ANALYTIC THREE DIMENSIONAL IMAGE RECONSTRUCTION FROM PROJECTIONS

By

Paul Eugene Kinahan

B.A.Sc., The University of British Columbia, 1985

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF PHYSICS

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

April 1988

3 9

Department of Physics

The University of British Columbia
Vancouver, Canada

Date 21 April 1988

Abstract

This work presents an analytic three dimensional image reconstruction algorithm that was developed for a proposed volume-imaging PET scanner. The development of the algorithm was motivated by the scanner's ability to collect an order of magnitude more data than current PET systems and the lack of an efficient algorithm that could use the extra data.

The algorithm is based on an extension of the Recovery Operator of Orlov[68] and operates by convolution in object space. This method of operation sets it apart from other analytic direct image reconstruction algorithms that rely on Fourier transforms.

The algorithm is tested with ideal data and parameters that are appropriate to the new PET scanner. The results of the test show that the algorithm behaves as expected except for a 17% overshoot in the reconstructed value in one area. An explanation of this artifact is suggested, although not verified.

Finally, the efficacy of the algorithm is demonstrated by proving that it is functionally equivalent to Fourier transform methods.

Contents

List of Figures

Acknowledgements

I acknowledge the following people whose contributions have made this thesis better than what it would otherwise be.

Dr. Joel G. Rogers, my research supervisor, who provided me with everthing that a graduate student could need. It is his knowledge and patient guidance that have made this work possible.

Dr. Richard R. Johnson, my faculty supervisor, who has supplied me with valuable encouragement and advice on many matters.

Dr. Ronald Harrop, who has followed my work carefully, and made important suggestions and corrections.

Drs. Tom Ruth and Wayne Martin, who cheerfully answered my questions about the rest of PET.

My Friends of Mickey Mouse; Chris Backhouse, Connie Baxter, Fraser Duncan, Shelli and Willie Fajber, Marcel Lefrancois, and Dave MacQuistan, who have helped me with many hours of often painless diversions.

My co-workers in the UBC/TRIUMF PET group; Ed Grochowski, Ken Pederson, Gord Riley, and Neil Wilkinson. These guys helped make work intresting.

The Science Council of British Columbia, which has provided me with financial support for two years.

My Mother and Father, whose love, support, and encouragement make everything possible.

Chapter 1

Basics of PET

1.1 Introduction

Positron Emission Tomography (PET)[100] is an imaging methodology primarily used in nuclear medicine. It is a system that can determine the position and amount of a radioactive tracer present at each location within a solid object. A PET system can determine the location and amount of tracer material an order of magnitude more accurately than other nuclear medicine imaging systems. This is done by taking advantage of the properties of the particular radioactive tracers used.

Tomography (from the Greek: *tomo* - slice) is a branch of the field of image reconstruction. Image reconstruction is common to many disciplines including radio and solar astronomy, electron microscropy, industrial nondestructive testing, radar and sonar systems, and several types of medical scanners. The common basis these fields have is the use of tomography to deduce information about the internal nature of an object solely from external views, called projections, of the object.

In nuclear medicine, image reconstruction as used by PET allows the study of in vivo biochemical reaction processes in objects as complicated as the human brain. PET systems are currently only used as research tools to study (mainly human) biochemistry, but there is a large amount of investigation into the possible roles of a PET scanner as a diagnostic aid for diseases such as Alzheimer's [62], Huntington's [41], and Parkinson's [12].

The potential of PET stems from its ability to determine internal function as opposed to internal structure. Structure can be measured with an X-ray Computerized Axial Tomography (CAT) scanner or a Nuclear Magnetic Resonance (NMR) scanner, but neither of these methodologies can determine what internal processes are occuring.

Although the next three sections give some of the necessary background for the rest of this thesis, the best general introduction to PET is the *Scientific American* article by Ter-Pogossian et al. [100]. More detailed introductions are the works by Sorenson and Phelps [93] and Phelps and Mazziotta [74].

1.2 Physical Processes of PET

The basic process of PET is the decay of a radioactive isotope by the emmision of a positron, and the subsequent mutual annihilation of the positron and an electron. The positron, also called a β^+ particle, can be considered to be a positively charged electron, a β^- particle. When the positron and the electron are annihilated, they are converted into energy in the form of two 511 kilo electron-Volt (keV) gamma rays that travel away at 180° to each other. This reaction is symbolized by,

$$\beta^+ + \beta^- \longrightarrow 2\gamma$$

The positron comes from the decay of specific radioisotopes such as ^{18}F, ^{15}O, ^{13}N, and ^{11}C, with half-lives ranging from minutes to hours. These radioactive isotopes are used to label compounds by replacing either naturally occuring isotopes or other atoms. The purpose of the PET system is to determine the amount and location of $\beta^+ + \beta^-$ annihilations, and this information is then interpreted as being the same for the labelled compound. For example, the compound ^{18}FDG is a labelled analogue of glucose that undergoes a metabolism in the brain similar to that of glucose. Since glucose is a metabolic fuel, by determining the distribution of ^{18}FDG within a brain, researchers can investigate metabolic activity in different regions on the brain [93,92,91].

This method, called tracer kinetic studies, has been used with other nuclear medicine systems, but to a much lesser degree of accuracy. The key process that sets PET apart is that the two gamma rays from the annihilation travel away in opposite directions along a 'line of interaction'. These photons can be detected in coincidence by devices around the annihilation point in a PET scanner detector as shown in figure 1.1. Note that in figure 1.1 the size of the nuclei and the positron, and the distances they travel, are greatly exagerated relative to the size of the detector circle. By using its two detected endpoints, the line of interaction can be measured more accurately and efficiently than with other nuclear medicine scanners.

Since the gamma ray photons are travelling at the speed of light, they will arrive at the detector within $\sim 10^{-9}$ sec. of each other. This coincidence detection allows the separation of different annihilation events in time. By detecting enough events it is possible to measure the location and quantity of positron emitters.

Although the fundamental process used by PET is straightforward, the PET systems themselves are not. Currently there are approximately 60 PET scanners with no two being exactly alike, with many quite different in method of operation and appearance. It is possible, though, to give a description of a typical system, and some common variations.

A standard system would consist of one or more rings of crystals capable of detecting the annihilation photons. The rings are stacked to form a cylindrical volume to enable detection of gamma rays from a three dimensional object inside the rings. The crystals, such as sodium iodide (NaI) or bismuth germinate (BGO), work by absorb-

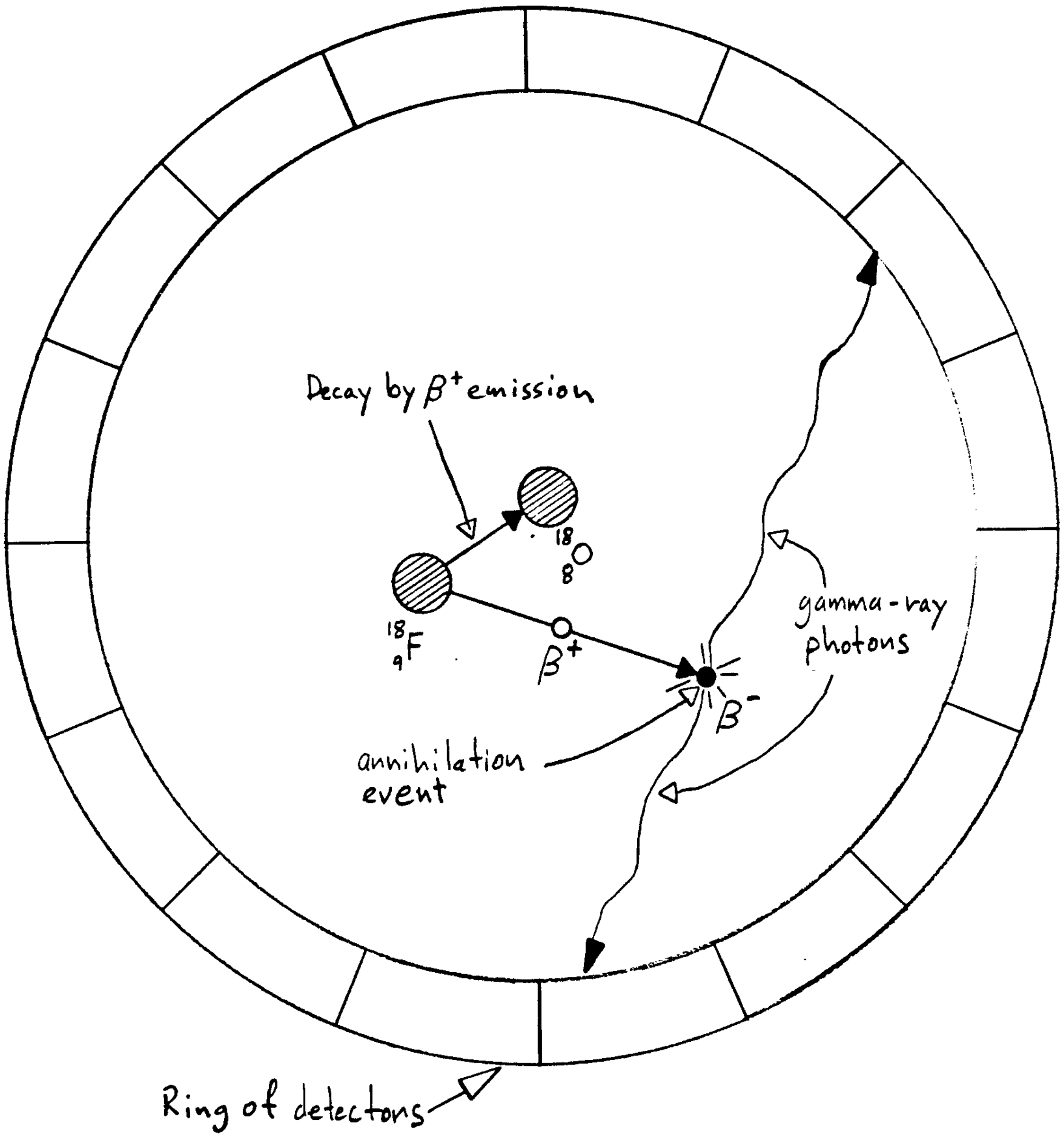

Figure 1.1: Basic physical processes of PET (not to scale)

ing the single 511 keV photon and then releasing the energy as a shower of thousands of secondary) photons [64,35,73]. Each detector ring can consist of as many as 1200 crystals [11] or as few as 6 [65]. As shown in figure 1.2 the secondary photons pass through some form of optical coupling, such as lucite, to the surface of a series of photo multiplier tubes (PMTs). The entire assembly is shielded from outside visible or UV light because of the sensitivity of the PMTs. Usually, the surfaces of the crystal not optically coupled to anything are covered with a reflective material in order to redirect more secondary photons to the PMTs. The PMTs can detect individual secondary photons and convert the energy of the photons to a series of electrical pulses that identify which crystal absorbed the original gamma ray. Some PET systems have been built around two rotating parallel plates that sweep out a cylindrical volume [5,2]. The plates are Multi Wire Proportional Chambers (MWPCs) in which the gamma ray causes an avalanche breakdown in a gas, and the avalanche charges are picked up by a series of closely spaced wires.

The electrical signals from the detector subsystem are analyzed to determine coincidence in time of two gamma rays, and the resulting information is stored in a computer as a single coincidence event. In addition some machines attempt to measure the difference in arrival times between the the two gamma rays in order to localize the annihilation point on the line of interaction [64,35,92]. Such Time-Of-Flight (TOF) systems have had mixed results because of the added complexity both of measuring and incorporating the TOF information. With the stored information of coincidence events (typically 100,000 to 10,000,000 events) it is possible to recover the internal distribution of positron emitters, and thus the radioactive tracer distribution. This will be shown in chapter 3.

The above describes the role that a PET system fulfills within the overall framework of PET studies. The two other major areas of a PET study are the preparation of

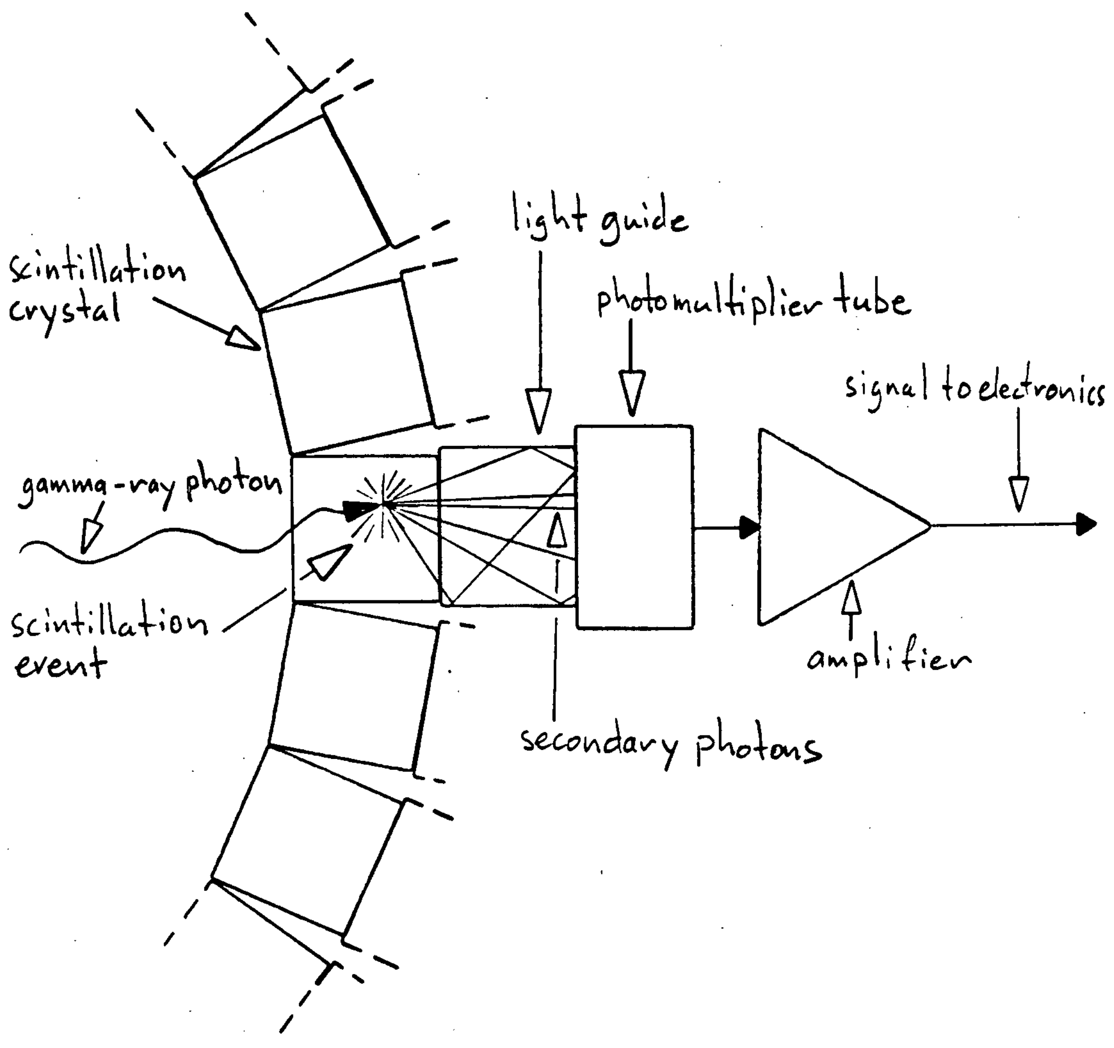

Figure 1.2: Portion of ring of detectors of typical PET system

the labelled compounds, which requires a cyclotron [106,39], and the mathematical modelling of the compound's reaction sequence in order to analyze the data from the PET scanner [47,61]. All three of these components are necessary to perform a nuclear medicine PET study [74].

This thesis is concerned with the last step in the PET scan phase of the PET study. This is the step where all the coincidence events that are stored in the computer are transformed into the reconstructed internal distribution of positron emitters - this is the process of image reconstruction.

1.3 Introduction to Image Reconstruction on PET scanners

This section gives a brief introduction to image reconstruction as it applies to PET. The theory of three dimensional image reconstruction will be covered in excruciating detail in chapters 3 to 5. Image reconstruction is a branch of applied mathematics that deals with recovering the internal details of an object from projections through the object. Projections are the integrations of internal distributions of some physical quantity on a straight line through an object. When the reconstruction is performed by a computer, the general process is sometimes called Computerized Tomography (CT). In PET scanners, projections are formed from the stored lists of coincidence events, and a computer performs an image reconstruction to recover the internal tracer distribution. This is akin to determining the structure of a three dimensional chinese puzzle inside a locked cupboard by taking a series of X-ray photographs of the cupboard at different angles. From this point on, the lists of coincidence events will be referred to as projection data or, simply, projections.

The problem of analytically determining a function from its projections was first solved in 1917 by Johan Radon [78]. Unfortunately his work was overlooked until after the same problem had been re-solved by others almost 60 years later. By the mid-1960's several electron microscopists had realized that they could determine the internal structure of small objects by using Fourier transforms of the projections [30,79]. Since this was before Fast Fourier Transforms (FFTs) were used, the transforms were time-consuming to compute and were only useful for objects with high degrees of rotational symmetry, such as bacteriophages.

In 1967 Bracewell, a radio astronomer, showed how to perform an exact image reconstruction from projections without resorting to Fourier transforms [8]. This work was soon duplicated by others[1] such as Berry and Gibbs [6]. Since then the study of image reconstruction has been driven by several practical applications, resulting in its establishment as a separate field of study.

The original tomographs developed for medicine employed X-rays to image a two dimensional plane, or slice, perpendicular to the long axis of the patient [9]. The image reconstructed from the collected projection data gave a picture of the internal distribution of tissue and bone within the slice. A series of slices next to each other were then used to build a picture of the three dimensional tissue and bone distribution inside the patient. For PET systems the same method of successive slices is used to build a three dimensional picture of tracer distribution. The major difference between PET and CAT scanners is that CAT scanners transmit X-rays through an object and look at the amount absorbed, whereas PET scanners look at the emission of gamma rays from sources inside the object.

PET system designers are constantly trying to improve resolution and sensitivity [64,35,99]. A broad definition of resolution is the ability to distinguish, or resolve, two

[1]This re-inventing of the wheel was a common occurence in the early days of CT, possibly due in part to the interdisciplinary nature of the field.

or more objects from a single larger object; so a system with a better resolution can distinguish separate points placed more closely together. Sensitivity can be thought of as the systems ability to receive data; thus a system with a higher sensitivity can detect more events in a fixed time period. The first few generations of PET scanners used two dimensional image reconstruction for each slice, but as the increasing experience of scanner designers has led to improved resolution and sensitivity, it has been realized that it is useful to aquire data that cannot be used by two dimensional reconstruction theory. A PET system that shows the need for direct three dimensional image reconstruction is being developed at TRIUMF[2]. This will be discussed in more detail in the next chapter.

1.4 Limitations of PET

Since it is important not to lose sight of the overall picture, several of the most important limiting factors will be listed in this section. This will be done in the order that information passes through a typical PET system as described in section 1.1, starting with the tracer decay by positron emission and ending with the display of the reconstructed image.

As the labelled compound decays, the positron is ejected from the isotope nucleus with a non-zero velocity [93]. This means that the positron will combine with an electron at a varying distance from the actual tracer. This positron 'wander' has a gaussian distribution about its emission point from the tracer that has a Full Width Half Maximum (FWHM) of 1 mm to 5 mm depending on the labelled compound. An added effect of the positron's velocity is that the two annihilation gamma rays vary in the angle that they travel apart. The separation angle has a gaussian variation about

[2]The TRI-University Meson Facility located in Vancouver B.C. Canada.

180° with a FWHM of $\sim 0.5°$. Both of these effects contribute to a fundamental limit of how accurately the line of interaction can be measured [64,35,33].

After the emission and before the detection of the two 511 keV gamma rays, the object itself can degrade the accuracy of measurement by its interaction with the gamma rays. This influence is mainly through self absorption and Compton scatter of the gamma rays by the object. Self absorption occurs when one or both of the gamma rays is absorbed by the object. A manifestation of this is seen in the centres of strong self-absorbers where the measured activity is lower than the true activity. If one or both of the gamma rays is scattered within the object but still detected, then an incorrect line of interaction is recorded. Both scatter and self-absorption can be corrected for with varying degrees of success [93,64,48,38,16,102].

At the stage where the gamma rays are detected by the PET scanner, the limiting factors become entirely machine dependent, with the most important factors being detection efficiency, intrinsic resolution, and detector dead time [35]. Detector efficiency refers to the probability that a gamma ray passing through a detector will interact with the detector and thus be observed. The probability of interaction can vary from about 10% for MWPC based detectors to 95% for systems with large BGO crystals [64,35,99,34]. The intrinsic detector resolution determines how accurately the point of interaction between the gamma ray and the detector can be known, and varies from about 1 mm to 15 mm. Usually one scintillation crystal is coupled to one PMT so that by looking at the outputs of the PMTs it is possible to find out which crystal absorbed the gamma ray, so in this method the intrinsic resolution is the crystal size. This accuracy can also be a function of several other factors, such as the amount of light from the scintallation and how much of it makes it to the PMT, and the efficiency of the PMT in converting the secondary UV photons into electrical signals [43]. These signals are processed by the detector electronics, and the processing of each event causes the detector to be shut down for a period known as the dead time, during which no other events can be detected [60,59]. The two dominant factors in system dead time are the time taken to reject single[3] events, and the time taken to process a coincidence event to the point where the information can be transfered to the computer. Additionally, the detector electronics cannot always differentiate between true coincidence events and random coincidences, where a random coincidence is a detection of two single events within the coincidence time window [64,14,73]. This is an error, since the two single events are the results of annihilations that are close in time but not necessarily in space.

Mechanical structure affects performance in many, often subtle, ways. The more obvious ways are the physical positioning of detector units, geometrical efficiency, and shielding. This shielding, which is usually lead or iron to efficiently absorb gamma rays, is used to reduce the detection of unwanted events [64,93,73,58]. These events are either gamma rays originating outside the scanners Field Of View (FOV), internally scattered gamma rays, or events that supply information that cannot be used in the image reconstruction process. This last case, and geometrical efficiency, will be discussed further in the next chapter. In addition, shielding can scatter gamma rays similar to the prompt scatter caused by the object.

Once the detected events are stored as projections, the factors affecting performance are quantization and statistics [93,18]. In the mathematics of analytic image reconstruction the projections are approximated as true integrals of activity along lines through the object. The data is actually a set of sums of individual events and the accuracy of the approximation depends on the number of events collected. Since the fundamental process is nuclear decay, which has Poisson statistics, the error of approx-

[3]Single events are those where a second gamma ray was not detected within the allowed coincidence time window. The ratio of single to coincidence events ranges from 50:1 to 100:1.

imation is proportional to $1/\sqrt{n}$, where n is the number of events. The error can be reduced by scanning for longer periods, but this may not be possible because of patient considerations or isotope decay. Increasing sensitivity always improves statistics, but may also increase the amount of scattered gamma rays that are detected, as will be shown in section 2.1.

Quantization is the subdivision of a continous variable into a series of discrete subunits, and always degrades accuracy [93,46,57]. It is necessary in a PET system because the image reconstruction is performed by a computer that can only work with discrete quantities. Quantization shows itself in several places; first in the discretization of the projection data. If the smallest measurable region contains an even smaller strong point source, then the activity of the point source will be averaged over the entire region, thus giving a false lower value. This is known as the partial volume effect. The second instance of quantization error is in the manner that the data is manipulated during the image reconstruction process. Finally, the reconstructed image can be presented in different degrees of quantization. Although quantization can usually be increased to the point where it no longer limits performance, this is often not done because of the consequent increases in needed computer memory and computation time. The effects of quantization on the accuracy of image reconstruction will be discussed in section 4.2, but analyzing the effects of the other limiting factors on the reconstructed image is beyond the scope of this work.

Chapter 2

Three Dimensional PET

Some of the concepts introduced in the previous chapter will be used in this chapter to show the advantages of scanning an entire volume at once instead of scanning an object as a series of separate planes. Some of the problems of volume imaging and their solutions will also be discussed. Since three dimensional imaging leads to three dimensional image reconstruction theory, the chapter ends with a survey of current three dimensional reconstruction algorithms.

2.1 The Case for Volume Imaging Systems

Standard two dimensional image reconstruction algorithms only use projection data that all lie in a single plane [9,84,3,52,87,105,45,44]. Partly for this reason the current generation of PET scanners form three dimensional images by reconstructing a series of adjacent slices and then stacking the planes together to form a volume image [10,20,19]. The separate planes, or slices, are called transaxial slices since they are perpendicular to the long axis of the patient. Cross-plane events are those in which the two gamma rays are detected in different detector planes. In other words, cross-plane data cannot be assigned to a single detector plane and thus cannot be used by standard two dimensional image reconstruction algorithms.

To keep detectors from picking up cross-plane events, annular shielding rings called inter-slice septa are placed between adjacent detector rings as shown in figure 2.1 [73]. Note that the detector surface has cylindrical symmetry about the centre line. Septa are also used within a slice to absorb gamma rays that could introduce parallax error [64]. This error arises when scintillation crystals are deep in comparison to the desired intrinsic resolution as shown in figure 2.2. Since the interaction point within the crystal is not known, there can be a large error in the calculated line of interaction for a gamma ray that arrives at an oblique angle to the crystal. This problem can be solved by determining the depth of interaction of the gamma ray in the crystal or by making the crystal shallow in comparison to the intrinsic resolution. The latter method also greatly reduces the probability of interaction between the crystal and the gamma ray and thus reduces overall system sensitivity.

There is often an exception made to the rule of not using cross-plane data. This is the practice of using data that is detected on two adjacent planes to reconstruct a plane lying midway between the two detector planes [73]. This type of image is called an inter-slice plane as opposed to a true plane[1]. Inter-slice planes do increase the number of planes, and thus the axial intrinsic resolution, but the sensitivity, resolution, and FOV are usually dramatically different from those of true planes. This can lead to substantial inaccuracies in a three dimensional image formed by stacking an alternating series of true and inter-slice planes.

In a multi-ring scanner the only other method of improving the axial resolution is to make the crystals smaller in the axial direction. Referring to figure 2.1, using smaller crytals is equivalent to bringing the inter-slice septa closer together. This decreases the total volume in the scanner's FOV unless more detector rings are added, consequently increasing complexity and cost. Additionally, experience with PET scan-

[1]Formed from events detected by a single detector ring.

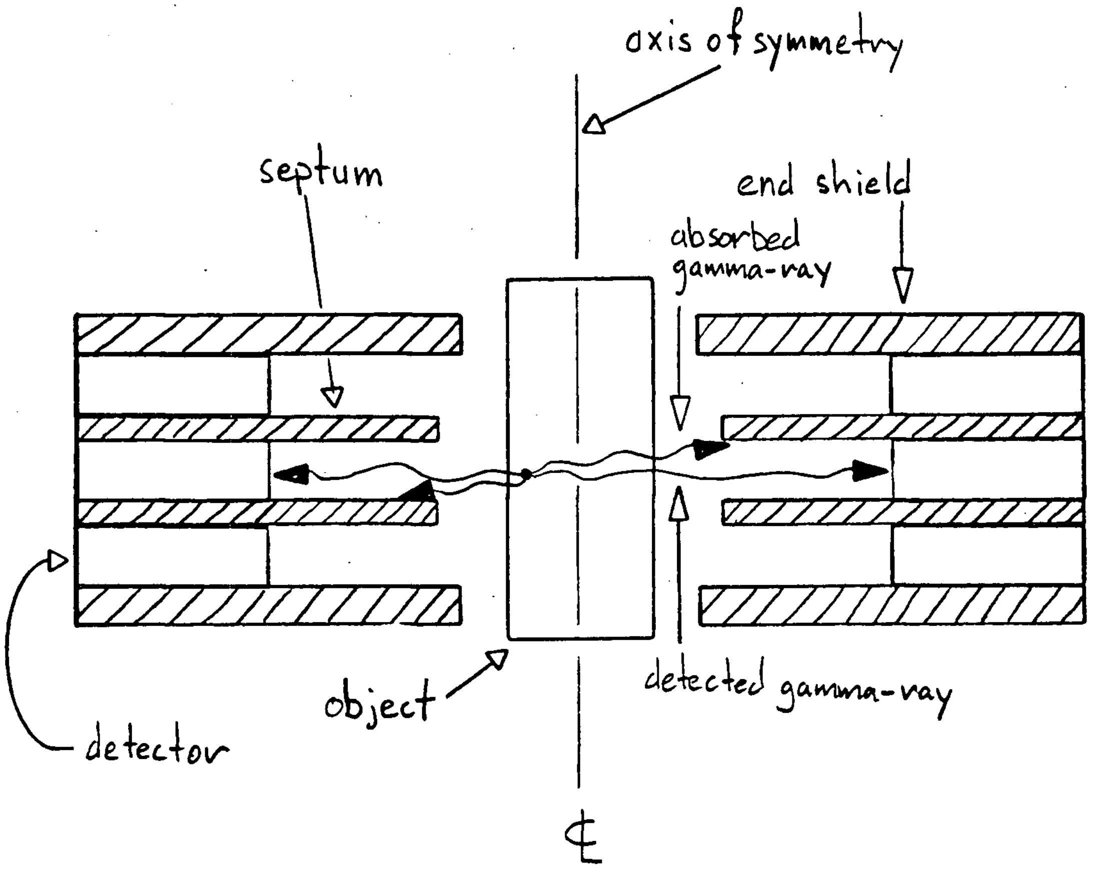

Figure 2.1: Cross section showing shielding for a planar imaging detector.

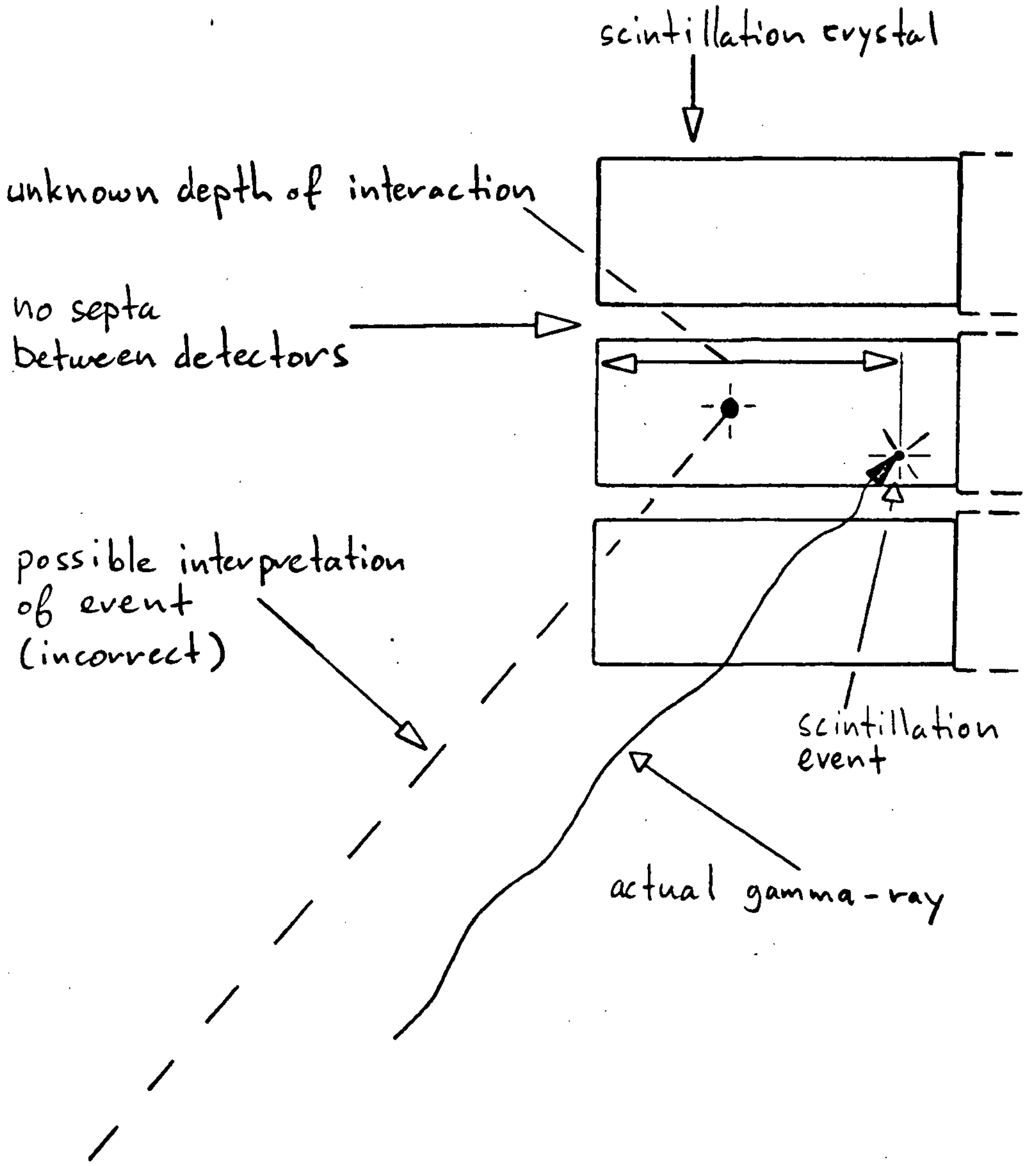
scintillation crystal
unknown depth of interaction
no septa between detectors
possible interpretation of event (incorrect)
scintillation event
actual gamma-ray

Figure 2.2: Illustration of parallax error

ners has led to the general rule that improving intrinsic resolution by using smaller discrete crystal/PMT detector units leads to reduced system sensitivity.

An illustration of this trade-off can be seen by comparing two prototype single ring PET scanners. The Donner 600 scanner [36] has a best case resolution of 2.6mm and a corresponding sensitivity of 7100 events/sec per μCi/ml of activity in a standard cylindrical flood phantom. Compare this to the PCR-I scanner [96] that has a best case resolution of 4.5mm and a sensitivity of 45,000 events/sec per μCi/ml of activity.

The way to escape this bottleneck and increase both resolution and sensitivity is to remove the inter-slice septa. This increases the solid angle of coverage inside the detector and leads to an increase in the scanner's geometric efficiency [64,73,2]. The geometric efficiency measures how well a scanner can receive data from a point source moved everywhere inside the detectors FOV. By removing septa, system designers can keep resolutions of $\sim$ 5mm and increase sensitivity 5 to 20 times [24], but there are three problems that immediately arise[2]: parallax error, increased scatter, and how to incorporate the additional data into the image reconstruction algorithm.

