

IMPLEMENTATION AND ANALYSIS OF 3-D TOMOGRAPHIC RECONSTRUCTIONS
FROM SPACE-BASED IMAGING PLATFORMS

by

Garrett Wesley Hinton


Copyright © Garrett Wesley Hinton 2021

A Dissertation Submitted to the Faculty of the
COLLEGE OF OPTICAL SCIENCES
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
In the Graduate College
THE UNIVERSITY OF ARIZONA

2021

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by **Garrett Wesley Hinton**, titled *Implementation and Analysis of 3-D Tomographic Reconstructions from Space-Based Imaging Platforms* and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.



Professor Matthew A. Kupinski

Date: April 30, 2021



Professor Eric W. Clarkson

Date: April 30, 2021




Professor Michael Hart

Date: April 30, 2021



Dr. Jed J. Hancock

Date: April 30, 2021

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College. 

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.



Professor Matthew A. Kupinski
Dissertation Committee Chair
Wyant College of Optical Sciences

Date: May 4, 2021

ACKNOWLEDGEMENTS

My experience in research and with every aspect of this dissertation has been made possible because of the support of my wife, Stephanie. Thank you for pushing me to be better and to pursue this goal. I am grateful to have someone like you to support me and to make me laugh during the hard times. My two young children, Ruby and Weston, have also provided entertainment, motivation, and sometimes some extra text in my dissertation that I would later discover and erase. My parents, Scott and Sharon Hinton, have given me counsel and support to get me to this point. If it had not been for a brief discussion with them at the end of my Junior year in college, I would have never considered getting a PhD and maybe even a master's degree.

I am grateful for my education and research experiences before coming to the University of Arizona that paved the way for me to get where I am. As an undergraduate, I did research for Jon Takemoto, which required the use of a confocal fluorescence microscope. This was the first interest that I had in optics as I become more interested in the microscope than in my other research. Thank you, Jon, for providing that experience. Also, in a turn of unexpected good luck, Jed Hancock ended up teaching an optics class my Senior year. His class explored the world of optics and helped me to decide to pursue this field of research. Thank you, Jed, for your support and your selfless labor of teaching a course that you did not have to teach. Thank you for also keeping in touch with me and helping me to see the kind of opportunity that the Space Dynamics Laboratory would be for me.

My research experience was made possible by Dr. Matthew Kupinski, who has supported my research and helped me to become more self-reliant in finding solutions to the problems presented in my research. His patience with me helped me to find a path forward and a natural place for my research to go. It was also inspiring to have zoom calls with him where his kids would interrupt, and he would patiently pause the meeting for a minute to talk with them. Behind the scenes of those meetings, my kids were interrupting too, and it helped me to see that family comes first.

Thank you, Dr. Harrison Barrett, who has always been supportive and helped me to find my research interests. Your friendliness helped me to decide to go to the University of Arizona. Thank you, Dr. Lars Furenlid, who was also patient with me and is an effective and inspiring researcher and teacher, always looking out for your students. Thank you, Dr. Eric Clarkson, for your help in my research and in OPTI 637, which was one of the most influential courses that I took and tied into my research in many ways.

Thank you to the Space Dynamics Laboratory and those who work there for providing a professional and friendly environment where quality research can be done.

This research was made possible by the Department of Defense and the Space Dynamics Laboratory.

DEDICATION

Dedicated to my wife, my children, and my parents

TABLE OF CONTENTS

1.1	Overview of the Atmospheric Waves Experiment	12
1.2	ISS as an Imaging Platform	12
1.3	AWE Background	12
1.4	AWE Instrument Overview	13
1.5	OH-Airglow Layer	15
1.6	Gravity Waves	16
1.7	AWE Detector Characteristics	17
1.7.1	Noise Characteristics	17
1.7.2	Correlated Double Sampling	18
1.7.3	Low Light Imaging	18
1.7.4	MTF Correction	19
1.8	Reflections in Nadir Imaging	20
2.1	Algorithms in Tomography	23
2.2	MLEM	24
2.2.1	Implementation of MLEM	24
2.2.2	Current State of MLEM	25
2.2.3	Penalized Maximum Likelihood (PML)	25
2.2.4	Implementation of PML	27
2.2.5	Current State of PML	27
2.2.6	MLEM-Total Variation (MLEM-TV)	27
2.3	GPU programming and Iterative Techniques	29
2.4	Image Quality	30
2.4.1	Developing Figures of Merit	30
2.4.2	Classification Tasks	30
2.4.3	Estimation Tasks	31
2.4.4	Developing Observers for Detection Tasks	32
2.4.4.1	Hotelling Observers	32
3.1	Implementation Overview	35
3.2	GPU and code discussions	36
3.2.1	Software implementation and commentary of MLEM	36
3.2.2	GPU Programming in Tomography	36

3.3	Basis Background for imaging simulations.....	37
3.4	Projection Algorithms (Sphere Voxels, standard delta basis projection).....	39
3.4.1	Common methods for forward and back projections.....	39
3.4.1.1	Distance Driven (DD) Approach	39
3.4.1.2	Separable Footprint (SF) Approach.....	39
3.4.2	Delta Basis Projection Algorithm.....	39
3.4.3	Sphere Voxels Projection Algorithm	40
3.5	Back-projection Algorithms.....	41
3.6	Testing of Projection Operators	41
3.6.1	Images and Commentary of Overall Projections	44
3.6.2	Discussion of Projection Error Images	47
3.6.3	Preliminary Decisions with Sphere Voxel Radius'	47
3.6.4	Overall Comparison to Distance Driven Projector	48
4.1	Introduction To Tomography From Space	51
4.2	Limited-Angle Tomography from Space.....	51
4.2.1	Increase the Number of Iterations.....	52
4.2.2	Adding More Detector Pixels	52
4.2.3	Initializing Voxel Space in Altitude-Axis	53
4.2.4	Total Variation (TV) Regularization	53
4.2.5	Adding More Projections	53
4.2.6	Different Iterative Techniques	54
4.2.7	Point Spread Function.....	55
4.2.8	Focusing on a Region of Interest.....	55
4.3	Implementation of Tomographic Reconstructions from Space	56
4.3.1	Projection Algorithm	56
4.3.2	MLEM Parameters and Speed.....	58
4.4	Results and Discussion	59
4.4.1	Initial Images	59
4.4.2	Altitude-Axis Resolution.....	61
4.5	Conclusion	66
5.1	Introduction	67
5.2	Models.....	67
5.2.1	Objects and Imaging.....	67

5.2.2	SWIR Model.....	67
5.2.2.1	Temperature Model	67
5.2.2.2	Gravity Wave Model.....	69
5.2.2.3	Reflection Model	70
5.2.2.4	Combining the Models	72
5.3	Image Quality	73
5.3.1	Defining a Task for Detecting Gravity Waves.....	73
5.3.2	Figures of Merit.....	73
5.4	Simulation Results	74
5.4.1	Amplitude, Method, and Detectability	75
5.4.2	Reconstruction Algorithm Iterations and Detectability	76
5.4.3	Frequency and Detectability	77
5.4.4	SNR and Detectability	78
5.4.5	Projections and Detectability.....	79
5.4.6	Nadir Imaging and Focused Imaging.....	80
5.4.7	Field of View and Detectability	81
5.4.8	TV Regularization and Initializing Voxels.....	81
5.5	Discussion and Looking Forward	82
5.5.1	MLEM Discussion	82
5.5.2	Wavelength Discussion	82
5.5.3	Discussion on the Number of Iterations	83
5.6	Conclusion	83
6.1	Review and Conclusions.....	84
6.2	Future Work	85
6.2.1	Altitude-Axis Resolution.....	85
6.2.2	AWE Model and Real AWE Data	86
6.2.3	Increasing the Speed and Accuracy of MLEM	86
6.3	Future of Tomography in Space	86
A.1	Back Projection Code	87
A.2	Projection Code	91
A.3	Projection Using Sphere Voxels.....	92

LIST OF FIGURES

Figure 1.1: Cross section of AWE	15
Figure 1.2: OH-airglow Temperature and Altitude	16
Figure 1.3: Components of AWE MTF.....	19
Figure 1.4: Overall MTF of AWE	20
Figure 1.5: MODTRAN analysis of transmittance through the atmosphere.....	21
Figure 1.6: Satellite Trajectory around Earth.....	22
Figure 2.1: Different images with the same mean-square error value.....	31
Figure 3.1: Delta Basis and Pixel Basis	38
Figure 3.2: An illustration of the sphere voxel projection algorithm.....	41
Figure 3.3: Sphere Voxel without line weights: projections of all ones.....	44
Figure 3.4: Sphere Voxel with line weights: projections of all ones	45
Figure 3.5: Distance driven: projections of all ones.....	45
Figure 3.6: Sphere Voxel without line weights: projections of a sphere	46
Figure 3.7: Sphere Voxel with line weights: projections of a sphere.....	46
Figure 3.8: Distance driven: projections of a sphere	47
Figure 3.9: Average projection error vs projection angle	48
Figure 3.10: Average projection error vs distance from center pixel	49
Figure 4.1: Pseudo code for a projection algorithm	57
Figure 4.2: Projection error plots for a 72° full field of view	58
Figure 4.3: Images from a simulated OH-airglow model	60
Figure 4.4: A slice of a reconstructed model	60
Figure 4.5: Image of the OH-airglow layer taken from Eckermann et al. 2016... ..	61
Figure 4.6: Slices of reconstructions with two boxes	62
Figure 4.7: Slices of reconstructions with two boxes: other methods	63
Figure 4.8: Slices of reconstructions of MLEM with 8 iterations	64
Figure 4.9: Slices of reconstructions of MLEM with 50 iterations	65
Figure 5.1: OH-airglow temperature model	68
Figure 5.2: Gravity waves in the atmosphere	69
Figure 5.3: Image from NIRAC.....	71
Figure 5.4: Reflection layer in the simulated model.....	72
Figure 5.5: OH-airglow layer model without any gravity waves.....	73
Figure 5.6: Amplitude vs Detectability.....	75
Figure 5.7: Iterations vs Detectability.....	76
Figure 5.8: Frequency vs Detectability.....	77
Figure 5.9: SNR vs Detectability	79
Figure 5.10: Number of Projections vs Detectability	80

LIST OF TABLES

<i>Table 3.1: Name, description, and formula for different simulated sizes of sphere voxels</i>	<i>43</i>
<i>Table 4.1: Summary of Methods to Improve Altitude-Axis Resolution</i>	<i>56</i>
<i>Table 4.2: Reconstruction Parameters.....</i>	<i>58</i>

Abstract

Imaging satellites that look nadir face a variety of obstacles. In addition to designing the system for the intense environment that the satellite will be experiencing, there are other factors to consider: reflections and emissions from the ground, from clouds, and from the OH-airglow layer. Depending on the desired object, these nuisance signals can significantly reduce image quality. The ground will have city lights, clouds will reflect light, and every material will have a different reflectance, some up to 60%. Performing a tomographic reconstruction can effectively separate a signal from other emissions and reflections. The Atmospheric Waves Experiment (AWE) is a prime example for use of tomographic reconstruction techniques from an imaging space platform. AWE is designed for studying the OH-airglow layer and atmospheric waves (also called gravity waves) which cause emission changes in the OH-airglow layer. A reconstruction for AWE would separate signals from the OH-airglow layer from reflected light from clouds and the ground.

Performing tomographic reconstructions for the Atmospheric Waves Experiment (AWE) and analyzing them is the primary focus of this dissertation. This work covers an implementation of MLEM for use in satellite images pointing nadir. The algorithm is fast enough to be performed in real-time for many applications. This work covers the details of the reconstruction implementation and the challenges it poses and then a detailed study of the image quality of the tomographic reconstructions is presented. Some of the useful tools developed during this study include the construction of a short-wave infrared (SWIR) model of the atmosphere, methods for projecting simulated models through the imaging system, performing tomographic reconstructions of the simulations, and using a Hotelling observer to determine the overall image quality. Tomographic reconstructions are found to be effective in many applications for space imaging. However, the severely limited projection angles do provide constraints on the overall reconstructed resolution.

CHAPTER 1

THE ATMOSPHERIC WAVES EXPERIMENT

1.1 Overview of the Atmospheric Waves Experiment

The Atmospheric Waves Experiment (AWE) is a short-wave infrared (SWIR) telescope that will be mounted onto the International Space Station (ISS). It will be pointed nadir and will collect signals from emissions of the OH airglow layer. The OH-Airglow layer is a region of altitude where atmospheric pressure waves, also called gravity waves (not to be confused with gravitational waves), deposit portions of their energy. These gravity waves connect terrestrial weather with space weather. Space weather has been of interest lately because of its effect on GPS, radio communications, and other technology. AWE's purpose of studying the airglow layer will provide key insights to the relationship that gravity waves play in space weather (AWE 2018, Potter 2020).

1.2 ISS as an Imaging Platform

The ISS is a unique and useful platform for a telescope. Its orbit covers $\pm 52^\circ$ latitudes (National Aeronautics and Space Administration 2010), and it orbits the earth every 90 minutes (Evans and Robinson n.d.). Moving at 7.66 km/sec, integration times can be prohibitive for imagers. As is shown in Section 1.6, the MTF correction for ISS movement becomes dominant. For this reason, the integration time for AWE is to be around one second.

1.3 AWE Background

There has been research that has looked at the OH-airglow layer from space (Miller et al. 2015, Chandran et al. 2010, Tsuda 2014), and some have even used tomography for 3D reconstructions of the OH-airglow layer (Song et al. 2017). Many of these satellites that have looked at the OH-airglow layer have focused on looking sideways into the OH-

airglow layer (called a limb measurement) to emphasize the differences in the emissions at different altitudes. SABER (Oberheide et al. 2006) is one of these instruments that makes limb measurements. What sets AWE apart from these other satellites is its wide observation coverage of the Earth due to being attached to the ISS, it is looking nadir, and it has four telescopes simultaneously collecting sufficient data to create temperature maps of the OH-airglow across these wide swaths.

Some previous groups have looked at the OH-airglow layer from the ground looking up (Taylor et al. 2010, Pautet et al. 2014, Zhao 2019, Lai et al 2019, Gavrilieva et al 2018, Matsuda et al 2014). The Advanced Mesospheric Temperature Mapper (AMTM), used by Taylor et al. (2010), served as a starting point in designing a telescope for space. There are differences that had to be considered when designing for space. Ground based imaging platforms are difficult to move, and therefore have a limitation in how much of the airglow layer can be studied. To make more of the sky visible, the full field of view of the AMTM was made to be 120 degrees. Ground-based imaging platforms have also allowed for longer, heavier telescopes.

1.4 AWE Instrument Overview

In adjusting the AMTM used by Taylor et al. (2010), many minor adjustments had to be made in the design. This section will briefly discuss some of the design considerations that went into the AWE instrument.

The mass and volume of the instrument had to be considered. Because the mass of the AMTM was too high, the lens barrel was shortened, and the lenses had to be lighter. In addition to lightening lenses, the number of lenses also had to be reduced. In considering lenses, the AMTM design had some cemented doublets that had to be adjusted because the thermal properties of cemented doublets were not adequate for the requirements of AWE. As an effect of these mass and volume considerations, as well as stray light considerations for space, the full field of view was narrowed to be around 90°.

Smaller and tighter tolerances were required for AWE. As a result of fewer lenses, each lens had to have a tighter tolerance to meet the design requirements. This led to more advanced alignment tools and techniques, as well as mounting techniques to ensure the

telescope could function properly (AWE 2018). The focus adjustment found on the AMTM had to be abandoned because once AWE was placed on the ISS, no operator could adjust the focus. This put priority on calibrating the instrument and having a reliable, athermal design.

The environment of space also led to minor changes. Radiation in space made many glass types unusable for lenses. Radiation hardened glass was required, especially for the outer lenses. Other environmental requirements to prepare for were the vibrations in the launch environment; thermal considerations which included survival and operating temperatures; and athermal design considerations to enhance the image quality throughout the operating temperature range.

On the ground, the OH-airglow layer and gravity waves incident on the layer move slow enough that exposure times could be longer than 10 seconds (Hart et al. 2012). The long exposure times gave increased signal-to-noise ratio (SNR) for the collected images. On the ISS, the velocity forces the integration time to be closer to one second. A balance of integration time and image smear had to be made to get adequate images. With the SNR sacrificed because of shorter integration times, other methods were considered being used to bring the SNR up to adequate levels, such as coadding frames and tomographic reconstructions.

On the ground, three different narrowband filters were used in sequence to get a temperature map of the OH-airglow layer. With there being limited time to change filters on orbit, four separate telescopes were made so that the filters could be used simultaneously. This allowed one extra filter for resiliency. An InGaAs detector was chosen that had satisfactory noise characteristics. The selected detectors were meant for high signal applications, but because of price and availability, they were chosen. The noise was high enough that correlated double sampling (CDS), was used to lower the noise (White et al. 1974). The detector was also cooled to -15°C to reduce noise and dark current.

Distortion, spectral response, and sensitivity to other signals was part of the calibration process needed for space. For calibration purposes, the outer 16 pixels on each side of the detector were masked with metal. This guaranteed that a dark frame measurement could be taken on orbit. This also helped with measuring the temperature of the detector because

the dark current is correlated with temperature. More detail on this topic is given in Section 1.6.3.

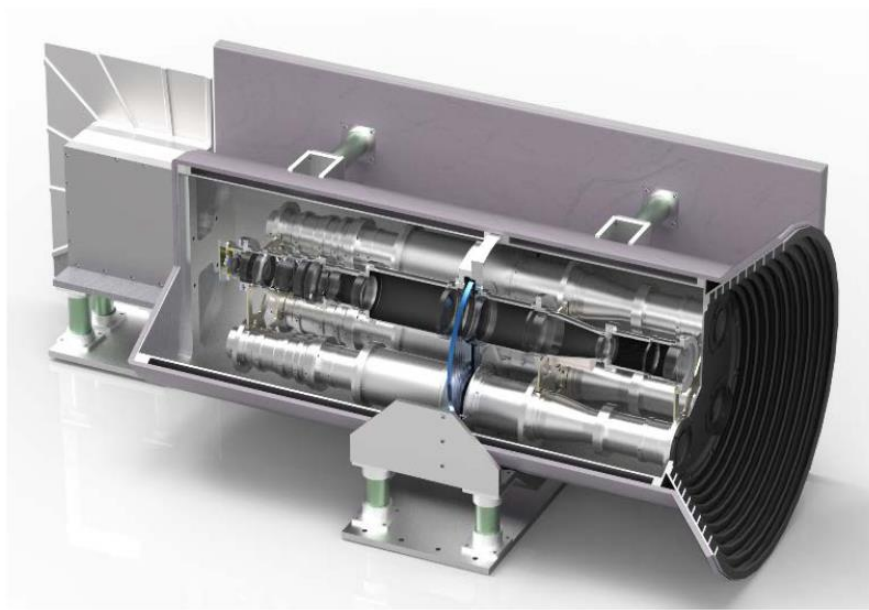


Figure 1.1: Cross section of AWE: There are four different telescopes. At least three of the four telescopes will have a different spectral filter. These spectral filters will help to provide images that, when combined, give a temperature map of the OH-airglow layer.

1.5 OH-Airglow Layer

Between the mesosphere and the thermosphere, at a roughly 87 km altitude, is the OH-airglow layer. It has a width of roughly 10 km at any given location, and its emissions can shift from 80-100 km. The temperature in the OH-airglow layer will generally be between 150-240K (Ammosov et al. 2019).

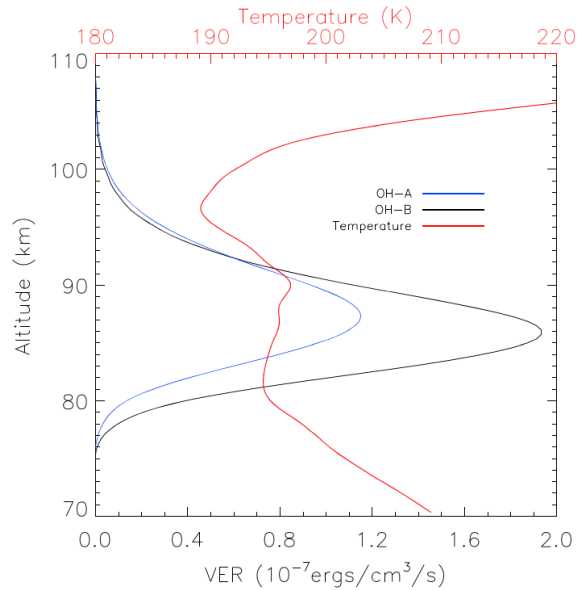


Figure 1.2: Taken from Liu et al. in 2015. This figure shows the SABER instrument's data. These are two channels of mean OH emissions displayed. The red line gives the mean values of temperatures. Used with permission from the Journal of Geophysical Research: Space Physics.

The airglow layer has emissions in the near infrared due to vibrational-rotational transitions. In vibrational-rotational transitions, a molecule's vibrations are analyzed like a harmonic oscillator with quantized energy states. The wavelengths of the emissions are due to vibrational energy level transitions combined with a change of total angular momentum due to rotational transitions, which have a selection rule of $\Delta J = \pm 1$ (Libretexts 2020). In this case, these transitions give rise to sharp peaks throughout the near-infrared spectrum (Oliva 1992, Ramsay 1992). The peak amplitudes are dependent on the temperature of the airglow layer, following a Boltzmann distribution (Dreier and Rakestraw 1990, Libretexts 2020). When looking at the ratio between vibrational-rotational transition band emissions (e.g., $P_1(2)$ and $P_1(4)$), an accurate temperature reading can be performed.

1.6 Gravity Waves

Atmospheric gravity waves, called gravity waves in this document, transport energy through the atmosphere and are a vital, little understood phenomenon that connect weather in the troposphere with space weather. Space weather is of interest to many groups because of communication disruptions and other issues that can occur (Akmaev 2011). Most people have experienced gravity waves through airplane turbulence, but gravity waves extend far

beyond those altitudes. If they get high enough, they get to the OH-airglow layer and beyond. When reaching the airglow layer, the waves begin to break, or in other words, they deposit energy into the OH molecules. In this way they are similar to waves in the ocean: they are present under water, but their energy is deposited at the boundary of where the water meets the air.

Gravity waves start from many sources: storms, other weather, mountains, valleys, islands, other geographic features, and many more. Nappo (2013) explains that at these events and locations, buoyancy forces propagate the air up and out, and gravity is the restoring force that pulls them down, creating a wave. When propagating, they can go almost any direction. If they go in a direction that increases with altitude, their amplitudes also increase. Nappo (2013) also explains that gravity waves can also break up into groups of smaller waves.

When gravity waves are incident on the OH-airglow layer, they cause pressure changes, which then cause temperature changes in the airglow layer due to the ideal gas law, $PV = nRT$, where P is pressure, V is volume, n is the amount of the gas present, R is a constant, and T is temperature. The temperature changes are related to the emission radiance of the airglow layer, as mentioned in Section 1.5 of this document. If these emissions across the airglow layer can be measured with proper precision and resolution, the gravity waves in the OH-airglow layer can be quantitatively measured as well as the temperatures of OH-airglow layer.

1.7 AWE Detector Characteristics

1.7.1 Noise Characteristics

The detectors on AWE experience much higher noise than traditional detectors that are flown in space. Grouping the noise into three dominant groups, there is read noise, dark noise, and extra noise. The read noise is the dominant noise source. Correlated double sampling (CDS), first introduced by White et al. (1974), was used to bring this noise down to acceptable levels. In the low emissions cases, the readings on the detector will be in the bottom 1-2% of its dynamic range. Because this dark current is present, there is also dark current noise that is present. This is not as dominant as the read noise, but it is significant. There is also noise that is labeled 'extra noise'. This noise ($\sim 70e^-$) comes from the read-

out integrated circuit (ROIC). Much of this noise comes from suspected ROIC Glow, which is explained in more detail in Le Goff (2020). The MOSFETs, temperature sensors, and other components near the focal plane array (FPA) have a glow in the SWIR region. This glow adds an unwanted signal, similar to dark current, to the detectors. This glow, unlike the dark noise, is less uniform and more unpredictable. To better control this ROIC glow, the detector control settings were tuned to give the lowest noise possible.

1.7.2 Correlated Double Sampling

Because the read noise was too high, a method called correlated double sampling (CDS) was used to bring the read noise down to acceptable levels. The method is given in detail in White et al. (1974). Read noise is independent of integration time (Janesick 2007). CDS takes two images: one at near-zero integration time, and one at the desired integration time. When both images are taken, the zero-integration time image is subtracted from the standard image. This will subtract off a portion of the read noise and still result in an image with the desired integration time.

1.7.3 Low Light Imaging

The AWE detectors will mostly be operating in the bottom 5% of their operating ranges. The detectors are 320x256 FPAs. Imaging in a low-light environment presents many challenges for detectors. Challenges to consider include noise, dark current gradients, temperature, and detector linear response. These are all related to each other and must be carefully tuned. AWE is keeping the temperature at roughly -15°C , which will have a higher dark signal and a higher noise than at lower temperatures.

When controlling detector temperatures, there is a chance for temperature gradients. Temperature gradients will create dark current gradients on the detector, further complicating calibration. One method is to mask the outer 16 rows of pixels on each side so that they only give a dark current reading. These dark current readings act as thermometers to guide the calibration of the images so that temperature gradients can be detected and are not an issue. As a result, the usable pixels on each detector will be the middle 256x256 pixel grid.

1.7.4 MTF Correction

The modulation transfer function is important because of the desired gravity wave wavelength range of 30-300km. An MTF analysis calculates how much of these frequencies will get through (Barrett and Myers 2004). The main items that will affect the MTF are the optical design, detector optical crosstalk, detector electric crosstalk, ISS jitter, ISS speed, pixel size, and possibly coadding. Their contributions to the MTF are shown in Figure 1.3 with ISS smearing and pixel size being combined on the same image.

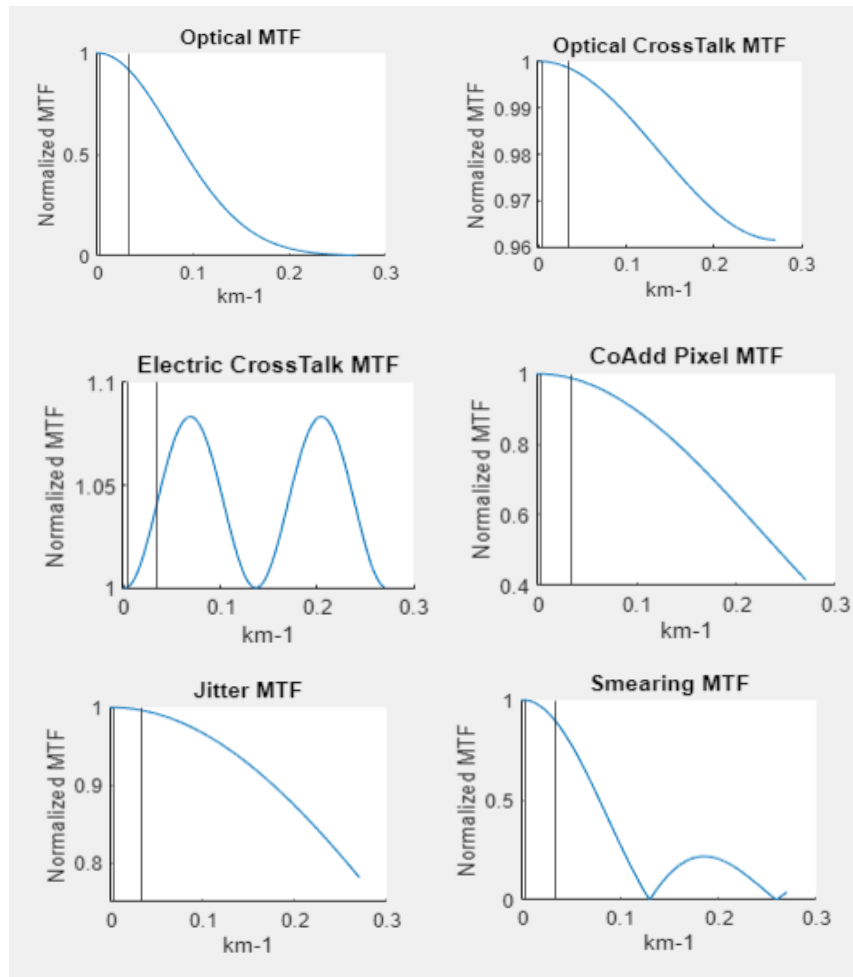


Figure 1.3: Components of the Overall MTF. The left vertical line on each graph corresponds to visibility for 300km waves and the rightmost vertical line corresponds to visibility for 30km waves. It is important to note that the electric crosstalk values range from 1 – 1.08.

The overall 2-dimensional MTF is given in Figure 1.4.

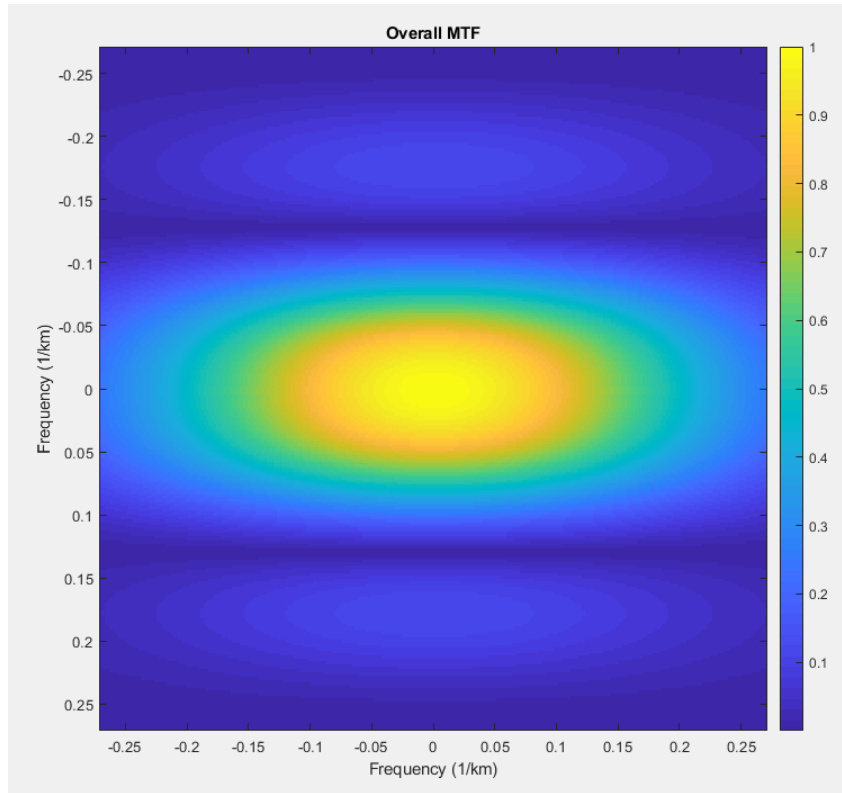


Figure 1.4: The overall MTF for the AWE system. It is asymmetric due to the ISS smearing and electric crosstalk being in only one direction.

