RESOLUTION ENHANCEMENT OF SEASAT SCATTEROMETER DATA

A Thesis
Submitted to the
Department of Electrical and Computer Engineering
Brigham Young University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Peter T. Whiting
April 1992

This thesis by Peter T. Whiting is accepted in its present form by the Department of Electrical and Computer Engineering of Brigham Young University as satisfying the thesis requirement for the degree of Master of Science.

David G. Long
Committee Chairman

Brian D. Jeffs
Committee Member

Richard H. Selfridge
Graduate Coordinator

Date

ACKNOWLEDGMENTS

I am grateful to Dr. David G. Long for the many hours he has spent assisting me in the development of the imaging techniques described in this paper. Also, the many revisional changes he has suggested for this and other related papers have been invaluable. In addition, I acknowledge Gary Skouson for his work in preparing the data, as well as his help in the implementation and development of many of the reconstruction algorithms.

Finally, I thank my wife, Natalie Whiting, for her support in my schooling and specifically for her help in editing this paper.

The data for this research was obtained from the Physical Oceanography DAAC at Jet Propulsion Laboratory/California Institute of Technology.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

On July 7th, 1978, NASA launched Seasat, an experimental satellite carrying a scatterometer (SASS). SASS was designed to measure ocean wind vectors, a job it did quite well. Previous ocean wind vector measurement techniques relied on ships and buoys. The inherent resolution of such data collection is quite low, as there are only a few measurements for every hundred thousand square kilometers. SASS, on the other hand, provided global coverage with a footprint of around fifty kilometers. Although SASS was designed for ocean wind vector measurement, it continued to record data as Seasat passed over land. It was thought that the land data could be used for the calibration of the instrument, as well as for other scientific studies. On September 9, 1978, only three months after launch, the Seasat mission ended due to a power systems failure.

In 1989 Kennett and Li investigated of the nature of the land data and found a good correlation between the SASS measurements and land cover type, allowing them to generate low resolution ($0.5^o \times 0.5^o$) land cover images. While the $0.5^o \times 0.5^o$ SASS resolution is acceptable for ocean studies, the low resolution is inadequate for detailed land cover imaging. However, the image resolution need not be restricted to the satellite footprint size, since SASS collected multiple overlapping measurements whose intersections theoretically define the maximum possible resolution. Viewing each measurement as a projection onto the surface of the earth the image reconstruction process is essentially a problem of image reconstruction from projections.

Traditional methods of image reconstruction from projections (such as Algebraic Reconstruction Technique [ART], Simultaneous Iterative Reconstruction Technique [SIRT], and Maximum Entropy [ME]) are difficult to apply effectively to the SASS data set for two reasons: First, scatterometer data is inherently noisy and most traditional reconstruction methods are relatively noise-intolerant. Second, traditional reconstruction methods are single-variate (per pixel), while the data generated by a scatterometer usually requires a multi-variate reconstruction

technique. The contribution of this research has been to develop a relatively noise-tolerant reconstruction algorithm capable of multi-variate image estimation, called Scatterometer Iterative Reconstruction with Filter (SIRF).

This thesis begins with some basic information in Chapter 2 related to scatterometers and to SASS in particular. Chapter 3 provides an overview of some of the existing methods of image reconstruction from projections. A discussion of the methodology used in the development of the SIRF algorithm follows in Chapter 4. Chapter 5 describes the particularities of the application of SIRF to the SASS data set. Finally, Chapter 6 compares of the performances of the different image reconstruction algorithms discussed in this paper to the SIRF algorithm.

CHAPTER 2

SCATTEROMETER BASICS

The basic function of a spaceborne radar scatterometer is to transmit electromagnetic pulses toward the earth and measure the backscattered power. The normalized radar cross-section (σ^o) is a function of this returned power, defined by the basic radar equation [2]

$$\sigma^o_{ns} = \frac{(4\pi)^3 R^4 L}{P_t G^2 \lambda^2 A} P_s \tag{2.1}$$

where R is the distance to the surface, L is the system losses, P_t is the transmitted power, G is the antenna gain, λ is the wavelength of the transmitted wave, A is the effective area of illumination, and P_s is the power received. Superimposed on P_s is a noise contribution due to unmodeled noise sources. The subscript ns on σ^o represents normal space. Throughout this thesis σ^o will be assumed to be expressed in log space,

$$\sigma^o = 10 \log \sigma^o_{ns}. \tag{2.2}$$

σ^o has been shown to be dependent on surface parameters, allowing scatterometers to collect data suitable for surface imaging [3]. As with other microwave imaging systems, a scatterometer enjoys the capability of twenty-four-hour, all-weather data collection.

2.1 The Seasat Scatterometer (SASS)

SASS used two side-looking fanbeam antennas each having two different azimuth angles to make /sigmao measurements, thus yielding a total of four different observations from a single point. Each antenna's instantaneous footprint was several hundred kilometers long and around ten kilometers wide (see fig 2.1), the along-track resolution being obtained by timing. Each antenna's footprint was Doppler filtered to segment the returned echo into twelve cross-track (along-beam) resolution elements (measurement cells) having an average width of 50 kilometers.

The resulting measurement cells were six-sided polygons (see figure 2.2) with dimesions being a function of the Doppler filter bandwidth, the antenna beamwidth at the earth's surface, the transmit pulse timing, the spacecraft ground track velocity, and the cell latitude.

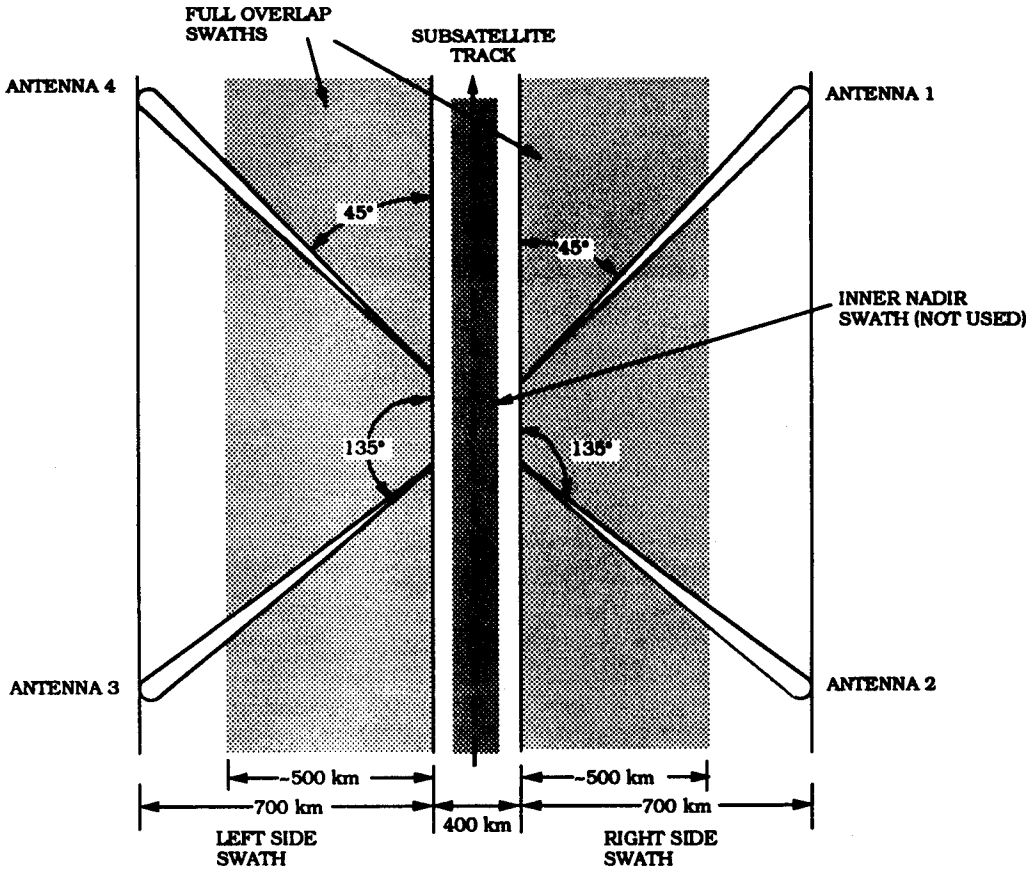

Figure 2.1: SASS antenna illumination pattern

For the first half of Seasat's mission the orbit precession shifted the cell measurement pattern longitudinally providing coverage of the entire earth's surface. For the last half of the mission the satellite went into an exact repeat orbit, causing some areas of the earth to have coverage that was limited and only existed at high angles of incidence. (This information is discussed in greated depth in [4].)

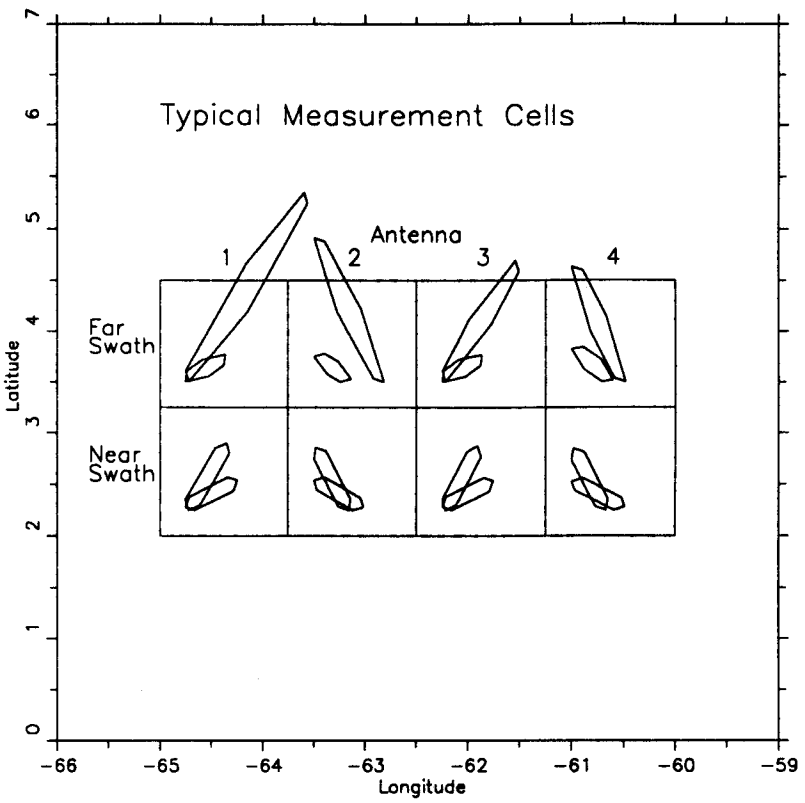

Figure 2.2: Typical SASS cell geometries

2.2 Dependence of σ^o on Incidence Angle

Spaceborne scatterometers usually measure σ^o over a wide range of incidence angles (ϕ - usually a known variable), allowing for increased spatial coverage. However, σ^o is dependant upon the angle of incidence, and therefore the cost for the increased spatial coverage is the postprocessing required to normalize all measurements to a common angle of incidence.

The model most commonly used to relate σ^o to the angle of incidence (θ) is $\sigma^o = \mathcal{A} + \mathcal{B}\theta$ [5, 3, 6, 7]. Using this model, the σ^o measurements can be normalized to a common angle, ϕ, by Eq. (2.3).

$$\sigma^o = \mathcal{A} + \mathcal{B}(\phi - \theta) \tag{2.3}$$

$\mathcal{A}$ represents the incidence angle-normalized backscatter coefficient, while $\mathcal{B}$ represents the dependence of σ^o on the angle of incidence or, in other words, the slope of σ^o versus θ. ϕ is the angle to which σ^o is being normalized. For the SASS data set this angle was selected to be the approximate mean of the angles of incidence of the entire data set, 40°. By selecting ϕ to be the mean of the θs, the contribution of $\phi - \theta$ is minimized, thus minimizing the effect of the estimate of $\mathcal{B}$ on the estimate of $\mathcal{A}$. Normalizing to a ϕ other than 0° superimposes some of the $\mathcal{B}$ image on the $\mathcal{A}$ image. Eq. (2.3) can be rewritten as

$$\sigma^o = (\mathcal{A} + \mathcal{B}40^o) - \mathcal{B}\theta = \mathcal{A}' - \mathcal{B}\theta \tag{2.4}$$

where $\mathcal{A}'$ represents the true y-intercept. This superposition of the $\mathcal{B}$ image on the $\mathcal{A}$ image may increase the discrimination between two land covers, while it might also increase the noise of the $\mathcal{A}$ image. Figure 2.3 shows the σ^o values for two regions of the Amazon rain forest as measured by SASS. Notice that the discrimination between the two land cover types increases slightly at larger incidence angles. However, if vegetation type 2 had a smaller $\mathcal{B}$ than type 1, the discrimination at larger incidence angles might be worse.

2.2.1 Impact of Errors in $\mathcal{B}$ on $\mathcal{A}$ as Related to SASS

Based on the SASS data set, $\mathcal{B}$ values are usually around -0.13 with a minimum of approximately -0.2 and a maximum near -0.07. Therefore, the

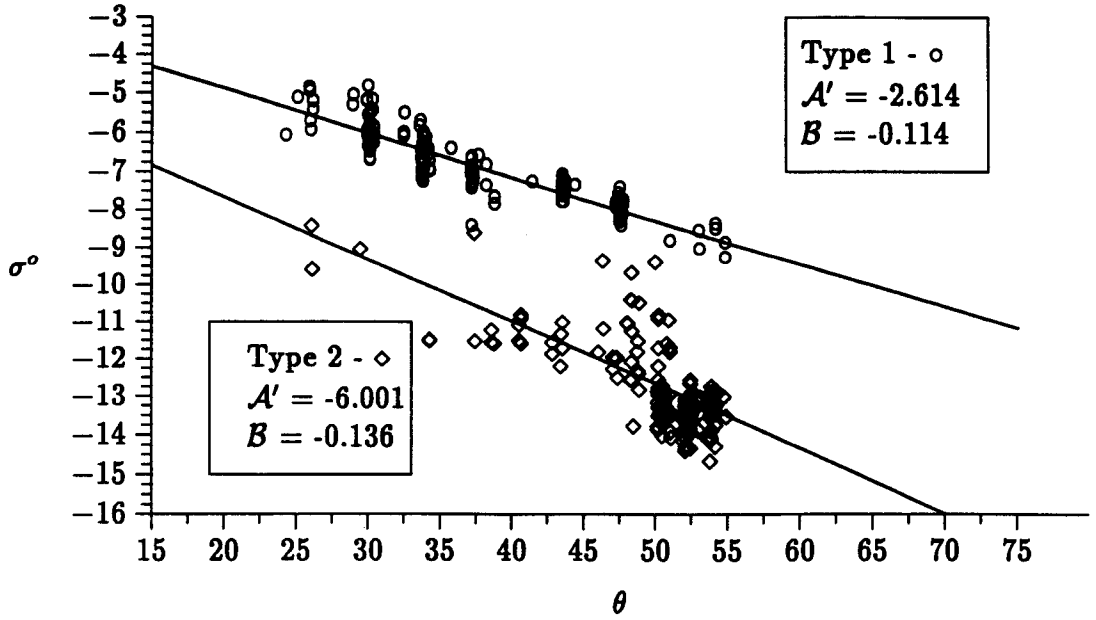

Figure 2.3: σ^o vs. θ for two different surfaces in the linear region

maximum error in a $\mathcal{B}$ image, assuming a constant $\mathcal{B}$ of -0.13, is about 0.07. The maximum magnitude of $40^o - \theta$ is approximately 17^o. (The θ thresholds for the data set are 57^o and 23^o as will be explained later.) Therefore, the maximum error introduced into the $\mathcal{A}$ estimate caused by assuming a constant $\mathcal{B}$ of -0.13 is 0.07 $\times$ 17 = 1.19 dB. On the average this error will be much smaller.

From the above discussion it should be clear that a small error in the estimate of $\mathcal{B}$ has only a small impact on the estimate of $\mathcal{A}$.

2.3 Invalid Measurement Identification/Elimination

While the scatterometer measurements are inherently noisy, some of SASS's σ^o measurements appear excessively noisy. Since excessive noise corrupts the generated images, it is desirable to eliminate as many of these excessively noisy measurements as possible.

The noise can be assumed to be either localized in time (e.g., noise caused by incorrect satellite altitude or extreme weather conditions) or random in time (e.g., bit errors). By plotting σ^o versus revolution number (time) over regions with small expected variation in σ^o possible bad measurements can be identified since measurement values localized in time and space should have similar σ^o values.

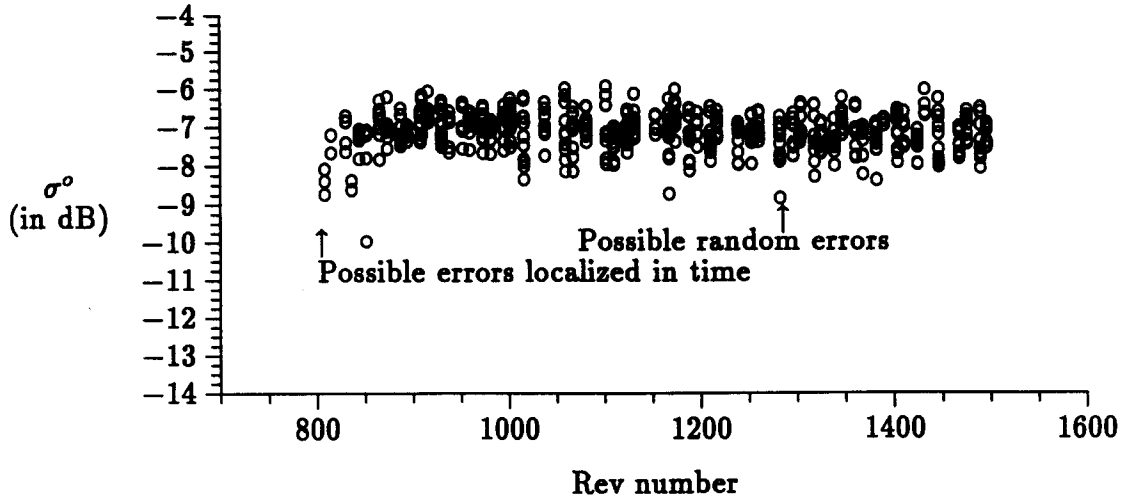

Figure 2.4: σ^o vs. orbit number

Figure 2.4 is a plot of some measurements hitting a relatively uniform region of the Amazon tropical forest. (Each measurement was normalized to 40^o as explained previously.)

There is a band of values of σ^o in which measurements are expected to fall. The variance within this band is caused by different surface parameters and small amounts of noise. The measurements falling outside of this band are assumed to have been corrupted by large amounts of noise. When a given orbit has a large number of bad measurements, the noise source is assumed to be localized in time, and no measurements from that orbit are used in the data processing.

