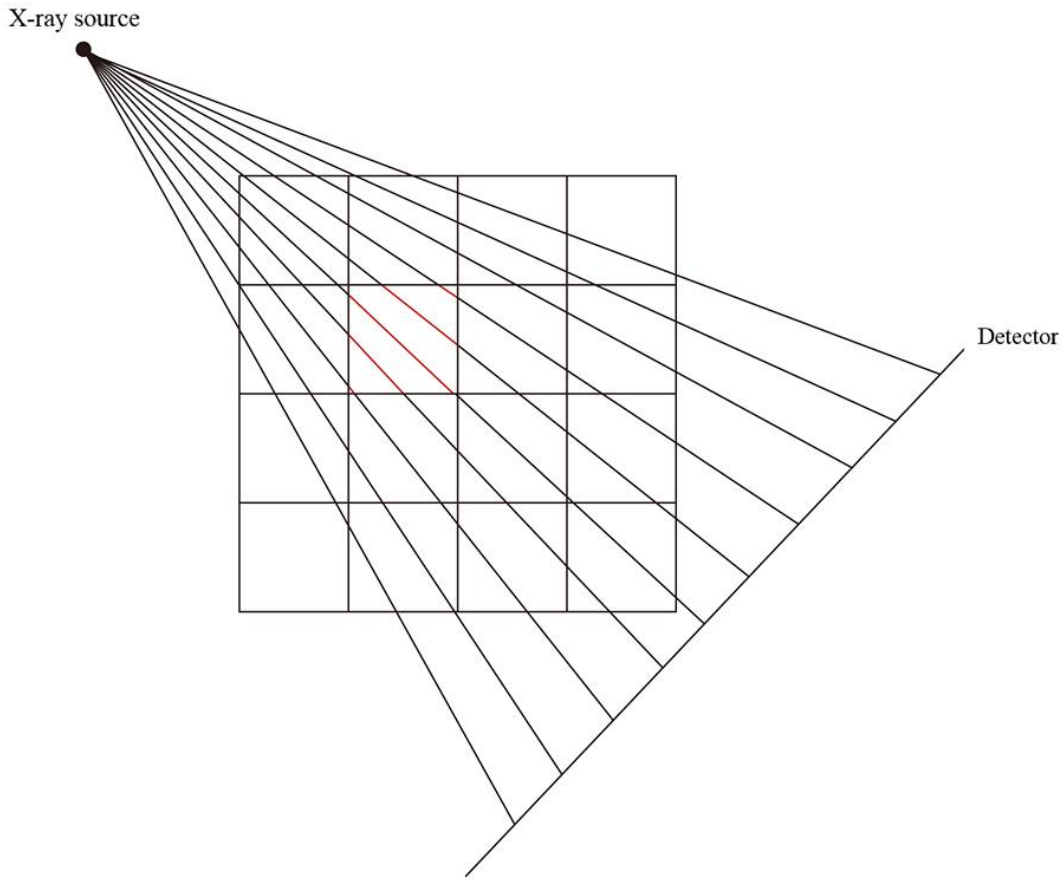


Figure 4-3: Illustration that pixel must be through by not just one ray in one iteration.

The SART computes an approximation of the solution of the linear systems of equations with a weighting matrix W [11]. The element w_{ij} in W denotes the contribution of f_{ij} to p_j. We assume that the width of each ray is equal to the image pixel size. Then the element w_{ij} can be calculated (or, at least, estimated) according to the projecting procedure as follows:

$$w_{ij} = \begin{cases} 1, \text{ the } j\text{-th ray passes the center of the } i\text{-th pixel} \\ 0, \quad \text{else} \end{cases}$$

Sum up the main points of the above, the SART computes an approximation of the solution of the linear systems of equations as follows [4]

$$f_{ij}^{q+1} = f_{ij}^{q} + \frac{\sum_{p_j \in P_\phi} \lambda \dfrac{p_j - \sum_{k=1}^{N} f_{ij}^{q} w_{ik}}{\sum_{k=1}^{N} w_{ik}}}{\sum_{p_j \in P_\phi} w_{ij}}$$

where q is iteration; p_ϕ is all projection values in one degree; p_j is the measured data for a projection j; λ is relaxation parameter; $\sum_{i=1}^{N} f_{ij}^{q}$ is the sum of the reconstructed elements along the ray; f_{ij} is an i-th element of the matrix A along the projection ray p_j; and N is the number of the reconstruction elements.

4.3 Filtered Back Projection

The Radon transform goes with Fourier slice theorem and Fourier transform, often called the Filtered Back Projection, which is another way to reconstruct image. Unlike ART or SART which are iterative methods, FBP is a direct method. It is a technique to correct the blurring encountered in simple back projection shown in Figure 4.2 [10].

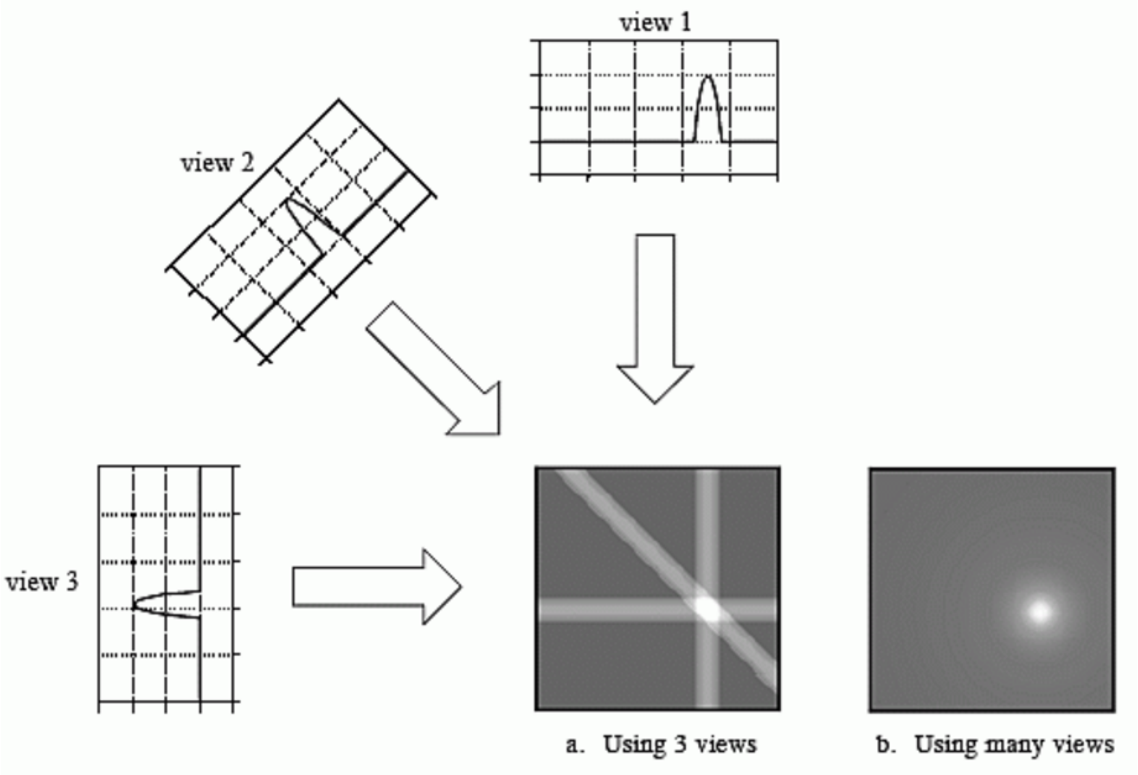


Figure 4-4: Back projection reconstructs an image by taking each view and smearing it along the path it was originally acquired. The resulting image is blur compare with the correct image [[10].