For a 30km wave, the current best estimate is for 90% of the signal visibility to make it through. For a 300km wave, it is >99%. If the optical PSF, optical crosstalk, or jitter were to get worse, the amount of 30km signal visibility that gets through could go down to as little as 80%. For the desired wavelength range, the MTF shows that there will be adequate amounts of visibility.

1.8 Reflections in Nadir Imaging

In observing the OH-Airglow layer, it is important to account for reflections. Different surface types will have different reflectivity. Different surfaces may also have different scattering properties. Some reflective surfaces have been reported to be as high as 60% for SWIR wavelengths (Tian and Philpot 2015, Curcio et al. 2013). For AWE, there are three different wavelengths that will have to be accounted for: 1400nm, 1524nm, and 1543nm. The atmospheric transmission of a range of wavelengths taken from MODTRAN software is shown in Figure 1.5 (Kaushal et al. 2017). For the 1400nm band, there will be almost

no reflection from the ground because that wavelength will not transmit well through the atmosphere.

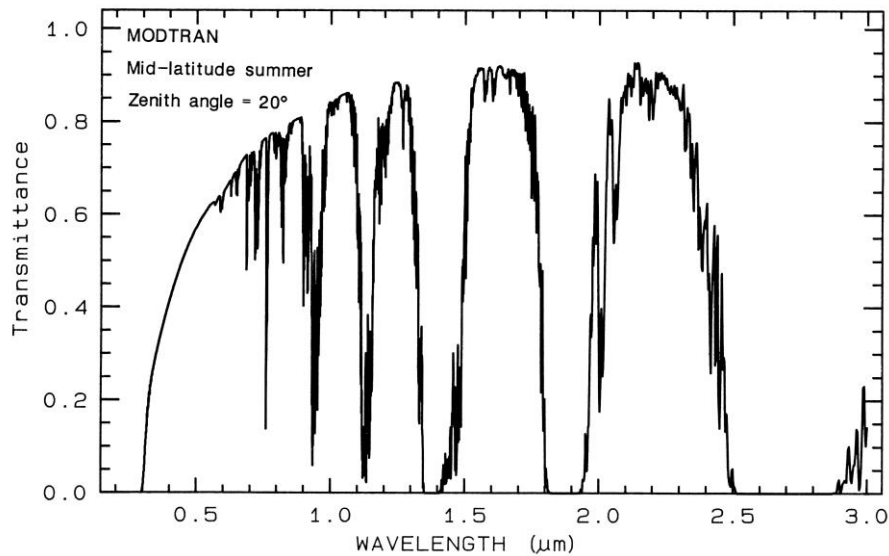


Figure 1.5: MODTRAN analysis of transmittance through the atmosphere. This image is in the public domain.

Another potential issue for nadir imagers is city lights. AWE is designed to gather night images, which is when city lights will be interfering. AWE's ultra-narrow bandpass filters (~1 nm wide) will provide some protection against this issue. Another help for AWE is that its background filter and processing will help to filter out the city lights. City lights are therefore not a primary issue that AWE will be facing.

The reflections and city lights will be superimposed with the emissions from the OH-airglow layer. The resulting images will have these nuisance reflections that will degrade image quality. The cloud reflections will most likely correlate with the emission values of the OH-airglow layer immediately above them, with some of the signal possibly scattering to other areas.

With the ISS moving over a region, the ground, clouds, and emissions will have different altitudes and different viewing angles. In other words, the same spot in XY space with different amplitudes will show up in different pixels on the detector depending on the ISS position. This is shown in Figure 1.6. With the gravity waves being roughly stationary compared with the speed of the ISS, and with the FOV being so wide, tomographic

reconstructions are a practical and straightforward method of separating the desired emissions of the OH-airglow layer from the undesired reflections closer to the ground.

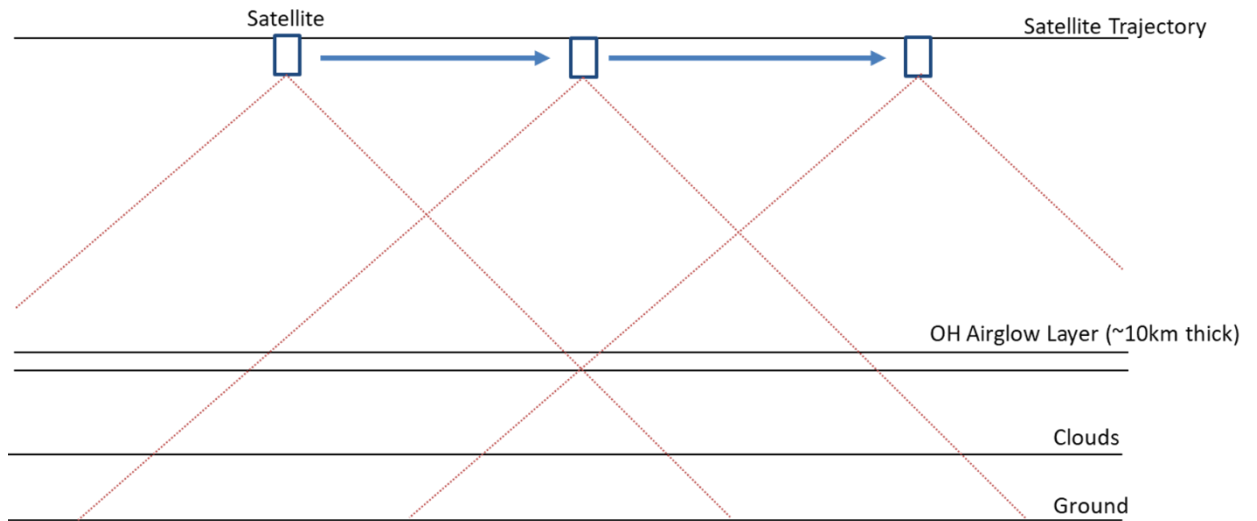


Figure 1.6: This shows a satellite's trajectory around the Earth. As it goes along its trajectory, it has many different viewing angles of the OH-airglow layer. Each viewing angle will have different background emissions. For example, the point in the center of the OH-airglow layer in this figure will have different background emissions contributing to the imaging pixels depending on the viewing angle of the ISS.

The tomographic reconstructions used for this imaging will be slightly different in nature compared to typical reconstructions found in research. They will be severely limited on projection angles. A 90° FFOV, though wide for optical imaging, may not give enough of an altitude-axis presence to completely separate out the reflections from the emissions.

CHAPTER 2

PRINCIPLES AND APPLICATIONS OF TOMOGRAPHY

2.1 Algorithms in Tomography

There are many algorithms for tomographic reconstruction. One way to categorize them is by separating them into three main groups: analytical (FDK, FBP, Katsevich, etc), algebraic (ART, MLEM, etc.), and statistical (MBIR, etc.) (Zhang et al. 2018). Different papers will have many different meanings for these terms, but to clarify for this document, algebraic algorithms have no regularization terms added, but statistical algorithms can. Algebraic algorithms, which are iterative, have been shown to produce better image quality than analytical algorithms in certain cases (Nam et al. 2019, Vaniqui et al. 2019). They require no prior knowledge about the object.

A statistical algorithm would be best for many situations, but this requires prior information about the object and background being imaged. Industry groups have currently implemented model based iterative reconstruction (MBIR) algorithms, which require prior information. These algorithms have different acronyms but are similar in that they use MAP estimations. These estimations have been shown to give higher image quality than some of their statistical iterative reconstruction (SIR) algorithm counterparts (Chang et al. 2019). One group has successfully integrated an MBIR algorithm with iterated coordinate descent (ICD) onto a GPU (Sabne et al. 2017). Another group has shown a separate algorithm with MBIR (stochastic group coordinate ascent (SGCA)) and compared it to OSEM and other common reconstruction algorithms (McGaffin and Fessler 2015).

When choosing an algorithm for reconstruction, the best algorithm for an imaging task is dependent on the object being imaged, the background, and the task to be performed.

One of the factors that led to iterative algorithms receiving more attention in research is the harm of X-ray radiation (De González et al. 2009, Domina et al. 2014, Gilbert 2009,

Kuefner 2015). Many groups have explored strategies of lessening the effect of X-ray and gamma ray radiation in humans (Brink and Boice 2012, Caceres et al. 2011, Carsten 2008, Cinkilic et al. 2013, Das et al. 2011, Jelveh et al. 2013, Kalpana et al. 2010, Mohammad et al. 2014, Nishimura et al. 2014, Pei et al. 2014, Prasad 2005, Smith et al. 2017, Stehli et al. 2014, Velauthapillai et al. 2017). The most practical way to keep people safe long term, though, is to simply lower the amount of X-rays going into a person who is getting imaged. With a lower radiation dose, iterative algorithms tend to perform better (Nam et al. 2019).

2.2 MLEM

2.2.1 Implementation of MLEM

The Maximum-Likelihood Expectation Maximization (MLEM) algorithm, which was presented by Richardson (1972) and Lucy (1974), has been used in tomography for many years (Shepp and Vardi 1982).

The formula for MLEM, with a change in variables, is as follows (Barrett and Myers 2004):

$$\theta_n^{k+1} = \frac{\theta_n^k}{S_n} \sum_{m=0}^M \frac{g_m}{(H\theta_n^k)_m} H_{mn} \quad (2.1)$$

Where θ^k is the current reconstruction solution; H_{mn} is the MxN system H-matrix; S_n is a sensitivity function which is an N-dimensional vector; and g_m is the measurement data, which is an M-dimensional vector.

In practical systems, a system matrix, H, is generally not known, or is too big to be useful (Matenine et al. 2018). To use this equation in CT, it is helpful to realize the following:

$$S_n = \sum_m^M H_{mn} \quad (2.2)$$

This shows that S_n is a backprojection of ones. Next:

$$\widetilde{BP} = g_m H_{mn} \quad (2.3)$$

$$\tilde{P} = H\theta^k \quad (2.4)$$

where \widehat{BP} is the backprojection of the data and \tilde{P} is a projection of the current reconstruction solution.

Putting this together, it forms the equation below (Zeng 2010):

$$\theta^{k+1} = \theta^k \frac{\text{Backprojection} \left\{ \frac{\text{Measurement}}{\text{Projection}(\theta^k)} \right\}}{\text{Backprojection}\{1\}} \quad (2.5)$$

2.2.2 Current State of MLEM

MLEM is popular in emission tomography reconstruction, but not as much in x-ray CT without regularizations (Zeng 2010). Still, there have been some uses of its pure form as well as applications for which it has been shown to be especially useful. MLEM has been popular with low dose and incomplete data sets, but these have mainly used alternative forms of the MLEM method, such as Penalized ML (PML) and a particular type of PML called MLEM-TV. Other methods use modified MLEM techniques for applications such as motion correction (Fotouhi et al. 2017). Other applications include Wang et al. (2016), who describe an ML algorithm that is used for the purpose of spectral CT.

There have been large amounts of research performed using MLEM with PET and SPECT image reconstruction and evaluating its noise properties (Liew et al 1993, Qi 2003, Li 2011). Much of this research has been done for low dose images with the hope of getting better image quality (Chávez-Rivera et al. 2015). Some research has combined CT and SPECT to obtain more data for a imaging task (Damle et al. 2011).

In addition to image quality, there has been interest in finding accelerated methods of MLEM. Because of long convergence times, different algorithms have been developed and tested (Van Slambrouck and Nuyts 2014, De Pierro 1995). Some of these, such as ordered subset expectation maximization (OSEM) and weighted least squares with conjugate gradient (WLS-CG), have a trade off with time and image quality (Tsui et al. 1991). Other research has gone into finding algorithms that minimize some of the bias that is present in MLEM (Van Slambrouck et al. 2014).

2.2.3 Penalized Maximum Likelihood (PML)

PML is a subclass of maximum likelihood which uses the equation (Fessler 2000):

$$\Phi(\mu) = L(\mu) - \beta R(\mu) \quad (2.6)$$

Where Φ is a penalized objective function, L is the log-likelihood, R is a regularization term (also called a penalty function), and μ is the vector of voxel attenuation values (Makeev et al. 2016). The penalty function, R , must be chosen carefully. For example, it can be separable to keep the algorithm's complexity mostly the same. Also, the penalty function should be chosen such that it does not take away valuable information from the image just to see a desired result.

Most implementations of MLEM have an addition of a penalty function, making them PML algorithms. In other applications, PML algorithms can be used by joint CT/PET scanners to get a joint image reconstruction and attenuation map (Bousse et al. 2015, Rezaei et al 2016). PML algorithms have been reported to outperform OSEM in a CT/PET image reconstruction (Otani et al. 2019). PML algorithms have also given good image quality in breast CT image reconstructions (Makeev et al. 2016). Penalty functions are useful for applying certain constraints, such as sparsity. These algorithms were found to outperform FDK and Katsevich reconstruction image quality for helical CT (Nam et al. 2019). As mentioned before, total variation (TV) regularization for MLEM is a popular modification to the MLEM algorithm and will be discussed in section 2.2.6.

To apply a regularization term to MLEM, the method given in Appendix B is applied (Clarkson 2020), which results in:

$$f_n^{(k+1)} = \left(\frac{f_n^{(k)}}{s_n} \right) \sum_{m=1}^M H_{mn} \left[\frac{g_m}{(Hf^{(k)})_m} \right] - \eta \frac{f_n^{(k)}}{s_n} \frac{\partial}{\partial f_n} R(f^{(k)}) \quad (2.7)$$

Where $f_n^{(k)}$ is the k^{th} reconstruction of the data, s_n is the sensitivity vector, H_{mn} is the H-matrix, g_m is the measured data vector, η is a regularization constant that can be chosen by the user, and R is the regularization function. The first part of this equation is the MLEM algorithm, while the second term in the equation is the regularization term.

2.2.4 Implementation of PML

To implement PML, a proper penalty function must be chosen. This should be representative of the properties of the imaging system and the reconstructed image. For example, in Makeev et al. (2016), the penalty function is implemented as:

$$R(\mu) = \frac{1}{2} \sum_{j=1}^p \sum_{k \in N_j} w_{jk} \varphi(\mu_j - \mu_k) \quad (2.8)$$

Where w_{jk} 's are the weights of each voxel, and φ is an implementation of the total variation norm (TV):

$$\begin{aligned} \varphi_{TV}(\mu_{i,j,k}) = & |\mu_{i,j,k} - \mu_{i-1,j,k}| + |\mu_{i,j,k} - \mu_{i+1,j,k}| + |\mu_{i,j,k} - \mu_{i,j-1,k}| + \\ & |\mu_{i,j,k} - \mu_{i,j+1,k}| + |\mu_{i,j,k} - \mu_{i,j,k-1}| + |\mu_{i,j,k} - \mu_{i,j,k+1}| \end{aligned} \quad (2.9)$$

There are many other penalty functions (regularization techniques) that target different qualities (Zhang et al. 2018). One group proposes using trainable convolutional neural networks as the penalty function in medical imaging (Wu et al. 2019).

2.2.5 Current State of PML

PML is used in research and has shown better image quality than FBP (Makeev et al. 2016). Certain studies have tried to better predict the qualities of penalty functions in cone-beam CT (CBCT) systems, such as spatial resolution and noise (Wang et al. 2019). Currently, PML tends to be used more than pure MLEM because if anything is known about the imaging system or the object being imaged, it can be applied to the penalty function to enhance the image reconstruction.

Because the MLEM algorithm can be implemented on a GPU, the penalty function is much easier to work with if it can also be implemented on a GPU.

2.2.6 MLEM-Total Variation (MLEM-TV)

The total variation (TV) penalty function has been used to reduce noise and blurring (Chávez-Rivera 2015). TV algorithms are used for data that have a sparsity constraint (Sidky and Pan 2008). It can allow edge preservation (Chávez-Rivera 2015).

The TV technique works in more than just MLEM, with Sun and Hayakawa (2018) applying TV regularization to the algebraic reconstruction technique (ART) and it gave better image quality than ART alone. Sánchez et al. (2015) explores the noise properties of TV regularization for any iterative reconstruction algorithm.

As shown in section 2.2.3, total variation in the format given above is given in Eq. 2.6 (Fessler 2000). In that equation, L is the likelihood, Φ is the function to minimize, β is the regularization factor, and R is given as:

$$TV(\mu) = R(\mu) = \int_{R^n} \nabla \mu \quad (2.10)$$

In this document, the TV regularization function is applied as shown in Panin et al. (1999), Zhang et al. (2018), and many others. It is given as follows:

$$R_{3DTV}(F^k) = \sqrt{(F_{j+1,k,l} - F_{jkl})^2 + (F_{j,k+1,l} - F_{jkl})^2 + (F_{j,k,l+1} - F_{jkl})^2} + \varepsilon \quad (2.11)$$

Where ε is a small number that helps with computations. To apply this to MLEM, the derivative must be used in the regularization portion of the algorithm.

$$\begin{aligned} \frac{\partial}{\partial R_{3DTV}(F^k)} = & \frac{F_{j+1,k,l} + F_{j,k+1,l} + F_{j,k,l+1} - 3F_{j,k,l}}{\sqrt{(F_{j+1,k,l} - F_{j,k,l})^2 + (F_{j,k+1,l} - F_{j,k,l})^2 + (F_{j,k,l+1} - F_{j,k,l})^2} + \varepsilon} \\ & + \frac{F_{j,k,l} - F_{j-1,k,l}}{\sqrt{(F_{j,k,l} - F_{j-1,k,l})^2 + (F_{j-1,k,l+1} - F_{j-1,k,l})^2 + (F_{j-1,k+1,l} - F_{j-1,k,l})^2} + \varepsilon} \\ & + \frac{F_{j,k,l} - F_{j,k-1,l}}{\sqrt{(F_{j,k,l} - F_{j,k-1,l})^2 + (F_{j+1,k-1,l} - F_{j,k-1,l})^2 + (F_{j,k-1,l+1} - F_{j,k-1,l})^2} + \varepsilon} \\ & + \frac{F_{j,k,l} - F_{j,k,l-1}}{\sqrt{(F_{j,k,l} - F_{j,k,l-1})^2 + (F_{j+1,k,l-1} - F_{j,k,l-1})^2 + (F_{j,k+1,l-1} - F_{j,k,l-1})^2} + \varepsilon} \end{aligned} \quad (2.12)$$

There are other ways to implement $R_{3D-TV}(F^k)$, such as the method shown in section 2.2.4 of this document. The general form for LP-norm TV regularizations has a similar form:

$$R_{3DTV_{General}}(F^k) = \left(\begin{array}{l} (F_{j+1,k,l} - F_{jkl})^p \\ + (F_{j,k+1,l} - F_{jkl})^p \\ + (F_{j,k,l+1} - F_{jkl})^p \end{array} \right)^{\frac{1}{p}} \quad (2.13)$$

Applying it to MLEM, its derivative is:

$$\begin{aligned} \frac{\partial}{\partial R_{3DTV_{General}}(F^k)} = & \\ & \left(-(F_{j+1,k,l} - F_{j,k,l})^{p-1} - (F_{j,k+1,l} - F_{j,k,l})^{p-1} - (F_{j,k,l+1} - F_{j,k,l})^{p-1} \right) \\ & * \left((F_{j+1,k,l} - F_{j,k,l})^p + (F_{j,k+1,l} - F_{j,k,l})^p + (F_{j,k,l+1} - F_{j,k,l})^p \right)^{\frac{1}{p}-1} \\ & + (F_{j,k,l} - F_{j-1,k,l})^{p-1} \left(\begin{array}{l} (F_{j,k,l} - F_{j-1,k,l})^p \\ + (F_{j-1,k,l+1} - F_{j-1,k,l})^p + (F_{j-1,k+1,l} - F_{j-1,k,l})^p \end{array} \right)^{\frac{1}{p}-1} \\ & + (F_{j,k,l} - F_{j,k-1,l})^{p-1} \left(\begin{array}{l} (F_{j,k,l} - F_{j,k-1,l})^p \\ + (F_{j+1,k-1,l} - F_{j,k-1,l})^p + (F_{j,k-1,l+1} - F_{j,k-1,l})^p \end{array} \right)^{\frac{1}{p}-1} \\ & + (F_{j,k,l} - F_{j,k,l-1})^{p-1} \left(\begin{array}{l} (F_{j,k,l} - F_{j,k,l-1})^p \\ + (F_{j+1,k,l-1} - F_{j,k,l-1})^p + (F_{j,k+1,l-1} - F_{j,k,l-1})^p \end{array} \right)^{\frac{1}{p}-1} \end{aligned} \quad (2.14)$$

2.3 GPU programming and Iterative Techniques

Because of the amount of time and computations that iterative reconstructions require, computation times can be prohibitive. Even analytical reconstruction techniques, computation times can be long (Ni et al. 2006). With GPU programming becoming so powerful, it is now necessary to perform reconstruction techniques mostly on a GPU. In this way, the projections and back-projections required to perform these algorithms can be done fast enough to achieve acceptable wait times (Chen et al. 2018). Many groups continue to implement MLEM and other iterative algorithms on a GPU (Vazquez et al. 2014). Chapter 3 of this document gives strategies and methods for writing these algorithms on a GPU.

General purpose GPU computing is still rapidly evolving and improving, and many languages currently do not support the data structures and strategies that traditional programming offers. Despite these limitations, clever programming practices and

optimizations help to bring the tomographic reconstruction times down to near real-time (Cui et al. 2013). Using only a CPU, the reconstruction times would go up over 100x in some cases.

2.4 Image Quality

2.4.1 Developing Figures of Merit

Figures of merit are important for quantitatively comparing results. When using different tomographic reconstruction techniques, a figure of merit is desirable to objectively state which method is best for a particular application. This section will discuss how figures of merit will be constructed for this document. Much of what is discussed can be obtained from Barrett & Myers (2004) Foundations of Image Science Chapter 13.

2.4.2 Classification Tasks

A classification task separates images into different groups, such as signal present and signal absent (Barrett and Myers 2004). One method for binary classification is to have a set of $N_s/2$ signal-free images, $N_s/2$ signal-present images, and run an observer to see how effective the observer is. Observers are discussed in more detail in section 2.4.4. The signal used for this document is a simulated gravity wave. In a classification task, an observer's output is called a test statistic, which is a value. A collection of test statistics can be used to assess the observer's performance. Based on the mean values of the observer's test statistics and the noise in the test statistics, a signal to noise ratio can be calculated from the observer as shown below (Barrett and Myers (2004), pg. 819):

$$SNR_t = \frac{\langle t \rangle_{Present} - \langle t \rangle_{Absent}}{\sqrt{\frac{\sigma_{Absent}^2}{2} + \frac{\sigma_{Present}^2}{2}}} \quad (2.15)$$

Where $\langle t \rangle_{Present}$ and $\langle t \rangle_{Absent}$ are the average values given by the observer when a signal is present and absent, respectively. $\sigma_{Present}^2$ and σ_{Absent}^2 are the average variance values from the observer when a signal is present and absent, respectively.

Plotting a receiver operating characteristic (ROC) curve is instructive to determining an imaging system's effectiveness in performing a task. The area under the ROC curve (AUC)

and SNR_t are both related and effective figures of merit. SNR_t is chosen as a figure of merit in this case because of its range and ability to not saturate at high values. SNR_t ranges from 0 to infinity while the AUC ranges only from .5 to 1. In comparing different reconstruction or processing methods, comparing SNR_t with the different methods can then aid in design decisions.

2.4.3 Estimation Tasks

For estimation tasks, such as finding the amplitude of a wave, there are a few options to consider. For a tomographic reconstruction, it may be insightful to find the bias or the variance of the pixels values in the reconstructions. The bias and the variance can be combined into one number, the mean-square error, as shown below:

$$MSE(\theta) = \langle |\hat{\theta} - \theta|^2 \rangle_{g|\theta} \quad (2.16)$$

where θ is the true value, $\hat{\theta}$ is the estimated value, and g is the image vector. MSE is commonly used and is convenient. In practice, MSE can be a flawed metric for images if used exclusively, as shown in Figure 2.1, which shows six different images, all of which have the same MSE.

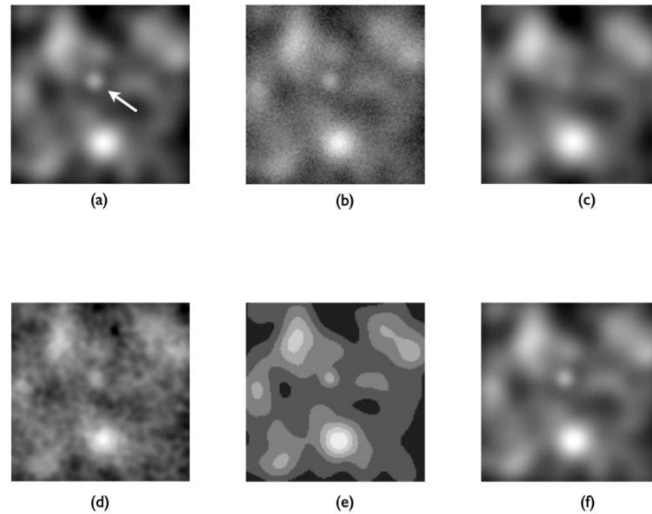


Figure 2.1 : Each image in this progression has the same mean-square error value. (a) Original image with the signal marked by the arrow (b) White noise added (c) Blurred (d) Structured noise added (e) Quantized (f) Shifted. Image credit: Matthew Kupinski

For other estimation tasks, such as finding an amplitude or phase of a gravity wave, to get a single value that can be used as a figure of merit for estimation tasks, the ensemble mean-square error (EMSE) can be used. It is useful if θ is a random vector. Its formula is given below:

$$EMSE = \langle \langle |\hat{\theta} - \theta|^2 \rangle_{g|\theta} \rangle_{\theta} \quad (2.17)$$

For an image, or set of images to be evaluated, EMSE gives a single value, which is convenient for a figure of merit. EMSE will have similar weaknesses to MSE. One weakness of EMSE is that it gives no information as to where incorrect values are located. If an imaging system is accurate at the center of the image, but inaccurate at the edges, the EMSE will not detect this tendency. Because tomographic reconstruction techniques can be scaled to fit data, an estimation task will not be heavily relied on as an imaging task.

2.4.4 Developing Observers for Detection Tasks

2.4.4.1 Hotelling Observers

A Hotelling observer is a practical and effective candidate for many imaging applications (Barrett et al. 1993). It is practical because it only requires first and second order statistical data from an imaging system. It is effective because in gaussian noise, it is the ideal linear observer. It takes the form

$$t = s^t K_g^{-1} g \quad (2.18)$$

Where s the signal to detect, g is the image vector, and K_g^{-1} is the inverse of the covariance matrix of the image vector. To apply this to an ROC curve, if the test statistic, t , is higher than a chosen threshold, it is labeled as a signal-present image, otherwise, it is a signal-absent image.

Some research has gone further and looked into Channelized Hotelling Observers (CHO) for assessing image quality because it is more representative of the human visual system (Fessler and Yendiki 2002, Yao and Barrett 1992). They can also reduce the amount of data that is needed to produce an adequate image quality measurement (Tseng, Fan, and Kupinski 2016).

2.4.4.1.1 Calculating and storing a covariance matrix and its inverse

Calculating a covariance matrix for a Hotelling observer is resource intense. For a simpler image that is 256x256 pixels, the covariance matrix is a 65536x65536 matrix. For a covariance matrix with a much larger image plane or a full 3D reconstruction, inverting it becomes a time-consuming task.

To address this problem, matrix-inversion lemma, found in Barrett and Myers (2004) Chapter 14.3.2 is useful to invert the covariance matrix for use in the Hotelling observer. It gives the covariance matrix, K_g , as:

$$K_g = \bar{K}_n + \hat{K}_{\bar{g}} = \bar{K}_n + WW^t \quad (2.19)$$

Where \bar{K}_n is the noise covariance matrix and is diagonal, $\hat{K}_{\bar{g}}$ is the covariance matrix of the noise-free background, and W is a set of sample background image vectors.

$$W = \frac{1}{\sqrt{N_s}} [\delta g_1, \delta g_2, \dots, \delta g_{N_s}] \quad (2.20)$$

Using matrix-inversion lemma:

$$[\bar{K}_n + WW^t]^{-1} = \bar{K}_n^{-1} - \bar{K}_n^{-1}W[I + W^t\bar{K}_n^{-1}W]^{-1}W^t\bar{K}_n^{-1} \quad (2.21)$$

By using matrix-inversion lemma, an $N_s \times N_s$ matrix can be inverted instead of the entire $M \times M$ covariance matrix. Using as few as 300 noise free images, the calculated covariance matrix inverse is effective in the Hotelling observer.

2.4.4.1.2 Signal-Known-Exactly, Signal-Known-Statistically, Fourier Methods

For a Hotelling observer to be as effective as possible, the signal vector, s , needs to be known exactly (SKE). This means size, location, and possibly phase need to be known. If s is only known statistically (SKS) and its location is unknown, the Hotelling observer loses its effectiveness. This is readily seen when s is a sine wave. If the location and phase are not known exactly, the test statistic will have lower values than an image with no signal.

Because gravity waves are so important for this work and are SKS, a proper implementation, or offshoot, of the Hotelling observer is required. This method will primarily be done in the Fourier domain to simplify the computations.

The method for the modified Fourier Hotelling Observer is as follows:

- 1- Choose a wavelength of interest for a gravity wave (e.g., 100 km).
- 2- Calculate $K_g^{-1}g$: Use matrix inversion lemma with noise-free images. Multiply this with the image data.
- 3- Take the absolute value of the Fast Fourier Transform of $K_g^{-1}g$.
- 4- For the wavelength of interest, make a filter in the Fourier domain that only chooses this wavelength within an acceptable range. This will be a ring around the origin that corresponds with the desired frequency. The ring will have values of one, every other value will be zero.
- 5- Multiply the filter in step 4 by the Fourier Transform found in step 3.
- 6- Take the sum of the resulting image to get the test statistic.

Looking at this method in a Hotelling point of view, the observer looks as follows:

$$t_{fourier} = FT(s_{filter}^t) | FT(K_g^{-1}g) | \quad (2.22)$$

Where $FT()$ is the Fourier transform.

