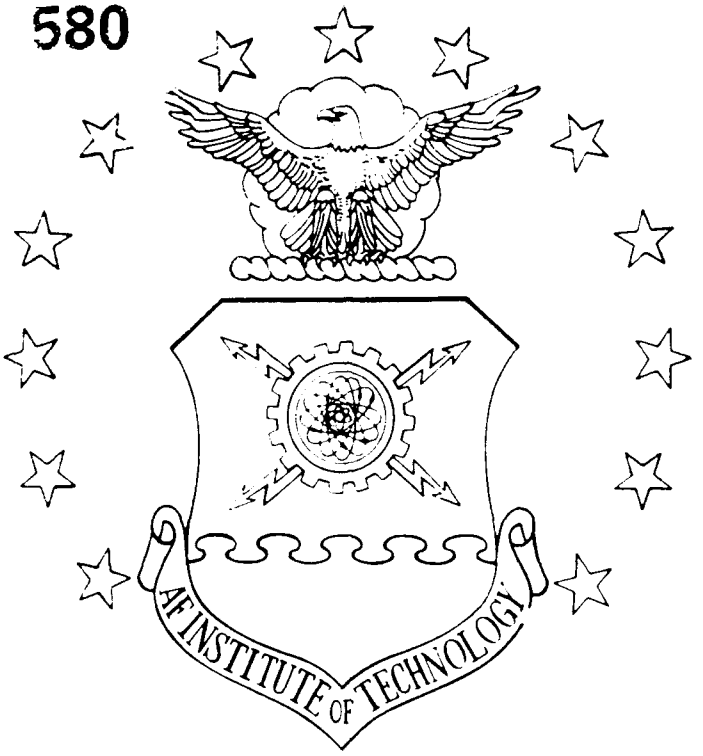

Gabor Filters and Neural Networks for Segmentation of Synthetic Aperture Radar Imagery

THESIS

Albert P. L'Homme
Captain, USAF

AFIT/GE/ENG/90D-31

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/90D-31

Gabor Filters and Neural Networks for Segmentation of Synthetic Aperture Radar Imagery

THESIS

Albert P. L'Homme
Captain, USAF

AFIT/GE/ENG/90D-31

AFIT/GE/ENG/90D-31

Gabor Filters and Neural Networks for Segmentation of Synthetic Aperture Radar Imagery

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Albert P. L'Homme, B.S.E.
Captain, USAF

December, 1990

Acknowledgments

I want to thank the many people who provided support and encouragement throughout my research. Much thanks and gratitude go to my thesis advisor Maj Steven Rogers whose ideas were the foundation of this research. His continued encouragement, helpful guidance and unending enthusiasm focused my efforts. I would also like to thank Dr Matthew Kabrisky and Maj Rogers for their enlightening pattern recognition courses. These courses have opened my eyes to a whole new aspect of engineering.

A deeply felt thanks goes to my sponsors, Kevin Willey and Mike Bryant from the Automatic Target Recognition Technology Group, Aeronautical Systems Division Wright-Patterson AFB OH. Their continued efforts to provide support and suggestions were beneficial in the success of this thesis. The unlimited use of their state of the art lab is highly appreciated. I also would like to thank Capt Joe Brickey for his suggestion to research segmentation of synthetic aperture radar imagery, and for his continued encouragement and helpful comments.

Finally, I want to thank my wife Deanna and children, Albert, Katie, Sarah, and Joey for their love, patience, and understanding. They were my pillars of strength.

Albert P. L'Homme

Table of Contents

List of Figures

List of Tables

AFIT/GE/ENG/90D-31

Abstract

This research investigates Gabor filters and artificial neural networks for autonomous segmentation of (1 foot by 1 foot) high resolution polarimetric synthetic aperture radar (SAR). Processing involved frequency correlation between the SAR imagery and biologically motivated Gabor functions. Methods for selecting the Gabor tuning parameters from the endless choices of frequency, rotation, standard deviation and bandwidth are discussed. Using these parameters, resulting Gabor correlation images were reduced in speckle, and more detailed than the original SAR images. This research used cosine Gabor functions and operated on single polarization HH magnitude data. Following selection of the appropriate Gabor features, multiple Gabor representations were generated and converted for ANN training. Networks investigated were the Kohonen and radial basis function (RBF) algorithms. Provided are results demonstrating a Kohonen network calibration technique and how combination of Gabor processing and RBF networks provide scene segmentation.

Gabor Filters and Neural Networks for Segmentation of Synthetic Aperture Radar Imagery

I. Introduction

1.1 General

For over twenty-five years, the Air Force Institute of Technology (AFIT) has been involved in research to develop an autonomous target recognizer (ATR). The hope is to develop a machine capable of determining from on-board sensor information where hostile targets are located within a scene.

To date, many techniques have been developed which provide limited solutions to object or texture recognition by machines. Complete machine recognition remains an unsolved problem. Limited recognition machines operate in relatively low noise and controlled environments such as supermarkets and industrial production lines. The environment for these machines often control object scale, orientation and position so the features are easier to extract.

The development of a target recognizer is considerably more complex. These machines will typically be housed in missiles, aircraft or spacecraft. In these applications, the environment is no longer controllable which creates new data processing problems. Conditions such as extreme noise and temperature, dusty surroundings, and as well as need of fast processing conditions drastically affect the analysis techniques that can be applied.

The ability of a target recognition machine to scan an image and pull out cultural items, such as targets, requires obtaining good sensor information. Assuming

high quality sensor information is available, templates or algorithms must be established that can separate interesting objects from background clutter. In hostile environments, models often require updating as the topography changes.

In comparison, people are trained from childhood to segment the world. Throughout life, recognition models are updated to classify and associate new objects. This is quite the type processing automatic pattern recognition devices are attempting to provide. Target recognizers must distinguish between background clutter, non-military cultural objects, and specific military targets. Military targets are usually items such as howitzers, tanks, planes and missile launch sites.

Even with the long list of problems requiring solutions, it seems reasonable that machines can be developed to automatically find interesting objects from a scene, focus on them, and categorize them as non-target, target and type target. Popular theory suggests

> the detection and classification of targets is basically a three step process. The first step, is the segmentation of an image to identify potential targets. The second, is the extraction of features of each distinct region within an image. The final step, is the use of these features in the classification process.(20)

Earlier, it was mentioned that methods have been developed which are useful in the target recognition processing. A few of these are

- Taxi, Euclidean and dot product distance rules
- Image Filtering (median and low pass filtering)
- Fast Fourier Transform
- Correlation.

Often, these methods are applied in "ad hoc fashion" (10) to solve pieces of the pattern recognition problem. An example of this, is the AFIT algorithm which

developed a position, scale and rotation invariant (PSRI) feature space. The PSRI problem is common to the area of target recognition.(13)

To get to the target recognition step, first cultural or man made objects must be segmented from the scene and features extracted from these. Typically sensor images are composed of millions of data bytes and most require processing. When multiple pattern recognition techniques are applied to extract objects and features powerful workstations are needed. Application of these techniques in an laboratory environment demonstrates the need for powerful processing. It is understood implementation of these processes in silicon or optics quicken processing time, but even these implementations would fall short of providing real time scene segmentation and image recognition for operational military platforms. With this gap between need and technological feasibility, pattern recognition research has become deeply involved in biological information processing and in particular artificial neural network research.(22)

Today most computers are Von Neumann based binary machines, and quite often measure their performance in million of operations per second. Even so, it seems apparent Von Neumann machines may never provide real time performance on massively complex pattern recognition processes.

Modern technology often compares computer processing capability to biological processing. Today artificial neural network research is attempting to mimic some of these processes. Many alternate computer architectures have been investigated and compared to neural processing. In 1943, McCulloch and Pitts published a paper discussing problems associated with computer science and brain modeling theory.(19) They presented models where output was based on receiving inputs above a certain threshold.(19) Over the years, research has continued into processing based on this type of weighted computation and is the basis for artificial neural networks (ANNs). Until recently ANNs weren't believed to be implementable. However, present day computing power has allowed simulations of artificial neural processing to advance

at an amazing rate.

As mentioned above, ANNs provide a method of computation based on weighted summations of input data rather than the binary calculations processed by Von Neumann machines. ANNs are usually based on some mathematical algorithm attempting to achieve macro or micro-functions found in a biological process. In biological processing, neurons are the basic information processing cell. These cells are densely interconnected to allow for parallel information processing. Output from biological cells provide approximate analog results based on summation or combination of the weighted inputs.

With the uncertainty of Von Neumann type processing to achieve an independent machine target recognizer, ANNs are being researched to determine their applicability. Past research here at AFIT has demonstrated the usefulness of these networks in target recognition.(20) A comparison was made between the recognition results of conventional versus ANN recognizers. It was determined ANN recognizers performed equally well in the identification process.(20) With results found to date, densely connected neural network processors by themselves or in combination with traditional processor techniques show promise for implementation as machine recognition systems.

This thesis is part of the continuing ANN research into the automatic target recognition (ATR) problem. Investigation is to determine features and processes useful for segmentation of radar imagery. Research here combines the biologically motivated Gabor transform and ANNs. These techniques are evaluated in combination to determine their are validity to the radar imagery segmentation process.

1.2 Problem Statement

This thesis applies biologically motivated pattern recognition processes to segmentation of high resolution SAR data. The two main processes are scene correlation with two-dimensional Gabor filters and scene separation using artificial neural net-

works. The major thrust of this research investigates the combination of Gabor filter texture matching with artificial neural network data separation techniques. Segmentation here attempts to find features capable of separating different natural regions and identifying cultural regions from images. To accomplish these tasks, many questions must be answered. These include:

- Is there a particular orientation, pitch or combination of Gabor filter(s) which correlate well with trees, and still others with fields?
- How are these frequencies, rotations, and combinations selected?
- What size Gabor filter provides the best segmentation of this imagery?
- What is the best method to train the Kohonen network to maximize data separation?
- Is there a particular Kohonen neighbor and conscience rule which combines to quicken or increase the separation provided by this type net?
- How is the Kohonen network calibrated?
- Does a Kohonen/radial basis function hybrid network provide automatic calibration of the Kohonen and data classification?
- What are the proper training features for ANN input?

1.3 Background

The data used during this research was collected using a high resolution synthetic aperture radar (SAR) sensor. The data was been provided by the Massachusetts Institute of Technology (MIT) Lincoln Labs through Wright Research and Development Center (WRDC/AARA), and was collected as part of the Advanced Detection Targeting Sensor (ADTS) tests conducted at Stockbridge New York and Portage Lake Maine. The data was collected during ADTS Mission 85 and 98. The sensor used was a 0.3 m by 0.3 m resolution, full polarimetric SAR operating in the stripmap mode. The data is expressed in complex format (in-phase and

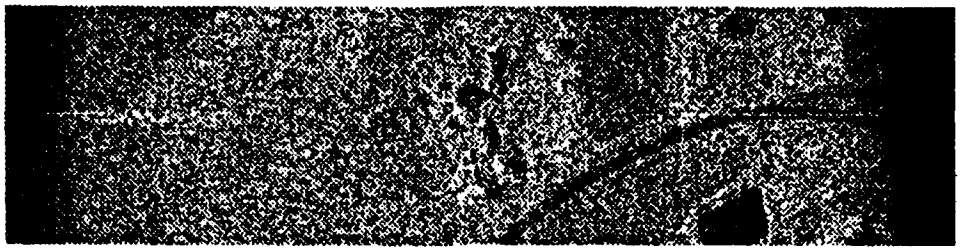

Figure 1. Sample ADTS Image

quadrature). Ground truth information was not available for this imagery; however, a limited number of photographs were available to aid in determining the segmentation effectiveness of the researched techniques. Figure 1.3 is an example of the SAR imagery to be operated upon during this research.

Unclassified 1 foot by 1 foot high resolution SAR imagery is rare, and as seen from 1.3, the majority of the provided data set imagery is composed of landscape and occasional man made items. This image is of a horse ranch with a nearby road and pond (lower right). Of the data available, this was the only image providing some form of cultural structure. Most images contained forests, fields, and shadow with occasional roads, or corner reflectors.

This thesis basically performs feasibility tests to determine the usefulness of combining ANNs and Gabor filters for radar imagery region segmentation. It is hoped segmentation of the following regions can be provided: trees, fields, shadows, ponds, roads, and cultural objects.

1.4 Definitions

Segmentation is the process of dividing up a scene based on the image structure to identify areas of interest.

Distance rules are used to measure data separability and for determining winning nodes within a ANN environment.

Feature extraction involves methods for enumerating specific traits of objects or image areas. Feature extraction is a main factor in ANN classification.

Correlation is a measure of similarity between two objects.

A Kohonen neural network refers to an algorithm developed by Teuvo Kohonen which is claimed to promote data self organization. Self organization is a process that occurs in higher levels of learning.(12)

Learning rule is an algorithm used to calculate and update weighted inputs to neural nodes.

Network convergence refers to the ability of the neural network to train and learn a separation process.

Conscience is a network node monitoring process which checks for excessive winning by the nodes. If a node wins excessively, it is suspended from competition for a period of time. Conscience is often used in Kohonen networks.

Neighborhood update rule is a weight updating technique commonly used in Kohonen network training. That is, using a neighborhood update rule modifies the weights of a block of nodes within the current neighborhood of the winner. This technique sometimes helps the convergence process.

Radial basis functions are a form of competitive learning where weight updates are proportional to how well the competitor performed in competition.(16) Typically these networks place radial function in the feature space to divided up the data.

1.5 Scope

A major focus of this research investigates which combinations of Gabor filters provide the best ANN input feature set. Following selection, these feature vectors are applied to Kohonen, and radial basis function networks for clustering and class identification.

1.6 Assumptions

It is assumed:

- Segmentation can be achieved using magnitude only of the HH SAR polarized image.
- Separate spatial frequencies can be determined which strongly match each of the regions.
- Kohonen networks provide clustering that aid data class identification.

1.7 Hypothesis

- Gabor processing and ANNs will provide SAR imagery segmentation.
- Combining selected Gabor frequencies and orientations are valid features to train ANNs for image segmentation.
- The ANN trained Gabor features are robust enough to be used to segment other SAR images.

1.8 General Approach

Initially, investigation will determine which set of Gabor filters provides the best input features for ANN training. A bank of these filters of various orientation, pitch and frequency are correlated with a selected image and saved as Gabor correlation coefficient data files. The resultant files are combined into vectors of pixels and fed into the network for training. These features are used to train Kohonen and radial basis function networks. Once network training is complete, the output weights are remapped to a segmented grayscale image. An additional calibration step is required for remapping trained Kohonen weights to grayscale images. Following training, other images can be Gabor processed through correctly trained weights and segmented.

1.9 Sequence of Presentation

Chapter II provides a literature review of Gabor transform research, and discusses the Kohonen and radial basis function networks as implemented.

Chapter III presents methodology used for this research. Provided is a description of the Gabor equations applied, techniques to determine the Gabor spatial frequency and orientation, Kohonen network equations, and radial basis function methodology.