The first problem has been solved by a new PET detector unit developed at TRIUMF [83,80,82]. This unit uses a large ($15 \times 15 \times 2.5$ cm) NaI crystal that is optically coupled to a position sensitive PMT. By analyzing the secondary photon pattern the three dimensional interaction point of a gamma ray inside the crystal can be located. This obviates the need for using separate rings and septa, and a complete scanner is being developed by arranging 9 of these units in a cylindrical pattern as shown in figure 2.3[3]. An additional important benefit is that intrinsic resolution is now the same in all directions, since the interaction point is now measured as a continuous variable as opposed to a discrete variable.

[2]Which is why the septa were there in the first place

[3]Figure 2.3 is courtesy of R. Moore, TRIUMF.

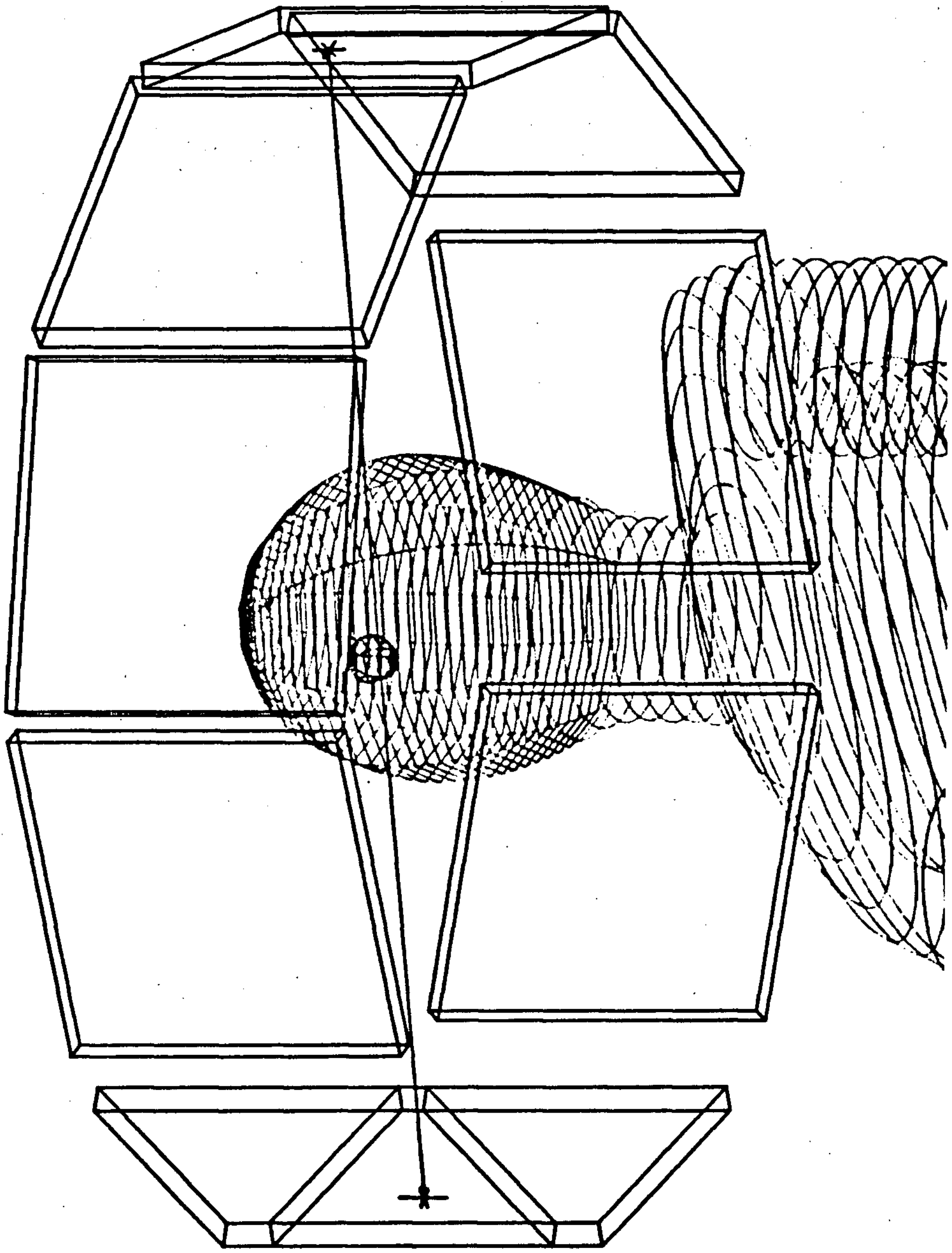

Figure 2.3: Detector geometry of TRIUMF PET scanner.

This ability to measure data that has not been directionally selected is an important one. The human brain and other organs are complicated three dimensional structures and are not designed to to be conveniently looked at in a series of planes. Any data collection system should be able to produce reconstructed images that have equal sensitivity and resolution in all directions.

It is known that the amount of scattered events that are detected increases faster than the rate of detection of true coincidences as the geometrical efficiency increases [73]. This has been verified by recent Monte Carlo simulations [82,94,101] of the effects of increasing geometric efficiency. Since scatter is largely featureless for extended sources, gains in statitstics are more important than reducing scatter. So the net effect of increasing the scanner's total solid angle is still an improvement in the signal to noise ratio.

The last problem of being able to use all the additional cross-plane data is the subject of this thesis.

2.2 Previous Work in Three Dimensional Image Reconstruction

Direct three dimensional image reconstruction differs from indirect reconstruction in that the three dimensional volume image is formed all at once, instead of as a stacked series of separate two dimensional image reconstructions.

An attempt to use cross-plane data in an indirect three dimensional image reconstruction has been made by Daube-Witherspoon and Muehllenher [28]. The basis of the method is to assign the cross-plane event to the plane midway between the two planes in which the gamma rays were detected, or to average the event information between all the intervening image planes. After this a standard image reconstruction is performed on the image data. Tests of this method have shown that even though statistical quality is improved, artifacts can be present in the final three dimensional image [28].

Several of the most common methods of direct three dimensional image reconstruction fall under the heading of iterative reconstruction techniques. Algorithms such as the Image Space Reconstruction Algorithm [27,103,29], also developed by Daube-Witherspoon and Muehllenher, and other series expansion methods [55,15,69,49,4] essentially all rely on many iterations of sucessively more accurate reconstructions. This is different from analytic reconstruction methods that perform a complete image reconstruction in one operation where no guess work is required. In general, these iterative algorithms work by trying to estimate an object that will give rise to the observed projections. The differences between the observed projections and projections formed from the estimated object are calculated and are used to modify the estimated object, at which point the process is repeated. The iterations of modifying the estimated object are stopped when the differences between the sets of projections are less than some pre-determined amount. These algorithms differ mainly in their method of using the differences between the projections as a guide for modifying the estimated object.

The advantages of iterative reconstruction methods are that they can use cross-plane data and a priori knowledge of the object, and in some cases they have low noise generation characteristics. Two relatively minor disadvantages are that edge artifacts and random noise form in the reconstructed image at high iteration counts [99], and that it is often difficult to know when to stop the iteration process. The major disadvantage is the computation time required for an image reconstruction. There are two dimensional iterative image reconstruction algorithms that do not require inordinate amounts of computation time [99,98,53,88,89,50,56], but three dimensional implementations of these algorithms require many hours of computation for a single image [27]. This amount of time for an image reconstruction is not acceptable, as a PET system can be expected to scan up to four times a day [85].

The slowness of the iterative algorithms combined with the advantages of using cross-plane data has resulted in efforts to develop an analytic three dimensional image reconstruction algorithm. Some of these efforts have been directed towards the cone-beam data collection method tried on CAT scanners [32,66,72,90], but these have been based on algorithms developed for the parallel-beamdata collection method characteristic of PET. These methods rely on the use of Fourier transforms [97,70,21,104,95,26,77,71,22,86,76]. These algorithms will be outlined in the next chapter, but the point to be made here is that in general any analytic reconstruction method is much faster than an iterative one.

Fourier transform techniques have the drawback that since a computer implementation requires the use of Discrete Fourier Transforms (DFTs), they also suffer from the side effects of DFTs. These effects are poor noise characteristics and the possibility of aliasing artifacts. Additionally, while Fourier methods can use some of the cross-plane data, they cannot use all of the available data. The amount of useable data depends on the relative geometries of the object and the PET scanner system.

Two new methods of three dimensional image reconstruction that are promising have been devised with specific detector geometries in mind. Both are intended for use with two rotating parallel MWPC detectors and both can use most of the detected data. The method of Defrise et al. [31] is essentially a multi-pass Fourier transform technique similar to that of Cho et al. [21] that requires more time than a single analytic image reconstruction, but should be considerably faster than a true iterative method. The only algorithm to date that uses all projection data in a single pass is that of Clack et al. [23]. This last method is based on a series of two dimensional image reconstructions at different angles, and is the most promising of the alternative methods surveyed.

The final method we will discuss, and the one on which this work is based, is the generalized analytic Recovery Operator of Orlov [67,68]. The recovery operator, which will be explained in the next chapter, is an analytic reconstruction technique that has the same requirement of an invariant detector PSF, but does not rely on DFTs. That is, it works entirely in object, or image, space as opposed to frequency space. It will be shown in chapter 3 how the recovery operator can be adapted to a PET scanner system.

Chapter 3

Reconstruction Algorithm

3.1 Analytic Form

Before developing a solution to the problem of direct three dimensional image reconstruction, it is necessary to describe the problem mathematically. The notation used is based on that of Orlov [67,68] and Denton et al. [32].

3.1.1 Mathematical Description of the Problem

An object of interest has an unknown internal distribution of positron emitting radioactive tracer. This distribution is completely described by the function $\rho(\vec{x})$, where $\vec{x}$ is a three dimensional vector. As mentioned above, the projection data, $\rho(\vec{x}_\tau;\hat{\tau})$[1], are approximated as the integrals of tracer activity along a straight line through the object. This approximation will be treated as exact from this point on. The relationship between the original distribution and the projections is described by the equation,

$$\rho(\vec{x}_\tau;\hat{\tau}) = \int_{-\infty}^{\infty} \rho(\vec{x}+t\hat{\tau})\,dt \tag{3.1}$$

[1]Semicolons are used throughout to delimit variables from parameters. For example in $\rho(\vec{x}_\tau;\hat{\tau})$, $\vec{x}_\tau$ is the variable and $\hat{\tau}$ is treated as a parameter.

where $\vec{x}_\tau = \vec{x} - (\vec{x} \cdot \hat{\tau})\hat{\tau}$ is a two dimensional projection of the three dimensional vector $\vec{x}$ onto the projection plane Σ_τ and t is a dummy integration variable along the direction $\hat{\tau}$. The projection plane is the plane passing through the origin and perpendicular to the unit direction vector $\hat{\tau}$. From this, the projection data is a two dimensional data set, which lies on the projection plane, that has $\vec{x}_\tau$ as a variable and $\hat{\tau}$ as a parameter. In other words, a different value of $\hat{\tau}$ picks a new two dimensional data set. The last few sentences describe a three dimensional object with two dimensional projections, but it is easier to illustrate with a two dimensional object and a corresponding one dimensional set of projections, as is done in figure 3.1. In figure 3.1 it is assumed that $\rho(\vec{x}) \equiv 0$ outside the object of interest. In order to extend the geometry of figure 3.1 to the three dimensional image reconstruction problem, it is only necessary to let $\vec{x}$ be a three dimensional vector and let $\vec{x}_\tau$ and $\hat{\tau}$ be two dimensional vectors, with corresponding changes in the dimensionality of $\rho(\vec{x})$ and $\rho(\vec{x}_\tau; \hat{\tau})$. In this case the projection data, $\rho(\vec{x}_\tau; \hat{\tau})$, is considered to be a four dimensional data set with each two dimensional projection plane located by the two dimensional direction vector $\hat{\tau}$.

The problem to be solved can now be described as follows; given the measured four dimensional projection data set, $\rho(\vec{x}_\tau; \hat{\tau})$, invert equation 3.1 to recover the original three dimensional tracer distribution, $\rho(\vec{x})$. This problem arises out of several other fields such as solar astronomy, seismology, and sonar. Conversely, any solution can be used by the same fields. Before leaping in to tackle the problem, it is instructive to consider the simpler two dimensional image reconstruction problem.

3.1.2 Two Dimensional Image Reconstruction

In two dimensional image reconstruction the direction vector, $\hat{\tau}$, is completely described by the polar angle θ as shown in figure 3.1. In this case the solution for the

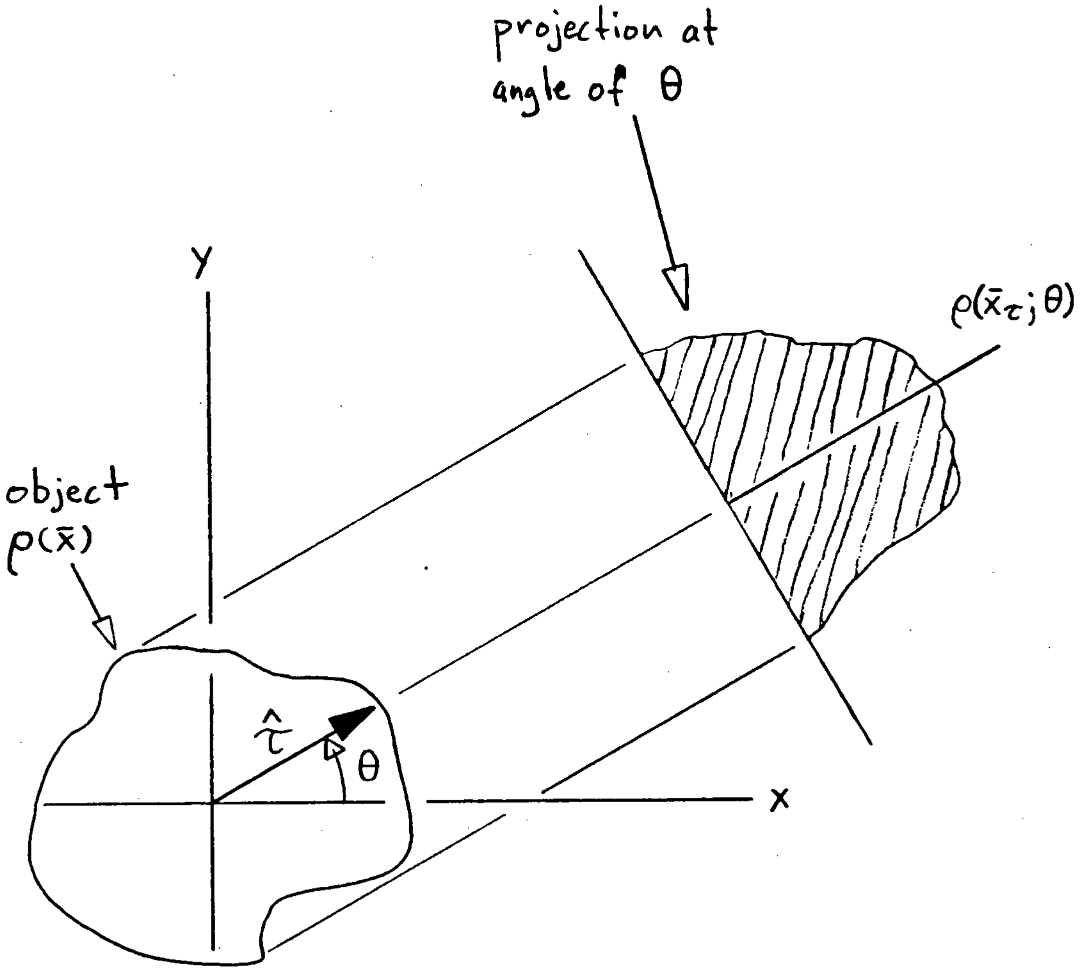

Figure 3.1: Two dimensional object and one of its one dimensional projections

original source distribution is [84],

$$\rho(\vec{x}) = \rho(x,y) = \int_{-\pi/2}^{\pi/2} d\theta \left[\int_{-\infty}^{\infty} dt'\, \rho(t-t';\theta)\, f(t')\right] \tag{3.2}$$

where $x = t\cos\theta$, $y = t\sin\theta$, and

$$f(t) = \delta_{1D}(t) - 1/t^2$$

where $\delta_{1D}(t)$ is the one dimensional Dirac-delta function. The variable t above is the one dimensional equivalent to $\vec{x}_\tau$. The operation inside the square brackets is called convolution and is represented by an asterisk, $*$, as in,

$$f(x) * g(x) = \int_{-\infty}^{\infty} dx'\, f(x-x')g(x')$$

Equation 3.2 can be interpeted as a one dimensional convolution filtering operation followed by a backprojection that is repeated for all values of θ between $-\pi/2$ and $\pi/2$. This is illustrated in figure 3.2 for one value of θ and a particular value of $t = t_0$.

The filtering operation with the filter function $f(t)$ removes the distortion caused by the backprojection operation. If the projections are directly backprojected without filtering, the two dimensional result, called $\Sigma(\vec{x})$ is

$$\Sigma(\vec{x}) = \int_{-\pi/2}^{\pi/2} [\rho(t;\theta)]\, d\theta = \int_{-\pi/2}^{\pi/2} \left[\int_{-\infty}^{\infty} \rho(\vec{x} - t\hat{\tau})\, dt\right] d\theta$$

or

$$\Sigma(\vec{x}) = \int_0^{\pi} \int_0^{\infty} \rho(\vec{x} - \vec{x}') \frac{|\vec{x}|\, d|\vec{x}|\, d\theta_\tau}{|\vec{x}|} = \rho(\vec{x}) ** \frac{1}{|\vec{x}|}$$

where $**$ represents two dimensional convolution. The two dimensional image formed by direct backprojection is the original object convolved with the function $1/|\vec{x}|$.

In the discussion so far in this section, the mathematical analysis assumes that the plane being imaged is infinitely thin. In reality this cannot be or else no data would be detected, so in practice the data is collected from a plane of finite width. This means

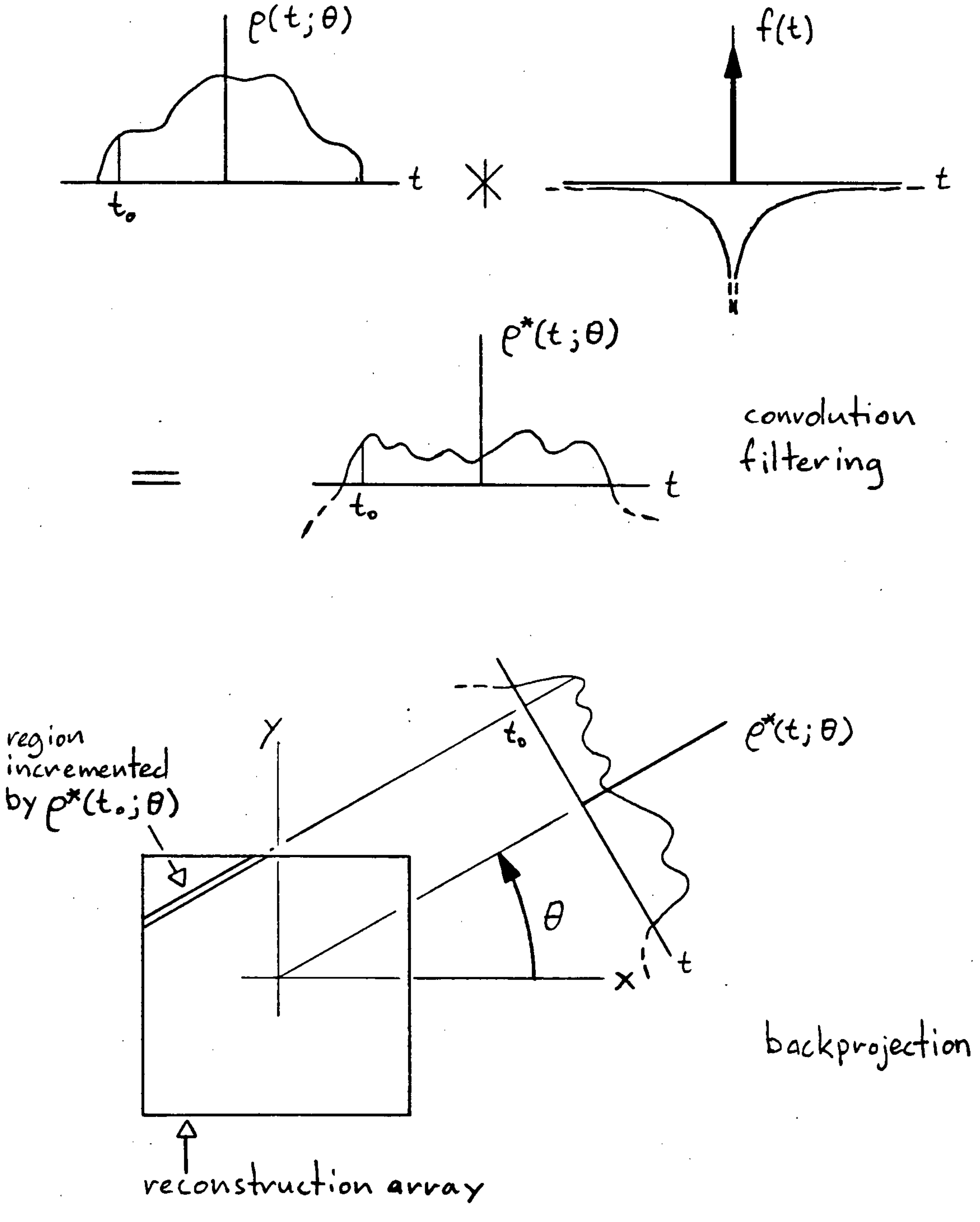

Figure 3.2: Filtering and backprojection of projection data

that the resulting reconstructed two dimensional image is assumed to have a constant value along a line perpendicular to the object plane.

At this point the scanner can be described using linear systems theory, that is, given a known input the output can always be determined. The input is related to the output by the transfer function or Point Spread Function (PSF). This definition comes from setting the input to the system, $\rho(\vec{x})$, to be the Dirac-delta function, $\delta(\vec{x})$, since for any function $f(\vec{x})$, $f(\vec{x}) ** \delta(\vec{x}) = f(\vec{x})$, so

$$\Sigma(\vec{x}) = \delta(\vec{x}) ** \frac{1}{|\vec{x}|} = \frac{1}{|\vec{x}|}$$

In other words this is the resulting image if a point source is placed in the detector. The definition of PSF will also be used in the next section.

It is possible to reverse the order of filtering and backprojection in equation 3.2 by starting with the backprojected image in the above equation and using a two dimensional convolution filter as in the BackProject and Filter (BPF) algorithm [42].

$$\rho(\vec{x}) = \Sigma(\vec{x}) ** \left\{ \delta_{2D}(\vec{x}) - \frac{1}{|\vec{x}|^3} \right\} \tag{3.3}$$

where $\delta_{2D}(\vec{x})$ is the two dimensional Dirac-delta function. The value of $\rho(\vec{x})$ in equations 3.2 and 3.3 are identical, since the filtering and backprojection operations are linear and so their order is interchangeable. The Filter and BackProject (FBP) method of equation 3.2 is used more often because it requires less computer memory. It also allows certain other operations such as those used in tracer kinetic modeling to be performed directly on the projection data [13].

From Fourier transform theory it is known that the convolution of two functions is equivalent to the multiplication (in frequency space) of their Fourier transforms [7]. For example, if $\mathcal{F}\{\}$ is the Fourier transform operator, and

$$F(f) = \mathcal{F}\{f(x)\}, \quad G(f) = \mathcal{F}\{g(x)\}$$

then

$$\mathcal{F}\{f(x) * g(x)\} = F(f)G(f)$$

so the convolution operation is equivalent to

$$f(x) * g(x) = \mathcal{F}^{-1}\{F(f)G(f)\}$$

where $\mathcal{F}^{-1}\{\}$ is the inverse Fourier transform operator. In this manner convolution filtering can be replaced with Fourier transform filtering, at least in the ideal case of no quantiation error and perfect data. In practical implementations, convolution methods are used because of advantages of speed and low noise [63].

3.1.3 Three Dimensional Image Reconstruction

In three dimensional image reconstruction the unit direction vector, $\hat{\tau}$, ranges over the two spherical coordinate angles θ_τ and ϕ_τ as shown in figure 3.3. If $0 \leq \theta_\tau \leq \pi$ and $0 \leq \phi_\tau \leq \pi$, then the three dimensional source distribution, $\rho(\vec{x})$, can be recovered from the set of projections, given by equation 3.1, by an extension of equation 3.2 to [32],

$$\rho(\vec{x}) = \int_0^\pi d\phi_\tau \int_0^\pi d\theta_\tau [\rho(\vec{x}_\tau; \theta_\tau, \phi_\tau) * * f(\vec{x}_\tau)] \tag{3.4}$$

where the reconstruction filter is

$$f(\vec{x}_\tau) = \delta_{2D}(|\vec{x}_\tau|) - \frac{1}{|\vec{x}_\tau|^3}$$

And similarily to equation 3.3 the result of backprojecting all the projections, $\Sigma(\vec{x})$, is given by,

$$\Sigma(\vec{x}) = \int_0^\pi d\phi_\tau \int_0^\pi d\theta_\tau \rho(\vec{x}_\tau; \theta_\tau, \phi_\tau)$$

then $\rho(\vec{x})$ is also given by [32],

$$\rho(\vec{x}) = \Sigma(\vec{x}) * * * \left\{ \delta_{3D}(\vec{x}) - \frac{1}{|\vec{x}|^4} \right\} \tag{3.5}$$

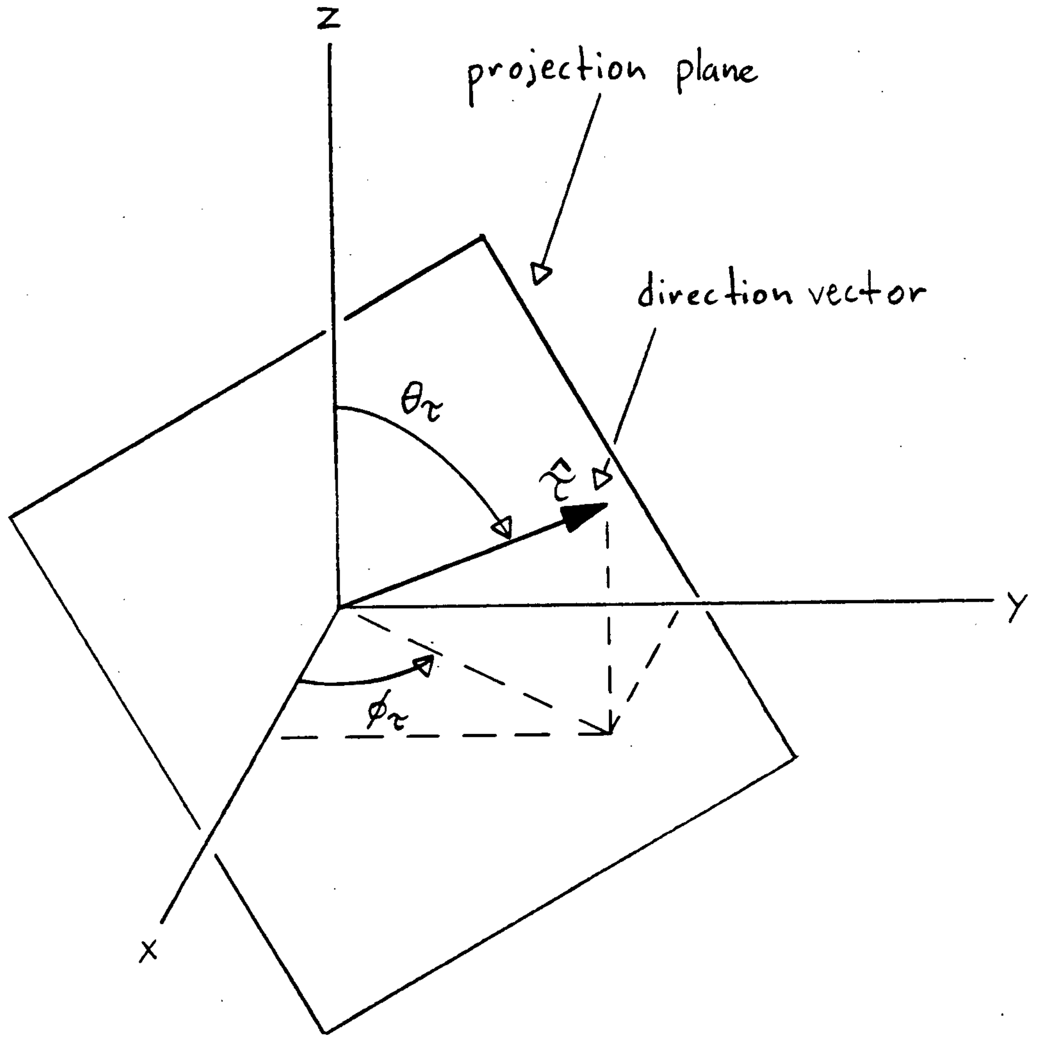

Figure 3.3: Coordinates of the direction vector $\hat{\tau}$

where $***$ is the three dimensional convolution operator and $\delta_{3D}(\vec{x})$ is the three dimensional Dirac-delta function. Analogous to the two dimensional image reconstruction case, the convolutions shown above can be replaced with Fourier transforms [26].

Since most PET scanners are cylindrical in nature, the projection coordinate θ_τ cannot range over 0 to π, but must lie within some smaller range. For a typical scanner this range is $\pi/2 - \psi \leq \theta_\tau \leq \pi/2 + \psi$ where ψ is a parameter of the scanner geometry as is illustrated in figure 3.4. Note that the origin of the coordinate system is at the centre of the cylindrical region.

As $\psi \to 0$, the detector becomes a two dimensional planar detector and if $\psi = \pi/2$ the detector becomes an infinitely long cylinder that can detect all possible gamma rays. From this it is easy to see how the geometrical efficiency is tied to the parameter ψ. The crux of the three dimensional image reconstruction problem is that for $0 < \psi < \pi/2$, none of the preceeding equations in this section can be used to directly recover $\rho(\vec{x})$. This is because the reconstruction filters discussed cannot work with a detector system that does not uniformly sample an object in all directions, nor explicitly take into account the non-planar nature of the images. In other words, the filters have the same form wherever they are applied in equations 3.2 to 3.5. From equation 3.1 and figure 3.4 it can be seen that the projection data gathered for a typical volume imaging scanner will have different qualities depending on the direction vector $\hat{\tau}$.

Appropriate reconstruction filters based on the Fourier transform method of three dimensional image reconstruction have been derived for the case where $0 < \psi < \pi/2$ [70,26,86]. However these algorithms still require a spatially invariant PSF for the detector.

To illustrate spatial invariance, imagine a detector that forms a complete sphere. A point source moving around inside the sphere appears equally bright wherever it is located. This is because the amount of data that the detector can receive from

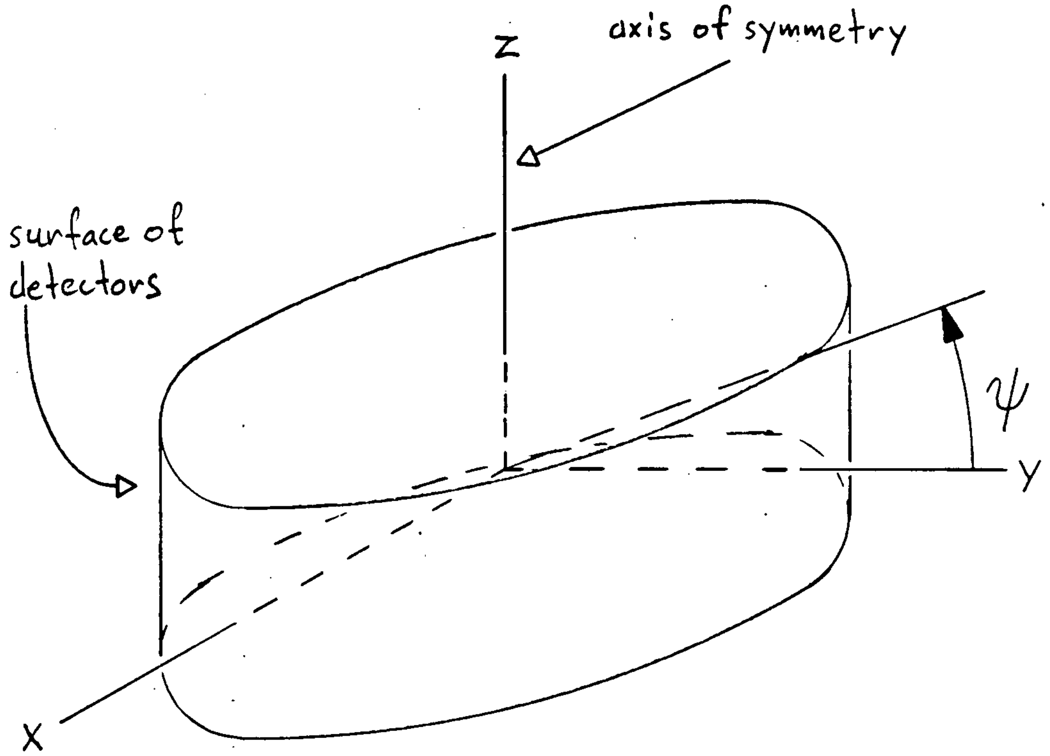

Figure 3.4: Typical volume imaging PET scanner geometry

the object does not depend on the position of the point source. For this impractical[2] scanner it can easily be shown that the PSF of this detector is $1/|\vec{x}|^2$ [68].

The case of a spatially variant PSF is illustrated by imagining a point source inside the detector shown in figure 3.4. In this case it is shown in appendix A that the PSF, called h($\vec{x}$), depends on position and in spherical coordinates is given by,

$$h(r,\theta,\phi) = h(r,\theta) = \frac{1}{r^2}\mathrm{rect}\left(\frac{\theta-\psi}{\pi/2}\right)$$

where

$$\mathrm{rect}(x) = \begin{cases} 1 & \text{if } |x| < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

and r, θ, and ϕ are standard spherical coordinates. This is slightly different from the result quoted by Colsher [26], since he also included an arbitrary scaling factor of $1/\psi$. From the above result it can be seen that as the point source moves around, its apparent brightness will seem to vary because of the changes in detector solid angle (called the acceptance angle) subtended. This is demonstrated in figure 3.5 for two locations of a point source. Note that figure 3.5 shows a side view of the detector surface shown in figure 3.4.