CHAPTER 3

IMPLEMENTATION OF MLEM ON A GPU

3.1 Implementation Overview

As shown in section 2.2, a readable formula for MLEM is given by:

$$\theta^{k+1} = \theta^k \frac{\text{Backprojection} \left\{ \frac{\text{Measurement}}{\text{Projection}(\theta^k)} \right\}}{\text{Backprojection}\{1\}} \quad (3.1)$$

Where θ^0 is usually set as a vector of ones. The measurement data will be a set number of projection images.

The MLEM algorithm will perform best when run primarily on a GPU. When performing an algorithm on a GPU, after each iteration of the algorithm, the resulting θ^{k+1} can, and in some instances should, be returned to host memory. This is the only time in MLEM that a vector needs to be read from GPU to host memory, which is advantageous for speed.

As can be seen from the above equation, back-projections and projections are the building blocks of MLEM. Back-projection and projection implementations need to have a certain measure of accuracy and speed to be effective. The projection especially needs to be accurate. If $\text{Projection}(\theta^k) = \text{Measurement}$, then the equation goes to a $\text{BP}\{1\}/\text{BP}\{1\}$, which provides somewhat of a safeguard for the back-projection. The projection does not have that same safeguard. If it is in error, the algorithm will eventually converge to a wrong solution.

In many algorithms, the projection and back-projection operators assume the source - or telescope in the case of tomography from space - to be a single point. The algorithms assume that an infinitely thin line goes from the source to a desired pixel or voxel. Some algorithms have tried to use a solid angle from the source to a desired pixel or voxel, but the computation time is generally too high (Fu et al. 2018).

3.2 GPU and code discussions

3.2.1 Software implementation and commentary of MLEM

Good coding practices are required to make an adequately fast MLEM implementation. When considering implementation, it is important to consider the basis to be used, namely: delta basis, pixel basis, or a Fourier basis. A Fourier basis may apply for certain reconstruction algorithms where filtering is applied but will not have further consideration in this section. The example code in this paper will be primarily done with both pixels and voxels being in a delta basis. More information on this subject is given in section 3.3.

Note that there are many ways to think about and write projection and back-projection code and the methods given in this paper are just a small sample. Many developers have used GPUs through MATLAB code to perform MLEM and those implementations are available for download. This code will be written in C++. There are many ways to access GPU's for general purpose use (OpenCL, CUDA, Metal, DirectX, etc.), and the code given will use a general pseudocode that can be applied to many of these GPU languages. This code will also use a thread-based GPU structure, instead of using GPU-eligible for-loops that libraries, such as ArrayFire, use.

3.2.2 GPU Programming in Tomography

GPUs have been used for some time to aid in tomographic reconstructions. There is expected growth in the GPU industry in the coming years, primarily due to smart phones (GPU Market Size, Share & Forecast by 2027: Graphics Processing Unit, 2020). The current speed of the best GPUs is in the tens of trillions of floating-point operations per second (TFLOPS). Current trends have GPUs doubling the number of FLOPS every 3-4 years. In the coming years, it is expected to continue to improve. With GPUs continually improving, the trade-off between accuracy and speed in tomographic applications will be minimized. The speed-up in using GPU's instead of CPUs will also continue to increase. In addition to hardware developments, there are also libraries developed that work with the GPUs and give accelerated performance, such as ray tracing libraries in Nvidia and Metal.

3.3 Basis Background for imaging simulations

The basis for a projection or back-projection algorithm affects the quality of the result. For this purpose, an overview of these bases will be presented here. This overview will assume that a thin ray, not a solid angle, is coming from the source.

A delta basis is representing a pixel, or voxel, by a single point. If a projection or back-projection is going to be performed in an entirely delta basis, it will look something like the top illustration in Figure 3.1.

A pixel basis is representing a pixel, or voxel, as having certain boundaries. They can be square, cube, spherical, etc. In algorithms, pixels for the pixel basis have three traditional ways of being calculated.

- 1) Each pixel can be subsampled into a grid of delta functions
- 2) Each pixel is a function with the edges being boundaries
- 3) Each pixel can be subsampled in Monte Carlo fashion.

Each method has its strengths.

There are many projection and back-projection algorithms, each with a different way to think about the problem. Each algorithm also aims to find a way around a speed/accuracy tradeoff for these operations. Many algorithms that are currently being researched use method 2 from above and use clever algorithms to reduce the computation speed.

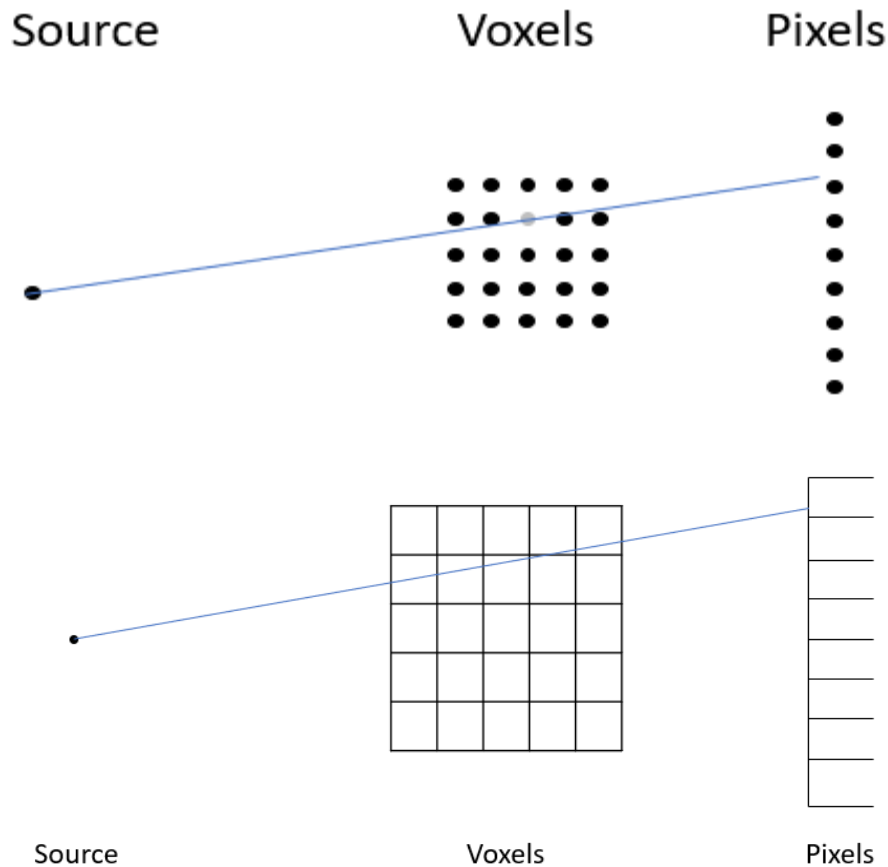


Figure 3.1: Here a point source is projected through the center of a voxel and onto the detector of pixels. Top: Delta basis: Each pixel and voxel are represented by a delta function. Bottom: Pixel basis: Each pixel and voxel are defined by a boundary.

A common algorithm for projections was presented by Siddon (1985). This algorithm uses detector pixels in a delta basis, and voxels in a pixel basis. It is also explained in Thompson and Lionheart (Thompson and Lionheart 2014) and Jacobs (Jacobs et al. 1998). Other well-known algorithms for projections are Joseph's algorithm (Joseph 1983) and Jacob's algorithm (Jacobs et al. 1998), the latter being an extension of Siddon's algorithm. Joseph's algorithm, though older, is still used because it is parallelizable (Dittmann et al. 2017).

3.4 Projection Algorithms (Sphere Voxels, standard delta basis projection)

3.4.1 Common methods for forward and back projections

Recently, there has been interest in Distance Driven (DD) algorithms and Separable Footprint (SF) algorithms, which have gained popularity because they are fast, accurate, and provide matched methods for forward and back-projection methods (Fu et al. 2018).

3.4.1.1 Distance Driven (DD) Approach

As detailed in De Man and Basu (2002, 2004, and 2006) and Fu et al. (2018), the DD algorithm is performed in two steps: 1) projection onto a common axis, and 2) Applying an overlap kernel to calculate how much an image voxel overlaps a detector pixel. To give more detail, the edges of the image pixels (or voxels), are projected onto the x-axis (or any chosen axis). The detector pixel edges are projected onto that same axis. At this point, there will be two arrays stored, one with the locations of a row of pixels projected onto the reference axis, and one with the locations of the detector elements' locations projected on the reference axis. The overlap kernel is then applied. This kernel determines how much of the image voxel is being overlapped onto the detector pixel. A higher number correlates with more overlap. This kernel can become extremely involved. Liu et al. (2017) show that the overlap kernel is parallelizable when DD is branchless, making the DD algorithm much faster. This branchless method is on GitHub and has a CUDA version available.

3.4.1.2 Separable Footprint (SF) Approach

The SF approach takes the shadow of a voxel and projects it onto the detector (Long et al. 2010). The 'function', or shape, of this shadow, is separable in the x and y directions. There are a variety of ways to calculate this function, and it is shown to be accurate (Zheng et al. 2017). Because the function is separable, it makes the computation time reasonable. It is generally more accurate than DD algorithms and can be roughly comparable in speed. Research has been done to make the algorithm work in parallel (Xie et al. 2017).

3.4.2 Delta Basis Projection Algorithm

A realization of a delta basis projection algorithm is detailed in section 4.3.1, and the code shown in Appendix A.2. This algorithm has similarities to the DD algorithm with some

key differences. Because it makes certain assumptions about the voxels, its accuracy will not be ideal. One major setback with this algorithm is that it requires atomic functions, which are functions where the GPU will write serially to a location. The atomic functions help to avoid a race condition but have potential to slow a GPU kernel down. Another issue with atomic functions is that Metal 2.0 GPU language does not support floating-point atomic functions.

3.4.3 Sphere Voxels Projection Algorithm

To avoid doing atomic functions, which slow down GPU algorithms, an alternative algorithm is presented in this section. This new projection algorithm avoids race conditions, which exist when at least two threads try to write to the same location at the same time. The device code for this algorithm has a 3D loop that iterates through all voxels. Each thread iterates through the entire voxel space. Code for the algorithm is given in Appendix A.3. The accuracy of this algorithm will be analyzed in section 3.6.

A simulated ray is produced between the detector's pixel position and the source's position (both assumed to be points). If the ray connected to the detector pixel gets close enough to a particular voxel, that voxel will contribute to the detector pixel's value in the projection. A question for implementing this algorithm is how to account for a ray's value when it gets close to a voxel. If it is within the sphere's radius, should the entire value of the sphere be added to the projection result, or should a value proportional to the length of the ray within the sphere voxel be added. This is also explored in section 3.6.

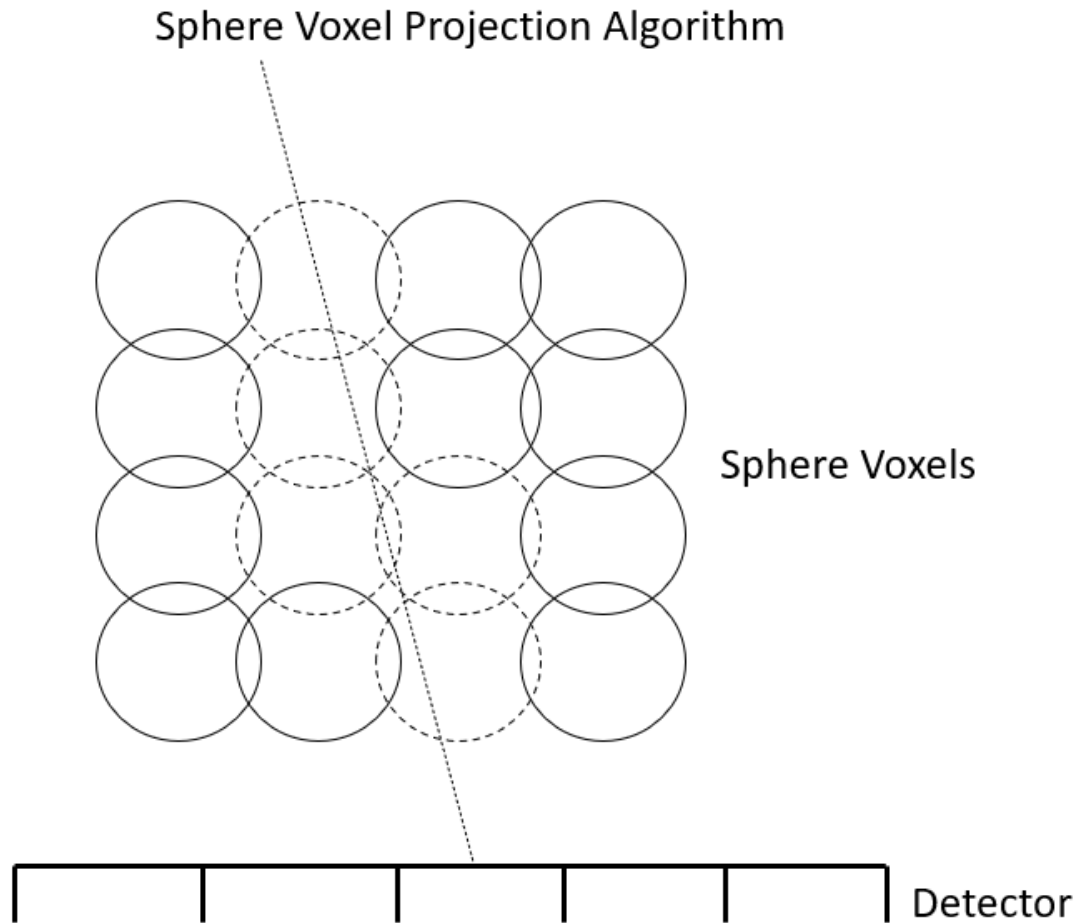


Figure 3.2: Above is an illustration of the sphere voxel projection algorithm. A ray is drawn from the source to a detector pixel. If the ray comes within a voxel's radius, that voxel's value will contribute to the detector pixel's projection value. A decision kernel can be added to determine if the amount that a voxel contributes to a detector is dependent on the how close a ray gets to the center of a voxel. The ray's distance to the center of a voxel is related to the length of a ray that is within a voxel.

3.5 Back-projection Algorithms

A sample back-projection algorithm that is delta based is shown in Appendix A.1. This back-projection algorithm is the same algorithm that is used for the delta-based projection algorithm in section 3.4.2. The difference is that the back-projection algorithm does not require an atomic function and therefore has slightly better performance.

3.6 Testing of Projection Operators

Testing this projection was done with two questions in mind,

- 1) What is the optimum size of the sphere voxels?

and

- 2) Should the projection algorithm weight the voxels' values that contribute to the projection by the length of the rays that are within each sphere voxel, or not? In other words, does the projection algorithm work better with:

$$Detector(\theta_{pixel}) += Voxels(\theta_{voxel}) * lengthOfRayWithinSphere \quad (3.2)$$

or

$$Detector(\theta_{pixel}) += Voxels(\theta_{voxel}) \quad (3.3)$$

Objects that could be computed analytically were needed to analyze the projection's accuracy. To this end, a projection was done with two 3D objects:

- 1) A reference sphere object that has values of one within the sphere and zero outside.
- 2) A reference object that fills the voxel space with all ones.

To determine the optimum size of a sphere voxel, there were 10 different radii tested along with the distance driven algorithm. For this test, the full field of view of the simulated source was 19.29°.

It is worth noting that for a given radius of a sphere voxel, a bigger sphere voxel size can blur out high frequencies. For a small sphere size, some rays will miss the voxels altogether.

Table 3.1: Gives a name, description, and formula for different simulated sizes of sphere voxels.

Sphere Radius Tested	Description of Sphere Radius	Radius Formula
Ideal Shadow Overlap	Varies by distance from Source. Each voxel in the voxel space will have a shadow on the detector.	$R = \frac{\text{Pixel Width}}{2} * \frac{\text{distance}(\text{source to voxel})}{\text{distance}(\text{source to Detector})}$
2X Ideal Shadow Overlap	Twice the 'Ideal Shadow Overlap' value.	$R = \text{Pixel Width} * \frac{\text{distance}(\text{source to voxel})}{\text{distance}(\text{source to Detector})}$
Center Voxel Shadow Overlap	The center of the voxel space has a shadow that is calculated and used as the radius.	$R = \text{Pixel Width} * \frac{\text{distance}(\text{source to center voxel})}{\text{distance}(\text{source to Detector})}$
2x Center Voxel Shadow Overlap	Twice the 'Center Voxel Shadow Overlap' value.	$R = \frac{\text{Pixel Width}}{2} * \frac{\text{distance}(\text{source to center voxel})}{\text{distance}(\text{source to Detector})}$
Cube Area	The radius chosen creates a sphere with equal volume to the cubic voxel it is representing.	$R = \left(\frac{3 * (\text{cubic voxel length})^3}{4\pi} \right)^{1/3}$
Sqrt(2)/2*cube voxel length:	The distance from the center of an edge of the cubic voxel to the center of the voxel.	$R = \frac{\text{sqrt}(2) * \text{cubic voxel length}}{2}$
No Overlap	The sphere voxels touch each other but do not overlap with each other.	$R = \frac{\text{cubic voxel length}}{2}$
Sqrt(3)/2*cube voxel length:	The distance from one corner of the cubic voxel to the center of the voxel.	$R = \frac{\text{sqrt}(3) * \text{cubic voxel length}}{2}$
Cube Voxel Length Overlap:	The length of a cubic voxel.	$R = \text{cubic voxel length}$
2x Cube Voxel Length Overlap:	Twice the length of a cubic voxel.	$R = 2 * \text{cubic voxel length}$

3.6.1 Images and Commentary of Overall Projections

Below are images of the error of a projected reference voxel space of all ones, and a projected reference sphere. The images below are for the first projection, which corresponds to 0° . The images are not shaded relative to each other and are only meant to give a general, qualitative idea of what is happening with each voxel radius. The error was calculated by the following:

$$error[i] = \text{sqrt} \left(\left(\frac{projectionData[i]}{\text{mean}(projectionData)} - \frac{Reference[i]}{\text{bestFitValue}} \right)^2 \right) \quad (3.4)$$

For each of these images, brighter values correspond to higher errors, and darker pixels correspond to lower errors.

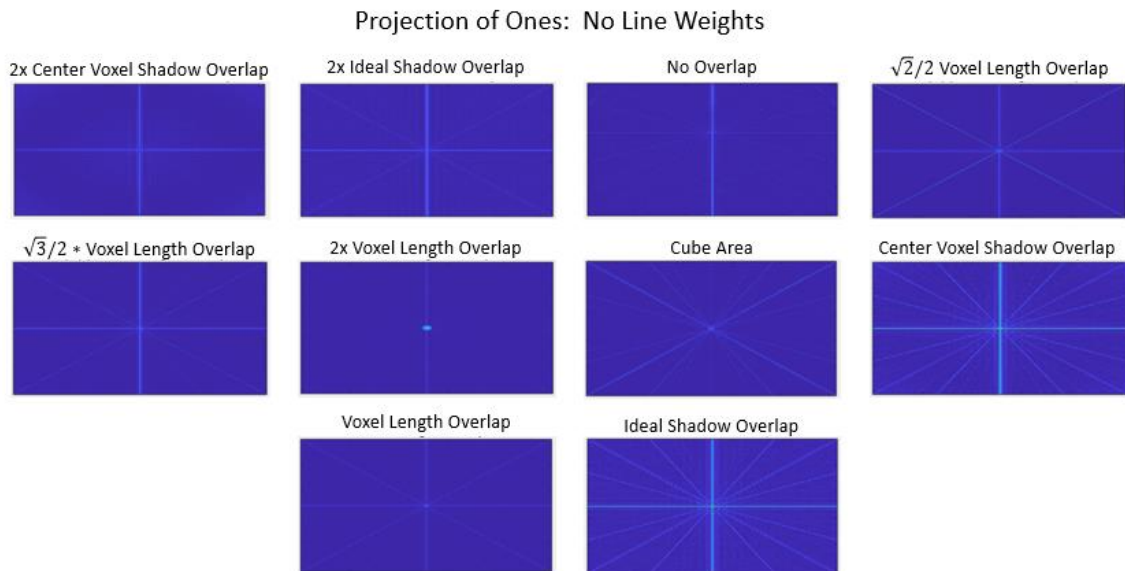


Figure 3.3: Projections of a voxel space made up of all ones onto a 512×512 grid. These images were taken at a 0° projection angle. In each image, there is a different radius for the sphere voxels being used. This implementation of the algorithm was written so that if a source-to-detector ray propagated within the sphere voxel radius, the whole weight of the ray was counted. It can be seen qualitatively that when there is potential for double counting a voxel, such as the center pixels, there will be errors.

Projection of Ones: With Line Weights

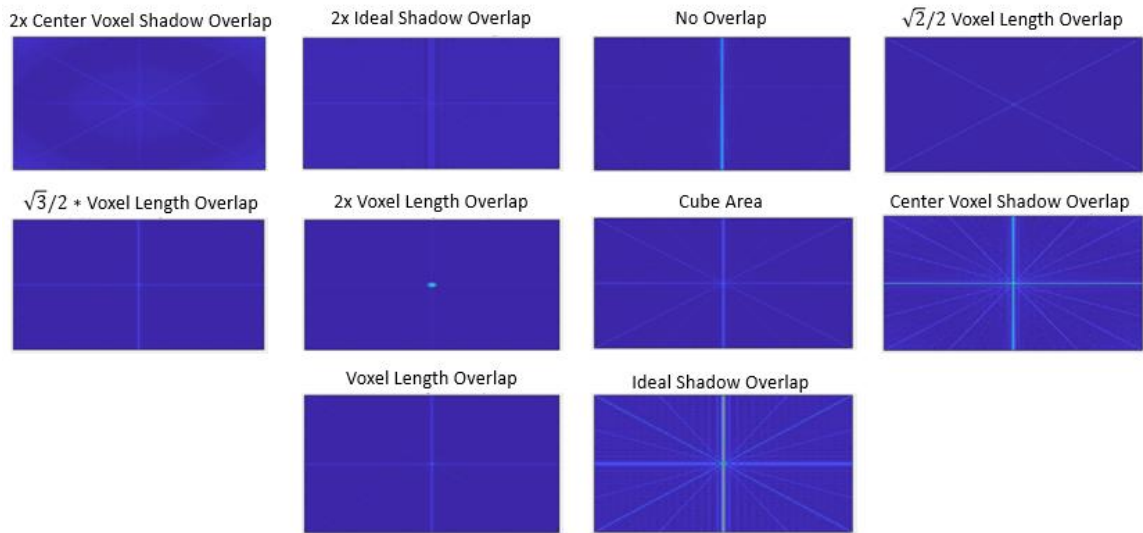


Figure 3.4: Projections of a voxel space made up of all ones onto a 512 x 512 grid. These images were taken at a 0° projection angle. In each image, there is a different radius for the sphere voxels being used. If a ray entered within a voxel's radius, the whole weight of the voxel was projected to the detector pixel that was pointed to by the ray. The result is similar to figure 3.3 in that there are errors when there is potential for double counting voxels.

Projection of Ones: Distance Driven

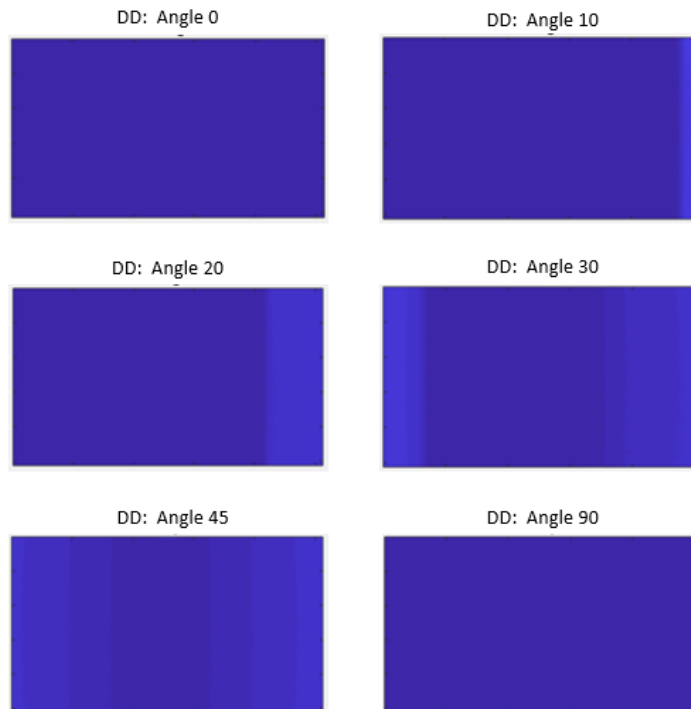


Figure 3.5: Projections taken from the distance driven algorithm at different angles onto a 512 x 512 grid. Although strictly qualitative, it is obvious that at 45 degrees, the projection has bigger errors.

Projection of Sphere: No Line Weights

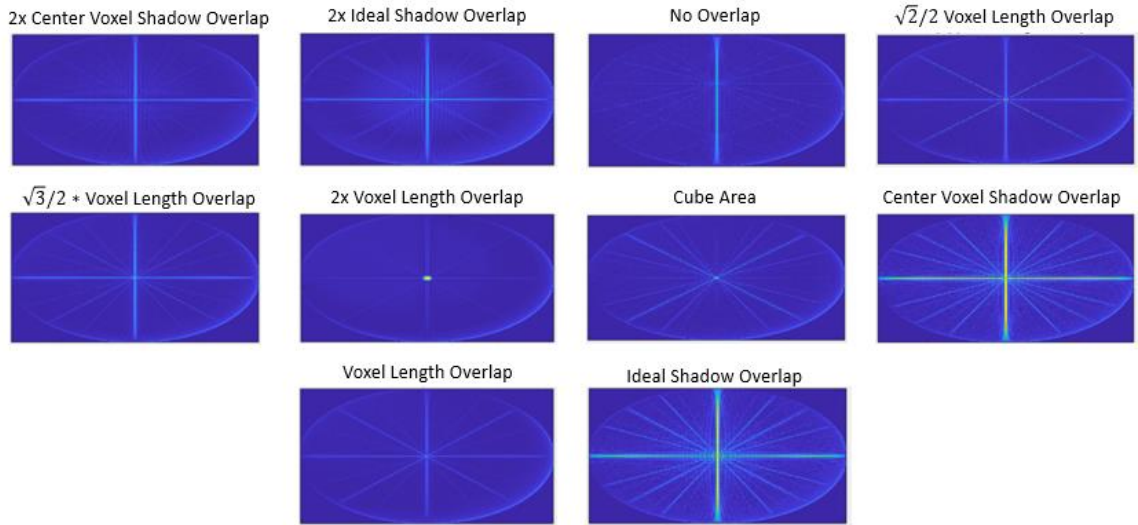


Figure 3.6: Projections of a voxel space made up of a reference sphere onto a 512 x 512 grid. These images were taken at a 0° projection angle. In each image, there is a different radius for the sphere voxels being used. Line weights are not used in these projections.

Projection of Sphere: With Line Weights

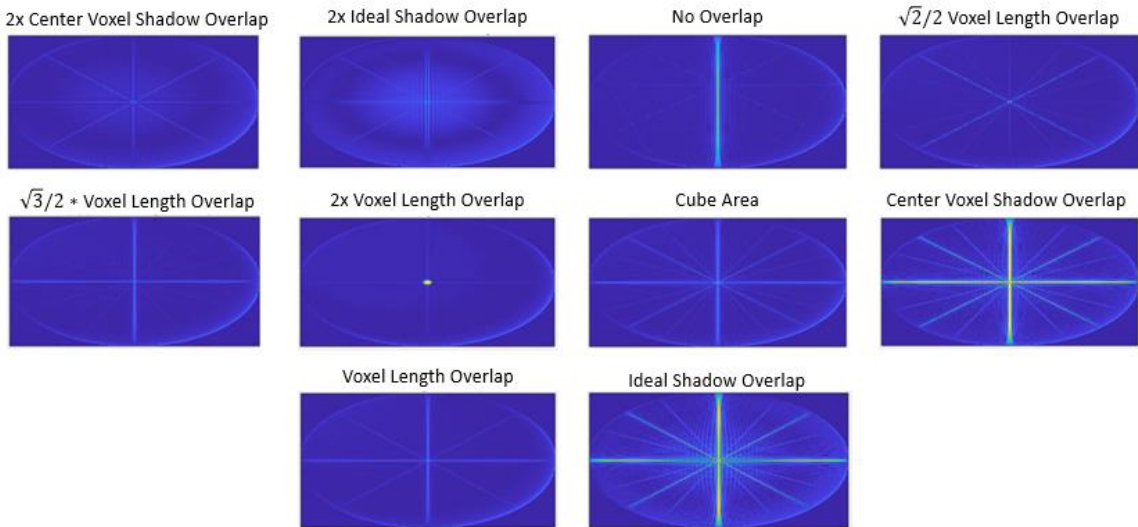


Figure 3.7: Projections of a voxel space made up of a reference sphere onto a 512 x 512 grid. These images were taken at a 0° projection angle. In each image, there is a different radius for the sphere voxels being used. Line weights are used in this projection.

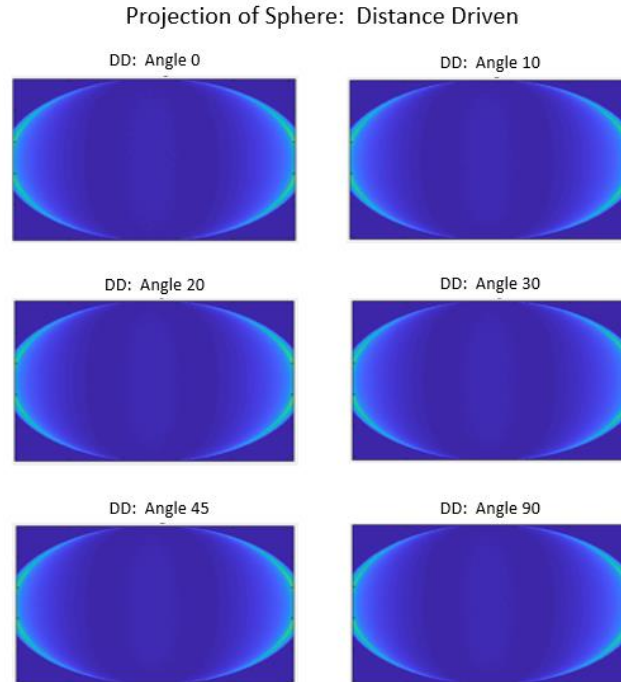


Figure 3.8: The distance driven algorithm used for projecting a reference sphere onto a 512 x 512 grid. The edges are especially poor in accuracy.