Chapter IV discusses the results and analysis of this research.

Chapter V presents conclusions and recommendations for further research.

II. Literature Review

2.1 Introduction

This chapter reviews literature found useful for application of the Gabor function, and radial basis functions. The perspective of interest is for application of these techniques to the segmentation of high resolution synthetic aperture radar imagery. Recall, segmentation is the processing of data to separate objects of interest from background regions, or textures. Pattern recognition has been described as a "combination of ad hoc steps " put to use to solve a particular data problem.(10:2) Some of the various techniques used during this thesis include:

- Euclidean distance rule is used to measure the separability of data.(10:6-7) The Euclidean distance is implemented in the Kohonen network for the distance calculations between the Kohonen layer weights and input vectors.
- Median filtering, Gaussian blurring, maximum filtering, minimum filtering, and equalization are used to evaluate the segmentation provided by single Gabor filter representations of a given SAR image.
- Fast Fourier Transforms (FFTs) convert the original image and Gabor filters to their frequency domain representations for the correlation process.
- Artificial Neural Networks are applied to learn the underlying structure of the resulting Gabor data for clustering and class categorizing.

Recent pattern recognition research here at AFIT is keenly involved in neural networks and the Gabor transform. Both of these techniques are motivated by biological processes. Neural networks are based on the weighted communication between neurological cells.(19:21-30) This interneuronal communication is believed

to be the basis of thought processes of the brain. Artificial neural networks are an attempt to process data in a fashion which mimics macroscopic or microscopic functions of the brain. This is not meant to imply that ANNs implement algorithms identical to those operating in our brains, but are simply trying to perform similar data operations. For example, the perceptron's non-linear function is just that, a non-linear function, and does not implement the exact algorithm which relays data between biological neurons.(19) As with neural networks, the Gabor function has found merit in neurophysiology. Specifically, interest in the Gabor function results from recent research demonstrating its close approximation to measured responses in the visual cortex of cats.(25)

This review covers literature concerning the topics mentioned above. Provided is a background discussion on the origin of the Gabor function, its connection to the visual cortex, and usefulness to pattern recognition. Literature covering radial basis functions (RBF) is presented to describe typical implementations and results from past RBF research.

2.2 The Gabor Transform

2.2.1 **Discovery** Dennis Gabor published his three part article "Theory of Communications" in 1946. In part one, Gabor described a new analytical technique for the communications field. He felt our intuitive sensations, especially the auditory tract, make concurrent use of both time and frequency information. Even though there are benefits to conventional analytical techniques, Gabor felt simultaneous description of signals both in time and frequency would provide additional information not available when analyzing signals as mutually exclusive events. Upon this basis, Gabor proposed his "elementary functions". He provided substantiating proof showing any signal could be represented by using these functions. His mathematical

proof made use of quantum theory and Heisenberg's time/frequency relation,

$$\Delta t \Delta f \approx 1 \tag{1}$$

In his article, Gabor defined the following equations as the basis for his elementary functions:
the time relation:

$$\psi(t) = exp[-\alpha^2(t - t_o)^2]cis(2\pi f_o t + \phi) \tag{2}$$

the frequency relation:

$$\Psi(f) = exp[-(\frac{\pi^2}{\alpha^2})(f - f_o)^2]cis[-2\pi t_o(f - f_o) + \phi] \tag{3}$$

where:

$$cis(2\pi f_o t + \phi) = exp[j2\pi f_o t + \phi] \tag{4}$$

where α refer to the pulse sharpness, t_o to the time when the pulse peak occurs, f_o to the sinusoidal frequency, and *phi* the phase of the sinusoidal modulator. Gabor noted that

> these elementary signals assure the best utilization of the information area. They possess the smallest product of effective time duration by effective frequency width giving these functions optimal reciprocal resolution.(8)

Gabor's goal was to find a method which made more efficient use of the frequency bands. With this in mind, Gabor investigated the reception, and data recognition rate of the human ear. In his second paper, he described experiments performed on the recognizability of speech. He found the maximum recognition rate of the human ear to be dependent upon the signal frequency. One of his findings showed the human

ear only distinguishes about 42 % of the information presented in the frequency range from 0 to 1000 hertz. This rate dropped off nonlinearly to about 15 % at 8000 hertz. He felt this attribute of the auditory system could be tricked by some technique. (8) With this notion, Gabor went to work to determine an information compression technique to meet both human auditory and frequency band specifications.

In his third paper, Gabor presented both a mechanical and an electrical technique for speech compression and expansion. Based on his proposed elementary functions, both techniques proved to be effective methods for signal compression and expansion with little loss in intelligibility. (8) Nearly thirty years passed before Gabor's work was extended to two dimensional image processing.

2.2.2 **The Biological Connection** Recently, "vision research has been enlivened by debate" over the fundamental character of the visual system. (5:1160) Daugman felt Dennis Gabor's work provided the clearest insight to the mathematical representation of the visual system. (5:1160) Porat and Zeevi view the mathematical completeness of the Gabor scheme, and the closeness of fit to the localized frequency and phase response of physiological and psychophysical processing, make the Gabor a worthy candidate as a computer vision algorithm. (17:432) Daugman and others have demonstrated how this family of two-dimensional (2D) filters provide an accurate description of the "various 2D receptive-field profiles encountered in simple cells within the mammalian visual cortex ". (5:1168) Webster and De Valois found this to be impressive considering the diversity of these cells. However, they too have matched the Gabor like characteristics of these cells, and believe this model to be sufficiently general to account for this variety. (25:1130) Jones and Palmer found the Gabor function, and the response of simple cells within the cat striate cortex to be indistinguishable in 33 of 36 tests. (9)

As a point of clarification, Gabor's mathematical notation is no longer used by researchers. Today, the one dimensional Gabor time relation is commonly written

as

$$h(t) = exp[-[t^2/(2\lambda^2\sigma^2)]]cos(2\pi Ft) \tag{5}$$

where t is once again the time of occurrence of the peak, F the frequency of the modulator, σ the Gaussian decay term and λ the aspect ratio. (3:57-58) Daugman expanded this into a two-dimensional relations for both the spatial and frequency domains.

Daugman's spatial relation:

$$\begin{aligned} f(x,y) = exp[-\pi[(x-x_o)^2\alpha^2 + (y-y_o)^2\beta^2]] \\ exp(-\pi j[u_o(x-x_o) + v_o(y-y_o)]) \end{aligned} \tag{6}$$

Daugman's frequency response:

$$\begin{aligned} F(u,v) = exp[-\pi[(u-u_o)^2/\alpha^2 + (v-v_o)^2/\beta^2]] \\ exp(-2\pi j[x_o(u-u_o) + y_o(v-v_o)]) \end{aligned} \tag{7}$$

Daugman defined the 4-D fundamental uncertainty principle as "the theoretical lower bound of joint uncertainty... in the visual space and spatial frequency domains" as:

$$(\Delta x)(\Delta y)(\Delta u)(\Delta v) \geq 1/(16\pi^2) \tag{8}$$

Daugman noted the 2-D Gabor filters achieved the lower bound of this inequality. (5)

Up to 1985 there had been only limited research into matching the 2-D Gabor filter with the striate cortex. However even at that time, researchers felt the four degrees of freedom known to be associated with visual cortex seemed to be modeled rather well by the 2-D Gabor representation. (5:1168) Continued investigation consistently supports this function to model the visual cortex and hence machine vision.

2.2.3 **Texture Discrimination** Extending use of these filters to texture discrimination (3), computer vision (17), and the detection of transient signals (7) has been a logical undertaking. Each of these processes can take advantage of the information encoding provided by the Gabor family of filters. Some benefits to be gained from these filters are their ability to increase the signal to noise ratio, provide for signal compression and expansion, and information representation for further processing. (3:62)

> At a global scale of perception, texture is regarded simply as a carrier of region information. Within this framework, the local structure of texture is described by the orientations and frequencies of the carriers, whereas information describing the spatial extent of the texture is contained in the envelopes of the channel outputs.(3:56)

Research by Bovik and others has shown that using an appropriately dense field of Gabor filters encodes an image into subimages which can be precisely recovered. The cost associated with using these filters results in the loss of the high frequency or local structure of the image; however, the spatial description of the texture held within the channels is maintained. Comparison of these channels provides image texture segmentation. Bovik and others noted this segmentation process can be simplified by using a hierarchy of filters to reduce the sampling density before performing channel comparisons. (3)

Porat and Zeevi developed a scheme for image representation (machine vision) in a combined space/spatial-frequency domain using the optimal properties of the Gabor function. They employed the biorthogonal function of Bastiaans (2) as a technique to sample both one and two dimensional signals. Porat and Zeevi showed that using only six Gabor coefficients allowed high quality reconstruction of both periodic and aperiodic signals. They also found under certain conditions Gabor coefficients provide for better reconstruction of aperiodic signals than periodic. They attributed this to a mismatch between the Gabor sampling frequency, and the frequency of the aperiodic signal. (17)

Studying the affects of Gabor quantization on a signal's phase, Porat and Zeevi showed as little as five quantization levels provided good reconstruction (approximately 93 %). However, as the number of quantization levels increased, better results were made. Twenty-four levels yielded about 99.7 % accuracy. One particular experiment looked into the oversampling of the time dimension while correspondingly decreasing the sampling along the frequency dimension. They noted a tradeoff between the number of spatial and frequency samples necessary to effectively sample and reproduce the original waveform. At this point, they seemed reluctant to attribute this tradeoff to the uncertainty principle. Rather, they felt it more a necessity to maintain localized spatial and frequency characteristics. To expound this point, Porat and Zeevi considered a d.c. signal. They found this type signal requires quite a number of high frequency Gabor coefficients for a good approximation. They pointed out, once the global frequency band of the signal is properly sampled then an exchange can be made between the out of frequency band components and additional spatial samples. (17)

With this result, their research carried them to the 2-D world of image processing. Their next experiments were performed using the polar representation, and made use of a limited number of quantization levels to ease the computational load. Once again, their results confirmed the space/frequency tradeoff, and found image

quality to be based on the dimensionality of the image as well as the degrees of freedom available in the Gabor representation. (17)

Benjamin Friedlander and Boaz Porat looked into the detection of transient signals using Gabor filters. They too made use of Bastiaans one-sided exponential Gabor representation. They felt this representation to be better suited to the detection of random signals. Using this technique, Friedlander and Porat sought to determine transient signals in the presence of noise while varying transient parameters such as arrival time and frequency of occurrence. Additionally, parameters of the Gabor detectors were varied to include the sampling interval, aspect ratio, and the number of Gabor coefficients. (7)

In their analysis, Friedlander and Porat looked to determine the affect of each Gabor function parameter change on the detection capability. In each case presented, Friedlander and Porat showed Gabor functions provided extremely good transient detection. In four of the seven experiments, the detection was nearly perfect. However, as the parameters become more mis-matched (that is window mismatch, frequency and noninteger arrival time) caused a noticeable decrease in the signal to noise ratio. Even with this type of mismatch, the transient peaks were well above the noise. The most drastic change resulted from variation in frequency rather than in arrival time or window variations. They concluded that for transient detection, a block or sequence of detector cells would provide for a more "robust" detector. (7)

2.3 Radial Basis Functions

2.3.1 **What are RBFs...** Radial basis functions (RBF), according to Moody and Darken (14), are a departure from the traditional McCullough and Pitts neuron. A McCullough and Pitts neuron provides summation of vector inputs through a threshold device to determine its boolean output. The radial basis function neuron represents localized Gaussian regions (receptive fields). RBF activation is provided by placement of the Gaussian centers (mean) and establishment of its inclusion

region (standard deviation). Moody and Darken, note these overlapping regions are reminiscent of those localized receptive fields found in numerous regions of the cortex. (14:131-134) Research performed by Moody and Darken used radial basis functions for prediction of the Glass-Mackey time series. They demonstrated RBFs require less training time while still achieving accuracy equivalent to the multi-layer perceptron.

Nowlan introduces the difference between soft and hard RBF network learning algorithms. Hard learning was defined to be updating weights based on the winner take all, and soft referred to weight updates based proportionally to the present input vector's strength. Nowlan made use of these functions in a one hidden layer network. The input to the hidden layer used a radially symmetric function (RBF) to compute the distance between the current input vector and hidden layer node weights. The output layer used a linear conversion of outputs from the hidden layer. (16:1-2)

Nowlan also described techniques for placing the RBF centers. One places the centers based upon adaptation from the input vectors using a K-means center selection then "adjusting the size of the RBFs for smoother interpolation". A second technique is a variation of the k-means rule using a two step process to make assignment based on the closest mean then recalculating the means as an average of the samples within its class. This last version is known as the Batch version. (16:2-6)

RBF training is accomplished using bidirectional information. That is the RBF centers are determined by some clustering algorithm (k-means, Kohonen ...) while the receptive strengths are found using feedback from a least mean square (LMS) rule. (14:136)

2.3.2 **Past Research Results** Radial basis functions have been used in time series predictions, hand written digit identification and vowel utterance recognition systems. Moody and Darken evaluated use of RBFs to estimate the Mackey-Glass chaotic time series. Nowlan made use of RBFs for digit recognition. Nowlan made

use of various combinations of soft, hard, spherical, and elliptical Gaussian RBFs and found the soft Gaussian resulted in percentages of correct classifications approaching those obtained by a linear back propagation net. Results of the digit recognition experiment showed the soft learning rule gave best RBF results coming only fractional percentage points shy of the back prop net. This improved performance was true across all RBF variations (spherical or elliptical Gaussians). In Moody and Darken's vowel recognition experiment, the soft classifier again performed better than the hard. However, use of normalization within the hard classifier scheme provided half the improvement between the percentage correct for the hard and soft classifiers.

2.3.3 **Conclusions Presented from Prior Research** Moody and Darken concluded that when data is abundant RBFs are able to achieve accuracies achieved by back propagation nets; however, for scarce data back propagation is their preferred technique. (14:141) Nowlan concludes that exact or soft maximum likelihood techniques out perform the winner-take-all. The increase in accuracy comes with only slightly longer processing times. Normalization also seems to improve RBF's performance by a couple of percentage points. Nowlan points out that the digit classification problem illustrates how RBFs can be used in high dimensional input space. (16:9-11)

2.4 Conclusions From the Literature

From its basis almost half a century ago to its renaissance in vision research, the Gabor function continues to show promise as an effective tool. Dennis Gabor demonstrated how his elementary functions could make more efficient use of the frequency spectrum by compressing then expanding communication signals. This literature review has shown some substantiating proof of the usefulness of the Gabor transform for vision research, texture discrimination, machine vision, and transient signal detection.

cluster data. Their speed in the training process and ability to be used in high dimensional and abundant data applications make them attractive for use in image segmentation.

III. Implementation Methodology

3.1 Introduction

This chapter describes the processes used during this thesis. It develops the Gabor equations as presented by the Bovik article (3), and discusses filter selection techniques. This is followed by a description of the Kohonen network equations used and the plan for Kohonen layer calibration. Additionally, the radial basis function network implementation is described. Finally, the overall system is presented from image input and network training to image segmentation and testing.