For FBP, each view is filtered before the back projection to counteract the blurring point spread function illustrated in Figure 4-3. That is, each of the one-dimensional view is convolved with a one-dimensional filter kernel to create a set of filtered views. These filtered views are then back projected to provide the reconstructed image which is much more correct than the result from back projection.

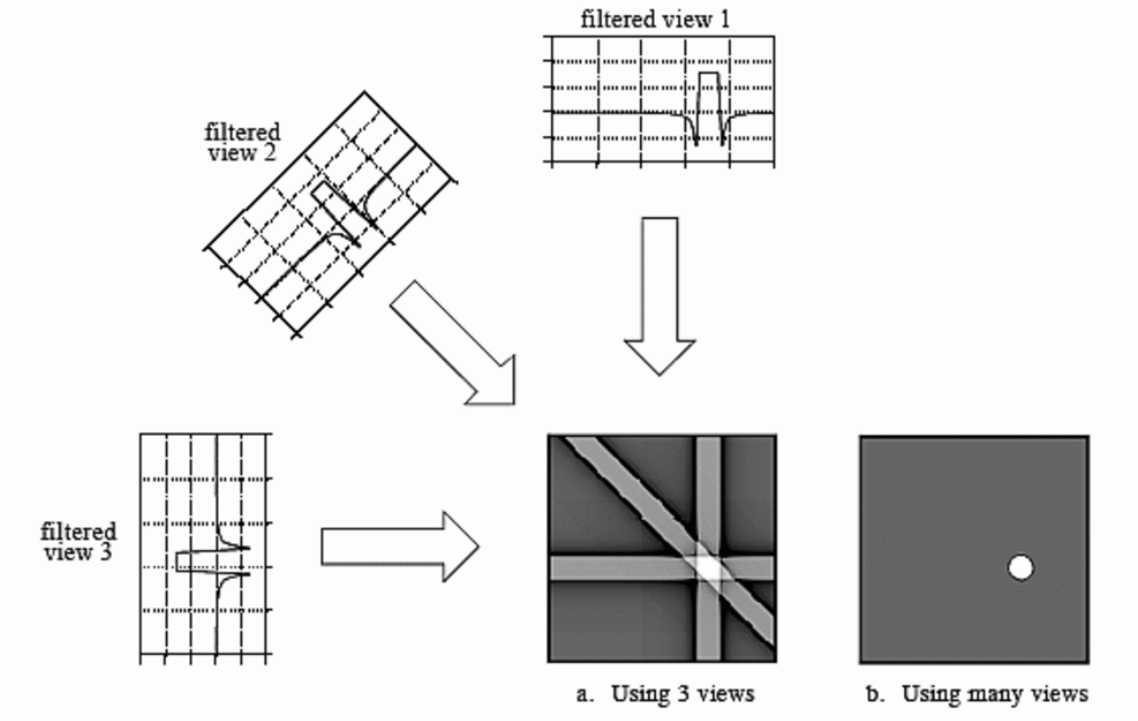


Figure 4-5: Filtered back projection reconstructs an image by filtering each view before back projection. This removes the blurring seen in simple back projection, and results in a mathematically exact reconstruction of the image.

In mathematics, first recall the definition for the 2D inverse Fourier Transform [7]

$$f(x,y) = \frac{1}{(2\pi)^2} \int \int_{R^2} F(u,v) e^{j(ux+vy)} dxdy$$

Make a change of variable from rectangular to polar coordinates and replace $F(\phi, w)$ with $G(\phi, w)$, we get

$$f(x,y) = \frac{1}{4\pi}\int\int_{R^2} G(\phi,\omega)e^{j\omega(xsin\phi+ycos\phi)}|w|dxdy$$

where $|w|$ is the determinant of the Jacobian of the change of variable from rectangular to polar coordinates. Notice that we have to multiply our projections by $|w|$ in the Fourier domain. Thus we can see $|w|$ as a filter.

$$Q(\phi,\omega) = G(\phi,\omega)|w|$$

This product is called the filtered back projection at angle ϕ.

To sum up, the FBP algorithm consist of the following steps [[8]:

(i) Compute a one-dimensional Fourier transform of each projection.

(ii) Multiply the Fourier transform of each projection by the weighting filter.

(iii) Compute the inverse one-dimensional Fourier transforms of the filtered projection.

(iv) Apply back projection to the processed projections in spatial domain to obtain the two-dimensional reconstructed object.

For the filter we use in this thesis, we generate a modified Hann filter

$$w(n) = \frac{1}{3}\left[1 + cos\left(\frac{2\pi n}{N-1}\right)\right], \qquad 0 \le n \le N-1$$

where Nrepresents the width.

The difference between Hann filter and our modified filter is the constant. We substitute $\frac{1}{3}$ for $\frac{1}{2}$. Our filter may make the image much smoother than Hann filter.

4.4 Back Projection

4.3.1 Value

The characteristics of the X-ray system as mentioned in Figure 1-1, the relative motion between the source and detector generates projections with angle $\theta\ (= 35°)$. Thus we generate a reconstruction architecture illustrated as follows:

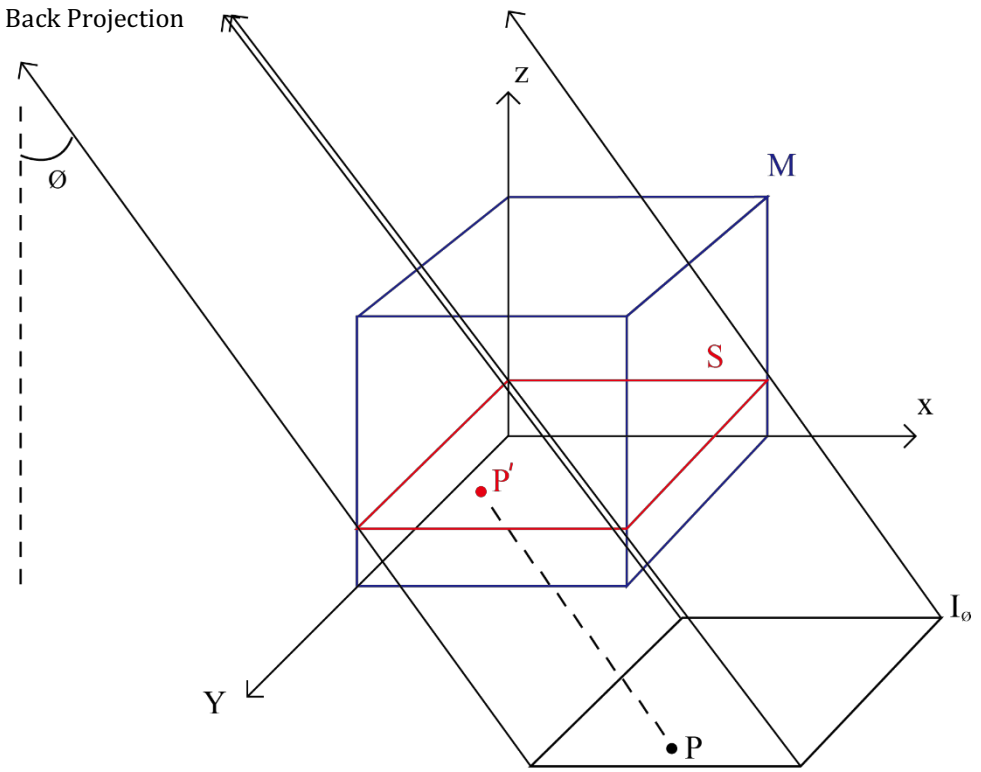


Figure 4-6: Reconstruction architecture, where matrix M contains whole PCB; p is a point on projection image I_ϕ with angle ϕ; p' is a target point corresponding to p on slice S; and θ is the angle between tube to detector and vertical line.

First, we build a three-dimensional matrix which contains the whole PCB we aim to reconstruct where I_ϕ is a projection image with rotated angle ϕ $(= \frac{360° \times i}{\text{the number of projection images}}, 0 < i < \text{the number of projection images})$, and we apply every I to reconstruct the image of slice S with different heights. Each pixel on slice may be passed by many X-ray. As a result, the intensity of each pixel is the accumulation of values of every X-ray passing through.

$$f_{ij} = \sum p_k$$

where f_{ij} is an intensity of i-th pixel on slice S with height j, and p_k is k-th projection passing through pixel f_{ij}.

4.3.2 Coordinate

Now we have the value to assign to target pixel. The next step is to define which pixel on the slice with different height corresponding to the pixel in projection images. Based on an idea that the objects on focal plane stay consistent in all projections, we apply focal plane as a reference to derive the corresponding coordinate. Illustrated as follows:

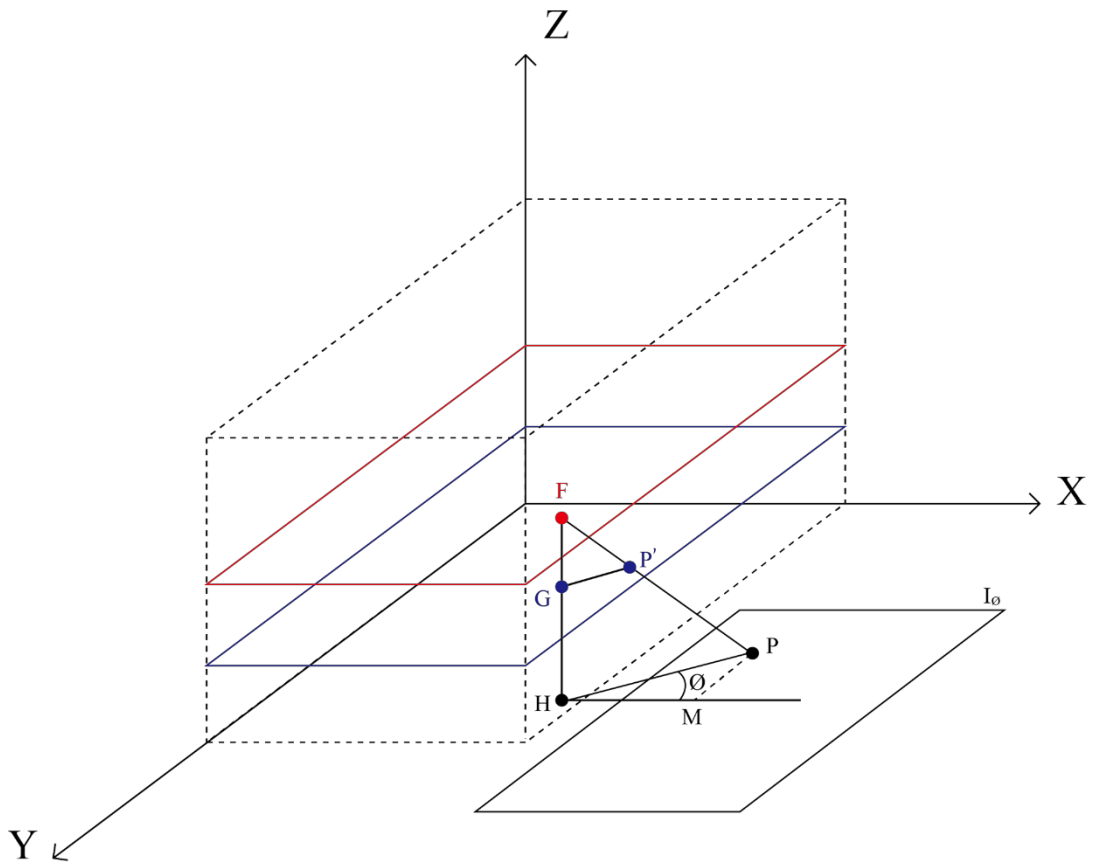


Figure 4-7: Illustrate the architecture to find the pixel on $z = z_{p'}$ passed by a certain X-ray.

According to Figure 4-7, the point F is on the focal plane $z = z_F$, and P is its corresponding point in the projection image. The projection ray $\overline{FP}$ intersects with plane $z = z_{P'}$ at P' where $\overline{FH}$ is a perpendicular line with plane *X*-*O*-*Y*, which intersects with plane $z = z_{p'}$ at G; ϕ is rotated degree; and θ is the angle between tube to detector and vertical line.

Assume the coordinate of F to be

$$F(x_F,\ y_F,\ z_F)$$

We can get the coordinates of G, H

$$G\left(x_F,\ y_F,\ z_{p'}\right)$$

$$H(\ x_F,\ y_F, 0)$$

The points P and F are on the same line. In triangle HMP,

$$\Delta\ x = z_F * tan\theta * cos\theta$$

$$\Delta\ y = -z_F * tan\theta * sin\theta$$

Thus, we can get the coordinate of P

$$P(\ x_F + z_F * tan\theta * cos\theta,\ y_F - z_F * tan\theta * sin\theta, 0)$$

Triangle FHP is similar to triangle FGP'. Thus

$$\frac{\overline{GP'}}{\overline{HP}} = \frac{\overline{FG}}{FH}$$

$$\Rightarrow \begin{cases} \dfrac{(x_{p'} - x_F)}{(x_F + z_F * tan\theta * cos\theta) - x_F} = \dfrac{z_F - z_{p'}}{z_F} \\ \dfrac{(y_{p'} - y_F)}{(y_F - z_F * tan\theta * sin\theta) - y_F} = \dfrac{z_F - z_{p'}}{z_F} \end{cases}$$

$$\Rightarrow \begin{cases} x_{P'} = x_F + tan\theta * cos\theta * (z_F - z_{p'}) \\ y_{P'} = y_F - tan\theta * sin\theta * (z_F - z_{p'}) \end{cases}$$

Finally, we can get the target point coordinate

$$p'(x_F + tan\theta * cos\theta * (z_F - z_{p'}), y_F - tan\theta sin\theta * (z_F - z_{p'}), z_{p'})$$

4.5 Contrast Enhancement

The image shown to users should be 8-bit image, but our reconstructed image is 16-bit in low-contrast. Rather than operating on 8-bit image, we decide to normalize the image with narrow range when converting 16-bit image to 8-bit image. We use half of minimum intensity and $\frac{1}{128}$ of maximum intensity in 16-bit image to normalize to 8-bit image. As a result, it can enhance the contrast and preserve some information which may be lost if we use normal converting.