It is possible to obtain an invariant PSF by limiting the acceptance angle at all points in the object to the minimum angle that is determined at the edge of the object. For a spherically symmetric object, with radius R, centred at the origin as in figure 3.6, the minimum acceptance solid angle is $2\pi\psi_{\mathrm{min}}$, where

$$\psi_{\mathrm{min}} = \psi + \arcsin(R\cos\psi/R_{\mathrm{d}}) \tag{3.6}$$

and R_{d} is the detector radius as shown in figure 3.6.

With ψ_{min} as the acceptance angle, any point source located within the bounds of the object will appear to have the same strength regardless of its location. By using

[2]A spherical PET scanner would be too large to be practical

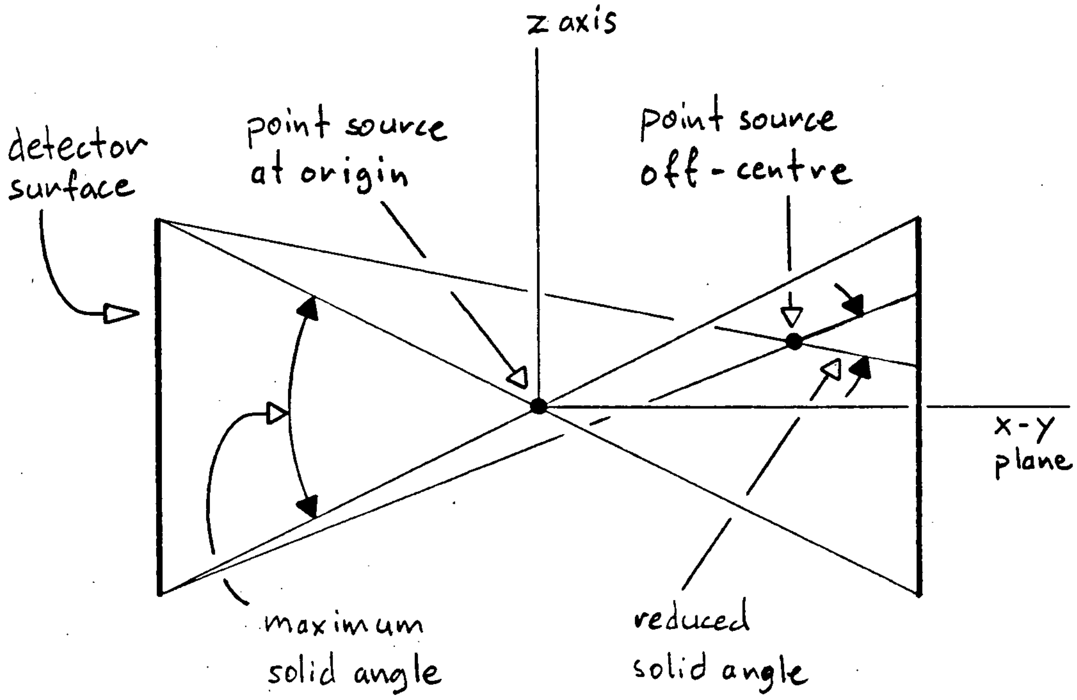

Figure 3.5: Variation of detector PSF with point source position.

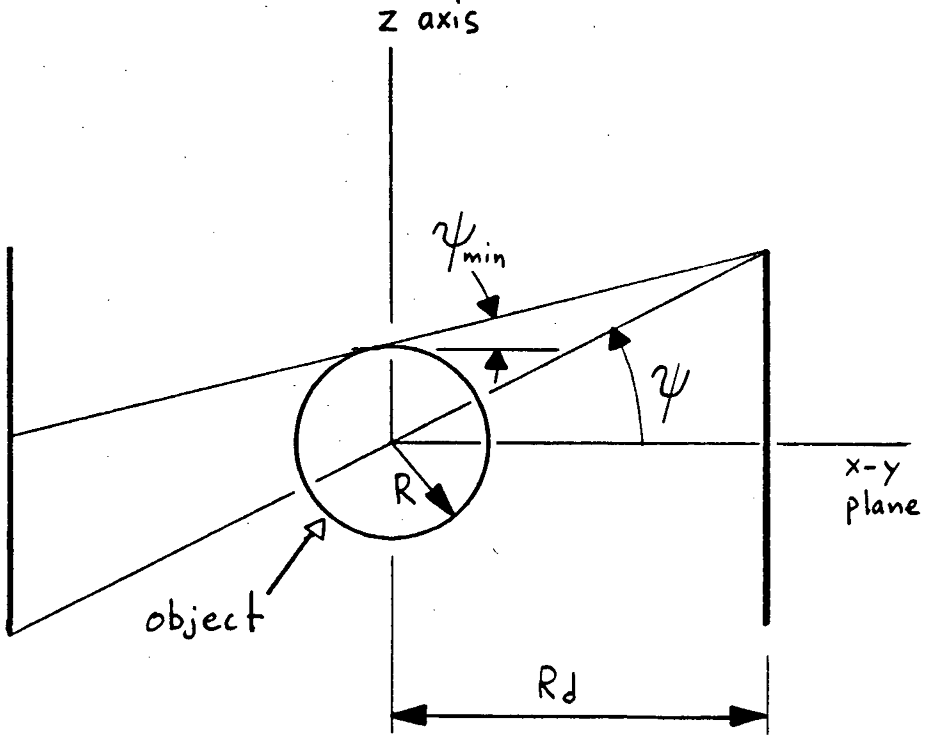

Figure 3.6: Minimum acceptance angle for an invariant detector PSF.

ψ_{min} as a fixed acceptance angle, data that comes from the centre of the detector with an angle $\theta_\tau < \pi/2 - \psi_{\text{min}}$ or $\theta_\tau > \pi/2 + \psi_{\text{min}}$ cannot be used. Also from figure 3.6 and equation 3.6 it can be seen that as $R \to R_{\text{d}}$, $\psi_{\text{min}} \to 0$. This means that for any object of finite size, requiring a spatially invariant PSF means rejecting detected data. In some cases up to 75% of the available data is lost [24,23].

To summarize the material presented in this section, it is possible to perform an analytic three dimensional image reconstruction of $\rho(\vec{x})$ using Fourier transform methods by discarding enough data to obtain a spatially invariant PSF for the detector. Convolution methods had only been developed for the case where $\psi = 0$ or $\psi = \pi/2$. This leads to the next section that starts the investigation into using convolution methods for direct analytic three dimensional image reconstructions.

3.1.4 Theory of Orlov

In 1975, the Russian electron microscopist S. S. Orlov showed that $\rho(\vec{x})$, of dimension n, can be recovered from a complete set of projections, $\rho(\vec{x}_\tau; \hat{\tau})$ of dimension $n-1$, by the following relationship [67,68],

$$\rho(\vec{x}) = -\frac{1}{4\pi^2}\nabla^2 \int\limits_{\mathbf{G}} d\mathrm{G}_\tau \int\limits_{\Sigma_\tau} dS' \, \frac{\rho(\vec{x}_\tau - \vec{x}_\tau{}'; \hat{\tau})}{|\vec{x}_\tau{}'| L(\hat{\tau} \times \vec{x}_\tau)} \tag{3.7}$$

where, for three dimensional image reconstruction, Σ_τ is the plane through the origin perpendicular to $\hat{\tau}$; **G** is the area on the unit sphere that decribes the set of all $\hat{\tau}$ vectors, or projection directions, and $L(\hat{\tau} \times \vec{x}_\tau)$ is a geometric term, "which depends only on the direction of its argument, is equal to the length of the arc of the great circle lying inside the region **G** and passing through the endpoints of the vectors $\hat{\tau}$ and $\vec{x} \cdot |\vec{x}|^{-1}$." [68]. Since the set of projections is defined by the region **G**, Orlov also showed that for **G** to contain a complete set of projections, it must contain the domain of at least one semi-great circle.

To illustrate the definitions of the terms in equation 3.10 it is convenient to model the detector geometry as a truncated sphere of unit radius as shown in figure 3.7. The results derived for a spherical-type detector can easily be converted for use with a cylindrical detector. Note that in figure 3.7 two views of the projection plane and its coordinates are shown, the correct one with only the l_y axis visible, and the same plane rotated about the l_y axis by 90° in order to show the l_x vector and the relationship between the $\vec{x}$, $\hat{\tau}$, and $\vec{l}$ vectors. The region **G** in the figure is simply the portion of a hemisphere lying within the bounds of $\pi/2 - \psi \le \theta_\tau \le \pi/2$.

The coordinate system used in the projection plane is defined by the following,

$$\begin{aligned} \vec{l} &\equiv \vec{x}_\tau = l_x\hat{l}_x + l_y\hat{l}_y \\ \hat{l}_x &= \frac{\hat{z}\times\hat{\tau}}{|\hat{z}\times\hat{\tau}|} \\ \hat{l}_y &= \hat{\tau}\times\hat{l}_x \end{aligned} \tag{3.8}$$

From this, l_x, l_y, and $\hat{\tau}$ form a right hand coordinate system since $\vec{x} = \vec{x}_\tau + t\hat{\tau}$. On the projection plane the rectangular and polar coordinates are given by (l_x, l_y) and (l, θ_l).

Also shown in figure 3.7 is an arc the length of which is the value of the function $L(\hat{\tau} \times \vec{x}_\tau)$. The explicit form of $L(\hat{\tau} \times \vec{x}_\tau)$ is derived in appendix B.

The validity of equation 3.7 was demonstrated for the case where $\psi = \pi/2$ for a complete spherical detector. In this case $L(\hat{\tau} \times \vec{x}_\tau) = \pi$ and equation 3.7 reduces to equation 3.4. If $\psi < \pi/2$, from figure 3.7 it is obvious that $L(\hat{\tau} \times \vec{x}_\tau)$ will not be a constant.

3.1.5 A New Life for Orlov's Recovery Operator

In this section equation 3.7 is reformulated so that it can be used in the case where $\psi < \pi/2$. The first step is to rewrite the inner integral of equation 3.7 as a two

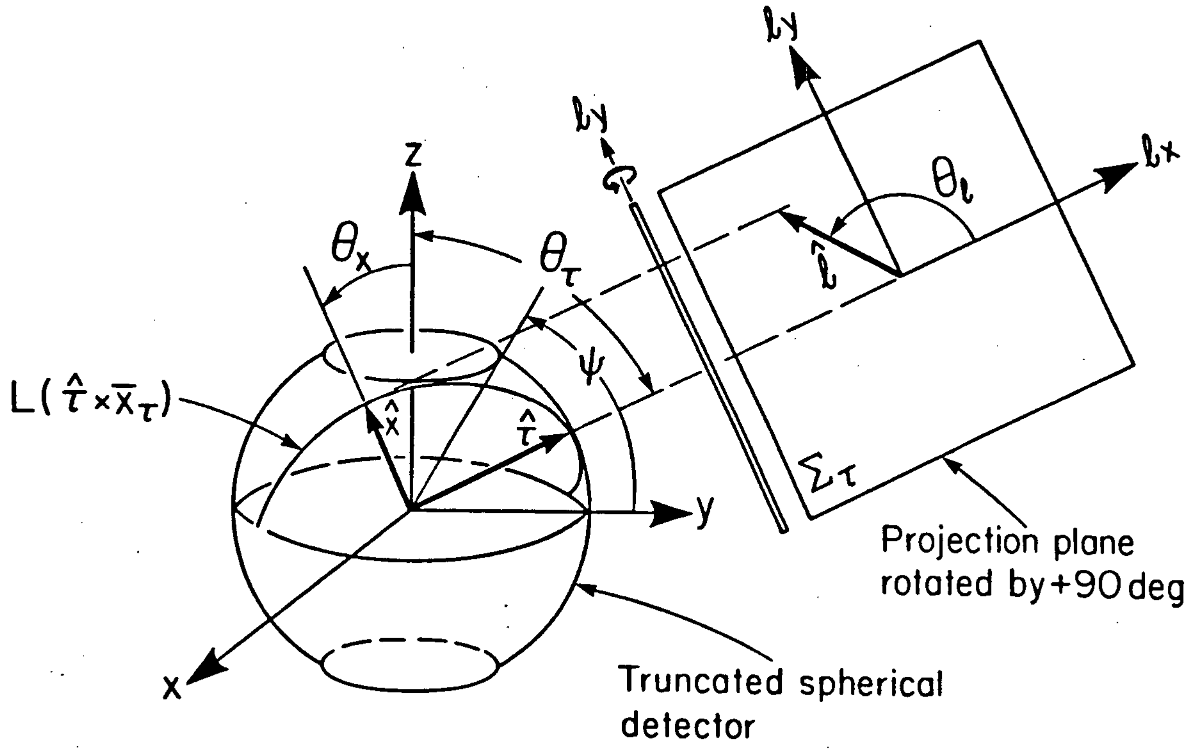

Figure 3.7: Truncated spherical detector and projection plane geometry

dimensional convolution, that is

$$\int_{\Sigma_\tau} dS' \frac{\rho(\vec{x}_\tau - \vec{x}_\tau{}'; \hat{\tau})}{|\vec{x}_\tau{}'| L(\hat{\tau} \times \vec{x}_\tau)} = \rho(\vec{x}_\tau; \hat{\tau}) ** \frac{1}{|\vec{x}_\tau{}'| L(\hat{\tau} \times \vec{x}_\tau)} \tag{3.9}$$

The outer integral of equation 3.7 is a backprojection into a volume of the two dimensional result of the inner integral. The entire equation can be then rewritten as the Laplacian of a three dimensional backprojection of a two dimensional convolution,

$$\rho(\vec{x}) = -\frac{1}{4\pi^2} \nabla^2 \int_{\mathbf{G}} d\mathbf{G}_\tau \left[\rho(\vec{x}_\tau; \hat{\tau}) ** \frac{1}{|\vec{x}_\tau{}'| L(\hat{\tau} \times \vec{x}_\tau)} \right]$$

After bringing the laplacian operator inside the integral, the inner term becomes

$$\nabla^2 \left[\rho(\vec{x}_\tau; \hat{\tau}) ** \frac{1}{|\vec{x}_\tau{}'| L(\hat{\tau} \times \vec{x}_\tau)} \right]$$

Now since for any two functions $f(\vec{x})$ and $g(\vec{x})$ [7],

$$\begin{aligned} \nabla^2 [f(\vec{x}) ** g(\vec{x})] &= [\nabla^2 f(\vec{x})] ** g(\vec{x}) \\ &= f(\vec{x}) ** [\nabla^2 g(\vec{x})] \end{aligned}$$

equation 3.7 can be rewritten as,

$$\rho(\vec{x}) = \int_{\mathbf{G}} d\mathbf{G}_\tau \, [\rho(\vec{x}_\tau; \hat{\tau}) ** f(\vec{x}_\tau; \hat{\tau})] \tag{3.10}$$

where

$$f(\vec{x}_\tau; \hat{\tau}) = -\frac{1}{4\pi^2} \nabla^2 \left(\frac{1}{|\vec{x}_\tau{}'| L(\hat{\tau} \times \vec{x}_\tau)} \right). \tag{3.11}$$

Since the projections are first convolved with $f(\vec{x}_\tau; \hat{\tau})$ before being backprojected, $f(\vec{x}_\tau; \hat{\tau})$ is referred to as the reconstruction filter. Equations 3.10 and 3.11 form the basis of the new algorithm as follows; the desired source density distribution, $\rho(\vec{x})$, is recovered by first filtering the measured two dimensional projections and then backprojecting the result into the three dimensional reconstruction volume. This algorithm is analogous to the standard two dimensional image reconstruction method based on equation 3.2. In two dimensional reconstruction, filtering is done by convolution over one linear dimension and the backprojection is done over one angular dimension. In three dimensional reconstruction, filtering is by convolution over two linear dimensions and backprojection is over two angular dimensions. Using this method of three dimensional image reconstruction, $\rho(\vec{x})$ can be recovered without the use of Fourier transforms, and the advantages of convolution filtering over Fourier transform filtering in the two dimensional case are expected to exist in the three dimensional case.

3.1.6 Analytic Filter Form

In order to use equation 3.10 as a reconstruction algorithm, it is necessary to find the explicit form of the reconstruction filter given by equation 3.11. In appendix B it is shown that the function $L(\hat{\tau} \times \vec{x}_\tau)$ can be expressed in the coordinates of figure 3.4 and figure 3.7 as,

$$L(\hat{\tau} \times \vec{x}_\tau) = L(\theta_l;\theta_\tau) = \begin{cases} \pi, & \text{if } |\cos\theta_l| \geq \cos\psi/|\sin\theta_\tau| \\ 2\arcsin\left(\dfrac{\sin\psi}{\sqrt{1-\cos^2\theta_l\sin^2\theta_\tau}}\right), & \text{otherwise} \end{cases} \tag{3.12}$$

In the same coordinate system, the filter function, $f(\vec{x}_\tau;\hat{\tau})$, of equation 3.11 becomes,

$$f(\vec{x}_\tau;\hat{\tau}) = f(l,\theta_l;\theta_\tau) = -\frac{1}{4\pi^2}\nabla^2\left(\frac{1}{lL(\theta_l;\theta_\tau)}\right) \tag{3.13}$$

and using the the cylindrical coordinate system (l,θ_l,t) with the Laplacian operator given by,

$$\nabla^2 = \frac{1}{l}\frac{\partial}{\partial l}\left(l\frac{\partial}{\partial l}\right) + \frac{1}{l^2}\frac{\partial^2}{\partial\theta_l{}^2} + \frac{\partial^2}{\partial t}$$

equation 3.13 now becomes

$$f(l,\theta_l;\theta_\tau) = -\frac{1}{4\pi^2 l^3}\left[\frac{1}{L(\theta_l;\theta_\tau)} + \frac{\partial^2}{\partial\theta_l{}^2}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right)\right]. \tag{3.14}$$

The explicit form of $f(l, \theta_l; \theta_\tau)$ is obtained by substituting equation 3.12 in for $L(\theta_l; \theta_\tau)$ in equation 3.14. The expansion of the second derivative is done in appendix C and is too lengthy to present here. However several features of the two dimensional reconstruction filter are already apparent upon inspection of equation 3.14. The filter has an overall $-1/l^3$ dependence that is modified by a term that varies only as a function of the polar angle θ_l. There are also two different types of singularities present, one at the origin where $l = 0$, and the other along the lines where $|\cos\theta_l| = \cos\psi/|\sin\theta_\tau|$. These singularities will be discussed in section 3.2. The filter also has no variation with respect to ϕ_τ, which is expected because of the symmetry of the detector about the z-axis as shown in figure 3.4. The filter is also symmetrical about $\theta_\tau = \pi/2$ and $\theta_l = 0, \pm\pi/2, \pi$, and both of these symmetries will be used later to simplify the calculation of the digital reconstruction filter.

3.1.7 Limits of Integration

The last theoretical hurdle remaining before equation 3.10 can be used is to determine the appropriate limits of integration. Writing out the two dimensional convolution explicitly in rectangular coordinates gives,

$$\rho(l_x, l_y; \theta_\tau, \phi_\tau) ** f(l_x, l_y; \theta_\tau) =$$
$$\int_{-\infty}^{\infty} dl_x' \int_{-\infty}^{\infty} dl_y' \, \rho(l_x', l_y'; \theta_\tau, \phi_\tau) f(l_x - l_x', l_y - l_y'; \theta_\tau)$$

where $f(l_x, l_y; \theta_\tau)$ is obtained from equation 3.14 by converting from polar to rectangular coordinates in the projection plane. In the above equation the limits of integration are $\pm\infty$ but the $-1/l^3$ term in equation 3.14 means that in a practical implementation it is possible to neglect contributions for l_x' and l_y' larger than some pre-determined maximum of $|\vec{l}\,|$, called l_{max}.

The integration over the region **G** is a double integral over the two spherical angles that describe $\hat{r}$.

$$\int\limits_{\mathbf{G}} d\mathbf{G}_\tau = \iint\limits_{\mathbf{G}} d\theta_\tau \, d\phi_\tau \, \sin\theta_\tau$$

It is not immediately obvious what the limits of θ_τ and ϕ_τ are. Since, however, the inner integral of equation 3.10 is a convolution of the projections with a spatially invariant filter, and from the discussion in section 3.1.3, it follows that the PSF of the detector in figure 3.7 must be made spatially invariant.

For an object centered at the origin that does not extend further than a distance R, and from the discussion preceding equation 3.6, the effective acceptance angle is given by $\psi_{\text{min}} = \psi + \arcsin(R\cos\psi/R_{\text{d}})$. This means that the limits of θ_τ are $\pi/2 \pm \psi_{\text{min}}$. The limits of ϕ_τ can be determined from figure 3.7 by noting that

$$\rho(l_x, l_y; \theta_\tau, \phi_\tau) = \rho(\tilde{l_x}, \tilde{l_y}; \pi - \theta_\tau, \pi + \phi_\tau)$$

where $\tilde{l_x}$ and $\tilde{l_y}$ are the reflections of the coordinates of l_x and l_y about the origin. In other words, all the projection data is contained, without duplication, in a set of projections described by[3],

$$\pi/2 - \psi \le \theta_\tau \le \pi/2 + \psi, \quad 0 \le \phi_\tau < \pi$$

In its full glory, equation 3.10 now becomes,

$$\begin{aligned} \rho(\vec{x}) \;=\; & \int_{\pi/2-\psi_{\text{min}}}^{\pi/2+\psi_{\text{min}}} d\theta_\tau \, \sin\theta_\tau \int_0^\pi d\phi_\tau \\ & \cdot \left[\int_{-l_{\text{max}}}^{l_{\text{max}}} dl_x{}' \int_{-l_{\text{max}}}^{l_{\text{max}}} dl_y{}' \, \rho(l_x{}', l_y{}'; \theta_\tau, \phi_\tau) f(l_x - l_x{}', l_y - l_y{}'; \theta_\tau) \right] \end{aligned} \tag{3.15}$$

Equation 3.15 is the basis for an algorithm that uses the same cross-plane data as the three dimensional reconstruction methods based on Fourier transforms that were discussed in section 3.1.3. However the derivation of the two dimensional filter and the reconstruction of the three dimensional image occur entirely in object space. This means that the use of DFTs can be avoided.

[3]An equivalent choice is $\pi/2 - \psi \le \theta_\tau \le \pi/2$ and $0 \le \phi_\tau \le 2\pi$.

3.2 Discrete Form

In any practical situation, variables that are continuous in theory can only be measured and manipulated in a discrete manner. The projections $\rho(l_x, l_y; \theta_\tau, \phi_\tau)$ are measured as averages over bins (or pixels), so the integrals of equation 3.15 must be approximated by a set of discrete summations, and the reconstructed estimate of $\rho(\vec{x})$ must be presented as a three dimensional array of values (or voxels). Since the algorithm will manipulate discrete data, equation 3.15 must be rewritten to reflect this. In the three remaining subsections of this chapter, 3.2.1 lays the groundwork for the discrete image reconstruction, 3.2.2 develops the backprojection algorithm, and 3.2.3 contains the calculation of the digital reconstruction filter.

3.2.1 Discrete Reconstruction

The first step in the discrete image reconstruction is to express the integrals in equation 3.15 as a set of sums of integrals over each bin.

$$\begin{aligned}\rho(\vec{x}) \;=\; & \sum_m \int_{\theta_m-\Delta\theta_\tau/2}^{\theta_m+\Delta\theta_\tau/2} d\theta_\tau \, \sin\theta_\tau \sum_n \int_{\phi_n-\Delta\phi_\tau/2}^{\phi_n+\Delta\phi_\tau/2} d\phi_\tau \qquad (3.16)\\ & \cdot \left[\sum_{i'} \sum_{j'} \int_{l_i'-d/2}^{l_i'+d/2} dl_x' \int_{l_j'-d/2}^{l_j'+d/2} dl_y' \, \rho(l_x - l_x', l_y - l_y'; \theta_\tau, \phi_\tau) f(l_x', l_y'; \theta_\tau) \right],\end{aligned}$$

where l_i' and l_j' are the centers of the bins in the projection plane and θ_m and ϕ_n are the centers of the projection direction bins. Each bin is referenced by four indices, the first two, i' and j', describe the location of the bin in a projection plane, and the last two indices, m and n, describe the orientation of the projection plane. If each bin is of square size $d \times d$ in the projection plane and each projection plane is separated by

$\Delta\theta_\tau$ in θ_τ and $\Delta\phi_\tau$ in ϕ_τ, then the region of each integration in equation 3.16 is

$$\begin{array}{rcccll}
l_i - d/2 & \leq & l_x & \leq & l_i + d/2, & \text{where } l_i = id \\
l_j - d/2 & \leq & l_y & \leq & l_j + d/2, & \text{where } l_j = jd \\
\theta_m - \Delta\theta_\tau/2 & \leq & \theta_\tau & \leq & \theta_m + \Delta\theta_\tau/2, & \text{where } \theta_m = m\Delta\theta_\tau \\
\phi_n - \Delta\phi_\tau/2 & \leq & \phi_\tau & \leq & \phi_n + \Delta\phi_\tau/2, & \text{where } \phi_n = n\Delta\phi_\tau
\end{array}$$

The maximum and minimum values of each index are determined as follows,

$$\begin{aligned}
l_{\max} &= i_{\max}d = j_{\max}d \\
i_{\min} &= j_{\min} = -i_{\max} \\
\psi_{\min} &= m_{\max}\Delta\theta_\tau \\
m_{\min} &= -m_{\max} \\
\pi &= n_{\max}\Delta\phi_\tau \\
n_{\min} &= 0
\end{aligned}$$

If $l_{\max}$ is finite and $\Delta\theta_\tau$, $\Delta\phi_\tau$, and d are not infinitesimally small, which means that the projections are discrete, equation 3.16 represents an estimate of $\rho(\vec{x})$. If $\rho(l_x, l_y; \theta_\tau, \phi_\tau)$ is frequency bandlimited such that it is adequately sampled by bins of size $\Delta\theta_\tau \times \Delta\phi_\tau \times d \times d$ then it can be represented by an average value that is constant over each of the bins. In order for $\rho(l_x, l_y; \theta_\tau, \phi_\tau)$ to be bandlimited, it must have no spatial frequencies greater than some frequency $f_{\max}$. The condition that must be satisfied for $\rho(l_x, l_y; \theta_\tau, \phi_\tau)$ to be adequately sampled is $d \leq 1/(2f_{\max})$, which is known as the Nyquist sampling criterion [7]. The equivalent criterion for sampling in θ_τ and ϕ_τ is that $\Delta\theta_\tau, \Delta\phi_\tau \leq 1/(2Rf_{\max})$ where R is the radius of the object [8]. It is not possible for any object of finite size to be bandlimited, but $\rho(l_x, l_y; \theta_\tau, \phi_\tau)$ could be bandlimited if it were possible to apply a low pass filter to $\rho(\vec{x})$ *before* the projection samples were made.

If the above criteria are met, then it is possible to define an exact band limited reconstruction of $\rho(\vec{x})$, called $\tilde{\rho}(\vec{x})$,

Now since $\rho(l_x - l_x', l_y - l_y'; \theta_T, \phi_T)$ is constant over each of the dl_x' and dl_y' integrals and only depends on the indices $i - i'$, $j - j'$, m, and n, then

$$\begin{aligned}\tilde{\rho}(\vec{x}) &= \sum_m \int_{\theta_m - \Delta\theta_T/2}^{\theta_m + \Delta\theta_T/2} d\theta_T \sin\theta_T \sum_n \int_{\phi_n - \Delta\phi_T/2}^{\phi_n + \Delta\phi_T/2} d\phi_T \\ &\cdot \left[\sum_{i'} \sum_{j'} p_{i-i',j-j';m,n} \int_{l_i' - d/2}^{l_i' + d/2} dl_x' \int_{l_j' - d/2}^{l_j' + d/2} dl_y' \, f(l_x', l_y'; \theta_T) \right]\end{aligned} \tag{3.17}$$

where $p_{i,j;m,n}$ are the bandlimited measured projections for each value of (i, j, m, n).

Since $f(l_x, l_y; \theta_T)$ can be determined apriori, as will be shown in section 3.2.3, equation 3.17 can be rewriten as,

$$\begin{aligned}\tilde{\rho}(\vec{x}) &= \sum_m \int_{\theta_m - \Delta\theta_T/2}^{\theta_m + \Delta\theta_T/2} d\theta_T \sin\theta_T \sum_n \int_{\phi_n - \Delta\phi_T/2}^{\phi_n + \Delta\phi_T/2} d\phi_T \\ &\cdot \left[\sum_{i'} \sum_{j'} p_{i-i',j-j';m,n} \, f_{i,j;}(\theta_T) \right]\end{aligned} \tag{3.18}$$

where $f_{i,j;}(\theta_T)$ is a two dimensional digital filter given by,

$$f_{i,j;}(\theta_T) = \int_{l_i' - d/2}^{l_i' + d/2} dl_x' \int_{l_j' - d/2}^{l_j' + d/2} dl_y' \, f(l_x', l_y'; \theta_T) \tag{3.19}$$

The evaluation of the above equation is performed in section 3.2.3.

The double sum inside the square brackets of equation 3.18 is a two dimensional discrete convolution filtering and can be written as,

$$p^*_{i,j;m,n} = p_{i,j;m,n} ** f_{i,j;}(\theta_T),$$

where $p^*_{i,j;m,n}$ is the filtered projection and it is understood that the convolution is over the two indices i and j, and $\hat{r}$ is now a discrete function of m and n.

3.2.2 Three Dimensional Backprojection

The sums over m and n in equation 3.18 are a three dimensional backprojection of the discrete filtered projections, $p^*_{i,j;m,n}$, where i and j are functions of $\vec{x}$, m, and n. The projections are constant over each $d\theta_\tau$ and $d\phi_\tau$ integral and $f_{i,j;}(\theta_\tau)$ has no ϕ_τ dependence, so that each integral over $d\phi_\tau$ reduces to

$$\int_{\phi_n-\Delta\phi_\tau/2}^{\phi_n+\Delta\phi_\tau/2} d\phi_\tau = \Delta\phi_\tau$$

Similarly, in practice the interval $\Delta\theta_\tau$ is small enough so that $\sin\theta_\tau$ and $f_{i,j;}(\theta_\tau)$ are slowly varying over each interval and so,

$$\int_{\theta_m-\Delta\theta_\tau/2}^{\theta_m+\Delta\theta_\tau/2} d\theta_\tau \sin\theta_\tau = \Delta\theta_\tau \sin\theta_m.$$

It is possible to now write equation 3.18 as

$$\tilde{\rho}(\vec{x}) = \sum_m \sum_n [p_{i,j;m,n} * * f_{i,j;m}],$$

where

$$f_{i,j;m} = \Delta\phi_\tau \Delta\theta_\tau \sin\theta_m f_{i,j;}(\theta_\tau).$$

To complete the description of the backprojection process it is necessary to convert $\tilde{\rho}(\vec{x})$ to a discrete form. The reconstructed image is manipulated by a computer so it is stored as a three dimensional array of discrete values, $r_{a,b,c}$, where r is the value of the image at the point (or voxel) indexed by the indices a, b, and c.

The idea behind backprojection is that for each point $\vec{x}$ the value of $\tilde{\rho}(\vec{x})$ is given by the sum of all the contributions to that point from all the filtered projections. This is illustrated for two dimensional backprojection in figure 3.2

There are several different methods used for backprojection, each with its own merits and disadvantages, and the trade-off is the usual one of accuracy vs. computation time [9,42]. The method that was chosen for the current image reconstruction algorithm is as follows; the coordinate of the center of the voxel in the three dimensional reconstruction volume, $r_{a,b,c}$, istransformed to the coordinates l_x and l_y on the (m, n) projection plane and the value of the filtered projection at that point is added to $r_{a,b,c}$. This is then repeated for all filtered projection planes. The coordinate transformation is illustrated in figure 3.7, where $\vec{x}$ is the point to be reconstructed, θ_τ and ϕ_τ describe the orientation of the filtered projection plane, and $\vec{l} = (l_x, l_y)$ is the transformation of $\vec{x}$ onto the plane. The value of the filtered projection at the point $\vec{l}$ is to be added to $r_{a,b,c}$ (pointed to by $\vec{x}$). Given the point (x, y, z) that is indexed by (a, b, c) and the direction vector (θ_τ, ϕ_τ), the needed coordinates (l_x, l_y) are found from,

$$\begin{aligned} l_x &= -x\sin\phi_\tau + y\cos\phi_\tau \\ l_y &= -x\cos\theta_\tau\cos\phi_\tau - y\cos\theta_\tau\sin\phi_\tau + z\sin\theta_\tau \end{aligned}$$

Since the filtered projection plane, $p^*_{i,j;m,n}$, is a discrete two dimensional histogram and it is unlikely that the transformed coordinate (l_x, l_y) will be at the precise center of a histogram bin, some means of interpolation is required. Three of the most common methods are, in order of increasing accuracy and computation time, nearest neighboor, bilinear interpolation, and bicubic spline. The bilinear interpolation method was chosen as a good compromise [40] between speed and accuracy, and is the method used in the reconstruction program listed in appendix E.