Recall, the data operated upon during this research is high resolution (1foot by 1foot) Synthetic Aperture Radar imagery. Each frame of data is presented in four files where each file is the complex (quadrature) representation of a single polarization (HH, HV, VH, VV). This thesis concentrates on the texture matching ability of Gabor filters. Therefore, the data is combined from its complex form into magnitude only. Additionally, the HH polarization was selected for all tests.

3.2 Gabor Implementation

As discussed previously in chapters one and two, the visual cortex of mammals has been found to be closely approximated by Gabor functions. Prior machine image segmentation research has applied these filters to the texture discrimination problem.(24, 3) A general purpose representation of the Gabor function defined by Bovik and others (3) is presented below and is quite similar to that used by Mueller and Fretheim (15) in VLSI reverse engineering.

$$h(x,y) = g(x'y')\exp[2\pi j(Ux + Vy)] \tag{9}$$

where:

$$x' = x\cos\phi + y\sin\phi \tag{10}$$

$$y' = -x\sin\phi + y\cos\phi \tag{11}$$

x' and y' are the Gabor rotational coordinates with ϕ referencing the filter's major axis. The major axis refers to the orientation of the longer axis of the filter (non-symmetric filters).

The Gaussian envelope with the aspect ratio λ accounting for envelope variance changes in both spatial directions is determined using:

$$g(x,y) = 1/(2\pi\lambda\sigma^2)\exp[-((x/\lambda)^2 + y^2)/(2\sigma^2)] \tag{12}$$

This thesis uses only circularly symmetric filters, $\lambda = 1$. This symmetry eliminates the need to define ϕ the major axis. As a result, $g(x',y')$ reduces to

$$g(x',y') = 1/(2\pi\sigma^2)\exp[-(x^2 + y^2)/(2\sigma^2)] \tag{13}$$

and allows $h(x,y)$ to be expressed as:

$$h(x,y) = 1/(2\pi\sigma^2)\exp[-(x^2 + y^2)/(2\sigma^2)]\exp[2\pi j(Ux + Vy)] \tag{14}$$

Converting the second exponential term using:

$$\exp(j\theta) = \cos\theta + j\sin\theta \tag{15}$$

gives the quadrature representation:

$$h(x,y) = 1/(2\pi\sigma^2)exp[-(x^2+y^2)/2\sigma^2](cos[2\pi(Ux+Vy)] + jsin[2\pi(Ux+Vy)]) \tag{16}$$

Finally, breaking this into its in-phase and quadrature components, results in the following component representations of Gabor filters:

$$h_c(x,y) = 1/(2\pi\sigma^2)exp[-(x^2+y^2)/2\sigma^2]cos[2\pi(Ux+Vy)] \tag{17}$$

$$h_s(x,y) = 1/(2\pi\sigma^2)exp[-(x^2+y^2)/2\sigma^2]sin[2\pi(Ux+Vy)] \tag{18}$$

Note, these filters are only approximately in phase quadrature. Their tunability results from adjustments of the modulating sinusoidal frequency (F), Gaussian envelope standard deviation (σ) and orientation bandwidth (B). The frequency F is defined as $F = \sqrt{U^2+V^2}$, and bandwidth is defined below.

Bovik and others (3) presented methods for determining the best frequencies and orientations to be used for image segmentation. They suggest choosing the lowest fundamental frequency for periodic texture segmentation, and two largest for aperiodic textures. (3:63) Due to the random nature of trees and fields the center frequencies are selected based on the strongest peaks founded within one half of the FFT plane. It is assumed here, this texture is highly "aperiodic".

Bovik suggests the following relationships can be used to determine the radial bandwidth and standard deviation of the Gaussian envelope. The radial bandwidth, B is determined by:

$$B = log_2[(\pi F\lambda\sigma + \alpha)/(\pi F\lambda\sigma - \alpha)] \tag{19}$$

where $\alpha = \sqrt{ln(2)/2}$.

The orientation bandwidth, σ is defined as :

$$\sigma = (\alpha/(\pi F)(2^B+1)/(2^B-1)) = 1 - 2^{-B} \tag{20}$$

These bandwidth calculations are helpful in tuning the filters once the frequencies are determined. Bovik suggests that as the center frequency is increased the bandwidth should be decreased. This in turn requires increasing the number of filters needed while reducing spatial resolution. The Bovik research used radial bandwidths of 0.7, 1.0 and 1.3. These bandwidths are used to guide the selection of Gabor filter bandwidths for this thesis.

3.3 Kohonen Clustering

The Kohonen network implemented here was written by Wayne Recla and Gary Barmore.(1) Their network implements the Kohonen algorithm as described by Dr. Richard Lippman. (12) This C programming language implementation provides numerous selections for neighborhood and gain reduction rules, and measures dot product rule separation between input vector and network layer weights. For this research, the code was modified to calculate a Euclidean distance measurement between input vectors and weights. Additionally as a baseline, the parameters selected for training were set as follows:

1. Network size 16 x 16 nodes.
2. Training vector size 4 elements.
3. Input vectors 1048576 (a whole image of vectors).
4. Conscience value 1.5 (low conscience).
5. Initial seed of 8 (initialized weights are between 0 and 0.5).
6. Exponential gain reduction rule for neighborhood weight updating.

Figure 2 illustrates the Kohonen network. Shown here are the input vectors, X_i, at the bottom feeding connecting weights, $w_{j,i}$. The connecting weights are initialized to be within the range of the input vector elements (the data). The Euclidean distance is calculated between the current input vector, $[X_1, X_2, ...X_n]$,

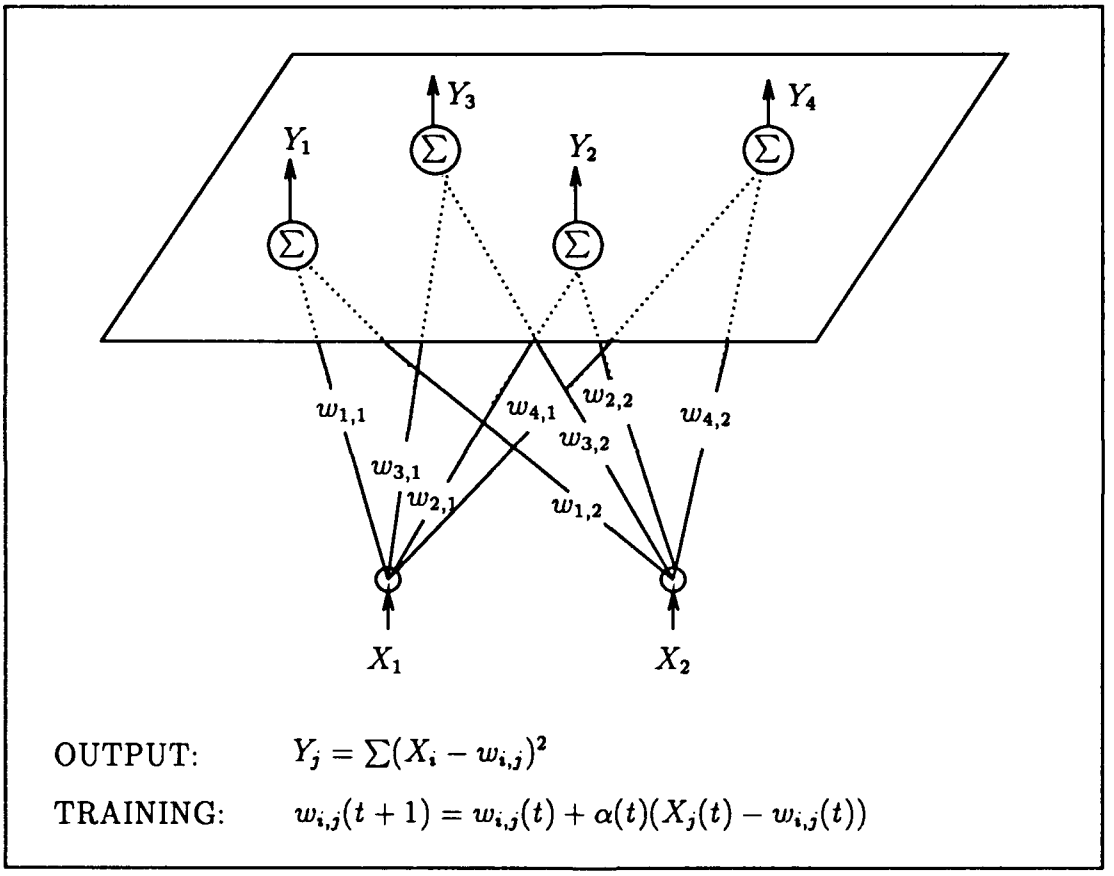

Figure 2. Kohonen Neural Network Architecture (21:19)

and each connecting node to determine the winning layer node. The layer nodes are shown in the rectangular region.

The learning or weight update rule is described below. The Kohonen weights are updated by adding the previous weight, w_{ji}^{-}, to the product of the current gain, η and the difference between the current input vector, X_i and previous weight, w_{ji}^{-}.

$$w_{ji}^{+} = w_{ji}^{-} + \eta(x_i - w_{ji}^{-}) \tag{21}$$

The exponential gain reduction rule is determined using:

$$gain = 0.1 * (1.0 - count\_factor) \tag{22}$$

where *count_factor* is the current iteration number divided by the total number of training vectors. Calculation of the neighborhood size is calculated as the time dependent range using:

$$time\_dep\_range = surface\_constant^{[1-(count\_factor)]} \tag{23}$$

The surface constant is calculated by:

$$surface\_constant = xsize^2 + ysize^2 \tag{24}$$

where xsize and ysize are the number of nodes in each Kohonen dimension. For the calculation of the gain constant the following is used:

$$\eta = 0.1(1.0 - iteration/total\_number\_of\_vectors) \tag{25}$$

Output from the Kohonen_net program prints periodic updates of the gain, iteration number, range factor, average number of nodes eliminated, and node uti-

lization. The final output file contains the size in both dimensions, number of vector elements and output Kohonen layer weights. This output file is saved in ASCII format.

Once these output layer weights are saved, the nodes require calibration. The calibration process uses a variety of files shown in the Appendix. What follows is a list of these files to include a brief description of their operation in the Kohonen training and calibration process.

- build_Koho_in.c reads the number of files to be converted into vector elements, file lengths, and file names from a data file. The resulting file is used to train the Kohonen layer.
- buildcal.c is a routine which converts image coordinates to locations within the Kohonen input vector file. The region is defined by its upper left and bottom right coordinates. To run this program requires entering from the command line prompt: buildcal input_vector_file output_cal_file. The output_cal_file is saved in ASCII format.
- Auto_net.c reads the trained Kohonen layer weights, and a calibration file then calculates their Euclidean distance. Each time a node wins its counter is incremented. The output file is a Kohonen "calibration layer" showing the corresponding win count for each node using the given calibration set. This process is repeated for each data class.
- Compare_net_nodes.c reads the filenames of the calibration layers to be compared from a input data file. Compare_net_nodes requires user input of the Kohonen layer dimensions, threshold value, and image size. The threshold value is used to establish arbitrary nodes. Nodes are labeled arbitrary when the difference between the win count of any two classes is less than threshold. Compare_net_nodes produces an output file containing the calibration layer showing the node class association.

- and final_net.c reads the trained Kohonen layer weights, node class assignments, and the training file. The layer weights were created by Kohonen_net, node assignments were written by compare_cal_nodes, training file was created by build_koho_in. Final_net reads the whole image vector file, calculates the Euclidean distance between the current input vector and layer weights, determines the winning node, and assigns a grayscale value to the winner. The grayscale values are written to an output file for comparison with the template.

3.4 Classification Using Radial Basis Functions

The RBF used for this thesis was selected from the network choices available in Dan Zahirniak's neural network environment.(26) Zahirniak's code provides four RBF selections to initialize the hidden layer weights.

- Node at the data points
- Kohonen training
- K-means Clustering
- Center at Class Averages

The center at class averages is used for this research. These hidden layer weights are linked to the output layer via matrix inversion. This routine initializes the hidden layer using the first vector as a single node. This node has a preselected radius. If the next vector is within this radius, the cluster center is updated. If not, a new cluster is added using the present vector value as cluster center. Each of these nodes are assigned class values when initialized. That is, the vector file must provide a class value following each input vector. Training continues until no more nodes are added. (26:3-23)

A complete description of this environment is presented in Dan Zahirniak's thesis. (26) Figure 3 demonstrates RBF feature space clustering.

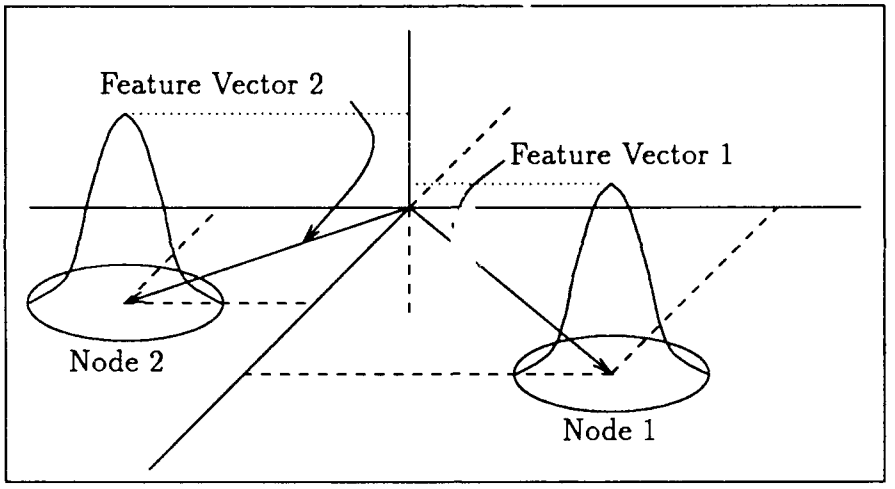

Figure 3. Two Dimensional RBF Clustering Environment (23:34)

3.5 System Description

3.5.1 **Introduction** This section describes the processing used during this thesis. Figure 4 shows the three paths taken. The path to the left shows Gabor processing followed by thresholding and median filtering. This path is used to test the separability provided by each Gabor image. The middle path shows the neural network training implying both Kohonen and RBF processing. The path on the right shows the steps taken to obtain image templates.

3.5.2 **Building Templates** To create image templates the following steps were used. Once the images were converted into Sun format they were converted into unsigned byte binary files and imported to a Macintosh for display. Using available ground site photographs, image regions were converted to grayscale values predetermined for each class. These grayscale images were saved and later used for image comparison.