4.6 Acceleration

Although we can correctly obtain the reconstructed images, comparing with the throughput of Volume Graphics, ours is not good enough. Based on the program including many for-loops, we decide to make them parallel. We choose Threading Building Blocks by Intel.

TBB is a library that supports scalable parallel programming using standard ISO (International Standards Organization) C++ code [3]. TBB does not require special languages or compilers. TBB is designed to promote scalable data parallel programming. Additionally, TBB fully supports nested parallelism, so users can build larger parallel components from smaller parallel components. To use the library, users specify tasks, not threads, and let the library map tasks onto threads in an efficient manner. The code section is shown as follows:

```
for(int z=0; z<200; z++)
{
    Reconstruction[z] = cvCreateMat(OutWidth, OutHeight, CV_32FC1);
    for(int i=0; i<Projection; i++)
    {
        int cosV = (float)(100-z) * 1.68 * cosf(RadiusUnit * i);
        int sinV = - (float)(100-z) * 1.68 * sinf(RadiusUnit * i);
        for(int x=0; x<InWidth; x++)
        {
            for(int y = 0; y < InHeight; y++)
            {
                Reconstruction[z]->data.fl[(x+sinV)*Reconstruction[z]->cols+cosV+y]
                += cvimage1[z][x][y];
            }
        }
    }
}
```

(a)

```
tbb::parallel_for(0, 200, 1, [&](int z)
{
    Reconstruction[z] = cvCreateMat(OutWidth, OutHeight, CV_32FC1);
    float constant = (float)(100-z) * 1.68;
    tbb::parallel_for(0, Projection, 1, [&](int i)
    {
        int cosV = constant * cosf(RadiusUnit * i);
        int sinV = - constant * sinf(RadiusUnit * i);
        tbb::parallel_for(0, InWidth, 1, [&](int x)
        {
            int ReconstructionIndex = (x+sinV)*Reconstruction[z]->cols+cosV;
            int cvimageIndex =(i*InWidth*InHeight)+x*InWidth;
            for(int y = 0; y < InHeight; y++)
                Reconstruction[z]->data.fl[ReconstructionIndex++] += cvimage[cvimageIndex++];
        });
    });
});
```

(b)

Figure 4-8: (a) The original code section with regular for-loop. (b) The new code section with TBB parallel for-loop.

With TBB, we not only change the original for-loop to the parallel for-loop, but need to consider modifying the architecture of the program. In the new code section in Figure 4-8, we take apart some code in original code section. Because of the characteristic of program execution, eliminating redundant execution of task can improve efficiency.

Two-dimensional images, 3D volumes, and other multi-dimensional data frequently require loops that sweep through an array to compute statistics, normalize values, or apply transfer functions [6]. Thus, optimizing memory access is another way we focus on. Based on characteristic of memory sequential accessing, row-major access is faster than column-major access. This characteristic leads us to access images in row-major parallel. On the other hand, maintaining a multi-dimensional array within a single linear array is a common performance technique. In original code section in Figure 4-8, the array is directly declared as a three-dimensional array. But in new code section, we declare the array as a one-dimensional array and the index is still correctly mapped to three-dimensional array. It will greatly improve efficiency, if we can use one-dimensional instead of three-dimensional array. However, race condition need to be cautious before using parallel for-loop. The comparison of execution time with two code sections is shown as follows:

FBP / Execution time (s)	with For-loop	with Parallel For-loop
Make Filter	0.12348	0.069422
Fourier Transform	2.55927	0.232338
Back Projection	15.844818	1.186021
Histogram Specification	5.349486	0.556932
Total	24.000535	2.114135

Table 4-1: The comparison of execution times (approximately 10 times acceleration).

Chapter 5 Experimental Result

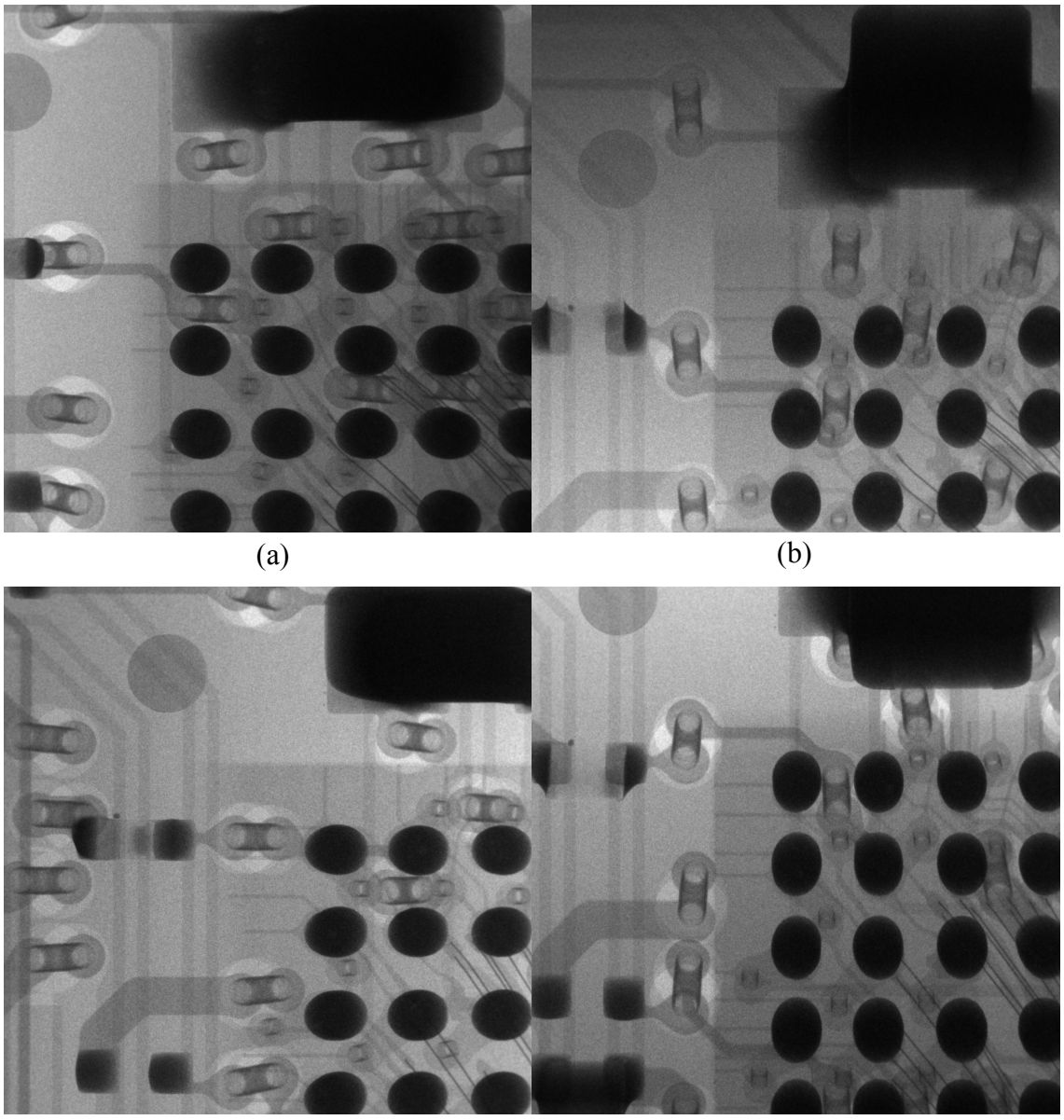
In our experiment, there are two samples: Sample1 contains 128 projection images with resolution 1496×1496 pixels reconstructing object with height 600 pixels, and Sample2 contains 16 projection images with resolution 1124×1000 pixels reconstructing object with height 200 pixels. We use Sample1 to compare with SART and FBP, and use Sample2 to compare with FBP and the results by Volume Graphics.