The data blocks received from Seasat included the variable Kp which is the predicted normalized standard deviation of the communication component of the noise. This measure was used to aid in the identification of noise which was random in time. The value of Kp was checked against a threshold to determine if the measurement would be used. Most image reconstruction algorithms are noise-intolerant to some degree. Therefore, one is tempted to select a very low threshold for Kp. However, the benefits gained by eliminating a large number of measurements with noise are sometimes outweighed by the lack of resolution associated with too few measurements. In general, better than 95% of the measurements from SASS data set had Kp values less than 15%. For this reason 15% was chosen as the threshold for Kp.

Unfortunately, Kp does not always give an accurate estimate of the amount of noise contained in a measurement. Therefore, the processing algorithms had to be capable of tolerating some measurements with noise levels higher than 15%.

2.3.1 Bad Measurements Due to Extreme Values of θ

It was noted after comparing many σ^o versus θ plots that σ^o measurements at either end of the incidence angle spectrum did not follow the linear model. Other researchers have made this same observation [5]. For this reason only measurements having θ greater than 23° and less than 57° were used. Figure 2.5 shows a plot of σ^o versus θ.

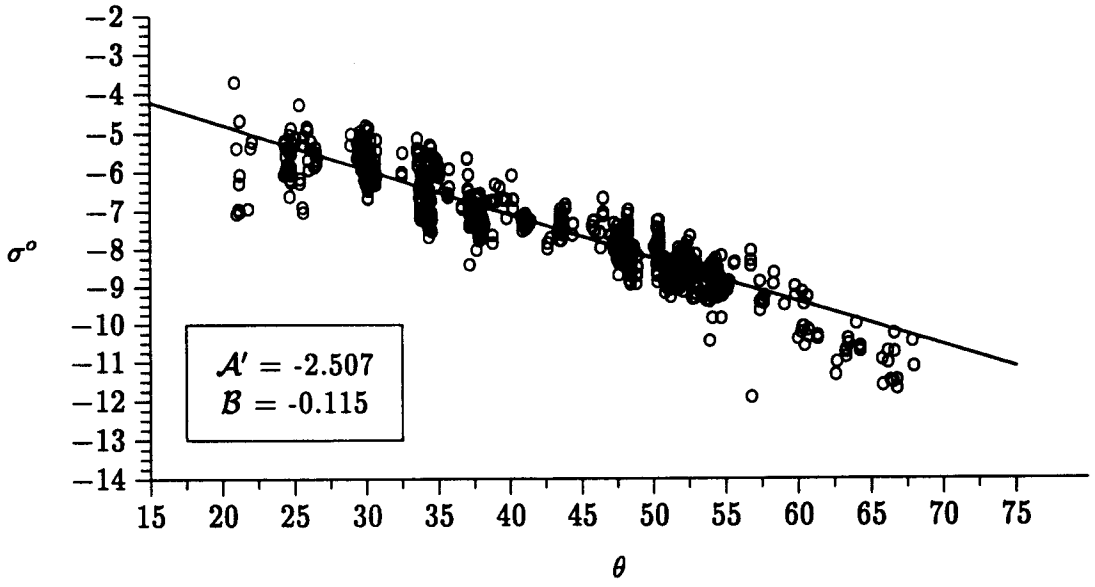

Figure 2.5: σ^o vs. θ

2.3.2 Recalibration

By observing σ^o over a region expected to have uniform σ^o values, it became apparent that some form of recalibration as a function of antenna number might be needed, as different antennas antennas produced different average values for the same regions. Also, the existence of regular noise patterns which followed the satallite track in large images suggests some form of recalibration as a function

of time might be warrented. As of this writing the details for recalibration are yet undecided.

2.4 Summary

This chapter discussed some of the basics of scatterometry and also explained some of the details essential to image reconstruction using the SASS data set. In short, SASS provided a very noisy data set, containing measurements having a wide range of incidence angles. The methodology for identifying and eliminating measurements containing excessive noise relies upon the assumption that noise is usually localized in time or, if not, is identifiable by SASS's variable Kp. The excessively noisy measurements which are not identified by these two methods must be tolerated until other forms of bad measurement identification are developed. The method for normalizing measurements taken over a wide range of incidence angles assumes a linear model in log space for σ^o versus θ. The slope of the line is an additional unknown which must be estimated by the image processing algorithm. The linear model, although not perfect over the entire range, is adequate for an acceptable range of incidence angles ($23^o - 57^o$). Measurements falling outside this range are not used. These two aspects of the SASS data set, noise and incidence angle dependence, make the application of traditional image reconstruction techniques to this data set difficult.

CHAPTER 3

IMAGE RECONSTRUCTION FROM PROJECTIONS

This chapter presents a brief overview of some of the existing techniques for image reconstruction from projections. The goal of such techniques is to solve for $\mathcal{A}$ the equation $\mathcal{S} = \mathcal{H}\mathcal{A}$ where $\mathcal{S}$ is the measurement vector, length N, $\mathcal{H}$ is the point spread matrix $(N \times M)$ for the measurement vector $\mathcal{S}$, and $\mathcal{A}$ is a vector of length M representing the image. Actual data sets usually require the solving of the equation $\mathcal{S} = \mathcal{H}\mathcal{A} + \mathcal{V}$ where $\mathcal{V}$ is the noise contribution. Additionally, due to the fact that the measurements are incident angle dependant, the added variable $\mathcal{B}$ must also be estimated for each pixel, ie. $\mathcal{S} = \mathcal{H}(\mathcal{A} + \mathcal{B}(\phi - \theta)) + \mathcal{V}$, where ϕ is a constant and each measurement in $\mathcal{S}$ has a θ associated with it. In order to accomidate existing reconstruction techniques it will initially be assumed $\mathcal{B}$ is known. These two assumptions will allow existing image reconstruction from projection techniques to be applied to the SASS data set.

3.1 AVE - The Averaging Algorithm

An intuitive processing approach to the problem of combining multiple overlapping measurements (image reconstruction from projections) is to assign each pixel the value of the average of all measurements hitting it, a method which will be referred to as the AVE algorithm. (The AVE algorithm using pixel sizes larger than the measurement cell sizes where measurements are "binned" and then averaged is referred to as LoRes, which is the traditional method of land image generation using scatterometer data.) An AVE pixel update is simply the back projection, which can be expressed as

$$a_j = \frac{\sum_{i=1}^{N} s_i h_{ij}}{\sum_{i=1}^{N} h_{ij}} \tag{3.1}$$

where a_j is the j^{th} pixel in $\mathcal{A}$, s_i is the i^{th} measurement, and h_{ij} is the element of the point spread matrix corresponding to the i^{th} measurement and the j^{th} pixel.

For this paper the "point spread matrix," $\mathcal{H}$, is assumed to contain only ones and zeros. If a given pixel j is to be affected by the i^{th} measurement, h_{ij} is a one. If the pixel is not the be affected, h_{ij} is a zero. This particular form of the point spread matrix changes the representation of the back projection of a measurement to an average instead of a sum. Using a conventional point spread matrix, a forward projection, p_i, corresponding to the i^{th} measurement is simply

$$p_i = \sum_{n=1}^{M} h_{in} a_n. \tag{3.2}$$

However, with the point spread matrix used here, the forward projection corresponding to the i^{th} measurement is

$$p_i = \frac{\sum_{n=1}^{M} h_{in} a_n}{\sum_{n=1}^{M} h_{in}}. \tag{3.3}$$

AVE is an intuitive solution to the problem of combining multiple measurements, but it is limited in its ability to resolve higher resolution features. Consider the following example.

Assume a hypothetical system which measures the height of trees. It has a resolution of two trees per measurement, that is, when it measures a height the resulting measurement is the average height of two adjacent trees. The goal is to be able to obtain the individual tree heights from a data set of four measurements.

The system passes over a row of trees having heights 10, 2, 3, 8, and 1. (These heights were selected at random, and other choices would yield a variety of results.) In this example the $\mathcal{H}$ matrix has the form

$$\mathcal{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

The corresponding $\mathcal{S}$ matrix is

$$\mathcal{S} = [6.0 \quad 2.5 \quad 5.5 \quad 4.5]^T.$$

The AVE algorithm yields tree heights of

	tree 1	tree 2	tree 3	tree 4	tree 5
s_1	6.00	6.00	0	0	0
s_2	0	2.50	2.50	0	0
s_3	0	0	5.50	5.50	0
s_4	0	0	0	4.50	4.50
Average	6.00	4.25	4.00	5.00	4.50
Actual	10.00	2.00	3.00	8.00	1.00

Notice the poor reconstruction of the original "image." Boundaries where pixel values change greatly tend to be blurred. (The sharpness of the boundaries defines the resolution of an image.) However, in general, if there are a large number of adjacent pixels having similar values, AVE does an acceptable job of reconstructing the image in those areas.

3.2 ART - Algebraic Reconstruction Technique

Over the past years much work has been done with various methods to solve the reconstruction from projections problem. One such method, Arithmetic Reconstruction Technique (ART), was proposed in 1971 by Gordon, Bender, and Herman [1]. Like the AVE algorithm, ART has a strong intuitive basis.

For each measurement in $\mathcal{S}$ the forward projection of the current reconstruction is compared to the actual measurement. If there is a discrepancy either a correction factor is added to each pixel covered by the measurement (additive ART) or each pixel covered by the measurement is multiplied by a scale factor (multiplicative ART) causing the forward projection, if immediately regenerated, to match the measurement. Subsequent measurements might further alter the pixels such that the forward projections for previously processed measurements might no longer match their measurements. Over multiple iterations (an iteration will be defined to be the complete cycle of processing of each measurement in $\mathcal{S}$ once), if the corrections approach zero or if the scale factors approach one, the algorithm is said to converge for the data set. (There are many other criteria which can be used to define converge, see [8].) The convergence of ART for data sets containing

no noise has been well documented [1, 9, 10].

The update of the j^{th} pixel due to the i^{th} measurement using additive ART and our convention for $\mathcal{H}$ is given by

$$a_j^{k+1} = a_j^k + \left(s_i - \frac{\sum_{n=1}^{M} h_{in} a_n^k}{\sum_{n=1}^{M} h_{in}} \right) h_{ij} = a_j^k + (s_i - p_i) h_{ij} \tag{3.4}$$

with $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{H}$ defined as above in the AVE algorithm.

The term

$$p_i = \left(\sum_{n=1}^{M} h_{in} a_n^k \right) \Big/ \left(\sum_{n=1}^{M} h_{in} \right) \tag{3.5}$$

is the forward projection due to the i^{th} measurement as defined in Eq. (3.3). The initial pixel values, a_j^0, are assumed to be all set to the same, non-zero value: the average of all the measurements. The difference between the back projection and the actual measurement s_i is multiplied by h_{ij} to determine if the j^{th} pixel is to be affected by the i^{th} measurement. (Again, as stated previously, for the systems considered in this paper $\mathcal{H}$ contains only ones and zeros, and these equations are formulated with that fact in mind.) Additive ART in the absence of noise has been shown to converge to the minimum L2 norm /citeART.

Similarly, multiplicative ART iterations can be represented using our notation by

$$a_j^{k+1} = a_j^k \left(s_i \frac{\sum_{n=1}^{M} h_{in}}{\sum_{n=1}^{M} h_{in} a_n^k} \right)^{h_{ij}} . \tag{3.6}$$

In multiplicative ART, a pixel update is the result of multiplying the current pixel value by a scale factor. From Eq(3.6), the term in parenthesis can be viewed as a scale factor which is simply the actual measurement divided by the forward projection.

$$scale_i^k = s_i / p_i^k \tag{3.7}$$

The scale factor is raised to the h_{ij} power to determine whether a_j^{k+1} will be affected by the i^{th} pixel. (The scale factor will become one if the measurement is not to affect the pixel a_j^{k+1}.)

Rewriting Eq. (3.6) with Eq. (3.7) yields

$$a_j^{k+1} = a_j^k (scale_i^k)^{h_{ij}}. \tag{3.8}$$

Multiplicitive ART, although quite similar to Additive ART, does not converge to the minimum L2 norm, but the the maximum entropy solution.

3.3 Block ART

The ART algorithm, although good at reconstructing images in the absence of noise, does not tend to converge in the presence of data inconsistencies (which can be assumed to be caused by noise) [10]. Convergence in the presence of small amounts of noise can be improved by using simultaneous iterative reconstruction techniques (SIRT) [11]. The basis of these techniques is similar to ART, but the pixels are updated only after all of the measurements have been processed, using an average of the pixel updates generated by each measurement. We will refer to this variation of the ART algorithm as block ART. It is possible to formulate both additive block ART and multiplicative block ART; however, the only the latter is used in this thesis. It will be referred to herein after as block ART.

Block ART can be expressed as

$$a_j^{k+1} = a_j^k \left(\sum_{i=1}^{N} s_i \left[\frac{\sum_{n=1}^{M} h_{in}}{\sum_{n=1}^{M} a_n^k h_{in}} \right] h_{ij} \right) \Bigg/ \left(\sum_{i=1}^{N} h_{ij} \right). \tag{3.9}$$

As with multiplicative ART, the scale factor due to the i^{th} measurement is given by

$$scale_i^k = \frac{s_i}{p_i^k} \tag{3.10}$$

where p_i is the forward projection corresponding to the i^{th} measurement as defined in Eq. (3.3). Substituting this into Eq. (3.9) yields

$$a_j^{k+1} = a_j^k \frac{\sum_{i=1}^{N} (scale_i^k) h_{ij}}{\sum_{i=1}^{N} h_{ij}}. \tag{3.11}$$

The term $a_j^k(scale_i)$ represents the update on the j^{th} pixel due to the i^{th} measurement,

$$update_{ij}^k = a_j^k(scale_i) = a_j^k \left(\frac{s_i}{p_i^k} \right). \tag{3.12}$$

$update_{ij}^k$ is multiplied by h_{ij} to select whether the j^{th} pixel is to be affected by the i^{th} measurement (this addaptation is introduced to facilitate the convention for $\mathcal{H}$.) (Note that although this is multiplicative block ART, the scale factor is multiplied by h_{ij} instead of being raised to the h_{ij} power as in multiplicative ART. For each iteration using Eq. (3.9), a_j^{k+1} is set to be the average of its updates from each measurement. The denominator, $\sum_{i=1}^N h_{ij}$, gives the number of measurements affecting the j^{th} pixel, or the number of updates which were summed. Rewriting Eq. (3.11) gives

$$a_j^{k+1} = \frac{\sum_{i=1}^{N} update_{ij}^k}{\sum_{i=1}^{N} h_{ij}}. \tag{3.13}$$

Changing Multiplicative ART to do simultaneos updates alters the solution converged to by multiplicative ART, possibly causing a joint minimization.

The basic operation of block ART is as follows. First, an initial value for each pixel is needed. This can be any non-zero value. If the same value is assigned to every pixel, the first iteration ($k = 1$) of block ART is equivalent to the AVE algorithm. If all a_j^0 have the same value, the back projections for each measurement in the first iteration are equal to a_j^0. Therefore the scale factor due to the i^{th} measurement is

$$scale_i^0 = \frac{s_i}{a_j^0}. \tag{3.14}$$

Substituting this into Eq. (3.11) yields

$$a_j^1 = a_j^0 \left(\sum_{i=1}^{N} \left[\frac{s_i}{a_j^0} \right] h_{ij} \right) \Big/ \left(\sum_{i=1}^{N} h_{ij} \right) \tag{3.15}$$

which, by bringing the a_j^0 from outside the summation to cancel with the a_j^0 in the scale factor, reduces to

$$a_j^1 = \frac{\sum_{i=1}^{N} s_i h_{ij}}{\sum_{i=1}^{N} h_{ij}} \tag{3.16}$$

which is equivalent to the AVE algorithm, Eq. (3.1).

To perform an iteration of block ART each measurement's cell boundaries are overlayed onto the initial image estimate and the forward projection is calculated. Returning to the tree example, the back projection would be the average tree height of the image pixels covered by the measurement. The scale factors are calculated as in multiplicative ART, but instead of immediatly updating the image pixel, the update is stored and averaged with all other updates affecting the same pixel after all measurements have been processed. This averaging improves noise tolerance and avoids projection noise [12].

Provided below is an example of the algorithm's operation on the same data set as processed above by the AVE algorithm. For initial conditions set each tree height equal to 1. The scale factors for the first iteration would be

$$scale_1^0 = \frac{6.0}{1.00} = 6.00$$

$$scale_2^0 = \frac{2.5}{1.00} = 2.50$$

$$scale_3^0 = \frac{5.5}{1.00} = 5.50$$

$$scale_4^0 = \frac{4.5}{1.00} = 4.50.$$

Using these scale factors, the updates for the first iteration are

	tree 1	tree 2	tree 3	tree 4	tree 5
s_1	6.00×1.00	6.00×1.00			
s_2		2.50×1.00	2.50×1.00		
s_3			5.50×1.00	5.50×1.00	
s_4				4.50×1.00	4.50×1.00
Average	6.00	4.25	4.00	5.00	4.50
Actual	10	2	3	8	1

which are exactly the same as the AVE algorithm. For the second iteration the back projections are

$$p_1^1 = \frac{6.00 + 4.25}{2} = 5.125$$

$$p_2^1 = \frac{4.25 + 4.00}{2} = 4.125$$

$$p_3^1 = \frac{4.00 + 5.00}{2} = 4.500$$

$$p_4^1 = \frac{5.00 + 4.50}{2} = 4.750.$$

The scale factors are

$$scale_1^1 = \frac{6.0}{5.125} = 1.17$$

$$scale_2^1 = \frac{2.5}{4.125} = 0.61$$

$$scale_3^1 = \frac{5.5}{4.500} = 1.22$$

$$scale_4^1 = \frac{4.5}{4.750} = 0.95.$$

The second iteration yields

	tree 1	tree 2	tree 3	tree 4	tree 5
s_1	1.17×6.00	1.17×4.25			
s_2		0.61×4.25	0.61×4.00		
s_3			1.22×4.00	1.22×5.00	
s_4				0.95×5.00	0.95×4.50
Average	7.02	3.78	3.66	5.42	4.26
Actual	10	2	3	8	1

The results of twenty-five iterations are shown below:

Iteration	tree 1	tree 2	tree 3	tree 4	tree 5
0	4.63	4.63	4.63	4.63	4.63
1	6.00	4.25	4.00	5.00	4.50
2	7.02	3.78	3.66	5.42	2.38
4	8.38	3.04	3.31	6.13	3.65
10	9.72	2.18	3.17	7.10	2.38
25	10.22	1.77	3.29	7.55	1.56
Actual	10	2	3	8	1

A measure of the true error of the image estimate can be obtained by summing the magnitudes of the differences between the image values and the corresponding actual values and then dividing by the number of pixels. (This can only be calculated when using simulated data sets, since in an actual data set the correct image values are unknown.) The above data has an error of 0.35 after twenty-five iterations. Since in most system the actual image isn't known (only in simulated data sets is this information available) another measure of the reconstruction error is generated. This term, referred to as the average reconstruction error is formulated by averaging the differences between the actual measurements and their back projections. The above data has an average reconstruction error of .02 after twenty-five iterations. If the noise contribution is relatively small, usually this error approaches a local minimum.