The bilinear interpolation method [75] is shown in figure 3.8, where the values P^*_1, P^*_2, P^*_3, and P^*_4 are located at the centers of four adjacent bins (separated by unit distance) of $p^*_{i,j;m,n}$ for a given m and n. The four points shown are those that surround the point (l_x, l_y). Using bilinear interpolation, the value of the filtered projection at the point (l_x, l_y) is given by,

$$\begin{aligned} p^*(l_x, l_y)_{m,n} &= (1-d_x)(1-d_y)P^*_{1;m,n} + d_x(1-d_y)P^*_{2;m,n} \\ &\quad +(1-d_x)d_yP^*_{3;m,n} + d_xd_yP^*_{4;m,n} \end{aligned}$$

A summary of the backprojection algorithm used in the reconstruction program of

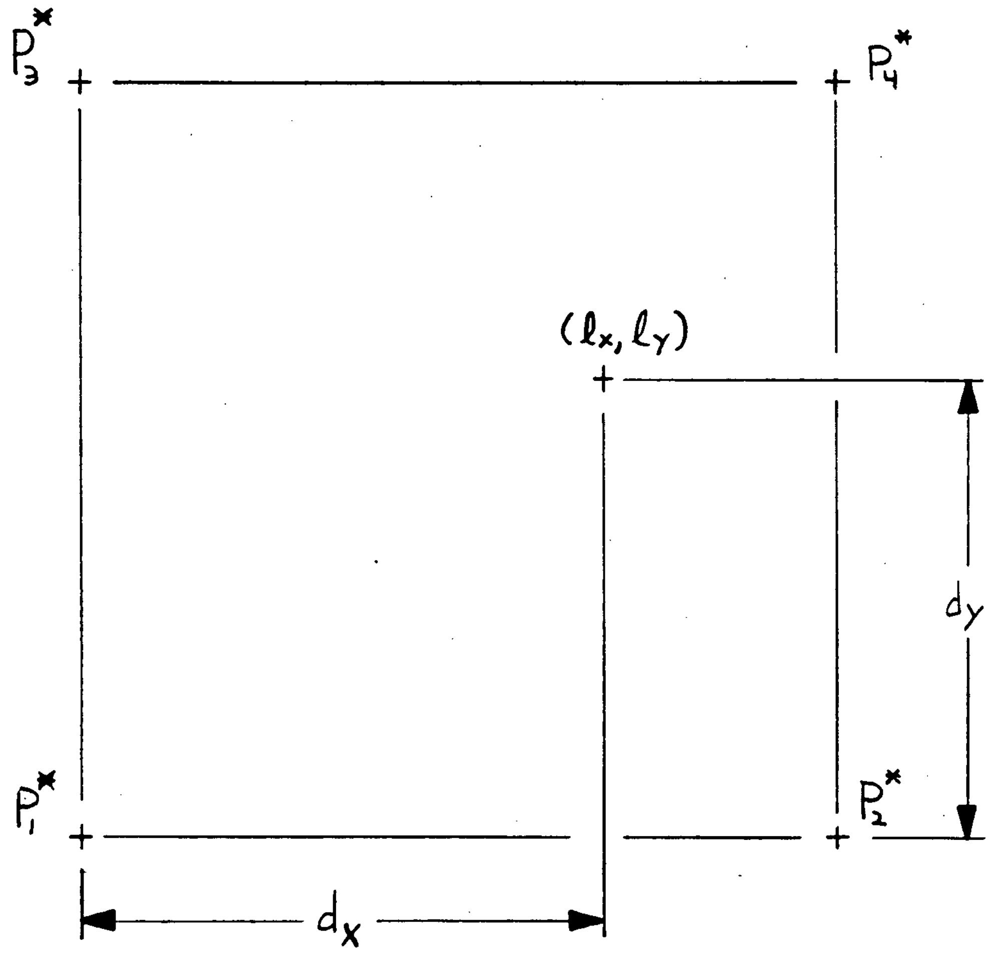

Figure 3.8: Bilinear interpolation

appendix E is as follows;

```
⊢ For each plane $p^*_{i,j;m,n}$ (indexed by $m$ and $n$) loop
⊢ For each voxel $r_{a,b,c}$ (indexed by $a$, $b$, and $c$) loop
    1 Given $(m,n)$ and $(a,b,c)$, transform $\vec{x}$ onto the projection plane to get
      $(l_x, l_y)$.
    2 Given $(l_x, l_y)$ find the interpolated value $p^*(l_x, l_y)_{m,n}$
    3 $r_{a,b,c} \rightarrow r_{a,b,c} + p^*(l_x, l_y)_{m,n}$
⊣ End loop
⊣ End loop
```

3.2.3 Calculation of Digital Filter

What now remains, in order to complete the discrete version of the reconstruction algorithm, is the calculation of the digital filter in equation 3.19. The transformation from the polar coordinates in equation 3.14 to the rectangular coordinates in equation 3.19 is given by,

$$\begin{aligned} l_x &= l\cos\theta_l \\ l_y &= l\sin\theta_l \end{aligned}$$

The difficulties encountered in calculating $f_{i,j;}(\theta_\tau)$ from equation 3.19 are due to the last term in equation 3.14, because of the discontinuities in the first derivative along the lines $\theta_l = \pm\arccos(\cos\psi_{\text{min}}/\sin\theta_\tau)$. To see this in the region $0 \le \theta_l \le \pi/2$, rewrite $1/L(\theta_l;\theta_\tau)$ as

$$\frac{1}{L(\theta_l;\theta_\tau)} = \frac{1}{\pi} + u(\theta_l - \overline{\theta})\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)} - \frac{1}{\pi}\right) \tag{3.20}$$

where $\overline{\theta} = \arccos(\cos\psi_{\text{min}}/\sin\theta_\tau)$ is the angle where $L(\theta_l;\theta_\tau)$ changes form, $u(x)$ is the Heaviside step function definied by

$$u(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x < 0, \end{cases}$$

and

$$\overline{L}(\theta_l;\theta_\tau) = 2\arcsin\left(\frac{\sin\psi_{\text{min}}}{\sqrt{1-\cos^2\theta_l\sin^2\theta_\tau}}\right)$$

which is defined to be π for $\theta_l \leq \overline{\theta}$.

Now using the relationship [54]

$$\frac{\partial}{\partial x}\left\{u(x-a)f(x)\right\} = u(x-a)\frac{\partial}{\partial x}f(x) + \delta(x-a)f(a)$$

the first derivative of $1/L(\theta_l;\theta_\tau)$ can be calculated as,

$$\frac{\partial}{\partial\theta_l}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right) = u(\theta_l-\overline{\theta})\frac{\partial}{\partial\theta_l}\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right) + \delta(\theta_l-\overline{\theta})\left(\frac{1}{\overline{L}(\overline{\theta};\theta_\tau)} - \frac{1}{\pi}\right)$$

and since $\overline{L}(\overline{\theta};\theta_\tau) = \pi$, the last term is identically zero everyhere.

For the second derivative,

$$\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right) = \delta(\theta_l-\overline{\theta})\left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right]_{\theta_l=\overline{\theta}} + u(\theta_l-\overline{\theta})\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right).$$

Using the above result in equation 3.15 yields

$$\begin{aligned} f(l,\theta_l;\theta_\tau) \;=\; & -\frac{1}{4\pi^2 l^3}\left(\frac{1}{\pi} + u(\theta_l-\overline{\theta})\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)} - \frac{1}{\pi}\right)\right. \\ & +\delta(\theta_l-\overline{\theta})\left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right]_{\theta_l=\overline{\theta}} \\ & \left. + u(\theta_l-\overline{\theta})\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right)\right) \end{aligned}$$

This means that the digital filter $f_{i,j;}(\theta_\tau)$ can be given as the integral of $f(l,\theta_l;\theta_\tau)$ over a square ($d \times d$) pixel.

$$f_{i,j;}(\theta_\tau) = \int_{l_i-d/2}^{l_i+d/2} dl_x \int_{l_j-d/2}^{l_j+d/2} dl_y\, f(l,\theta_l;\theta_\tau) \tag{3.21}$$

where $l^2 = {l_x}^2 + {l_y}^2$ and $\theta_l = \arctan(l_y/l_x)$.

To separate the integral in equation 3.21 into manageable parts, $f_{i,j;}(\theta_\tau)$ is treated as the sum of three separate integrals,

$$f_{i,j;}(\theta_\tau) = I1_{i,j;}(\theta_\tau) + I2_{i,j;}(\theta_\tau) + I3_{i,j;}(\theta_\tau), \tag{3.22}$$

where

$$I1_{i,j;}(\theta_\tau) = -\frac{1}{4\pi^3}\int_{l_i-d/2}^{l_i+d/2} dl_x \int_{l_j-d/2}^{l_j+d/2} dl_y \, \frac{1}{l^3},$$
$$I2_{i,j;}(\theta_\tau) = \frac{1}{4\pi^3}\int_{l_i-d/2}^{l_i+d/2} dl_x \int_{l_j-d/2}^{l_j+d/2} dl_y \, \frac{u(\theta_l - \overline{\theta})}{l^3},$$

and

$$\begin{aligned} I3_{i,j;}(\theta_\tau) = -\frac{1}{4\pi^2}\int_{l_i-d/2}^{l_i+d/2} dl_x \int_{l_j-d/2}^{l_j+d/2} dl_y \, \frac{1}{l^3} \\ \cdot \left(u(\theta_l - \overline{\theta}) \left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)} + \frac{\partial^2}{\partial {\theta_l}^2}\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right)\right) \right. \\ \left. + \, \delta(\theta_l - \overline{\theta}) \left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right]_{\theta_l = \overline{\theta}} \right). \end{aligned}$$

The two terms $I1_{i,j;}(\theta_\tau)$ and $I2_{i,j;}(\theta_\tau)$ will be evaluated explicitly below, but $I3_{i,j;}(\theta_\tau)$ is too complicated for analytic integration, and must evaluated by numerical approximation. Each term will now be considered in turn.

For each term, the pixel of consideration is demarked by the four corners (x_1, y_1), (x_1, y_2), (x_2, y_1), and (x_2, y_2), where

$$\begin{aligned} x_1 &= l_i - d/2, \quad x_2 &= l_i + d/2 \\ y_1 &= l_j - d/2, \quad y_2 &= l_j + d/2. \end{aligned}$$

The direct evaluation of $I1_{i,j;}(\theta_\tau)$ gives.

$$I1_{i,j;}(\theta_\tau) = \frac{1}{4\pi^3}\left[\left[\frac{\sqrt{x^2+y^2}}{xy}\right]_{x=x_1}^{x=x_2}\right]_{y=y_1}^{y=y_2}. \tag{3.23}$$

In the case where $x_1 = 0$ then

$$I1_{i,j;}(\theta_\tau) = \frac{1}{4\pi^3}\left[\frac{\sqrt{x_2^2+y^2}}{x_2 y}\right]_{y=y_1}^{y=y_2},$$

and if $y_1 = 0$ instead, then

$$I1_{i,j;}(\theta_\tau) = \frac{1}{4\pi^3}\left[\frac{\sqrt{x^2+y_2^2}}{x y_2}\right]_{x=x_1}^{x=x_2}.$$

For $I2_{i,j;}(\theta_\tau)$ and $I3_{i,j;}(\theta_\tau)$, it is necessary to consider the four different ways that the line $\theta_l = \overline{\theta}$ can intersect the pixel being integrated over, as shown in figure 3.9.

In evaluating $I2_{i,j;}(\theta_\tau)$, it suffices to use one of the cases shown in figure 3.9, which is enlarged upon in figure 3.10, where the pixel is divided into three regions of integration. In region R_1 the integrand in the expression for $I2_{i,j;}(\theta_\tau)$ is zero since $\theta_l < \overline{\theta}$. For region R_2, $I2_{i,j;}(\theta_\tau)$ has the form,

$$\frac{1}{4\pi^3}\int_{y_a}^{y_b} dl_y \int_{x_1}^{y/\tan\overline{\theta}} dl_x \left(\frac{1}{(x^2+y^2)^{3/2}}\right) = \frac{1}{4\pi^3}\left[\frac{\sqrt{x_2^2+y^2}}{x_2 y} - \frac{\cos\overline{\theta}}{y}\right]_{y=y_a}^{y=y_b},$$

where $y_a = x_1 \tan\overline{\theta}$ and $y_b = x_2 \tan\overline{\theta}$. The functional form of $I2_{i,j;}(\theta_\tau)$ over the region R_3 is similar to $I1_{i,j;}(\theta_\tau)$, so the complete form of $I2_{i,j;}(\theta_\tau)$ for the case shown in figure 3.11 is,

$$I2_{i,j;}(\theta_\tau) = -\frac{1}{4\pi^3}\left(\left[\frac{\sqrt{x_2^2+y^2}}{x_2 y} - \frac{\cos\overline{\theta}}{y}\right]_{y=y_a}^{y=y_b} - \left[\left[\frac{\sqrt{x^2+y^2}}{xy}\right]_{x=x_1}^{x=x_2}\right]_{y=y_b}^{y=y_2}\right) \tag{3.24}$$

The forms of $I2_{i,j;}(\theta_\tau)$ in the other three cases shown in figure 3.9 are similar to that of equation 3.24 with minor variations to take into account how the line $\theta_l = \overline{\theta}$ intersects the pixel.

The last and most difficult of the terms to evaluate is $I3_{i,j;}(\theta_\tau)$. From the form of the second derivative of $1/L(\theta_l;\theta_\tau)$ shown in appendix C, it is obvious the the integrations needed to calculate $I3_{i,j;}(\theta_\tau)$ must be done numerically. Starting with

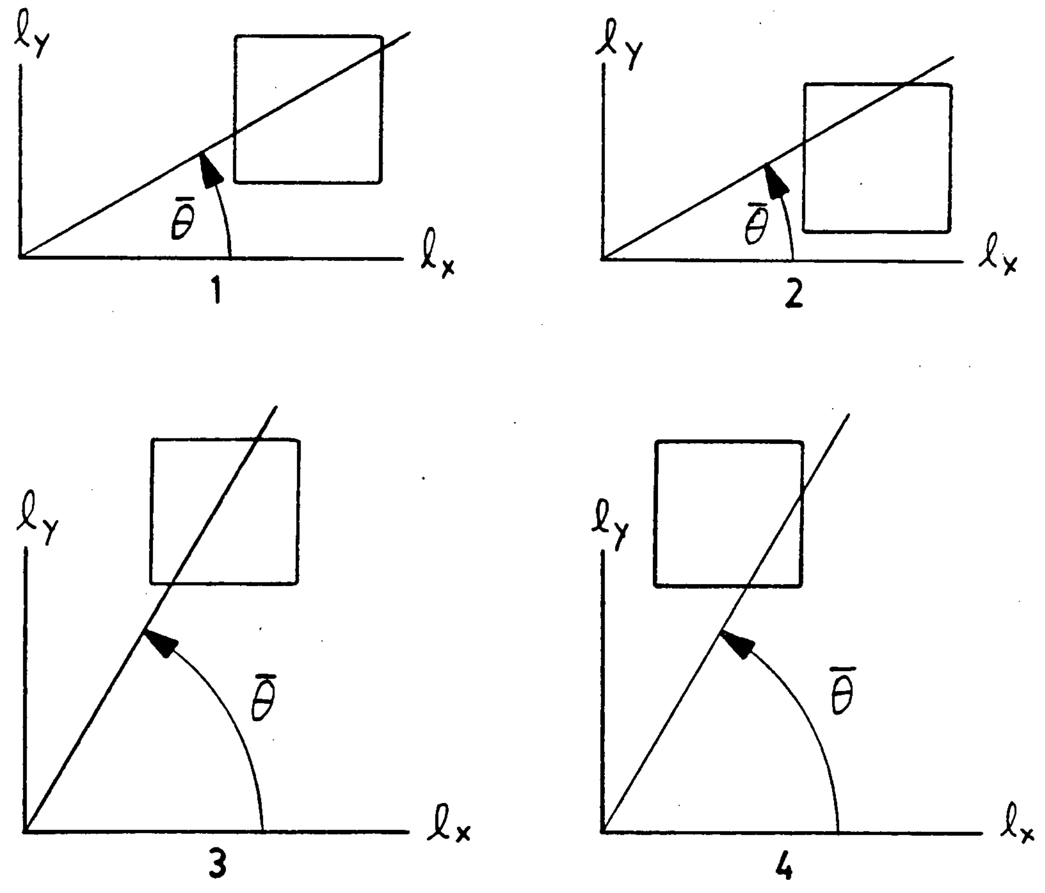

Figure 3.9: Possible intersections of the line $\theta_l = \overline{\theta}$ with a pixel

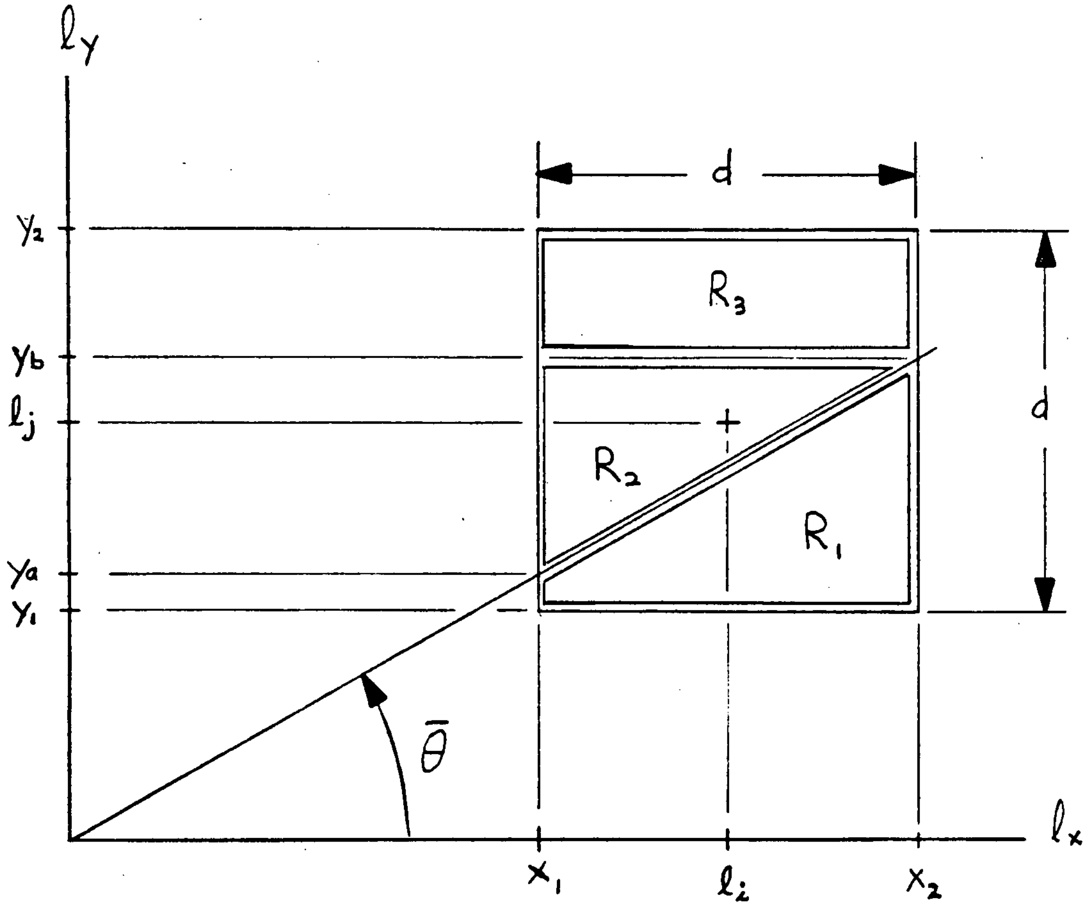

Figure 3.10: Regions of integration in a pixel for $I2_{i,j;}(\theta_\tau)$.

[25],

$$\int_a^b dx\,\delta(x-x_0)f(x) = \begin{cases} f(x_0), & \text{if } x_0 \in [a,b] \\ 0, & \text{if } x_0 \notin [a,b] \end{cases}$$

the term in $I3_{i,j;}(\theta_\tau)$ involving $\delta(\theta_l - \overline{\theta})$ is straightforward to evaluate. It turns out, however, not to be necessary to do because of the following. By analyzing the form of the second derivative of $1/L(\theta_l;\theta_\tau)$, shown in appendex C, it can be seen that,

$$\lim_{\epsilon\to 0}\left\{\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{L(\overline{\theta}-\epsilon;\theta_\tau)}\right)\right\} = 0$$

and

$$\lim_{\epsilon\to 0}\left\{\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{L(\overline{\theta}+\epsilon;\theta_\tau)}\right)\right\} = -\infty.$$

This is illustrated in figure 3.11. Because of the behaviour of the second derivative as $\theta_l \to \overline{\theta}^+$, any straightforward numerical integration technique will underestimate the contribution of the second derivative term to $I3_{i,j;}(\theta_\tau)$. In order to circumvent this problem, a different approximation approach that takes into account the behaviour of the second derivative and incorporates the above Dirac-delta function was adopted.

Since

$$\frac{\partial}{\partial\theta_l}\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right) = 0$$

at $\theta_l = 0$ or $\pi/2$, it can be seen that

$$\int_0^{\pi/2} d\theta_l\left\{\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right)\right\} = 0$$

and since the second derivative term is zero for $\theta_l < \overline{\theta}$ and $\theta_l = \pi/2$,

$$\int_{\overline{\theta}_{-\epsilon}}^{\overline{\theta}_{-\epsilon+\alpha}} d\theta_l\left\{\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right)\right\} = -\int_{\overline{\theta}_{-\epsilon+\alpha}}^{\pi/2} d\theta_l\left\{\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right)\right\}$$

so that

$$\int_{\overline{\theta}_{-\epsilon}}^{\overline{\theta}_{-\epsilon+\alpha}} d\theta_l\left\{\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right)\right\} = \left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right]_{\theta_l=\overline{\theta}_{-\epsilon+\alpha}} \tag{3.25}$$

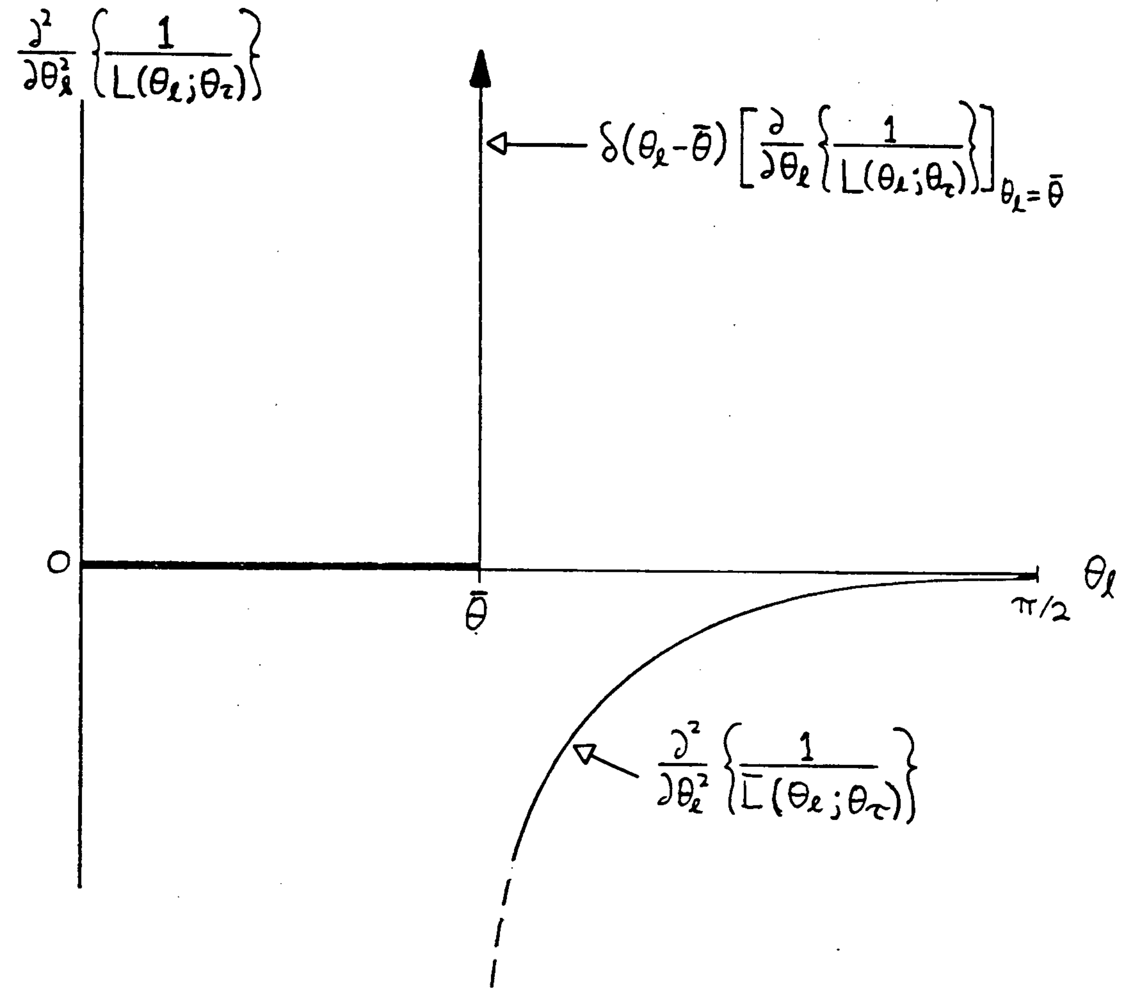
$\frac{\partial^2}{\partial\theta_l^2}\left\{\frac{1}{L(\theta_l;\theta_\tau)}\right\}$
$\delta(\theta_l-\bar{\theta})\left[\frac{\partial}{\partial\theta_l}\left\{\frac{1}{L(\theta_l;\theta_\tau)}\right\}\right]_{\theta_l=\bar{\theta}}$
0
$\bar{\theta}$
$\pi/2$
θ_l
$\frac{\partial^2}{\partial\theta_l^2}\left\{\frac{1}{\bar{L}(\theta_l;\theta_\tau)}\right\}$

Figure 3.11: Sketch of the second derivative of $1/L(\theta_l;\theta_\tau)$.

This means that if there is an angular integration range of α that includes the value $\theta_l = \overline{\theta}$, it is possible to replace the integral with a non-singular term that avoids dealing directly with the singularities of the second derivative. The difficulty now arises that the integration in equation 3.21 is over a square pixel - but it is necessary to perform the integration on polar coordinates in order to avoid the singularity at $\theta_l = \overline{\theta}$. This difference in coordinate systems can be reconciled by approximating a thin strip about the line singularity to be an annular segment as shown in figure 3.12.

Since $I3_{i,j;}(\theta_\tau)$ is to be calculated by a two dimensional numerical integration, such as Simpson's rule, the integrand would normally be evaluated at the grid points shown, and,

$$\begin{aligned} I3_{i,j;}(\theta_\tau) \approx & \sum_{s=0}^{N}\sum_{t=0}^{N} w_{s,t}\left[-\frac{1}{4\pi^2 l^3}\left(u(\theta_l-\overline{\theta})\left(\frac{1}{\overline{L}(\theta_l;\theta_\tau)}+\frac{\partial^2}{\partial\theta_l{}^2}\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right)\right.\right. \\ & \left.\left. + \, \delta(\theta_l-\overline{\theta})\left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_\tau)}\right]_{\theta_l=\overline{\theta}}\right)\right] \end{aligned} \qquad (3.26)$$

where $h = d/N$ is the spacing between grid points in the pixel, $d = x_2 - x_1 = y_2 - y_1$, $w_{s,t}$ are the appropriate quadrature weights, and l and θ_l come from $l_x = x_1 + sh$ and $l_y = y_1 + th$ (so that $l^2 = l_x{}^2 + l_y{}^2$ and $\theta_l = \arctan(l_y/l_x)$). But because of the nature of the second derivative of $1/L(\theta_l;\theta_\tau)$ as shown by figure 3.11, the accuracy of the integration in equation 3.26 would depend in an arbitrary way on the spacing of the grid points near $\theta_l = \overline{\theta}$. This is the underestimation of $I3_{i,j;}(\theta_\tau)$ mentioned above that can now be corrected for by using equation 3.25 to calculate the contribution to the integral from those points in equation 3.26 that fall inside the strip shown in figure 3.12. The narrower the strip, the smaller the deviations from a true annular segment are. For the annular segment $l_1 \leq l \leq l_2$, $\overline{\theta} - \epsilon \leq \theta_l \leq \overline{\theta} - \epsilon + \alpha$, where $l_1 = x_1 \cos\overline{\theta}$ and $l_2 = x_2 \cos\overline{\theta}$, the part of the sum in equation 3.26 for grid points

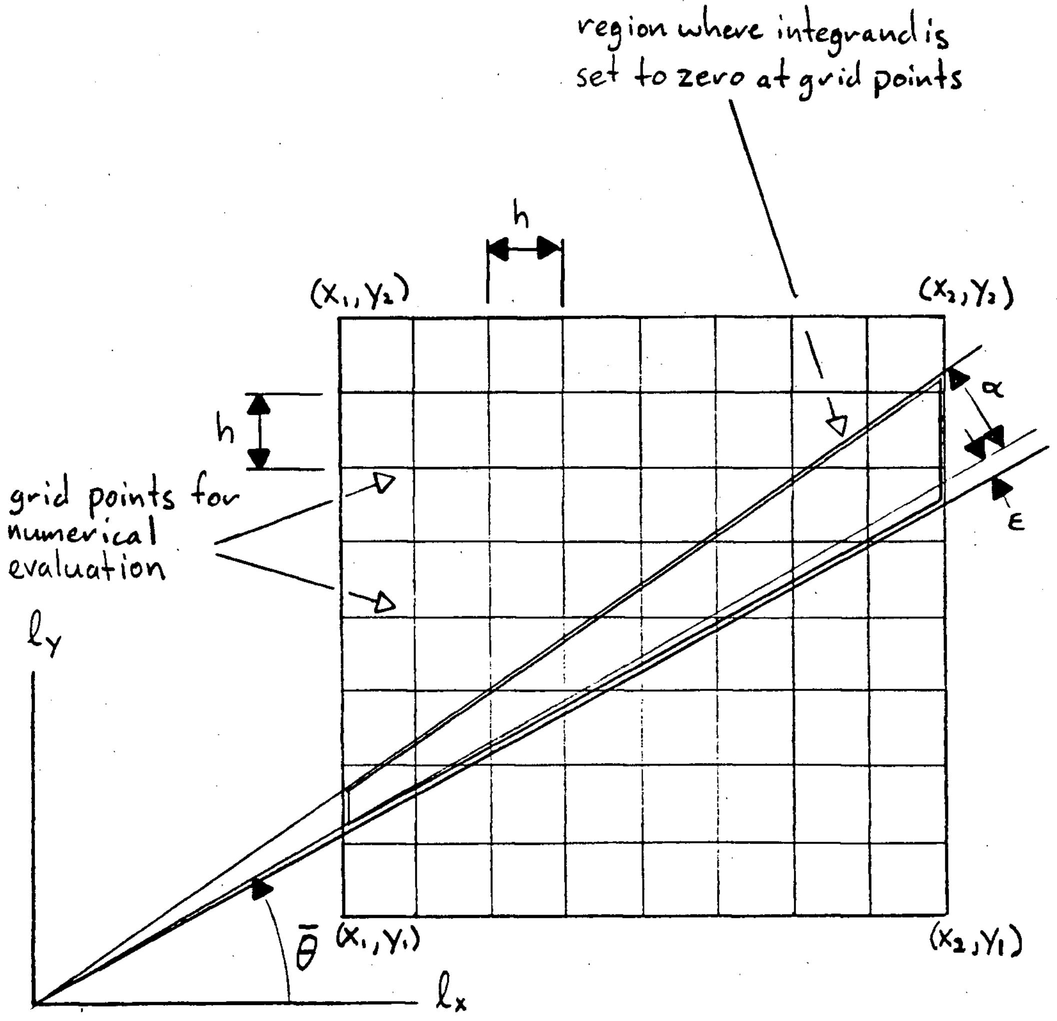

Figure 3.12: Integration scheme over a pixel for $I3_{i,j;}(\theta_\tau)$.

inside the segment, denoted by s' and t', can be replaced as follows,

$$\begin{aligned}\sum_{s'}\sum_{t'} w_{s,t}\left[-\frac{1}{4\pi^2 l^3}\left(u(\theta_l-\overline{\theta})\frac{\partial^2}{\partial\theta_l^2}\frac{1}{\overline{L}(\theta_l;\theta_r)}+\delta(\theta_l-\overline{\theta})\left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_r)}\right]_{\theta_l=\overline{\theta}}\right)\right] \\ \cong -\frac{1}{4\pi^2}\int_{l_1}^{l_2}\frac{l\,dl}{l^3}\int_{\overline{\theta}-\epsilon}^{\overline{\theta}-\epsilon+\alpha} d\theta_l\left\{\frac{\partial^2}{\partial\theta_l^2}\frac{1}{\overline{L}(\theta_l;\theta_r)}\right\} \\ = \left[\frac{1}{4\pi^2 l}\right]_{l=l_1}^{l=l_2}\cdot\left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_r)}\right]_{\theta_l=\overline{\theta}-\epsilon+\alpha}\end{aligned}$$

Now by letting $\epsilon \to 0$, the final form for $I3_{i,j;}(\theta_r)$ is,

$$\begin{aligned}I3_{i,j;}(\theta_r) \approx & \sum_{s=0}^{N}\sum_{t=0}^{N} w'_{s,t}\left[-\frac{1}{4\pi^2 l^3}\left(\frac{u(\theta_l-\overline{\theta})}{\overline{L}(\theta_l;\theta_r)}+u(\theta_l-(\overline{\theta}+\alpha))\left(\frac{\partial^2}{\partial\theta_l^2}\frac{1}{\overline{L}(\theta_l;\theta_r)}\right)\right)\right] \\ & +\frac{1}{4\pi^2 l}\left(\frac{1}{l_2}-\frac{1}{l_1}\right)\left[\frac{\partial}{\partial\theta_l}\frac{1}{\overline{L}(\theta_l;\theta_r)}\right]_{\theta_l=\overline{\theta}+\alpha}\end{aligned} \tag{3.27}$$

where it is understood that $w'_{s,t} = 0$ for s and t inside the segment.

The accuracy of the numerical approximation depends asymptotically on the value of N with the usual trade-off between speed and accuracy. From equation 3.25 it is seen that equation 3.27 should be relatively insensitive to the value of the parameter α as long as α is small enough that the segment is close to being annular. Also, from the geometry of figure 3.12, α must be larger than d/l_1 so that it is guaranteed that some grid points will lie in the segment. In trials, by evaluating equation 3.27 for different values of α, it was indeed found that for $N = 50$, $I3_{i,j;}(\theta_r)$ did not change significantly for $d/l_1 < \alpha < 5d/l_1$.

Beyond $N = 30$ the value of $\sum_i \sum_j f_{i,j;}(\theta_r)$ changed little with increasing N, but even $N = 30$ results in evaluating first term of equation 3.27 at 900 grid points for *each* pixel. To reduce the computational burden it is possible to use much smaller values of N for pixels in regions where $f(l, \theta_l; \theta_r)$ is slowly varying. The gridding algorithm adopted was to use $N = 8$ unless the pixel was within two pixels of the origin or the line $\theta_l = \overline{\theta}$, in which case $N = 30$ was used. It was found that this gave the same values for $I3_{i,j;}(\theta_r)$ as when $N = 30$ was used for all pixels. For a 41×41 digital filter this method reduced the computation time by $\sim 80\%$. The weights $w_{s,t}$ in equation 3.27 were the ones from the extended Simpson's rule [75].