5.1 Comparison between SART and FBP

Experimental Environment1
1. CPU: Intel® Core i5 2.7GHz processor 2. Memory: 8GB 3. OS: OS X El Capitan 4. Programming Language: C/C++, OpenCV 2.4.9

Method	SART	FBP
Execution time (s)	291	210

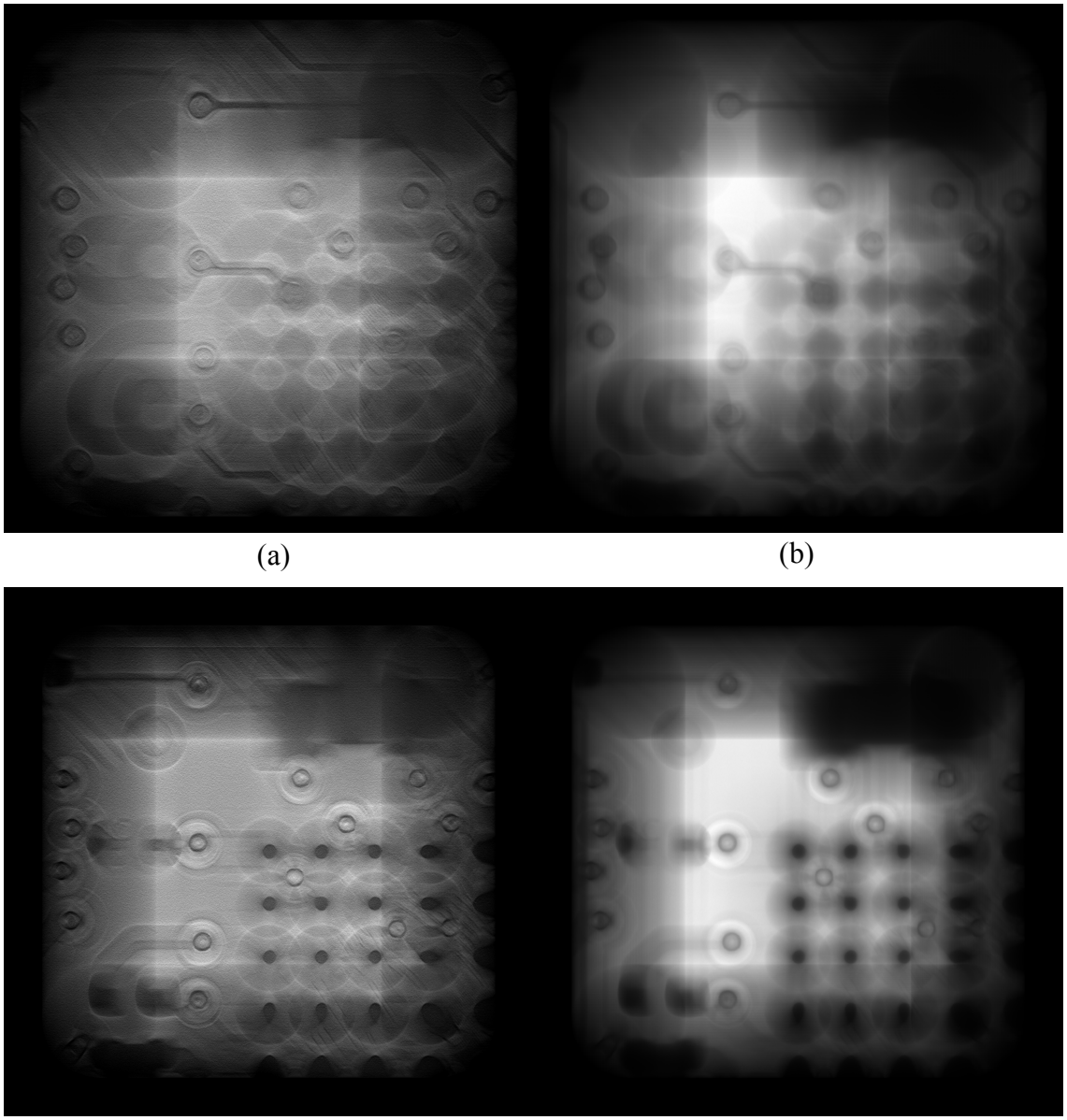
Table 5-1: Our execution time of SART and FBP.