3.5.3 **Applying Gabors** The technique used for segmentation of this imagery is based on frequency correlation of Gabor filters with SAR images. Image

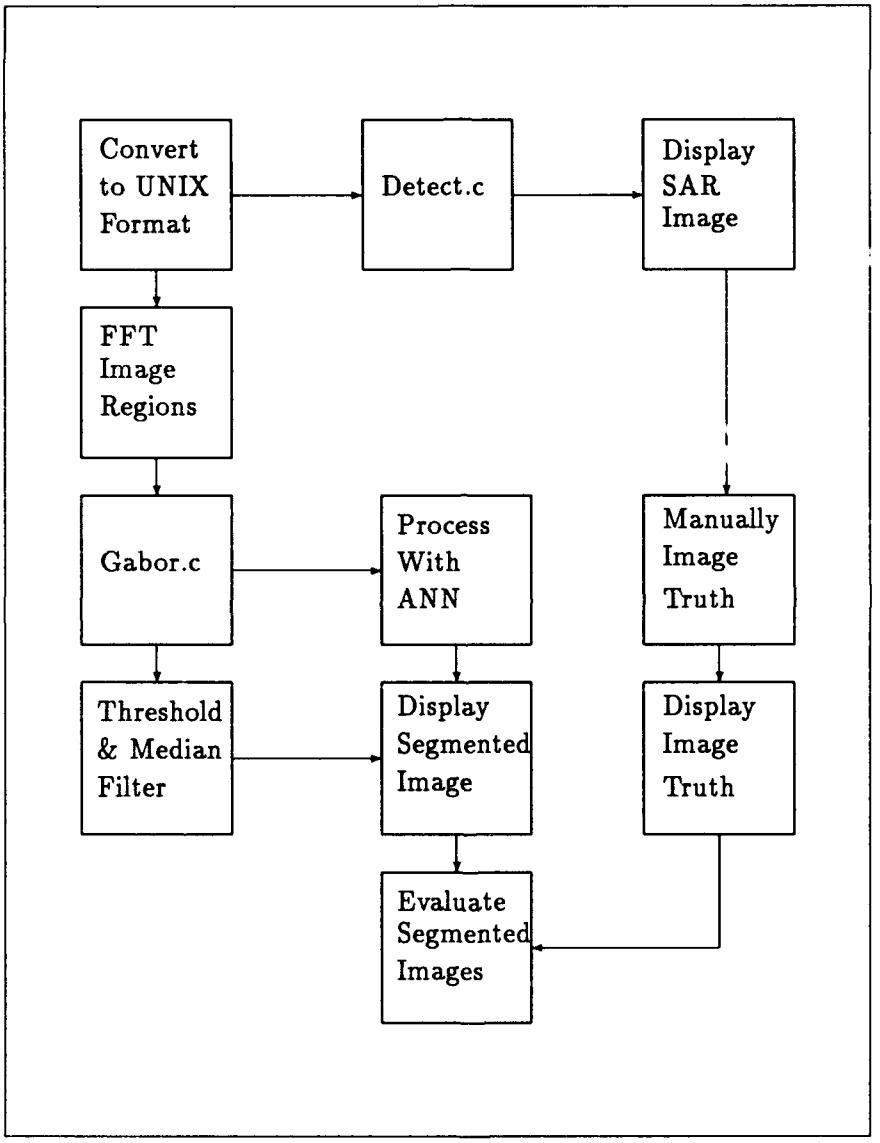

Figure 4. Block Diagram of Methodology

FFTs were used to determine which Gabor frequencies best fit the image texture. Regions were selected from different missions and different frames then their FFTs computed. Selected were the strongest two components for each class.

Early Gabor processing made use of the gabortf.c program written by Eric Frethiem. This program, written in C, dynamically allocates memory to five arrays for the whole Gabor/image correlation process. This program was modified to allow for binary input and output, reading input data from a data file, requires entering the input image file name and data file name from the command line. This is allowed by the argc and argv commands of C programming. The modified routine is called max_gbr_slct.c

Once the setup data is read, image blocks of 256 x 256 were read, and converted to magnitude. The blocks are read from left to right and top to bottom. Once an image block is in memory, it is correlated with each selected Gabor wavelet. It should be noted that once the image block is read-in, its bottom row and right-hand column are zeroed out. This makes allowances for the aliasing effects of the Gabor wavelet.(6) Next requires the calculation of the two dimensional fast Fourier transform (FFT) of the image. Fretheim's FFT.c routine written by Fretheim was linked to the max_gbr_slct routine and performs all FFTs required for Gabor processing.

At this point, the max_gbr_slct program computes a wavelet using the data file information provided as follows:

- number of frequency and rotation combinations
- wavelet window width
- wavelet window height
- type sinusoid (1 = cosine, 0 = sine)
- portion of the output filename
- and lists of the required frequencies, rotations, and standard deviations (both x and y) used.

The frequency, rotation, and standard deviation list was read each time a 256 by 256 block was read.

This user provided data is used in the max_gbr_slct inner loops (rotation within frequency). In other words, each frequency is held constant and used during calculation of each wavelet rotation. Upon completion of each wavelet rotation, the frequency is incremented and is held constant throughout the next rotation loop. Additionally, during the calculation of the wavelet its array is remapped from the center of the wavelet grid (typically 16 x 16, or 32 x 32) to the outer corners of a 256 x 256 zero padded array for frequency correlation.

Operating on images in this manner makes for interesting data input and output problems. To allow for an expandable output file, a file was created and zero filled for each rotation and frequency combination. Each file must be the size of the resulting output Gabor files. This is allowed using the lseek command to move the file pointer to corresponding image locations.

3.5.4 **Preliminary Segmentation Evaluation and Additional Preprocessing** Following the Gabor/image correlation, the output files are converted to unsigned byte and scaled to be between 0 and 255. This allows for evaluation of the segmentation provided by Gabor processing. Using thresholding and filtering techniques available on the Macintosh, Gabor images were analyzed. This is shown in the left-hand path of Figure 3.

Next, combinations of multiple Gabor images were used to train a artificial neural network. The original system plan was to implement a hybrid neural network composed of a Kohonen unsupervised clustering layer and a RBF supervised classification layer. Early results obtained using the RBF alone were substantial enough to change course to clustering and classification using only RBFs. Limited results were obtained for the Kohonen and are shown in chapter 4. The Kohonen implementation is described in the following section.

3.5.5 **Clustering the data using a Kohonen Network** The next step requires training and calibrating a Kohonen artificial neural network to help cluster the data. The Kohonen training process requires data to be presented in a vectorized fashion. To do this, numerous Gabored images were read and converted to vectors of corresponding pixels. The total number of vectors available is simply the total number of pixels. The number of vector elements is dependent upon the number of Gabor representations chosen for the process.

Next, the calibration process performs a Euclidean distance measurement on extracted chunks from the training file. These chunks are selected from representative image regions so the data class was known and monitoring of the Kohonen output nodes was hoped to provide node classification. Multiple 64 x 64 pixel regions were chosen and read from the training file using the buildcal. Once these regions are extracted the auto_net.c routine tests the trained Kohonen layer weights and outputs a file containing the win count of each node from the calibration process. The compare_net_nodes routine compares calibrated nodes against a threshold. The threshold is used to determine which nodes should be labeled arbitrary. The comparison process determines the node class assignments. Using these node assignments, final_net reads in the training file, calculates the Euclidean distance between each vector and the defined nodes, and outputs a remapped grayscale segmented image.

3.5.6 **Radial Basis Function Processing** Selections were made from Zahirniak's neural net environment.(26) The netmenue.c routine contains the parameters requiring selection. For RBF processing images were reduced to ease the training process. Windows of sixteen by sixteen pixels were scanned for the maximum value. The maximum values were then used to represent these windows. That is the files were reduced form 1024 by 512 to 64 by 32. Training consisted of selecting 300 training vectors These were limited to shadow, trees and grass due to data set limitations. These regions were extracted from one image for training and another for partial testing. Partial testing provided output results from the RBF programming

environment. Netmenue also provides for running the net on the shrunken test image. These results were compared to hand segmented images.

Output files from the RBF partial testing provides a summary of training selections, test filenames, and percentage of correct responses for the given vectors. The RBF run file output is compared pixel for pixel with the hand segmented image and percentages are found for each data class.

The training set was limited to 300 vectors to allow timely processing of the matrix inversion calculated for the hidden RBF layer. The center at class averages training rule was selected. Testing image segmentation involved adjusting the sigma threshold, average threshold, interference threshold and sigma factor parameters. Chapter four discusses results of these selections.

IV. Experimental Applications and Results

4.1 Gabor Processing

4.1.1 **Images Processed During This Research** The images processed during this thesis were:

- mission 85, pass 5, frame 27, horizontal/horizontal polarization (m85f27hh)
- mission 85, pass 5, frame 28, horizontal/horizontal polarization (m85f28hh)
- mission 98, pass 3, frame 08, horizontal/horizontal polarization (m98f08hh)
- mission 98, pass 3, frame 09, horizontal/horizontal polarization (m98f09hh)
- mission 98, pass 3, frame 11, horizontal/horizontal polarization (m98f11hh)
- mission 98, pass 3, frame 12, horizontal/horizontal polarization (m98f11hh)

Processing was typically performed on 2048 x 512 or 1024 x 512 images. Shown below are the above listed images. All are shown as 2048 x 512 images, except m98f12. Since only the HH polarization is used during this thesis, the images will be referred to without the HH extension. This data set provides images composed mostly of naturally occurring regions of grass, trees, and shadow. Some limited man made data are available in frames m98f08 and m98f09. Both m98f08 and m98f09 contain corner reflectors and roads, and m98f09 contains a horse ranch at its center.

Image m85f27, shown in Figure 5, contains a section of trees in the center, field on the left and right, and shadow in the middle.

Items of interest for image m85f28, shown in Figure 6, is the trail that cuts across the top of the tree region (upper left), a large tree section in the image center, field to the left and right, and shadow in the middle.

Figure 7 shows image m98f08. This frame contains: a pond in the lower left, paved road through the middle, field to the left of the road, forest on the lower right,

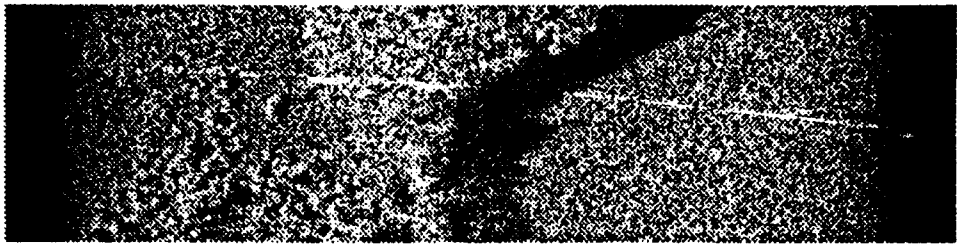

Figure 5. Original SAR Log Mapped Image m85f27

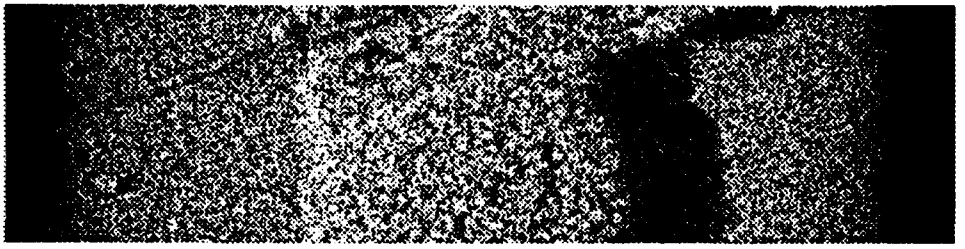

Figure 6. Original SAR Log Mapped Image m85f28

tree farm between the forest and road, and a set of corner reflectors to the right of the pond.

Figure 7. Original SAR Log Mapped Image m98f08

The most interesting image is m98f09 shown if Figure 8. This image contains multiple man-made objects. Directly in the center is a horse ranch to include vehicles, houses, barns, and sheds. In the lower right is a pond with some object in its upper left corner. Additionally, other houses are located in the upper right.

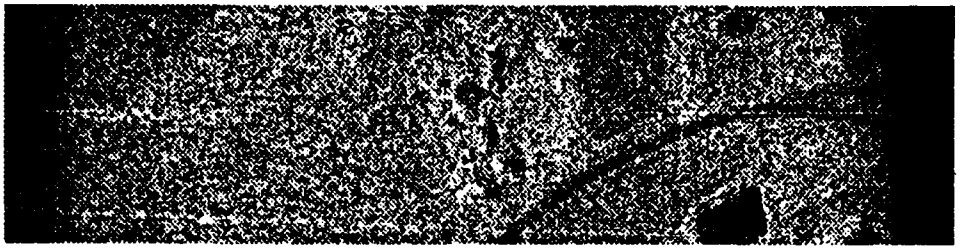

Figure 8. Original SAR Log Mapped Image m98f09hh

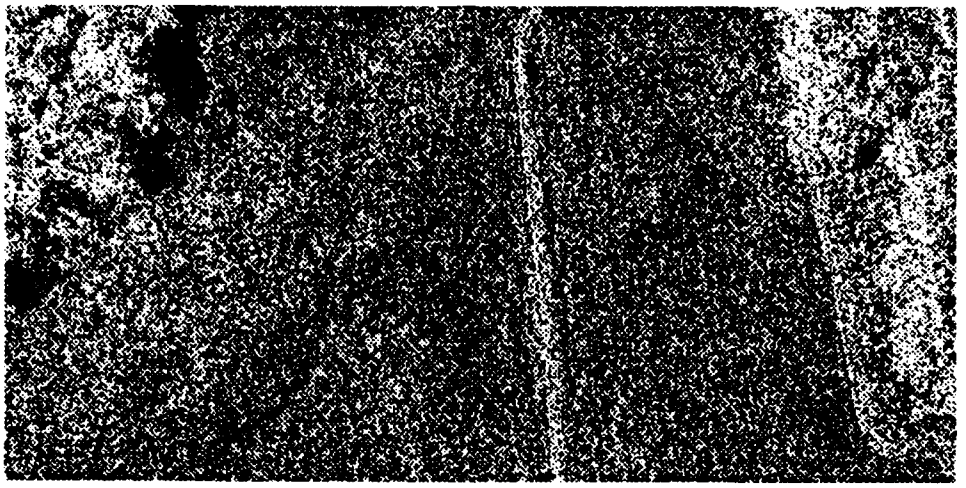

Figure 9. Original SAR Log Mapped Image m98f11hh

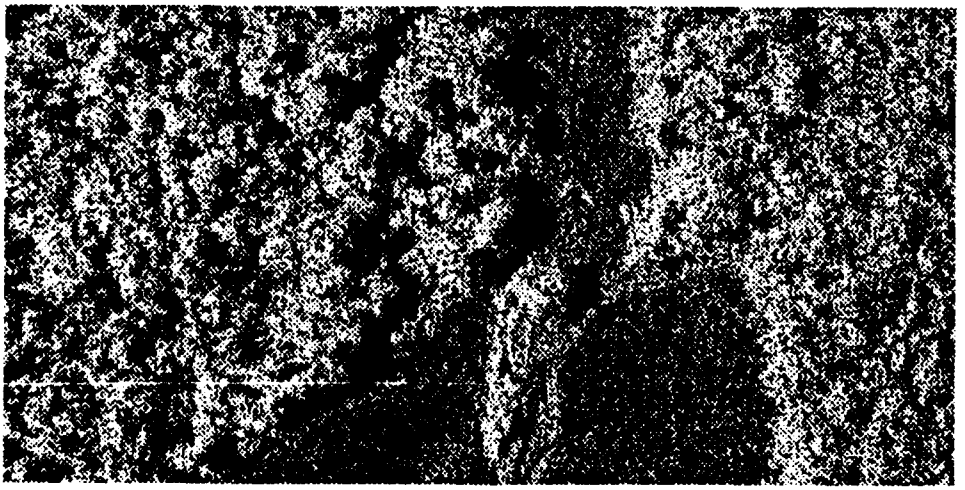

Figure 10. Original SAR Log Mapped Image m98f12hh

Images m98f11 and m98f12 are shown in Figures 9 and 10. M98f11 contains two regions of trees in the both upper corners, a road through its middle, and field in the center. Figure 10 is quite close to being a two class problem since it contains mostly tree and field regions with only limited areas of shadow. Note image m98f12 was only extracted as a 1024 by 512 image hence the difference in size.