3.4 Summary

This chapter provided explanations of the basic ART-type algorithms, which were used as a foundations for the SIRF algorithm. Specifically, multiplicitive block ART is formulated, which is the foundation for the SIRF algorithm. Each of the algorithms discussed are capable of image reconstruction in the absence of noise, but when applied to data sets containing noise, all except the AVE algorithm have poor reconstructions. Each also lacks the capability for multi-variate reconstruction, theoretically essential to scatterometer image reconstruction. For these reasons, modifications to the block ART algorithm were developed for use with SASS data.

CHAPTER 4

SIRF DEVELOPMENT

Measurement noise leads to an inconsistent system of equations. It is a well-documented fact that ART does not necessarily converge for such systems [1, 9, 10, 12]. This chapter describes the non-linear constraining function which, when used with block ART, forms the basic Scatterometer Image Reconstruction (SIR) algorithm. The modifications needed to perform a multi-variate estimation with SIR and SIRF (SIR with a modified median Filter) are also described.

4.1 Single-variate SIR Development

When the amount of noise is relatively small, one solution to the instability problem is the introduction of a convergence or relaxation factor, thus constraining the change in the image estimate. As proposed here, the relaxation factor includes in the update a weighted sum of the current pixel value, (a_j^k), and the rescaled pixel value, $(a_j^k * scale_i^k)$, by damping the scale factors.

It is desirable to have the damping of the scale factors be symmetric about 1 in log space, which is to say, the scale factors corresponding to a predamped 2 and 0.5 remain inverses after the damping, as do zero and infinity. This is desirable since the pixels will be multiplied by the scale factors, and the symmetry is such that doubling a pixel's value should be the inverse of halving it, which assures that scale factors above unity will not be favored or disciminated against with respect to scale factors below unity. Raising the scale factor to any power, w, meets the symmetry condition.

A plot in log space of the scale factors versus damped scale factors shows that raising the scale factors to a power simply changes the slope of the line, which remains symmetric about 1. (See Figure 4.1.) Different powers (1.0, 0.75, 0.5, 0.25) were tested as the damping power and square root damping was determined to be the best. (See Chapter 6 for more details regarding the method of determination.)

Damping the scale factor in multiplicative ART by raising it to a fixed power w is equivalent to the maximum entropy algorithm proposed by Elfving in

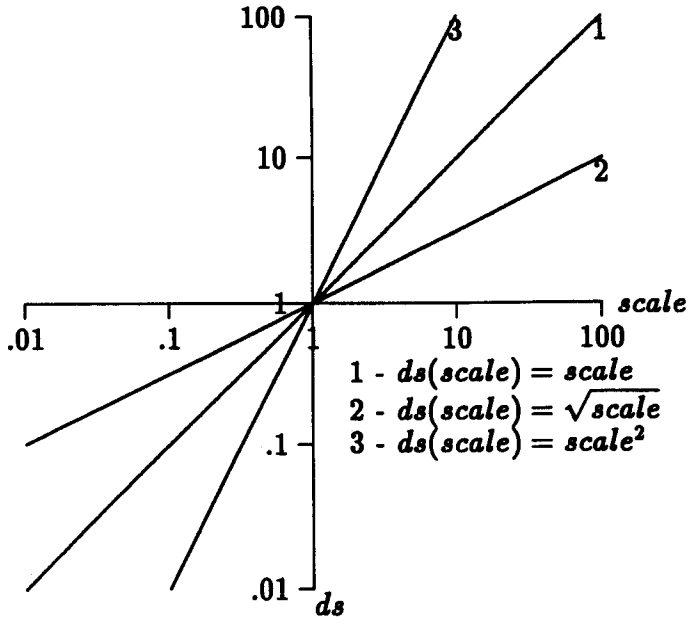

Figure 4.1: Effect of raising the scale factor to a power

[13]. The basic Gauss-Seidel maximum entropy formula can be expressed as [13]

$$a_j^{k+1} = a_j^k exp[wh_{ij}\lambda_i^k] \tag{4.1}$$

where λ_i^k is defined as the natural log of the i^{th} measurement divided by its forward projection. For the SASS data this is simply the natural log of the scale factor defined in Eq. (3.10):

$$\lambda_i^k = ln(scale_i^k). \tag{4.2}$$

Another possible definition of λ_i^k is $\lambda_i^k = s_j - p_i^k$. This is the same as the update term for additive ART.

Using Eq. (4.2), Eq. (4.1) can be rewritten as

$$\begin{aligned} a_j^{k+1} &= a_j^k \exp\left[wh_{ij} ln(scale_i^k)\right] && (4.3)\\ &= a_j^k \exp\left[ln(scale_i^k)^{(wh_{ij})}\right] && (4.4)\\ &= a_j^k (scale_i^k)^{wh_{ij}}. && (4.5) \end{aligned}$$

This is equivalent to multiplicative ART, Eq. (3.8), with the scale factor being

raised to w. Block ART with a damping power w can be expressed as

$$a_j^{k+1} = a_j^k \frac{\sum_{i=1}^{N} (scale_i^k)^w h_{ij}}{\sum_{i=1}^{N} h_{ij}}. \tag{4.6}$$

In the absense of noise the solution converged to by block ART with a damping power is probably simmilar to that converged to by block ART without the damping power, only now the convergence is slower.

This form of damping is effective in reducing the effects of small amounts of noise, but the algorithm may still become unstable after a large number of iterations. Experimentation with the SASS data set showed this damping alone to be an unacceptable solution because of the large amounts of noise in the data.

Other approaches to stabilization used by previous researchers usually involved smoothing the resulting images [10]. However, too much smoothing is undesirable, since it reduces the effective resolution. Some smoothing techniques have been employed to a limited degree in SIRF, but the amount of smoothing needed to stabilize the algorithms for the SASS data set is unacceptable; therefore, an alternate method of stabilization was developed.

The stabilization method chosen is, in a sense, a combination of non-linear smoothing and non-linear damping. For the ART-type algorithms discussed above, the update for a pixel is the current pixel value, a_j, multiplied by a scale factor (or summed with an error magnitude in the case of additive ART). However, pixels which are being multiplied by large values tend to be especially susceptible to noise. To resolve this problem, the update for a pixel was chosen to be a non-linear combination of the back projection for the particular measurement and the current estimate, a_j^k. When the scale factor is relatively harsh in that it will cause a large change in the current pixel value, a larger portion of the back projection is included in the update. When the scale factor is relatively mild (near 1), the update emphasizes the current estimate of the pixel in the update. The non-linear update scheme incorporates a "soft limit" to constrain updates to between 2 and 0.5. This aids in the control of noise, because extremely noisy measurements have a limited effect on the update. The equation chosen is the spline of two symmetric functions in log space. There are an infinite number of functions with similar characteristics,

some not requiring the spline fit, but this particular form was found to be simple to implement and it performed exceptionally well.

The algorithm is best illustrated by its mathematical form:

$$update_{ij}^{k} = \begin{cases} \left[\frac{1}{2p_i^k}(1-\frac{1}{d_{ij}^k})+\frac{1}{a_j^k d_{ij}^k}\right]^{-1} h_{ij} & d_i^k \geq 1 \\ \left[\frac{1}{2}p_i^k(1-d_{ij}^k)+a_j^k d_{ij}^k\right] h_{ij} & d_{ij}^k < 1 \end{cases} \tag{4.7}$$

where p_i^k is the back projection of the i^{th} measurement, given by Eq. (3.3), and d_i^k is the damped scale factor

$$d_i^k = \left(\frac{s_i}{p_i^k}\right)^w \tag{4.8}$$

for which w can be any damping power (1 for no damping, 0.5 for square root damping).

Using this $update_{ij}^k$ the j^{th} pixel is modified each iteration using the formula

$$a_j^{k+1} = \frac{\sum_{i=1}^{N} update_{ij}^k}{\sum_{i=1}^{N} h_{ij}}. \tag{4.9}$$

First, note that $update_{ij}^k$ approaches a_j^k as d_i^k approaches 1. This satisfies a minimum requirement, in that the algorithm should be able to remain at a solution if a solution is found.

Consider a relatively smooth region, in which the back projection is approximately equal to the current pixel value. For such a case $update_{ij}^k$ for $(d_i^k < 1)$ is given by

$$update_{ij}^{k} = \left[\frac{1}{2}p_i^k(1-d_i^k)+a_j^k d_i^k\right] h_{ij} \tag{4.10}$$

$$\approx a_j^k \frac{1}{2}(1+d_i^k)h_{ij} \tag{4.11}$$

$$= a_j^k q(d_i^k) h_{ij} \tag{4.12}$$

where $q(d_i^k)$ is a function of the scale factor defined as

$$q(s) = \frac{1}{2}(1+s). \tag{4.13}$$

Similarly, for $(d_i^k \geq 1)$ $update_{ij}^k$ is

$$update_{ij}^k = \left[\frac{1}{2p_i^k}(1-\frac{1}{d_i^k})+\frac{1}{a_j^k d_i^k}\right]^{-1} h_{ij} \tag{4.14}$$

$$\approx \left[\frac{1}{2a_j^k}(1-\frac{1}{d_i^k})+\frac{1}{a_j^k d_i^k}\right]^{-1} h_{ij} \tag{4.15}$$

$$= \frac{2a_j^k}{1+\frac{1}{d_i^k}} h_{ij} \tag{4.16}$$

$$= a_j^k \frac{1}{q(\frac{1}{d_i^k})} h_{ij}. \tag{4.17}$$

A plot of $q(s)$ versus s is shown in figure 4.2. This plot shows the soft limiting effect of the new scale factor. Remember, however, this is only for the special case where the back projection is nearly equal to the current pixel value (which is true for relatively smooth areas).

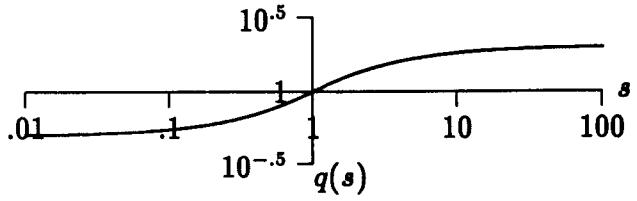

Figure 4.2: $q(s)$ versus s

If the only desired property of the modification of $update_{ij}^k$ was to soft limit the scale factor, then $q(s)$ could be used as a scale factor damping function. However, when the current pixel is not near its adjacent pixels, some form of smoothing is desired. By including in the update a portion of the back projection, the update contains a portion of a spatial average over the cell area. In the event the forward projection nearly matches the actual measurement, the scale factor will be near 1, in which case a smaller portion of the back projection is included in the update. This allows the image to focus. For large differences between the back projection and the actual measurement, large portions of the back projection are included in the update, which essentially adds smoothed values. After a few iterations, these areas should have back projections which nearly match the actual measurements, and can then begin to focus.

Scatterometer Iterative Reconstruction (SIR) is a block ART algorithm modified to use the constraining function described above. It can be expressed as

$$a_j^{k+1} = \frac{\sum_{i=1}^{N} update_{ij}^k}{\sum_{i=1}^{N} h_{ij}} \tag{4.18}$$

Ωwhere

$$update_{ij}^k = \begin{cases} \left[\frac{1}{2p_i^k}(1-\frac{1}{d_{ij}^k})+\frac{1}{a_j^k d_{ij}^k}\right]^{-1} h_{ij} & d_i^k \geq 1 \\ \left[\frac{1}{2}p_i^k(1-d_{ij}^k)+a_j^k d_{ij}^k\right] h_{ij} & d_{ij}^k < 1 \end{cases} . \tag{4.19}$$

Because of the non-linear nature of this approach a comprehensive mathematical analysis is difficult. Instead, analysis has been done using simulated and actual noisy data. These results have shown that the non-linear constraining function effectively stabilizes the system, while still providing resolution enhancement. Further exploration into its performance and convergence properties is ongoing.

4.2 $\mathcal{B}$ Estimation

All of the algorithms discussed previously were single parameter algorithms. For scatterometer application this would require knowledge of $\mathcal{B}$ in order to normalize the $\mathcal{S}$ matrix to a common angle of incidence. Therefore, for each measurement, σ_i^o,

$$s_{ij}^k = \sigma_i^o - b_j^k(\theta_i - 40^o). \tag{4.20}$$

The normalized measurements, $\mathcal{S}$, are now functions of the pixel number as well as the iteration number.

4.2.1 Multi-variate AVE Algorithm

Perhaps the easiest algorithm to adapt for multi-variate reconstruction is the AVE algorithm. This can be done by simply storing for each measurement the information needed to perform a linear regression on each pixel. A linear regression yields two parameters, a slope ($\mathcal{B}$ image) and a y-intercept (which can be normalized to 40^o and used as the $\mathcal{A}$ image). This technique is straight-forward and simple, but its resulting images can be misleading. Besides producing noisy $\mathcal{B}$ images, regions containing boundaries only in their $\mathcal{A}$ images are reconstructed to show boundaries in their $\mathcal{B}$ image. However, the $\mathcal{A}$ images usually remain quite clean regardless of the $\mathcal{B}$ images. The main flaw of the AVE algorithm with $\mathcal{B}$ estimation is, as before, the resulting poor resolution.

4.2.2 Multi-variate SIR

In order to utilize the superior resolution capabilities of the SIR algorithm, a corresponding method for $\mathcal{B}$ estimation was developed. $\mathcal{B}$ estimates are obtained by doing a linear regression on the updates on a pixel versus θs for each measurement hitting that pixel. When performing an iteration in SIR the σ^o measurement is first normalized to 40^o using the estimate of $\mathcal{B}$ and the first order linear model explained in Section 2.2. After $update_{ij}^k$ for the i^{th} measurement is calculated and stored for the $a_j^k + 1$ update, $update_{ij}^k$ is "unnormalized," or scaled back to the angle of incidence of the original measurement using the current $\mathcal{B}$

estimate for that pixel. The result is an "unnormalized" $update_{ij}^k$, or $(\hat{\sigma}^o)_{ij}^k$:

$$(\hat{\sigma}^o)_{ij}^k = update_{ij}^k + b_j^k(\theta_i - 40^o). \tag{4.21}$$

This "unnormalizing" uses the current $\mathcal{B}$ estimate as if it is correct. Although the $\mathcal{B}$ estimate is not necessarily correct, the information contained in $(\hat{\sigma}^o)_{ij}^k$ is still useful in improving the estimate of $\mathcal{B}$. As the scale factor approaches 1, $update_{ij}^k$ is approximately equal to s_{ij}^k, in which case $(\hat{\sigma}^o)_{ij}^k$ is approximately equal to σ^o . For such a situation the information stored for the $\mathcal{B}$ estimation is the same as if a linear regression were to be done on the unmodified measurements (as in the AVE algorithm with $\mathcal{B}$ estimation). In the event the scale factor is not near 1, the resulting $(\hat{\sigma}^o)_{ij}^k$ has a magnitude offset from the original σ^o. However, other measurements hitting the same pixel should also on the average have a magnitude offset in the same direction, in effect changing the estimate of the y-intercept when a linear regression is performed, but not the estimate of the slope. Realizing that this method only approximates the proper $\mathcal{B}$ values, the resulting estimates are heavily filtered as explained later in this section.

To implement the $\mathcal{B}$ estimation the algorithm first requires an initial estimate of $\mathcal{B}$. The value of -0.13 was selected, since it is an average of what is expected over the entire earth. Next, the entire data set is first processed for one iteration to obtain the $(\hat{\sigma}^o)_{ij}^k$ values. Then, a linear regression is performed for each pixel, given by Eq. (4.22), to obtain the new $\mathcal{B}$ estimate.

$$\tilde{b}_j^k = \frac{c_j \sum_{i=1}^{N} \theta_i (\hat{\sigma}^o)_{ij}^k h_{ij} - \sum_{i=1}^{N} \theta_i h_{ij} \sum_{i=1}^{N} (\hat{\sigma}^o)_{ij}^k h_{ij}}{c_j \sum_{i=1}^{N} \theta_i^2 h_{ij} - (\sum_{i=1}^{N} \theta_i h_{ij})^2} \tag{4.22}$$

$$c_j = \sum_{i=1}^{N} h_{ij} \tag{4.23}$$

In Eq. (4.22) $\tilde{b}_j^k$ is the estimated $\mathcal{B}$ value for the j^{th} pixel and h_{ij} is used as before to select whether a given measurement affects the estimate. Because $\tilde{b}_j^k$ is a very rough estimate of what the true $\mathcal{B}$ value is, it is weighted and then combined with the current $\mathcal{B}$ value. To obtain a weighting function, a measure of

the variance is first determined, from which 1 is subtracted. The resulting number (termed *weight*) is very near zero if the variance in angle is small. (A small variance can lead to a large slope error.)

$$weight = \frac{c_j \sum_{i=1}^{N} \theta_i^2 h_{ij}}{\left(\sum_{i=1}^{N} \theta_j h_{ij}\right)^2} - 1 \tag{4.24}$$

Larger θ variations result in larger weights which allows greater confidence in the accuracy of the update of $\mathcal{B}$. The estimated $\mathcal{B}$ value, $\tilde{b}_j^k$, is multiplied by *weight*, then added to the current $\mathcal{B}$ estimate b_j^k. Finally, the sum is divided by $1 + weight$ to yield the updated $\mathcal{B}$ estimate.

$$b_j^{k+1} = \frac{\tilde{b}_j^k * weight + b_j^k}{1 + weight} \tag{4.25}$$

The weighting acts as a damping term to minimize the effects of large $\mathcal{B}$ errors and slow the convergence of the $\mathcal{B}$ image. This weighting damps the $\mathcal{B}$ image in such a way as to add noise, since small groups of $\mathcal{B}$ pixels have been weighted differently than those in adjacent groups. Therefore, a nine-point averaging filter is applied to the entire $\mathcal{B}$ image, in an effort to spread some of the $\mathcal{B}$ estimation. This allows pixels which are not having their $\mathcal{B}$ values updated due to a small range of θ to still experience a net update on their $\mathcal{B}$ values due to changes in spatially related pixels. Recall it was determined earlier that the effects of $\mathcal{B}$ are small; therefore, the smoothing has little impact on the $\mathcal{A}$ image. Some of the effects of the $\mathcal{B}$ image on the $\mathcal{A}$ image are considered in the results chapter (Chapter 6).

As before, the non-linear nature of this estimation makes a rigorous mathematical analysis difficult. Therefore, it was tested extensively and, as can be seen in Chapter 6, it is capable of a suitable $\mathcal{B}$ estimation for the data.