(a) (b)

(c) (d)

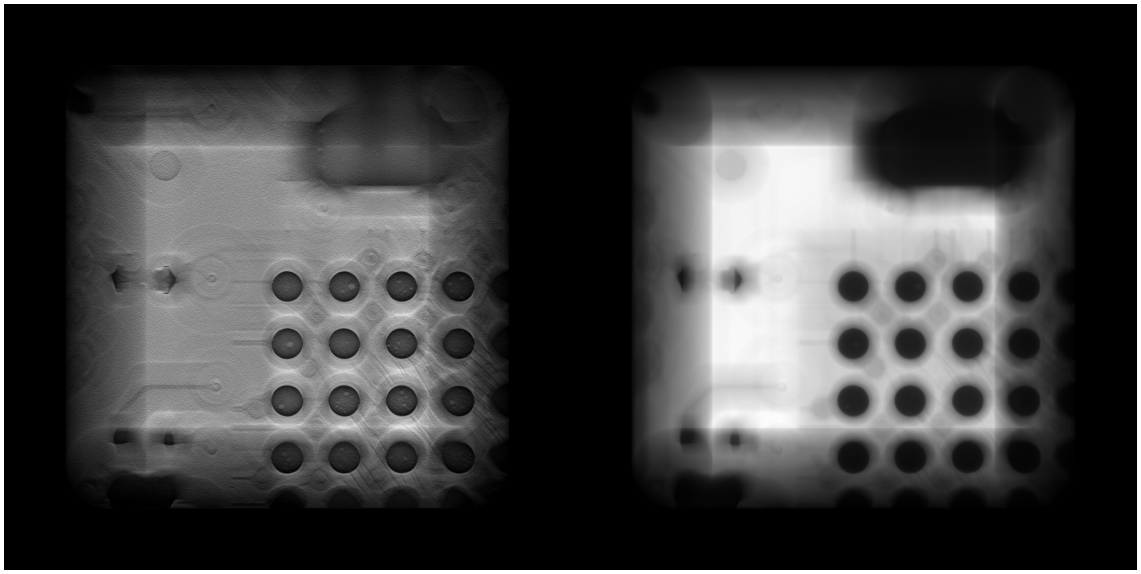
Figure 5-1: The projection images in Sample1 (totally 128 images). (a) The projection image with tilt angle 0°. (b) The projection image with tilt angle 90°. (c) The projection image with tilt angle 180°. (a) The projection image with tilt angle 270°.



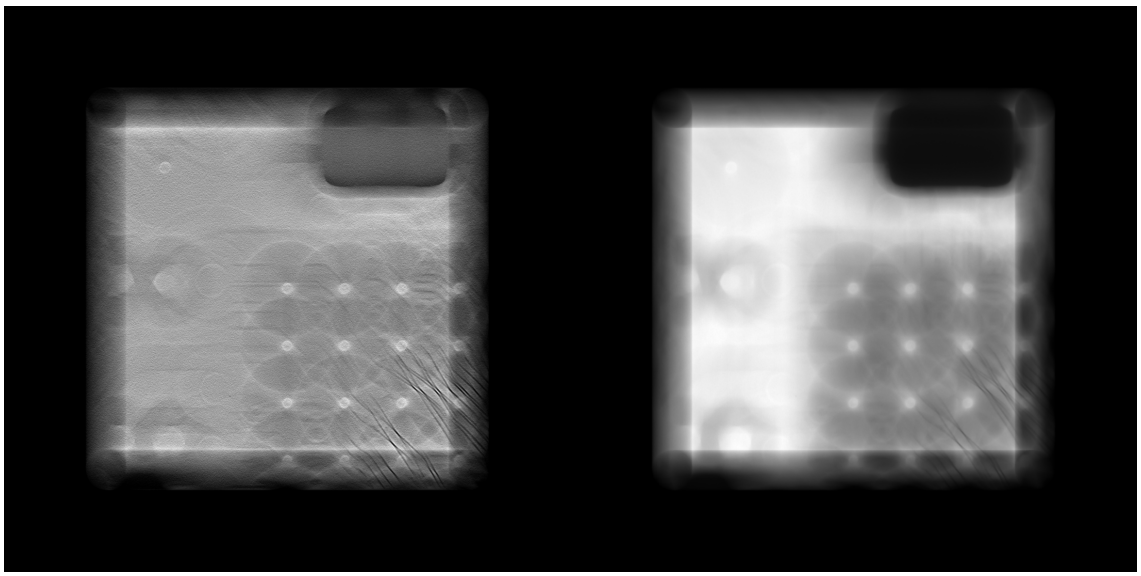
(a) (b)

(c) (d)

Figure 5-2: Our reconstructed images in Sample1 with SART and FBP. (a) The reconstructed image with height 100 pixels by FBP. (b) The reconstructed image with height 100 pixels by SART. (c) The reconstructed image with height 240 pixels by FBP. (d) The reconstructed image with height 240 pixels by SART.



(a) (b)



(c) (d)

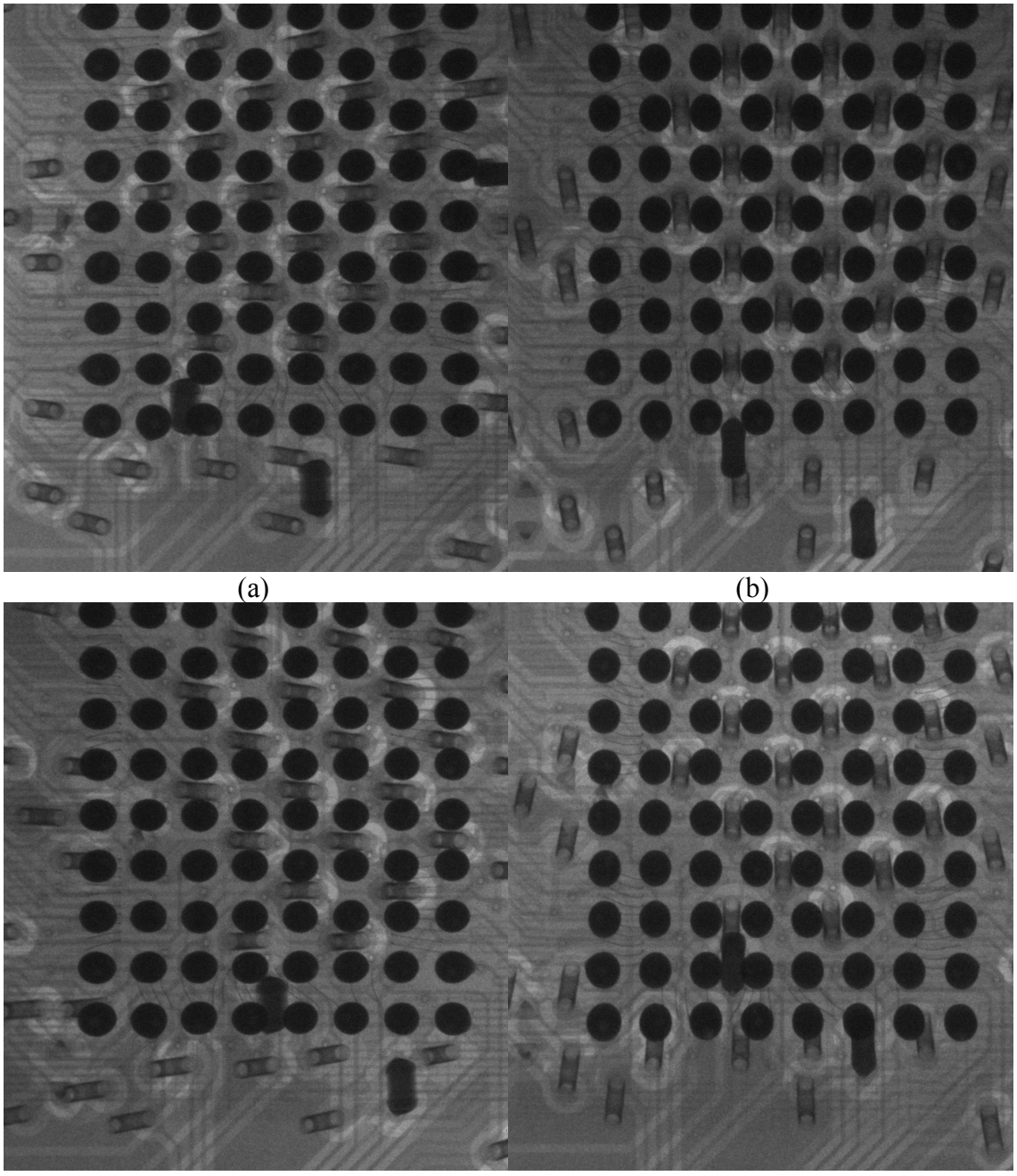
Figure 5-3: Our reconstructed images in Sample1 with SART and FBP. (a) The reconstructed image with height 360 pixels by FBP. (b) The reconstructed image with height 360 pixels by SART. (c) The reconstructed image with height 480 pixels by FBP. (d) The reconstructed image with height 480 pixels by SART.