4.1.2 **Basic Processing Methodology** Due to image sizes of 2048 x 512 and 1024 x 512, processing was performed 256 x 256 or 128 x 128 blocks. Typically, multiple Gabor representations were generated during a single processing run. Creating nine Gabor filtered images required approximately 45 minutes on a Sun4 Sparc station (assuming dedicated CPU processing). The procedures implemented are described below.

Initially, the Gabor correlation process made use of Fretheim's gabortf1.c program. This program reads an image block, calculates a Gabor filter, performs image and Gabor filter frequency correlation then writes the resulting Gabor image file to disk. If multiple Gabor representations were needed, the original image block was held in computer memory, a new Gabor filter was calculated, and the correlation process repeated. Operating on images in blocks required creating zero filled output files of size 2048 x 512 or 1024 x 512 before Gabor processing. These files were overwritten during processing.

4.1.3 **Ad Hoc Gabor Filter Selection** Early Gabor processing was performed in an ad hoc manner and made use of the gabortf1.c program. The procedure was to calculate numerous Gabor image representations as described above, and the result displayed as a grayscale image. This is the threshold and median filtering step shown in Figure 4. A Macintosh computer was used for display, and additional image processing software was available to test Gabor images for their segmentation properties.

Initially, filters were selected using integer frequencies and rotations. Frequen-

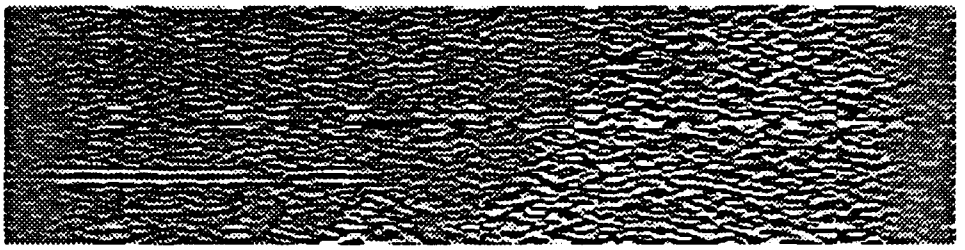

Figure 11. Image m98f08, Freq 2, Rotation 0

cies chosen ranged from one to seven cycles per window. Additionally, various filter rotations were calculated using stepped increments of 15 degrees. All filters were circularly symmetric (same variance in x and y) 16 x 16 pixels. Later processing used only 32 x 32 pixel filters. The majority of the images shown here were processed using 32 x 32 filters.

Integer frequency representations often resulted in wave patterns superimposed on the imagery, but with some of the original image distinguishable as in Figure(11). Images were then processed using thresholding and median filtering then visually evaluated for results. Thresholding and median filtering in this manner provided region segmentation. That is different region classes were separated using threshold setting.

Both figure 12 and 13 are examples of thresholding integer frequency Gabor representations. These m85f27 images were processed using filters of 16 by 16 pixels with a Gaussian standard deviation of 2 pixels.

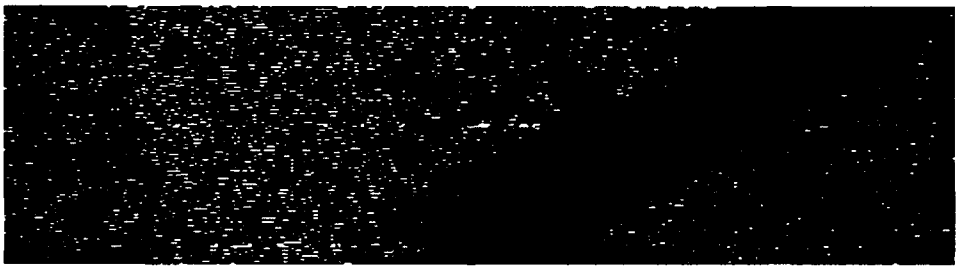

Figure 12. Image m85f27, Freq 2, Rotation 0, threshold 132

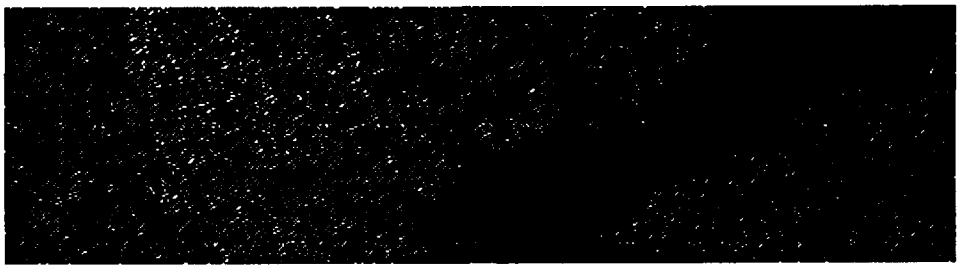

Figure 13. Image m85f27, Freq 2, Rotation 0, threshold 142

Results to this point were marginal at best. It was noted very small (≈ 1) frequency Gabor images provided better image reproduction. Also, reconstruction seemed to be fairly independent, but not entirely, of the circularly symmetric filter's spatial rotation. Better results were wanted prior to attempting ANN processing.

At this point, it was noted that image pixel values were quite large. Sometimes as wide spread as $\pm 30,000$. These values were dependent upon the class of data they represented. That is processing on naturally occurring textures (grass and trees) resulted in values in the wanted $\pm$*hundreds* range. However, when corner reflectors or highly reflective objects were included, returned energy was quite large and greatly expanded the dynamic range of the pixel values. Hence, the Gabor image became excessively dark requiring equalization prior to display.

Up to this point, the standard deviation of the Gaussian envelope was held constant at 2 (16 pixels/8). It was now believed, that finer σ tuning would improve image representation. With this in mind, an experiment using a larger variance of 4 was tried using both 16 and 32 pixel windows. As expected, changes in the variances caused drastic variations in resulting pixel values. Additionally, the increased filter size of 32 pixels provided less image detail than 16 x 16 pixel filters. Findings to this point made it clear a better method was needed for determining filter frequencies and rotations.

4.1.4 Using FFTs to Determine Frequency and Rotation Angles

Techniques found in the Bovik article (3:63) suggested two methods for determining

Gabor filter center frequencies. Suggested were:

- calculating image FFTs and selecting the two strongest peaks
- or convolving the image spectrum with a Gaussian and selecting the strongest peaks.

The technique chosen made use of the Image_fft software package available on WRDC's Macintosh computer. Analysis with this tool involved taking FFTs of 64 x 64 pixel blocks of selected image regions. The interest here was to determine independent frequencies representing separate data classes. The Image_fft program returns two windows. One showing the power spectrum density plot and the other providing spectra information. That is, the results window showed relative power (0 to 254), frequency (pixels/cycle) and angular location (θ) of the power spectral elements when the spectral element was selected.

For this FFT analysis, image blocks were chosen to included only pixels of the class type to be analyzed. However, exceptions had to be made for roads, cultural objects and corner reflectors. These objects often weren't more than 10 to 20 pixels wide or long. Originally, it was thought these overlapping areas would allow selecting out of class frequencies in place of those representing the desired class. However, frequencies for roads, cultural objects and corner reflectors were found to be less random and more strongly aligned with the frequency axes than naturally occurring textures. For the man made objects, the power spectrum window showed significantly whiter power spectrums (dark being high spectrum values) containing dark spots more strongly aligned with the two dimensional frequency axes. Results showed roads, paved and dirt, were largely represented by the same frequencies and orientations. In relation, roads that were basically paths worn into grassy areas by vehicles had a more random frequency spectrum in both orientation and frequency. Cultural objects and corner reflectors rang loudest at 0.500 cycles per 32 pixel win-

dow, and by no means were they represented by the extreme number of frequencies needed to define the power spectrum for trees, grass and shadow.

The SAR imagery FFT analysis was performed across images and missions to include the following: m85f27, m85f28, m98f08, m98f09, m98f11 and m98f12. An attempt was made to find multiple frequencies for each class which were independent of those composing other classes. A summary of the analysis is shown in Table(1).

Shown are frequencies where the spectrum contained the maximum value of 254 in at least one image and strongly represented (relative power spectrum value of 100 or higher) in at least one other. As can be seen from Table(1), some of the strong frequencies were consistent across classes. Namely, 0.500 cycles per window (32 pixel window size) was representative of all data classes. This table shows class type, frequency (cycles/window), rotation (degrees) and whether the frequency held across images. The starred entries in the across image column are measurements taken from only one image.

Using results presented in Table(1), nine frequencies were selected for Gabor processing. The bandwidths were calculated using equation 19 and are shown Table(2). Shown are frequencies (cycles/window), orientations (degrees), standard deviations (number of pixels), window size (number of pixels) and calculated radial bandwidths.

The frequency selections shown in Table 2 used Bovik's radial bandwidths (0.7, 1.0 and 1.3) as guideline for selecting the bandwidths for higher frequency processing.(3:60) Also, the Bovik article suggests when tuning Gabor filters, the radial bandwidth should be decreased as center frequencies are increased to maintain spatial resolution.(3:60) Additionally, Bovik points out higher frequencies may require increasing the number of filters and at some point, namely 0.2 cycles / pixel, the "filters may require sampling above that possible for the available image".(3:60) It was with this guidance the frequencies representations of Table 2 were calculated. Attempts using frequencies higher than 1.5 cycles/window (32 pixels) often produced

Table 1. SAR Imagery Frequency Analysis

Class Type	*Freq (cycles/window)*	*Rot (degrees)*	*Across Images*
Trees	0.500	0 or 90	Yes
	0.707	45 or 135	Yes
	1.000	0 or 90	Yes
	1.118	63 or 153	Yes
	1.414	45 or 135	Yes
Grass	0.500	0 or 90	Yes
	1.000	0 or 90	Yes
	1.800	146	No
	2.120	135	No
	3.040	9 or 81	Yes
	3.900	140	Yes
	4.611	13 or 77	Yes
Shadow	0.500	0	Yes
	3.540	98 or 135	Yes
	4.500	0 or 90	Yes
	5.100	11	Yes
Roads	0.500	0	Yes
	0.707	135	Yes
	1.580	18 or 108	Yes
Water	0.500	0	Yes
	1.117	153	Yes
Culture	0.500	0 or 90	No*
	1.118	26	No*
	2.500	37	No*
Corner Reflector	0.500	0	No*

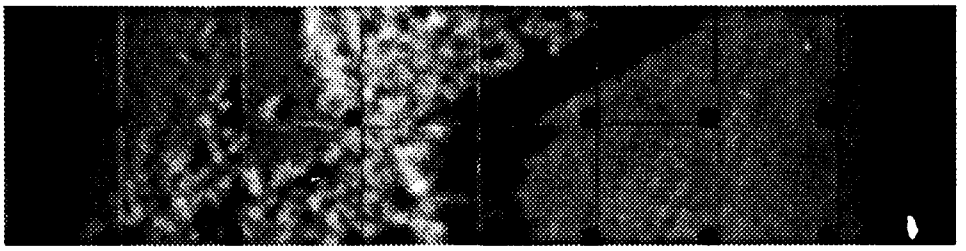

Figure 14. Freq 0.5, Rotation 0, Msn 85 Frame 27

Attempts using frequencies higher than 1.5 cycles/window (32 pixels) often produced images that were essentially random noise.

Table 2. Tweaking the Bandwidths

Freq (cycles/window)	*Rot(degrees)*	σ	*Window Size(pixels)*	Bandwidth
0.500	0	11	32	6.4104
0.707	45	11	32	2.9515
1.000	0	11	32	1.7640
1.414	135	8	32	1.7032
1.500	0	7	32	1.5838
2.000	0	4	16	1.1368
2.828	45	3	16	1.0656
4.000	0	3	16	0.7365
5.000	0	3	16	0.5846

The images for frequencies shown in Table 2, except for frequencies four and five, are shown in Figure 14 through Figure 20. Notice in Figure 17 the Gaussian smoothing caused by the Gabor. Images for frequencies four and five were considerably more random than Figure 20. As a result these frequencies were eliminated from further processing.

The boxiness of the Gabor images may have been noticed prior to now. The larger boxes are 256 x 256 (or 128 x 128) blocks used during image processing. The blackened squares are zeroed out regions which resulted from compensation for Gabor filter aliasing.