An averaging filter was not used directly on the $\mathcal{A}$ images, because such a filter would significantly lower the resolution. However, because of the noise, some filtering is desirable. The modified median filter performed as a normal median filter as long as the difference between the second-to-the-largest value and the second-to-the-smallest value exceeded some threshold. This is likely to occur near region boundaries. The second-to-the-largest and second-to-the-smallest were

chosen to allow the two extreme values to be attributed to noise. If the difference was below the threshold, an averaging filter was applied. A subjective decision was made to use 0.25 dB as the threshold after observing multiple outputs for different threshold values. The concept behind such a filter is to avoid averaging in areas where there are large changes (to keep edges sharp) but to allow averaging in areas where there is reasonably small variance in pixel values. Clearly, this method is only justifiable if the output maps produced using it have better characteristics than those produced without it. The results chapter contains comparisons of the algorithm using this filter (SIRF) to the algorithm using a normal median filter (SIR).

4.3 Noise/Resolution Trade-off

In most image-processing systems there exists a trade-off between noise and resolution. If the resolution of an image is to be enhanced, the amount of noise in the image can be expected to increase. However, the trade-off is such that doubling the resolution does not necessarily double the noise. Therefore, a subjectively optimal image would balance the noise with the resolution. Many parameters in the SIRF algorithm can be adjusted to yield different noise/resolution characteristics. Such parameters are often referred to as a hyper-parameters. For example, the modified median filter above has a threshold of 0.25. If a smoother image is desired this parameter can be increased; if a sharper one is desired this parameter can be decreased. The non-linear constraining function itself can be considered a hyper-parameter or perhaps a hyper-function; clearly, an infinite number of ad-hoc algorithms could be developed which would perform similarly to SIR/SIRF with minor variations in output. The non-linear constraining function in SIR/SIRF was chosen because of its simplicity and because it works. In short, the algorithms can be tuned to yield subjectively better images in terms of the balance between noise and resolution.

4.4 Summary

This chapter has explained the methodology used in the development of the SIRF algorithm. Multiplicitive block ART was used as the foundation for

SIRF, with the scale factor being modifed to incorperate a non-linear constraining function. The constraining function combines a portion of a damped scale factor multiplied by the current pixel with the back projection to form the update for a single pixel. The inclusion of the back projection effectively includes a spatial average in the update. The relative weightings are determined by the magnitude of the scale factor. The B estimation is accomplished by storing information for a linear regression which is performed at the end of each iteration. The resulting B image is heavily filtered. A special median filter was developed to preserve edges, but allow smoothing in regions already moderately smooth. The results chapter contains detailed information on the SIRF algorithm's performance.

CHAPTER 5

APPLICATION OF SIRF TO SASS DATA SET

Although the SIRF algorithm was developed specifically for SASS, its applicability is not limited to SASS, nor is it limited to scatterometers. For this reason the descriptions of the algorithm have been as general as possible. Provided in this chapter is the additional information needed to apply the SIRF algorithm to the SASS data set.

For the SASS data set, p_i^k is modified to be the forward projection in log space, calculated in normal space. This is due to the fact that although are images are generated in log space, the normal space model for forward projection generation is a better model of how the scatterometer makes measurements. The pixels, a_i, are in log space, and therefore are first converted to normal space, then averaged, with the result being converted back to log space,

$$p_i^k = 10\log_{10}\left[\frac{\sum_{n=1}^{M} 10^{(a_n/10)}h_{in}}{\sum_{n=1}^{M} h_{in}}\right]. \tag{5.1}$$

Also, for the SASS data set, using a $\mathcal{B}$ estimate, the measurement values become functions of the pixel number, because the $\mathcal{B}$ estimate varies with the pixel number,

$$s_{ij}^k = \frac{\sigma_i^o - b_j^k(\theta_i - 40^o)}{p_i^k}. \tag{5.2}$$

The damped scale factor also becomes a function of pixel number as well as measurement number,

$$d_{ij}^k = (s_{ij}^k)^w. \tag{5.3}$$

Applying these minor modifications to SIRF allows the algorithm to process the SASS data set and yield an image in log space.

5.1 Summary of Final SIRF Algorithm

The complete algorithm is presented below in a mixed mathematical/pseudo-code form.

Repeat n times:

For every measurement, s_i, in the data set, first compute its forward projection, p_{ij}^k, using the equation

$$p_i^k = 10\log_{10}\left[\frac{\sum_{n=1}^{M} 10^{(a_n/10)}h_{in}}{\sum_{n=1}^{M} h_{in}}\right]. \tag{5.4}$$

Next, using this back projection, calculate the update for each pixel, using the equation

$$update_{ij}^k = \begin{cases} \left[\frac{1}{2p_i^k}(1-\frac{1}{d_{ij}^k})+\frac{1}{a_j^k d_{ij}^k}\right]^{-1} h_{ij} & d_i^k \geq 1 \\ \left[\frac{1}{2}p_i^k(1-d_{ij}^k)+a_j^k d_{ij}^k\right] h_{ij} & d_{ij}^k < 1 \end{cases} \tag{5.5}$$

where the term d_{ij} is

$$d_{ij}^k = (s_{ij}^k)^{.5} \tag{5.6}$$

$$= \left[\frac{\sigma_i^o - b_j^k(\theta_i - 40^o)}{p_i^k}\right]^{.5}. \tag{5.7}$$

Finally, store the update for each pixel in a summing register to be averaged after the entire data set has been processed. Also, for $\mathcal{B}$ estimation, calculate the "unnormalized" update given by

$$(\hat{\sigma}^o)_{ij}^k = update_{ij}^k + b_j^k(\theta_i - 40^o) \tag{5.8}$$

and add the information needed for a linear regression to the j^{th} pixel's summing registers ($\hat{\sigma}^o$, $\hat{\sigma}^o\theta$, θ, and θ^2).

After all measurements have been processed, replace the old $\mathcal{A}$ and $\mathcal{B}$ estimates as follows:

For each pixel both parameters, a_j and b_j, can be updated only if one or more measurements affected that pixel, i.e., $c_j > 0$. If this is true, set a_j equal

to the average of the updates calculated for that pixel and update b_j according to

$$\tilde{b}_j^k = \frac{c_j \sum_{i=1}^{N} \theta_i (\hat{\sigma}^o)_{ij}^k h_{ij} - \sum_{i=1}^{N} \theta_i h_{ij} \sum_{i=1}^{N} (\hat{\sigma}^o)_{ij}^k h_{ij}}{c_j \sum_{i=1}^{N} \theta_i^2 h_{ij} - (\sum_{i=1}^{N} \theta_i h_{ij})^2} \tag{5.9}$$

$$weight = \frac{c_j \sum_{i=1}^{N} \theta_i^2 h_{ij}}{\left(\sum_{i=1}^{N} \theta_j h_{ij}\right)^2} - 1 \tag{5.10}$$

$$b_j^{k+1} = \frac{\tilde{b}_j^k * weight + b_j^k}{1 + weight}. \tag{5.11}$$

After every pixel has been processed, perform a modified median filter on the $\mathcal{A}$ image and an averaging filter on the $\mathcal{B}$ image. After resetting the measurement file, another iteration can begin.

5.2 Computational Considerations

The processing time needed to generate a large image is enormous. Just to store the data for a one million-pixel image (such as the Amazon basin at 25 pixels per degree), its updates, and its $\mathcal{B}$ estimation summing registers requires 30 megabytes. This large memory requirement generates large numbers of page faults on our VAX series 4000 model 300 (with 32 megabytes of internal memory), which significantly slows the program. Also slowing the program is the large amount of floating point operations that take place to process a single measurement. And finally, a large amount of I/O occurs as the data blocks are read into memory. (However, this was found to contribute only slightly to the total processing time.)

Speed became a major concern when processing the Amazon region. Using the first version of the code it was apparent that it would take nearly a week to run fifty iterations. Therefore, in order to lower the memory usage, a two-byte integer quantization format was used. In order to make maximal use of the two-byte integers, averaging registers were used instead of summing registers.

To add a new value to an averaging register, the contents of the averaging register are first converted from an integer to a real, then it is multiplied by the total number of values previously combined in the averaging register. It

then has the value to be added to the averaging register added to it, and this sum is divided by the new number of values in the averaging register (one more than before). Finally, this number is converted back to an integer and stored in the averaging register.

Let x be an averaging register containing the average of c values, and y be the value to be added. The process explained above can then be rewritten as

$$\tilde{x} = (integer)\left(\frac{(real)(x) * c + y}{c+1}\right). \tag{5.12}$$

Not shown implicitly above, the values are scaled to make maximum use of the 2 bytes integers. For example, since the $\mathcal{B}$ estimates were not expected to vary much from -0.13, they were all scaled by 10^4.

The change from summing registers to averaging registers caused some minor modification to the implementation of the SIRF algorithm. After all measurements were processed, the a_js were set to the contents of the corresponding averaging registers. The Ωlinear regression changed because, to get the sums required for a linear regression, the averages had to be multiplied by the number of values they had averaged. However, this in fact simplified the linear regression to eliminate the variable representing the number of values in the sum. A linear regression of x versus y to obtain the slope parameter is

$$slope = \frac{c\sum xy - \sum x \sum y}{c\sum x^2 - (\sum x)^2} \tag{5.13}$$

where c is the number of values in the sum. Given averaging registers $\sum \hat{x}$, $\sum \hat{xy}$, $\sum \hat{x^2}$, and $\sum \hat{xy}$, and recognizing that $\sum x = c\sum \hat{x}$, Eq. (5.13) can be rewritten as

$$slope = \frac{c^2\sum \hat{xy} - c^2\sum \hat{x}\sum \hat{y}}{c^2\sum \hat{x^2} - c^2(\sum \hat{x})^2} = \frac{\sum \hat{xy} - \sum \hat{x}\sum \hat{y}}{\sum \hat{x^2} - (\sum \hat{x})^2}. \tag{5.14}$$

The noise added by the integer quantization was random and negligible. The integer quantization scheme added some complexity to the code, and also added computation time to convert locally to and from reals. However, halving the memory requirements paid big dividends, allowing the program to process the entire Amazon region (fifty iterations) in around twenty-four CPU hours.

CHAPTER 6

RESULTS

A three-stage process was used to evaluate the performance of the SIRF algorithm. The first stage involved using simulated measurements of a synthetic scene without noise to verify the basic ability of the algorithm to provide resolution enhancement. The second stage involved using simulated measurements from the same synthetic scene, but with added noise, to test noise tolerances. The final stage involved using the actual SASS data set to generate images.

To obtain a relative measure of performance, maps were generated for each stage using (1) a $0.5^o \times 0.5^o$ binning algorithm similar to the one used by Kennett and Li (LoRes), (2) the AVE algorithm at its highest resolution, (3-4) both ART algorithms, (5) the damped ART algorithm, and (6-7) the SIR and SIRF algorithms. To speed computational time the region processed was relatively small, $7^o \times 7^o$. For a final comparison, the entire Amazon region was processed using AVE and SIRF, both of which were compared to the results of Kennett and Li.

6.1 Test Data Set Generation

The test data set was generated from two synthetic images: an $\mathcal{A}$ image and a $\mathcal{B}$ image. The $\mathcal{A}$ image had $\mathcal{A}$ values ranging from -20 dB to -5 dB, while the $\mathcal{B}$ image had values ranging between -0.05 and -0.2, which are similar to what was expected from the real data set. The features were made to be reasonably small, some far smaller than the cell resolution, to verify the increased resolution for features anticipated on the real data. Within each image some regions were held constant while others varied with spatial position.

The $\mathcal{B}$ image was specifically designed to explore the effects of the $\mathcal{B}$ estimate on the $\mathcal{A}$ image, as well as to experiment with SIRF's $\mathcal{B}$ estimation routine. In some regions, $\mathcal{B}$ was made to vary with the $\mathcal{A}$ image, which is probably a good model of the real world. In other regions $\mathcal{B}$ varied independently of the $\mathcal{A}$ values. And finally, some regions were left with a constant $\mathcal{B}$ value. The bottom

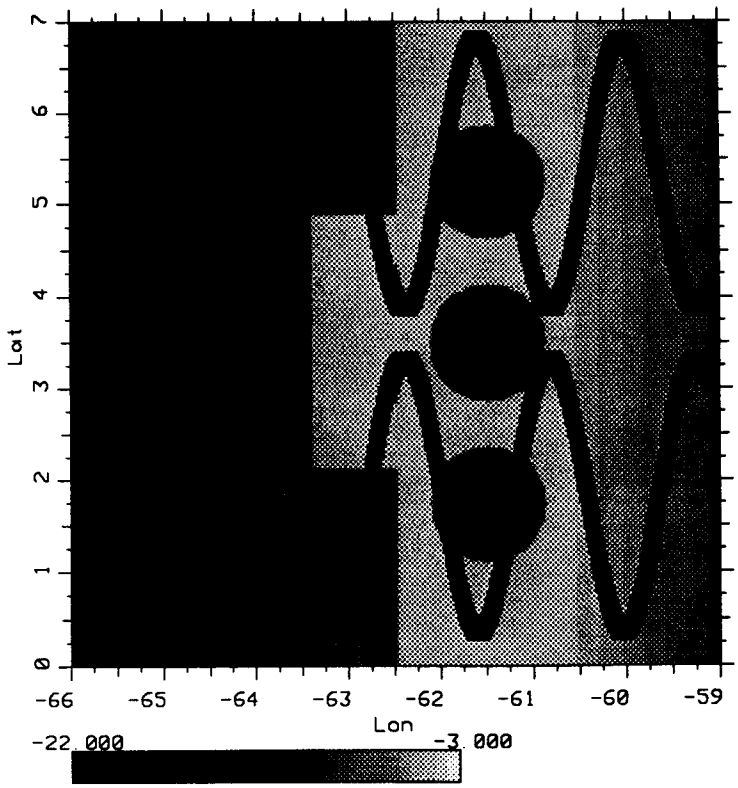

Figure 6.1: Original test map - $\mathcal{A}$ image

half of the $\mathcal{B}$ image was left constant at -0.13, which is the expected average $\mathcal{B}$ value for the given region. Holding this half constant, and allowing the algorithms without $\mathcal{B}$ estimation to use this same value when processing, allows the bottom halves of the resulting $\mathcal{A}$ images to be generated without error in $\mathcal{B}$ and compared to the tops.

Figures 6.1 and 6.2 are the $\mathcal{A}$ and $\mathcal{B}$ images used to create the simulated data sets.

To generate simulated SASS σ^o measurements for the synthetic data sets, the actual cell locations, sizes, angles (θ), and shapes were taken from the SASS GDR data set and overlayed on the synthetic maps. The forward projections were calculated using Eq. (5.1) and the σ^o values of the original data set were replaced with these projections.

To simulate noise, a zero-mean Gaussian noise source (ν) with variance

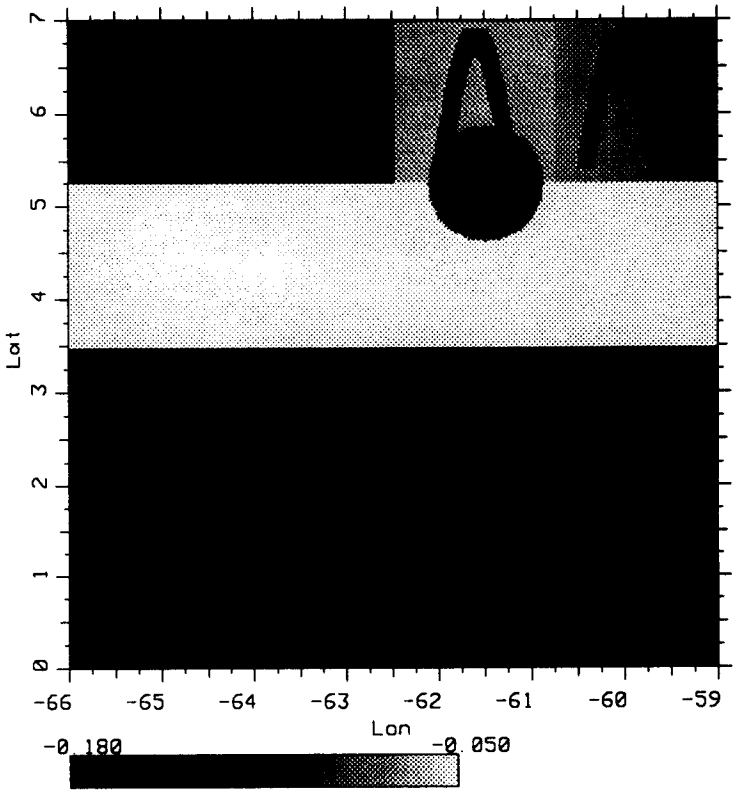

Figure 6.2: Original test map - $\mathcal{B}$ image

$(\sigma^o Kp)^2$ was added to the forward projections generated from the synthetic scenes.

$$\sigma^o = p_{i,j}^k(1 + \frac{kp}{100}\nu) \tag{6.1}$$

This σ^o replaced that of the actual measurement taken by SASS, and the data block was stored to disk.

After comparing the outputs of the maps generated from the actual data sets to the maps generated from the synthetic data sets, it was observed that there was more noise in the actual data than was estimated by Kp. For this reason an additional 10% of Gaussian noise was added to the synthetic measurements.

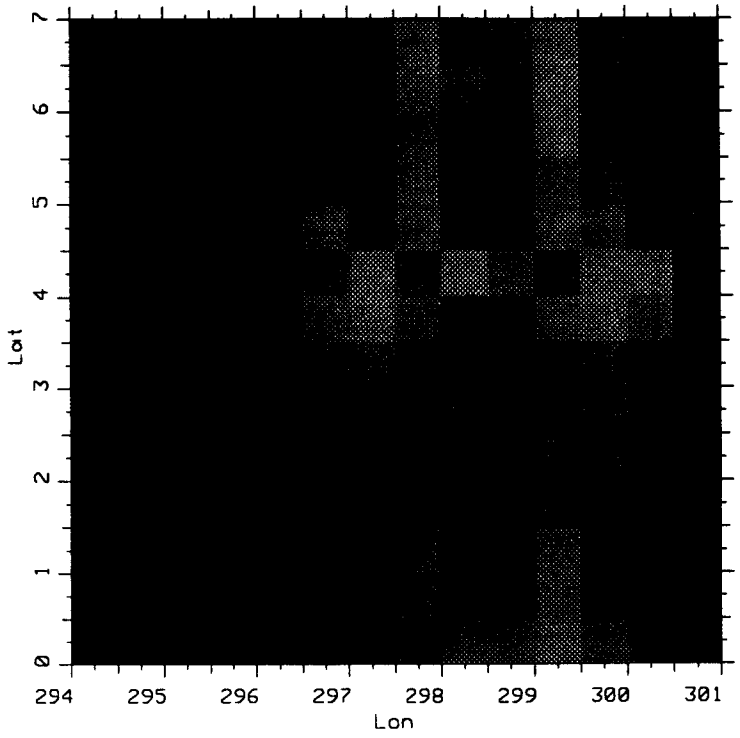

Figure 6.3: Test map - LoRes - noise free - $\mathcal{A}$ image

6.2 Noise-free Simulations

We first consider the noise-free case ($\nu = 0$).