5.2 Comparison FBP with Volume Graphics

Experimental Environment2	Our Environment	Environment of Test Research Incorporation
CPU	Intel® Xeon E5-2620 2.00GHz processor	Intel® E5-2687 2.53GHz processor
Memory	65GB	96GB
OS	Linux	Windows
Programming Language	C/C++, OpenCV 2.4.9	C/C++

	Our Result	Result from Volume Graphics
Execution time (s)	2.2	2.5

Table 5-2: Execution times of our result and Volume Graphics result.



(a) (b)

(c) (d)

Figure 5-4: The projection images in Sample2 (totally 16 images). (a) The projection image with tilt angle 0°. (b) The projection image with tilt angle 90°. (c) The projection image with tilt angle 180°. (a) The projection image with tilt angle 270°.

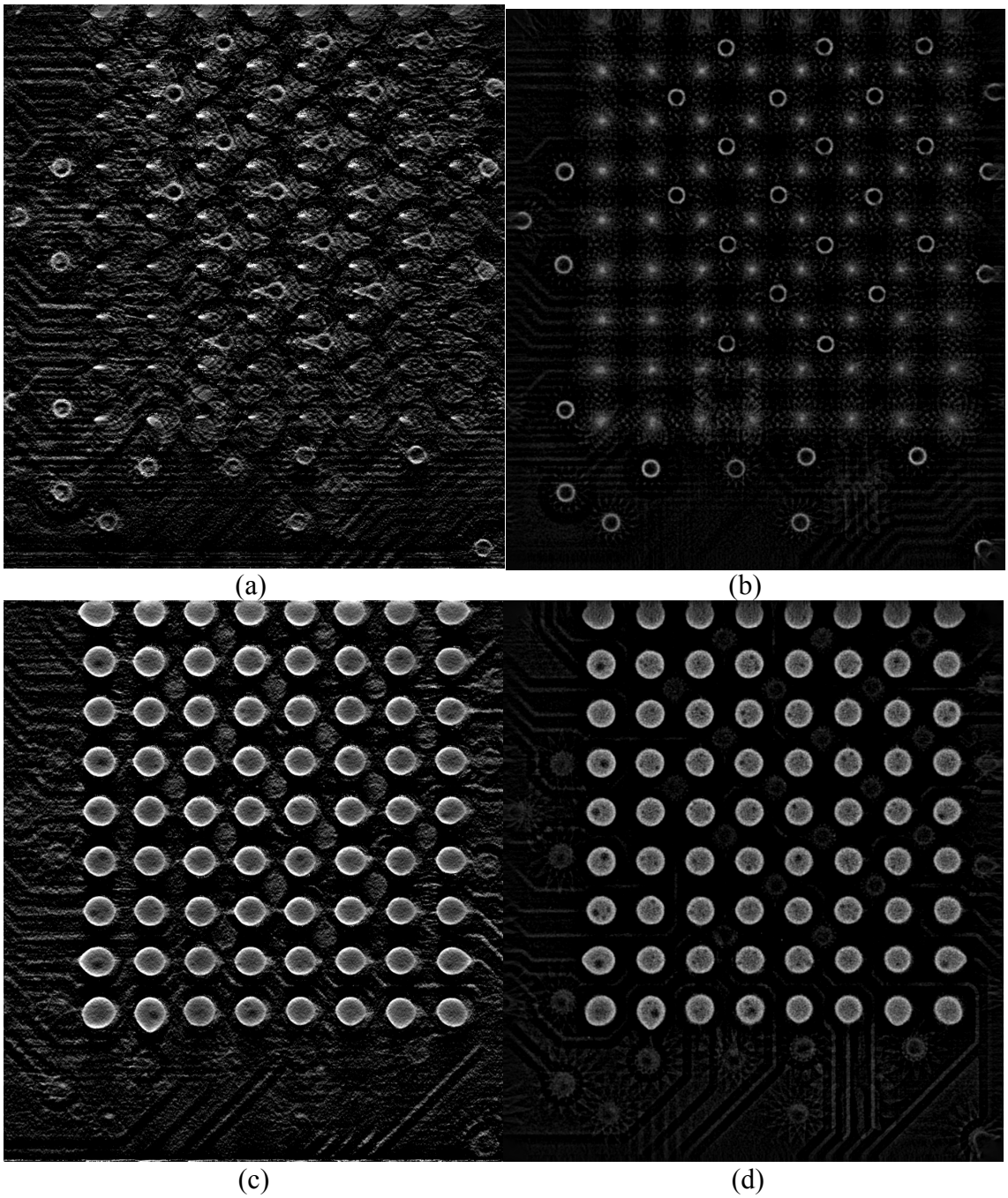
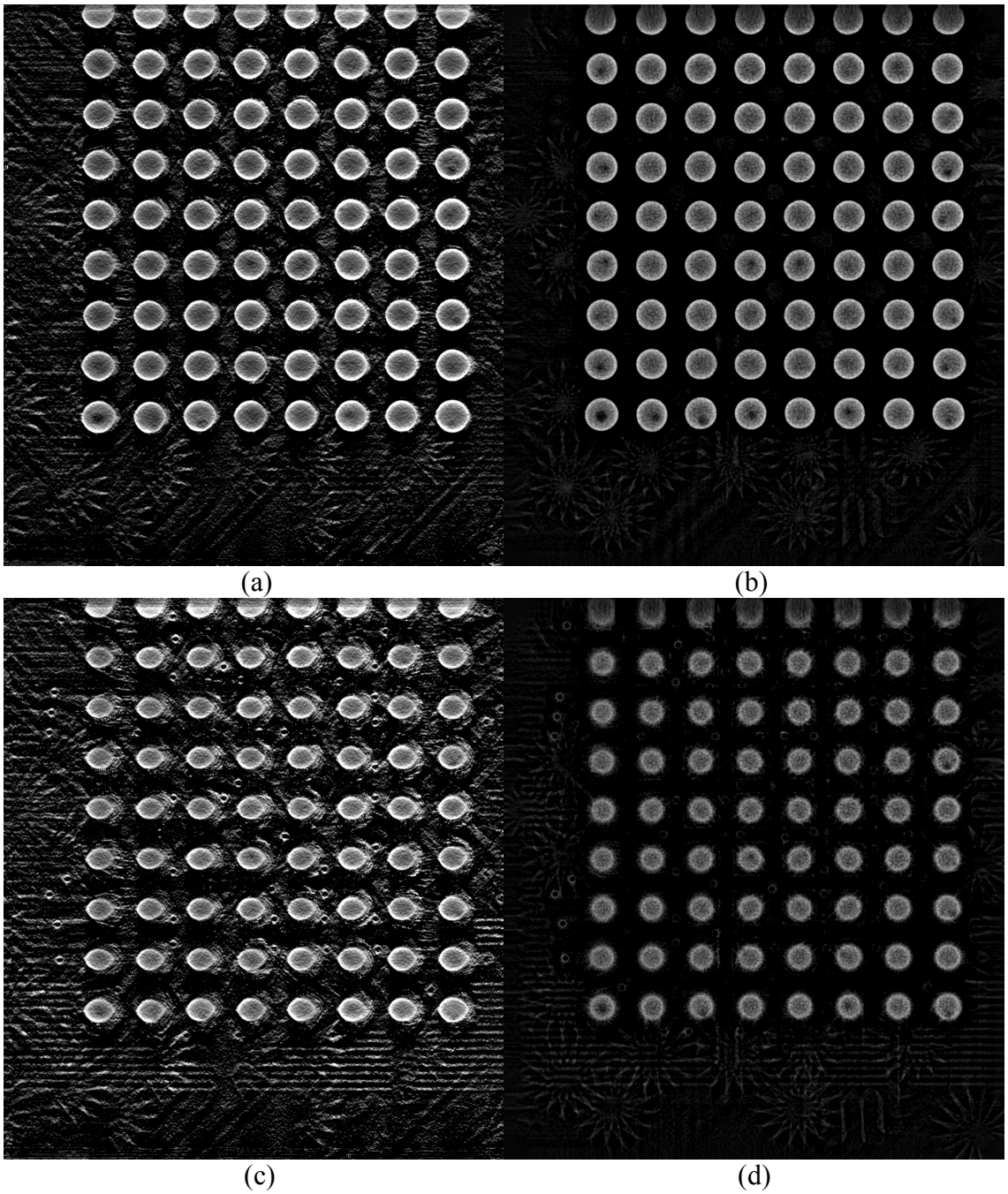