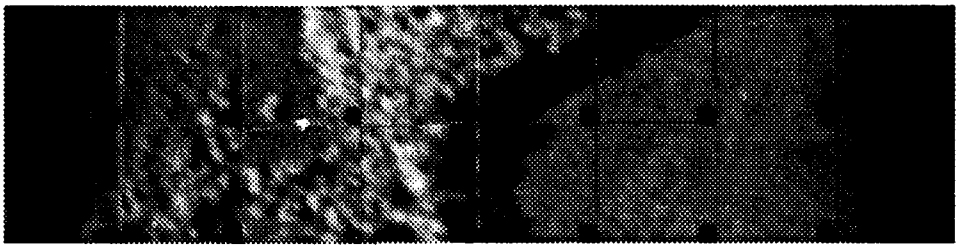

Figure 15. Freq 0.707, Rotation, Msn 85 Frame 27

Figure 16. Freq 1.000, Rotation 0, Msn 85 Frame 27

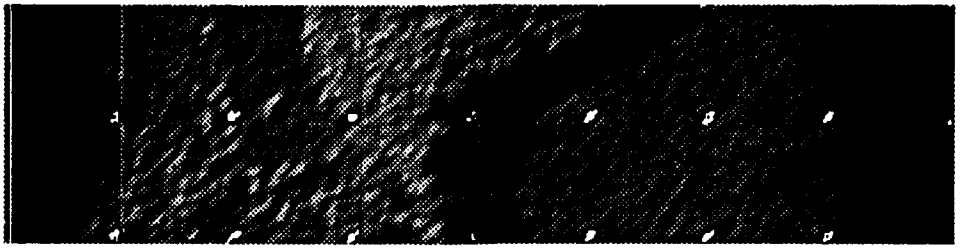

Figure 17. Freq 1.414, Rotation 135, Msn 85 Frame 27

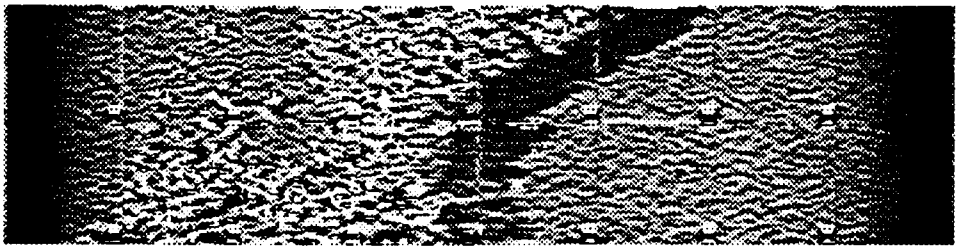

Figure 18. Freq 1.5 Rotation 0, Msn 85 Frame 27

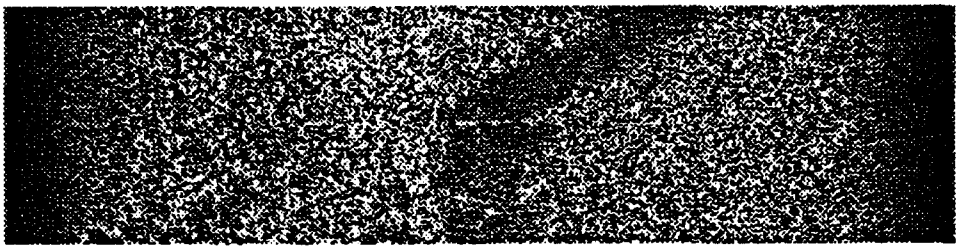

Figure 19. Freq 2.0 Rotation 0, Msn 85 Frame 27

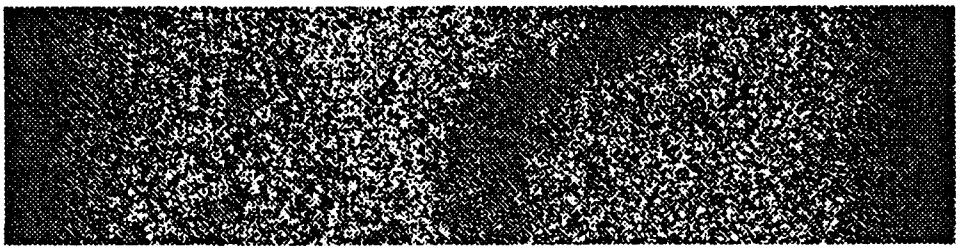

Figure 20. Freq 2.828 Rotation 45, Msn 85 Frame 27

From the results obtained in Figure 14 through Figure 20, it appears the bandwidth relation defined by Bovik effectively defines the Gabor tuning characteristics to provide high quality reduced speckle HH polarization images. That is once the frequencies were selected the Bovik bandwidth equation provided proper selection of the envelope standard deviation. Notice the decreased speckle in the first three representations. With these results, the lower frequencies, up to frequency 2.828 cycles/window, were selected for the remaining processing.

Figures 21 through 23 demonstrate using the calculated frequencies, and σ on image m85f28. The results are similar to that obtained for image m85f27.

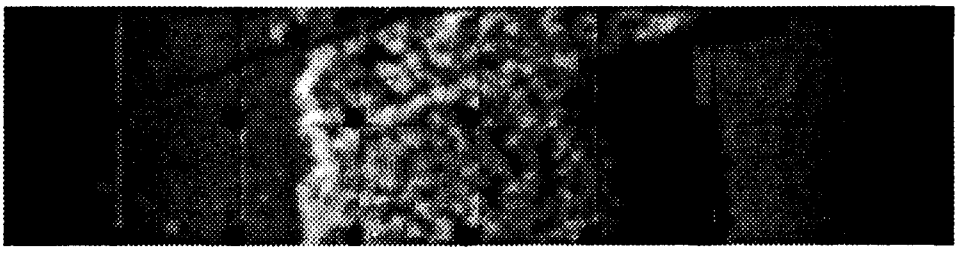

Figure 21. Image m85f28, Freq 0.500, Rotation 0

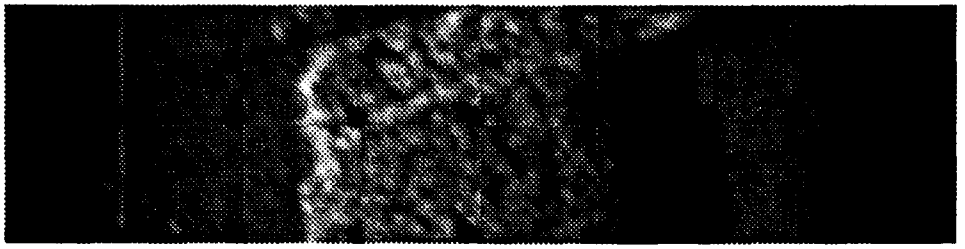

Figure 22. Image m85f28, Freq 0.707,Rotation 45

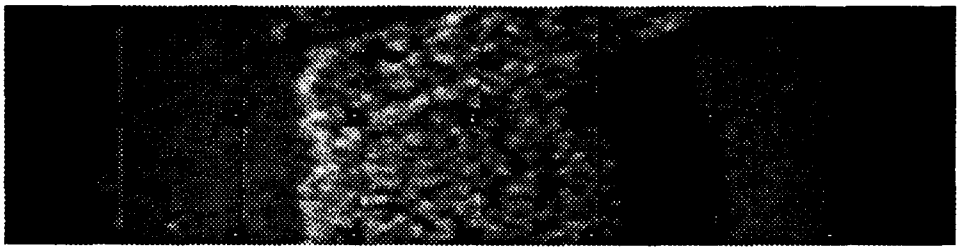

Figure 23. Image m85f28, Freq 1.000, Rotation 0

Figure 24 shows the original m98f09hh SAR image. Figure 25 through 31 show Gabor processing results for the frequency and rotation shown in each caption. These images were equalized to improve their display quality to the level achieved for m85f27 and m85f28. It was noted earlier, the inclusion of corner reflectors or objects with good radar returns expanded the dynamic range of resulting images. This is the case for both m98f08 and m98f09.

Notice once again above 1.5 cycles/window Gabc mages became quite indecipherable.

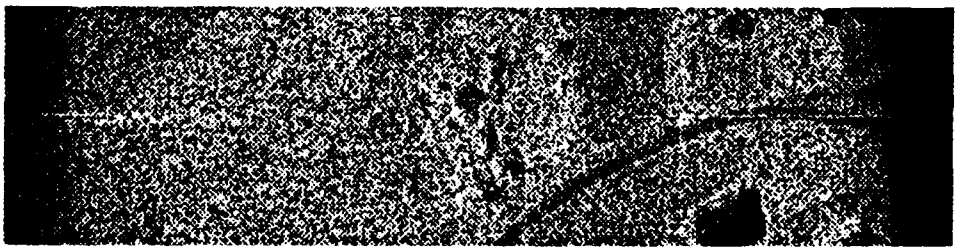

Figure 24. Image m98f09

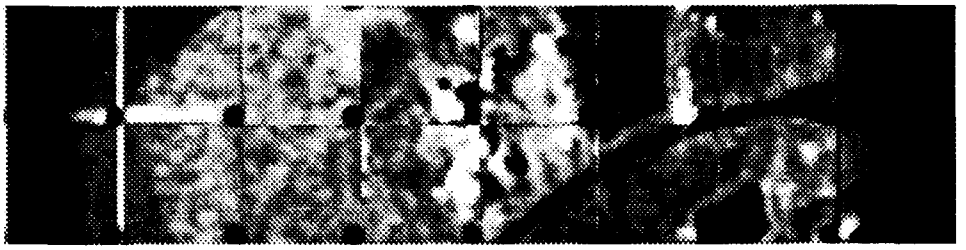

Figure 25. Image m98f09, Freq 0.500, Rotation 0

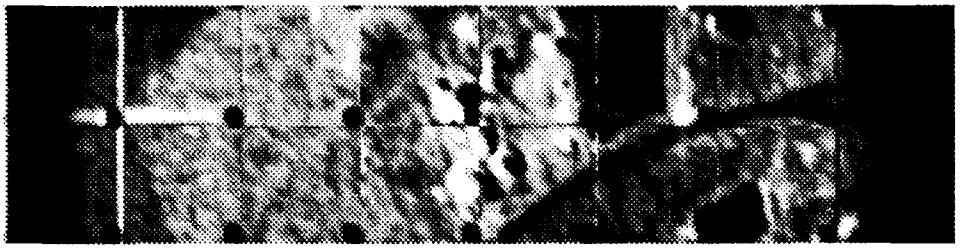

Figure 26. Image m98f09, Freq 0.707, Rotation 45

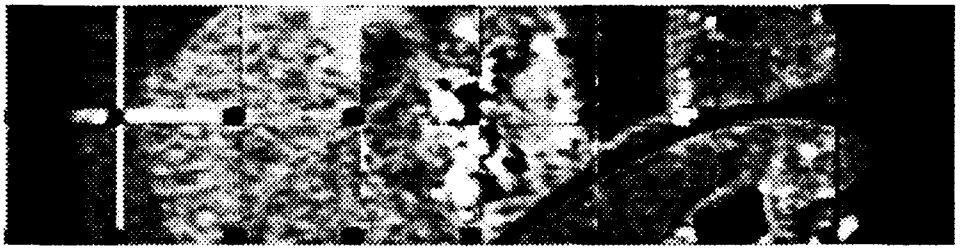

Figure 27. Image m98f09, Freq 1.000, Rotation 0

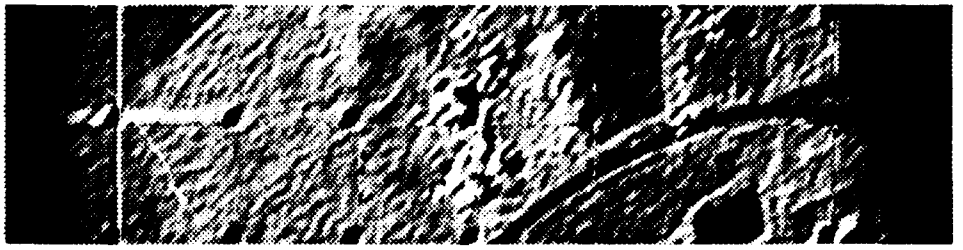

Figure 28. Image m98f09, Freq 1.414, Rotation 135

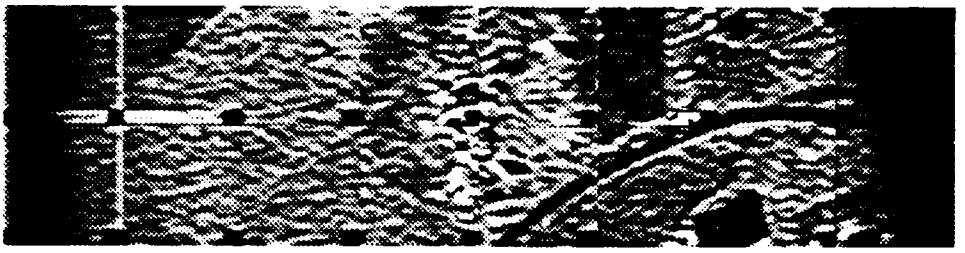

Figure 29. Image m98f09, Freq 1.500, Rotation 0

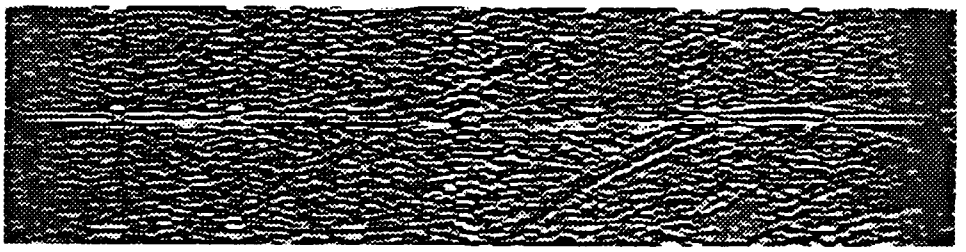

Figure 30. Image m98f09, Freq 2.000, Rotation 0

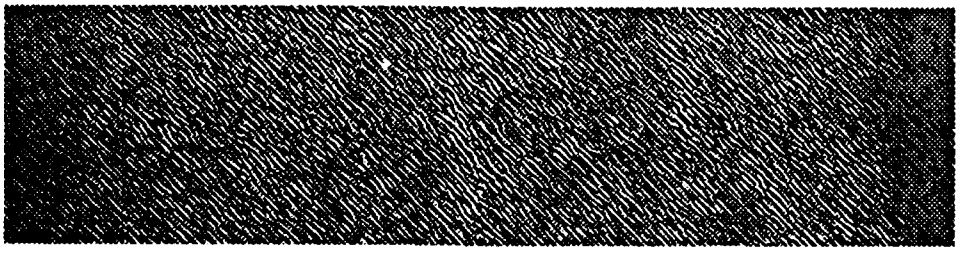

Figure 31. Image m98f09, Freq 2.828, Rotation 45

Figure 32. Tweaked Image m98f08, Freq 0.500, Rotation 0

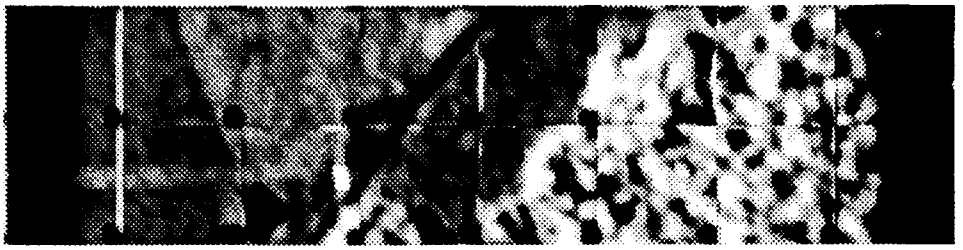

Figure 33. Tweaked Image m98f08, Freq 0.707, Rotation 45

Figure 32 through 34 demonstrate using the lower three calculated frequencies on m98f08. Once again the corner reflectors expanded the pixel value dynamic range, so equalization was needed to display these results.

Other experiments demonstrating good results are shown in Figures 35 and 36. These images were processed using 128 x 128 image blocks and Gabor filters of 16 x 16 pixels. Importing images to the Macintosh computer, and processing with thresholding techniques demonstrated thresholds could be defined for separate image regions. That is cultural items such as the horse ranch, vehicles and corner reflectors were usually segmented using relatively high thresholds. Other the regions

Figure 34. Tweaked Image m98f08, Freq 1.000, Rotation 0

Figure 35. Experimental Image m98f09, Freq 1.0, Rotation 0

Figure 36. Experimental Image m98f09, Freq1.0, Rotation 15

were also segmentable by varying the threshold levels. Thresholding of the original SAR images was investigated and proved to be an invalid segmentation technique.