Now by using equations 3.22, 3.23, 3.24, and 3.27 it is possible to compute $f_{i,j;}(\theta_\tau)$ for $i, j, > 0$. To calculate $f_{i,j;}(\theta_\tau)$ for i and/or $j < 0$ it is only necessary to observe the two-fold symmetry of $L(\theta_l; \theta_\tau)$, in equation 3.12, about $\theta_l = 0, \pm\pi/2, \text{and } \pi$. The values of $f_{i,j;}(\theta_\tau)$ for such i and j are then found by reflection of the values for i and/or $j > 0$ about the l_x and/or l_y axes. If one of i or j is zero then $f(l, \theta_l; \theta_\tau)$ is symmetric about the axis through the middle of the pixel and $f_{i,j;}(\theta_\tau)$ only has to be calculated for the portion where $0 \leq \theta_l \leq \pi/2$, and then doubled.

The last part of $f_{i,j;}(\theta_\tau)$ to be calculated is the origin, or $f_{0,0;}(\theta_\tau)$. Because of the $-1/l^3$ term in $f(l, \theta_l; \theta_\tau)$, it is not possible to directly calculate $f_{0,0;}(\theta_\tau)$. The center pixel can be obtained by imposing the condition,

$$\int_{-\infty}^{\infty} dl_x \int_{-\infty}^{\infty} dl_y \, f(l_x, l_y; \theta_\tau) = 0$$

This follows from the property that the integral over all space of the filter represents the value of the Fourier transform of $f(l_x, l_y; \theta_\tau)$ evaluated at zero spatial frequency [7], and by comparison with the frequency space filters in chapter 5, the filter must be zero at zero frequency [45]. From this comes the relationship,

$$\sum_i \sum_j f_{i,j;}(\theta_\tau) = 0$$

or

$$f_{0,0;}(\theta_\tau) = -\sum_{(i,j)\neq(0,0)}\sum f_{i,j;}(\theta_\tau) \tag{3.28}$$

Figure 3.13 shows a histogram plot of the two dimensional digital filter, $f_{i,j;}(\theta_\tau)$, for $\theta_\tau = 90^\circ$ and $\psi = 10^\circ$[4]. Because of the magnitude of $f_{0,0;}(\theta_\tau)$ no detail is visible, so in figure 3.14 the center 9×9 pixels are set to zero and the histogram is flipped upside

[4]A value of ψ appropriate to the new PET machine

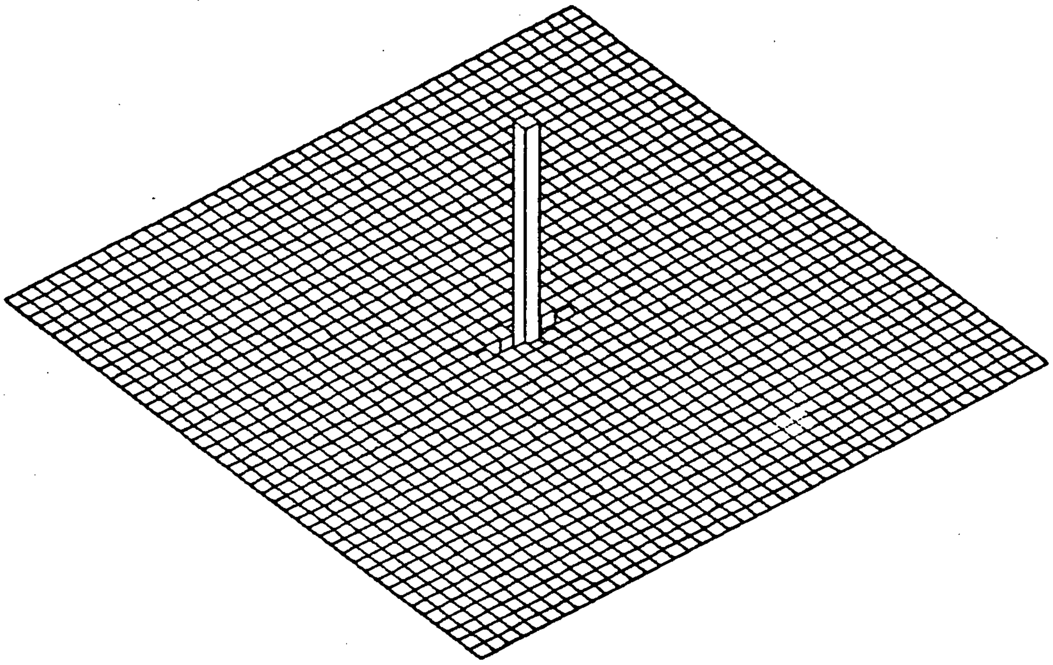

Figure 3.13: Digital filter for $\theta_r = 90°$ and $\psi = 10°$

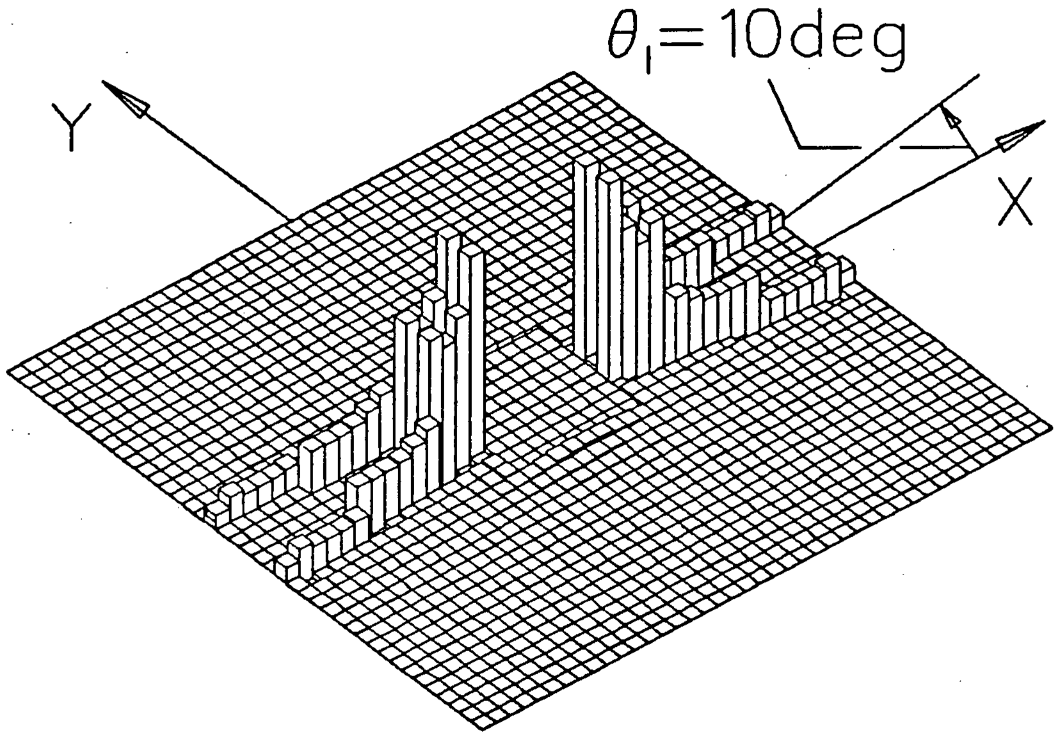

Figure 3.14: Negative of the filter with the center 9 × 9 pixels set to zero

down (by mutiplying it by -1) in order to show more detail. The ridges at $\theta_l = \pm 10°$ are due to the discontinuity at the same angle that is shown in figure 3.11 that is in turn due to the discontinuity in slope of the function $L(\hat{r} \times \vec{x}_r)$ in equation 3.12. By looking at $f_{i,j;}(\theta_r)$ vs. ψ and θ_r it was noticed that the overall shape of $f_{i,j;}(\theta_r)$ depends more strongly on ψ than on θ_r. Unfortunately it is difficult to develop an intuitive understanding of how changes in the shape of the filter affect the reconstructed image.

It is apparent that the calculation of the digital filter is non-trivial and time consuming. However once the detector parameter ψ and the projection angles θ_m and ϕ_n are known, $f_{i,j;}(\theta_r)$ need only be calculated once and then stored in computer memory. The computer program used to calculate $f_{i,j;}(\theta_r)$ is listed in appendix D.

Chapter 4

Implementation

This chapter reports on the results of using the three dimensional reconstruction program listed in appendix E. Minor modifications were made to the program in order to provide a simple test case for the reconstruction algorithm and to speed up the test cycle.

4.1 Reconstruction Details

The object chosen to provide a test for the algorithm was a sphere of radius R with a uniform density,

$$\rho(\vec{x}) = \begin{cases} 1, & \text{if } |\vec{x}| \leq R \\ 0, & \text{if } |\vec{x}| > R \end{cases} \tag{4.1}$$

The projections of a uniform sphere at any angle are given by,

$$\rho(\vec{x}_\tau;\hat{\tau}) = \rho(l,\theta_l;\theta_\tau,\phi_\tau) = \begin{cases} 2\sqrt{R^2 - |\vec{l}|^2}, & \text{if } |\vec{l}| \leq R \\ 0, & \text{if } |\vec{l}| > R \end{cases}$$

so the discrete projections are,

$$p_{i,j;m,n} = \begin{cases} 2\sqrt{R^2 - (i^2+j^2)d^2}, & \text{if } i^2+j^2 \leq (R/d)^2 \\ 0, & \text{if } i^2+j^2 > (R/d)^2 \end{cases}$$

where i and j are the pixel index numbers on the projection plane.

The parameter ψ_{min} was chosen to be 10° and projections of the sphere where taken at,

$$\begin{aligned} \theta_m = &\ \pi/2 + m\Delta\theta_\tau, \quad m = -3, \ldots, 3 \quad \text{(7 angles)} \\ \phi_n = &\ n\Delta\phi_\tau, \qquad\qquad n = 0, \ldots, 59 \quad \text{(60 angles)} \end{aligned}$$

where $\Delta\theta_\tau = \psi_{\text{min}}/3$ and $\Delta\phi_\tau = \pi/60$.

The reconstruction volume was $40 \times 40 \times 40$ voxels, each of unit size, and the sphere had a radius of 10 units. The projection planes were 40×40 pixels of unit size, and because of the symmetry of the sphere all the projections were identical. The chosen values for ψ_{min} and θ_m were used by the filter generation program listed in appendix D to generate a 41×41 digital reconstruction filter.

In order to reduce aliasing artifacts, it is necessary to bandlimit an object before sampling it. This is not possible with a PET scanner because the data are *acquired* discretely. This means that the projections are sampled before any bandlimiting can be done, although a low-pass Hamming type filter will help to reduce any resultant aliasing. Instead of first low-pass filtering the projections, since the filtering operations are linear it is equivalent to low-pass filter the reconstruction filter,$f_{i,j;m}$, and then use this composite filter on the projections. This shifts some of the computational burden from the reconstruction program to the filter generation program; thus saving time during an image reconstruction.

To construct the two dimensional low-pass filter, the most straightforward method is to calculate the equivalent one dimensional filter and use it to form a rotationally symmetric two dimensional filter [37] that has a frequency space form of [75],

$$W(f) = \text{rect}\left(\frac{f}{2f_n}\right) \cdot \frac{1}{2}\left(1 + \cos\left(\frac{\pi f}{f_n}\right)\right). \tag{4.2}$$

where f_n is the Nyquist frequency limit given by $f_n = 1/(2\Delta)$, Δ is the sampling interval, and the rect() function is defined on page 33. The first term of the RHS of equation 4.2 is the low-pass filter term that abruptly cuts off any frequency greater than f_n to avoid aliasing. The second term on the RHS is the Hamming window function that causes $W(f)$ to roll off smoothly to zero as $f \to f_n$, in order to reduce rippling effects in object space [17], and may also help to reduce aliasing that is already present. Although the choice of the window function is arbitrary, there is little difference between the effects of windowing functions with similar functional forms. To find the object space version of equation 4.2, it is straightforward to inverse Fourier transform $W(f)$ to get,

$$w(x) = f_n\left(\text{sinc}(2f_nx) + 1/2\text{sinc}(2f_nx - 1) + 1/2\text{sinc}(2f_nx + 1)\right)$$

and since the pixels have unit size, $\Delta = 1$ and $f_n = 1/2$. To make the filter two dimensional via rotational symmetry, x is replaced with $l = |\vec{l}|$,

$$w(l) = \frac{1}{2}\text{sinc}(l) + \frac{1}{4}\text{sinc}(l-1) + \frac{1}{4}\text{sinc}(l+1) \tag{4.3}$$

This low-pass filter is used to filter each of the two dimensional reconstruction filters.

4.2 Results

An ideal three dimensional reconstruction of the sphere would return an object that could be described by equation 4.1. Since a band-limited reconstruction is being performed, sharp boundaries cannot be recovered. With this in mind the expected ideal reconstruction is an object that has uniform density for $|\vec{x}| \leq R - \delta/2$, is zero for $|\vec{x}| \leq R + \delta/2$, and has a smooth transition between the two regions over the range $R - \delta/2 \leq |\vec{x}| \leq R + \delta/2$. The width, δ, of the transition region depends on the nature of the low-pass filter used (equation 4.2).

A histogram plot of the x-y plane of the band-limited reconstruction, $\tilde{\rho}(\vec{x})$, of the sphere is shown in figure 4.1. The z-x plane histogram is shown in figure 4.2. Values

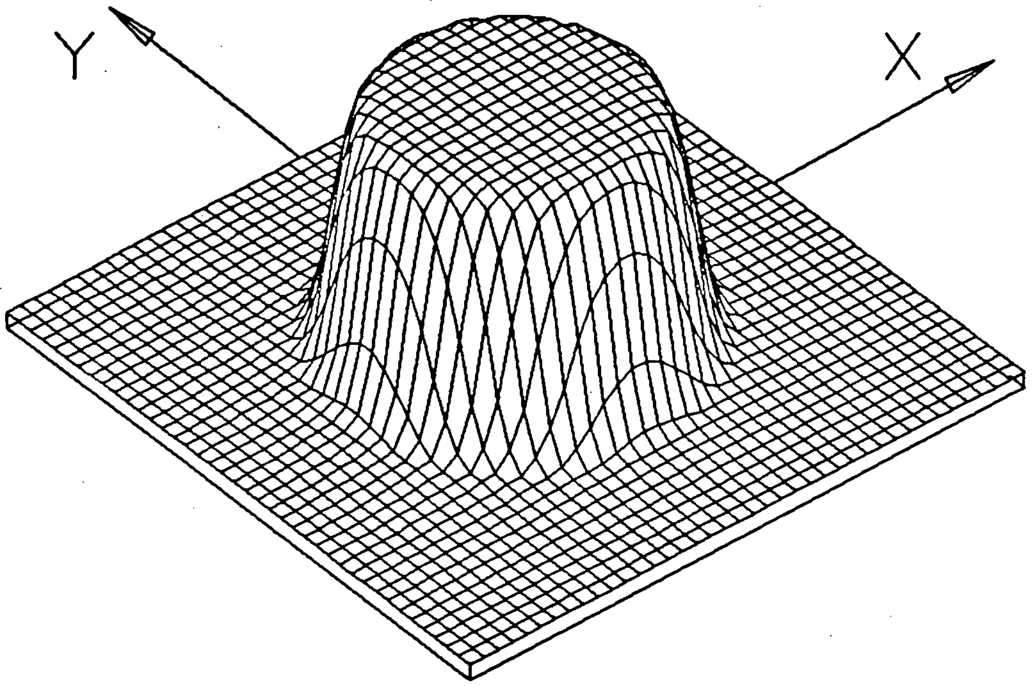

Figure 4.1: The x-y plane of $\tilde{\rho}(\vec{x})$.

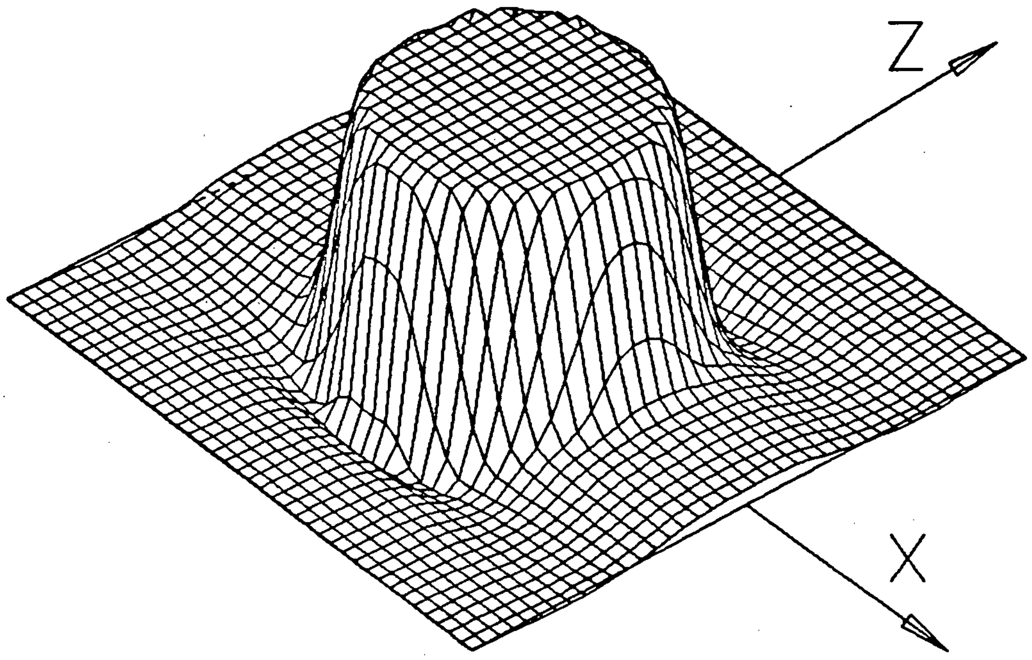

Figure 4.2: The z-x plane of $\tilde{\rho}(\vec{x})$.

of $\tilde{\rho}(\vec{x})$ along the x-axis are shown in figure 4.3 and the values along the z-axis are shown in figure 4.4.

From figure 4.3 it can be seen that the transition width is $\delta \simeq 4$. The values on the x-y plane in figure 4.1 are seen to be rotationally symmetric so that the values of $\tilde{\rho}(\vec{x})$ along any line in the x-y plane passing though the origin are identical to those shown in figure 4.3. The value of $\tilde{\rho}(\vec{x})$ does not go completely to zero for $|\vec{x}| > 10$ and this is a consequence of equation 3.28. Since the reconstruction filter is not infinite in extent, the filter will have a slightly non-zero value at zero frequency that leads to an overall offset in the reconstructed image. In all other respects the results shown in figures 4.1 and 4.3 are as expected.

In figures 4.2 and 4.4 there is a deviation from circular symmetry. The largest manifestation of this lack of symmetry is a 17% undershoot of the reconstructed values on the z-axis, just outside the radius of the sphere. This undershoot is an unexpected artifact that is also believed due to the manner of implementation of equation 3.28, because the contributions to $f_{0,0;}(\theta_T)$ from $f_{i,j;}(\theta_T)$ outside the 41×41 range of the filter were neglected. Because of this, the undershoot artifact is believed to be an implementation error rather than a theoretical one. This is borne out by the results of chapter 5, where it is shown that the algorithm is functionally equivalent to analytic Fourier transform methods that do not show such an artifact.

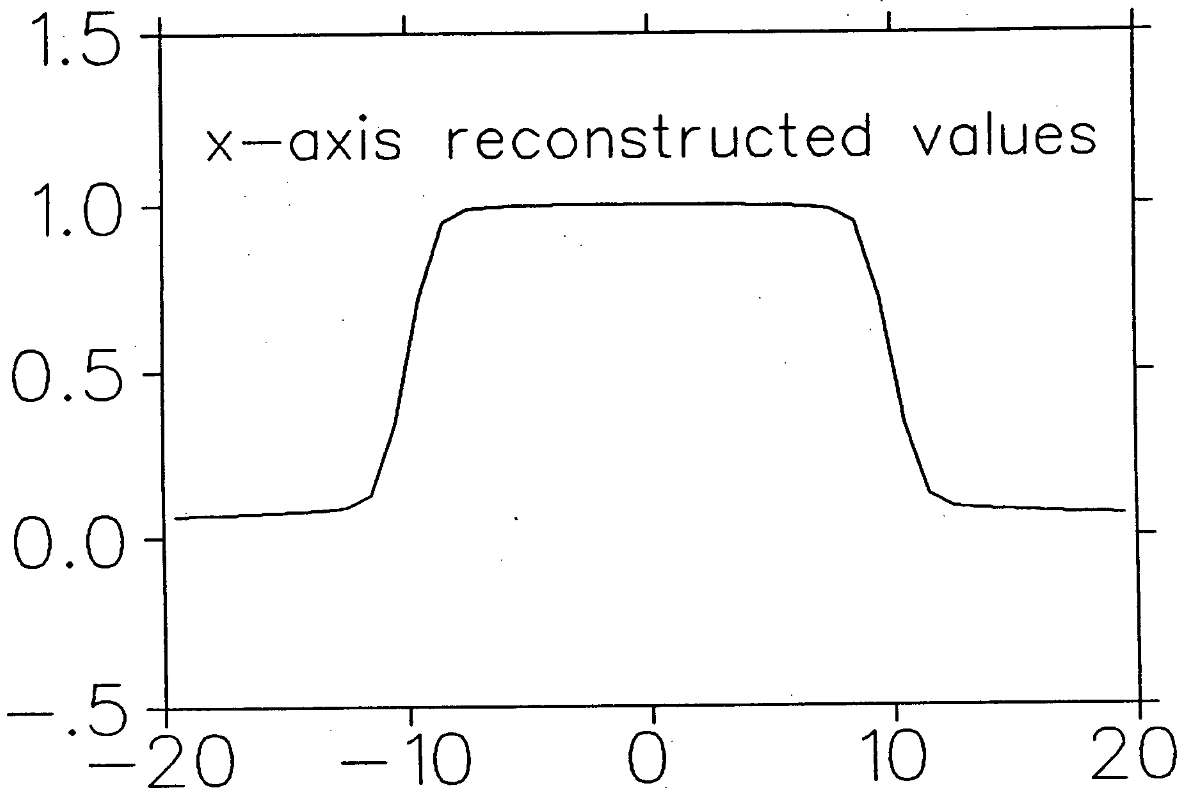

Figure 4.3: Values of $\tilde{\rho}(\vec{x})$ along the x-axis

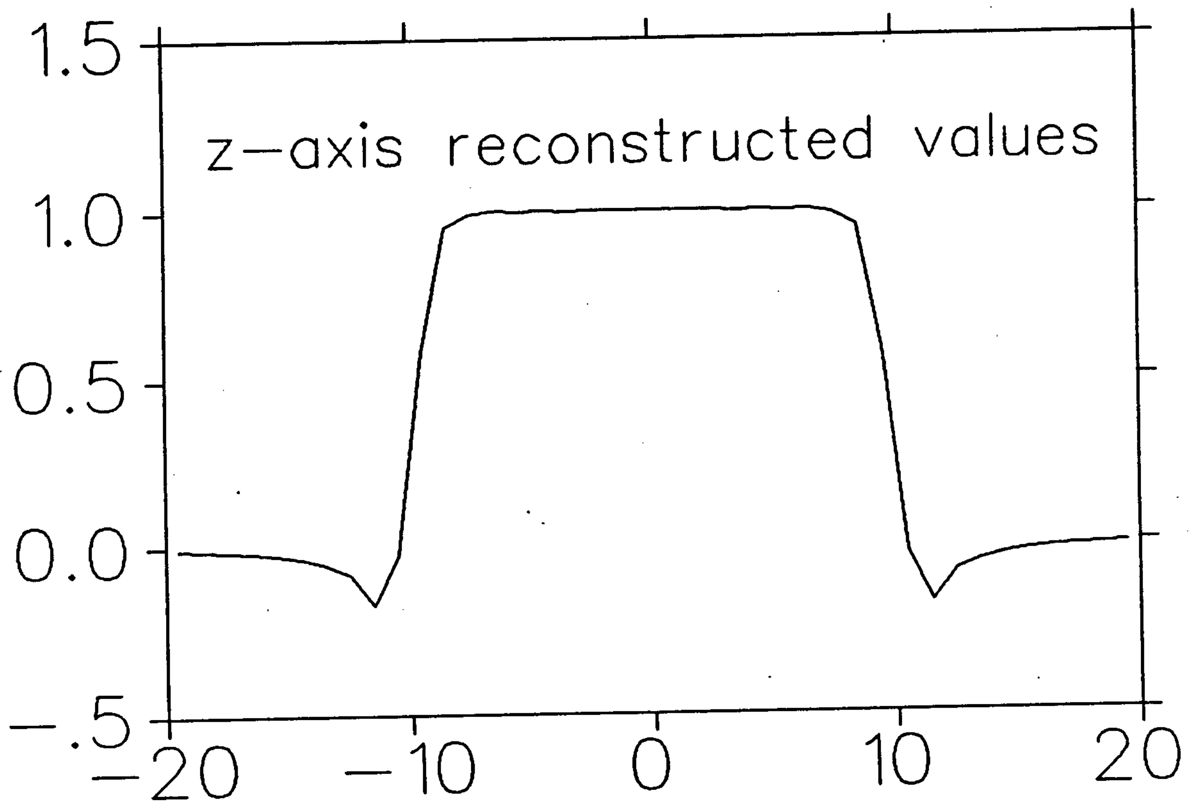

Figure 4.4: Values of $\tilde{\rho}(\vec{x})$ along the z-axis

Chapter 5

Equivalence to Fourier Methods

The three dimensional reconstruction algorithm in chapter 3 is derived in, and only operates in, object space. Previously developed analytic three dimensional image reconstruction methods rely on Fourier transform methods [97,70,21,104,95,26,77,71,22,86,76]. In order to see how the object space reconstruction filter compares to the earlier frequency space filters, the two dimensional object space filter is Fourier transformed into its frequency space form.

5.1 Fourier Transform of the Reconstruction Filter

To begin, $F(R, \Phi; \theta_\tau)$ is defined as the two dimensional Fourier transform of the filter function, $f(l, \theta_l; \theta_\tau)$, that is defined by equation 3.14 where R and Φ are the polar coordinates on the transformed plane.

$$F(R, \Phi; \theta_\tau) \equiv \mathcal{F}_{2D}\left\{f(l, \theta_l; \theta_\tau)\right\}.$$

Now since for any two dimensional function $g(r, \theta)$ [7],

$$\mathcal{F}_{2D}\left\{\nabla^2 g(r, \theta)\right\} = -4\pi^2 r^2 \mathcal{F}_{2D}\left\{g(r, \theta)\right\}$$

the Fourier transform of equation 3.13 is given by

$$F(R,\Phi;\theta_r) = R^2 \mathcal{F}_{2D}\left\{\frac{1}{lL(\theta_l;\theta_r)}\right\}$$

and writing out the transform explicitly in polar coordinates [7],

$$F(R,\Phi;\theta_r) = R^2 \int_0^{2\pi} d\theta_l \int_0^{\infty} l\,dl\,\frac{e^{-2\pi i R l \cos(\theta_l - \Phi)}}{lL(\theta_l;\theta_r)}. \qquad (5.1)$$

Because of the form of the function $L(\theta_l;\theta_r)$ in equation 3.12, it does not seem possible to directly perform the integrations of equation 5.1. Instead, the propeties of the Dirac-delta function are used to simplify the evaluation of $F(R,\Phi;\theta_r)$. To start with, equation 5.1 is rewritten as

$$F(R,\Phi;\theta_r) = R^2\left[\int_0^{\pi} d\theta_l + \int_{\pi}^{2\pi} d\theta_l\right]\int_0^{\infty} dl\,\frac{e^{-2\pi i R l \cos(\theta_l - \Phi)}}{L(\theta_l;\theta_r)},$$

so

$$F(R,\Phi;\theta_r) = R^2 \int_0^{\pi} d\theta_l \int_0^{\infty} dl\left[\frac{e^{-2\pi i R l \cos(\theta_l - \Phi)}}{L(\theta_l;\theta_r)} + \frac{e^{-2\pi i R l \cos(\theta_l + \pi - \Phi)}}{L(\theta_l + \pi;\theta_r)}\right].$$

From equation 3.12 it can be seen that $L(\theta_l + \pi;\theta_r) = L(\theta_l;\theta_r)$, and since $\cos(\theta_l + \pi - \Phi) = -\cos(\theta_l - \Phi)$, the second integrand above can be rewritten so that,

$$F(R,\Phi;\theta_r) = R^2 \int_0^{\pi} d\theta_l \int_0^{\infty} dl\left[\frac{e^{-2\pi i R l \cos(\theta_l - \Phi)}}{L(\theta_l;\theta_r)} + \frac{e^{2\pi i R l \cos(\theta_l - \Phi)}}{L(\theta_l;\theta_r)}\right].$$

Now by setting $l \to -l$ in the second integrand, so that $dl \to -dl$

$$\begin{aligned} F(R,\Phi;\theta_r) &= R^2 \int_0^{\pi} d\theta_l \frac{1}{L(\theta_l;\theta_r)} \\ &\quad \cdot\left[\int_0^{\infty} dl\, e^{-2\pi i R l \cos(\theta_l - \Phi)} - \int_0^{-\infty} dl\, e^{-2\pi i R l \cos(\theta_l - \Phi)}\right]. \end{aligned}$$

Upon reversing the limits in the last integral and combining the two integrations with respect to l, the result is,

$$F(R,\Phi;\theta_r) = R^2 \int_0^{\pi} d\theta_l \frac{1}{L(\theta_l;\theta_r)} \int_{-\infty}^{\infty} dl\, e^{-2\pi i R l \cos(\theta_l - \Phi)}. \qquad (5.2)$$

Recalling the integral form of the Dirac-delta function defined by [25]

$$\delta(x) = \int_{-\infty}^{\infty} dt\, e^{-2\pi i x t}.$$

This means,

$$\int_{-\infty}^{\infty} dl\, e^{-2\pi i R l \cos(\theta_l - \Phi)} = \frac{1}{|R|}\delta(\cos(\theta_l - \Phi)). \tag{5.3}$$

where R is the polar radial coordinate so $|R| = R$. The term $\delta(\cos(\theta_l - \Phi))$ can be expanded to a simpler form by the relationship [25],

$$\delta\left(f(x)\right) = \sum_i \frac{1}{|f'(x_i)|}\delta(x - x_i),$$

where $f'(x)$ is the derivative of $f(x)$ and x_i are the simple zeros of $f(x)$. Note that if any of the zeros of $f(x)$ are greater than first order, then the expression $\delta\left(f(x)\right)$ has no meaning. For $\theta_l \in [0, \pi]$, the zeros of $\cos(\theta_l - \Phi)$ are $\theta_l = \Phi \pm \pi/2$, and since $|-\sin(\pm\pi/2)| = 1$,

$$\delta(\cos(\theta_l - \Phi)) = \delta(\theta_l - \Phi - \pi/2) + \delta(\theta_l - \Phi + \pi/2),$$

and equation 5.3 becomes

$$\int_{-\infty}^{\infty} dl\, e^{-2\pi i R l \cos(\theta_l - \Phi)} = \frac{1}{R}\left(\delta(\theta_l - \Phi - \pi/2) + \delta(\theta_l - \Phi + \pi/2)\right).$$

Using the above result in equation 5.2 yields

$$F(R, \Phi; \theta_\tau) = R\int_0^{\pi} d\theta_l\, \frac{\delta(\theta_l - \Phi - \pi/2)}{L(\theta_l; \theta_\tau)} + R\int_0^{\pi} d\theta_l\, \frac{\delta(\theta_l - \Phi + \pi/2)}{L(\theta_l; \theta_\tau)}. \tag{5.4}$$

Recalling that [54],

$$\int_a^b dx\, f(x)\delta(x - x_0) = \begin{cases} f(x_0), & \text{if } x_0 \in [a, b] \\ 0, & \text{if } x_0 \notin [a, b] \end{cases},$$

the first integral in equation 5.4 is

$$\int_0^{\pi} d\theta_l\, \frac{\delta(\theta_l - \Phi - \pi/2)}{L(\theta_l; \theta_\tau)} = \begin{cases} \dfrac{1}{L(\Phi + \pi/2; \theta_\tau)}, & \text{if } -\pi/2 \le \Phi \le \pi/2 \\ 0, & \text{otherwise.} \end{cases}$$

Similarly for the second integral in equation 5.4,

$$\int_0^\pi d\theta_l \frac{\delta(\theta_l - \Phi + \pi/2)}{L(\theta_l;\theta_\tau)} = \begin{cases} \dfrac{1}{L(\Phi - \pi/2;\theta_\tau)}, & \text{if } \pi/2 \le \Phi \le 3\pi/2 \\ 0, & \text{otherwise} \end{cases}$$

Now once again using the relationship $L(\theta_l + \pi; \theta_\tau) = L(\theta_l; \theta_\tau)$, the above two results can be combined to get,

$$F(R,\Phi;\theta_\tau) = \frac{R}{L(\Phi - \pi/2;\theta_\tau)}$$

Substituting in equation 3.12 for $L(\theta_l;\theta_\tau)$ yields the frequency space form of the two dimensional reconstruction filter,

$$F(R,\Phi;\theta_\tau) = \begin{cases} \frac{R}{\pi}, & \text{if } \sin\Phi \ge \cos\psi_{\text{min}}/\sin\theta_\tau \\ \dfrac{R}{2\arcsin\left(\dfrac{\sin\psi_{\text{min}}}{\sqrt{1-\sin^2\Phi\sin^2\theta_\tau}}\right)}, & \text{otherwise} \end{cases} \qquad (5.5)$$

5.2 Comparison to Frequency Space Filters

To see how the frequency space form of the two dimensional reconstruction filter compares to previously derived frequncy space filters, it is necessary to briefly review Fourier reconstruction methods. These fall into two classes, two dimensional filtering and three dimensional filtering. In two dimensional filtering the algorithm is as follows; the projections are Fourier transformed to frequency space and are then filtered by multiplying them by the appropriate two dimensional filter. At this point the three dimensional image can be reconstructed in two ways, either by inverse transforming the filtered projections to object space and then backprojecting them, or by first *placing* the filtered projections into the reconstruction volume (in frequency space), and then inverse transforming the three dimensional result [95].