3.6.2 Discussion of Projection Error Images

From the previous figures, some conclusions can be drawn about the sphere voxels projection algorithm:

- 1) The areas with the most error are those where the rays have the biggest chance of being double counted.
- 2) When line weights are added, there is higher potential for fringe patterns to occur if the voxel radius size is too small.
- 3) The error seems to be somewhat constant with respect to distance from the center pixel

The figures also show that the distance driven algorithm gets worse with distance from the center voxel.

3.6.3 Preliminary Decisions with Sphere Voxel Radius'

After running preliminary tests for accuracy, it was clear that the sphere voxel sizes that perform best are the no overlap, equivalent cube area, $\frac{\sqrt{2}}{2}$, $\frac{\sqrt{3}}{2}$, and the voxel length radius'.

Sphere voxel radius' that are bigger produced lower error, to a point, but the cost in resolution was not worth pursuing. When a sphere voxel radius became too small, other errors would show up that made it undesirable to explore further. The best voxel radius sizes provided a range where a trade-off occurs with lower error and good resolution.

3.6.4 Overall Comparison to Distance Driven Projector

When comparing accuracy of the sphere voxels and the distance driven algorithm, they behave differently. The accuracy of the distance driven projection tends to get off when the shadow of the voxels onto the detector is not a perfect rectangle. Accuracy of both projection algorithms seems to be similar, but the best sphere radius' performances are much better than distance driven as shown in Figure 3.9 and Figure 3.10.

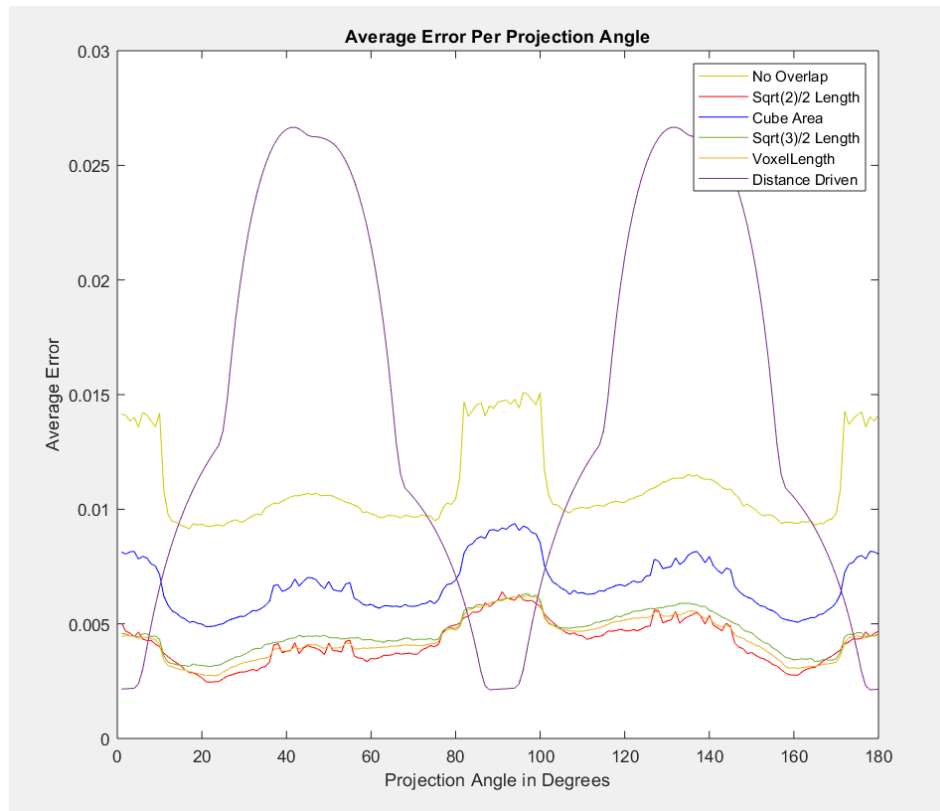


Figure 3.9: This plot shows the relative average projection error on the y-axis. On the x-axis are different projection angles that were tested. There were 180 projection angles in all, going from 0-180 degrees. The voxel space that was projected was a 512x512x512 grid of all ones. The detector was a 512x512 pixel detector. The full field of view was 19.29°. The distance driven algorithm struggled when the angles were higher than about 20 degrees relative to normal. The sphere voxel algorithm showed decent consistency throughout the projection angles. The jumps in error for the sphere voxels are when there are the highest chances for double counting a voxel's value in a projection. The best sphere voxel radii far exceed the performance of the distance driven algorithm, except in cases of almost 0-degree projection angles. The sphere voxel algorithm was implemented with weighting the amount of a ray that is within a voxel.

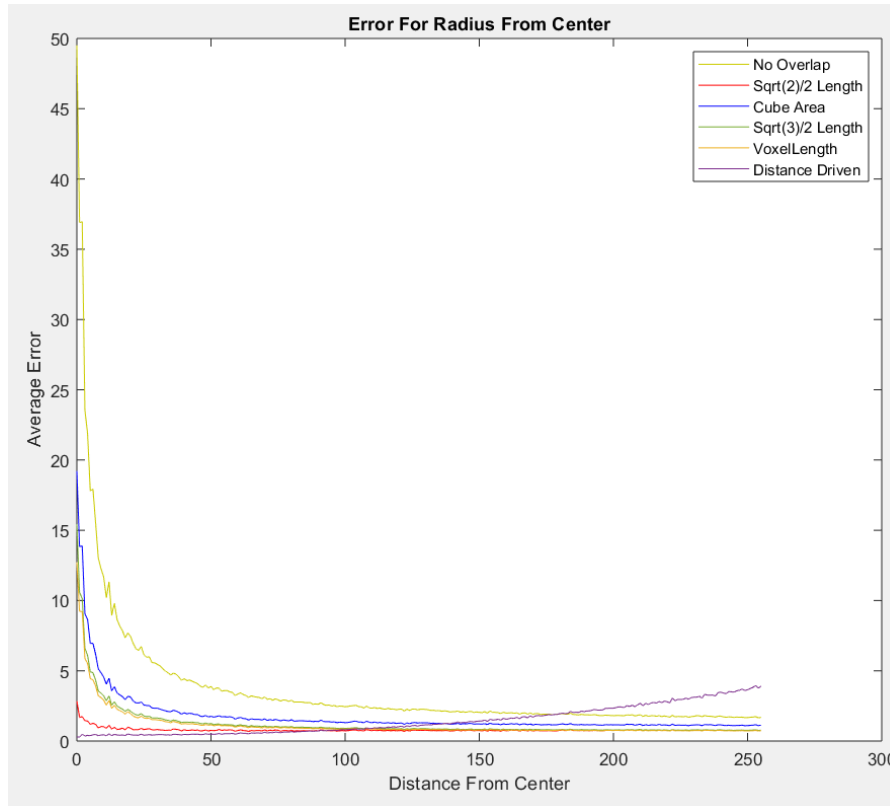


Figure 3.10: The plot above shows the relative average projection error on the y-axis. The x-axis is the distance that a pixel is from the center of the detector. The voxel space that was projected was a 512x512x512 grid of all ones. The detector was a 512x512 pixel detector. The full field of view was 19.29°. The distance driven approach gives little error for the pixels near the center of the detector, but as the pixels get further from the detector, the error increases. The sphere voxel projections tend to have poor performance near the center pixel, due to double counting voxels. As the distance from the center increases, the error becomes much smaller. The sphere voxel algorithm was implemented with weighting the amount of a ray that is within a voxel.

With accuracy favoring the sphere voxel approach, the speed of the algorithm is also informative. Currently the distance driven method is much faster than the sphere voxel method by about 10x. While assumptions can be made to speed up the sphere voxel algorithm, the overall speed will still likely favor the distance driven method. Depending on the application, the accuracy of the distance driven method could be acceptable given the right conditions, tasks, and computer hardware being used.

More can be done to explore faster methods for the sphere voxel algorithm. Currently, it is launched on a GPU, with each thread corresponding to a detector pixel. In each kernel, the voxel space is iterated through, using a nested while-loops. Much has been done to exclude unneeded voxels from being iterated through, but it is still a great time constraint to iterate through so many voxels. GPUs are not currently optimized to use loops within

the kernel, and therefore the performance hit is largely due to the presence of the nested while-loops.

The delta basis projection algorithm described in Section 3.4.2 uses a GPU differently. Each thread corresponds to a voxel in the voxel space. It then determines which detector pixel the voxel will come closest to, given the source-to-voxel ray direction. When a full set of voxels is done computing, the next detector position is assumed, and the algorithm is run again. This algorithm has much better speed and if the sphere voxel algorithm incorporated this strategy, it may not be quite as fast due to the extra calculations required, but it could still be much faster than the current while-loop method.

CHAPTER 4

TOMOGRAPHY FROM SPACE: TECHNIQUES AND LIMITATIONS

4.1 Introduction to Tomography from Space

Imaging objects on the Earth from a satellite above the Earth poses some challenges. Design of the instrument is only part of the difficulty. The atmosphere limits and blocks certain wavelengths. Clouds can block objects on the ground or reflect onto objects of interest above them. If an object is above the ground, the ground will have a certain reflectance depending on if it is soil, water, ice, snow, vegetation, etc. If the object of interest is on the ground and an imaging satellite has an altitude higher than ~85 km, the OH-airglow layer emits in certain wavelengths that need to be accounted for. These complications make it difficult to get effective quantitative results for regions or objects of interest.

If an imaging satellite is moving across the region of interest, a tomographic reconstruction can be possible that can separate out the unwanted signal layers and provide more accurate information about a desired object.

MLEM has been discussed in Chapter 2 and Chapter 3 and has been shown to be effective in low dose (Chávez-Rivera et al. 2015) and limited projection angle reconstructions (Elbakri and Fessler 2002, Dempster et al. 1977). These two limitations are common in space imaging, and therefore MLEM is a good candidate for tomographic reconstructions in images from a space platform.

4.2 Limited-Angle Tomography from Space

In tomographic reconstruction applications, complete reconstruction data from many angles is desired. Tuy's condition (Tuy 1983) plays an important role in design considerations. However, in space, many satellites look nadir and do not focus on a region of interest, which severely limits the altitude-axis resolution of tomographic

reconstructions. For nadir imaging, there are trade-offs between reconstruction speed, field of view, altitude-axis resolution (z-axis), and XY-axis resolution. Some satellites will have large fields of view, such as the Atmospheric Waves Experiment (AWE), which has a roughly 90° FFOV. When the field of view is large, the XY-axis resolution will decrease due to each detector pixel capturing a larger area. To increase its resolution, a detector with more pixels could be used, but that would make the reconstructions go proportionally slower.

With altitude-axis resolution decreased, there are certain processing and reconstruction strategies that can be used to help with the limited-angle scenarios. None of these strategies will be ideal for every situation. Knowing when to use these strategies can serve as a toolbox for different situations that are encountered when imaging in a limited projection-angle environment.

4.2.1 Increase the Number of Iterations

As outlined in Alenius, Ruotsalainen, and Astola (1998), when there are many projections and higher frequencies to reconstruct, more iterations will be required. They also explain how using only a few iterations of the MLEM algorithm is basically using a low-pass filter for the reconstruction. The space applications of tomography will get better image quality with more iterations, but the computational cost may be prohibitive.

4.2.2 Adding More Detector Pixels

One way to increase the resolution of a reconstruction is to increase the resolution of the detector. If a design does not allow for changing detectors to get better reconstruction resolution, sometimes interpolation can be performed to get an approximation for smaller pixels. Many times, an imager will not be able to circle a region of interest but will just be flying by in a straight line. In this case, the projection operation can give intense artifacts as part of the reconstruction, which greatly diminishes image quality if there are too many pixels. While adding more pixels will not dramatically increase the altitude-axis resolution, it does allow for a finer reconstruction which can give an observer more information.

4.2.3 Initializing Voxel Space in Altitude-Axis

When prior knowledge can be obtained about the voxel space, it can improve the reconstruction. Hart II (2012) shows its usefulness in limited-angle tomography and how initializing voxels at a chosen altitude range can help place signals at that altitude. By knowing the altitude of the signals, the signals can be better placed and more accurate in the reconstructions.

It is worth noting that initializing voxels will not necessarily improve altitude-axis resolution. It just controls the resolution problem by placing the signals at a correct altitude or XY location. Initializing voxels will not help to resolve two close objects separated by a small altitude.

4.2.4 Total Variation (TV) Regularization

Using TV regularization is common in medical CT, which is explained in section 2.2.6. It is also reportedly an effective method for getting better reconstructions in noisy and limited-angle applications (Islam 2013, Yan et al. 2014). There is a large body of research that covers the strengths of TV regularization. Some research has looked at $0 < L < 1$ norms for TV regularization to get better reconstruction quality (Zhang et al. 2018).

Much of the research has not dealt with severely limiting projection angles. For an instrument like AWE, which has a 90° full field of view, projection angles are very restricting, even for TV regularization.

4.2.5 Adding More Projections

While adding more projections - or in this case images - and angles is the best way to improve resolution, it is not always possible in space applications because there are only so many quality images that can be obtained as a satellite flies by a region of interest. Generally, a satellite does not fly around the region of interest, but over it, which limits the usefulness of excessive images. Obtaining images of a moving target will pose yet another limitation on obtaining more useful data. If an object is moving quickly enough, it can potentially blur the reconstruction. Another limitation for adding more images is longer reconstruction times. The number of images play a large role in reconstruction times, especially if it is desirable to reconstruct a collection of images in a reasonable timeframe.

4.2.6 Different Iterative Techniques

While MLEM has been shown to be effective for limited angle, low SNR situations, other algorithms are also effective and have been attempted (Hart 2012). PCART and the Landweber reconstruction techniques were used for comparisons. There are many other techniques that were not explored for this section.

These two algorithms did not obtain better results with limited iterations, but they are important to consider for certain applications because they are additive algorithms and not multiplicative. This means that negative numbers are allowed, and that initialization of voxels and regularization techniques can be performed with different strategies. With MLEM, when a voxel is set to zero, it stays at zero because MLEM is multiplicative. That is not the case for PCART and Landweber.

The formula used for PCART (Hart 2012) is:

$$f_{m,n} = f_{m-1,n} + \lambda \frac{(p_m - p_m^i)w_{m,n}}{N_m} \quad (4.1)$$

Where f is the reconstructed volume, p_m is the measured data, p_m^i is the projected data of the i^{th} iteration, N_m is a sensitivity matrix, and $w_{m,n}$ is the system matrix. This is almost identical to MLEM except for substituting an addition and a subtraction for a multiplication and a division.

The formula used for the Landweber algorithm is:

$$\hat{f}^{k+1} = (1 - \eta)\vec{f}^k + H^\dagger(\vec{g} - H\vec{f}^k) \quad (4.2)$$

Where f is the reconstructed volume, H^\dagger is a back-projection operator, \vec{g} is the measured data, $H\vec{f}^k$ is a projection of the k^{th} guess of the volume, and η is a small number that can be tuned to give the best results. If η is tuned correctly, this is almost identical to PCART, except that the right side of the operation is not divided by the sensitivity matrix.

If a reconstruction application does not have intense time constraints, these additive algorithms can give superior reconstructions, but it can take many more iterations to converge to an adequate solution (Luo et al. 2018).

4.2.7 Point Spread Function

Finding a point spread function (PSF), or an average point spread function, for a reconstruction would appear to be effective for giving better resolution. Unfortunately, not all situations will give a stable result. If the application allows, though, it can be an effective way to get some resolution back (Feng et al. 2018).

A PSF is meant for linear, spatially invariant systems. A reconstruction, especially in limited-angle circumstances, will generally not be spatially invariant. Each voxel will have its own unique point response function. This makes using a PSF computationally intense. If the whole reconstruction is not being used and there is a much smaller region of interest, a local PSF could be calculated and applied for that region. A PSF needs to be applied carefully because it can negatively affect resolution in all axes if it is applied carelessly.

4.2.8 Focusing on a Region of Interest

If the situation allows, having a satellite tilt to focus on a region of interest as it is moving past is an effective way to get better altitude-axis resolution. This is intuitive because the detector pixels in the center of the image for optical instruments will generally be higher quality data than those on the edges. Focusing on a region of interest is also effective because it is more projections with the region of interest in the field of view, with potentially more projection angles. This is an intuitive improvement to resolution for imagers with a limited field of view. With a large field of view, the benefit of focusing on a region is diminished because there is less information that is stored for that region. It can still have benefits, though, as is shown in Table 4.1.

Table 4.1: Summary of Methods to Improve Altitude-Axis Resolution

Summary of Methods to Improve Altitude-Axis Resolution		
Method	Optimal Conditions for Use	Reconstruction Time Cost
Increase Iterations	When computation times allow	Moderate to Severe
Increase Detector Pixels	When computation times allow	Minimal to Moderate
Initializing Voxel Space in Altitude-Axis	Known signal locations, XY-axis visibility is less important.	No effect
TV Regularization	Sparsity constraint	Minimal
Adding More Projections	Stationary, or slow-moving, object	Moderate to Severe
Different Iterative Techniques (Besides MLEM)	No positivity constraint	Moderate to Severe
Point Spread Function	Stable PSF, or local region of interest	Moderate: varies by method
Focusing on a Region of Interest	When circumstances allow, narrow FOV	No effect

4.3 Implementation of Tomographic Reconstructions from Space

4.3.1 Projection Algorithm

Implementing the MLEM algorithm is highly dependent on the projection and back-projection algorithms. Each iteration requires one projection and at least one back-projection. They must have reasonable accuracy for reliable solutions and be quick enough for practicality. An algorithm was developed, similar to the distance driven algorithm discussed in Basu and De Man (2006).

For a back-projection, the image scene pixels and each voxel were taken to be points in space. The imager point has rays connecting it to each voxel point and extending to the detector plane. The detector plane is projected to the voxel being evaluated for the back-

projection. A decision kernel is then applied which splits the weight of the four closest detector pixels and adds those values to the voxel being evaluated.

For projections, this same algorithm was used, but the decision kernel had each of the four detector pixels being added to instead of the voxel. This required an atomic addition to be used in the GPU. While atomic operations tend to slow down GPU kernels, the projection time is still quick. Pseudo-code is given in Figure 4.1.

```

1 //GPU block and thread indices
2 threadVal = short3(blockIdx.x, blockIdx.y, threadIdx.x)
3 bufferIdx = threadVal.x + numberOfVoxelsInX * threadVal.y + numberOfVoxelsInX * numberOfVoxelsInY * threadVal.z
4
5 //Check to see if the ray from source to detector is going to hit the right side of the detector
6 check = dot(detectorNormalVector, detectorPosition - sourcePosition)/dot(detectorNormalVector, sourceDirection)
7 if (dot(detectorNormalVector, sourceDirection) < 1e-6)
8     return
9 end
10
11 if (check < 0)
12     return
13 end
14
15 //Find intersection point in space
16 intersection = sourcePosition + check * sourceDirection.x
17
18 //Find the intersection point relative to the detector
19 xyPosition.x = dot((intersection - detPosition),float3(detectorNormalVector.z, 0, -detectorNormalVector.x))
20 xyPosition.y = dot((intersection - detPosition),cross(detNormal,xHat))
21
22 //If the intersection point is not on the detector, return
23 if(xyPosition < -detectorWidth || xyPosition > detectorWidth)
24     return
25 end
26
27 //Find the pixel number on the detector where the intersection point is
28 detectorXYPixelPos = (xyPosition + detectorWidth)/(2 * detectorWidth)*numPixInDetectorRow
29 pixel = floor(normXYPos)
30 fractX = detectorXYPixelPos.x - pixel.x
31 fractY = detectorXYPixelPos.y - pixel.y
32
33 //Get the index for the pixel
34 projIndex = pixel.x + pixel.x * numPixInDetectorRow + numImage * numPixInDetectorRow * numPixInDetectorRow
35
36 //If it is a back-projection, add to the volumeBuffer with an adjustmentFactor to account for ray angle.
37 //If it is a projection, use an atomic operation to add the values to the detectors
38 detectorBuffer[projIndex] = atomicAdd((1-fractX)*(1-fractY)*volumeBuffer[bufferIdx])
39 detectorBuffer[projIndex + 1] = atomicAdd((fractX)*(1-fractY)*volumeBuffer[bufferIdx])
40 detectorBuffer[projIndex + numPixInDetectorRow] = atomicAdd((1-fractX)*(fractY)*volumeBuffer[bufferIdx])
41 detectorBuffer[projIndex + 1 + numPixInDetectorRow] = atomicAdd((fractX)*(fractY)*volumeBuffer[bufferIdx])

```

Figure 4.1: Pseudo code for the projection algorithm discussed above. The back-projection algorithm that can be made from this algorithm makes changes after line 25 so that the code will write to volumeBuffer instead of detectorBuffer.

To check for accuracy, a projection was performed on a volume of all ones and compared to an analytical solution. A normal projection algorithm analysis would determine the projection accuracy for many projection angles. The distance driven algorithm, which is most similar to this algorithm, has worse accuracy as the voxels can no longer be assumed as rectangular. This condition is obvious when a projection angle is at a 45° angle in relation to the voxel space. In the case of applications from space, especially nadir imagers, that problem will generally not exist except potentially at the edge of the field of view. As

it is currently being used for a wide FOV application, a test for the error as the angle increases was also performed. The accuracy results are shown in Figure 4.2.

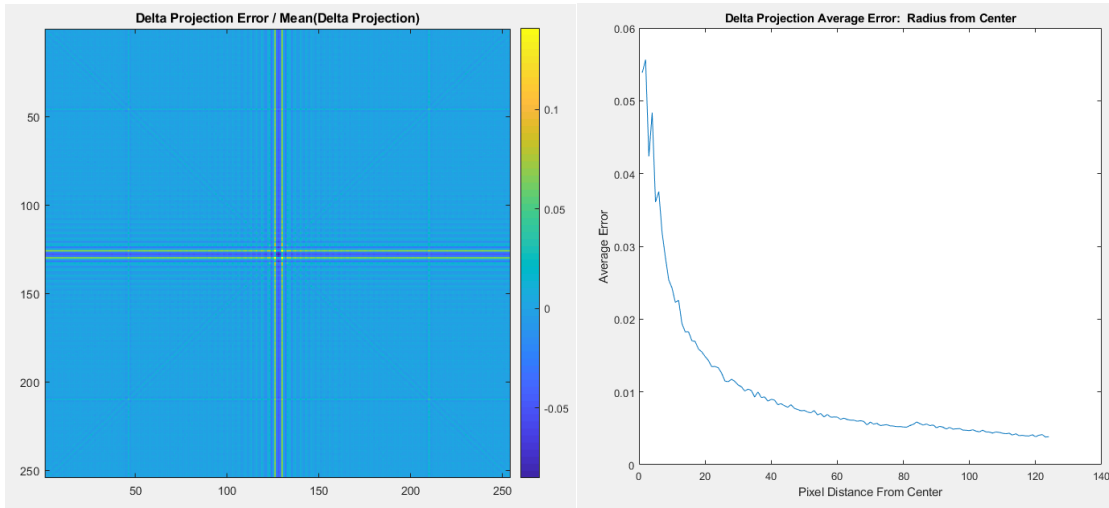


Figure 4.2: The conditions for these images was for a 72-degree full field of view. Left Image: Shows the error of each pixel of the projection operation. The middle pixels of each axis give the most error. Right Image: The average error of the projection image given a pixel’s distance from the center pixel. As the distance from the center increases, the average error decreases.

4.3.2 MLEM Parameters and Speed

An MLEM reconstruction with no regularization factors was run with the following conditions:

Table 4.2: Reconstruction Parameters

Condition	Value
Detector Pixels	256x256
Voxel Space	256x256x64
Iterations	8
Number of Projection Images	80
GPU	NVIDIA Quadro P3200
CPU	Intel Core i5-7200U CPU @2.5 GHz
FFOV	72°

Given the conditions in Table 2, the final reconstruction speed for this MLEM algorithm was 14 seconds. This includes the back-projection taking ~ 0.75 seconds and the projection taking slightly less than one second. At this speed, real-time 3D reconstruction of groups of image data is possible.

AWE, which is explained in detail in chapter 1, has a one second integration time. With a roughly 600 km x 600 km FFOV on the OH-airglow layer (~ 820 km x 820 km on the ground), and an ISS velocity of 7.66 km/sec, at least 78 images will be taken before the imager has passed by a region. A one second integration time with the current MLEM algorithm implementation means that every 14 images, a new 3D reconstruction can be performed, which will overlap significantly with previous 3D reconstructions. It would likely not be desirable to overlap reconstructions so often, which means that for the purposes of AWE, real-time 3D reconstructions can be performed.

4.4 Results and Discussion

4.4.1 Initial Images

Using a simplified model of the atmosphere which included the OH-Airglow layer, basic reflections from clouds, and basic reflections from ground and water, a 3D reconstruction was realized. In this case, it is the OH-airglow layer that is of most interest and getting good altitude-axis resolution in that 20 km region. The other regions are still important enough to include, especially the reflection layer, but they are not stressed as much. Figure 4.3 shows images of the simulated OH-Airglow layer. Figure 4.4 shows an image taken of the airglow layer with a gravity wave present (Eckermann et al. 2016).

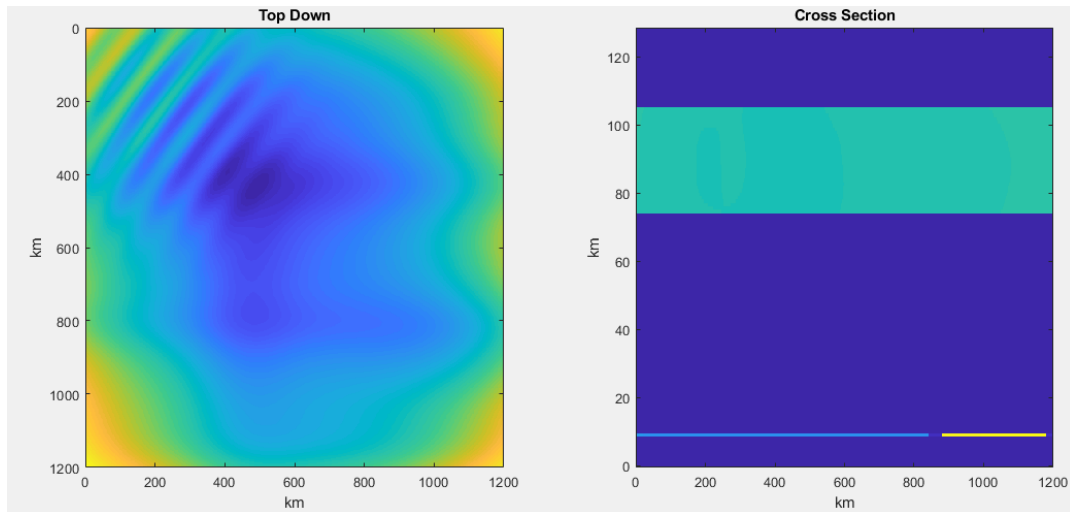


Figure 4.3: Left: A top-down image of the simulated model, which is mainly the OH-airglow layer. In the top left corner is a simulated atmospheric gravity wave. There are other background structures throughout the image. Right: A cross-section of the model. At the bottom of the image is the reflection layer, which represents reflections from the ground and clouds. The band of green, near the top of the image, is the OH-airglow layer.

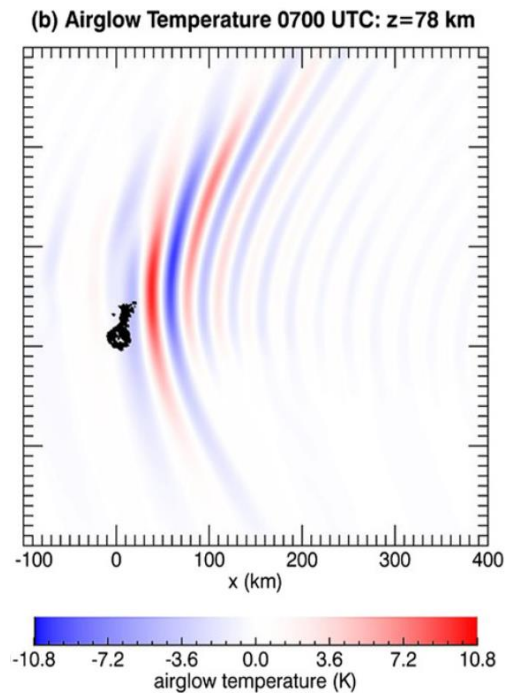


Figure 4.4: Image of the OH-airglow layer taken by Eckermann et al. 2016. Eckermann, Stephen D., et al. "Dynamics of orographic gravity waves observed in the mesosphere over the Auckland Islands during the Deep Propagating Gravity Wave Experiment (DEEPWAVE)." *Journal of the Atmospheric Sciences* 73.10 (2016): 3855-3876. © American Meteorological Society. Used with permission.

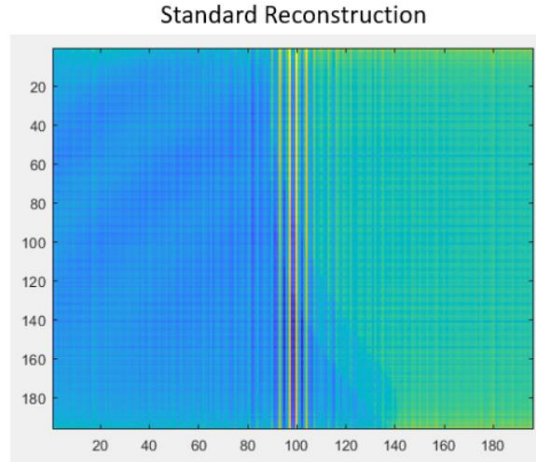


Figure 4.5: A slice of the reconstruction, which corresponds to an 87 km altitude. There are artifacts from the P/BP operations, due to the direction of propagation being only in the y -direction. The right part of the image has a green mass that is a bright reflection from 10 km that is not completely filtered out. The left of the image has a barely noticeable wave pattern that are the faint atmospheric gravity waves.

4.4.2 Altitude-Axis Resolution

The limited-angle constraints of performing tomographic reconstructions from space were discussed in section 4.2 and were seen in these reconstructions. While there was no single method that significantly helped resolution, some methods stood out as effective. The model that was reconstructed was also given the addition of two bright 3D boxes right on top of each other, but vertically spaced by about 20 km. This was done because the OH-airglow layer was difficult to see in the reconstructions, which was due to the restrictions in projection angles. Having these two boxes on top of each other highlighted the altitude-axis resolution constraints that exist.