Figures 35 and 36 were processed on image m98f09 using 16 x 16 Gabor filter, and 128 x 128 image blocks. Both figures used a standard deviation of 4 pixels. Notice upon comparison with Figure 27 using sixteen pixel windows provide better image detail. These results were found while including these images into this final draft. Distinguishable items are the building shapes, field textures and houses in the upper right corner. These features were overlooked and hence not used in ANN processing.

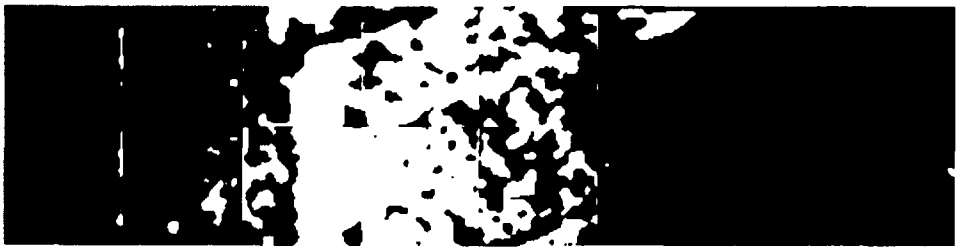

Figure 37. Threshold 121

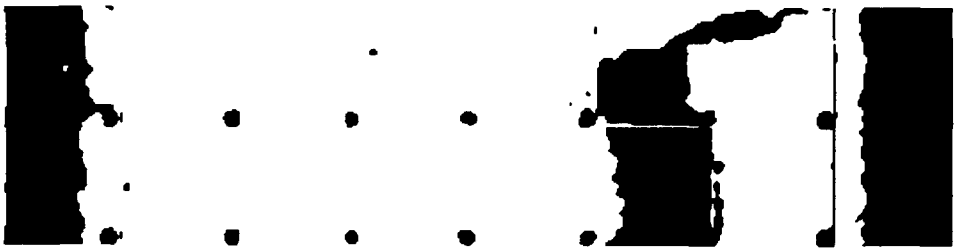

Figure 38. Threshold 62

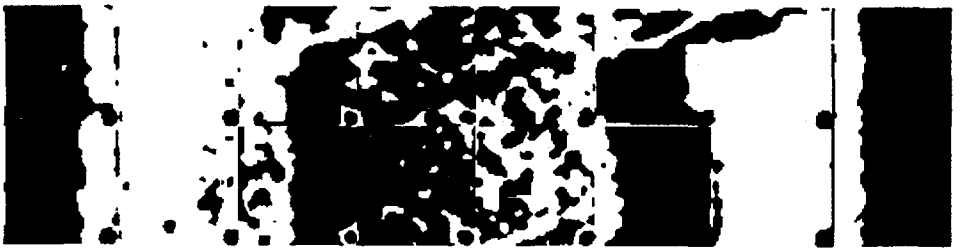

Figure 39. Difference

To test Gabor images prior to use as input features for neural networks, each Gabor image was tested for threshold separability. These tests were performed on each Gabor representation. This testing proved that simple thresholding following Gabor preprocessing provides image segmentation. Due inconsistent values for similar regions between images, thresholds weren't consistent across images. Even so, it was found that separate thresholds could be defined for each major region (grass, trees and shadow). Figure 37 through Figure 39 demonstrate segmentation of the regions contained in image m85f28. Figure 37 had the grayscale level set to 121 to segment the tree area, Figure 38 at 62 to segment the shadow area and Figure 39 segments the field regions by taking the difference between Figures 37 and 38. Similar results were obtained for images m85f27, m98f08, m98f09 and others.

4.2 Kohonen Processing

4.2.1 **Overview** The following results were obtained from Kohonen training sessions. Image m85f27 was the training image. This image contains three data classes trees, shadow, and grass. The goal was to train on image m85f27 and test

the Kohonen layer with other images containing similar data classes. Only limited amounts of cultural regions were available in the mission 85 and 98 data set, so processing focused more upon images such as m85f27. Described below are the procedures taken to train and calibrate Kohonen network layers.

4.2.2 **Training and Calibration** Training vectors were composed of the lower four frequency Gabor representations from Table 2. Only four representations were chosen to reduce the size of the training file. Training files were created by converting image files to vectors using the build_koho_in.c program listed in the appendix. Using pixel for pixel information, these vectorized files contained over one million training vectors. Saved in ASCII format, these training files were often on the order of 37 megabytes.

Kohonen layers were trained using the Barmore/Recla Kohonen_net program. (1, 18) Following training, known image regions were chosen from the original image and used to calibrate the Kohonen layer. Image coordinates, obtained using the Image 1.30 software package on the Macintosh computer, were entered into the buildcal.c program to create calibration files. Figure ?? shows an example of regions selected from m85f28 for RBF training. Similar regions were selected from m85f27 for Kohonen training. The auto_net program used output Kohonen layers to count the number of times each node won during calibration. Layer comparisions showed many nodes were arbitrary. Shown below are several Kohonen layers demonstrating results from Kohonen calibration.

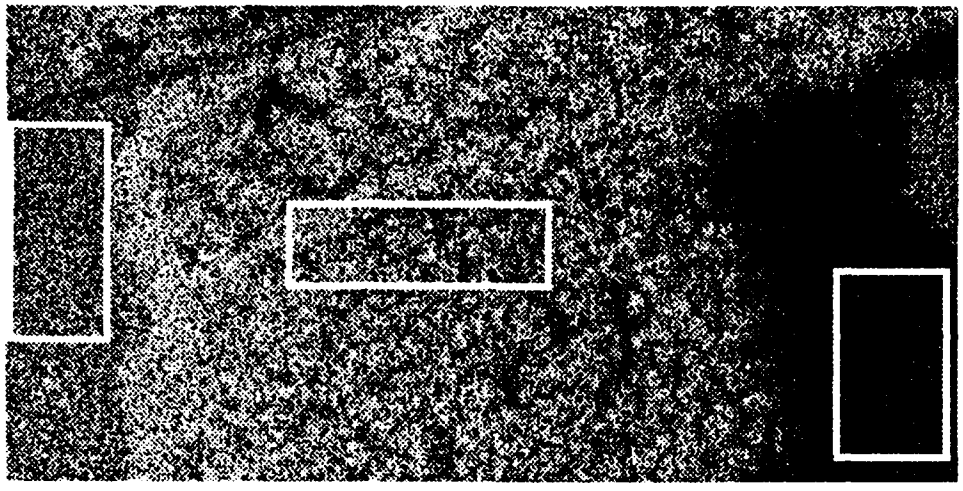

Figure 40. Sample Training Regions

Kohonen Example One

f27shadow_III.cal				f27trees_III.cal				f27field_III.cal			
0	0	0	0	687	332	687	2910	1852	207	1852	360
0	0	0	0	332	332	332	2786	207	207	207	1677
0	0	0	0	687	332	687	291	1852	207	1852	360
0	0	0	4096	291	2786	291	0	360	1677	360	0

Kohonen Example Two

f27shadow_final.cal				f27trees_final.cal				f27field_final.cal			
3844	78	0	0	0	0	0	4	0	0	0	0
47	0	0	0	0	0	52	131	0	0	87	199
0	0	0	0	74	153	359	390	0	585	1529	988
0	0	0	0	111	296	783	1616	72	297	228	111

These output Kohonen layers resulted from testing with vectors from known regions of image m85f27. Notice in example one, the first three layers, contain more than 4096 winning vectors even though only 4096 vectors were used for calibration. This resulted from using to much conscience (1.5). The last three layers,example two, were trained using very little conscience (5.0). The layers from example two

seem to imply this data is clusterable. These three layers were calibrated using only 3969 vectors. Example two training and calibration show that shadow vectors are clustered in the upper left portion of the layer. Trees and field nodes are more arbitrary. The nodes in the lower right corner appear arbitrary, but establishing a difference threshold could be used to extrapolate node class assignments. Comparing the bottom two rows of f27trees_final.cal and f27field_final.cal it can be argued given some arbitrary threshold that three of the four bottom nodes are tree nodes and the whole second row from the bottom are field nodes. The exception in the bottom row is the second node from the left. This node and the three in the upper right are quite arbitrary.

Numerous Kohonen layer sizes were trained on image m85f27. Networks larger than four by four contained too many nodes where many nodes were zero during the calibration step.

Using these results an image was converted back to grayscale and is shown in figure 41. This image was remapped from a Kohonen layer trained using too much conscience. Even so, comparing Figure 41 with the original Figure 5 shows the regions are somewhat segmented. The white areas are trees, black grass, and gray shadow.

Figure 41. Remapped Grayscale Image m85f27

At this point, Dan Zahirniak's radial basis function network (26) was tested and demonstrated good results. This in combination with Kohonen calibration difficulties eliminated the Kohonen network from further processing.

4.3 Radial Basis Function Processing

It was decided to reduce the training vector data size. The Gabor files were reduced by finding maximum values within a sixteen by sixteen block. This reduced the 1024 by 512 images to 64 by 32. These were converted to vectors and used to train the RBF network. Shown in Figure 42 is an example of the imagery used for RBF testing. This reduced 64 x 32 image was enlarged by 800 percent for display and inclusion here. Figure 43 shows the affect of selecting the maximum value from an 8 x 8 window. This result is 128 x 64 and has been enlarged by 400 percent for inclusion here. Images sampled by 8 x 8 windows were not process, but rather are shown to suggest processing could continue using these reduced images. Both images represent 1024 x 512 portions extracted from image m85f27.

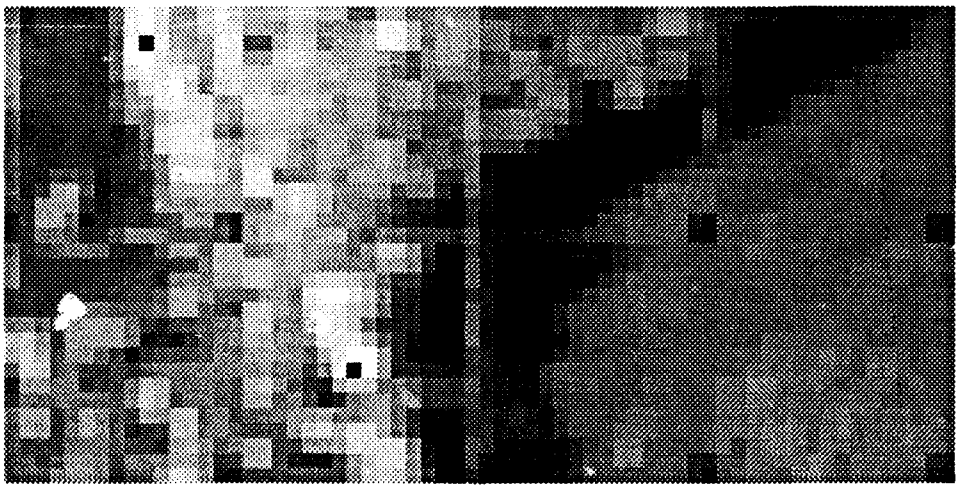

Figure 42. Sampled Image Using Max Value From 16 by 16 Block

Two techniques were used to evaluate Gabor/RBF segmentation. The first used a pixel for pixel comparison between the reduced RBF images and hand segemented templates. The second, used data obtained from the RBF network environment. The RBF network data files are percent correct results obtained during network training.

A'l RBF processing used the center at class averages selection using the following parameters to adjust the network:

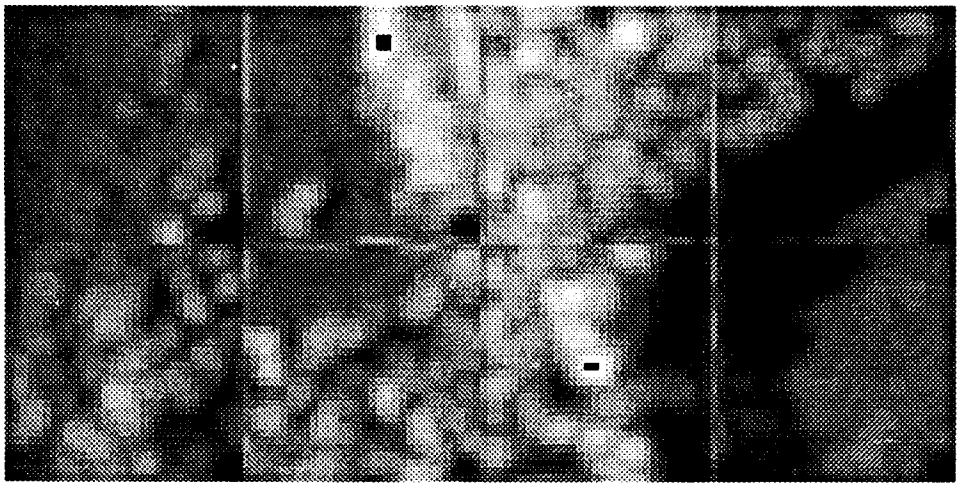

Figure 43. Sampled Image Using Max Value From 8 by 8 Block

- average threshold
- sigma threshold
- constant interference threshold
- sigma factor

The average threshold is similar to vigilance.(26) This parameter is used to determine addition of new clusters within data classes. If the data is within the average threshold and of the same class, its center is updated to center on the cluster. Sigma threshold is used to set the RBF spread or standard deviation. The sigma factor is applied to reduce the sigma threshold after all the nodes have been added to the network. Sigma factor in other words controls the rate of sigma threshold reduction. The interference threshold determines allowed overlap of cluster regions between classes.

4.3.1 **Comparison With Image Templates** For these comparisons, test files were converted to vector files with arbitrary class assignments appended to the end of each vector. These files were selected as "run files" in the netmenue

Table 3. RBF Trained Image Msn 85 F28

Test Image	*% Agree Shadow*	*% Agree Trees*	*% Agree Field*	*Total % Agree*
M85F27	61.2	76.6	73.1	72.9
M85F28	78.2	89.6	93.1	88.0
M98F08	0.0	69.2	37.4	46.3
M98F11	0.0	47.8	52.6	51.3
M98F12	0.0	34.9	63.0	39.6

routine.(26) Network processing determined vector (pixel) class assignments and converted each vector to a corresponding grayscale value.

Comparison with hand segmented templates was performed twice. The first used non-scaled Gabor values from the sampled image set. The second used scaled versions of mission 98 images. Image m85f28 was the training image. One hundred vectors representing the three classes were selected and used for training. These regions are shown in Figure 40.

To train and run the net, appropriate filenames were entered into netmenue.c, and training parameters were chosen. The initial parameters (baseline settings) were:

- sigma threshold, 4.0
- average threshold, 0.1
- interference threshold, 0.4
- sigma factor, 0.1

Both comparisons were made using unsigned byte grayscale values. Comparison using unscaled data was first made between image templates and unfiltered RBF outputs. A second comparision was performed after using a 5 by 5 median filter on RBF outputs. Tables 3 and 4 show results of each comparison.