The equivalence of these last two steps is a consequence of the Fourier-slice theoem that states; if $\Xi(u,v,w)$ is the three dimensional Fourier transform of $\rho(x,y,z)$ such that $\Xi(u,v,w) = \mathcal{F}_{3D}\{\rho(x,y,z)\}$ and if $P(l_u,l_v;\theta_\tau,\phi_\tau) = \mathcal{F}_{2D}\{\rho(l_x,l_y;\theta_\tau,\phi_\tau)\}$ is the two dimensional filtered projection, where the relationships between (l_u,l_v) and (u,v,w) are analogous to those between (l_x,l_y) and (x,y,z) in equation 3.8, then

$$P(l_u,l_v;\theta_\tau,\phi_\tau) = \Xi(-l_u\sin\phi_\tau - l_v\cos\phi_\tau\cos\theta_\tau, l_u\cos\phi_\tau - l_v\sin\phi_\tau\cos\theta_\tau, l_v\sin\theta_\tau).$$

Stated in words, the Fourier transform of a projection of an object is identical to a slice, at the same orientation as the projection, through the three dimensional Fourier transform of the object [71].

For three dimensional filtering methods the order of filtering and backprojection is reversed. First the projection data is backprojected into the three dimensional reconstruction volume, where it is Fourier transformed into frequency space and then filtered by multiplication with a three dimensional filter. The result is then inverse transformed into object space to become the three dimensional reconstructed image.

For all (except possibly one) of the published frequency space methods the two dimensional frequency space filter can be related to a three dimensional version by the Fourier-slice theorem. In other words the appropriate two dimensional frequency-space filter is a slice at the appropriate angle through the origin of the three dimensional filter. The possible exception to this is the composite filter derived by Ra et al. [77], where the analytic filter form is different from all other two dimensional filters [70,26,86].

To construct a three dimensional filter from equation 5.5 it is only necessary to incorporate the spherical projection angle, θ_τ, and the planar polar coordinates R and Φ into a three dimensional frequency space coordinate system. From figure 5.1 it is seen that $\cos\Theta = \sin\Phi\sin\theta_\tau$ [1], where Θ is the spherical polar coordinate. Now since $\sqrt{1-\sin^2\Phi\sin^2\theta_\tau} = |\sin\Theta|$ and R is the three dimensional radial coordinate,

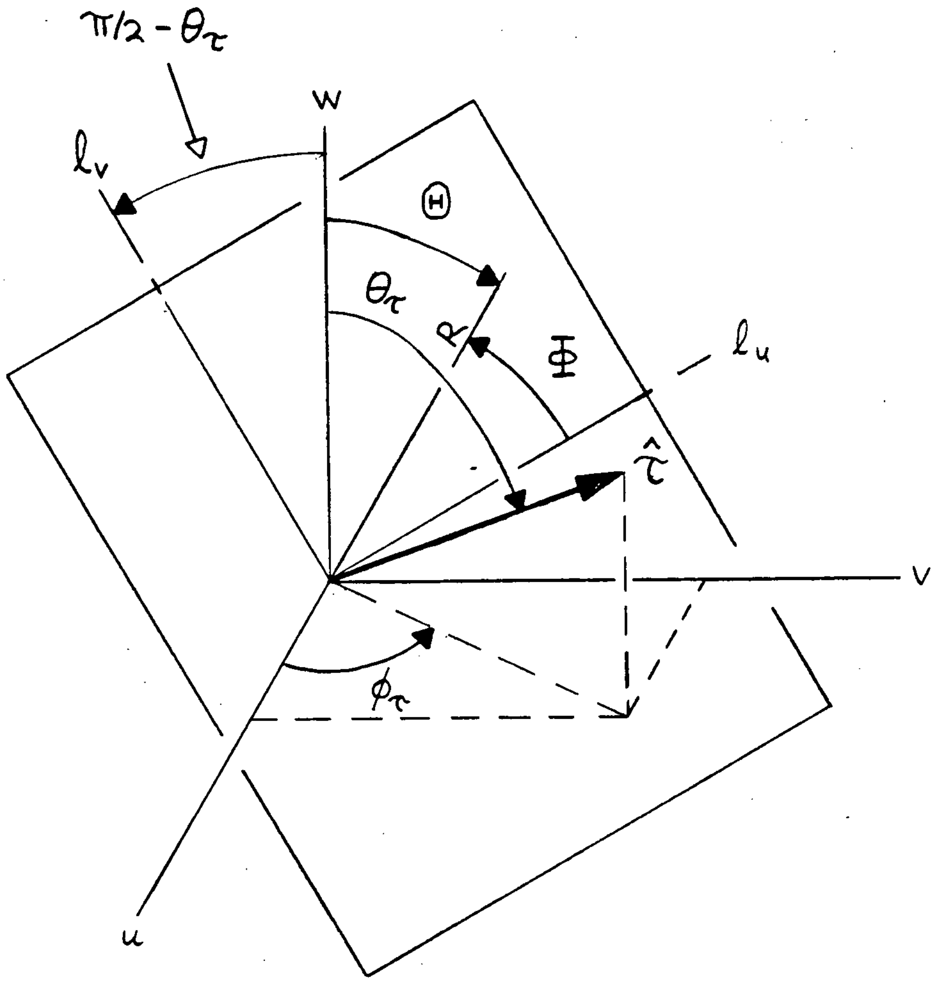

Figure 5.1: Two dimensional and three dimensional coordinates in frequency space

the three dimensional version of the frequency space filter equation 5.5 is,

$$F(R,\Theta) = \begin{cases} \frac{R}{\pi}, & \text{if } |\Theta| \geq \Phi \\ \dfrac{R}{2\arcsin\left(\dfrac{\sin\psi_{\min}}{|\sin\Theta|}\right)}, & \text{if } |\Theta| < \Phi \end{cases} \tag{5.6}$$

This is identical to the frequency space filter derived independently by Pelc [70], Colsher [26], and Schorr et al. [86]. The filter derived by Pelc requires some minor trigonometric substitutions in order to show that it is the same as equation 5.6.

Since equation 5.6 shows that the Recovery Operator of Orlov [68] is equivalent to frequency space reconstruction methods, it also independently confirms the validity of the three dimensional object space image reconstruction algorithm.

Summary

There is one major advantage to building a volume imaging PET system; that of increasing sensitivity while keeping resolution constant. The two principal problems that such a system has to overcome are parallax error and the difficulty of incorporating all the collected data into the reconstructed image. The first problem has been solved by a new position-sensitive detector developed at TRIUMF that has the ability to measure depth-of-interaction information. A proposed PET system incorporates 15 of these $15 \times 15 \times 2.5$ cm detector units into a cylindrical ring to form a volume-imaging scanner that has the ability to detect cross-plane gamma rays.

Previous work in three dimensional image reconstruction that can use all of the cross-plane information mostly falls into two classes; iterative reconstruction and analytic reconstruction using Fourier transforms. Iterative algorithms have the two advantages of being able to use all cross-plane data and the ability to use apriori knowledge of the object being reconstructed. The primary disadvantage of iterative algorithms is that they require too much computation time.

The Fourier transform algorithms require much less computational time than iterative algorithms. By using Fourier transforms, the requirement of a spatially-invariant scanner PSF means that the amount of collected data that can actually be used is limited by the relative geometries of the PET scanner and the object in its FOV. In order to make Fourier transform algorithms computationally efficient, they employ FFT filtering methods that add more computational noise than convolution filtering

methods.

The new algorithm presented in this thesis was based on an extension of Orlov's Recovery Operator. In its original form the Recovery Operator could only reconstruct objects for which there were projections in all directions. To put it into a form suitable for use by the new PET scanner, it was first reformulated as a FBP method, analogous to standard two dimensional image reconstruction, and then extended to work with incomplete three dimensional coverage.

By analyzing the angular limits of integration for the backprojection part of the algorithm, it was shown that the new algorithm can use the same amount of cross-plane data as the Fourier transform algorithms. An extension to the algorithm has been proposed by my colleagues and me [51,81] that allows the use of all the cross-plane data. This extension is suited for three dimensional image reconstruction algorithms that operate in object space.

In the discrete form of the algorithm, a band-limited form was created that dealt with the extensive singularities present in the analytic form of the reconstruction filter. The calculation of the digital filter is complicated and time consuming, but it only has to be calculated once and then stored in computer memory for retrieval during the reconstruction of an image.

The new algorithm was tested by reconstructing a sphere from perfect projection data that was low-pass filtered with a Hamming window,. The sphere was faithfully reconstruced except for a 17% undershoot in the reconstructed value on the z-axis just past the edge of the sphere. This artifact is believed due to the method of normalizing the filter at its centre. This algorithm has not yet been tested with data that has noise added.

To demonstrate that the overshoot artifact is not an error in analytic form of the algorithm, a result was derived showing for the first time the functional equivalence of

Orlov's Recovery Operator and the Fourier transform algorithms.

Bibliography

[1] *CRC Standard Mathematical Tables.* The Chemical Rubber Company, seventeeth edition, 1969.

[2] Medical imaging for the masses. Staff Article in *New Scientist*, page 33, 7 May 1987.

[3] M D Altschuler, Y Censor, G T Herman, A Lent, R M Lewitt, S N Srihar, H Tuy, and J K Udupa. Mathematical apects of image reconstructions from projections. In L N Kanal and A Rosenfeld, editors, *Progress in Pattern Recognition*, pages 323–375, North-Holland Publishing Company, 1981.

[4] M D Altschuler and G T Herman. Fully-three-dimensional image reconstruction using series expansion methods. In A B Brill, R R Price, W J Mclain, and M W Landy, editors, *Fifth International Conference on Information Processing in Medical Imaging*, pages 125–140, January 1978. Held at Vanderbuilt University, Nashville, Tennessee. June 27 - July 1 1977. Published by the Biomedical Computing Technology Information Centre at Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830.

[5] J E Bateman, J F Connolly, G J Stephenson, and A C Flesher. The Rutherford Appleton laboritory's mark I multiwire proportional chamber. *Nuclear Instruments and Methods in Physics Research*, 225:209–231, 1984.

[6] M V Berry and D F Gibbs. The interpretation of optical projections. *Proceedings of the Royal Society of London, Series A*, 314:143–152, 1970.

[7] R N Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, second edition, 1978.

[8] R N Bracewell and A C Riddle. Inversion of fan-beam scans in radio astronomy. *Astophysics Journal*, 150:427–434, 1967.

[9] R A Brooks and G DiChiro. Principles of computer assisted tomography (CAT) in radiographic and radioisotopic imaging. *Physics in Medicine and Biology*, 32(5):689–732, 1976.

[10] T F Budinger and G T Gulberg. Three-dimensional reconstruction in nuclear medicine emission imaging. *IEEE Transactions on Nuclear Science*, NS-21(3):2–20, 1974.

[11] C A Burnham, D E Kaufman, D A Chesler, C W Stearns, D R Wolfson, and G L Brownell. Cylindrical PET detector design. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[12] D B Calne, J W Langston, W R W Martin, A J Stoessl, T J Ruth, M J Adam, B D Pate, and M Schulzer. Positron emission tomography after MPTP: Observations relating to the cause of Parkinson's disease. *Nature*, 317:246–248, 1985.

[13] R E Carson. A maximum likelyhood method for region-of-intrest evaluation in emission tomography. *Journal of Computer Assisted Tomography*, 10(4):654–663, 1986.

[14] M E Casey and E J Hoffman. Quantitation in positron emission tomography:7. A technique to redice noise in accidental coincidence measurements and coincidence efficiency claibration. *Journal of Computer Assisted Tomography*, 10(5):845–850, 1986.

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

[16] L T Chang. A method for attenuation correction in radionuclide computed tomography. *IEEE Transactions on Nuclear Science*, NS-25(1):638–643, 1978.

[17] D A Chesler and S J Riederer. Ripple suppression during reconstruction in transverse tomography. *Physics in Medicine and Biology*, 20(4):632–636, 1975.

[18] D A Chesler, S J Riederer, and N J Pelc. Noise due to photon counting statistics in computed X-ray tomography. *Journal of Computer Assisted Tomography*, 1(1):64–74, 1977.

[19] Z H Cho and J R Burger. Construction restoration and enhancement of 2 and 3-dimensional images. *IEEE Transactions on Nuclear Science*, NS-24(2):886–899, 1977.

[20] Z H Cho, J K Chan, E L Hall, R P Kruger, and D G McCaughy. A comparitive study of 3-d image reconstruction algorithms with reference to number of projections and noise filtering. *IEEE Transactions on Nuclear Science*, NS-22:344–358, February 1975.

[21] Z H Cho, J B Ra, and S K Hilal. True three-dimensional reconstruction (TTR) - Application of algorithm toward full utilization of oblique rays. *IEEE Transactions on Medical Imaging*, MI-2(1):6–18, 1983.

[22] G Chu and K-C Tam. Three-dimensional imaging in the positron camera using fourier techniques. *Physics in Medicine and Biology*, 22(2):245–265, 1977.

[23] R Clack, D Townsend, and M Defrise. An algorithm for three-dimensional reconstruction incorporating cross-plane rays. Submitted to: *IEEE Transactions on Medical Imaging*, December 1987.

[24] R Clack, D Townsend, and A Jeavons. Increased sensitivity and field of view for a rotating positron camera. *Physics in Medicine and Biology*, 29(11):1421–1431, 1984.

[25] C Cohen-Tannoudji, B Diu, and F Laloë. *Quantum Mechanics*, pages 1468–1479. Volume two, Wiley-Interscince, 1978.

[26] J G Colsher. Fully three-dimensional positron emission tomography. *Physics in Medicine and Biology*, 25(1):103–105, 1980.

[27] M E Daube-Witherspoon and G Muehllehner. An iterative image space reconstruction algorithm suitable for volume ECT. *IEEE Transactions on Medical Imaging*, MI-5(2):61–66, 1986.

[28] M E Daube-Witherspoon and G Muehllehner. Treatment of axial data in three-dimensional PET. *Journal of Nuclear Medicine*, 28(11):1717–1724, 1987.

[29] A R de Pierro. On the convergence of the iterative image space reconstruction algorithm for volume ECT. *IEEE Transactions on Medical Imaging*, MI-6(2):174–175, 1987.

[30] D J de Rosier and A Klug. Reconstruction of three dimensional structures from electron micrographs. *Nature*, 217:130–134, 1968.

[31] M Defrise, S Kuijk, and F Deconinick. A new three dimensional reconstruction method for positron cameras using plane detectors. *Physics in Medicine and Biology*, 33(1):43–51, 1988.

[32] R V Denton, B Friedlander, and A J Rockmore. Direct three-dimensional image reconstruction from divergent rays. *IEEE Transactions on Nuclear Science*, NS-26(5):4695–4703, 1979.

[33] S E Derenzo. Mathematical removal of positron range blurring in a high resolution tomography. *IEEE Transactions on Nuclear Science*, NS-33(1):565–569, 1986.

[34] S E Derenzo. Monte carlo calculations of the detection efficiecy of arrays of NaI(Tl), BGO, CsF, Ge, and plastic detectors for 511 kev photons. *IEEE Transactions on Nuclear Science*, NS-28(1):131–136, 1981.

[35] S E Derenzo. Recent developments in positron emission tomograph (pet) instrumentation. In *SPIE vol.671: Physics and Engineering of Computerized Multidimensional Imaging and Procesing*, pages 232–243, 1986.

[36] S E Derenzo, R H Huesman, J L Cahoon, A B Geyer, W W Moses, D C Uber, T Vuletich, and T F Budinger. A positron tomograph with 600 BGO crystals and 2.6 mm resolution. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[37] D E Dudgeon and R M Mersereau. *Multidimensional Digital Signal Processing*. Prentice-Hall, 1984.

[38] L Eriksson and Z H Cho. A simple absorption correction in positron (annihilation gamma coincidence detection) transverse axial tomography. *Physics in Medicine and Biology*, 21(3):429–433, 1976.

[39] J S Flower and A P Wolf. Positron emitter-labelled compounds: Priorities and problems. In M E Phelps, J C Mazziotta, and H R Schelbert, editors, *Positron Emission Tomography and Autoradiography*, pages 391–492, Raven Press, 1986.

[40] G Harauz and F P Ottensmeyer. Interpolation in computing forward projections in direct three-dimensional reconstruction. *Physics in Medicine and Biology*, 28(12):1419–1429, 1983.

[41] M R Hayden, W R W Martin, A J Stoessl, C Clark, S Hollenberg, M J Adam, W Ammann, R Harrop, J Rogers, T Ruth, C Sayre, and B D Pate. Positron emission tomography in the early diagnosis of Huntington's disease. *Neurology*, 36(7):888–894, 1986.

[42] G T Herman. *Image Reconstruction from Projections.* Academic Press, 1980.

[43] E J Hoffman, M Dahlbom, A R Ricci, and I N Weinberg. Examination of the role of detection systems in quantitation and image quality in PET. *IEEE Transactions on Nuclear Science*, NS-33(1):420–424, 1986.

[44] E J Hoffman and M E Phelps. Positron emission tomography: Principles and quantitation. In M E Phelps, J C Mazziotta, and H R Schelbert, editors, *Positron Emission Tomography and Autoradiography*, pages 237–286, Raven Press, 1986.

[45] B K P Horn. Density reconstruction using arbitrary ray-sampling schemes. *Proceedings of the IEEE*, 66(5):551–562, 1978.

[46] S-C Huang, E J Hoffman, M E Phelps, and D E Kuhl. Quantitation in positron emission tomography:3. Effect of sampling. *Journal of Computer Assisted Tomography*, 4(6):819–826, 1980.

[47] S-C Huang and M E Phelps. Principles of tracer kinetic modeling in positron emission tomography and autoradiography. In M E Phelps, J C Mazziotta, and H R Schelbert, editors, *Positron Emission Tomography and Autoradiography*, pages 287–390, Raven Press, 1986.

[48] R H Huesman, S E Derenzo, J L Cahoon, A B Geyer, W W Moses, D C Uber, T Vuletich, and T F Budinger. Orbiting transmission source for positron tomography. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[49] A Imiya and H Ogawa. Direct reconstruction of three dimensional image from its line integrals. In *SPIE vol.671: Physics and Engineering of Computerized Multidimensional Imaging and Procesing*, pages 42–49, 1986.

[50] L Kaufman. Implementing and accelerating the EM algorithm for positron emission tomography. *IEEE Transactions on Medical Imaging*, MI-6(1):37–51, 1987.

[51] P E Kinahan, J G Rogers, R Harrop, and R R Johnson. Three-dimensional image reconstruction in object space. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[52] R M Lewitt. Reconstruction algorithms: transform methods. *Proceedings of the IEEE*, 71(3):390–408, 1983.

[53] R M Lewitt and G Muehllehner. Accelerated iterative reconstruction for positron emission tomography based on the EM algorithm for maximum likelyhood estimation. *IEEE Transactions on Medical Imaging*, MI-5(1):16–22, 1989.

[54] M J Lighthill. *Introduction to Fourier Analysis and Generalized Functions.* Cambridge University Press, 1958.

[55] C B Lim, A Cheng, D P Boyd, and R S Hatner. A 3-d iterative reconstruction method for stationary planar positron cameras. *IEEE Transactions on Nuclear Science*, NS-25(1):196–201, 1978.

[56] J Llacer, A Veklerov, and E Veklerov. Towards a practical implementation of the MLE algorithm for positron emission tomography. *IEEE Transactions on Nuclear Science*, NS-33(1):468–477, 1986.

[57] A Macovski. Physical problems of computerized tomography. *Proceedings of the IEEE*, 71(3):373–378, 1983.

[58] D A Mankoff and G Muehllehner. Performance of positron imaging systems as a function of energy threshold and shielding depth. *IEEE Transactions on Medical Imaging*, MI-3(1):18–24, 1984.

[59] D A Mankoff, G Muehllehner, and J S Karp. The effect of detector performance on high countrate PET imaging with a tomograph based on positron-sensitive detectors. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[60] B M Mazoyer, M S Roos, and R H Huesman. Dead time correction and counting statistics for positron tomograph. *Physics in Medicine and Biology*, 30(5):385–399, 1985.

[61] J C Mazziotta and M E Phelps. Positron emission tomography sudies of the brain. In M E Phelps, J C Mazziotta, and H R Schelbert, editors, *Positron Emission Tomography and Autoradiography*, pages 493–579, Raven Press, 1986.

[62] P L McGeer, H Kamo, R Harrop, D K B Li, H Tuokko, E G McGeer, M J Adam, W Ammann, B L Beattie, D B Calne, W R W Martin, B D Pate, J G Rogers, T J Ruth, C I Sayre, and A J Stoessl. Positron emission tomography in patients with

clinically diagnosed Alzheimer's disease. *Canadian Medical Association Journal*, 134:597–607, 1986.

[63] R M Mersereau. Direct fourier transform techniques in 3-d image reconstruction. *Computers in Medicine and Biology*, 6:247–258, 1976.

[64] G Muehllehner and J S Karp. Positron emission tomography imaging - technical considerations. *Seminars in Nuclear Medicine*, XVI(1):35–50, 1980.

[65] G Muehllehner, J S Karp, D A Mankoff, D Beerbohm, and C E Ordonez. Design and performance of a new positron emission tomograph. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[66] O Nalcioglu and Z H Cho. Reconstruction of 3-d objects from cone bean projections. *Proceedings of the IEEE*, 66(11):1584–1585, 1978.

[67] S S Orlov. Theory of three dimensional reconstruction. I conditions for a complete set of projections. *Soviet Physics Crystallography*, 20(2):312–314, 1976.

[68] S S Orlov. Theory of three dimensional reconstruction. II the recovery operator. *Soviet Physics Crystallography*, 20(4):429–433, 1976.

[69] L M Pecora. 3d tomographic reconstruction from 2d data using spherical harmonics. *IEEE Transactions on Nuclear Science*, NS-34(2):642–650, 1987.

[70] N J Pelc. *A Generalized Filtered Backprojection Algorithm for Three Dimensional Reconstruction.* PhD thesis, Harvard School of Public Health, 1979.

[71] N J Pelc and D A Chesler. Utilization of cross-plane rays for three-dimensional reconstruction by filtered back-projection. *Journal of Computer Assisted Tomography*, 3(3):385–395, 1979.

[72] F C Peyrin. The generalized back projection theorm for cone beam reconstruction. *IEEE Transactions on Nuclear Science*, NS-32(4):1512–1519, 1985.

[73] M E Phelps, E J Hoffman, S-C Huang, and D E Kuhl. Positron emission tomography: present and future design alternatives. In *IAEA-SM-247/92*, 1980.

[74] M E Phelps and J C Maziotta. Positron emission tomography: human brain function and biochemistry. *Science*, 228(4701):799–809, 1985.

[75] W H Press, B P Flannery, S A Teukolsky, and W T Vetterling. *Numerical Recipes*. Cambridge University Press, 1986.

[76] J B Ra and Z H Cho. Generalized true three-dimensional reconstruction algorithm. *Proceedings of the IEEE*, 69(5):668–670, 1981.

[77] J B Ra, C B Lim, Z H Cho, S K Hilal, and J Corell. A true three-dimensional reconstruction algorithm for the spherical positron emission tomograph. *Physics in Medicine and Biology*, 27(1):37–50, 1982.

[78] J Radon. On the determination of functions from their integral values along certain manifolds. *IEEE Transactions on Medical Imaging*, MI-5(5):170–176, 1986. This is a translation by P C Parks of the original German text by Johann Radon published in *Berichte der Sächsischen Akadamie der Wissenschaft*, Vol. 69, pp. 262–277, 1917. (Session of April 30,1917).

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

[80] J G Rogers. Testing an improved scintillation camera for PET and SPECT. *IEEE Transactions on Nuclear Science*, NS-33(1):519–522, 1986.

[81] J G Rogers, R Harrop, and P E Kinahan. The theory of three-dimensional image reconstruction for PET. *Physics in Medicine and Biology*, MI-6(3):239–243, September 1987.

[82] J G Rogers, R Harrop, P E Kinahan, N A Wilkinson, P W Doherty, and D P Saylor. Conceptual design of a whole body PET machine. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[83] J G Rogers, D P Saylor, R Harrop, X G Yao, C V M Leitao, and B D Pate. Design of an efficient position sensitive gamma ray detector for nuclear medicine. *Physics in Medicine and Biology*, 31(10):1061–1090, 1986.

[84] A Rosenfeld and A C Kak. *Digital Picture Processing*, chapter 8. Volume I, Academic Press, second edition, 1982.

[85] T Ruth. Private communication.

[86] B Schorr, D Townsend, and R Clack. A general method for three-dimensional filter computation. *Physics in Medicine and Biology*, 28(9):1009–1019, 1983.

[87] L A Shepp and B F Logan. The fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*, NS-21:21–43, 1974.

[88] L A Shepp and Y Vardi. Maximum likelyhood reconstruction for emission tomography. *IEEE Transactions on Medical Imaging*, MI-1(2):113–122, 1982.

[89] L A Shepp, Y Vardi, J B Ra, S K Hilal, and Z H Cho. Maximum likelyhood PET with real data. *IEEE Transactions on Nuclear Science*, NS-31(2):910–912, 1984.

[90] B D Smith. Image reconstruction from cone-beam projections: necessary and sufficient conditions and reconstruction methods. *IEEE Transactions on Medical Imaging*, MI-4(1):14–25, 1985.

[91] L Sokoloff. Cerebral circulation, energy metabolism, and protein synthesis: general characteristics and priciples of measurement. In M E Phelps, J C Mazziotta, and H R Schelbert, editors, *Positron Emission Tomography and Autoradiography*, pages 1–71, Raven Press, 1986.

[92] L Sokoloff. Localization of functional activity in the central nervous system by measurement of glucose utilization with radioactive deoxyglucose. *Journal of Cerebral Blood Flow and Metabolism*, 1:7–36, 1981.

[93] J A Sorenson and M E Phelps. *Physics in Nuclear Medicine*, chapter 20. Grune and Straton, second edition, 1987.

[94] C W Stearns, C A Burnham, D A Chesler, and G L Brownell. Simulation studies for cylindrical positron tomography. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[95] C W Stearns, D A Chesler, and G L Brownell. Three dimensional image reconstruction in the fourier domain. *IEEE Transactions on Nuclear Science*, NS-34(1):374–378, 1987.

[96] C W Stearns, D A Chesler, J E Kirsch, and G L Brownell. Quantitative imaging with the MGH analog ring positron tomograph. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[97] K C Tam, G Chu, V Perez-Mendez, and C B Lim. Three-dimensional reconstruction in planar positron cameras using fourier deconvolution of generalized tomograms. *IEEE Transactions on Nuclear Science*, NS-25(1):152–159, 1978.

[98] E Tanaka. A fast reconstruction algorithm for stationary positron emission tomography based on a modified EM algorithm. *IEEE Transactions on Medical Imaging*, MI-6(2):98–105, 1987.

[99] E Tanaka. Recent progress on single photon and positron emission tomography - from detectors to algorithms. *IEEE Transactions on Nuclear Science*, NS-34(1):313–320, 1987.

[100] M M Ter-Pogossian, M E Raichle, and B E Sobel. Positron-emission tomography. *Scientific American*, 243(4):170+, October 1980.

[101] C J Thompson. The effect of collimation on scatter fraction in multi-slice PET. To appear in: *IEEE Transactions on Nuclear Science*, February 1988.

[102] C J Thompson, A Dagher, D N Lunney, S C Strother, and A C Evans. A technique to reject scattered radiation in PET transmission scans. In *SPIE vol.671: Physics and Engineering of Computerized Multidimensional Imaging and Procesing*, 1986.

[103] D M Titterington. On the iterative image space reconstruction algorithm for ECT. *IEEE Transactions on Medical Imaging*, MI-6(1):52–56, 1987.

[104] D Townsend, R Clack, R Maganini, P Frey, and A Donath. Image reconstruction for a rotating positron tomograph. *IEEE Transactions on Nuclear Science*, NS-30(1):594–600, 1983.

[105] B K Vainshtein and S S Orlov. Theory of the recovery of functions from their projections. *Soviet Physics - Crystallography*, 17(2):213–216, 1972.

[106] A P Wolf. Special characteristics and potential for radiopharmaceuticals for positron emission tomograph. *Seminars in Nuclear Medicine*, XVI(1):2–12,

1981.

Appendix A

PSF Derivation

This appendix contains the derivation of the Point Spread Funtion (PSF) for a rotationally symmetric solid angle detector, such as a rotating parallel-plate PET scanner. This derivation follows that of Orlov [68].

A.1 Introduction

From linear systems theory [7], for the system shown below,

$$\begin{tikzcd} f(\vec{x}) \arrow[r] & \boxed{h(\vec{x})} \arrow[r] & g(\vec{x}) \end{tikzcd}$$

if $f(\vec{x})$[1] is the input to a linear and invariant system that has a PSF given by $h(\vec{x})$, then the output, $g(\vec{x})$, is given by,

$$g(\vec{x}) = f(\vec{x}) * h(\vec{x}). \tag{A.1}$$

[1]In this appendix vectors are denoted by an arrow, and unit vectors are denoted by a caret.

where $*$ represents convolution of the appropriate dimensionality.

For the case of interest, $f(\vec{x})$ is the three-dimensional source density distribution which is being imaged, $g(\vec{x})$ is the backprojection of the measured projections, and $h(\vec{x})$ is the PSF of the solid angle detector which we want to calculate. The backprojection, $g(\vec{x})$, is formed by first taking the two-dimensional projections of $f(\vec{x})$ at different angles and then projecting back the same values (constant along each ray) through a three-dimensional volume. At each point in the volume the values of all the projection rays passing through each point are summed up, which results in the backprojected image $g(\vec{x})$. Our purpose is to find $h(\vec{x})$ such that equation A.1 describes the same operation mathematically.

A.2 Detector Geometry

To model the geometry of a rotating-plate detector, it is convenient to use a unit sphere with the top and bottom end caps removed as shown in figure 3.6

The object to be imaged, $f(\vec{x})$, is located entirely within the detector surface, where ψ is the cutoff angle of the detectors and $\hat{\tau}$ is a unit vector perpendicular to the projection or $\hat{\tau}$-plane. Thus the $\hat{\tau}$-plane is determined by the two spherical coordinate angles (θ, ϕ) and the truncated sphere of figure 3.6 describes all possible orientations of the unit vector $\hat{\tau}$.

The two-dimensional projections, $p(\vec{l}; \hat{\tau})$, are the integrals of $f(\vec{x})$ along the rays parallel to $\hat{\tau}$ so that for each plane $\hat{\tau}$ can be considered to be a parameter. The projection values for a given $\hat{\tau}$ are thus given by

$$p(\vec{l}; \hat{\tau}) = \int_{-\infty}^{\infty} f(\vec{x} + t\hat{\tau})dt \qquad \text{(A.2)}$$

A.3 Derivation of PSF

To construct the backprojection, $g(\vec{x})$, the projection data are integrated over all possible values of $\hat{\tau}$ for $\pi/2 - \psi \leq \theta \leq \pi$.

$$g(\vec{x}) = \int_{\phi=0}^{2\pi} \int_{\theta=\pi/2-\psi}^{\pi/2} p(\vec{l};\hat{\tau}) \sin\theta d\theta d\phi \tag{A.3}$$

Now since $p(\vec{l};\hat{\tau}) = p(\vec{l};-\hat{\tau})$,

$$g(\vec{x}) = 1/2 \int_{\phi=0}^{2\pi} \int_{\theta=\pi/2-\psi}^{\pi/2+\psi} p(\vec{l};\hat{\tau}) \sin\theta d\theta d\phi \tag{A.4}$$

Recalling the definition of the function $\mathrm{rect}(x)$,

$$\mathrm{rect}(x) = \begin{cases} 1 & \text{if } |x| \leq 1/2 \\ 0 & \text{if } |x| > 1/2 \end{cases}$$

so that

$$\mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right) = \begin{cases} 1 & \text{if } |\theta| \leq \psi \pm \pi/2 \\ 0 & \text{if } |\theta| > \psi \pm \pi/2 \end{cases}$$

equation A.4 is now rewriten as

$$g(\vec{x}) = 1/2 \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi} \mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right) p(\vec{l};\hat{\tau}) \sin\theta d\theta d\phi \tag{A.5}$$

and using equation A.2,

$$g(\vec{x}) = 1/2 \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi} \mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right) \int_{t=-\infty}^{\infty} f(\vec{x} + t\hat{\tau}) dt \sin\theta d\theta d\phi \tag{A.6}$$

or

$$g(\vec{x}) = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi} \int_{t=-\infty}^{\infty} \frac{f(\vec{x} + t\hat{\tau})}{t^2} \mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right) t^2 dt \sin\theta d\theta d\phi \tag{A.7}$$

where the change in limits of integration is justified since in equation A.6 the integration is over all space twice.

Now by considering (t, θ, ϕ) as a spherical coordinate system, it can be converted to a rectangular coordinate system, $\vec{x}' = (x', y', z')$, by,

$$\begin{array}{ccc}
(t, \theta, \phi) & \longrightarrow & (x', y', z') \\
t\hat{r} & \longrightarrow & \vec{x}' \\
t^2 & \longrightarrow & |\vec{x}'|^2 \\
\theta & \longrightarrow & \theta' \\
t^2 dt \sin\theta d\theta d\phi & \longrightarrow & dx'dy'dz'
\end{array}$$

where $\theta' = \arccos\left(\frac{z'}{\sqrt{x'^2 + y'^2}}\right)$. Equation A.3 now becomes

$$g(\vec{x}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{f(\vec{x} + \vec{x}')}{|\vec{x}'|^2} \mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right) dx'dy'dz' \tag{A.8}$$

Recognizing the right hand side of equation A.8 as the three-dimensional convolution of $f(\vec{x})$ with the function

$$\frac{1}{r^2}\mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right)$$

where r is understood to be the spherical radial coordinate, that is $r^2 = |\vec{x}|^2$. Now rewriting equation A.8 using convolution notation,

$$g(\vec{x}) = f(\vec{x}) * \frac{1}{r^2}\mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right). \tag{A.9}$$

Finally, by comparing equations A.1 and A.9 it can be seen that

$$h = h(r, \theta, \phi) = \frac{1}{r^2}\mathrm{rect}\left(\frac{\theta - \pi/2}{2\psi}\right)$$

This is the PSF of the solid angle detector for a given cutoff angle ψ that is described on page 33.