The MLEM method with eight iterations was indeed effective and one of the better performers. PCART and Landweber algorithms did not perform as well, but this could be due to only eight iterations being performed. Generally, many more iterations are performed for these additive algorithms. Focusing on the center voxel of the reconstruction space, rather than looking nadir, provided some separation in the boxes, but did not provide significant improvement of altitude-axis resolution. TV-regularization did provide some benefit, especially in making the top box more visible. Increasing the number of pixels in the detector did minimal benefit for this case. Changing the FOV changed the resolution of the reconstructions significantly. The 50° FFOV reconstruction barely had any separation, while the 90° FFOV provided a slight increase in the resolution. Combining

TV and increased detector pixels did not significantly change the result from just using TV regularization. The best results came from using 50 iterations of the MLEM algorithm. By doing more iterations, even the OH-airglow layer with the image values only slightly higher, was barely visible, as shown in Figure 4.8 and Figure 4.9.

Another area to be explored is the implementation of the MLEM algorithm. The projection and back-projection operators in this application are designed for speed, with some error permitted. As GPUs continue to provide greater speed, the trade-offs between accuracy and speed begin to go away. In Figure 4.6 and Figure 4.7, there are clear artifacts from reconstruction that can diminish image quality. Algorithms such as the separable footprint (SF) and distance driven (DD) techniques provide matched projection and back-projection techniques that could keep the speed high and reduce the overall error.

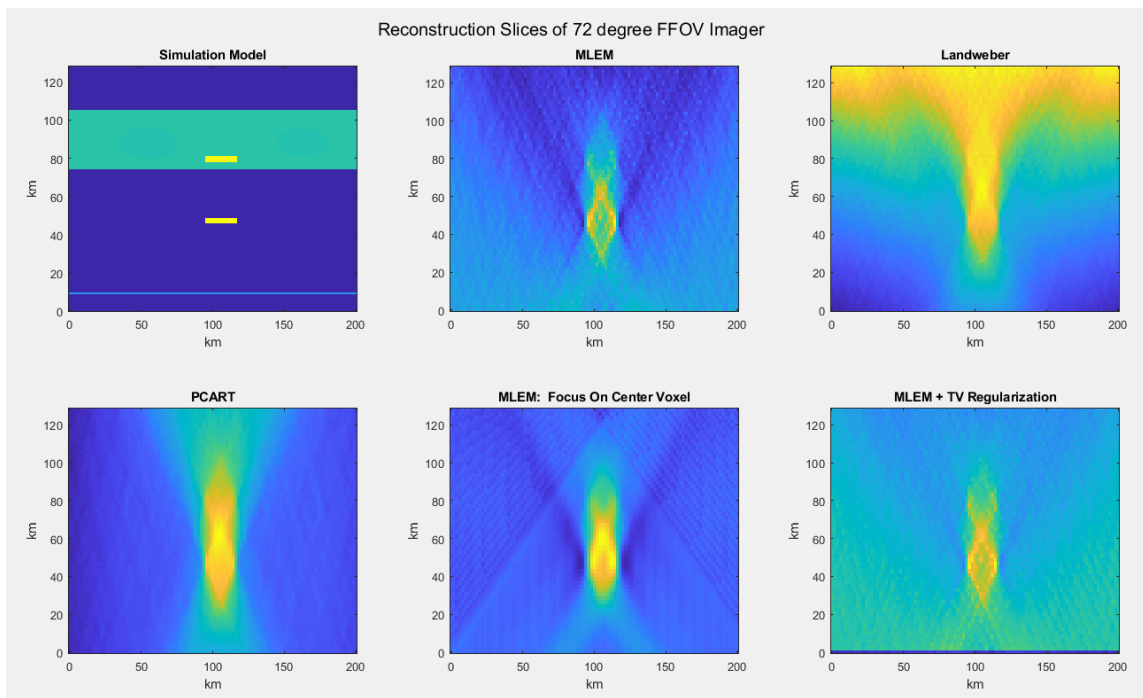


Figure 4.6: Slices of the reconstruction, with the image plane parallel to the direction of propagation. Top Left: The model which is reconstructed in the other five images. The SWIR model has the addition of two bright rectangular boxes. These boxes are not reconstructed properly in the reconstructions. Top Middle: Standard MLEM reconstruction. The two boxes barely have enough separation to be visible and their smear is affecting other altitudes of the reconstruction. There is some edge formation to the two boxes that make them distinguishable. Top Right: The Landweber reconstruction. The upper rectangle can be seen better than that of MLEM, but it is still muddled. Bottom Left: Similar to the standard MLEM reconstruction, but the edge distinctions are not as amplified. Bottom Middle: This is the MLEM algorithm, with the same field of view, but no longer nadir and focusing on the center voxel of the reconstruction space. There are fewer reconstruction artifacts, but the two boxes are still smearing each other. Bottom Right: MLEM with TV regularization. The top and bottom boxes have their edges more pronounced, which is the effect of the TV regularization.

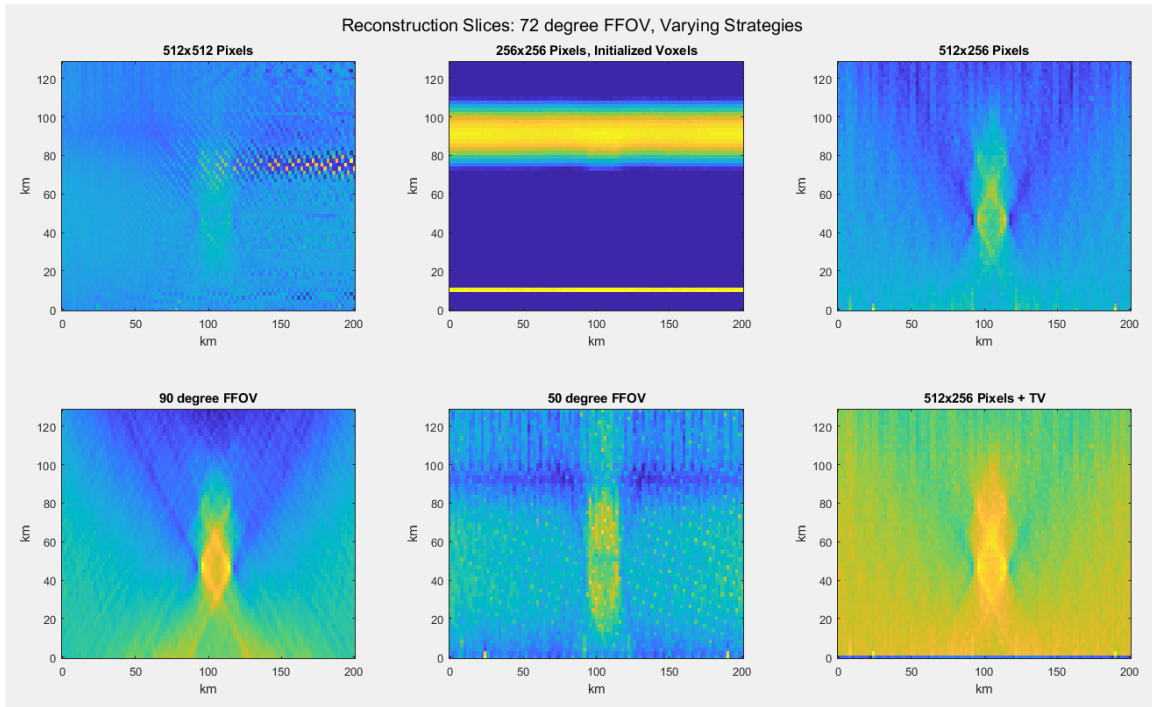


Figure 4.7: Using the same model as figure 4.6, different conditions were attempted with the detector pixels. Top Left: MLEM and 512x512 pixel detector. At the middle right of this image, there are clear artifacts that will affect the overall quality of the image. If there is an effect on the altitude-axis resolution, it is lost in the artifacts present. Top Middle: MLEM with initialized voxels set to amplify the OH-airglow layer and reflection layer. This does not have the bottom box in the initialized portion, so it is not visible in the reconstruction. The top box is barely visible in the 80km layer, but still smears into the layers below. Top Right: MLEM and 512x256 Pixel detector. There are 512 pixels in the direction of propagation. The artifacts from the 512x512 pixel images are gone. There is decent resolution, comparable to 256x256 pixel MLEM. Bottom Left: MLEM with 90-degree FFOV: The resolution is better, but there are streak artifacts due to the wide FOV. Bottom Middle: 50-degree FFOV. There are many artifacts in this image, with a poor resolution. It is interesting to note, though, that the two boxes are still distinguishable. Bottom Right: MLEM with 512x256 detector and TV regularization. This image shows the boxes brighter in the center, something not seen in the other MLEM images.

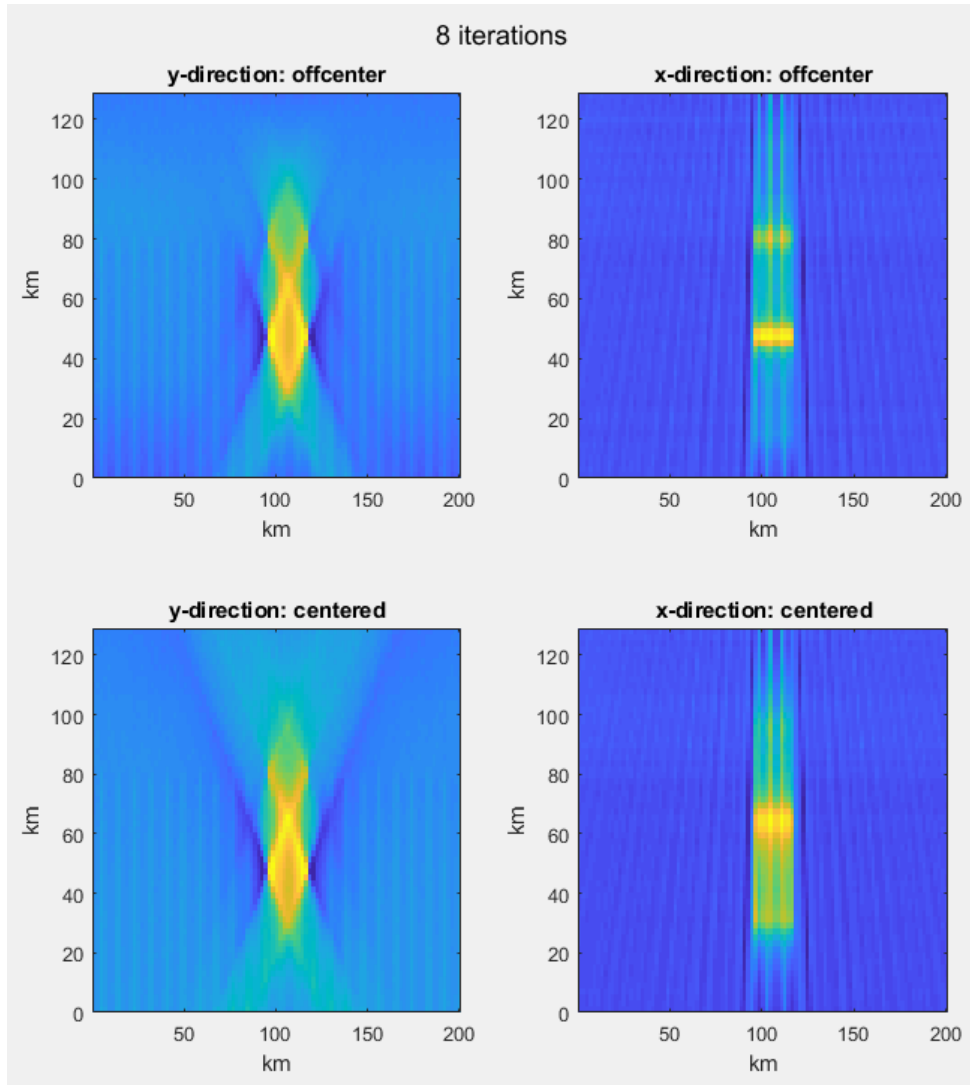


Figure 4.8: Eight iterations of MLEM were performed on the same model that was used in Figure 4.6. Top Left: Looking at a plane that is parallel to the y -direction, and slightly off-center. The rectangles look blended. Top Right: Looking at a plane that is parallel to the x -direction. The two rectangles are separated, though the top one is faint. Bottom Left: The same as the top left image, except that the image is perfectly centered in the voxel space. Bottom Right: Same as the top right, except that the image is perfectly centered in the voxel space. In this image, the smears from both the top and bottom rectangles combine and form what looks like another rectangle near the ~ 64 km altitude.

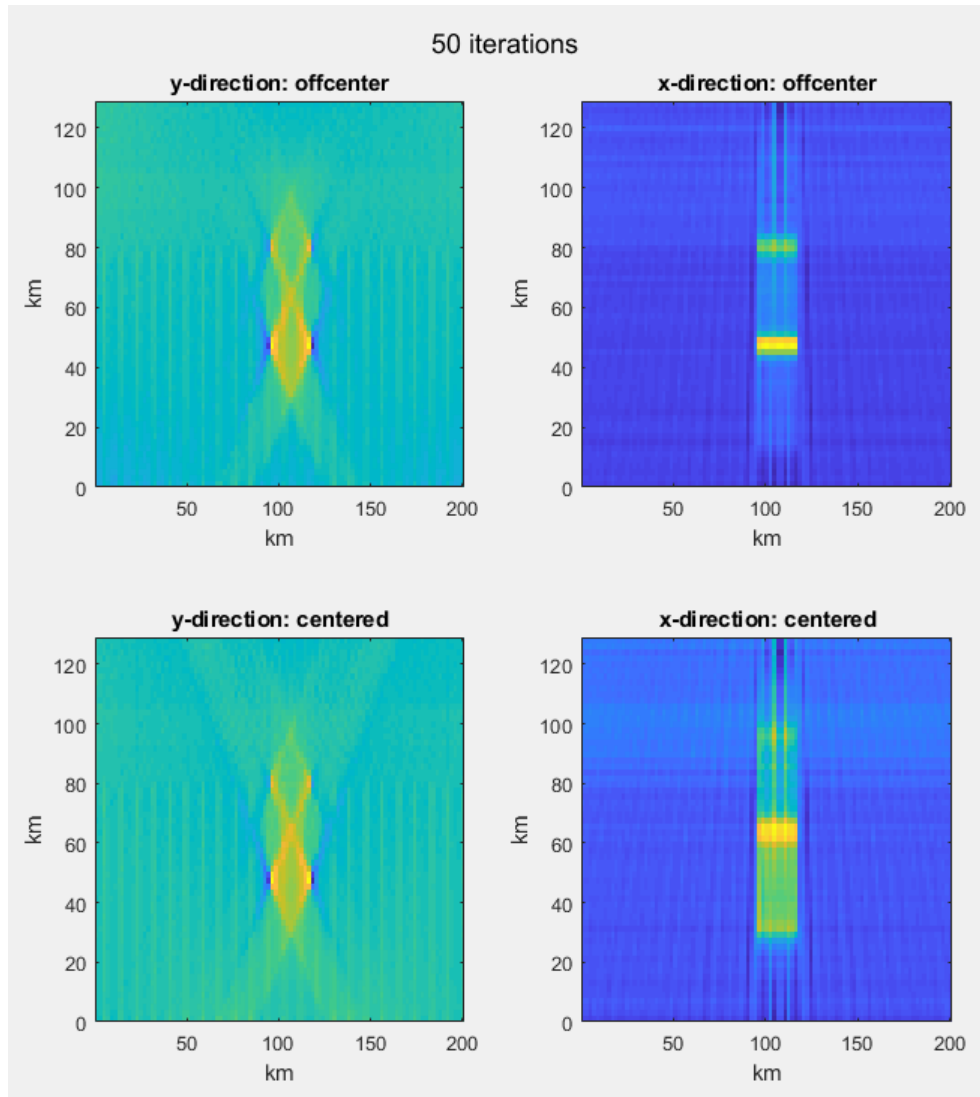


Figure 4.9: Fifty iterations of MLEM were performed on the same mode that was used in Figure 4.6. Top Left: Looking at a plane that is parallel to the y-direction, and slightly off-center. The rectangles have some separation. Top Right: Looking at a plane that is parallel to the x-direction. The two rectangles are separated, though the top one is still faint. Bottom Left: The same as the top left image, except that the image is perfectly centered in the voxel space. Bottom Right: Same as the top right, except that the image is perfectly centered in the voxel space. In this image, the smears from both the top and bottom rectangles combine and form what looks like another rectangle near the ~64 km altitude. There are also extra smear artifacts at ~100 km and ~30 km.

With a 72° FFOV and voxels that are roughly cubic, the best z-axis resolution that would occur for a back-projection ray would be slightly more than one pixel, but as most rays will not be at the outer angles, the expected z-axis resolution of a reconstruction would be far worse. If an object is bright, the reconstruction will also smear the object into altitudes above and beneath it, which degrades those altitudes' image quality. While this may not seem to be great altitude-axis resolution, it is still enough to effectively separate a reflection

in a cloud from an emission in the OH-airglow layer, which have around 30 to 40 pixels of separation in the current model used above.

Each of the methods in section 4.2, apart from a PSF, was attempted to help increase image quality, but there was no ideal solution to improve altitude-axis image quality. The techniques to be used will be dependent on the object or region of interest and its qualities, the imager being used, and the imaging path being taken.

4.5 Conclusion

Limited-angle tomography from space presents a unique set of challenges and opportunities. While each situation is unique, a common goal for tomography from space is finding ways to deal with poor altitude-axis resolution. Despite this setback, implementing real-time MLEM reconstructions is possible and helpful for many situations. As GPUs continue to progress, the applications for using tomography from space will continue to grow and more computing resources can be dedicated to overcoming resolution issues.

CHAPTER 5

3D TOMOGRAPHIC RECONSTRUCTIONS OF SWIR OH-AIRGLOW MODEL

5.1 Introduction

In Chapter 4, the implementation of 3D tomographic reconstructions from space was detailed. Testing the image quality of the reconstructions is important to determine if efforts of perfecting tomography from space are worth investigating. A model was developed to test the tomographic reconstruction algorithm. A figure of merit was then developed and used to determine the effectiveness of the reconstructions.

5.2 Models

5.2.1 Objects and Imaging

A model of the OH-airglow layer is detailed in this section. This is not a comprehensive model of the OH-airglow layer, but a simplified model based on images taken from previous studies from the 1520 nm – 1550nm range.

5.2.2 SWIR Model

The 1550nm OH-Airglow model is based on the altitude and width of the airglow layer, average temperatures of the airglow layer, and the reflections from clouds and ground. It has been simplified to emphasize separating cloud reflections and the OH-Airglow layer.

5.2.2.1 Temperature Model

Using data from SABER reported by Marsh et al. (2006), the temperature in the OH airglow layer ranges between 150 – 240°K. The average temperature is roughly 195K. The temperature is related to the emissions of the OH airglow layer by the equation below (Pautet et al. 2014).

$$T_r = \frac{259.58}{\ln\left(2.644 * \frac{P_1(2)}{P_1(4)}\right)} \quad (5.1)$$

Where $P_1(2)$ and $P_1(4)$ are the brightness of those bands. This makes temperature a good beginning for the backgrounds in a model. If emissions were used, there would be nonlinearities in the reconstructions: the radiance of an individual band does not necessarily increase linearly with temperature. This method is used in AWE to create temperature maps of the OH-airglow layer, which can then be reconstructed using tomography.

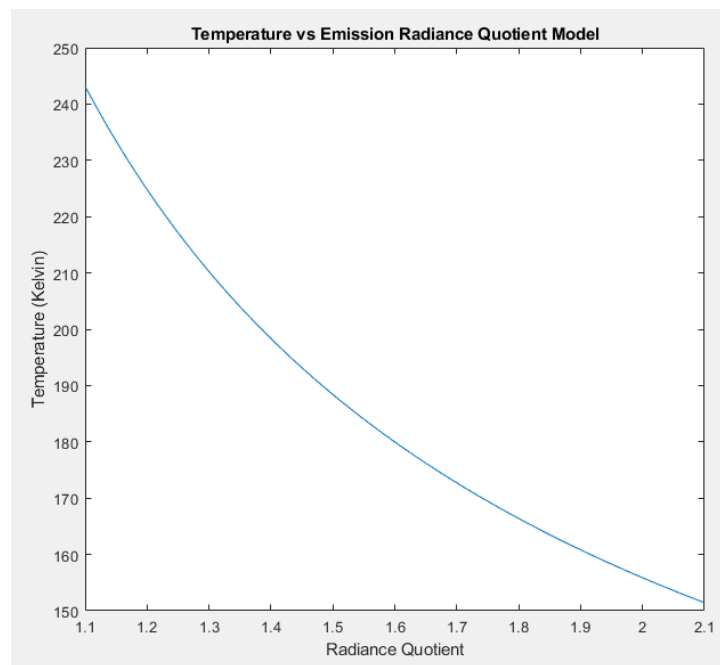


Figure 5.1: This image shows the temperature, in Kelvin, of the OH airglow layer given the radiance quotient of the $P1(2)$ band and $P1(4)$ band. This model is for temperatures between roughly 150–240 K. Using this model, a temperature map can be created from raw images.

The data from SABER also shows that the OH airglow layer is generally only about 10 km wide, but that this width can shift to altitudes between 80-100 km (Marsh et al. 2006, Makhlof, Picard, and Winick 1995). This means that for a model, a conservative estimate could be made where a gaussian distribution of emissions is centered at ~90km and goes to 10 km in both directions. This simplifies the emissions, but still gives the main, brightest emissions to be between 87 – 93 km, which is accurate with previous data reported in

Marsh et al. (2006). By broadening the OH-Airglow layer in this model, it will also help to evaluate the altitude-axis resolution more clearly.

5.2.2.2 Gravity Wave Model

Gravity waves in the OH-airglow layer are generally lower amplitude, but some can have amplitudes of $\pm 20^\circ\text{C}$ or greater. Gravity waves in practice will not be perfect sine waves, but many will be close enough that they can be modeled as sine waves, as shown below in Figure 5.2. Eckerman et al. (2016) also shows a variety of images of gravity waves that appear to be close to sine waves.

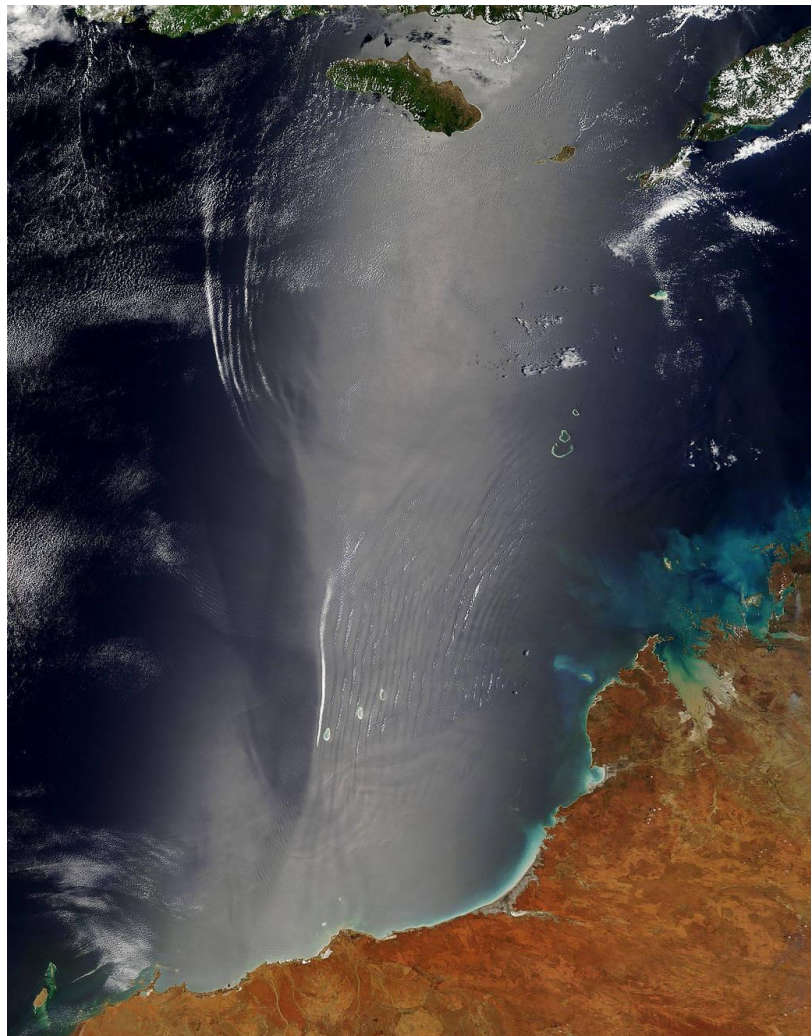


Figure 5.2: Image Credit: NASA/GSFC/MODIS Land Rapid Response Team and Jeff Schmaltz. Image taken from https://www.nasa.gov/multimedia/imagegallery/image_feature_484.html. The image shows gravity waves in the atmosphere. They are not quite sinusoidal, but they are close enough that for a simple model, a sinusoid could work well. This image is in the public domain.

One characteristic of gravity waves is that other minor gravity waves will come and deposit energy at the OH airglow layer. It is assumed that there is a significant number of these minor gravity waves on any given image, and therefore the central limit theorem applies. In other words, this collection of minor gravity waves will look like gaussian distributions on the OH airglow layer. A proper background, then, would be to use a lumpy object model used by Rolland (1990). The formula for this lumpy object model, given in Rolland and Barrett (1992) is as follows:

$$b(r) = \sum_{j=1}^K \frac{b_0}{\pi r_b^2} e^{-\frac{|r-r_j|^2}{r_b^2}} \quad (5.2)$$

Rolland and Barrett (1992) have performed research in creating and assessing observers looking for signals within the lumpy object background.

5.2.2.3 Reflection Model

A reflection model for cloud and ground surfaces is difficult to consider fully. Clouds can range from 0 – 18 km. The reflectivity of clouds can be difficult to simulate as well, as low clouds will have significantly less reflectivity than high clouds. As shown in MODIS images, the Earth is going to be cloudy for the majority of the time (King et al. 2013).

The Earth’s surface will have a variety of reflectances because soil, vegetation, liquid water, ice, and snow all have different reflectances (Wang et al. 2017; Park, Lee, and Jung 2012; Huete 2004). Even within each of these categories, there will be differences. For example, there are key characteristics of soil that affect reflectances in the SWIR range (Jiang 2016).

Because it is difficult to even give educated guesses of how reflections will look, the experience from a previous SWIR imaging satellite, NIRAC, gives insight. Figure 5.3 is an image captured by NIRAC. It shows that the ground reflections come back sharp.

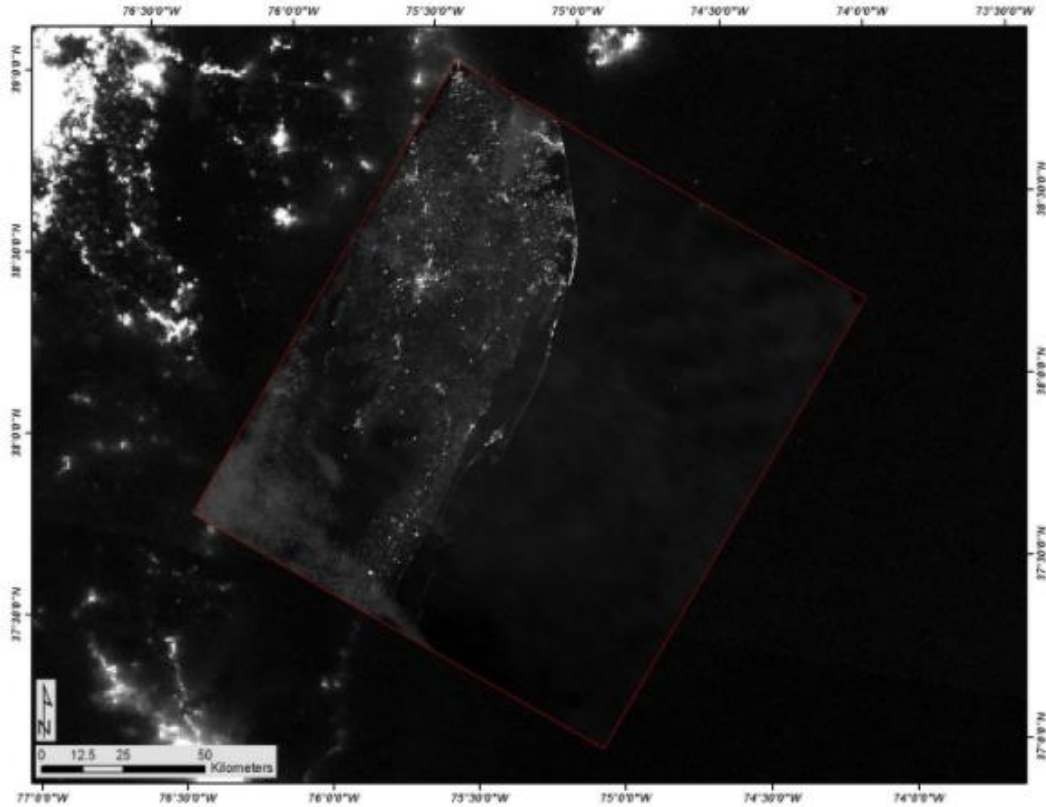


Figure 5.3: Image from NIRAC, taken from <https://aerospace.org/press-release/novel-camera-gives-scientists-night-vision-iss>. The image is of Wallops Island, Virginia. Photo Credit: The Aerospace Corporation

To put this information into a simple model, the following steps were taken:

- 1) High clouds were simulated. This included adding together many 2D gaussian distributions. If the values on the resultant grid were greater than a threshold, those values were kept.
- 2) Low clouds were simulated. The process of simulating low clouds was similar to the process of simulating high clouds, except that if there were high clouds already on a spot where there was already a low cloud, the high cloud had precedence and would be kept.
- 3) Soil was simulated. This was the same as step 2 except that low clouds and high clouds had precedence.
- 4) Water was simulated. This was every pixel that was not cloud or soil.
- 5) These grids were combined to make a single grid. This was put into one single altitude, 10 km.

While this model does not consider intricacies of clouds, soil, vegetation, and the altitude differences of these reflectances, it does provide significant insight into how a tomographic reconstruction could perform for a task that involves these nuisance reflections.

5.2.2.4 Combining the Models

These models were combined by inserting the simulated gravity waves within the simulated OH airglow layer. To finish the reflection layer, the sum of the values of the airglow layer at each XY location was taken and multiplied by the reflectance of the material directly below it, as shown in Figure 5.4. This allowed for sharp edges to be seen, as given in image from NIRAC, and for different reflecting surfaces to be simulated.