6.2.1 LoRes

Shown in Figure 6.3 is the image generated at $0.5^o \times 0.5^o$ resolution. Very few of the original map features are apparent.

Also included is the $\mathcal{B}$ image (Figure 6.4) generated by estimating the $\mathcal{B}$ value by simply taking the linear regression of all measurements hitting a given pixel.

6.2.2 AVE

Figures 6.5 and 6.6 are the outputs generated by the AVE algorithm using a data set with no noise. Again, a $\mathcal{B}$ estimate was obtained by taking the linear regression of all measurements hitting a given pixel.

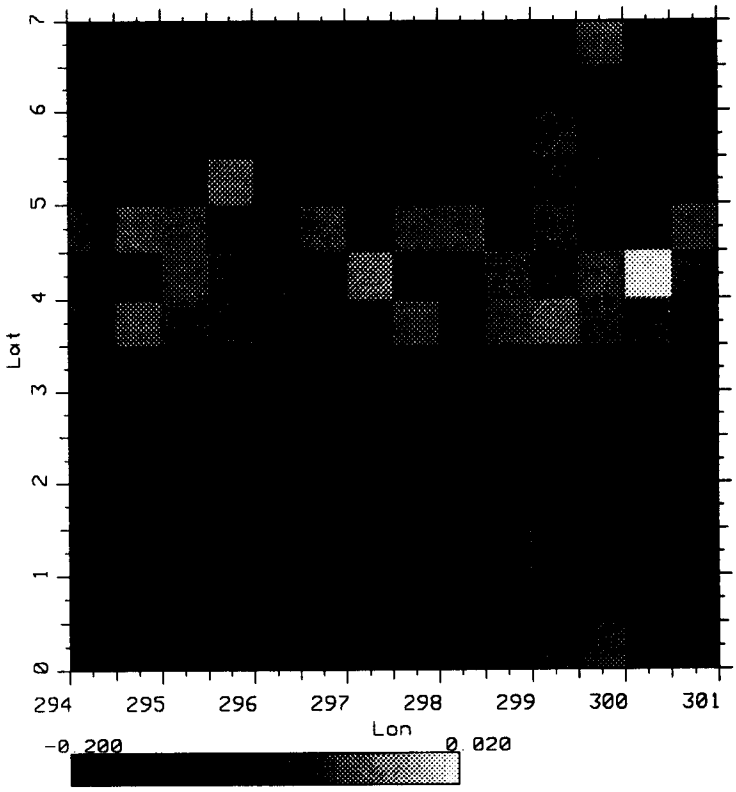

Figure 6.4: Test map - LoRes - noise free - B image

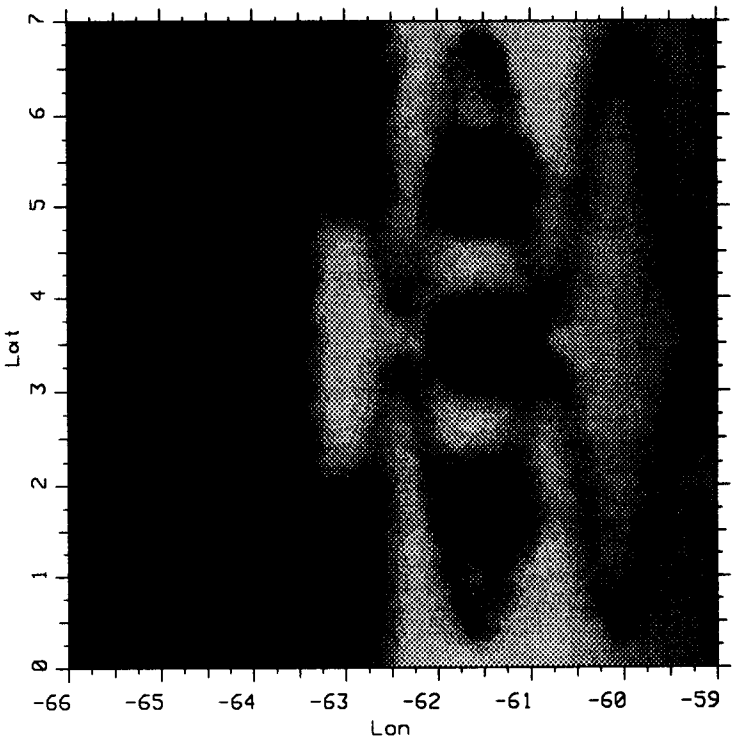

Figure 6.5: Test map - AVE - noise-free - $\mathcal{A}$ image

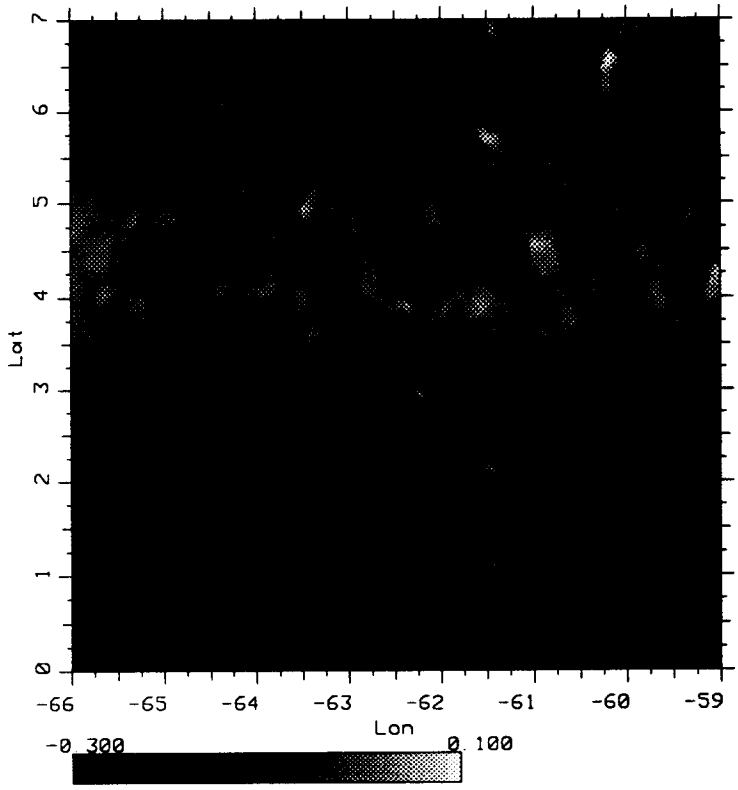

Figure 6.6: Test map - AVE - noise-free - $\mathcal{B}$ image

6.2.3 ART

For the ART demonstration both ART algorithms, additive (Figure 6.7) and multiplicative (Figure 6.8), were used. Each was run for fifty iterations. ART algorithms usually have to be stopped after a number of iterations due to their instability. Fifty iterations was selected to maximize image quality after observing the outputs up to fifty iterations and noting the image was not changing much between iterations. (Throughout this paper fifty iterations was used for all iterative algorithms, except in cases where an algorithm's output has gone unstable, in which case the best image was selected and the number of iterations is explicitly stated.)

For the ART algorithms, a $\mathcal{B}$ estimation routine was not developed, therefore the image was processed assuming a constant $\mathcal{B}$ value of -0.13.

Both ART algorithms did an exceptional job processing the bottom halves of the image where the $\mathcal{B}$ values were correct. The top halves, however, are quite noisy, due to the data inconsistencies caused by incorrect $\mathcal{B}$ values. The sine wave in both images is well defined for the bottom halves, and the edges of the regional boundaries are nearly perfect. This is not surprising, as many others have studied the performance of ART and found it quite capable of image reconstruction in the absence of noise.

6.2.4 Block ART

Shown next are the results of Block ART (Figure 6.9). (Recall that block ART is nearly the same algorithm as multiplicative ART, except that it takes block averages before image updates.) The key difference to note is that block ART is not quite as sharp around the edges, but in the top half the data inconsistencies caused by the incorrect $\mathcal{B}$ image have a smaller effect.

6.2.5 SIR

Shown in Figure 6.10 is the resulting $\mathcal{A}$ image after processing the data set with the SIR algorithm. In contrast to the previous three images, the top half is not noise-corrupted. This is due to two factors: First, the SIR algorithm is multi-variate and estimates the $\mathcal{B}$ image as it estimates the $\mathcal{A}$ image. Second, the SIR algorithm is more noise-tolerant and can better handle the data inconsistencies

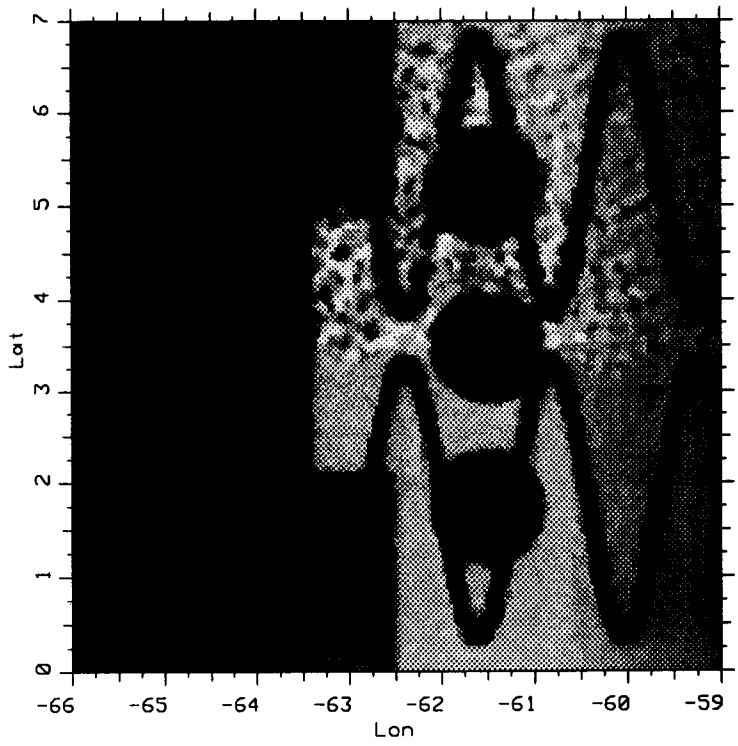

Figure 6.7: Test map - additive ART - noise free - $\mathcal{A}$ image

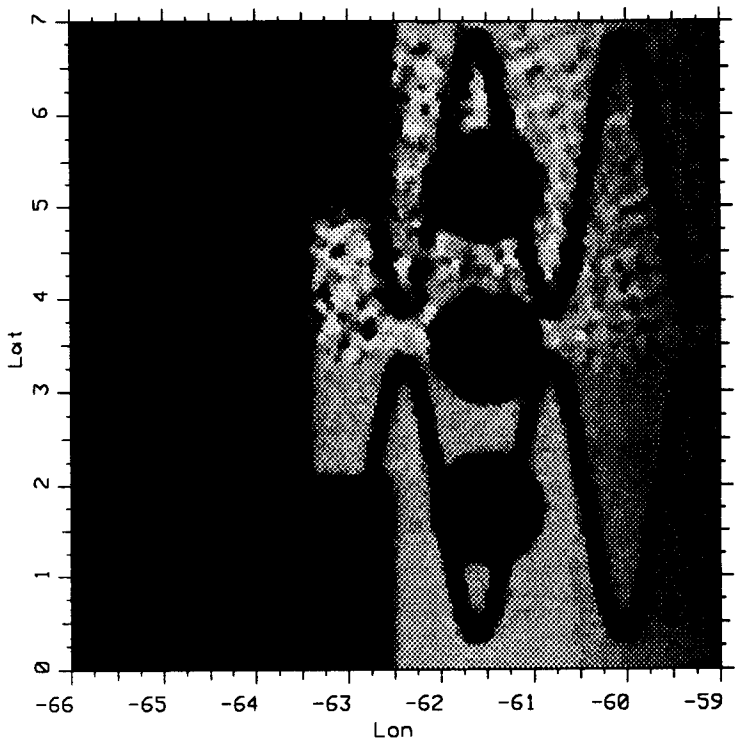

Figure 6.8: Test map - multiplicative ART - noise free - $\mathcal{A}$ image

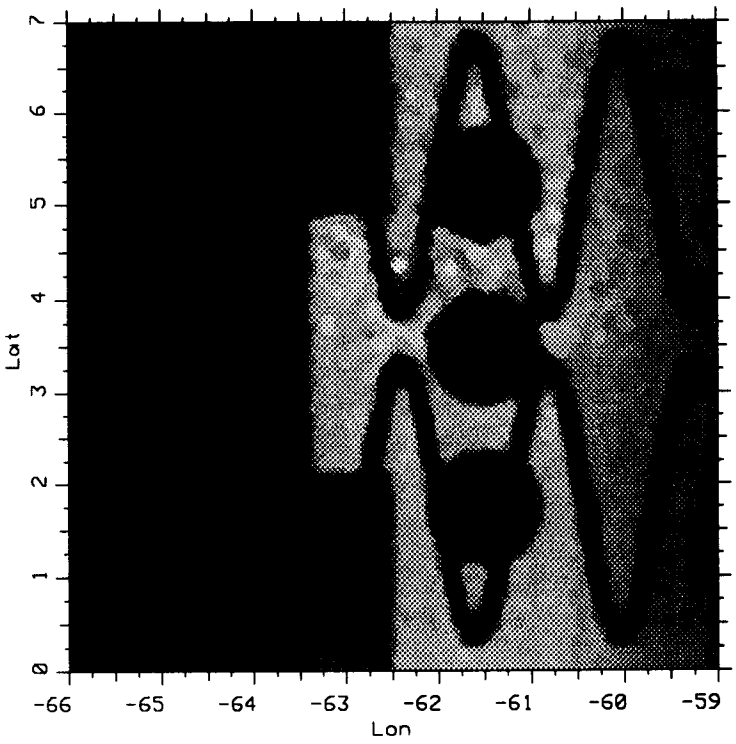

Figure 6.9: Test map - block ART - noise free - $\mathcal{A}$ image

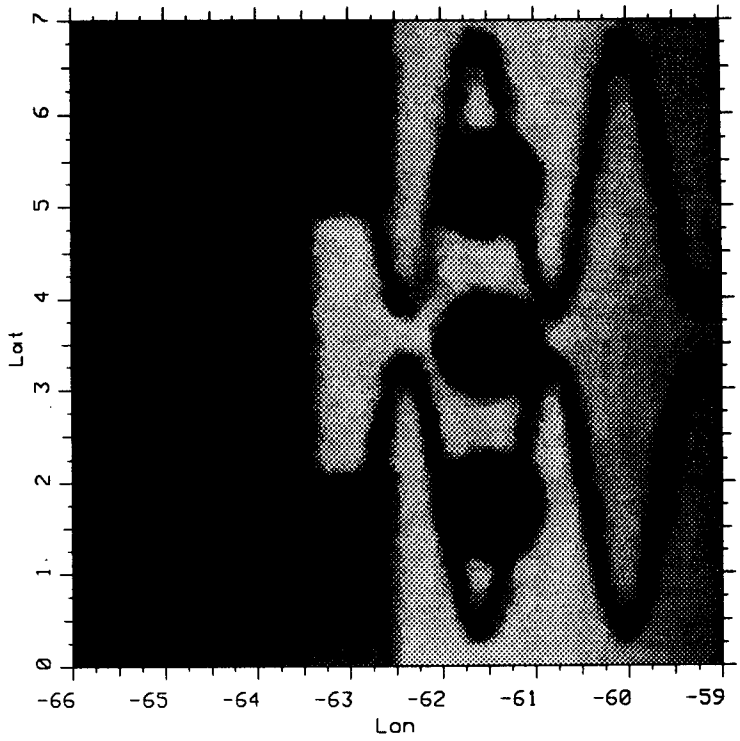

Figure 6.10: Test map - SIR - noise free - $\mathcal{A}$ image

caused by incorrect $\mathcal{B}$ values.

The image is slightly less sharp than the previous three, but retains good resolution, especially when compared to the output of the AVE algorithm. The regional boundaries are quite sharp, and the sine waves are nearly the proper width.

Figure 6.12 is the $\mathcal{B}$ image corresponding to Figure 6.11. At first sight the image might seem to be an unacceptable estimation of the $\mathcal{B}$ image. However, note that, although the sine wave is not visible, the other features can be seen relatively well in the image.

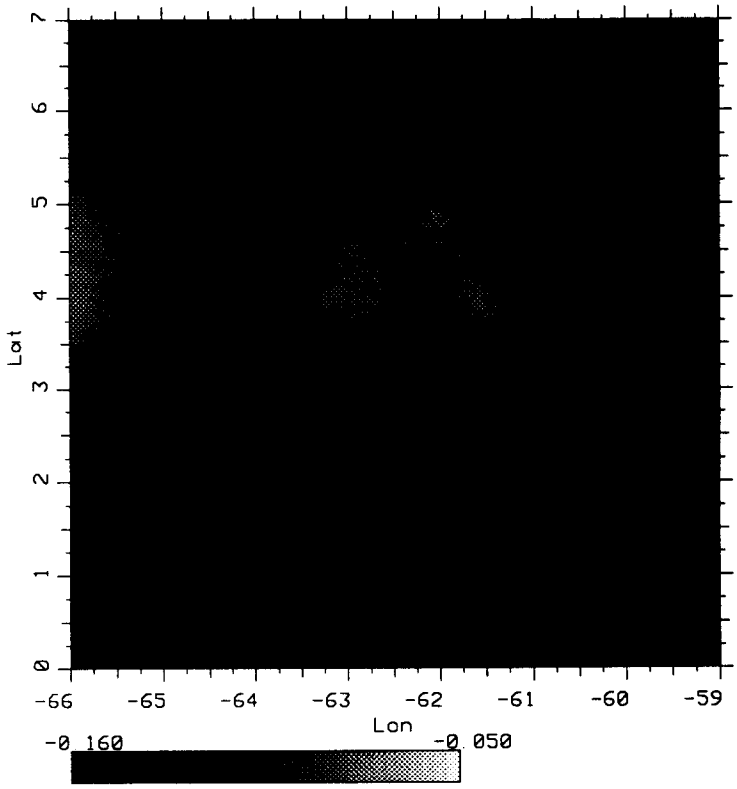

Figure 6.11: Test map - SIR - noise free - $\mathcal{B}$ image

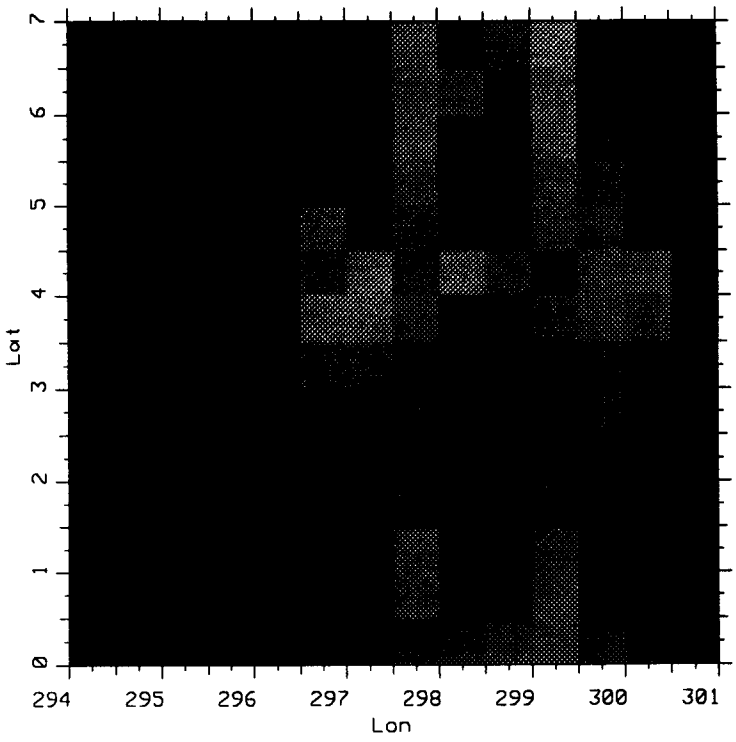

Figure 6.12: Test map - LoRes - noise present - $\mathcal{A}$ image

6.3 Data-with-Noise Simulations

Following are the results using the data set created with added noise. The amount of noise added was determined by Kp using the equation

$$\sigma^o = p_{ij}^k(1 + \frac{kp + 10}{100}\nu). \tag{6.2}$$

The added 10% was to account for unmodeled noise not included in Kp. Images were generated with data sets created using only Kp, but are not included in the interest of space.