Appendix B

Derivation of $L(\hat{\tau} \times \vec{x}_\tau)$

The definition of the $L(\hat{\tau} \times \vec{x}_\tau)$ function supplied by Orlov [68] is ,

$$L(\hat{\tau} \times \vec{x}_\tau) = \frac{1}{|\vec{x}|} \int\limits_{\mathbf{G}} d\mathbf{G}_{\tau'}\, \delta(\hat{\tau}' \cdot (\vec{x} \times \hat{\tau})). \tag{B.1}$$

The verbal definition of $L(\hat{\tau} \times \vec{x}_\tau)$ by Orlov is quoted in section 3.1.4. In the development of the three dimensional image reconstruction algorithm an explicit form for $L(\hat{\tau} \times \vec{x}_\tau)$ is needed for the case where the region **G** is the surface of a truncated hemisphere, as shown in figure B.1

Since $\delta(\hat{\tau}' \cdot (\vec{x} \times \hat{\tau})) = 0$ unless $\hat{\tau}'$, $\vec{x}$ and $\hat{\tau}$ are coplanar, the integral in equation B.1 sweeps out the length of the portion of the semi-great circle on the plane that is in the region **G**. This length is shown as the solid line part of the semi-great circle. The dashed line part cannot contribute to the value of $L(\hat{\tau} \times \vec{x}_\tau)$ since it lies outside the region **G**. Also note in figure B.1 that $\hat{x} = \vec{x}/|\vec{x}|$ and the parameter ψ is used to indicate where the unit hemisphere is truncated along the z-axis. There is an additional constraint added that $\hat{x} \cdot \hat{\tau} = 0$ because the vector of intrest is $\vec{x}_\tau = \vec{x} - (\vec{x} \cdot \hat{\tau})\hat{\tau}$, which is the projection of the vector $\vec{x}$ onto the plane perpendicular to $\hat{\tau}$.

The statement of the problem is now, given $\hat{x}$, $\hat{\tau}$, ψ, and the constraint $\hat{x} \cdot \hat{\tau} = 0$, find $L(\hat{\tau} \times \vec{x}_\tau)$ which is the total length of the solid line shown in figure B.1. This can be simplified to finding the length of the dashed line, $\mathcal{L}$, between points P_1 and P_2,

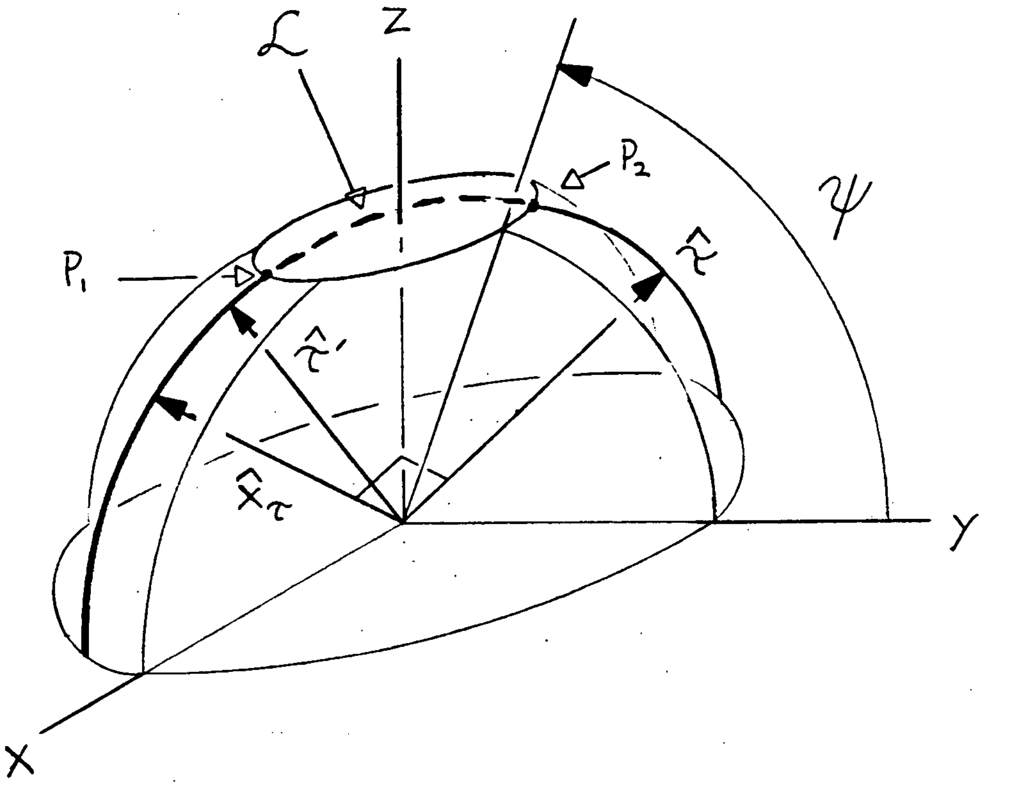

Figure B.1: Geometry of the $L(\hat{\tau} \times \vec{x}_\tau)$ function

since $L(\hat{\tau} \times \vec{x}_\tau) = \pi - \mathcal{L}$

The curved line segments lie on the plane where $\hat{\tau}' \cdot (\hat{x} \times \hat{\tau}) = 0$. This plane has a normal unit vector given by $(a, b, c) = \hat{x} \times \hat{\tau}$, so the plane is called the (a, b, c) plane. The points P_1 and P_2 are the intersection of the plane $z = \sin\psi$ and the semi-great circle lying on the (a, b, c) plane. Looking now at the (a, b, c) plane in figure B.2, the straight line segment from P_1 to P_2 has a length of d and since the circle has unit radius the angle between the line segments $\overline{OP_1}$ and $\overline{OP_2}$ is $\mathcal{L}$. This means that $\mathcal{L} = 2\arcsin(d/2)$ and if $P_1 = (x_1, y_1, z_1)$ and $P_2 = (x_2, y_2, z_2)$,

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

but $z_1 = z_2 = \sin\psi$, so

$$L(\hat{\tau} \times \vec{x}_\tau) = \pi - \mathcal{L} = \pi - 2\arcsin\left(\frac{1}{2}\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\right) \tag{B.2}$$

and the problem now reduces to finding the points (x_1, y_1) and (x_2, y_2).

The points P_1 and P_2 satisfy three conditions: they lie on the unit radius semi-great circle, they are on the $z = \sin\psi$ plane, and they are also on the (a, b, c) plane. Listing these constraints in the above order,

$$\begin{aligned} x^2 + y^2 + z^2 &= 1 \\ z &= \sin\psi \\ ax + by + cz &= 0 \end{aligned} \tag{B.3}$$

The last equation above comes from the definition of a plane passing through the origin that has a unit normal of (a, b, c) [1]. Now in order to solve equations B.3 for (x, y, z) it is necessary to first find the values a, b, and c.

Using standard spherical coordinates as shown in figure 3.6,

$$\begin{aligned} \hat{x} &= (\sin\theta_x \cos\phi_x, \sin\theta_x \sin\phi_x, \cos\theta_x) \\ \hat{\tau} &= (\sin\theta_\tau \cos\phi_\tau, \sin\theta_\tau \sin\phi_\tau, \cos\theta_\tau) \end{aligned}$$

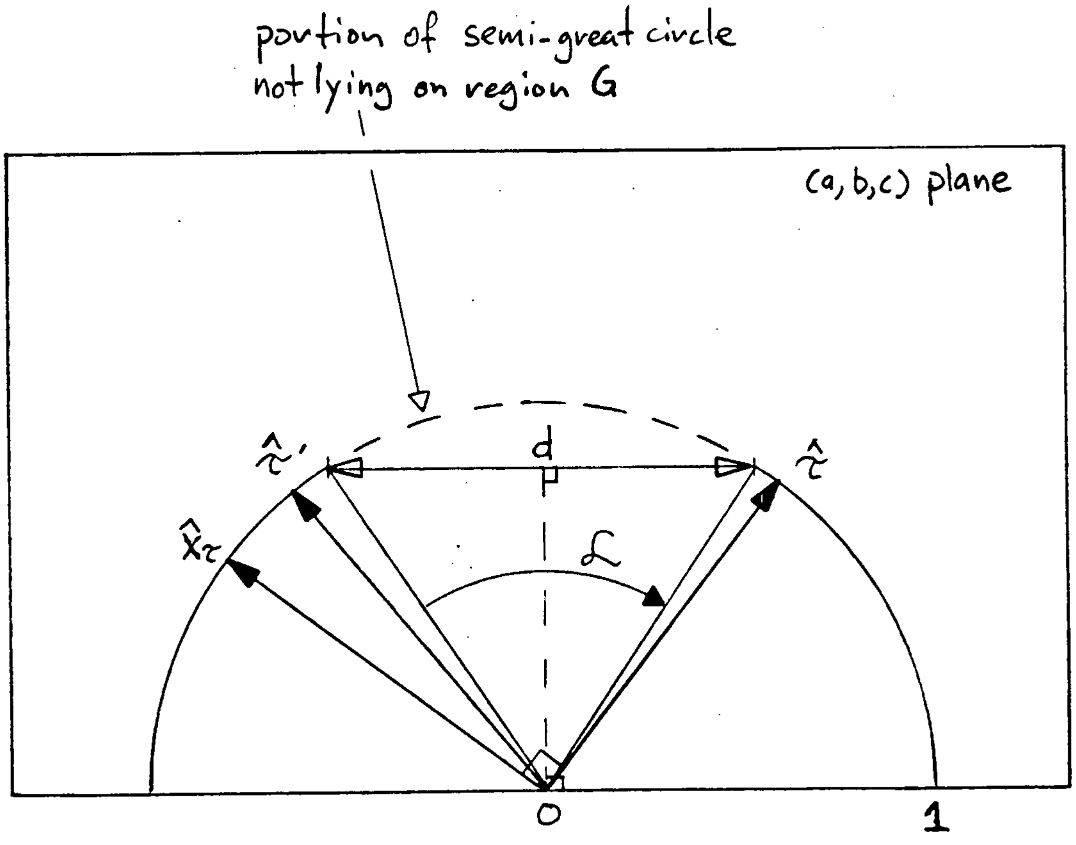

Figure B.2: Intersection of the (a, b, c) plane and the region **G**.

Another usefull constraint for a plane through the origin and coplanar with both $\hat{x}$ and $\hat{\tau}$ is [1],

$$\begin{vmatrix} x & y & z \\ \sin\theta_x\cos\phi_x & \sin\theta_x\sin\phi_x & \cos\theta_x \\ \sin\theta_\tau\cos\phi_\tau & \sin\theta_\tau\sin\phi_\tau & \cos\theta_\tau \end{vmatrix} = 0$$

This can be rewritten as,

$$ax + by + cz = 0$$

where

$$\begin{aligned} a &= \sin\theta_x\sin\phi_x\cos\theta_\tau - \sin\theta_\tau\sin\phi_\tau\cos\theta_x \\ b &= \sin\theta_\tau\cos\phi_\tau\cos\theta_x - \sin\theta_x\cos\phi_x\cos\theta_\tau \\ c &= \sin\theta_x\cos\phi_x\sin\theta_\tau\sin\phi_\tau - \sin\theta_x\sin\phi_x\sin\theta_\tau\cos\phi_\tau \\ &= \sin\theta_x\sin\theta_\tau\sin(\phi_x - \phi_\tau) \end{aligned} \tag{B.4}$$

From equations B.3 and B.4 it is now possible to solve for P_1 and P_2 in terms of the coordinates (θ_x, ϕ_x) of $\hat{x}$ and (θ_τ, ϕ_τ) of $\hat{\tau}$. In the solution it useful to note that since $(a, b, c) = \hat{x} \times \hat{\tau}$ and $\hat{x} \cdot \hat{\tau} = 0$, $a^2 + b^2 + c^2 = 1$. Using this, there are two solutions for (x, y, z) and,

$$\frac{1}{2}\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = \mathcal{L} = \sqrt{1 - \frac{\sin^2\psi}{\cos^2\theta_x + \cos^2\phi_x}}$$

This means that,

$$L(\hat{\tau} \times \vec{x}_\tau) = L(\theta_x; \theta_\tau) = \pi - 2\arcsin\left(\sqrt{1 - \frac{\sin^2\psi}{\cos^2\theta_x + \cos^2\phi_x}}\right)$$

and since, if $0 \leq x \leq 1$ [1],

$$\pi - 2\arcsin\left(\sqrt{1 - x^2}\right) = 2\arcsin(x)$$

the explicit form of $L(\hat{\tau} \times \vec{x}_\tau)$is

$$L(\theta_x;\theta_\tau) = 2\arcsin\left(\frac{\sin\psi}{\sqrt{\cos^2\theta_x + \cos^2\phi_x}}\right).$$

In the above equation however, the arcsin function becomes undefined if $\sin^2\psi \geq \cos^2\theta_x + \cos^2\theta_\tau$. In figure B.1 this is equivalent to the semi-great circle lying entirely within the region **G** so that the points P_1 and P_2 do not exist. But in this case $\mathcal{L} = 0$ and $L(\theta_x;\theta_\tau) = \pi$, so

$$L(\theta_x;\theta_\tau) = \begin{cases} \pi, & \text{if } \sin^2\psi \geq \cos^2\theta_x + \cos^2\theta_\tau \\ 2\arcsin\left(\frac{\sin\psi}{\sqrt{\cos^2\theta_x + \cos^2\theta_\tau}}\right), & \text{if } \sin^2\psi < \cos^2\theta_x + \cos^2\theta_\tau \end{cases} \tag{B.5}$$

Note that for $\sin^2\psi = \cos^2\theta_x + \cos^2\theta_\tau$, the points P_1 and P_2 are coincident and $L(\theta_x;\theta_\tau) = \pi$, thus $L(\theta_x;\theta_\tau)$ is continuous but has a discontinuity in slope.

It is necessary in chapter 3 to have the function $L(\theta_x;\theta_\tau)$ expressed in the coordinates of the projection plane as shown in figure 3.6. Using spherical trigonometry similar to that of figure 5.1, the angle θ_x is related to θ_l and θ_τ by the relation $\cos\theta_x = -\sin\theta_l\sin\theta_\tau$. This means that

$$\cos^2\theta_x + \cos^2\theta_\tau = 1 - \cos^2\theta_l\sin^2\theta_\tau$$

and

$$L(\theta_l;\theta_\tau) = \begin{cases} \pi, & \text{if } |\cos\psi| \leq |\cos\theta_l|\cdot|\cos\theta_\tau| \\ 2\arcsin\left(\frac{\sin\psi}{\sqrt{1-\cos^2\theta_l\sin^2\theta_\tau}}\right), & \text{if } |\cos\psi| > |\cos\theta_l|\cdot|\cos\theta_\tau| \end{cases}.$$

This is the result quoted in section 3.1.6.

Appendix C

Derivatives of $L(\theta_l;\theta_\tau)$

This appendix shows the calculation of the derivatives of $L(\theta_l;\theta_\tau)$ with respect to θ_l. From appendix B, $L(\theta_l;\theta_\tau)$ is given by,

$$L(\theta_l;\theta_\tau) = \begin{cases} \pi, & \text{if } |\cos\psi| \le |\cos\theta_l|\cdot|\cos\theta_\tau| \\ 2\arcsin\left(\dfrac{\sin\psi}{\sqrt{1-\cos^2\theta_l\sin^2\theta_\tau}}\right), & \text{if } |\cos\psi| > |\cos\theta_l|\cdot|\cos\theta_\tau| \end{cases}$$

In the case where $L(\theta_l;\theta_\tau) = \pi$, all derivatives are zero and it is only necessary to to find the derivatives where $L(\theta_l;\theta_\tau) \neq \pi$. Note that this means there is a discontinuity of the derivatives at $|\cos\psi| = |\cos\theta_l|\cdot|\cos\theta_\tau|$. For consistency with chapter 3, a new function $\overline{L}(\theta_l;\theta_\tau)$ is defined as,

$$\overline{L}(\theta_l;\theta_\tau) = 2\arcsin\left(\frac{\sin\psi}{\sqrt{1-\cos^2\theta_l\sin^2\theta_\tau}}\right).$$

C.1 First Derivative

To start with;

$$\frac{\partial}{\partial\theta_l}\overline{L}(\theta_l;\theta_\tau) = \frac{\sin\psi}{\sqrt{1-\dfrac{\sin^2\psi}{\cos^2\theta_l\sin^2\theta_\tau}}}\cdot\frac{\partial}{\partial\theta_l}(1-\cos^2\theta_l\sin^2\theta_\tau)^{-1/2}$$

and

$$\frac{\partial}{\partial\theta_l}(1-\cos^2\theta_l\sin^2\theta_\tau)^{-1/2} = -\frac{\sin\theta_l\cos\theta_l\sin^2\theta_\tau}{2(1-\cos^2\theta_l\sin^2\theta_\tau)^{3/2}}$$

so

$$\frac{\partial}{\partial\theta_l}\overline{L}(\theta_l;\theta_\tau) = \frac{\sin\psi}{\sqrt{1-\dfrac{\sin^2\psi}{\cos^2\theta_l\sin^2\theta_\tau}}}\cdot -\frac{\sin\theta_l\cos\theta_l\sin^2\theta_\tau}{2(1-\cos^2\theta_l\sin^2\theta_\tau)^{3/2}}$$

Finally,

$$\frac{\partial}{\partial\theta_l}\overline{L}(\theta_l;\theta_\tau) = -\frac{\sin 2\theta_l\sin^2\theta_\tau\sin\psi}{(1-\cos^2\theta_l\sin^2\theta_\tau)\sqrt{\cos^2\psi-\cos^2\theta_l\sin^2\theta_\tau}}.$$

C.2 The Second Derivative

Using the above result and continuing on,

$$\frac{\partial^2}{\partial{\theta_l}^2}\overline{L}(\theta_l;\theta_\tau) = \frac{\partial}{\partial\theta_l}\left[\frac{\partial}{\partial\theta_l}\overline{L}(\theta_l;\theta_\tau)\right] \equiv \frac{\partial}{\partial\theta_l}\left[A(\theta_l;\theta_\tau)B(\theta_l;\theta_\tau)\right]$$

or

$$\frac{\partial^2}{\partial{\theta_l}^2}\overline{L}(\theta_l;\theta_\tau) = \frac{\partial}{\partial\theta_l}A(\theta_l;\theta_\tau)B(\theta_l;\theta_\tau) + A(\theta_l;\theta_\tau)\frac{\partial}{\partial\theta_l}B(\theta_l;\theta_\tau)$$

where by definition,

$$A(\theta_l;\theta_\tau) = -\frac{\sin 2\theta_l\sin^2\theta_\tau\sin\psi}{(1-\cos^2\theta_l\sin^2\theta_\tau)}$$

and

$$B(\theta_l;\theta_\tau) = (\cos^2\psi-\cos^2\theta_l\sin^2\theta_\tau)^{-1/2}.$$

The derivatives of $A(\theta_l;\theta_\tau)$ and $B(\theta_l;\theta_\tau)$ are given by,

$$\frac{\partial}{\partial\theta_l}A(\theta_l;\theta_\tau) = -\frac{2\cos 2\theta_l(1-\cos^2\theta_l\sin^2\theta_\tau)\sin^2\theta_\tau\sin\psi - \sin^2 2\theta_l\sin^4\theta_\tau\sin\psi}{(1-\cos^2\theta_l\sin^2\theta_\tau)^2}$$

and

$$\frac{\partial}{\partial\theta_l}B(\theta_l;\theta_\tau) = -\frac{\sin 2\theta_l\sin^2\theta_\tau}{2(\cos^2\psi-\cos^2\theta_l\sin^2\theta_\tau)^{3/2}}.$$

The expression for the second derivative of $\overline{L}(\theta_l;\theta_\tau)$is too long to write out properly, but it is explicitly expressed by the above five equations. To usehe results of this section in equation 3.27, it is only necessary to note that,

$$\frac{\partial}{\partial\theta_l}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right) = -\frac{1}{L(\theta_l;\theta_\tau)^2}\frac{\partial}{\partial\theta_l}L(\theta_l;\theta_\tau)$$

and

$$\frac{\partial^2}{\partial{\theta_l}^2}\left(\frac{1}{L(\theta_l;\theta_\tau)}\right) = \frac{2}{L(\theta_l;\theta_\tau)^3}\left(\frac{\partial}{\partial\theta_l}L(\theta_l;\theta_\tau)\right)^2 - \frac{1}{L(\theta_l;\theta_\tau)^2}\left(\frac{\partial^2}{\partial{\theta_l}^2}L(\theta_l;\theta_\tau)\right).$$