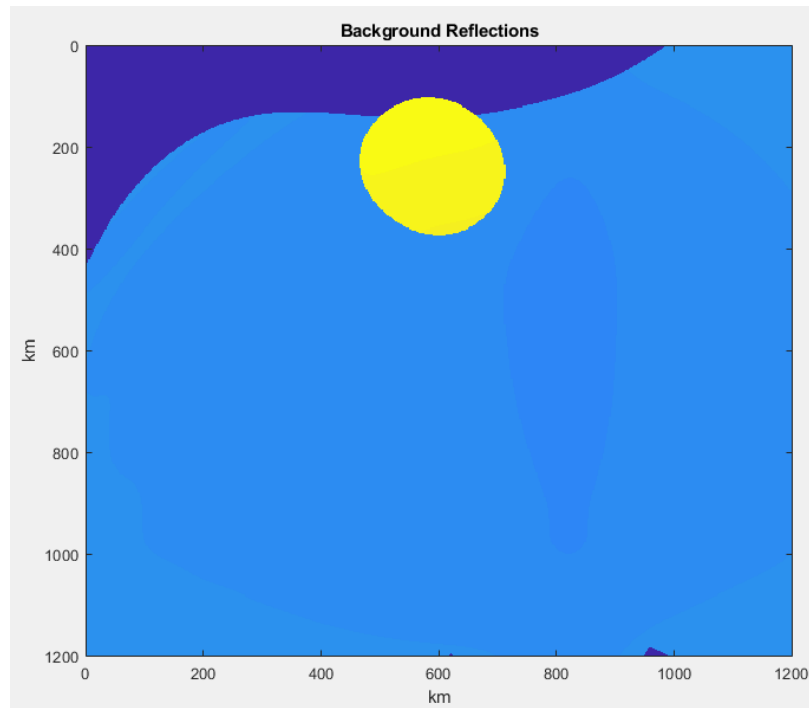


Figure 5.4: An example of the reflection layer in the simulation. There are some sharper edges between water (dark blue), low clouds (dark blue), and high clouds (yellow). This reflection pattern will be present in the projection images generated. The higher values mean that reflectivity is higher, not that emissions are higher.

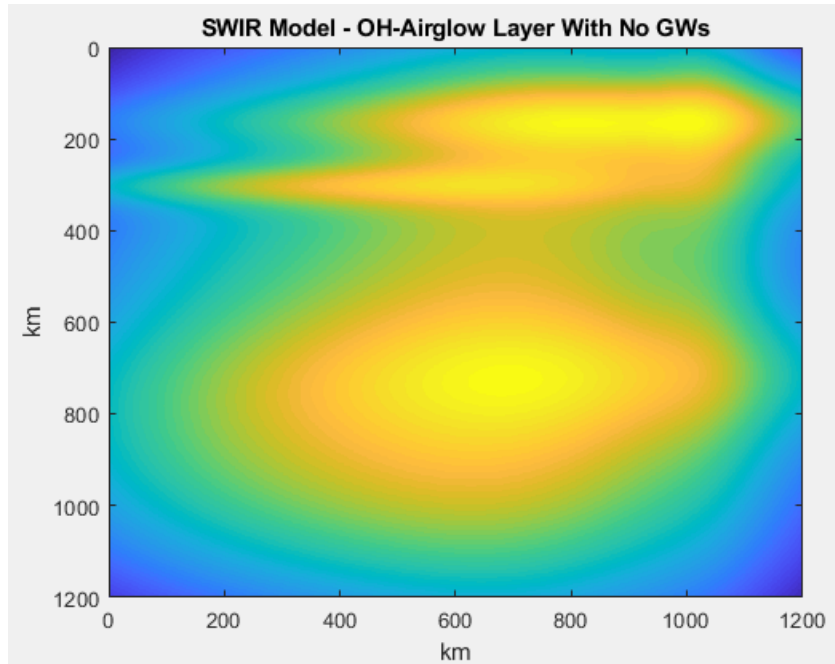


Figure 5.5: An example of the model OH-Airglow layer without any obvious atmospheric gravity waves. The gravity waves are inserted into the model and given an amplitude. These minor disturbances in the OH-Airglow layer are modeled as low frequency fluctuations. When gravity waves are present, these small disturbances are barely visible, but they still affect the image quality.

5.3 Image Quality

With a model defined, tomographic reconstructions were performed. This section details what methods were used to determine how effective the 3D reconstructions were.

5.3.1 Defining a Task for Detecting Gravity Waves

When performing a 3D reconstruction of the SWIR model, a reasonable task is to determine if the detectability of the gravity waves in the OH-airglow layer increases. Higher detectability will show that a 3D reconstruction is effective. A task that was considered was to measure the altitude-axis resolution. Such a task, though, would be dependent on location in the reconstruction and on the object being reconstructed. Because of the uncertainty around this task, it was not pursued with an observer.

5.3.2 Figures of Merit

Section 2.4.4.1.2 describes a process for obtaining a figure of merit for a signal-known-statistically (SKS) case. It takes the Hotelling Observer:

$$t_{fourier} = \vec{s}^t K_g^{-1} \vec{g} \quad (5.3)$$

And modifies it to:

$$t_{fourier} = FT(\vec{s}_{filter}^t) |FT(K_g^{-1} \vec{g})| \quad (5.4)$$

In the case of detecting gravity waves in the SWIR model, the slice corresponding to an altitude of 87 km is taken from the 3D reconstructions, and that will be \vec{g} . The inverse covariance matrix, K_g^{-1} , is calculated using 300 noise-free images and using the method described in Barrett and Myers (2004) Ch 14.3.2. \vec{s}_{filter}^t is a 2D bandpass filter that chooses a desired frequency in the image. In this case, it is a ring around the origin: one where the desired frequency is, and zero where it is not.

A collection of these test statistics, $t_{fourier}$ are then used to create an overall figure of merit for simulations. The equation for the overall figure of merit used is given in Section 2.4.2 but is given below with a change of variables. In the plots given in Section 5.4, SNR_t is described as the detectability of a signal.

$$SNR_t = \frac{\langle t_{fourier} \rangle_{Present} - \langle t_{fourier} \rangle_{Absent}}{\sqrt{\frac{\sigma_{Absent}^2}{2} + \frac{\sigma_{Present}^2}{2}}} \quad (5.5)$$

$\langle t_{fourier} \rangle_{Present}$ is test statistic for the signal-present case, and $\langle t_{fourier} \rangle_{Absent}$ is the test statistic for the signal-absent case. $\sigma_{Present}^2$ and σ_{Absent}^2 are the variance of the signal-present and signal-absent test statistics, respectively.

5.4 Simulation Results

Each simulation involved taking 300 noise-free backgrounds, five sets of 50 simulations, and modified Hotelling observers to get a test statistic. Different parameters were simulated to find the most effective conditions for detectability. Most of the simulations used a 256x256 detector, a 256x256x64 reconstruction space, 80 projection images, and a 72° FFOV which corresponds to a 600 km x 600 km area.

5.4.1 Amplitude, Method, and Detectability

A test was performed where the MLEM, PCART, and Landweber algorithms were all tested with gravity wave (GW) amplitudes of 2°C, 3°C, 5°C, and 10°C. The results of the test are shown in Figure 5.6.

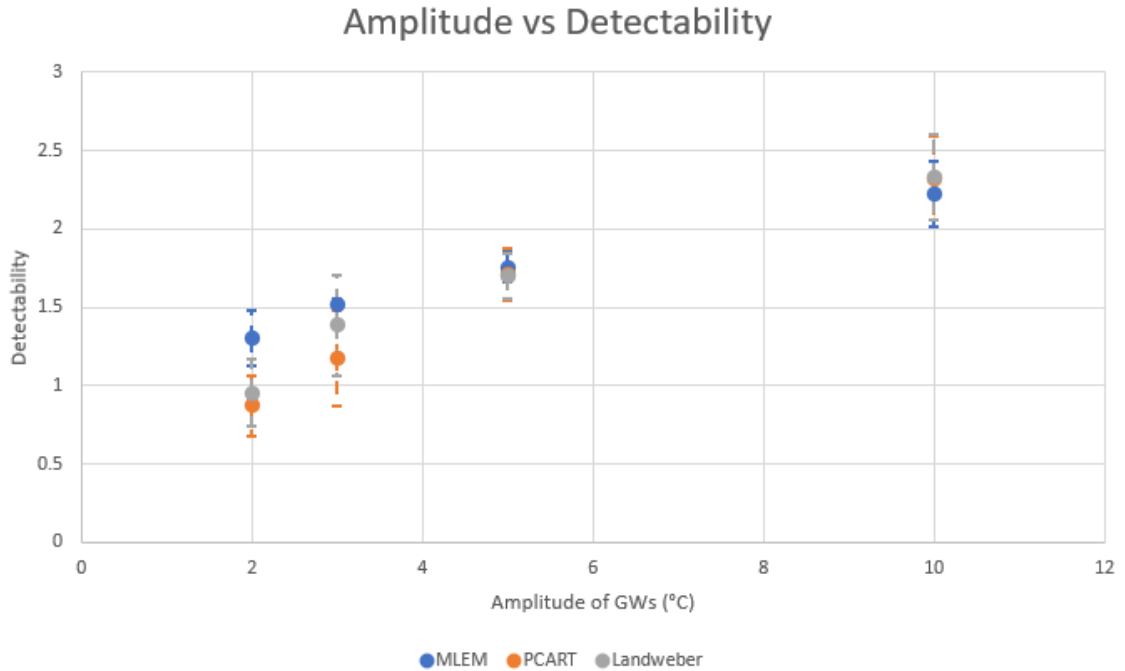


Figure 5.6: For 100 km GWs and eight iterations, this plot shows the relationship between the amplitude of GWs and the detectability of the GWs. Three different reconstruction methods were attempted to test which is most effective. MLEM is shown to be the best for low amplitude signals. The error bars are the standard deviation of each data point.

The simulation shows the effectiveness of MLEM at low signal amplitudes. While PCART and Landweber algorithms perform very well at high amplitudes, they are not as effective at low amplitudes in the current conditions. This is probably due to the multiplicative nature of MLEM, which helps it to converge to a solution faster than the additive PCART and Landweber algorithms. While not shown in the graph above, each of these algorithms is far more effective than looking at one projection image. MLEM's effectiveness in low signal scenarios shows the possibility of detecting gravity waves with amplitudes of even less than 2°C in tomographic reconstructions.

5.4.2 Reconstruction Algorithm Iterations and Detectability

In space applications, speed may be a priority. A straightforward way to cut time from the reconstruction algorithm is to try to minimize the number of iterations that are run. The optimal number of iterations will vary by object and reconstruction parameters, but it is informative to see what it is for a simple case with somewhat lower frequencies.

A simulation was run to test 2, 8, and 12 iterations for resulting image quality. The results are shown in Figure 5.7.

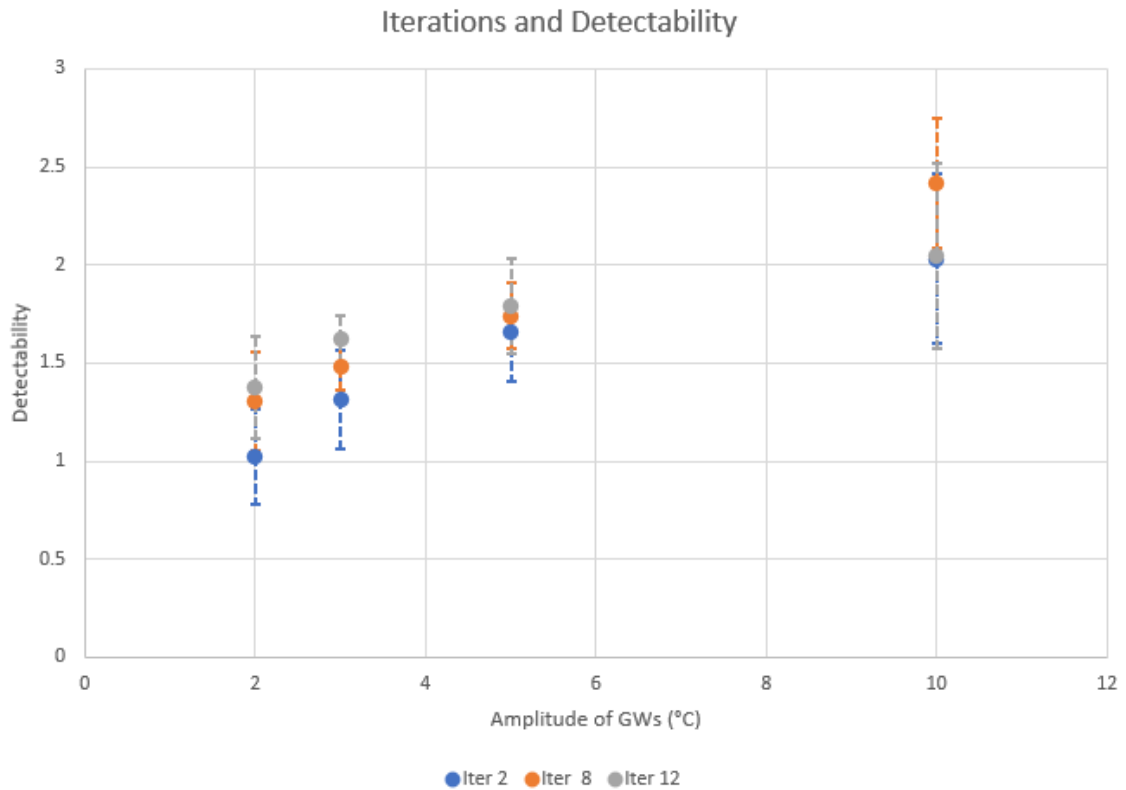


Figure 5.7: This figure shows the detectability of GWs given their amplitude and given the number of iterations run. The error bars are the standard deviation of each data point.

Figure 5.7 shows that for a higher amplitude signal, MLEM with two iterations is adequate. For lower amplitude signals, there is a need for more iterations. Figure 5.7 also shows that there is not much difference between 8 and 12 iterations in this model. This is promising because it shows that optimal image quality can be obtained quick enough that a full reconstruction can be completed before another reconstruction needs to occur for real-time data processing. While there is a slight increase in detectability for 12 iterations instead of

8, the results are still well within the standard deviation and the benefit of extra iterations with detectability appears to approach a limit.

While this approach shows that the detectability of GWs is not improved within OH-airglow layer, it is important to determine the overall imaging tasks at hand when deciding about the number of iterations. Section 5.5 and Section 4.2.1 discuss the relationship between iterations and the frequencies that are retained. If there are higher frequencies that are desired in the final reconstruction, going far beyond 12 iterations may be needed.

5.4.3 Frequency and Detectability

With there being a range of gravity waves that are present in the OH-airglow layer, it is worth investigating if there will be detectability issues with some of these gravity wave wavelengths. Most of the gravity waves that are of concern for AWE are within 30-300 km. Other applications for tomographic reconstructions could have applications for much higher frequencies, so those were investigated as well.

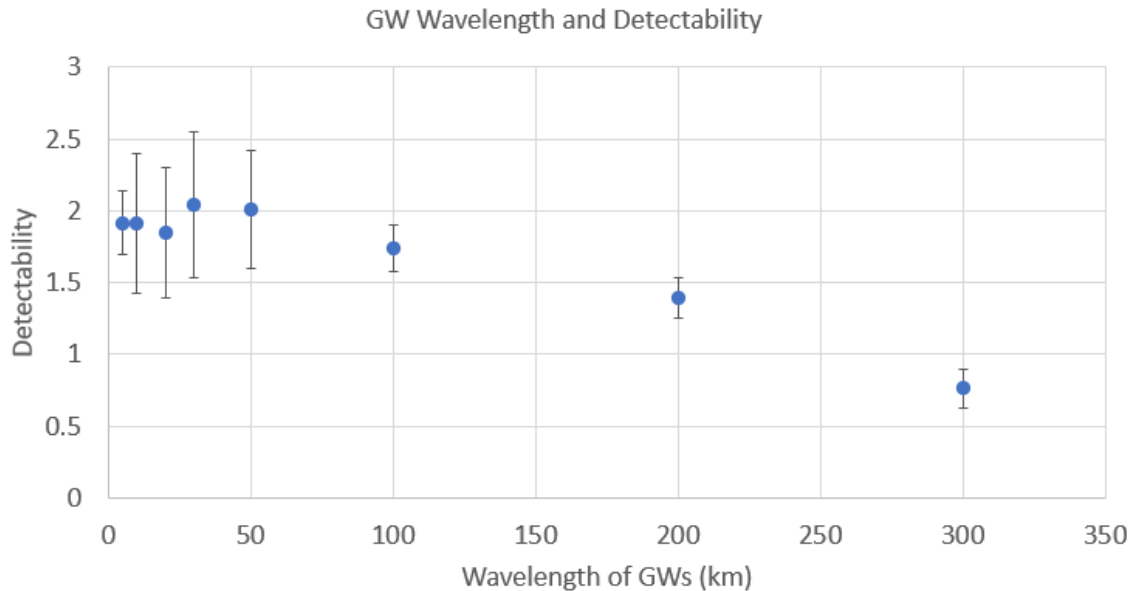


Figure 5.8: The detectability of 5°C amplitude GWs with varying wavelengths. The error bars are the standard deviation. The reconstruction space is 600 x 600 x 128 km, and the image covers a 600 x 600 km swath. With a 256 x 256 detector, to satisfy Nyquist sampling the smallest detectable wavelengths will be about 4.7 km. The error bars are the standard deviation of each data point.

In Figure 5.8, the higher frequency terms have higher detectability. This could be the result of the background structures having lower frequencies and will be explored further in Section 5.5.2.

The detectability of higher frequencies has a much broader impact than gravity waves. Other imagers will have applications that involve smaller phenomenon, which correspond to high spatial frequencies, such as objects on the ground, in the sky, or details within clouds. With higher spatial frequencies giving better detectability, the design of future imaging systems where tomographic reconstructions are performed can try to focus on that sweet spot of detectability.

5.4.4 SNR and Detectability

In MLEM, noise in the reconstruction is dependent on noise in the projections, and is carried through each iteration (Barrett, Wilson, and Tsui 1994; Wilson, Tsui, and Barrett 1994). For this reason, a simulation was run where the noise was set to high values to see how it would affect detectability.

This simulation is especially relevant for the case of AWE, where the lower portion of the detector wells will be primarily used. AWE's detectors are mainly meant for high signal applications, with more noise than most detectors that are used in space, which means that for the lower portion of the detector wells, there will be much higher noise than is normal for a space application. The results for the relationship between SNR and detectability are found in Figure 5.9.

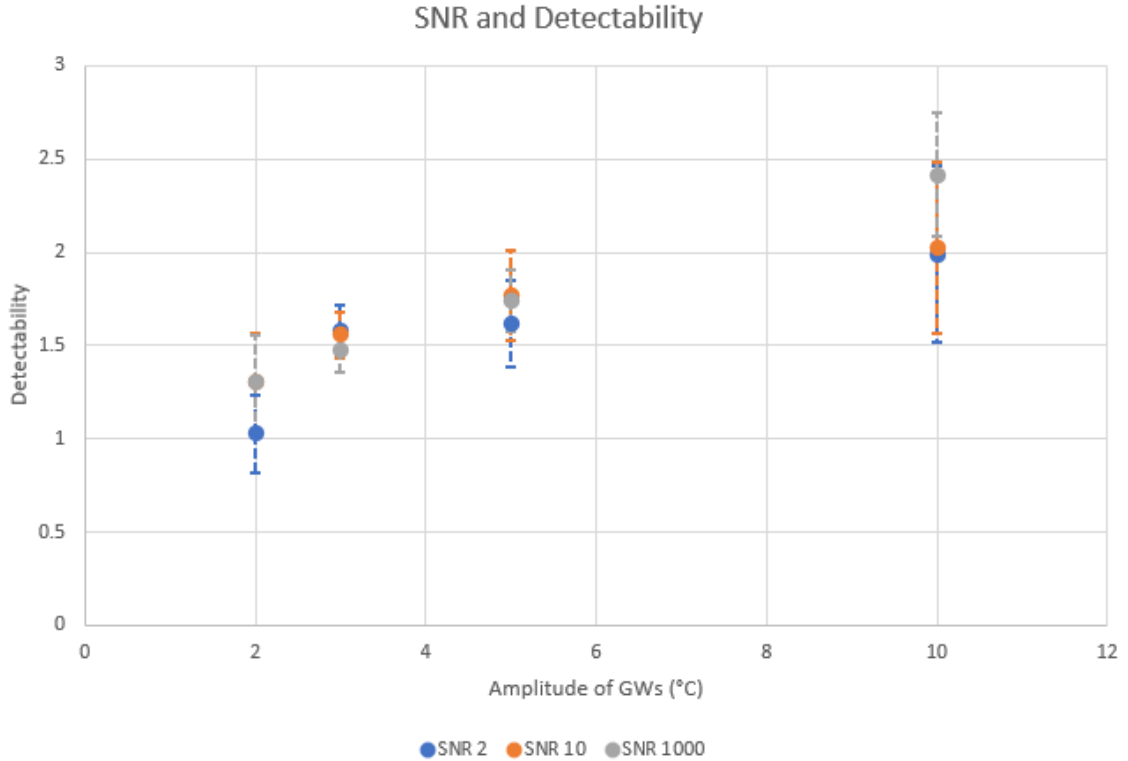


Figure 5.9: This plot, which reconstructs 100 km GWs, shows the relationship between the SNR of images and their reconstruction detectability. There are not many changes for SNR 10 and above, and above SNR 2 for GWs that have an amplitude of at least 3°C. The error bars are the standard deviation of each data point.

The results in Figure 5.9 show that for low signal, and low SNR, there can be a performance hit in detectability. This performance decrease does not completely take away the detectability though. It also shows that for an SNR of 10 and above, there is really no difference between them at lower signal levels. This is a beneficial result for AWE, where the worst-case scenario for one of the detectors will be an SNR of 10. Most cases will be much higher.

5.4.5 Projections and Detectability

Using fewer projections will have effects on altitude-axis resolution, but for AWE, it may be necessary. Some gravity waves will be moving up to 180 m/s. This may not be fast compared to the ISS moving at 7.66 km/s, but it is fast enough that GWs could begin moving out of phase. Simulations were run for 30 and 80 projections to compare image quality. It is expected that performance will decrease, because if the SNR is calculated, the signal will be proportional to the number of images that are used. For the noise, it will

be the variances that add, not the standard deviations, which means that using fewer projections will result in an SNR loss. Figure 5.10 shows the results.

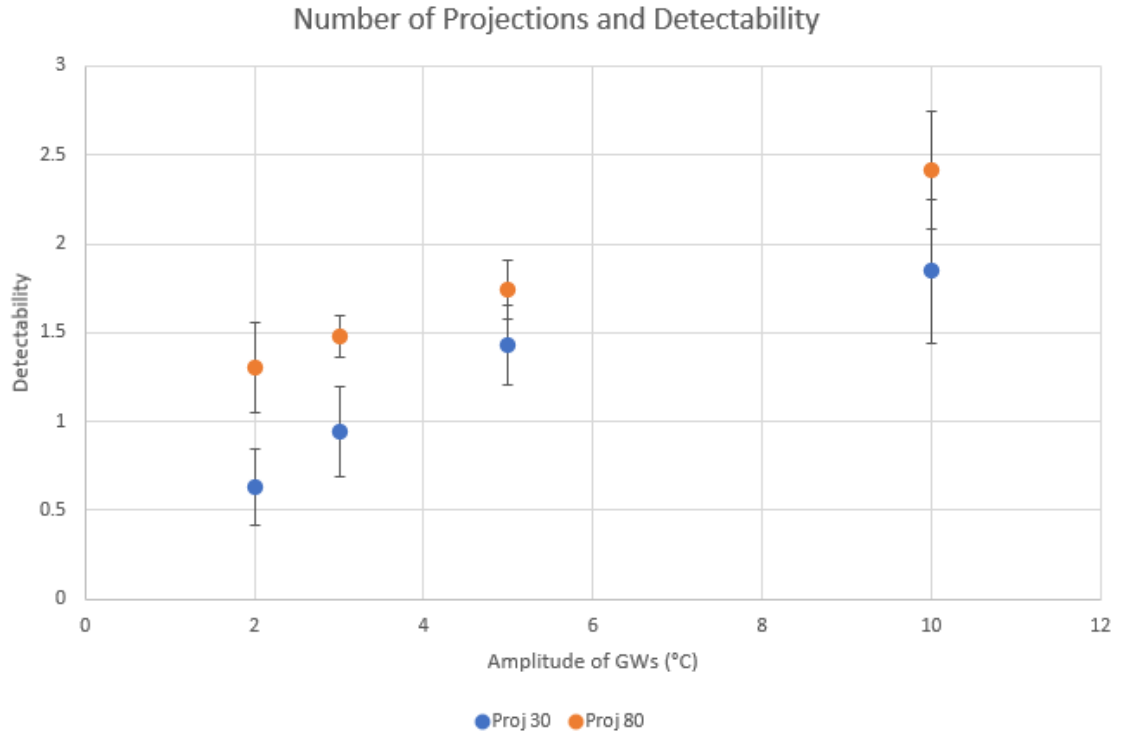


Figure 5.10: This plot shows the relationship between the number of projections that are reconstructed and the detectability of 100 km gravity waves. The x-axis is the amplitude of the GWs being reconstructed and the y-axis is the detectability. The error bars are the standard deviation of each data point.

Even at high amplitudes, the detectability is diminished. For nadir space imaging applications, using as many projections as possible for the region or object of interest is desired for maximum detectability.

5.4.6 Nadir Imaging and Focused Imaging

While AWE is facing nadir, other applications may have the opportunity to focus on a region of interest. This will give better altitude-axis resolution in some cases, as discussed in section 4.2.8, but the question remains for how this will affect XY-axis image quality. A simulation was run for a low signal case. The result was no discernible change from nadir facing imagers. This is a desirable because it shows a way to get better altitude-axis resolution, in certain cases, without sacrificing XY-axis resolution. For the given case, though, the field of view is large enough that focusing on a region of interest does not

provide much altitude-axis benefit. Future research could focus on narrowing the field of view and repeating the experiment.

5.4.7 Field of View and Detectability

In designing telescopes for use in tomographic applications, it would be beneficial to know the effects of the field of view on the design. If the field of view is too wide, the XY-axis resolution will suffer. If the field of view is too narrow, the altitude-axis resolution suffers unless the satellite is focusing on a region of interest. To better understand the trade-off, particularly in the XY-axis, a simulation was run testing different fields of view and the detectability of 100 km gravity waves. As shown in Section 5.4.3, if the GWs have a wavelength above the Nyquist sampling rate, there should be decent detectability.

In running the simulation, the 90° field of view imager had a 15% increase in detectability over the 72° field of view imager. This is consistent with Figure 5.8, which shows higher detectability for higher frequencies, to a certain point. The wider FOV imager in this case will have 100 km GWs as higher frequencies, increasing its detectability. The trade-off is that each pixel covers over 3 km instead of 2.5 km.

5.4.8 TV Regularization and Initializing Voxels

Past research has used initialization of voxels, and TV regularizations to attempt to increase image quality. Low signal simulations were run to see if either of these methods provided an improvement to detectability in the XY-axis.

When initializing voxels to only have values for the OH-airglow layer and the reflection layer, the XY-axis detectability decreased by 14% compared to having an initial voxel space guess of all ones. This is an interesting result, as most of the signal is placed in the correct altitudes. For future research, it would be interesting to go more than eight iterations and to try different initialization strategies to determine the cause for the decrease in detectability.

In using TV-regularization, the L-.9 norm was used, due to reports that the $.8 < p < 1$ Lp-norm perform the best for TV-regularization (Zhang et al. 2018). The result was that TV-regularization did not contribute to increased detectability of GWs. This could be because

of the already low frequency of GWs being used and the relatively simple task of detecting sine-like waves.

5.5 Discussion and Looking Forward

The previous sections give a framework for approaching questions in reconstruction effectiveness and in system design. Some of the main findings for these simulations were as follows:

- 1) MLEM is an effective method for low signal cases
- 2) Higher frequencies have better detectability than low frequencies
- 3) Performing more than about 8 iterations is not useful for the SWIR model

Each of these findings will be investigated here.

5.5.1 MLEM Discussion

What characteristics about MLEM makes it effective for low signal cases? Of the three algorithms tested, MLEM is the only multiplicative algorithm. This can help to achieve a better convergence time, although MLEM convergence times have been criticized by others and modifications have been taken to speed up the convergence of the MLEM algorithm (Slambrouck and Nuyts 2014, De Pierro 1995). Other algorithms, such as OSEM could be tested to compare. In looking at research given in section 2.2, one of the reasons why MLEM and regularizations of MLEM are so popular is because of their performance in low signal, high noise situations. These simulations further show the effectiveness of using MLEM in these strained imaging environments.

5.5.2 Wavelength Discussion

Why do higher spatial frequencies have higher detectability? The SWIR model includes a background that is comparable to the lumpy object model discussed in section 5.2.1.2. These resulting structures in the background tend to have lower frequencies. Some of the models will have higher frequency components, but these will be more rare than common in the airglow layer.

In looking at the characteristics of the model, there will be interference in the lower frequencies of the model and the lower frequencies of GWs, which will bring the

detectability down. The higher frequencies of GWs will not have the same interference with the model. For future models, a higher frequency background could be tested to check if the detectability of the GWs is lower for a higher frequency background.

5.5.3 Discussion on the Number of Iterations

Why does the benefit of more iterations stop at around 8 iterations? The SWIR model has many low frequency components. Even the high frequency gravity waves, at 30 km wavelengths, are low frequency compared to ~4 km Nyquist wavelength of the given imaging system. The reflections have outlines that are sharp, but these outlines are only in certain parts of the reflections. Most of the reflections are smooth. Because there are relatively few projections to reconstruct, and the objects, background, and reflections are lower frequencies, the number of iterations needed will decrease (Alenius, Ruotsalainen, and Astola 1998). If the gravity waves that were of interest had shorter wavelengths, such as 10 km, it would be worth investigating how more iterations change the image quality for the XY-axis.

A determining factor for the number of iterations needed in a limited-angle space application will be the desired altitude-axis resolution. Because there are fewer angles, there will need to be more iterations run to get adequate altitude-axis image quality. The number of iterations can be prohibitive, so it is important that the reconstruction algorithm be fast and that the trade-off between speed and accuracy is explored for the desired application.

5.6 Conclusion

Many parameters were simulated and discussed in this chapter. This work focused on determining the image quality of the XY-axis. Future work could objectively test the altitude-axis resolution. While this chapter was just a beginning of the analysis that can be performed for limited-angle tomographic reconstructions from space, it showed the effectiveness of MLEM, the decision of how many iterations to do, and how the frequency of the object of interest-in combination with the background- can affect the detectability of a desired signal.