6.3.1 LoRes

Shown in Figure 6.13 is the image generated at $0.5^o \times 0.5^o$ resolution. Very few of the original map features are apparent. Also included is the $\mathcal{B}$ image (Figure 6.14) generated by estimating the $\mathcal{B}$ value by simply taking the linear regression of all measurements hitting a given pixel.

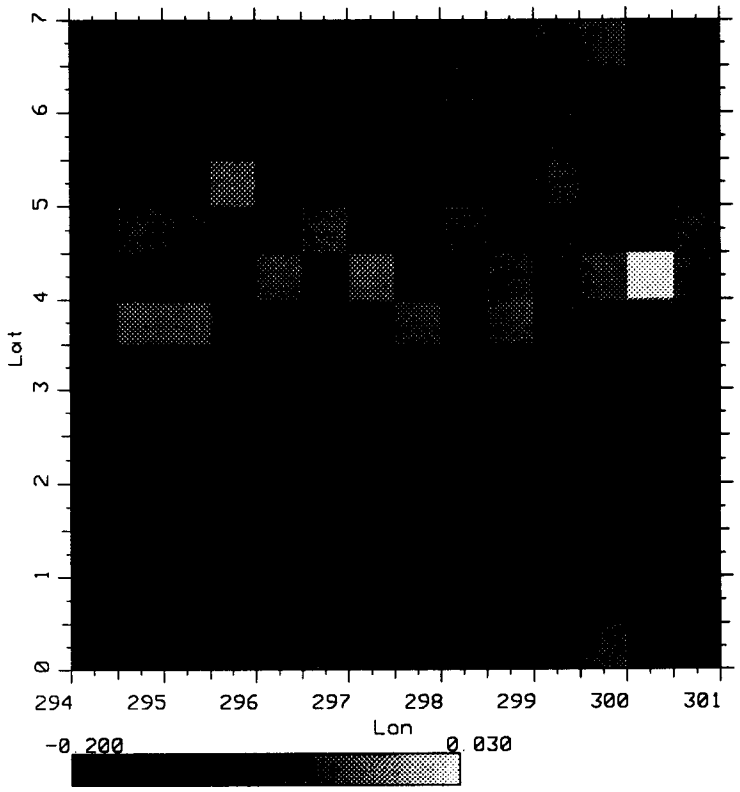

Figure 6.13: Test map - LoRes - noise present - $\mathcal{B}$ image

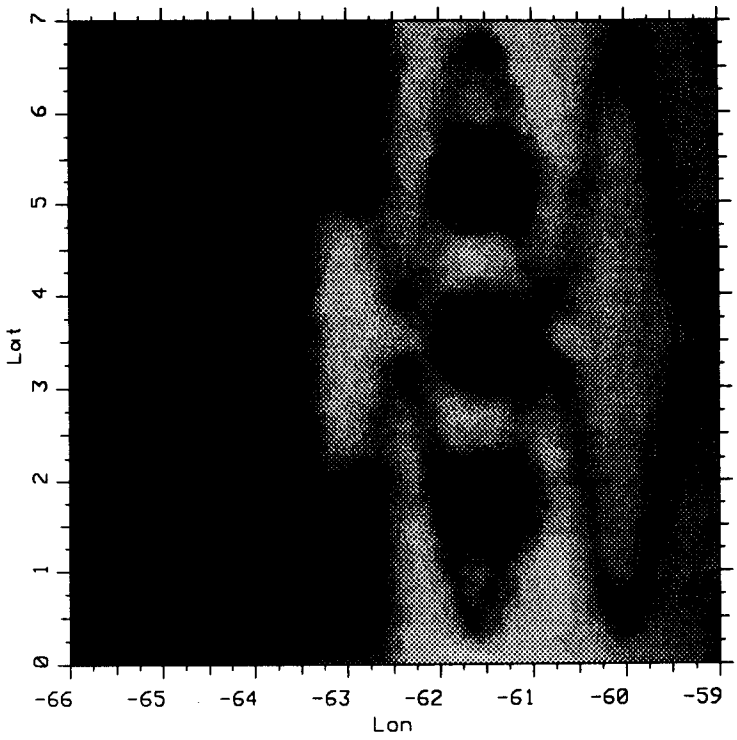

Figure 6.14: Test map - AVE - noise present - $\mathcal{A}$ image

6.3.2 AVE

Figures 6.15 and 6.16 show the outputs resulting when the AVE algorithm processed the data set including noise. Again, as stated before, the AVE algorithm is quite noise-tolerant; in fact, it is difficult to distinguish between the output using the noise-free data set and the output using the noisy data set.

6.3.3 ART

Figures 6.17 and 6.18 show the outputs from the ART algorithms operating on the noise-corrupted data sets. The $\mathcal{B}$ estimation problem is mute in this case, because it appears that the estimate is equally as bad in both halves of the image. As mentioned previously, the instability of ART in the presence of noise is not a new discovery. It is this instability that motivated the pursuit of other processing methods.

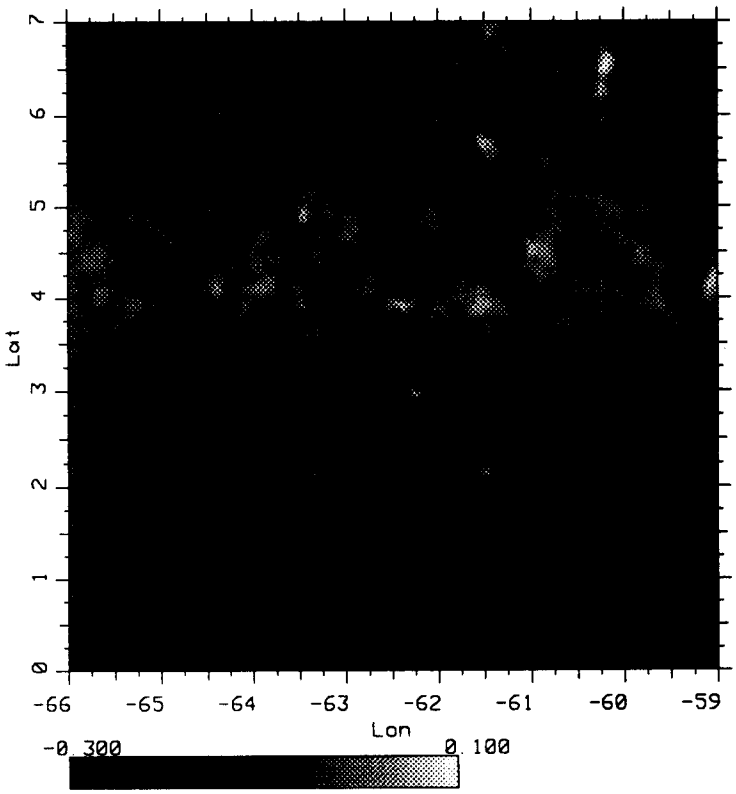

Figure 6.15: Test map - AVE - noise present - $\mathcal{B}$ image

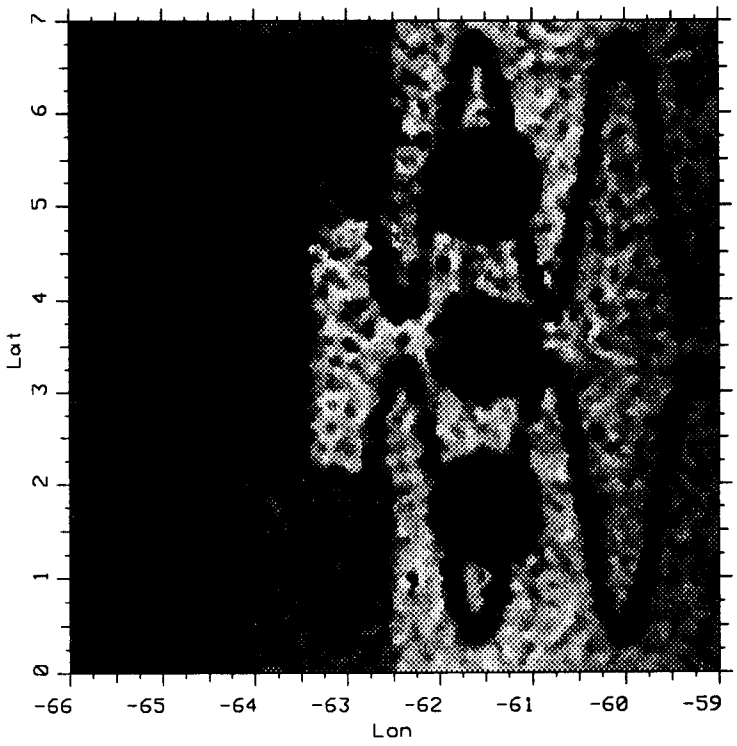

Figure 6.16: Test map - additive ART - noise present - $\mathcal{A}$ image

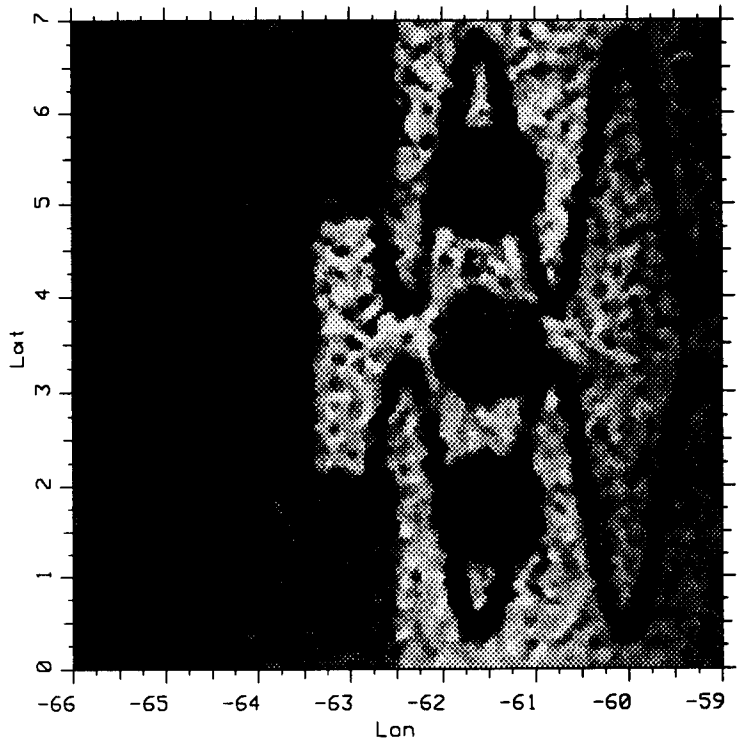

Figure 6.17: Test map - multiplicative ART - noise present - $\mathcal{A}$ image

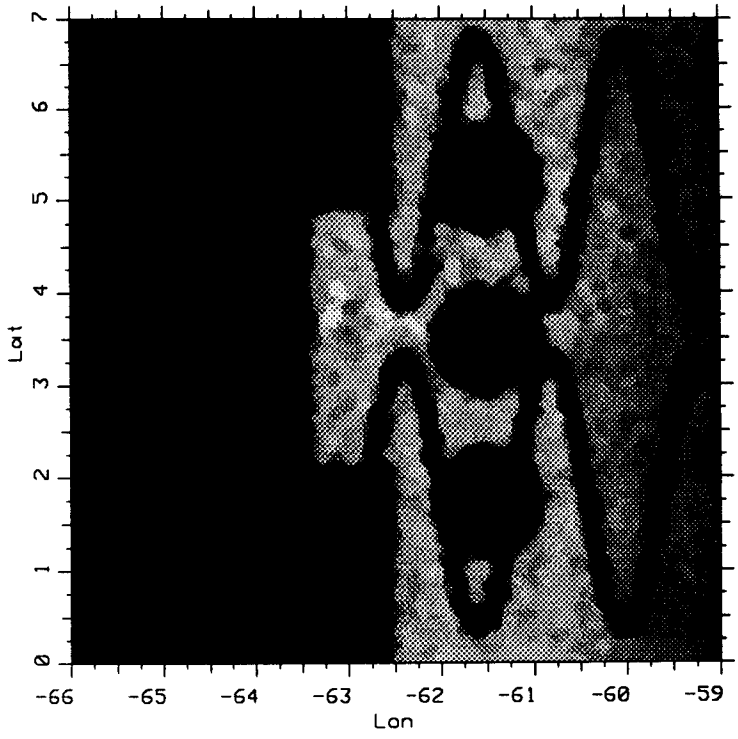

Figure 6.18: Test Map - Block ART - Noise present - $\mathcal{A}$ image

6.3.4 Block ART

Figure 6.19 shows the output of the block ART algorithm. While it is smoother than the multiplicative ART algorithm results, it still contains an unacceptable amount of noise. Figure 6.20 is the resulting image when the block ART algorithm is used with square root damping of the scale factor.

6.3.5 SIR

Figures 6.21 and 6.22 are the $\mathcal{A}$ and $\mathcal{B}$ estimates generated using the SIR algorithm. As with the AVE algorithm, the images are quite similar to those generated using the noise-free data sets.

6.3.6 SIRF

Figure 6.23 contains the $\mathcal{A}$ image resulting from the use of SIRF. The image is very similar to the one shown previously for SIRF, but smoother. Notice,