Appendix D

Filter Program Listing

```
      program generate_filter
C-----------------------------------------------------------------------
C GENERATE_FILTER
C Generates a set of 2D reconstruction filters.
C The filters are n x n and there are t of them, so each value of t
C represents a different 2D filter.
C 1. The filter parameters (n,t,psi,pixel_size) are read by get_parameters()
C    from the file par_file.
C 2. An array for storing the filters is created and passed with the parameters
C    to calculate_filter(), which does all the actual calculations.
C 3. A unique filename for the filter set is created by make_filename()
C    and passed to output_filter(), which copies the unformatted 4-byte
C    filter values to the file.
C 4. An ASCII formated output of the filter (for humans) along with the
C    filter parameters and filenames is produced by hardcopy().
C 5. Finally the array storage space is released.
C
C History: 2nd. version written 19-feb-1988 Paul Kinahan
C
C-----------------------------------------------------------------------
      implicit none
      integer n,t,filter_name_len,max_name_len,par_name_len,wordsize,
     +        lib$get_vm,lib$free_vm,status,filter_addr,filter_size
      parameter(max_name_len=50,wordsize=4)
      real psi,pixel_size
      character*(max_name_len) filter_file,par_file

      call get_parameters(n,t,psi,pixel_size,par_file,par_name_len)
      filter_size = (n**2)*t*wordsize
      status=lib$get_vm(filter_size,filter_addr)
      if(.not.status) call lib$signal(%val(status))
      call calculate_filter(n,t,psi,pixel_size,%val(filter_addr))
      call make_filename(n,t,psi,pixel_size,filter_file,filter_name_len)
      call output_filter(n,t,psi,pixel_size,%val(filter_addr),
     +                   filter_file,filter_name_len)
      call hardcopy(n,t,psi,pixel_size,%val(filter_addr),
     +              par_file,par_name_len,filter_file,filter_name_len)
      status=lib$free_vm(filter_size,filter_addr)
      if(.not.status) call lib$signal(%val(status))
      end
      subroutine calculate_filter(n,t,psi,pixel_size,filter)
C-----------------------------------------------------------------------
C CALCULATE_FILTER
C Does the actual calculation of the set of 2D reconstruction filters
C as explained in my thesis. Only the first quadrant is calculated
C since the filters are symmetrical about the x- and y-axes (and about
C theta_tau = 90deg). The filter values are calculated for each pixel
C by calculate_pixel(), which also uses the weighting arrays for numerical
C integration. If the pixel is close to the origin or the line of
C singularities,the fine-gridding array is passed, otherwise the course-
C gridding array is passed.
C
C History: 2nd version written 4-mar-1988 PK
C
C-----------------------------------------------------------------------
      implicit none
      logical use_fine_gridding
      integer n,t,ti,i,j,fine_divs,course_divs,numpix,totalpix,
     +        timer_addr,lib$init_timer,lib$show_timer,lib$free_timer
      parameter(fine_divs=50,course_divs=8)
      real theta_tau,psi,pixel_size,theta_bar,xi,yi,pi,
     +     fine_weights(0:fine_divs),course_weights(0:course_divs),
     +     filter(0:n-1,0:n-1,0:t-1),calculate_pixel,quad1,pos_axes
      parameter(pi=3.1415927)

! Set up 1D arrays of weights for numerical integration (Simpsons)
! weights for fine gridding of pixel
      fine_weights(0) = 1.0                ! endpoints
      fine_weights(fine_divs) = 1.0
      do i=1,fine_divs-3,2
        fine_weights(i) = 4.0              ! even points
        fine_weights(i+1) = 2.0            ! odd points
      enddo
      fine_weights(fine_divs-1) = 4.0      ! penultimate point

! weights for course gridding of pixel
      course_weights(0) = 1.0              ! endpoints
      course_weights(course_divs) = 1.0
      do i=1,course_divs-3,2
        course_weights(i) = 4.0            ! even points
        course_weights(i+1) = 2.0          ! odd points
      enddo
      course_weights(course_divs-1) = 4.0  ! penultimate point

      totalpix = (n**2)*t
      type '(/,a,i5,a,/)',' There are a total of',
     +                    totalpix,' pixels to calculate'
      call lib$init_timer(timer_addr) ! to start collecting program statistics

      do ti=0,t-1 ! Calculate filter values for each theta_tau.
        theta_tau = pi/2.0 -  psi*float(ti)/float(t-1)
        theta_bar=acos(cos(psi)/sin(theta_tau)) ! critical angle

        do j=0,n-1 ! Calculate filter values for 1st. Quadrant
        do i=0,n-1
          numpix = i + 1 + j*n + ti*n**2
          type '(a,i5)','+ Doing pixel number: ',numpix
          xi=float(i)*pixel_size
          yi=float(j)*pixel_size
          use_fine_gridding = ((i.le.2).and.(j.le.2)) ! if near origin or
     +     .or.((yi*cos(theta_bar)-xi*sin(theta_bar)).le.pixel_size) ! critical
          if(use_fine_gridding) then                               ! angle
            filter(i,j,ti)=calculate_pixel(xi,yi,theta_tau,psi,
     +                     theta_bar,pixel_size,fine_divs,fine_weights)
          else ! use course gridding to speed up computation time
            filter(i,j,ti)=calculate_pixel(xi,yi,theta_tau,psi,
     +                 theta_bar,pixel_size,course_divs,course_weights)
          endif
        enddo
        enddo

! set centre of filter to negative of the rest of filter, calulating axes
! seperately
        quad1 = 0.0
        pos_axes = 0.0
        do j=1,n-1
        do i=1,n-1
          quad1 = quad1 + filter(i,j,ti)
        enddo
        enddo
        do i=1,n-1
          pos_axes = pos_axes + filter(0,i,ti) + filter(i,0,ti)
        enddo
        filter(0,0,ti) = -(4*quad1 + 2*pos_axes)

        call window(filter,n,ti) ! to low-pass filter the reconstruction filter
      enddo ! end of loop for each value of theta_tau

      type '(a)','+ Finished                         '
      type '(/,a)','  Filter generation program statistics'
      type '(a,/)','  -------------------------------------'
      call lib$show_timer(timer_addr)
      call lib$free_timer(timer_addr)
      type '(a)',' '
      end
      real function calculate_pixel(xi,yi,theta_tau,psi,theta_bar,
     +                              pixel_size,divs,weights)
C-----------------------------------------------------------------------
C CALCULATE_PIXEL
C Calculates the value of a single pixel of width pixel_size centred
C at (xi,yi) as explained in my thesis.
C The calculation is broken into 3 parts depending on where the pixel
C is in relation to the critical angle theta_bar. The 4 corners of the
C pixel are given by x1,x2,y1, and y2.
C
C History: 2nd version written 19-feb-1988 PK
C-----------------------------------------------------------------------
      implicit none
      logical intersection
      integer divs
      real xi,yi,theta_tau,psi,theta_bar,pixel_size,
     +     x1,x2,y1,y2,theta_max,theta_min,weights(0:divs),
     +     integral_1,integral_2,integral_3,term_1,term_2,term_3

      calculate_pixel = 0.0
! quickly return 0.0 if we're at the origin
      if ((xi.eq.0.0).and.(yi.eq.0.0)) goto 999

! set up initial values
      x1 = amax1(0.0,xi - pixel_size/2.0) !in case we are on y-axis
      x2 = xi + pixel_size/2.0
      y1 = amax1(0.0,yi - pixel_size/2.0) !in case we are on x-axis
      y2 = yi + pixel_size/2.0
      theta_min = atan2(y1,x2) ! min value of theta_1 in pixel
      theta_max = atan2(y2,x1) ! max value of theta_1 in pixel

! subdivide calculation depending on where critical ray is
! relative to pixel

      if (theta_bar.gt.theta_max) then ! intersection =.false.
        term_1 = integral_1(x1,x2,y1,y2)
        term_2 = 0.0
        term_3 = 0.0
      elseif((theta_min.le.theta_bar).and.(theta_bar.le.theta_max))then
        intersection =.true.
        term_1 = integral_1(x1,x2,y1,y2)
        term_2 = integral_2(x1,x2,y1,y2,theta_bar)
        term_3 = integral_3(x1,x2,y1,y2,theta_tau,psi,theta_bar,divs,
     +                      weights,intersection)
      elseif (theta_min.gt.theta_bar) then
        intersection =.false.
        term_1 = integral_1(x1,x2,y1,y2)
        term_2 = term_1
        term_3 = integral_3(x1,x2,y1,y2,theta_tau,psi,theta_bar,divs,
     +                      weights,intersection)
      endif
      calculate_pixel = term_1+term_2+term_3

  999 continue ! goto (gasp) point for pixel at origin
      end
      real function integral_1(x1,x2,y1,y2)
C-----------------------------------------------------------------------
C INTEGRAL_1
C Calculates the I1 term (see thesis) for a pixel with corners at x1,x2,y1,
C and y2.
C Tests real values for zero since it can be assigned by the calling routine.
C
C History: written 19-feb-1988 PK
C------------------------------------------------------------------------
      implicit none
      real x1,x2,y1,y2,pi
      parameter(pi = 3.1415927)

      if ((x1.lt.0.0).or.(y1.lt.0.0)) then
        print *,'%USR-F-NaughtyVal  Negative pixel coordinates passed ',
     +                             'to routine integral_1()'
        call exit
      elseif (x1.eq.0.0) then ! pixel is on y-axis
        integral_1 = (1.0/(2.0*pi**3))*( sqrt(x2**2 + y2**2)/(x2*y2)
     +                                 - sqrt(x2**2 + y1**2)/(x2*y1) )
      elseif (y1.eq.0.0) then ! pixel is on x-axis
        integral_1 = (1.0/(2.0*pi**3))*( sqrt(x2**2 + y2**2)/(x2*y2)
     +                                 - sqrt(x1**2 + y2**2)/(x1*y2) )
      else ! pixel is somewhere in 1st quadrant
        integral_1 = (1.0/(4.0*pi**3))*( sqrt(x2**2 + y2**2)/(x2*y2)
     +                                 - sqrt(x1**2 + y2**2)/(x1*y2)
     +                                 - sqrt(x2**2 + y1**2)/(x2*y1)
     +                                 + sqrt(x1**2 + y1**2)/(x1*y1) )
      endif
      end
      real function integral_2(x1,x2,y1,y2,theta_bar)
C-----------------------------------------------------------------------
C INTEGRAL_2
C Calculates the I2 term (see thesis) for a pixel with corners at x1,x2,y1,
C and y2.
C Tests real values for zero since it can be assigned by the calling routine.
C
C History: written 19-feb-1988 PK
C------------------------------------------------------------------------
      implicit none
      real x1,x2,y1,y2,theta_bar,pi,integral_1,y_start,y_end,x_end,
     +     upper_y_intersection,lower_y_intersection
      parameter(pi = 3.1415927)

! decide upon region of evaluation

      upper_y_intersection = x2*tan(theta_bar)
      lower_y_intersection = x1*tan(theta_bar)
      x_end = amin1(x2,y2/tan(theta_bar))
      y_start = amax1(y1,lower_y_intersection)
      y_end = amin1(y2,upper_y_intersection)

! evaluate part of pixel defined by intersection with ray

      if ((x1.lt.0.0).or.(y1.lt.0.0)) then
        print *,'%USR-F-NaughtyVal  Negative pixel coordinates passed ',
     +                               'to routine integral_2()'
        call exit
      elseif (x1.eq.0.0) then ! pixel is on y-axis
        integral_2 = -(1.0/(2.0*pi**3))*cos(theta_bar)
     +                                   *(1.0/y_end - 1.0/y_start)
      elseif (y1.eq.0.0) then ! pixel is on x-axis
        integral_2 = (1.0/(2.0*pi**3))
     +    *( (sqrt(y_end**2 + x1**2)/(x1*y_end)-cos(theta_bar)/y_end)
     + -(sqrt(y_start**2 + x1**2)/(x1*y_start)-cos(theta_bar)/y_start))

      else ! it must be that we are somewhere else in the 1st quadrant
        integral_2 = (1.0/(4.0*pi**3))
     +    *( (sqrt(y_end**2 + x1**2)/(x1*y_end)-cos(theta_bar)/y_end)
     + -(sqrt(y_start**2 + x1**2)/(x1*y_start)-cos(theta_bar)/y_start))
      endif

! if there is a rectangular region left, pass it to integral_1()
! to evaluate and multiply by 2 if the pixel is on an axis

      if (y_end.lt.y2)
     +     integral_2 = integral_2 - integral_1(x1,x2,y_end,y2)
      if ((x1.eq.0.0).or.(y1.eq.0.0))
     +     integral_2 = 2*integral_2
      end
      real function integral_3(x1,x2,y1,y2,theta_tau,psi,theta_bar,
     +                         divs,weights,intersection)
C-----------------------------------------------------------------------
C INTEGRAL_3
C Calculates the I3 term (see thesis) for a pixel with corners at x1,x2,y1,
C and y2.
C Uses a 2D Simpson's rule for numerical integration. The 2D quadrature
C weights are calculated by multiplying the appropriate 1D weights passed
C in the 1D array weights(0:divs). The logical variable 'intersection' is
C true if the critical ray (theta_bar) passes through the pixel.
C
C History: written 19-feb-1988 PK
C------------------------------------------------------------------------
      implicit none
      logical intersection
      integer i,j,divs
      real lx,ly,x1,x2,y1,y2,theta_tau,psi,theta_bar,alpha_scale,
     +     h,k,r1,r2,r,alpha,theta_l,weights(0:divs),grid_weight,
     +     l_fcn,l_fcn_d1,l_fcn_d2,pi
      parameter(alpha_scale=1.0,pi=3.1415927)

      h = (x2 - x1)/float(divs) ! delta x grid width
      k = (y2 - y1)/float(divs) ! delta y grid width

      if (intersection) then ! calculate alpha
        r1 = sqrt( amax1(x1,y1/tan(theta_bar))**2
     +           + amax1(y1,x1*tan(theta_bar))**2 )
        r2 = sqrt( amin1(x2,y2/tan(theta_bar))**2
     +           + amin1(y2,x2*tan(theta_bar))**2 )
        alpha = alpha_scale*2*asin(amax1(h,k)/(r1 + r2))! to be sure to get
      else                                              ! the right distance
        alpha = 0.0
      endif

      integral_3 = 0.0

      do j=0,divs ! evaluate using 2d simpsons rule
      do i=0,divs
        lx = x1 + (float(i))*h ! grid points
        ly = y1 + (float(j))*k
        r = sqrt(lx**2 + ly**2) ! polar coordinate of grid point
        theta_l = atan2(ly,lx)
        grid_weight = (h*k/9.0)*weights(i)*weights(j)
        integral_3 = integral_3 + grid_weight*-1.0/(4.0*(pi**2)*(r**3))
     +                 *(l_fcn(theta_l,theta_tau,psi,theta_bar)
     +               + l_fcn_d2(theta_l,theta_tau,psi,theta_bar+alpha))
      enddo
      enddo

      if (intersection) then ! add any contribution from the singularity
        integral_3 = integral_3 + (1.0/(4.0*pi**2))*(1.0/r2 - 1.0/r1)
     +               *l_fcn_d1(theta_bar+alpha,theta_tau,psi,theta_bar)
      endif

      if ((x1.eq.0.0).or.(y1.eq.0.0)) integral_3=2*integral_3 ! since on axis
      end
      real function l_fcn(theta_l,theta_tau,psi,theta_bar)
C---------------------------------------------------------------------------
C L_FCN
C Calculates the Orlov L() function term as given in thesis.
C
C History: written 19-feb-1988 PK
C---------------------------------------------------------------------------
      implicit none
      real theta_l,theta_tau,psi,theta_bar,pi
      parameter(pi=3.1415927)

      if (theta_l.le.theta_bar) then
        l_fcn = pi
      else
        l_fcn = 2*asin(sin(psi)
     +                 /sqrt(1.0-(cos(theta_l)*sin(theta_tau))**2))
      endif

      end
      real function l_fcn_d1(theta_l,theta_tau,psi,theta_bar)
C-----------------------------------------------------------------------
C L_FCN_D1
C Calculates the value of the first derivative of 1/L(), where L() is
C the Orlov function term as given in thesis. Statement functions are
C used to abbreviate final term
C
C History: written 19-feb-1988 PK
C-----------------------------------------------------------------------
      implicit none
      real theta_l,theta_tau,psi,theta_bar,
     +     d1,d2,d3,f1,f2,f3,a,ad1,b,bd1,l,ld1,ld2 ! dummy arguements

!  define statement functions
!  note that for dummy arguments:
!   d1 = theta_l
!   d2 = theta_tau
!   d3 = psi (when used)
!
!  also note that ad1,bd1,ld1 are the first derivatives of a,b,&l w.r.t d1
!  and   "    "      ld2     is  "  second    "            l      "   "

      f1(d1,d2) = 1.0 - (cos(d1)**2)*(sin(d2)**2)

      f2(d1,d2,d3) = cos(d3)**2 - (cos(d1)**2)*(sin(d2)**2)

      f3(d1,d2) = sin(2*d1)*(sin(d2)**2)

      l(d1,d2,d3) = 2*asin(sin(d3)/sqrt(f1(d1,d2))) ! the L() function again

      a(d1,d2,d3) = - sin(d3)*f3(d1,d2)/f1(d1,d2)

      b(d1,d2,d3) = 1.0/sqrt(f2(d1,d2,d3))

      ld1(d1,d2,d3) = a(d1,d2,d3)*b(d1,d2,d3) ! 1st derivative of L()

! - End of statement function definitions

      if (theta_l.le.theta_bar) then
        l_fcn_d1 = 0.0
      else
        l_fcn_d1 = -ld1(theta_l,theta_tau,psi)
     +              /(l(theta_l,theta_tau,psi)**2)
      endif
      end
      real function l_fcn_d2(theta_l,theta_tau,psi,theta_bar)
C-----------------------------------------------------------------------
C L_FCN_D2
C Calculates the value of the second derivative of 1/L(), where L() is
C the Orlov function term as given in thesis. Statement functions are
C used to abbreviate final term
C
C History: written 19-feb-1988 PK
C-----------------------------------------------------------------------
      implicit none
      real theta_l,theta_tau,psi,theta_bar,
     +     d1,d2,d3,f1,f2,f3,a,ad1,b,bd1,l,ld1,ld2 ! dummy arguements

!  define statement functions
!  note that for dummy arguments:
!   d1 = theta_l
!   d2 = theta_tau
!   d3 = psi (when used)
!
!  also note that ad1,bd1,ld1 are the first derivatives of a,b,&l w.r.t d1
!  and   "    "      ld2     is   "  second    "           l      "  "

      f1(d1,d2) = 1.0 - (cos(d1)**2)*(sin(d2)**2)

      f2(d1,d2,d3) = cos(d3)**2 - (cos(d1)**2)*(sin(d2)**2)

      f3(d1,d2) = sin(2*d1)*(sin(d2)**2)

      l(d1,d2,d3) = 2*asin(sin(d3)/sqrt(f1(d1,d2))) ! the L() function again

      a(d1,d2,d3) = - sin(d3)*f3(d1,d2)/f1(d1,d2)

      ad1(d1,d2,d3) = - 2*sin(d3)*(sin(d2)**2)*cos(2*d1)/f1(d1,d2)
     +              + sin(d3)*(f3(d1,d2)/f1(d1,d2))**2

      b(d1,d2,d3) = 1.0/sqrt(f2(d1,d2,d3))

      bd1(d1,d2,d3) = - f3(d1,d2)/(2*(f2(d1,d2,d3)**(1.5)))

      ld1(d1,d2,d3) = a(d1,d2,d3)*b(d1,d2,d3) ! 1st derivative of L()

      ld2(d1,d2,d3) = ad1(d1,d2,d3)*b(d1,d2,d3) ! 2nd derivative of L()
     +              + a(d1,d2,d3)*bd1(d1,d2,d3)

! - End of statement function definitions

      if (theta_l.le.theta_bar) then
        l_fcn_d2 = 0.0
      else
        l_fcn_d2 = (1.0/(l(theta_l,theta_tau,psi)**2))
     +            *( 2.0*ld1(theta_l,theta_tau,psi)**2
     +                /l(theta_l,theta_tau,psi)
     +             - ld2(theta_l,theta_tau,psi) )
      endif
      end
      subroutine window(array,n,t)
C-----------------------------------------------------------------------
C WINDOW
C Low-pass filters the 2D array indexed by 'array(,,t)' by using 2D
C convolution. The low-pass filter is the  object space version of a
C Hamming window. The original array only contains the first quadrant
C values so values in the other 3 quadrants are obtained by reflection
C of the indicies i3 and j3 about the axes.
C
C History: written 9-oct-1987
C-----------------------------------------------------------------------
      implicit none
      integer n,i,j,i2,j2,i3,j3,nmax,tmax,t
      parameter(nmax=51,tmax=10)
      real array(0:n-1,0:n-1),copy(0:nmax-1,0:nmax-1),r,w,sinc,pi
      parameter(pi=3.141592654)

      if(n.gt.nmax) then
        type *,' Dummy array in routine WINDOW too small'
        call exit
      endif

      do j=0,n-1 ! for first quadrant
      do i=0,n-1
        copy(i,j)=0
        do j2=-(n-1),n-1 ! for entire plane
        do i2=-(n-1),n-1
          j3 = j2
          i3 = i2
          if (j2.lt.0) j3 = -j2
          if (i2.lt.0) i3 = -i2
          r = sqrt(float(i-i2)**2 + float(j-j2)**2)
          w = 0.5*sinc(r) + 0.25*(sinc(r-1.0) + sinc(r+1.0)) ! Hamming term
          copy(i,j) = copy(i,j) + array(i3,j3)*w
        enddo
        enddo
      enddo
      enddo

      do j=0,n-1 ! put filtered values in original array
      do i=0,n-1
        array(i,j) = copy(i,j)
      enddo
      enddo

      end
C-----------------------------------------------------------------------
      real function sinc(x)
      implicit none
      real x,pi
      parameter (pi=3.141592654)

      if (x.eq.0.0) then
        sinc = 1.0
      else
        sinc = sin(pi*x)/(pi*x)
      endif
      end
      subroutine output_filter(n,t,pixel_size,psi,filter,filename,
     +                         filename_len)
C-------------------------------------------------------------------
C OUTPUT_FILTER
C Writes out the unformatted 3D filter array to the file 'filename'.
C The filter parameters are written on the first line of the file, with
C the filter values starting in the second line.
C This file is the main goal of the filter generation.
C
C History: written 21-feb-1988 PK
C-------------------------------------------------------------------
      implicit none
      logical exists
      integer n,t,i,j,k,index,filename_len,lun,lib$get_lun,status,
     +        max_name_len,length
      parameter(max_name_len=50)
      real filter(0:n-1,0:n-1,0:t-1),pi,psi,pixel_size
      parameter(pi=3.1415927)
      character*(max_name_len) filename,newname,fullname
      character answer*20,char1*1,blank*1
      parameter(blank=' ')

! check with user if file name isn't unique

   10 inquire(file=filename(1:filename_len),exist=exists)
      if (exists) then
   20    type '(3a,/,a,$)',' The file ',
     +              filename(1:filename_len),' already exists',
     +           '  Do you want to use a different file name? [N]: '
         read '(a20)',answer
         char1 = answer(1:1)
        if ((char1.ne.'Y').and.(char1.ne.' ').and.(char1.ne.'N')) then
            type '(a)','  Type YES, NO, or return for default [NO]'
            go to 20
         elseif ((char1.ne.'N').and.(char1.ne.' ')) then
            type '(a,$)','  Enter new file name: '
            read '(q,a)', length,newname
            if (length.gt.max_name_len) then
               type'(a,i2,a)',' Maximum name length is ',
     +                           max_name_len,' characters'
               go to 20
            else
               filename=newname(1:filename_len)
               go to 10
            endif
         else
            type '(3a)','  A new version of ',
     +                    filename(1:filename_len),
     +                   ' will be created'
         endif
      endif

      status=LIB$GET_LUN(lun)
      if(.not.status) call LIB$SIGNAL(%val(status))
      open(unit=lun,file=filename(1:filename_len),status='NEW',
     +     form='UNFORMATTED')

      write(lun) 2*n-1,2*t-1,psi,pixel_size ! convert to full values
      write(lun) (((filter(i,j,k),i=0,n-1),j=0,n-1),k=0,t-1)
      close(lun)
```

```
! Get the full file name

      inquire(file=filename(1:filename_len),name=fullname)
      filename_len = index(fullname,blank)-1
      filename(1:filename_len) = fullname(1:filename_len)

      type '(2a)',' Filter data is in file ',
     +              filename(1:filename_len)
      end
      subroutine get_parameters(n,t,psi,pixel_size,filename,name_len)
C-----------------------------------------------------------------------
C GET_PARAMETERS
C Reads filter generation parameters in from the file 'filename'.
C The parameters are checked and then converted for use by the routine
C calculate_filter().
C
C History: written 21-feb-1988 PK
C         - for now the parameter file is set to 'FILTER.PAR'
C-------------------------------------------------------------------------
      implicit none
      logical exists
      integer n,n_max,t,t_max,name_len,max_name_len,lun,status,
     +        two,one,zero
      parameter(max_name_len=50,n_max=101,t_max=11,zero=0,one=1,two=2)
      real psi,pixel_size
      parameter(pi=3.1415927)
      character*(max_name_len) filename,answer,newname,blank*1
      parameter(blank=' ')

! include condition codes and declare external routines

      include '($FORDEF)'
      integer LIB$GET_LUN

! get the file containg filter parameters

      filename = 'FILTER.PAR' ! default file
      name_len = index(filename,blank)-1
   10 continue
      inquire(file=filename(1:name_len),exist=exists,name=newname)
      if(.not.exists) then
          type '(3a)',' Parameter file ',filename(1:name_len),
     +                ' does not exist or is in use'
   20     continue
          type '(2a,$)',' Input parameter filename ',
     +                  '(or return to quit) : '
          read '(q,a)', name_len,answer
          if(answer(1:1).eq.' ') go to 1010
          if(name_len.gt.max_name_len) then
              type '(a,i2,a)',' File name must be less than '
     +                        ,max_name_len,' characters'
              goto 20
          endif
          filename(1:name_len) = answer(1:name_len)
          goto 10
      else
          name_len = index(newname,blank)-1
          filename(1:name_len)=newname(1:name_len)
      endif
      type '(/,2a)','  Filter parameters from file ',
     +              filename(1:name_len)

! Open file and read parameter values, lines with a non-blank 1st character
! are comment lines and are skipped over

      status=LIB$GET_LUN(lun)
      if(.not.status) call LIB$SIGNAL(%val(status))
      open(unit=lun,file=filename(1:name_len),carriagecontrol='LIST',
     +     status='OLD',err=1000)
```

```
      call skip_comments(lun,filename,name_len)
      read(lun,*,err=1000) n
      if(n.gt.n_max) then
          type '(2a,i3)',' Parameter n is greater than max allowed',
     +                   ' value of ',n_max
          goto 1000
      elseif(mod(n,two).ne.one) then
          type '(a)',' Parameter n must be odd'
          goto 1000
      endif

      call skip_comments(lun,filename,name_len)
      read(lun,*,err=1000) t
      if(t.gt.t_max) then
          type '(2a,i2)',' Parameter t is greater than max allowed',
     +                   ' value of ',t_max
          goto 1000
      elseif(mod(t,two).ne.one) then
          type '(a)',' Parameter t must be odd'
          goto 1000
      endif

      call skip_comments(lun,filename,name_len)
      read(unit=lun,fmt='(f5.2)',err=1000) psi
      if((psi.lt.1.0).or.(psi.gt.85.0)) then
          type '(2a)',' Parameter psi is out of the allowed range',
     +                ' of 1 to 85 degrees'
          goto 1000
      endif

      call skip_comments(lun,filename,name_len)
      read(unit=lun,fmt='(f7.5)',err=1000) pixel_size
      if(pixel_size.le.zero) then
          type '(2a)',' Parameter pixel size cannot be less than',
     +                ' or equal to zero'
          goto 1000
      endif

! Convert values for use by routine calculate_filter() since values read
!  in are for entire reconstruction and we only need to calculate 1st.
!  quadrant and for theta_tau .le. 90deg

      n = (n+1)/2 ! since we only need to do 1st. quadrant
      t = (t+1)/2 ! because of symmetry about theta=90deg
      psi = psi*(pi/180.0) ! convert to radians
      goto 1020 ! skip error handling

! Error handling

 1000 continue ! error exit point
      type '(a)',' Fatal error opening or reading parameter file'
 1010 continue ! non-error exit point
      type '(a)',' Exiting filter generation program'
      call exit

 1020 continue ! normal end of routine
      close(lun)
      end
      subroutine skip_comments(lun,filename,name_len)
!-----------------------------------------------------------------------------
! Lines with a non-blank first character are comment lines. This positions file
! so that the next non-comment line is the next one to be read in.
!-----------------------------------------------------------------------------
      implicit none
      logical comment_line
      integer lun,name_len
      character filename*30,in_line*265

      comment_line =.true.
      do while (comment_line)
      read(unit=lun,fmt='(a)',err=1000,end=1000) in_line
      if(in_line(1:1).eq.' ') comment_line =.false.
      enddo

! Go back 1 line since we just read a data line

      backspace(unit=lun,err=1000)
      goto 1010 ! skip error exit

 1000 continue ! error exit
      type '(3a)',' Fatal error trying to skip over comment lines in',
     +             ' parameter file',filename(1:name_len)
      type '(a)',' Exiting filter generation program'
      call exit

 1010 continue ! non-error exit point
      end
      subroutine make_filename(n,t,psi,pixel_size,filename,name_len)
C-------------------------------------------------------------------------------
C MAKE_FILENAME
C Makes a filename using the parameters n, t, psi, and pixel_size - in that
C order. pixel_size is scaled so that pixel_size=1 -> "D010"
C and psi is in degrees. n and t are set to the full reconstruction values
C
C History: written 7-mar-1988 PK
C-------------------------------------------------------------------------------
      implicit none
      integer int_psi,n,t,int_px,name_len,max_name_len,index
      parameter(max_name_len=50)
      real psi,pi,pixel_size,pixel_scale_factor
      parameter(pixel_scale_factor=10.0,pi=3.1415927)
      character*10 char_psi,char_px,char_n,char_t ! integers (*4) have 10 digits
      character*(max_name_len) filename, blank*1
      parameter(blank=' ')

! Convert values to ASCII characters

      write(unit=char_n,fmt='(i10)') 2*n-1
      if (char_n(9:9).eq.' ') char_n(9:9) = '0'
      if (char_n(8:8).eq.' ') char_n(8:8) = '0'

      write(unit=char_t,fmt='(i10)') 2*t-1
      if (char_t(9:9).eq.' ') char_t(9:9) = '0'

      int_psi = nint(psi*(180.0/pi))
      write(unit=char_psi,fmt='(i10)') int_psi
      if (char_psi(9:9).eq.' ') char_psi(9:9) = '0'

      int_px = nint(pixel_size*pixel_scale_factor) ! this will cause values
      write(unit=char_px,fmt='(i10)') int_px       ! too large or small
      if (char_px(8:8).eq.' ') char_px(8:8) = '0'  ! to be written as "000"
      if (char_px(9:9).eq.' ') char_px(9:9) = '0'
      if (char_px(10:10).eq.' ') char_px(10:10) = '0'

      filename='N'//char_n(8:10)//'T'//char_t(9:10)//
     +         'P'//char_psi(9:10)//'D'//char_px(8:10)//'.SPE'

      name_len = index(filename,blank)-1
      end
      subroutine hardcopy(n,t,psi,pixel_size,filter,par_file,
     +                    par_name_len,out_file,out_name_len)
C-------------------------------------------------------------------
C HARDCOPY
C Outputs a formatted copy of the filter to a specified file. Each theta
C index (angle) is printed on a new page along with the filter parameters
C and file names.
C
C History: written 4-mar-1988
C-----------------------------------------------------------------------
      implicit none
      integer n,t,ti,status,lun,i,j,max_width,name_len,index,
     +        par_name_len,out_name_len,max_name_len
      parameter(max_name_len=50)
      real filter(0:n-1,0:n-1,0:t-1),pixel_size,psi,delta_theta,
     +     pi,theta_tau,theta_bar,scale_factor,edgesum,penedge
      parameter(pi=3.1415927,scale_factor=1.0)
      character*(max_name_len) filename,answer,char1*1,par_file,
     +                         out_file,blank*1,fullname
      parameter(blank=' ')

! Declare external routines and condition codes

      integer LIB$GET_LUN
      include '($FORDEF)'

      filename = 'FILTER.LOG' ! Set to this for now
      name_len = index(filename,blank)-1

! Open file and write filter data to it, a new page for each angle

      status=LIB$GET_LUN(lun)
      if(.not.status) call LIB$SIGNAL(%val(status))
      open(unit=lun,file=filename(1:name_len),status='NEW')

      delta_theta = (psi/float(t-1))*(180.0/pi) ! convert to degrees

! Start big loop.

      do ti=0,t-1 ! Calculate filter values for each theta_tau.

      theta_tau = 90.0 - (psi*float(ti)/float(t-1))*(180.0/pi) ! in degrees
      theta_bar = acosd(cos(psi)/sind(theta_tau)) ! critical angle "   "

! Page header

      write(unit=lun,fmt='(a,//,a,f6.3,a)')
     +      '1','  Filter data for psi = ',psi*(180.0/pi),' degrees'
      write(unit=lun,fmt='(/,2a,/,2a,/,a,i3,a,i3,a,i2,a,f6.3,2a,f7.4)')
     +    '  Parameters are from file    : ',par_file(1:par_name_len),
     +    '  Data copied to section file : ',
     +    out_file(1:out_name_len),
     +    '  Filter size is ',2*n-1,' x ',2*n-1,' x ',2*t-1,
     +    '  Delta theta_tau = ',delta_theta,'deg.',
     +    '  Pixel size = ',pixel_size
      write(unit=lun,fmt='(/,a,f5.2,a,f7.1)')
     +    '  First quadrant of filter for theta_tau = ',theta_tau,
     +    ' degrees.     Data has been multiplied by',scale_factor
      write(unit=lun,fmt='(a,f5.2,//)')
     +    '  Critical angle theta_bar = ',theta_bar
```

```
! Print out filter data

      max_width = min(n,12)

      do j=n-1,0,-1
        write(unit=lun,fmt='(/,1x,I2,12(1x,g9.2))')
     +        j,(filter(i,j,ti)*scale_factor,i=0,max_width-1)
      enddo
      write(unit=lun,fmt='(/,12(8x,i2))')
     +     (i,i=0,max_width-1)

      enddo

! End of Big Loop

      close(lun)

! Get the full file name

      inquire(file=filename(1:name_len),name=fullname)
      name_len = index(fullname,blank)-1
      filename(1:name_len) = fullname(1:name_len)

      type '(2a)',' Filter generation info is in log file ',
     +            filename(1:name_len)
      end
```

Appendix E

Reconstruction Program Listing

```
      program recon
C-----------------------------------------------------------------------
C RECON
C Performs a 3D image reconstruction by the method described in my thesis.
C The projection data is in a 4D array in the file 'data.dat'. The
C reconstruction filter is in a 3D array in the file 'filter.dat'. The
C reconstructed image is stored in a 3D array in the file 'image.dat'.
C
C The first line of each file contains the necessary parameters for that
C data set. The data is all stored as  unormatted 4-byte real values.
C The arrays are all handled with subroutines. Variables that begin with
C d (for data) refer to the data array. Similarily for f (for filter) and
D i (for image).
C
C 1. The parameters from the data and filter files are checked to be sure
C    that the arrays are of the right size
C 2. Space for storing the 3 arrays is created and the data and filter
C    are read in.
C 3. The arrays and the parameters are passed to reconloop(), which does
C    all the work.
C 4. The image array returned from reconloop() is copied to a file and the
C    array storage space is all released.
C
C History: 2nd version written 10-sep-1987 Paul Kinahan
C
C---------------------------------------------------------------------------
      implicit none
      logical exists
      integer n,dn,dt,dp,fn,ft,dsize,fsize,isize,daddr,faddr,iaddr,
     +        lun1/1/,lun2/2/,lun3/3/,status,wordsize/4/,
     +        lib$get_vm,lib$free_vm,lib$init_timer,lib$show_timer,
     +        lib$free_timer,timer_addr/0/
      real psi,dpsi,fpsi
      character*(*) datafile,filterfile,imagefile
      parameter(datafile='data.dat',filterfile='filter.dat',
     +          imagefile='image.dat')

      status=lib$init_timer(timer_addr) ! start collecting program statistics.
      if(.not.status) call lib$signal(%val(status))

      inquire(file=datafile,exist=exists)
      if(.not.exists) call quit('No data file')
      inquire(file=filterfile,exist=exists)
      if(.not.exists) call quit('No filter file')

      open(lun1,file=datafile,status='old',form='unformatted')
      read(lun1) dn,dp,dt,dpsi
      open(lun2,file=filterfile,status='old',form='unformatted')
      read(lun2) fn,ft,fpsi
      if((dn.ne.fn).or.(dt.ne.ft).or.(dpsi.ne.fpsi))
     +    call quit('Filter and data file parameters do not match')
      n = dn ! use these values - since they are the same as fn and fpsi
      psi = dpsi

      dsize = (dn**2)*dp*dt*wordsize ! size of data array
      status = lib$get_vm(dsize,daddr) ! get storage space
      if(.not.status) call lib$signal(%val(status))
      call readdat(lun1,dn,dp,dt,%val(daddr))
      close(lun1)
      fsize = ((2*fn)**2)*((ft+1)/2)*wordsize ! size of filter array
      status = lib$get_vm(fsize,faddr) ! get storage space
      if(.not.status) call lib$signal(%val(status))
      call readfil(lun2,fn,ft,%val(faddr))
      close(lun2)

      isize = (n**3)*wordsize ! size of image array
      status = lib$get_vm(isize,iaddr) ! get storage space
      if(.not.status) call lib$signal(%val(status))
      call zeroimage(n,%val(iaddr)) ! zero image array

      call reconloop(n,fn,dp,dt,psi,%val(daddr),%val(faddr),
     +               %val(iaddr))

      open(lun3,file=imagefile,status='new',form='unformatted')
      write(lun3) n
      call copyimage(n,%val(iaddr),lun3)
      close(lun3)

      status = lib$free_vm(dsize,daddr) ! free up all temporary storage
      if(.not.status) call lib$signal(%val(status))
      status = lib$free_vm(fsize,faddr)
      if(.not.status) call lib$signal(%val(status))
      status = lib$free_vm(isize,iaddr)
      if(.not.status) call lib$signal(%val(status))

      type '(a)',' Program statistics'
      status=lib$show_timer(timer_addr) ! print out program statistics
      if(.not.status) call lib$signal(%val(status))
      status=lib$free_timer(timer_addr)
      if(.not.status) call lib$signal(%val(status))

      end ! of main routine

c----------------------------------------------------------------------
c  Auxilliary routines used by main routine
c----------------------------------------------------------------------
      subroutine quit(outstring)
      implicit none
      character*(*) outstring
      type '(2x,a)', outstring
      call exit
      end
c----------------------------------------------------------------------
      subroutine readdat(lun,n1,n2,n3,array)
      implicit none
      integer lun,n1,n2,n3,i,j,k,l
      real array(n1,n1,n2,n3)
      read(lun) ((((array(i,j,k,l),i=1,n1),j=1,n1),k=1,n2),l=1,n3)
      end
c----------------------------------------------------------------------
      subroutine readfil(lun,n,t,array)
      implicit none
      integer lun,n,t,i,j,k
      real array(2*n,2*n,0:(t-1)/2)
      read(lun) (((array(i,j,k),i=1,2*n),j=1,2*n),k=0,(t-1)/2)
      end
c----------------------------------------------------------------------
      subroutine zeroimage(n,array)
      implicit none
      integer n,i,j,k
      real array(n,n,n)
      do k=1,n
      do j=1,n
      do i=1,n
        array(i,j,k) = 0.0
      enddo
      enddo
      enddo
      end
c-------------------------------------------------------------------------
      subroutine copyimage(n,array,lun)
      implicit none
      integer lun,n,i,j,k
      real array(n,n,n)
      write(lun) (((array(i,j,k),i=1,n),j=1,n),k=1,n)
      end
      subroutine reconloop(n,fn,p,t,psi,data,filter,image)
C--------------------------------------------------------------------------
C RECONLOOP
C This filters the data with the filter and returns the reconstructed
C 3D image.
C
C The filtering and backprojection takes place one projection at a time.
C Phi_tau ranges between 0 and 180 deg. and theta_tau ranges between 90-psi
C and 90+psi. The sets of 2D filter data are indep. of phi_tau and are
C symmetrical about 90 deg. in theta_tau.
C
C Two temporary arrays are created for manipulating the data and are
C passsed to the called routines by their addresses.
C
C History : written 20-jun-1987 PK
C--------------------------------------------------------------------------

      implicit none
      integer n,fn,p,t,pn,psize,paddr,rn,ressize,resaddr,status,
     +        done,tot,pindex,tindex,tmax,tmin,
     +        lib$get_vm,lib$free_vm,wordsize/4/
      real data(n,n,0:(p-1),-(t-1)/2:(t-1)/2),
     +     filter(0:(fn-1)/2,0:(fn-1)/2,0:(t-1)/2),
     +     image(n,n,n),
     +     phi_tau,delta_phi_tau,theta_tau,delta_theta_tau,pi,psi
      parameter(pi=3.141592653)

      pn = n + 2*fn - 2 ! dimension of padding array (pn x pn)
      psize = (pn**2)*wordsize ! size of padding array
      status = lib$get_vm(psize,paddr) ! get storage space
      if(.not.status) call lib$signal(%val(status))
      call zero2darray(pn,%val(paddr)) ! initialize to zero

      rn = n + fn - 1 ! dimension of result array (rn x rn)
      ressize = (rn**2)*wordsize ! size of result array
      status = lib$get_vm(ressize,resaddr) ! get storage space
      if(.not.status) call lib$signal(%val(status))
      call zero2darray(rn,%val(resaddr)) ! initialize to zero

      tmin = -(t-1)/2 ! max value of theta_tau
      tmax =  (t-1)/2 ! min value of theta_tau
      delta_theta_tau = psi/float(tmax)
      delta_phi_tau   = pi/float(p)
      tot = t*p
      type '(a,i4,a)',' There are ',tot,' projections to do'
      done = 1
      type '(a,i4)',' doing projection ',done ! once now for overwriting.

C Reconstruction loop: filter and backproject one projection at a time.
C Because filters have rotational symmetry about z-axis, we alter theta last
C in the loop. The theta index (i) ranges from max. neg. value of theta (-psi)
C to max. pos. value of theta (+psi)

      do tindex=tmin,tmax ! repeat for each different filter
        theta_tau = pi/2.0 - float(tindex)*delta_theta_tau
        do pindex=0,p-1
          phi_tau = float(pindex)*delta_phi_tau
          type '(a,i4)','+ doing projection ',done
          done = done + 1
          call filterdata(pindex,tindex,n,fn,p,t,data,filter,
     +                       pn,%val(paddr),rn,%val(resaddr))
          call backproject(rn,%val(resaddr),phi_tau,theta_tau,n,image)
        enddo
      enddo
      type '(a)','+ Finished                    '
      end ! of filtering routine

C-----------------------------------------------------------------------
C Auxilliary routines
C-----------------------------------------------------------------------
      subroutine zero2darray(n,array)
      implicit none
      integer n,i,j
      real array(n,n)
      do j=1,n
      do i=1,n
        array(i,j)=0.0
      enddo
      enddo
      end
      subroutine filter_data(p_index,t_index,data_ndim,filter_ndim,
     +                       data_pdim,data_tdim,data,filter,pad_ndim,
     +                       pad,result_ndim,result)
C-----------------------------------------------------------------------
C FILTER_DATA
C Filters the 2D real data in data(,,,) by first putting the data in
C the over-sized zero-padded array pad(,), and then performing a 2D
C convolution with the array filter(,,) with the filtered data
C going into the array result(,). This entire operation is done for
C only one value of the higher dimension indicies (t_index,p_index).
C
C Arrays:-data(,,,) is dimensioned s.t. data(,,0,0) refers to phi=0 and
C        theta=pi/2
C       -filter(,,) is dimensioned s.t. filter(0,0,0) refers to theta=pi/2
C        and x=y=0
C Note that filter_t_index = abs(t_index)
C
C Because of 4-fold filter symmetry, we only use 1st. quadrant of filter
C values (thus the dimensioning of the array filter(,,)).
C
C The matching of the array dimensions is critical, but since this routine
C can be called very often, we'll leave the dimensional checking to the
C calling routine
C
C History: written 20-jun-1987 PK
C-----------------------------------------------------------------------
      implicit none
      integer data_ndim,data_pdim,data_tdim,filter_ndim,pad_ndim,
     +        result_ndim,t_index,p_index,fti,pad_offset,
     +        li,lj,i,j,filter_offset,fo
C Arrays
      real data(data_ndim,data_ndim, 0:data_pdim-1,
     +                  -(data_tdim-1)/2:(data_tdim-1)/2)
      real pad(pad_ndim,pad_ndim)
      real filter(0:filter_ndim-1, 0:filter_ndim-1, 0:(data_tdim-1)/2)
      real result(result_ndim,result_ndim)

C Abbreviations

      fti = abs(t_index) ! since filter is symmetric about 90 degrees
      pad_offset = 2*filter_ndim - 2 ! using full filter size here
      filter_offset = filter_ndim - 1 ! using full filter size here
      fo = filter_offset

C Copy data into centre of zero-padded array

      do j=1,data_ndim
      do i=1,data_ndim
          pad(i+pad_offset,j+pad_offset) = data(i,j,p_index,t_index)
      enddo
      enddo

C Filter

      do lj=1,result_ndim
      do li=1,result_ndim

  ! first add contribution at origin of filter

          result(li,lj) = filter(0,0,fti)*pad(li+fo,lj+fo)
```

```
! then add contributions from axes, using 2-fold symmetry along axes

      do i=1,filter_ndim-1 ! x-axis first
      result(li,lj) = result(li,lj)
  +             + filter(i,0,fti)*( pad(li+fo-i,lj+fo)
  +                               + pad(li+fo+i,lj+fo))
      enddo

      do j=1,filter_ndim-1 ! y-axis
      result(li,lj) = result(li,lj)
  +             + filter(0,j,fti)*( pad(li+fo,lj+fo-j)
  +                               + pad(li+fo,lj+fo+j))
      enddo

! finally do the rest of the filter, only using the 1st. quadrant
! because of 4-fold filter symmetry

      do j=1,filter_ndim-1
      do i=1,filter_ndim-1
      result(li,lj) = result(li,lj)
  +                  + filter(i,j,fti)*( pad(li+fo-i,lj+fo-j)
  +                                    + pad(li+fo-i,lj+fo+j)
  +                                    + pad(li+fo+i,lj+fo-j)
  +                                    + pad(li+fo+i,lj+fo+j))
      enddo
      enddo
    enddo
    enddo
    end
```

```
      subroutine backproject(m,plane,phi,theta,n,volume)
C------------------------------------------------------------------------
C BACKPROJECT
C Backproject a 2D array (at a given theta and phi) into a 3D array by
C forward-projecting the center of each voxel onto the 2D array and
C using bilinear interpolation. See thesis for explanation of the algorithm
C
C History: written 23-jun-1987 PK
C------------------------------------------------------------------------
      implicit none
      integer m,n,li,lj,i,j,k
      real plane(m,m),volume(n,n,n),theta,phi,lx,ly,dx,dy,p,v

      p = 1.0 ! pixel size. For now there is no reason to differ from this
      v = 1.0 ! voxel size.

      do k=1,n
      do j=1,n
      do i=1,n
       ! find forward projected point on plane
          lx = (v/p)*( -(i-(float(n)+1.0)/2.0)*sin(phi)
     +                 +(j-(float(n)+1.0)/2.0)*cos(phi) )
          ly = (v/p)*( -(i-(float(n)+1.0)/2.0)*cos(theta)*cos(phi)
     +                 -(j-(float(n)+1.0)/2.0)*cos(theta)*sin(phi)
     +                 +(k-(float(n)+1.0)/2.0)*sin(theta) )
       ! find nearest gridpoints
          li = int(lx + (float(m)+1.0)/2.0)
          lj = int(ly + (float(m)+1.0)/2.0)
       ! interpolation parameters
          dx = (lx - float(li)) + (float(m)+1.0)/2.0
          dy = (ly - float(lj)) + (float(m)+1.0)/2.0
       ! add interpolated value to voxel
          volume(i,j,k) = volume(i,j,k)
     +                   + (1.0-dx)*(1.0-dy)*plane(li,lj)
     +                   + (dx)*(1.0-dy)*plane(li+1,lj)
     +                   + (1.0-dx)*(dy)*plane(li,lj+1)
     +                   + (dx)*(dy)*plane(li+1,lj+1)
      enddo
      enddo
      enddo
      end
```