CHAPTER 6

Conclusion and Future Work

6.1 Review and Conclusions

In chapter 1, the Atmospheric Waves Experiment (AWE) was detailed. The mission's purpose is to investigate atmospheric gravity waves in the OH-airglow layer. There is current interest in the connection of terrestrial weather and space weather, and GWs are a clear link between the two. The emissions of the OH-airglow layer are dim enough that the AWE instrument will mainly operate in the bottom 5% of its detectors' capacities. Working in the bottom of the detectors' capacity will lead to greater noise, but the SNR will be adequate to effectively image GWs. The MTF of the system also allows for detection of the desired wavelength range.

In chapter 2, the MLEM algorithm was discussed. The formula and current state of MLEM was given in detail. Penalized methods were also discussed. Practical implementation was briefly discussed, as well as the total variation (TV) regularization technique. Image quality of tomographic reconstructions was investigated and a method for creating an observer using Fourier methods was put forward.

Chapter 3 focused on implementing MLEM. Projection and back-projection algorithms were reviewed. A projection algorithm where each voxel was assumed to be a sphere was created and tested. The algorithm's accuracy was found to be superior to the distance driven algorithm in many cases, but the speed of the algorithm was prohibitive for fast reconstructions. A quick algorithm for a back-projection was also reviewed. The code for these algorithms is found in Appendix A.

Performing a tomographic reconstruction from space was the purpose of chapter 4. The limitations and implementation of MLEM from space were explained. A projection algorithm that is similar to the distance driven approach was explored and evaluated for use in nadir image reconstruction. A main theme in reconstructions was the limitation of altitude-axis resolution. With a nadir facing imager, there will be a trade-off between XY

axis resolution and altitude-axis resolution. One of the more promising methods for increasing the resolution of both XY and altitude-axis resolution is to perform more iterations, at the cost of reconstruction time. The speed of the MLEM algorithm made continuous real-time 3D reconstructions possible in certain situations.

Chapter 5 focused on an objective assessment of image quality for the tomographic reconstructions. A model of the OH-airglow layer, clouds, and ground was presented. While it was a simplified model, it proved effective in determining the strengths and weaknesses of reconstruction techniques reconstructing certain characteristics, such as low signal images. MLEM was shown to be the most effective reconstruction technique for low signal imaging. Another finding was that higher frequency signals tended to have better detectability in the SWIR model that was reconstructed. Other simulations were performed which showed the use of the TV algorithm, number of iterations for effective reconstruction, the effectiveness of reconstructing noisy images, reducing the number of images used for reconstructions, focusing on a region of interest, and initializing the voxel space to ones in the OH-airglow layer and zeros outside of it.

6.2 Future Work

There are several items that can be investigated further. These include investigating altitude-axis resolution, comparing the AWE model to real AWE data, and finding ways to increase the speed and accuracy of the MLEM algorithm.

6.2.1 Altitude-Axis Resolution

The altitude-axis limitations of tomographic reconstructions from space are detailed in chapter 4. There are ways that the resolution can be increased, but the most effective methods, such as a full PSF correction for a shift-variant system or increasing the number of iterations, tend to be the most computationally complex. Other methods, such as TV regularization, can show some improvement, but in this document the TV regularization techniques did not show significant enough improvement to focus solely on that method. If the fundamental limit for altitude-axis resolution can be resolved in a practical way, the use of tomographic reconstructions from space can be expanded to applications beyond filtering out high reflectance clouds from the OH-airglow layer emissions.

6.2.2 AWE Model and Real AWE Data

The model used in chapter 5 is a simplified model designed to help determine image quality for tomographic reconstructions. While this model proved to be useful, getting real data from the AWE mission, which launches roughly August 2022, will be instructive for modifying the model for future simulations. One of the main corrections to be accounted for is the amount of reflection that is detected by AWE. An accurate SWIR model can then be expanded for use in other applications besides signals in the OH-airglow layer.

6.2.3 Increasing the Speed and Accuracy of MLEM

Currently, the speed for the AWE reconstructions is about 14 seconds for 8 iterations. A trade-off in projection accuracy and projection speed is seen in the current implementation of MLEM. The sphere voxel projection algorithm is more accurate in many applications but is also up to 10x slower. With better GPUs, the difference in speed could go down, or the overall speed of both algorithms could decrease enough that choosing between the two algorithms will not be as important for real-time imaging.

Further research could also make a projection and back-projection algorithm more specific to the problem at hand. The code given in Appendix A is meant to be general code used for many applications. Making certain assumptions, which makes the code less general, can speed up the code for a specific application. It would be beneficial to resolve some of the bottlenecks in the projection and back-projection algorithms.

6.3 Future of Tomography in Space

While the current limits of speed, accuracy, and resolution tend to restrict the applications of tomography in space, as computing power continues to increase, these barriers can be overcome. There is a large body of research dedicated to projection and back-projection algorithms and their improvement. Even with incremental improvement, it translates into better reconstructions and better image quality.

Appendix A: GPU Code

A.1 Back Projection Code

```
1  /*
2  Back-Projection Algorithm
3
4  Summary:   This code gives a C++ implementation of a back-projection function that
5             is run on the GPU. It does not contain a complete, compilable code,
6             but gives a commented, straightforward implementation. The device code
7             is done in C++ also, but every API that connects to GPU is going to be
8             different. It is also of note that the Host code is not complete.
9             Calling threads and loading buffers is also different for many API's.
10
11 Revision History
12 Author          Date          Revision
13 Matthew Kupinski Apr 24 2019  Original Commit
14 Garrett Hinton  May 30 2019  Commented and Revised
15 */
16
17 /*
18 Host Code
19
20 backProjection: This is the host code that the prepares the device and the
21                buffers for the device. It loops through every projection
22                and reads back the voxels from device to host.
23 Parameters:
24     float *voxels:      The voxels array that will be transferred to the device,
25                        and which will store the final projection
26     float *projectionData: The projection data
27 Output:
28     float *voxels
29 */
30 void backProjection(float *voxels,
31                   float *projectionData)
32 {
33     // Transfer the projection data, voxel array, reconstruction parameters, etc.
34     // to the device memory.
35     // Most of the time, a GPU will not be able hold all data, so some of the
36     // memory may have to be pinned, or set in such a way as to increase transfer
37     // time of the data from host to device.
38     transferBuffersToGPU();
39     //
40     for(int i = 0; i < numProjections; i++)
41     {
42         // This is a call to prepare any buffers and other function called to
43         // prepare the GPU so that the threads can be called
44         prepThreads();
45
46         //Call the kernel
47         backProject(*d_volumeBuffer, // Where to store the voxel values
48                   *d_volumeInfoBuffer, // Position, size, and number of voxels
49                   // to be used in the reconstruction image
50                   *d_projectionBuffer, // Projection data
51                   *d_sourceBuffer, // Position (x,y,z) and cone angle of the
52                   // source
53                   *d_detectorBuffer); // Position, orientation, height, width,
54                                     // and pixel numbers for
55     detector
56     }
57     // Read the voxels from Device memory to Host memory
58     DeviceToHost(voxels, volumeBuffer);
59 }
60
61
62
63
64 /*
65 Device Code:
66
67 backProject: This kernel performs a back-projection onto a set of voxels.
68             Note that there is no information about the angle of the
```

```

69         projection. This is because the detector's position and
70         normal vectors as well as the source's position and direction
71         vectors take this into account. There are different data
72         types used in this kernel (VolumeInfo, Source, and Detector)
73         that are simple structs.
74     Parameters:
75         device float *volumeBuffer:      The voxel data is stored here.
76         device VolumeInfo *volumeInfoBuffer:  Position, size, and number of voxels to
77                                             be used in the reconstruction image
78         device float *projectionBuffer:      Projection data to be backprojected
79         const device Source *sourceBuffer:   Position (x,y,z) and cone angle of the
80                                             source
81         const device Detector *detectorBuffer: Position, orientation, height, width,
82                                             and pixel numbers for detector
83         uint3 id:                            Thread position in grid. Depending on
84                                             the language, this can sometimes be
85                                             called from the kernel instead of
86                                             as a parameter.
87     Output:
88         device float *volumeBuffer
89     */
90     void backProject(   device float *volumeBuffer,
91                       device VolumeInfo *volumeInfoBuffer,
92                       device float *projectionBuffer,
93                       const device Source *sourceBuffer,
94                       const device Detector *detectorBuffer,
95                       uint3 id)
96     {
97         // Position of the voxel
98         float3 voxelPos = ((float3)(2 * (int3)id -
99                             (int3)volumeInfoBuffer->numVoxels + 1) /
100                            ((float3)(volumeInfoBuffer->numVoxels) * 2.0)) *
101                            volumeInfoBuffer->size + volumeInfoBuffer->position;
102
103         // Thread index
104         int bufferIdx = id.x + id.y * volumeInfoBuffer->numVoxels.x +
105                         id.z * volumeInfoBuffer->numVoxels.x *
106                         volumeInfoBuffer->numVoxels.y;
107
108         // This is always set to 0 because each buffer has an offset
109         // when it is called into the kernel
110         int idx = 0;
111
112         // Source position and the direction of propagation
113         float3 sourcePosition = sourceBuffer[idx].position;
114         float3 sourceDirection = normalize(voxelPos - sourcePosition);
115
116         // Position Vector and Normal Vector of the detector.
117         // The normal vector is pointing towards the source
118         float3 detPosition = detectorBuffer[idx].position;
119         float3 detNormal = detectorBuffer[idx].normal;
120
121         // By performing the dot products (dot(detNormal,sourceDirection)
122         // and dot(detNormal,detPosition - sourcePosition)),they give information
123         // about the direction of the xrays with respect to the detector's orientation.
124         float denom = dot(detNormal,sourceDirection);
125         float numer = dot(detNormal,detPosition - sourcePosition);
126         if (abs(denom) < 1e-6)
127         {
128             return;
129         }
130
131         // If 'check' is negative, it means that the detector is not oriented
132         // correctly to receive any photons from the source.
133         float check = numer / denom;
134         if (check < 0.0)
135         {
136             return;
137         }

```



```

138
139 // Gives the point where the source rays will intersect the detector plane
140 float3 intersection = sourcePosition + check * sourceDirection;
141
142 // 1) Computes the cross product of (0,1,0) with the detector normal.
143 // Vector (0,1,0) is used because it is always orthogonal to the
144 // source-to-detector line
145 // 2) Computes the cross product of the detector normal with 1).
146 // Together, xHat and yHat are vectors that define a relative XY
147 // coordinate system on the detector.
148 float3 xHat = cross(float3(0,1,0),detNormal);
149 float3 yHat = cross(detNormal,xHat);
150
151 // Gives a vector that points from the detector position to the
152 // intersection point.
153 float3 inPlane = intersection - detPosition;
154
155 // Projects the point (where the ray hits the detector plane) onto
156 // the new relative coordinates.
157 float xPos = dot(inPlane,xHat);
158 float yPos = dot(inPlane,yHat);
159
160 // These are the dimensions of the detector, giving the edges of the detector.
161 float xmin = -detectorBuffer[idx].width / 2.0;
162 float xmax = detectorBuffer[idx].width / 2.0;
163 float ymin = -detectorBuffer[idx].height / 2.0;
164 float ymax = detectorBuffer[idx].height / 2.0;
165
166 // If the point where the ray hits is not on the detector, return
167 if ((xPos < xmin) || (xPos >= xmax) || (yPos <= ymin) || (yPos > ymax))
168 {
169     return;
170 }
171
172 // At this point, a pixel from the current projection is going to
173 // contribute to the solution
174
175 // Gives the pixel location that will contribute to the voxel.
176 float normXPos = (xPos - xmin) / detectorBuffer[idx].width *
177 (float)detectorBuffer[idx].numPixX;
178 float normYPos = (ymax - yPos) / detectorBuffer[idx].height *
179 (float)detectorBuffer[idx].numPixY;
180
181 // Returns the pixel value (a whole number) in X and Y for the
182 // pixel that will contribute to the voxel.
183 float pixelX = floor(normXPos);
184 float pixelY = floor(normYPos);
185
186 // Returns the fractional part of normXPos and normYpos that is
187 // between 0 and 1.
188 float fractX = normXPos - pixelX;
189 float fractY = normYPos - pixelY;
190
191 // This give the index of the pixel in the projection image
192 int projIndex = pixelX + pixelY * detectorBuffer[idx].numPixX;
193
194 // val11, val12, val21, and val22 are the pixel values that will
195 // contribute to the voxel which corresponds with the current thread.
196 // Because the pixel values are whole numbers and the fractional part
197 // of them was removed, an interpolation needs to be done to get a more
198 // accurate value.
199 float val11 = projectionBuffer[projIndex];
200 float val12 = 0.0;
201 float val21 = 0.0;
202 float val22 = 0.0;
203
204 // These if statements ensure that the pixels are not on an edge,
205 // and if they are, they set the pixel that is off of the detector to 0.
206 if ((pixelX + 1) >= detectorBuffer[idx].numPixX)

```

```

207     {
208         fractX = 0.0;
209         val21 = 0.0;
210         val22 = 0.0;
211     }
212     else
213     {
214         val21 = projectionBuffer[projIndex + 1];
215     }
216     if ((pixelY + 1) >= detectorBuffer[idx].numPixY)
217     {
218         fractY = 0.0;
219         val12 = 0.0;
220         val22 = 0.0;
221     }
222     else
223     {
224         val12 = projectionBuffer[projIndex + detectorBuffer[idx].numPixX];
225     }
226
227     // The other control statements effected fractX and fractY.
228     // If they are still intact, val22 is adjusted.
229     if ((fractX != 0.0) && (fractY != 0.0))
230     {
231         val22 = projectionBuffer[projIndex + detectorBuffer[idx].numPixX + 1];
232     }
233
234     // The vector from the source to the voxel.
235     float3 voxelRelPos = voxelPos - sourcePosition;
236
237     // Object's relative position: The vector from source to the center
238     // of the voxels.
239     float3 objectRelPos = volumeInfoBuffer->position - sourcePosition;
240
241     // Length of the object's relative position
242     float factor1 = length(objectRelPos);
243
244     // Projection of the current voxel's position relative to the center of
245     // the voxels.
246     float factor2 = dot(voxelRelPos, normalize(objectRelPos));
247
248     // Update volumeBuffer:
249     // 1) The first part takes the 4 pixel values of interest
250     //    and interpolates them linearly in 2D.
251     // 2) Takes a ratio of the (length to the center)/(length to the voxel).
252     // Together, these update the voxel value of volumeBuffer[bufferIdx].
253     volumeBuffer[bufferIdx] += ((1.0-fractX) * (1.0-fractY) * val11 +
254                               (fractX)*(1.0-fractY)*val21 + (1.0-fractX)*
255                               fractY*val12 + fractX*fractY*val22) *
256                               (factor1*factor1)/(factor2*factor2);
257 }
258

```

A.2 Projection Code

A projection algorithm is given below where both the voxels and pixels are in the delta basis. The projection algorithm presented in this paper is very similar to the back-projection code, with the last few lines differing as follows:

```
234 // Ratio to account for the attenuation of the beam
235 float attenRatio = (factor2*factor2)/(factor1*factor1);
236
237 // Update projectionBuffer:
238 // 1) The first part takes the fraction of the voxel that is to be added
239 //    to the projectionBuffer.
240 // 2) Takes a ratio of the (length to the voxel)/(length to the center).
241 // Together, these update the pixel values in projectionBuffer.
242 projectionBuffer[projIndex] = atomic_add(projectionBuffer[projIndex],
243     (1.0-fractX) * (1.0-fractY) *
244     volumeBuffer[bufferIdx] * attenRatio);
245 projectionBuffer[projIndex+1] = atomic_add(projectionBuffer[projIndex],
246     fractX * (1.0-fractY) *
247     volumeBuffer[bufferIdx] * attenRatio);
248 projectionBuffer[projIndex + detectorBuffer[idx].numPixX] =
249     atomic_add(projectionBuffer[projIndex],
250     (1.0-fractX) * fractY *
251     volumeBuffer[bufferIdx] * attenRatio);
252 projectionBuffer[projIndex + detectorBuffer[idx].numPixX + 1] =
253     atomic_add(projectionBuffer[projIndex],
254     fractX * fractY *
255     volumeBuffer[bufferIdx] * attenRatio);
256 }
257
```

For the projection code, atomic functions are required. Some methods may use locks or semaphores to do the same task. Some GPU languages, such as Metal 2 (as of 2019), do not have atomic functions for floating point values, which eliminates the usefulness of this program as a general projection operator.

A.3 Projection Using Sphere Voxels

```

1  /*
2  project1:    This kernel performs a projection onto a set of projections.
3              Note that there is no information about the angle of the
4              projection. This is because the detector's position and
5              normal vectors as well as the source's position and direction
6              vectors take this into account. There are different data
7              types used in this kernel (VolumeInfo, Source, and Detector)
8              that are simple structs.
9
10 Parameters:
11 device float *volumeBuffer:    The voxel data is stored here.
12 device VolumeInfo *volumeInfoBuffer:    Position, size, and number of voxels to be
used in the reconstruction image
13 device float *projectionBuffer:    Projection data to be backprojected
14 const device Source *sourceBuffer:    Position (x,y,z) and cone angle of the source
15 const device Detector *detectorBuffer:    Position, orientation, height, width, and
pixel numbers for detector
16 device float *adjustmentFactor    Denominator that the projection will be
divided by.
17 uint3 id:    Thread position in grid. Depending on the
language, this can sometimes be called from
the kernel instead of as a parameter.
18 Output:
19 device float *projectionBuffer
20 */
21 kernel void project1(    device float *volumeBuffer [[buffer(0)]],
22                        const device VolumeInfo *volumeInfoBuffer [[buffer(1)]],
23                        device float *projectionBuffer [[buffer(2)]],
24                        device Source *sourceBuffer [[buffer(3)]],
25                        device Detector *detectorBuffer [[buffer(4)]],
26                        device float *adjustmentFactor [[buffer(5)]],
27                        const uint3 id [[ thread_position_in_grid ]])
28 {
29     // detectorNumPix: aid to calculating pixelPos
30     // PixelPos: Position of the pixel with respect to the first detector position
31     int3 detectorNumPix = int3(detectorBuffer[0].numPixX, detectorBuffer[0].numPixY, 0);
32     float3 pixelPos = ((float3)(2 * (int3)id - detectorNumPix + 1) /
33                        ((float3)(detectorNumPix) * 2.0)) *
34     float3(detectorBuffer[0].width, detectorBuffer[0].height, 0) +
35     detectorBuffer[0].position;
36
37     // Thread index
38     uint bufferIdx = id.x + id.y * detectorBuffer[0].numPixX +
39     id.z * detectorBuffer[0].numPixX * detectorBuffer[0].numPixY;
40
41     // To map the old detector pixel location to the new one, new (xHat, yHat, zHat)
42     vectors must be defined.
43     // The normal vector of the detector will be the new zHat.
44     // The detector and source will always be orthogonal to a y-vector (0,1,0), which
45     is the axis of rotation. This is the old and new yHat. The y-vector crossed with the
46     detector's normal vector will always give the correct reference for the new xHat.
47     float3 xHat = cross(float3(0.0,1.0,0.0),float3(detectorBuffer[id.z].normal));
48
49     // To continue mapping the old detector pixel position onto the new reference, take
50     the old x, y, and z coordinates and multiply them by the new xHat, yHat, and zHat
51     vectors.
52     // These give the positions of each cartesian direction
53     float3 xPos = xHat * pixelPos.x;
54     float3 zPos = (detectorBuffer[id.z].normal * pixelPos.z);
55
56     // Sum these values up and get a new position vector.
57     float3 detectorPos = xPos + float3(0.0,pixelPos.y,0.0) + zPos;
58
59     // The vector from the source position to the detector position.
60     float3 sourceDirection = (detectorPos - sourceBuffer[id.z].position);
61
62     // Iteration variables
63     int3 index = int3(0,0,0);
64
65     // The distance from the voxel to the ray.

```

```

58     float distToRay = 1000.0;
59
60     // A ratio used to help calculate distToRay. Calculated outside of the loop to
    save time. The z-axis needs no scaling. The y-axis needs to be scaled by sqrt(2)
    because that is the distance from the center of a square to the corner. The x-axis
    needs to be scaled by sqrt(3) because that is the distance from the center of a cube to
    a corner. The nested while loops coming up are oriented with x on the outside, y in
    the middle, and z in the inside.
61     float3 realVoxelSideLength = float3(0.0,0.0,0.0);
62     realVoxelSideLength.x =
    (float)volumeInfoBuffer->size.x/(float)volumeInfoBuffer->numVoxels.x*1.75;
63     realVoxelSideLength.y =
    (float)volumeInfoBuffer->size.y/(float)volumeInfoBuffer->numVoxels.y*1.45;
64     realVoxelSideLength.z =
    (float)volumeInfoBuffer->size.z/(float)volumeInfoBuffer->numVoxels.z;
65
66     // The real size of each voxel (mm) divided by 2.0.
67     float3 sizePerVoxelForPos = volumeInfoBuffer->size / (2.0 *
    (float3)(volumeInfoBuffer->numVoxels));
68
69     // The current voxel's position
70     float3 voxelPos = 0;
71
72     // The length of the source-to-pixel vector
73     float lengthSourceDirection = length(sourceDirection);
74
75     // Calculate the bounds for the loop
76     // Convert the detector pixel positions to voxel minMax iterations in each dimension
77     int2 valX;
78     int2 valY;
79     int2 valZ;
80
81     // The slope of the ray for this thread. Source-to-detector-pixel vector
82     float3 slope = normalize(sourceDirection);
83
84     // This is vital for narrowing down the loop min and max. From the source to the
    center of the voxels is a certain length, x. If the size of the voxels in mm is y,
    then we can find the ray's position at x + y, and x - y to get a location of where the
    ray is right before it enters the voxel space, and where it is immediately after. Once
    that is obtained, the position can be converted to a voxel number, which will be an
    edge of the loop.
85     // farEdge is x + y
86     float3 farEdge = slope * (length(volumeInfoBuffer->position -
    sourceBuffer[id.z].position) + (sqrt(3.0) * volumeInfoBuffer->size.x/2.0)) +
    sourceBuffer[id.z].position;
87
88     // nearEdge is x - y
89     float3 nearEdge = slope * (length(volumeInfoBuffer->position -
    sourceBuffer[id.z].position) - (sqrt(3.0) * volumeInfoBuffer->size.x/2.0)) +
    sourceBuffer[id.z].position;
90
91     // This converts farEdge and nearEdge from a position to a voxel position
92     float3 highEdgeVoxIndex = farEdge/(sizePerVoxelForPos * 2.0);
93     float3 lowEdgeVoxIndex = nearEdge/(sizePerVoxelForPos * 2.0);
94
95     // Temporary variable for sorting
96     float tempEdgeVal = 0.0;
97
98     // Sort through highEdgeVoxIndex and lowEdgeVoxIndex. If highEdgeVoxIndex.a is
    smaller than lowEdgeVoxIndex.a, switch them. The while loop increments from low to
    high, so this is necessary for a working while loop.
99     if(highEdgeVoxIndex.x < lowEdgeVoxIndex.x)
100    {
101        tempEdgeVal = highEdgeVoxIndex.x;
102        highEdgeVoxIndex.x = lowEdgeVoxIndex.x;
103        lowEdgeVoxIndex.x = tempEdgeVal;
104    }
105     if(highEdgeVoxIndex.y < lowEdgeVoxIndex.y)
106    {

```

```

107     tempEdgeVal = highEdgeVoxIndex.y;
108     highEdgeVoxIndex.y = lowEdgeVoxIndex.y;
109     lowEdgeVoxIndex.y = tempEdgeVal;
110 }
111 if(highEdgeVoxIndex.z < lowEdgeVoxIndex.z)
112 {
113     tempEdgeVal = highEdgeVoxIndex.z;
114     highEdgeVoxIndex.z = lowEdgeVoxIndex.z;
115     lowEdgeVoxIndex.z = tempEdgeVal;
116 }
117
118 // These are the actual parameters that will be checked by the while loop to
evaluate if the loop will be continued or not. The low index is subtracted by 3 as a
safety measure to ensure a voxel is not lost from the solution. The high index is
added by 3 for the same reason, just a safety measure.
119     valX =int2( lowEdgeVoxIndex.x + volumeInfoBuffer->numVoxels.x/2 - 3,
highEdgeVoxIndex.x + volumeInfoBuffer->numVoxels.x/2 + 3);
120     valY =int2( lowEdgeVoxIndex.y + volumeInfoBuffer->numVoxels.y/2 - 3,
highEdgeVoxIndex.y + volumeInfoBuffer->numVoxels.y/2 + 3);
121     valZ =int2( lowEdgeVoxIndex.z + volumeInfoBuffer->numVoxels.z/2 - 3,
highEdgeVoxIndex.z + volumeInfoBuffer->numVoxels.z/2 + 3);
122
123 // Checks to make sure that the bounds of the loop are within the bounds of the
voxel array
124 if(valX.x < 0){valX.x = 0;}
125 if(valY.x < 0){valY.x = 0;}
126 if(valZ.x < 0){valZ.x = 0;}
127 if(valX.y < 0){valX.y = 0;}
128 if(valY.y < 0){valY.y = 0;}
129 if(valZ.y < 0){valZ.y = 0;}
130 if(valX.y > (int)volumeInfoBuffer->numVoxels.x){valX.y =
volumeInfoBuffer->numVoxels.x;}
131 if(valY.y > (int)volumeInfoBuffer->numVoxels.y){valY.y =
volumeInfoBuffer->numVoxels.y;}
132 if(valZ.y > (int)volumeInfoBuffer->numVoxels.z){valZ.y =
volumeInfoBuffer->numVoxels.z;}
133 if(valX.x > (int)volumeInfoBuffer->numVoxels.x){valX.x =
volumeInfoBuffer->numVoxels.x;}
134 if(valY.x > (int)volumeInfoBuffer->numVoxels.y){valY.x =
volumeInfoBuffer->numVoxels.y;}
135 if(valZ.x > (int)volumeInfoBuffer->numVoxels.z){valZ.x =
volumeInfoBuffer->numVoxels.z;}
136
137 // This is part of the calculation to give the exact voxel position. Because this
part of the calculation is not dependant on the voxel position, it is out of the loop
to optimize for speed.
138     float3 posTemp = (((float3)volumeInfoBuffer->numVoxels + 1) * sizePerVoxelForPos);
139
140 // This variable is being adjusted to be the size of a voxel, instead of half the
size of a voxel.
141     sizePerVoxelForPos = 2.0 * sizePerVoxelForPos;
142
143 //voxel volume
144     float voxVolume = sizePerVoxelForPos.x * sizePerVoxelForPos.y * sizePerVoxelForPos.z;
145
146 // This is the radius of the spherical voxels. Currently the same area as a cube
voxel.
147     float decisionThreshold = pow(voxVolume*3.0/(4.0 * 3.14159265), (1.0/3.0));
148 // Other common possibilities for decisionThreshold:
149 // detectorBuffer[0].height/(float)detectorBuffer[0].numPixX *
150 //     length(volumeInfoBuffer->position - sourceBuffer[id.z].position)
151 //     /lengthSourceDirection;
152 // sizePerVoxelForPos.x*sqrt(2.0)/2.0;
153
154 // This is used by the X and Y dimension in the while loop to know when, and how
far ahead, to jump in the loop
155     float2 closest = float2(1000.0,1000.0);
156
157 // The array location of the voxel value needed

```

```

158     uint loopID = 0;
159
160     // This is used in the loop in connection to the 'closest' variable above to jump
161     // ahead in the loop.
162     int3 temp = int3(0,0,0);
163
164     // The starting index, or x-min for the x-dimension
165     index.x = valX.x;
166     /*
167     Loop through every voxel. If the ray is within a certain distance from a voxel, it
168     will contribute to the pixel on the detector. If it is not within that certain
169     distance voxels, add the iteration variable in the Z-dimension by the distance to the
170     ray to provide some speedup. After the Z-dimension has been iterated through, if a ray
171     has not contributed to a pixel in that line, a jump in the y-direction will be
172     proportional to the closest distance that a ray was to that line. If, in the entire
173     YZ plane, no ray was hit, a jump in the X-direction will be proportional to the closest
174     distance that a ray was to hitting that plane. If a ray was hit in the z-dimension, no
175     jumps are made in the Y or X dimension.
176     */
177     while (index.x < valX.y)
178     {
179         // Set the y-min
180         index.y = valY.x;
181
182         // Set the closest x-value to a high number so that it can be replaced with
183         // smaller values.
184         closest.x = 1000.0;
185         while (index.y < valY.y)
186         {
187             // Set the z-min
188             index.z = valZ.x;
189
190             // Set the closest y-value to a high number so that it can be replaced with
191             // smaller values.
192             closest.y = 1000.0;
193             /*
194             Do the calculations for index 511 in order to get complete data. If this
195             is not done, a hole in the closest.y and closest.x data occurs and a jump can be made
196             that is bigger than expected.
197             */
198             // Get the current voxel's position in float3(mm,mm,mm)
199             voxelPos = (float3(index.x,index.y,volumeInfoBuffer->numVoxels.z - 1) *
200             sizePerVoxelForPos - posTemp);
201
202             // Calculate the closest distance to the ray. The shortest distance to the
203             // ray will be magn[(detectorPos - sourcePos) x (voxelPos -
204             // SourcePos)]/magn[detectorPos-sourcePos]. Where --> SourceDirection = (detectorPos -
205             // SourcePos)
206             distToRay = length(cross(sourceDirection, voxelPos -
207             sourceBuffer[id.z].position)) / lengthSourceDirection;
208
209             //Ideal decisionThreshold for Sphere voxel: Not currently used
210             //decisionThreshold =
211             detectorBuffer[0].height/(float)detectorBuffer[0].numPixX * length(voxelPos -
212             sourceBuffer[id.z].position) /lengthSourceDirection;
213
214             // If the distance to the ray is within a certain distance, add the voxel
215             // value to the pixel.
216             if(distToRay < decisionThreshold)
217             {
218                 // Get the array ID for the voxel.
219                 loopID = volumeInfoBuffer->numVoxels.x - 1 +
220                 (volumeInfoBuffer->numVoxels.x - index.y) * volumeInfoBuffer->numVoxels.x + (index.z) *
221                 volumeInfoBuffer->numVoxels.y * volumeInfoBuffer->numVoxels.x;
222
223                 // Add the voxel value to the pixel value
224                 projectionBuffer[bufferIdx] +=
225                 volumeBuffer[loopID]/adjustmentFactor[0];/**(2*sqrt(decisionThreshold *
226                 decisionThreshold - distToRay*distToRay));**/2.74;

```

```

202     }
203
204     else
205     {
206         // If the distance to a ray is not close enough to count toward a
pixel, find out if it is smaller than the previous closest values to a ray, and keep
the smallest value.
207         closest.y = min(distToRay,closest.y);
208         closest.x = min(distToRay,closest.x);
209     }
210
211     while (index.z < valZ.y)
212     {
213         // Current position of the voxel
214         voxelPos = ((float3)index * sizePerVoxelForPos - posTemp) +
volumeInfoBuffer->position;
215
216         //Ideal decisionThreshold for Sphere voxel: Not currently used
217         //decisionThreshold =
detectorBuffer[0].height/(float)detectorBuffer[0].numPixX * length(voxelPos -
sourceBuffer[id.z].position) /lengthSourceDirection;
218
219         // Calculate the closest distance to the ray. The shortest distance to
the ray will be magn[(detectorPos - sourcePos) x (voxelPos -
SourcePos)]/magn[detectorPos-sourcePos]. Where --> SourceDirection = (detectorPos -
SourcePos)
220         distToRay = length(cross(sourceDirection, voxelPos -
sourceBuffer[id.z].position)) / lengthSourceDirection;
221
222         //increment the counter by how far away the normal line is. If it is
less than 1 voxel, add it to the solution and then increment by 1.
223
224         // If the distance to the ray is within a certain distance, add the
voxel value to the pixel.
225         if(distToRay < decisionThreshold)/(realVoxelSideLength.z/1.414))
226         {
227             // Get the array ID for the voxel.
228             loopID = index.x + (volumeInfoBuffer->numVoxels.x - index.y) *
volumeInfoBuffer->numVoxels.x + (index.z) * volumeInfoBuffer->numVoxels.y *
volumeInfoBuffer->numVoxels.x;
229
230             // Add the voxel value to the pixel value
231             projectionBuffer[bufferIdx] +=
volumeBuffer[loopID]/adjustmentFactor[0];/**(2*sqrt(decisionThreshold *
decisionThreshold - distToRay*distToRay));**//2.74;
232
233             // Because a ray is close, only increment in the z-direction by 1.
This is because a ray will be close to other voxels in this area and they may
contribute to the solution.
234             index.z++;
235         }
236
237     else
238     {
239         // If the distance to a ray is not close enough to count toward a
pixel, find out if it is smaller than the previous closest values to a ray, and keep
the smallest value.
240         closest.y = min(distToRay,closest.y);
241         closest.x = min(distToRay,closest.x);
242
243         // Temp.z is an int that will be added to the current z-dimension
iterator. It is the final voxel value that the z-dimension will jump. If the distance
to a ray is less than a voxel's size, only increment the iterator by 1, otherwise, add
to the iterator by the distance the voxel is from the ray. The farther from a ray a
voxel is, the bigger the jump.
244         temp.z = (distToRay / realVoxelSideLength.z) - 1;
245
246         // Be sure that the loop index is added by at least 1 so that it
does not get stuck.