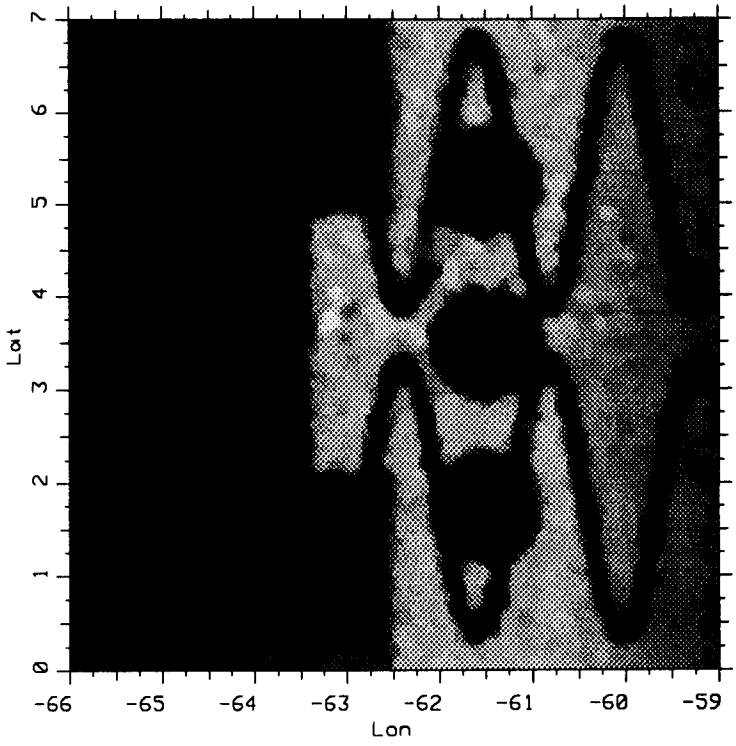

Figure 6.19: Test Map - Damped Block ART - Noise present - $\mathcal{A}$ image

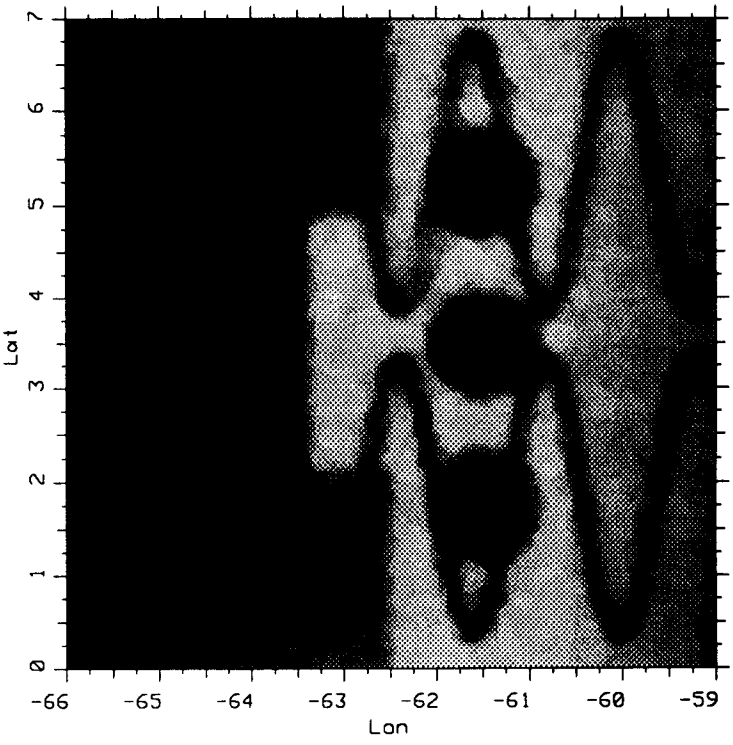

Figure 6.20: Test map - SIR - noise present - $\mathcal{A}$ image

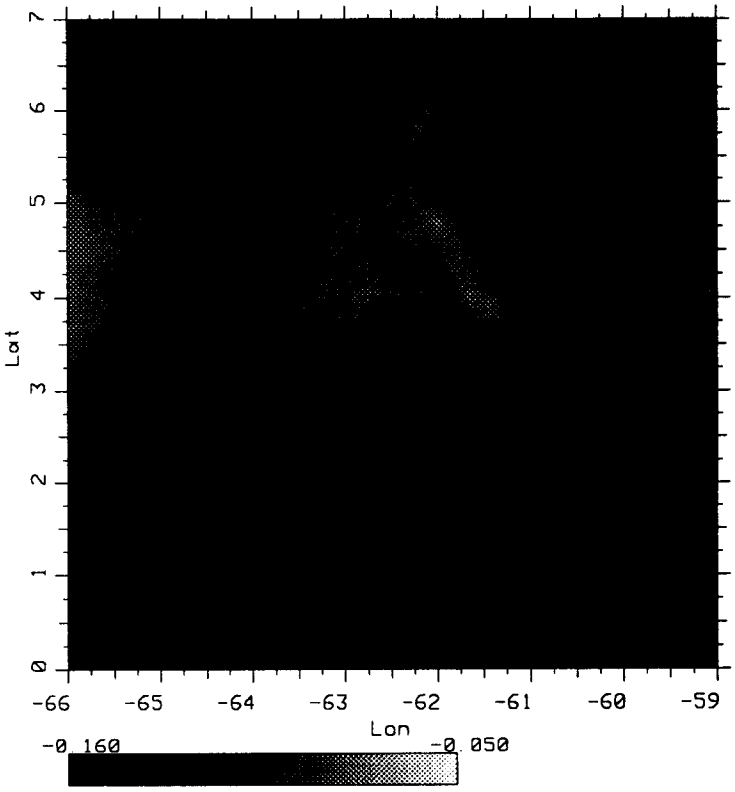

Figure 6.21: Test map - SIR - noise present - $\mathcal{A}$ image

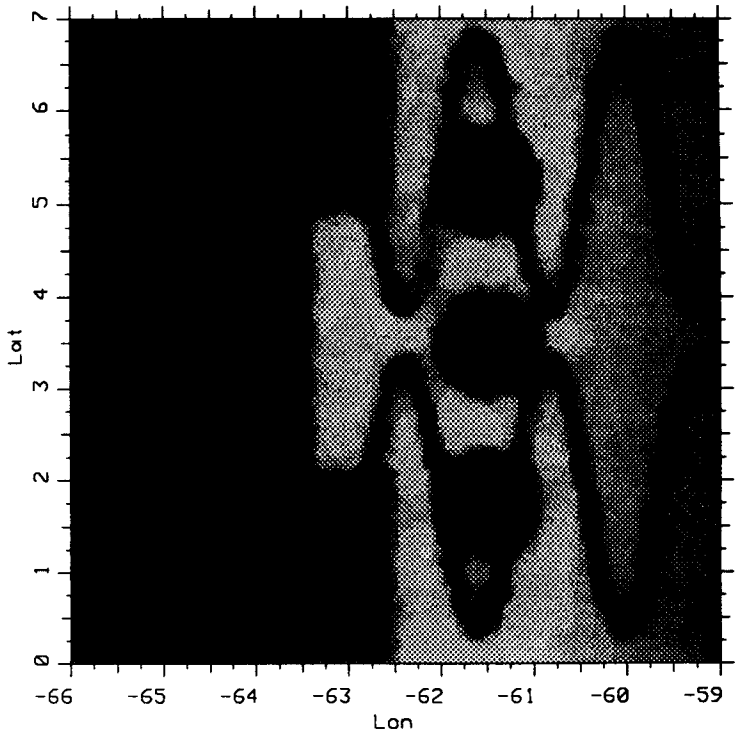

Figure 6.22: Test map - SIRF - noise present - $\mathcal{A}$ image

however, that the edges have not been smoothed and, in fact, appear to have been sharpened in many areas. (The $\mathcal{B}$ image is equivalent to Figure 6.22.)

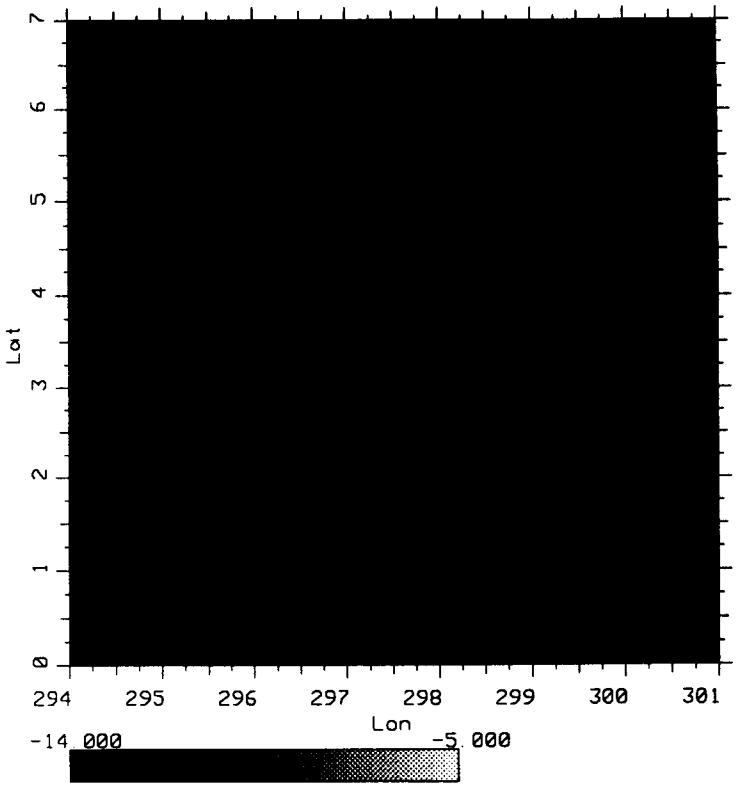

Figure 6.23: LoRes - $\mathcal{A}$ image

6.4 Real Data

Presented next are maps of a small region of the Amazon Basin generated using actual SASS data.

6.4.1 LoRes

Figures 6.24 and 6.25 are images generated using the AVE algorithm with $\mathcal{B}$ estimation. Both images were generated at 0.5° degree resolution.

6.4.2 AVE

Figures 6.26 and 6.27 show the results of the AVE algorithm on the actual data set. It is far superior to Figure 6.24 in that the shape of the darkened region is much more defined.

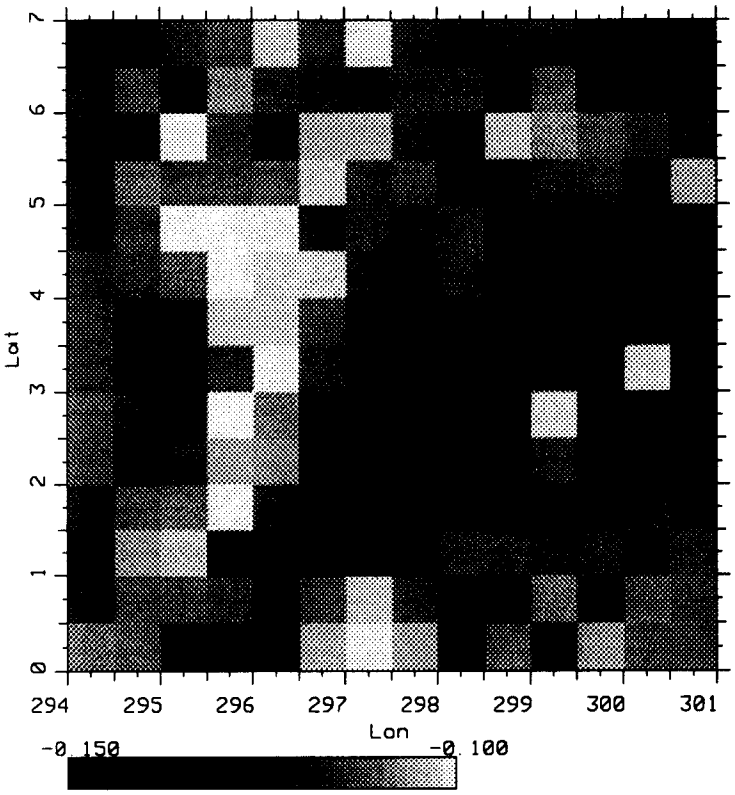

Figure 6.24: LoRes - $\mathcal{B}$ image

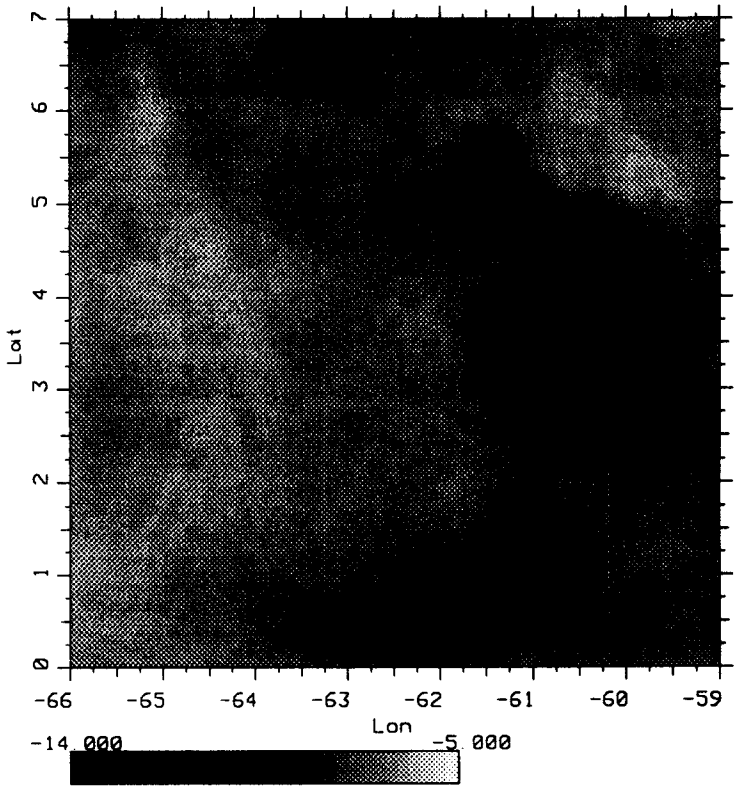

Figure 6.25: AVE - $\mathcal{A}$ image

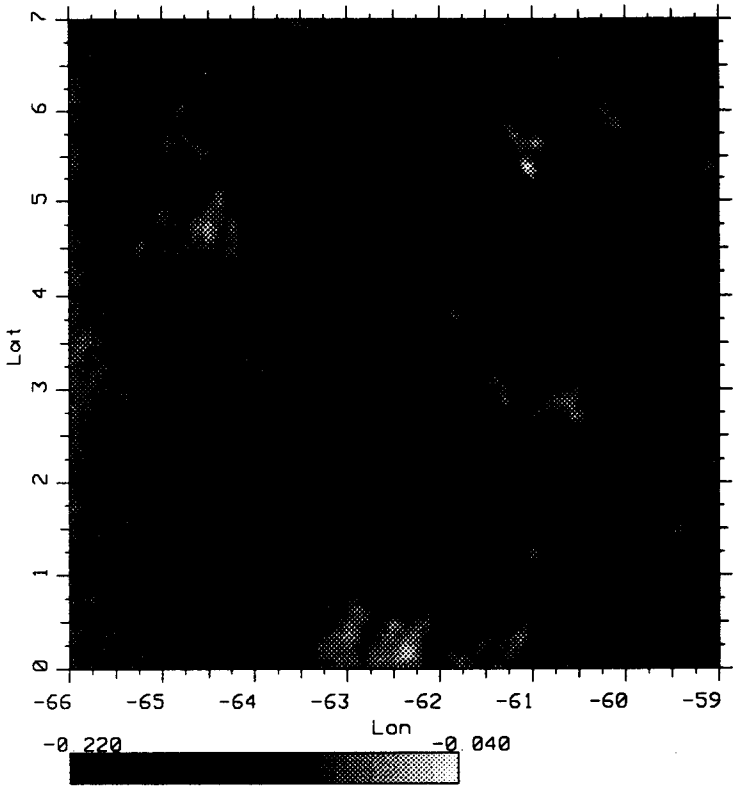

Figure 6.26: AVE - $\mathcal{B}$ image

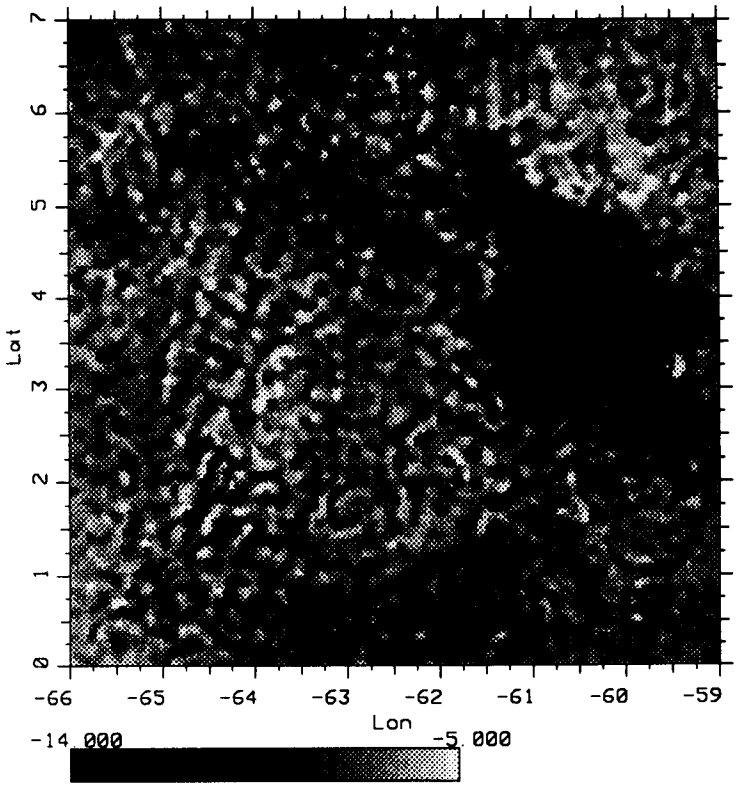

Figure 6.27: Additive ART - $\mathcal{A}$ image

6.4.3 ART

Figures 6.28 and 6.29 are the images generated by the ART algorithms. Each map is the result of ten iterations, due to instability problems. Any resolution enhancement accomplished by either algorithm was quickly overwhelmed by the noise intolerance of both methods.

6.4.4 Block ART

Figure 6.30 shows the output of the block ART algorithm on the real data set. As before, it is smoother than the image generated by the multiplicative ART algorithm, but unfortunately it retains an unacceptable amount of noise. Figure 6.31 demonstrates the effect of damping the scale factor used by block ART.