Figure 5-5: The reconstructed images in Sample2. (a) The FBP reconstructed image with height 85 pixels by our method. (b) The FBP reconstructed image with height 85 pixels by Volume Graphics. (c) The FBP reconstructed image with height 99 pixels by our method. (d) The FBP reconstructed image with height 99 pixels by Volume

Graphics.



(a) (b)

(c) (d)

Figure 5-6: The reconstructed images in Sample2. (a) The FBP reconstructed image with height 105 pixels by our method. (b) The FBP reconstructed image with height 105 pixels by Volume Graphics. (c) The FBP reconstructed image with height 112

pixels by our method. (d) The FBP reconstructed image with height 112 pixels by Volume Graphics.

Chapter 6 Conclusion and Future Work

In our work, we implement Simultaneous Algebraic Reconstruction Technique (SART) and Filtered Back Projection (FBP) with modified filter to reconstruct three-dimensional object with two-dimensional projection images. We also apply Threading Building Blocks (Intel® TBB) by Intel to accelerate our program and achieve approximately 10 times acceleration.

Although our execution time is slightly faster than Volume Graphics, it can be further accelerated by computing with Graphics Processing Unit (GPU), such as using Computed Unified Device Architecture (CUDA). However, our FBP reconstructed images have similar quality to Volume Graphics. For the void in solder ball, the low-contrast problem still exists in our result. This is the issue to resolve in the future.

References

[1]. R. Badawi, “Introduction to PET Physics,” http://depts.washington.edu/nucmed/IRL/pet_intro/toc.html, 2017.

[2]. L. Han, “Tools for 2-D Tomographic Reconstruction,” http://www.mathworks.com/matlabcentral/fileexchange/43008-tools-for-2-d-tomographic-reconstruction?s_tid=srchtitle, 2017.

[3]. Intel, “Introducing the Intel® Threading Building Blocks (Intel® TBB),” https://software.intel.com/en-us/node/506042#introducing_main, 2017.

[4]. H. K. Lin, “Algebraic Reconstruction Technique,” Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, 2016.

[5]. Y. Y. Lin, “The Study of Using Simultaneous Algebraic Reconstruction Technique on BGA Inspection,” http://140.113.39.130/cgi-bin/gs32/hugsweb.cgi?o=dnthucdr&s=id=%22GH009533591%22.&searchmode=basic, 2017.

[6] D. R. Nadeau, “C/C++ Tip: How to Loop through Multi-Dimensional Arrays Quickly,” http://nadeausoftware.com/articles/2012/06/c_c_tip_how_loop_through_multi_dimensional_arrays_quickly#Benchmarkresultsndashcompiledwithoptimizations, 2017.

[7]. C. Pan, “Image Projections and the Radon Transform,” https://www.clear.rice.edu/elec431/projects96/DSP/bpanalysis.html, 2017.

[8]. P. A. Penczek, “Fundamentals of three-dimensional reconstruction from projections,” https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3165033/, 2017.

[9]. N. Rezvani, “Iterative Reconstruction Algorithms for Polyenergetic X-Ray Computerized Tomography,” ftp://www.cs.toronto.edu/na/reports/Nargol.Rezvani.PhD.Thesis.pdf, 2017.

[10]. S.W. Smith, “The Scientist & Engineer's Guide to Digital Signal Processing,” http://www.dspguide.com/ch25/5.htm, 2017.

[11]. X. Wan, F. Zhang, and Z. Liu, “Modified Simultaneous Algebraic Reconstruction Technique and Its Parallelization in Cryo-electron Tomography,” http://ieeexplore.ieee.org/document/5395300/, 2017.

[12]. R. Wang, “Image Enhancement by Contrast Transform,” http://fourier.eng.hmc.edu/e161/lectures/contrast_transform/contrast_transform.html, 2017.

[13]. Wikipedia, “Algebraic Reconstruction Technique,” https://en.wikipedia.org/wiki/Algebraic_Reconstruction_Technique, 2017.

[14]. Wikipedia, “Projection-Slice Theorem,” https://en.wikipedia.org/wiki/Projection-slice_theorem, 2017.

[15]. Wikipedia, “Radon Transform,” https://en.wikipedia.org/wiki/Radon_transform, 2017.

[16]. Wikipedia, “Simultaneous Algebraic Reconstruction Technique,” https://en.wikipedia.org/wiki/Simultaneous_Algebraic_Reconstruction_Technique, 2017.

[17]. Wikipedia, “Window Function,” https://en.wikipedia.org/wiki/Window_function#Hamming_window, 2017.


國立臺灣大學
愛國愛學・勵品教人