```



```

247         if (temp.z <= 0){temp.z = 1;}
248         index.z += temp.z;
249     }
250
251 }
252
253 // Jump in the y-direction by the closest distance from a ray.
254 temp.y = closest.y/realVoxelSideLength.y;
255
256 // Be sure that the loop index is added by at least 1 so that it does not
get stuck.
257 if(temp.y == 0){ temp.y = 1; }
258 index.y += temp.y;
259
260 }
261
262 // Jump in the x-direction by the closest distance from a ray.
263 temp.x = closest.x/realVoxelSideLength.x;
264
265 // Be sure that the loop index is added by at least 1 so that it does not get
stuck.
266 if(temp.x == 0){ temp.x = 1; }
267 index.x += temp.x;
268
269 }
270 }
271
272

```

Appendix B: Regularization Technique for MLEM

This is taken from OPTI 637 Class Notes at the University of Arizona, taught by Dr. Eric Clarkson:

Using cross entropy, which is also called the Kullback Leibler divergence, for the data agreement term and regularizer:

$$Q(f, g) = \sum_{M=1}^M \left\{ (Hf)_m - g_m + g_m \ln \left[\frac{g_m}{(Hf)_m} \right] \right\} + \eta R(f)$$

Minimize f :

$$\frac{\partial}{\partial f_n} Q(f, g) = \sum_{M=1}^M \left\{ H_{mn} - g_m \left[\frac{H_{mn}}{(Hf)_m} \right] \right\} + \eta \frac{\partial R}{\partial f_n}(f) = 0$$

Another way to write this equation is:

$$\sum_{M=1}^M H_{mn} - \sum_{m=1}^M H_{mn} \left[\frac{g_m}{(Hf)_m} \right] + \eta \frac{\partial R}{\partial f_n}(f) = 0$$

The sensitivity vector, s , is defined as follows:

$$s_n = \sum_{m=1}^M H_{mn}$$

Putting it in vector form:

$$s - H^\dagger \left(\frac{g}{Hf} \right) + \eta \nabla_f R(f) = 0$$

Multiply both sides component-wise by f :

$$fs = f \left[H^\dagger \left(\frac{g}{Hf} \right) \right] - \eta f \nabla_f R(f)$$

Component-wise division by s :

$$f = \frac{f}{s} \left[H^\dagger \left(\frac{g}{Hf} \right) \right] - \eta \frac{f}{s} \nabla_f R(f)$$

Convert to an iterative algorithm:

$$f^{(k+1)} = \frac{f^{(k)}}{s} H^\dagger \left(\frac{\mathbf{g}}{Hf^{(k)}} \right) - \eta \frac{f^{(k)}}{s} \nabla_f R(f^{(k)})$$

Written in component form:

$$f_n^{(k+1)} = \frac{f_n^{(k)}}{s_n} \sum_{m=1}^M H_{mn} \left[\frac{g_m}{(Hf^{(k)})_m} \right] - \eta \frac{f_n^{(k)}}{s_n} \frac{\partial}{\partial f_n} R(f^{(k)})$$

References

- Akmaev, R. A. "Whole atmosphere modeling: Connecting terrestrial and space weather." *Reviews of Geophysics* 49.4 (2011).
- Alenius, S., U. Ruotsalainen, and J. Astola. "Using local median as the location of the prior distribution in iterative emission tomography image reconstruction." *IEEE Transactions on Nuclear Science* 45.6 (1998): 3097-3104.
- Ammosov, P. P., et al. "Cross-calibration of ground-based measurement of rotational temperature of OH (3-1) at altitude~ 87 km in Maimaga (63° N, γ = 129.5° E) and SABER/TIMED satellite data." *Journal of Physics: Conference Series*. Vol. 1152. No. 1. IOP Publishing, 2019.
- AWE Phase A Concept Study Report. Space Dynamics Laboratory, July 2018.
- Barrett, H., & Myers, K. (2004). *Foundations of image science* (Wiley series in pure and applied optics). Hoboken, NJ: Wiley-Interscience.
- Barrett, Harrison H., Donald W. Wilson, and Benjamin MW Tsui. "Noise properties of the EM algorithm. I. Theory." *Physics in Medicine & Biology* 39.5 (1994): 833.
- Barrett, Harrison H., et al. "Model observers for assessment of image quality." *Proceedings of the National Academy of Sciences* 90.21 (1993): 9758-9765.
- Basu, Samit, and Bruno De Man. "Branchless distance driven projection and backprojection." *Computational Imaging IV*. Vol. 6065. International Society for Optics and Photonics, 2006.
- Bousse, Alexandre, et al. "Maximum-likelihood joint image reconstruction/motion estimation in attenuation-corrected respiratory gated PET/CT using a single attenuation map." *IEEE transactions on medical imaging* 35.1 (2015): 217-228.
- Brink, James A., and John D. Boice Jr. "Science to practice: can antioxidant supplements protect against the possible harmful effects of ionizing radiation from medical imaging?." *Radiology* 264.1 (2012): 1-2.

- Caceres, Lucila G., et al. "An early treatment with 17- β -estradiol is neuroprotective against the long-term effects of neonatal ionizing radiation exposure." *Journal of neurochemistry* 118.4 (2011): 626-635.
- Carsten, Ronald E., et al. "Resveratrol reduces radiation-induced chromosome aberration frequencies in mouse bone marrow cells." *Radiation research* 169.6 (2008): 633-638.
- Chandran, A., et al. "Polar mesospheric cloud structures observed from the cloud imaging and particle size experiment on the Aeronomy of Ice in the Mesosphere spacecraft: Atmospheric gravity waves as drivers for longitudinal variability in polar mesospheric cloud occurrence." *Journal of Geophysical Research: Atmospheres* 115.D13 (2010).
- Chang, De-Hua, et al. "Low-dose computed tomography of urolithiasis in obese patients: a feasibility study to evaluate image reconstruction algorithms." *Diabetes, metabolic syndrome and obesity: targets and therapy* 12 (2019): 439.
- Chávez-Rivera, Lucia B., et al. "ML-EM reconstruction model including total variation for low dose PET high resolution data." *2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. IEEE, 2015.
- Chen, Ken, et al. "GPU based parallel acceleration for fast C-arm cone-beam CT reconstruction." *Biomedical engineering online* 17.1 (2018): 1-14.
- Cinkilic, Nilufer, et al. "Radioprotection by two phenolic compounds: chlorogenic and quinic acid, on X-ray induced DNA damage in human blood lymphocytes in vitro." *Food and Chemical Toxicology* 53 (2013): 359-363.
- Clarkson, Eric. *Optical Sciences 637: Principles of Image Science*. 30 Aug. 2020, University of Arizona, Tucson. Class lecture.
- Cui, Jingyu, et al. "Distributed MLEM: An iterative tomographic image reconstruction algorithm for distributed memory architectures." *IEEE transactions on medical imaging* 32.5 (2013): 957-967.
- Curcio, D., et al. "Prediction of soil texture distributions using VNIR-SWIR reflectance spectroscopy." *Procedia Environmental Sciences* 19.494 (2013): e503.

- Damle, Nishikant Avinash, et al. "SPECT/CT in the diagnosis of skull base osteomyelitis." *Nuclear medicine and molecular imaging* 45.3 (2011): 212-216.
- Das, Birajalaxmi, et al. "Melatonin protects human cells from clustered DNA damages, killing and acquisition of soft agar growth induced by X-rays or 970 MeV/n Fe ions." *International journal of radiation biology* 87.6 (2011): 545-555.
- De González, Amy Berrington, et al. "Projected cancer risks from computed tomographic scans performed in the United States in 2007." *Archives of internal medicine* 169.22 (2009): 2071-2077.
- De Man, Bruno, and Samit Basu. "Distance-driven projection and backprojection." *2002 IEEE Nuclear Science Symposium Conference Record*. Vol. 3. IEEE, 2002.
- De Man, Bruno, and Samit Basu. "Distance-driven projection and backprojection in three dimensions." *Physics in Medicine & Biology* 49.11 (2004): 2463.
- De Pierro, Alvaro R. "A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography." *IEEE transactions on medical imaging* 14.1 (1995): 132-137.
- Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977): 1-22.
- Dittmann, Jonas, Randolph Hanke, and E. Z. R. T. Fraunhofer. "Simple and efficient raycasting on modern GPU's read-and-write memory for fast forward projections in iterative CBCT reconstruction." *The 14th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*. 2017.
- Domina, E. A., O. P. Pylypchuk, and V. M. Mikhailenko. "Destabilization of human cell genome under the combined effect of radiation and ascorbic acid." *Experimental oncology* (2014).
- Dreier, Thomas, and David J. Rakestraw. "Measurement of OH rotational temperatures in a flame using degenerate four-wave mixing." *Optics letters* 15.1 (1990): 72-74.
- Eckermann, Stephen D., et al. "Dynamics of orographic gravity waves observed in the mesosphere over the Auckland Islands during the Deep Propagating Gravity Wave

- Experiment (DEEPWAVE)." *Journal of the Atmospheric Sciences* 73.10 (2016): 3855-3876.
- Elbakri, Idris A., and Jeffrey A. Fessler. "Statistical image reconstruction for polyenergetic X-ray computed tomography." *IEEE transactions on medical imaging* 21.2 (2002): 89-99.
- Evans, Cynthia A. and Robinson, Julie A. "Space Station Orbit Tutorial" *Gateway To Astronaut Photography Of Earth*, NASA, n.d.,
<https://eol.jsc.nasa.gov/Tools/orbitTutorial.htm#:~:text=Each%20orbit%20takes%2090%2D93,reboosts%20adjust%20the%20ISS%20orbit.>
- Feng, Yuemeng, et al. "Total variation and point spread function priors for MLEM reconstruction in Compton camera imaging." *2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC)*. IEEE, 2018.
- Fessler, Jeffrey A., and Anastasia Yendiki. "Channelized Hotelling observer performance for penalized-likelihood image reconstruction." *2002 IEEE Nuclear Science Symposium Conference Record*. Vol. 2. IEEE, 2002.
- Fessler, Jeffrey A., M. Sonka, and J. Michael Fitzpatrick. "Statistical image reconstruction methods for transmission tomography." *Handbook of medical imaging* 2 (2000): 1-70.
- Fotouhi, Javad, et al. "Can real-time RGBD enhance intraoperative cone-beam CT?." *International journal of computer assisted radiology and surgery* 12.7 (2017): 1211-1219.
- Gavrilyeva, G. A., et al. "Ground-based spectrographs for measurement of rotational temperature OH (3-1) installed at a meridional network in Yakutia." *24th International Symposium on Atmospheric and Ocean Optics: Atmospheric Physics*. Vol. 10833. International Society for Optics and Photonics, 2018.
- Geyer, Lucas L., et al. "State of the art: iterative CT reconstruction techniques." *Radiology* 276.2 (2015): 339-357.
- Gilbert, Ethel S. "Ionising radiation and cancer risks: what have we learned from epidemiology?." *International journal of radiation biology* 85.6 (2009): 467-482.

- “GPU Market Size, Share & Forecast by 2027 : Graphics Processing Unit.” Allied Market Research, June 2020, www.alliedmarketresearch.com/graphic-processing-unit-market.
- Hart II, Vern Philip. "The application of tomographic reconstruction techniques to ill-conditioned inverse problems in atmospheric science and biomedical imaging." (2012).
- Hart, Vern P., et al. "Three-dimensional tomographic reconstruction of mesospheric airglow structures using two-station ground-based image measurements." *Applied optics* 51.7 (2012): 963-974.
- Huete, Alfredo. (2004). Remote Sensing for Environmental Monitoring. 10.1016/B978-012064477-3/50013-8.
- Islam, Fahima Fahmida. "Insufficient CT data reconstruction based on directional total variation (DTV) regularized maximum likelihood expectation maximization (MLEM) method." (2013).
- Jacobs, Filip, et al. "A fast algorithm to calculate the exact radiological path through a pixel or voxel space." *Journal of computing and information technology* 6.1 (1998): 89-94.
- Janesick, James R. . *Photon Transfer*. Bellingham, WA USA, SPIE, 2007
- Jelveh, Salomeh, et al. "Investigations of antioxidant-mediated protection and mitigation of radiation-induced DNA damage and lipid peroxidation in murine skin." *International journal of radiation biology* 89.8 (2013): 618-627.
- Jiang, Qinghu, et al. "Estimating soil organic carbon of cropland soil at different levels of soil moisture using VIS-NIR spectroscopy." *Remote Sensing* 8.9 (2016): 755.
- Joseph, P. M. "An improved algorithm for reprojecting rays through pixel images." *Journal of Computer Assisted Tomography* 7.6 (1983): 1136.
- Kalpana, K. B., et al. "Protection against X-ray radiation-induced cellular damage of human peripheral blood lymphocytes by an aminothiazole derivative of dendrodoine." *Chemico-biological interactions* 186.3 (2010): 267-274.
- Kaushal, Hemani, V. K. Jain, and Subrat Kar. "Free-space optical channel models." *Free space optical communication*. Springer, New Delhi, 2017. 41-89.

- King, Michael D., et al. "Spatial and temporal distribution of clouds observed by MODIS onboard the Terra and Aqua satellites." *IEEE transactions on geoscience and remote sensing* 51.7 (2013): 3826-3852.
- Kuefner, M. A., et al. "Radiation induced DNA double-strand breaks in radiology." *RöFo-Fortschritte auf dem Gebiet der Röntgenstrahlen und der bildgebenden Verfahren*. Vol. 187. No. 10. © Georg Thieme Verlag KG, 2015.
- Lai, Chang, et al. "Automatic extraction of gravity waves from all-sky airglow image based on machine learning." *Remote Sensing* 11.13 (2019): 1516.
- Le Goff, T., et al. "ROIC glow reduction in very low flux short wave infra-red focal plane arrays for astronomy." *X-Ray, Optical, and Infrared Detectors for Astronomy IX*. Vol. 11454. International Society for Optics and Photonics, 2020.
- Libretexts. "13.2: Rotations Accompany Vibrational Transitions." Chemistry LibreTexts, 17 Sept. 2020, [chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Map%3A_Physical_Chemistry_\(McQuarrie_and_Simon\)/13%3A_Molecular_Spectroscopy/13.02%3A_Rotations_Accompany_Vibrational_Transitions](https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Map%3A_Physical_Chemistry_(McQuarrie_and_Simon)/13%3A_Molecular_Spectroscopy/13.02%3A_Rotations_Accompany_Vibrational_Transitions).
- Li, Yusheng. "Noise propagation for iterative penalized-likelihood image reconstruction based on Fisher information." *Physics in Medicine & Biology* 56.4 (2011): 1083.
- Liew, Soo Chin, et al. "Noise propagation in SPECT images reconstructed using an iterative maximum-likelihood algorithm." *Physics in Medicine & Biology* 38.12 (1993): 1713.
- Liu, Rui, et al. "GPU-based branchless distance-driven projection and backprojection." *IEEE transactions on computational imaging* 3.4 (2017): 617-632.
- Liu, Weijun, et al. "Comparison of rotational temperature derived from ground-based OH airglow observations with TIMED/SABER to evaluate the Einstein coefficients." *Journal of Geophysical Research: Space Physics* 120.11 (2015): 10069-10082.
- Long, Yong, Jeffrey A. Fessler, and James M. Balter. "3D forward and back-projection for X-ray CT using separable footprints." *IEEE transactions on medical imaging* 29.11 (2010): 1839-1850.

- Luo, Fulin, et al. "Adaptive weighted total variation minimization based alternating direction method of multipliers for limited angle CT reconstruction." *IEEE Access* 6 (2018): 64225-64236.
- Makeev, Andrey, et al. "Investigation of optimal parameters for penalized maximum-likelihood reconstruction applied to iodinated contrast-enhanced breast CT." *Medical Imaging 2016: Physics of Medical Imaging*. Vol. 9783. International Society for Optics and Photonics, 2016.
- Makhlouf, UoB, R. H. Picard, and J. R. Winick. "Photochemical-dynamical modeling of the measured response of airglow to gravity waves: 1. Basic model for OH airglow." *Journal of Geophysical Research: Atmospheres* 100.D6 (1995): 11289-11311.
- Marsh, Daniel R., et al. "SABER observations of the OH Meinel airglow variability near the mesopause." *Journal of Geophysical Research: Space Physics* 111.A10 (2006).
- Matenine, Dmitri, et al. "System matrix computation vs storage on GPU: A comparative study in cone beam CT." *Medical physics* 45.2 (2018): 579-588.
- Matsuda, Takashi S., et al. "New statistical analysis of the horizontal phase velocity distribution of gravity waves observed by airglow imaging." *Journal of Geophysical Research: Atmospheres* 119.16 (2014): 9707-9718.
- McGaffin, Madison G., and Jeffrey A. Fessler. "Alternating dual updates algorithm for X-ray CT reconstruction on the GPU." *IEEE transactions on computational imaging* 1.3 (2015): 186-199.
- Miller, Steven D., et al. "Upper atmospheric gravity wave details revealed in nightglow satellite imagery." *Proceedings of the National Academy of Sciences* 112.49 (2015): E6728-E6735.
- Mohammad, Mohd Khairul Amran, et al. "Watermelon (*Citrullus lanatus* (Thunb.) Matsum. and Nakai) juice modulates oxidative damage induced by low dose X-ray in mice." *BioMed research international* 2014 (2014).
- Nam, Haewon, et al. "Tensor framelet based iterative image reconstruction algorithm for low-dose multislice helical CT." *PloS one* 14.1 (2019): e0210410.

- Nappo, Carmen J. *An introduction to atmospheric gravity waves*. Academic press, 2013.
- National Aeronautics and Space Administration. "Reference Guide To The International Space Station." 2010. PDF File.
- Ni, Jun, et al. "Analysis of performance evaluation of parallel katsevich algorithm for 3-d ct image reconstruction." *First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)*. Vol. 1. IEEE, 2006.
- Nishimura, Yoshikazu, et al. "Radioprotection of mice by lactoferrin against irradiation with sublethal X-rays." *Journal of radiation research* 55.2 (2014): 277-282.
- Oberheide, J., et al. "Intercomparison of kinetic temperature from 15 μm CO₂ limb emissions and OH*(3, 1) rotational temperature in nearly coincident air masses: SABER, GRIPS." *Geophysical research letters* 33.14 (2006).
- Oliva, E., and L. Origlia. "The OH airglow spectrum-a calibration source for infrared spectrometers." *Astronomy and Astrophysics* 254 (1992): 466.
- Otani, Tomoaki, et al. "Evaluation and optimization of a new PET reconstruction algorithm, Bayesian penalized likelihood reconstruction, for lung cancer assessment according to lesion size." *American Journal of Roentgenology* 213.2 (2019): W50-W56.
- Panin, V. Y., G. L. Zeng, and G. T. Gullberg. "Total variation regulated EM algorithm [SPECT reconstruction]." *IEEE Transactions on Nuclear Science* 46.6 (1999): 2202-2210.
- Park, Sung-Hwan, Moun-Jin Lee, and Hyung-Sup Jung. "Analysis on the snow cover variations at Mt. Kilimanjaro using Landsat satellite images." *Korean Journal of Remote Sensing* 28.4 (2012): 409-420.
- Pautet, P-D., et al. "Advanced mesospheric temperature mapper for high-latitude airglow studies." *Applied optics* 53.26 (2014): 5934-5943.
- Pei, H., et al. "GANRA-5 protects both cultured cells and mice from various radiation types by functioning as a free radical scavenger." *Free radical research* 48.6 (2014): 670-678.
- Potter, Sean. "NASA Selects Mission to Study Space Weather from Space Station." NASA, 17 Mar. 2020, www.nasa.gov/press-release/nasa-selects-mission-to-study-space-weather-from-space-station.

- Prasad, K. N. "Rationale for using multiple antioxidants in protecting humans against low doses of ionizing radiation." *The British journal of radiology* 78.930 (2005): 485-492.
- Qi, Jinyi. "A unified noise analysis for iterative image estimation." *Physics in Medicine & Biology* 48.21 (2003): 3505.
- Ramsay, S. K., C. M. Mountain, and T. R. Geballe. "Non-thermal emission in the atmosphere above Mauna Kea." *Monthly Notices of the Royal Astronomical Society* 259.4 (1992): 751-760.
- Rezaei, Ahmadreza, et al. "Simultaneous reconstruction of the activity image and registration of the CT image in TOF-PET." *Physics in Medicine & Biology* 61.4 (2016): 1852.
- Rolland, J. P., and Harrison H. Barrett. "Effect of random background inhomogeneity on observer detection performance." *JOSA A* 9.5 (1992): 649-658.
- Rolland, Jannick Paule Yvette. "Factors influencing lesion detection in medical imaging." (1990).
- Sabne, Amit, et al. "Model-based iterative CT image reconstruction on GPUs." *ACM SIGPLAN Notices* 52.8 (2017): 207-220.
- Sánchez, Adrian A. "Estimation of noise properties for TV-regularized image reconstruction in computed tomography." *Physics in Medicine & Biology* 60.18 (2015): 7007.
- Shepp, Lawrence A., and Yehuda Vardi. "Maximum likelihood reconstruction for emission tomography." *IEEE transactions on medical imaging* 1.2 (1982): 113-122.
- Siddon, Robert L. "Fast calculation of the exact radiological path for a three-dimensional CT array." *Medical physics* 12.2 (1985): 252-255.
- Sidky, Emil Y., and Xiaochuan Pan. "Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization." *Physics in Medicine & Biology* 53.17 (2008): 4777.
- Smith, Tyler A., et al. "Radioprotective agents to prevent cellular damage due to ionizing radiation." *Journal of Translational Medicine* 15.1 (2017): 1-18.
- Song, Rui, et al. "Tomographic reconstruction of atmospheric gravity wave parameters from airglow observations." *Atmospheric Measurement Techniques* 10.12 (2017): 4601-4612.

- Stehli, Julia, et al. "Antioxidants Prevent DNA Double-Strand Breaks From X-Ray Based Cardiac Examinations: A Randomized, Double-Blinded, Placebo-Controlled Trial." *Journal of the American College of Cardiology* 64.1 (2014): 117-118.
- Sun, Bing-Yu, and Yoshihiko Hayakawa. "Impact of statistical reconstruction and compressed sensing algorithms on projection data elimination during X-ray CT image reconstruction." *Oral radiology* 34.3 (2018): 237-244.
- Taylor, Michael J., et al. "Development of an advanced mesospheric temperature mapper (AMTM) for high-latitude research." *38th COSPAR Scientific Assembly* 38 (2010): 6.
- Thompson, William M., and William RB Lionheart. "GPU Accelerated Structure-Exploiting Matched Forward and Back Projection for Algebraic Iterative Cone Beam CT Reconstruction." *The Third International Conference on Image Formation in X-Ray Computed Tomography, 22-25 June 2014, Salt Lake City, Utah, USA..* 2014.
- Tian, Jia, and William D. Philpot. "Relationship between surface soil water content, evaporation rate, and water absorption band depths in SWIR reflectance spectra." *Remote Sensing of Environment* 169 (2015): 280-289.
- Tseng, Hsin-Wu, Jiahua Fan, and Matthew A. Kupinski. "Design of a practical model-observer-based image quality assessment method for x-ray computed tomography imaging systems." *Journal of Medical Imaging* 3.3 (2016): 035503.
- Tsuda, Toshitaka. "Characteristics of atmospheric gravity waves observed using the MU (Middle and Upper atmosphere) radar and GPS (Global Positioning System) radio occultation." *Proceedings of the Japan Academy, Series B* 90.1 (2014): 12-27.
- Tsui, Benjamin MW, et al. "Comparison between ML-EM and WLS-CG algorithms for SPECT image reconstruction." *IEEE Transactions on Nuclear Science* 38.6 (1991): 1766-1772.
- Tuy, Heang K. "An inversion formula for cone-beam reconstruction." *SIAM Journal on Applied Mathematics* 43.3 (1983): 546-552.
- Van Slambrouck, Katrien, and Johan Nuyts. "Reconstruction scheme for accelerated maximum likelihood reconstruction: The patchwork structure." *IEEE Transactions on Nuclear Science* 61.1 (2014): 173-181.

- Van Slambrouck, Katrien, et al. "Bias reduction for low-statistics PET: maximum likelihood reconstruction with a modified Poisson distribution." *IEEE Transactions on medical imaging* 34.1 (2014): 126-136.
- Vaniqui, Ana, et al. "The effect of different image reconstruction techniques on pre-clinical quantitative imaging and dual-energy CT." *The British journal of radiology* 92.1095 (2019): 20180447.
- Vazquez, C., et al. "Parallelization of MLEM algorithm for PET reconstruction based on GPUs." *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. IEEE, 2014.
- Velauthapillai, Nivethan, et al. "Antioxidants taken orally prior to diagnostic radiation exposure can prevent DNA injury." *Journal of Vascular and Interventional Radiology* 28.3 (2017): 406-411.
- Wang, Cong, et al. "A snow-free vegetation index for improved monitoring of vegetation spring green-up date in deciduous ecosystems." *Remote sensing of environment* 196 (2017): 1-12.
- Wang, Wenying, et al. "Predicting image properties in penalized-likelihood reconstructions of flat-panel CBCT." *Medical physics* 46.1 (2019): 65-80.
- Wang, Yingmei, et al. "A framelet-based iterative maximum-likelihood reconstruction algorithm for spectral CT." *Inverse problems* 32.11 (2016): 115021.
- White, Marvin H., et al. "Characterization of surface channel CCD image arrays at low light levels." *IEEE Journal of Solid-State Circuits* 9.1 (1974): 1-12.
- Wilson, Donald W., Benjamin MW Tsui, and Harrison H. Barrett. "Noise properties of the EM algorithm. II. Monte Carlo simulations." *Physics in Medicine & Biology* 39.5 (1994): 847.
- Wu, Dufan, Kyungsang Kim, and Quanzheng Li. "Computationally efficient deep neural network for computed tomography image reconstruction." *Medical physics* 46.11 (2019): 4763-4776.

- Xie, Xiaobin, et al. "Accelerating separable footprint (SF) forward and back projection on GPU." *Medical Imaging 2017: Physics of Medical Imaging*. Vol. 10132. International Society for Optics and Photonics, 2017.
- Yan, Hao, et al. "Towards the clinical implementation of iterative low-dose cone-beam CT reconstruction in image-guided radiation therapy: Cone/ring artifact correction and multiple GPU implementation." *Medical physics* 41.11 (2014): 111912.
- Yao, Jie, and Harrison H. Barrett. "Predicting human performance by a channelized Hotelling observer model." *Mathematical Methods in Medical Imaging*. Vol. 1768. International Society for Optics and Photonics, 1992.
- Zeng, Gengsheng Lawrence. *Medical image reconstruction: a conceptual tutorial*. New York: Springer, 2010.
- Zhang, Hao, et al. "Regularization strategies in statistical image reconstruction of low-dose x-ray CT: A review." *Medical physics* 45.10 (2018): e886-e907.
- Zhang, Lin, et al. "Resolution and noise performance of sparse view X-ray CT reconstruction via Lp-norm regularization." *Physica Medica* 52 (2018): 72-80.
- Zhao, Yucheng, et al. "Investigating an unusually large 28-day oscillation in mesospheric temperature over Antarctica using ground-based and satellite measurements." *Journal of Geophysical Research: Atmospheres* 124.15 (2019): 8576-8593.
- Zheng, Jiabei, Jeffrey A. Fessler, and Heang-Ping Chan. "Segmented separable footprint projector for digital breast tomosynthesis and its application for subpixel reconstruction." *Medical physics* 44.3 (2017): 986-1001.