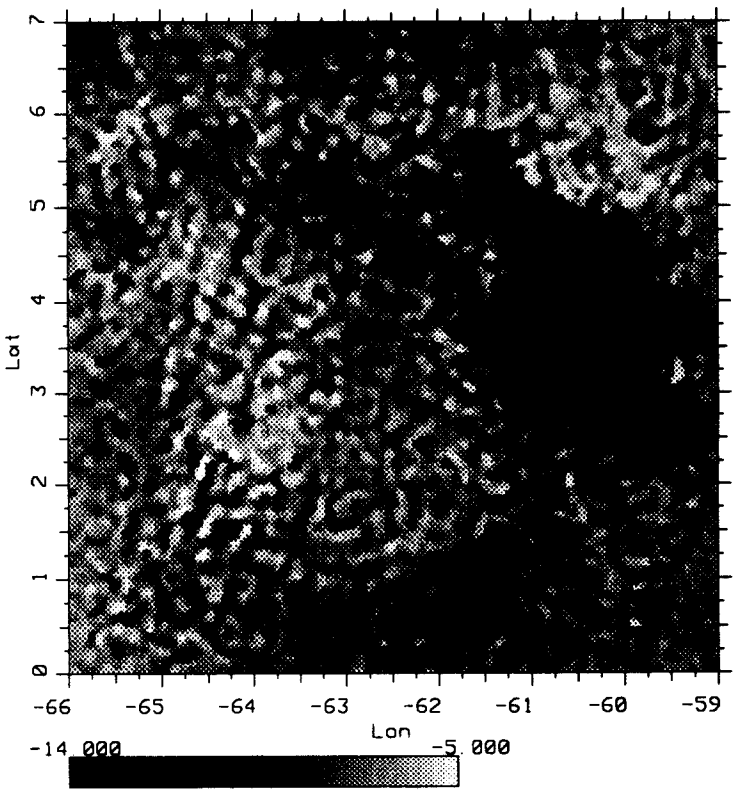

Figure 6.28: Multiplicative ART - $\mathcal{A}$ image

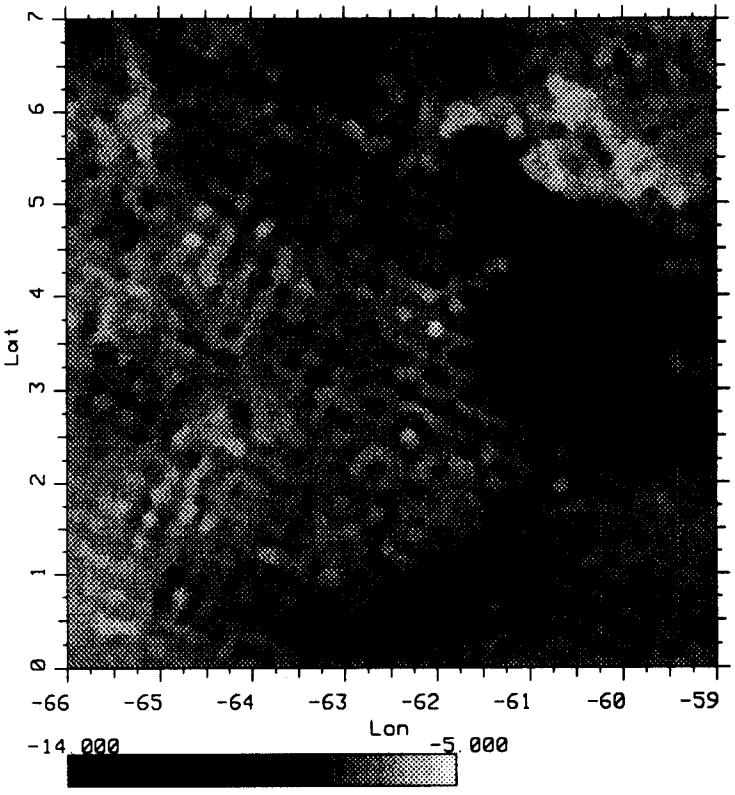

Figure 6.29: Block ART - $\mathcal{A}$ image

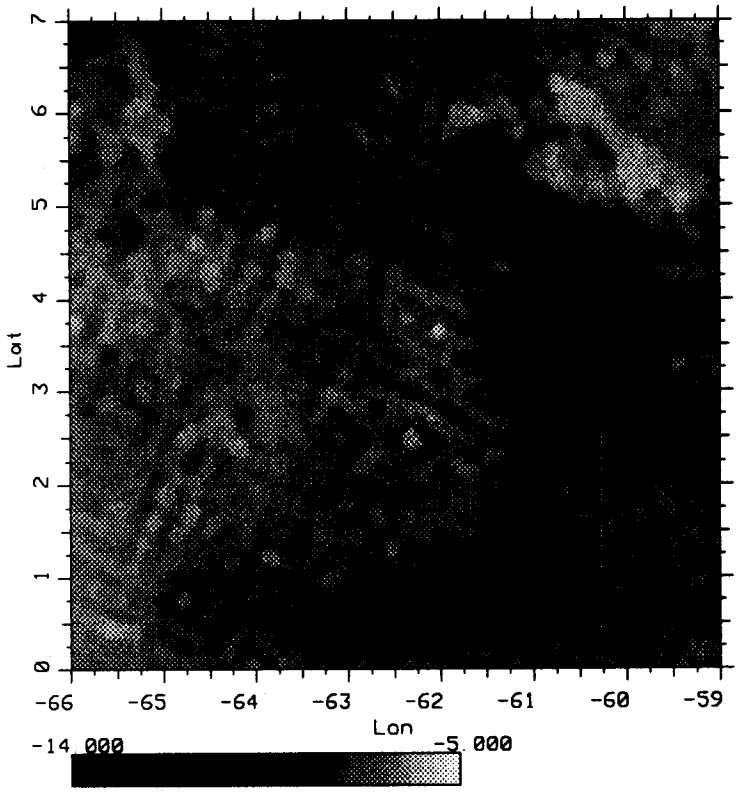

Figure 6.30: Damped block ART - $\mathcal{A}$ image

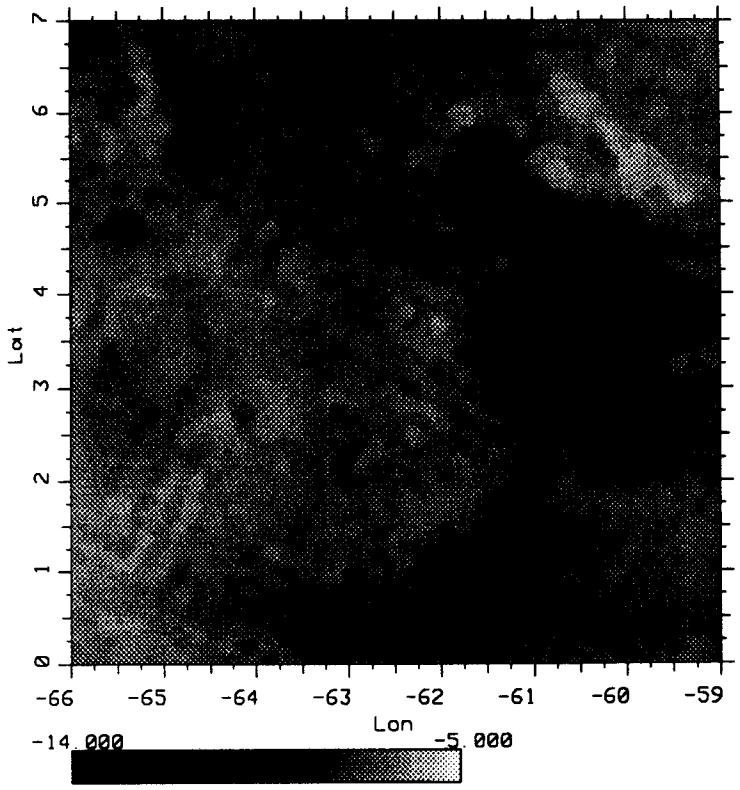

Figure 6.31: SIR - $\mathcal{A}$ image

6.4.5 SIR

Shown next in Figure 6.32 are the results of the SIR algorithm on the real data set. A little more noise is apparent in this image than was seen on the test images, which infers that perhaps the unmodeled noise was higher than the 10% used in generating the synthetic data sets. However, the noise level is tolerable. This image is sharper than the AVE image but includes more noise. As expected there will always be a trade-off between noise and resolution.

6.4.6 SIRF

Figure 6.34 demonstrates the effect of the modified median filter on the real data set. This image has reasonable edge definition, as well as a low noise level. Its visual appeal caused it to be the method selected to image the entire Amazon region.

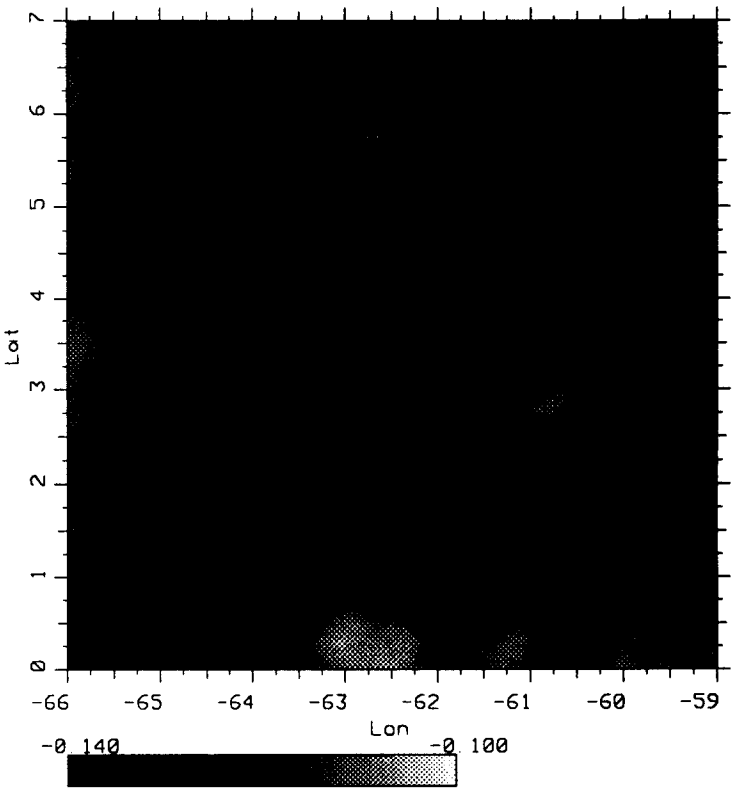

Figure 6.32: SIR - $\mathcal{B}$ image

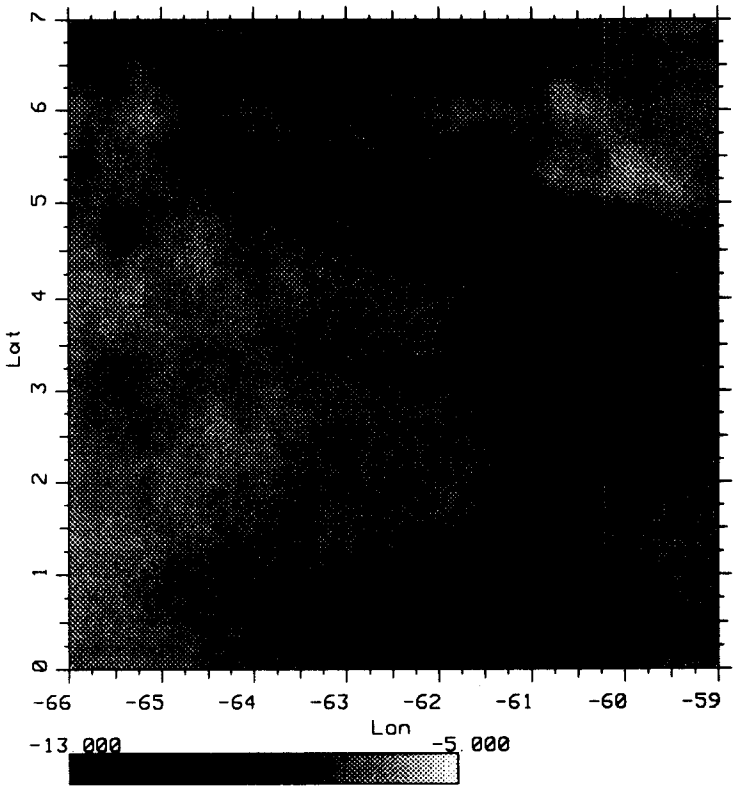

Figure 6.33: SIRF - noise present - $\mathcal{A}$ image

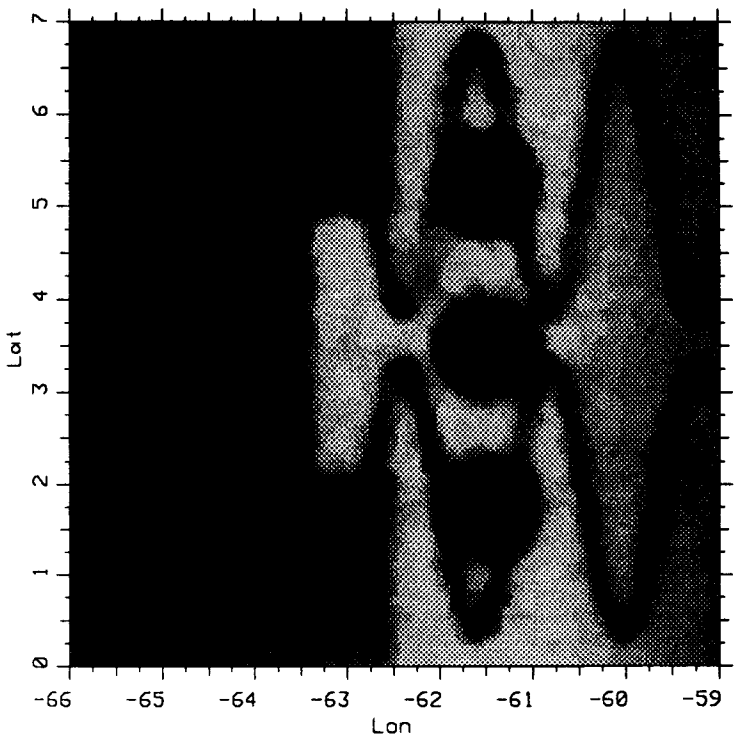

Figure 6.34: Test map - SIR with 0.25 damping power - noise present - $\mathcal{A}$ image

6.5 Comparison of the Effect of Damping Powers

Figures 6.35 and 6.36 show the test map, with noise, processed by SIR using damping factors of 0.25 and 0.75, respectively. These images can be compared with Figure 6.21 to show the effect of the damping powers. Changing the damping power had a limited effect on the images. Since 0.5 (Figure 6.21) gave subjectively better results, it was selected for use in SIR.

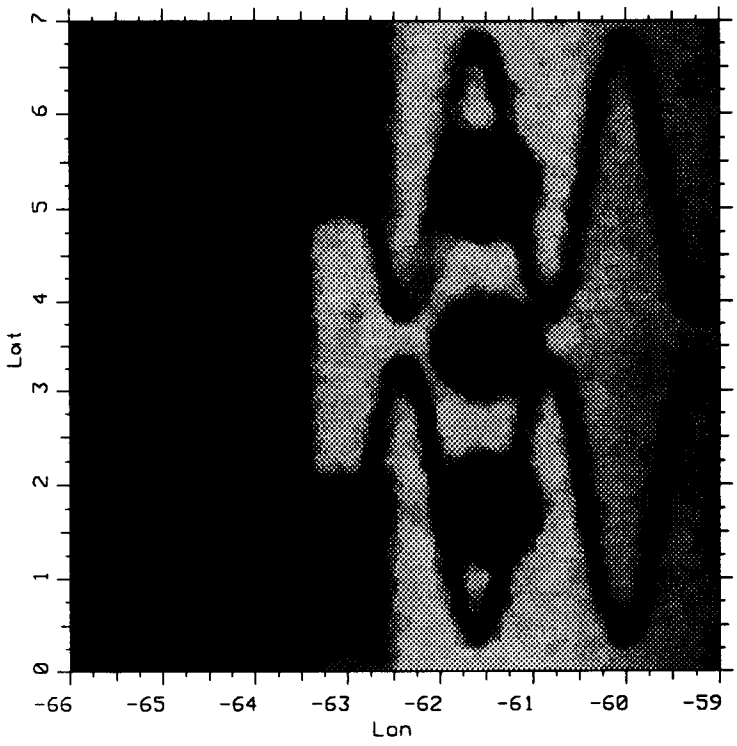

Figure 6.35: Test map - SIR with 0.75 damping power - noise present - $\mathcal{A}$ image

6.6 Full Amazon Region

Figures 6.37 and 6.38 are the $\mathcal{A}$ and $\mathcal{B}$ images of the Amazon basin created with the LoRes algorithm. The extent of the rain forest as of 1978 can be seen as the lighter region near the center of the continent.

Finally, Figures 6.39 and 6.40 show the $\mathcal{A}$ and $\mathcal{B}$ images generated by the SIRF algorithm. Notice that the extent of the Amazon River is more defined in the $\mathcal{A}$ image, as are the regional boundaries between different land covers. The noise pattern in the $\mathcal{A}$ image is probably caused by miscalibration and can hopefully be eliminated after further research. Currently, studies are being performed on the $\mathcal{A}$ image to estimate its vegetation classification accuracy, which is believed to be near 85%.

Figure 6.36: Amazon region, LoRes - $\mathcal{A}$ image

Figure 6.37: Amazon region, LoRes - $\mathcal{B}$ image

Figure 6.38: Amazon region, SIR - $\mathcal{A}$ image

Figure 6.39: Amazon region, SIR - $\mathcal{B}$ image

CHAPTER 7

CONCLUSIONS

Although Seasat flew for only a short period of time, the data it collected has proven valuable in the development of new processing techniques which can be applied to data sets from future scatterometer missions. Over the past year I have been involved in research involving the generation of land cover images from scatterometer land measurements. While such images have been generated in the past, low resolution has hindered their usefulness. It has been my focus to increase the resolution of these land cover images using the added information contained in disjoint spatial overlaps of the measurement cells, in essence, generating an image from its projections.

Traditional methods of image reconstruction from projections are difficult to apply effectively to scatterometer data for two reasons: First, scatterometer data is inherently very noisy, and most traditional reconstruction methods are relatively noise intolerant. Second, traditional reconstruction methods are single-variate, while the data generated by a scatterometer usually requires a multi-variate reconstruction technique. My contribution to this field of research has been to develop a noise-tolerant reconstruction algorithm capable of multi-variate image estimation, Scatterometer Iterative Reconstruction with Filter (SIRF).

The SIRF algorithm combines the non-linear constraining function to improve noise tolerance, the A/B estimation for multi-variate reconstruction, and a non-linear modified median filter. SIRF has been extensively tested on both synthetic and actual data sets, and was used to generate relatively high resolution images (as compared to the intrinsic measurement cell resolution) of the Amazon basin which are currently being used in scatterometer vegetation classification experiments. Hardin and Long have found the images capable of classifying 18 catagories with 57% accuracy. Broading the groupings to encompass only forest, woodland, shrubland, and grassland increases the classification accuracy to 81%. Finally, combining shrubland with grassland yields classification accuracy of 94% [14]. If maximum classification accuracy were desired (for the creation of a 1978

vegitation map of the Amazon region for example) much of the confusion in the purely backscatter-based classification could be removed by incorperating into the classification other know topographical, climatic, and ancillary data sourses. However, the focus of this research was to generate the best possible backscatter-based images possible.

Because scatterometers are low resolution radar instruments, land-based postprocessing will provide a valuable link between the low resolution data and the high resolution images needed for scientific studies.

7.1 Future Research

Future research should involve obtaining a better understanding of the hyper-parameters used in the algorithm, as well as exploring other constraining functions similar to that used in SIRF which might yield better noise/resolution characteristics. Also, large data inconsistencies attributable to calibration error in SASS might be identified and eliminated, thus increasing the resulting image quality. Finally, the applications of SIRF are not limited to scatterometry and can be employed generally in a large number of other fields, as well as in other satellite imaging systems.

BIBLIOGRAPHY

[1] R. Gordon, "A tutorial on ART," *IEEE Transactions on Nuclear Science*, vol. NS-21, pp. 78–93, june 1971.

[2] F. T. Ulaby, R. K. Moore, and A. K. Fung, *Microwave Remote Sensing, Active and Passive*, vol. 1. Norwood, MA: Artech House Inc., 1981.

[3] R. G. Kennett and F. K. Li, "Seasat over-land scatterometer data, part I: Global overview of ku-band backscatter coefficients," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 27, pp. 592–605, september 1989.

[4] D. G. Long, P. Hardin, and P. Whiting, "High-resolution imaging of land/ice using spaceborne scatterometry part I: The imaging technique." Paper to be published, January 1992.

[5] E. M. Bracalente, D. H. Boggs, W. L. Grantham, and J. L. Sweet, "The SASS scattering coefficient algorithm," *IEEE Journal of Oceanic Engineering*, vol. 0E-5, pp. 145–154, April 1980.

[6] R. K. Moore and A. K. Fung, "Radar determination of winds at sea," *Proceedings of the IEEE*, vol. 65, pp. 1504–1521, November 1979.

[7] F. T. Ulaby, "Vegetation clutter model," *IEEE transactions on Antennas Propagation*, vol. AP-28, pp. 538–545, July 1980.

[8] H. J. Trussell, "Convergence criteria for iterative restoration methods," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, pp. 129–136, Feb 1983.

[9] G. T. Herman, A. Lent, and S. W. Rowland, "ART: Mathematics and applications," *Journal of Theoretical Biology*, vol. 42, pp. 1–32, 1973.

[10] K. A. Dines and R. J. Lytle, "Computerized geophysical tomography," *Proceedings of the IEEE*, vol. 67, pp. 1065–1073, july 1979.

[11] P. Gilbert, "Iterative methods for the three-dimensional reconstruction of an object from projections," *Journal of Theoretical Biology*, vol. 36, pp. 105–117, 1972.

[12] Y. Censor, "Finite series-expansion reconstruction methods," *Proceedings of the IEEE*, vol. 71, pp. 409–419, March 1983.

[13] T. Elfving, "On some methods for entropy maximization and matrix scaling," *Linear Algebra and its Applications*, vol. 34, pp. 321–339, 1980.

[14] D. G. Long and P. Hardin, "High-resolution imaging of land/ice using space-borne scatterometry part II: Vegetation studies of the amazon basin and other applications." Paper to be published, January 1992.