The poor results from images F08, F11 and F12 resulted from Gabor coefficient values drastically out of range of those obtained for images F27 and F28. Throughout

Table 4. RBF Trained Image Msn 85 F28 After Median Filtering

Test Image	*% Agree Shadow*	*% Agree Trees*	*% Agree Field*	Total % Agreement
M85F27	58.8	94.7	80.8	84.0
M85F28	86.1	95.6	89.2	93.2
M98F08	-	61.3	57.8	43.5
M98F11	-	43.7	53.3	50.6
M98F12	-	23.7	85.0	33.9

Table 5. Scaled Result, RBF Trained Image Msn 85 F28

Test Image	*% Agree Shadow*	*% Agree Trees*	*% Agree Field*	Total % Agree
M98F08	-	0.5	99.5	35.2
M98F11	-	61.9	81.7	76.1
M98F12	-	62.7	77.4	65.2

processing, images from Mission 98 provided out of line results. A Lambertization process (11) would probably provide the localized scaling necessary to allow Gabor processing to hold across mission sets.

A quick fix approach was taken to bring the mission 98 values in line with mission 85 values. The results shown are from scaling each mission 98 file by a factor of $\frac{1}{30}$. Using the same netmenue parameters, the RBF results are shown in Table 5 and 6. Using this simple scaling improved results in tree and field categories, but eliminated detection of the shadow regions. Additionally using this technique, m98f08 remainded unclassifiable; however, images m98f11 and m98f12 came closer to being in line with results obtained for image m85f27. Median filtering also improved image comparisons between m98f11, m98f12 and hand segmented templates.

Figures 44 through 46 show hand segmented templates used for image comparisons.

Table 6. Scaled Result, RBF Trained Image Msn 85 F28 Median Filtered

Test Image	*% Agree Shadow*	*% Agree Trees*	*% Agree Field*	Total % Agree
M98F08	-	-	-	-
M98F11	-	65.9	87.2	81.3
M98F12	-	73.6	90.9	76.5

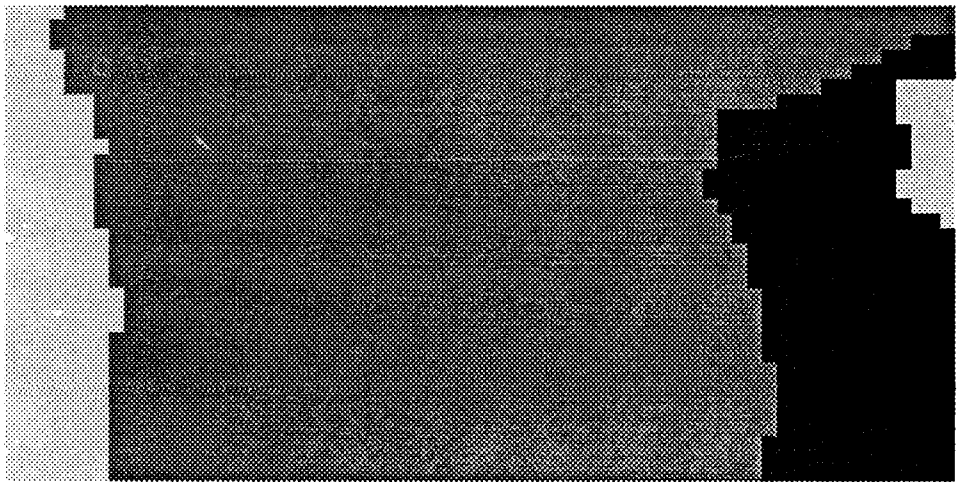

Figure 44. Hand Segmented Image m85f28

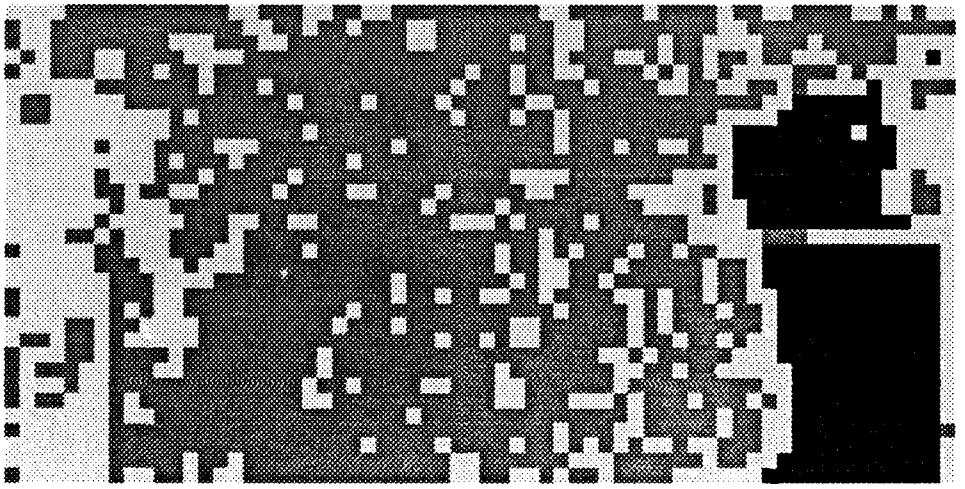

Figure 45. RBF Segmentation of F28, Trained on F28

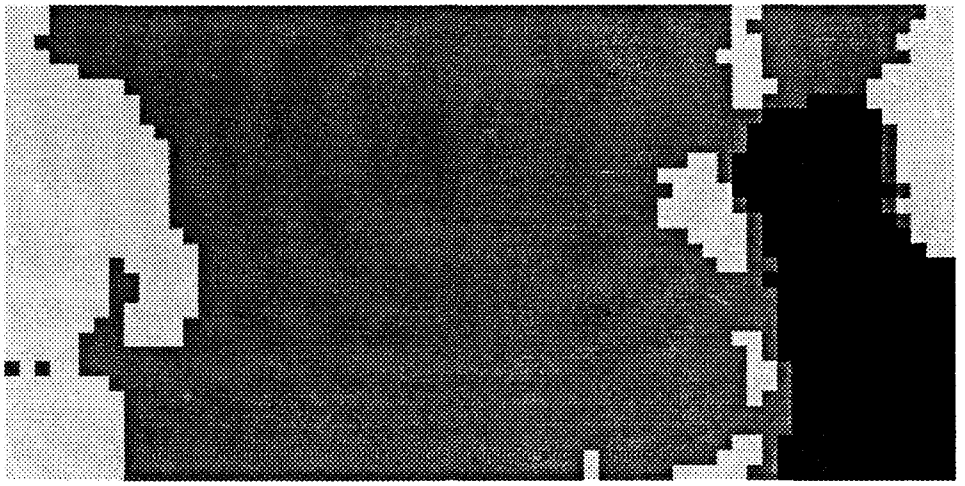

Figure 46. RBF Segmentation of F28, 5 x 5 Median Filter

Other test images are shown in the Appendix. For each case these templates are followed by its resulting RBF segmented image then a 5 x 5 median filtered result. All the RBF outputs were generated using m85f28 as the training file using the baseline settings mentioned earlier. The mission 98 images are results following the quick fix scaling.

4.3.2 **RBF Network Results** Using 300 training vectors from image m85f28 and 300 testing vectors form m85f27 network parameters were varied to test the affects of each. The percentages used here were based on RBF network class selections versus read input classes.

This processing was performed using image m85f28 as the training file and m85f27 as the test file. Processing was performed while adjusting netmenue's average threshold, sigma threshold, sigma factor and interference threshold parameters. Figures 47 through 51 show the results of this processing. Here the settings for these parameters were:

- average threshold 0.1
- sigma threshold 2.0

- interference threshold 0.4
- sigma factor 0.1

One parameter was varied while holding the others at the baseline setting. Figure 47 shows the effects of varying average threshold form 0.1 to 1.0. Best results were obtained using thresholds below 0.3 and at 0.6. The results of the 0.6 threshold setting is probably be attributed to close match with the data as the clusters were shrunk. Figure 48 shows the nodes required to cover the training data. Results were as expected and agree with those found by Zahirniak.(26:5-11) That is, by increasing the average threshold less RBF clusters are needed to cover the data space. Figure 49 shows the results from varying the sigma threshold form 0.1 to 10.0 for both testing and training image results remained approximately constant. Figure 50 shows the affect of varying sigma factor between 0.1 and 1.0. Results, for this data set, seem to indicate results are fairly constant when varied between 0.1 and 0.8. Results above 0.8 show the percentages quickly drop to zero. The interference threshold was allowed to vary between 0.01 and 1.0. This plot seems to imply varying this parameter between 0.1 and 0.8 for the training set provides percentages in nearly the same range. The peak at 0.6 once again may imply closeness of fit between the parameter settings and the data.

Shown here are some resulting Gabor/RBF segmented images. Figure 52 is the result from setting the RBF baseline parameters as mentioned above and setting the average threshold to 0.6. Figure 53 is the result following a 5 x 5 median filter.

Figures 54 and 55 show results from using the baseline settings with sigma threshold set to nine.

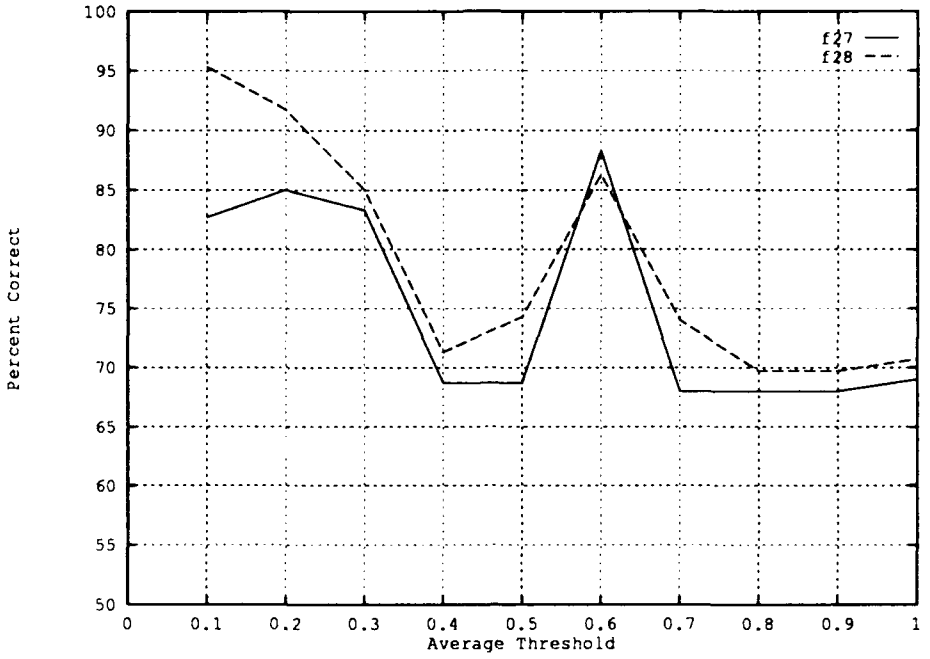

Figure 47. Percent Agree vs Average Threshold

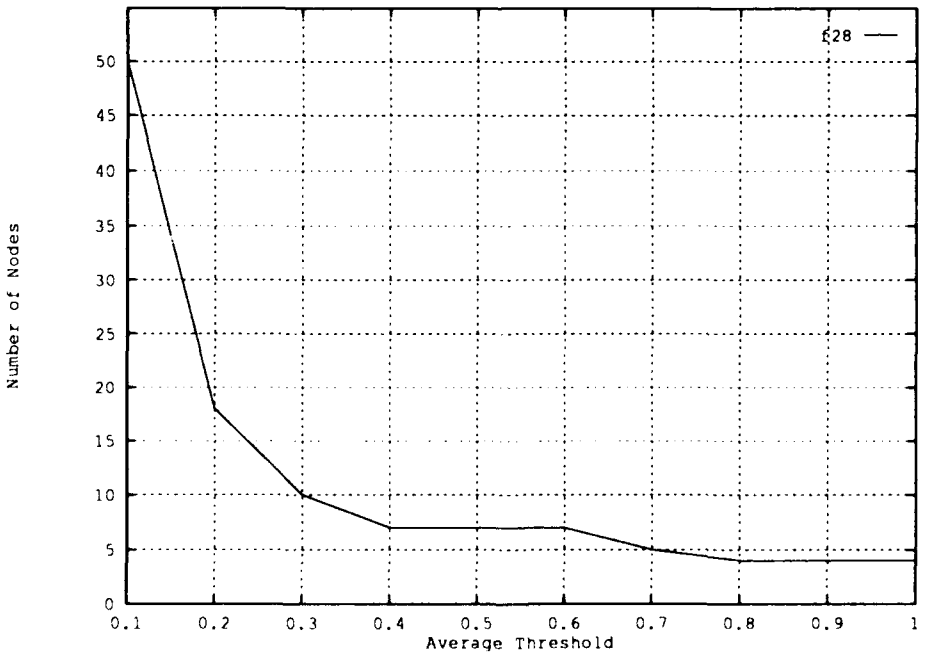

Figure 48. Number of Nodes vs Average Threshold

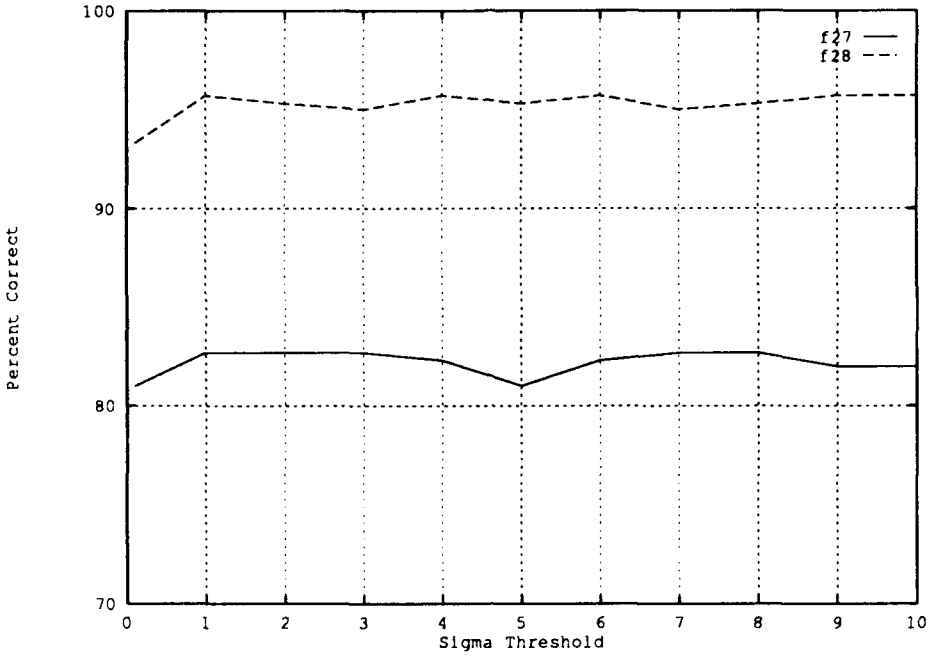

Figure 49. Percent Agree vs Sigma Threshold

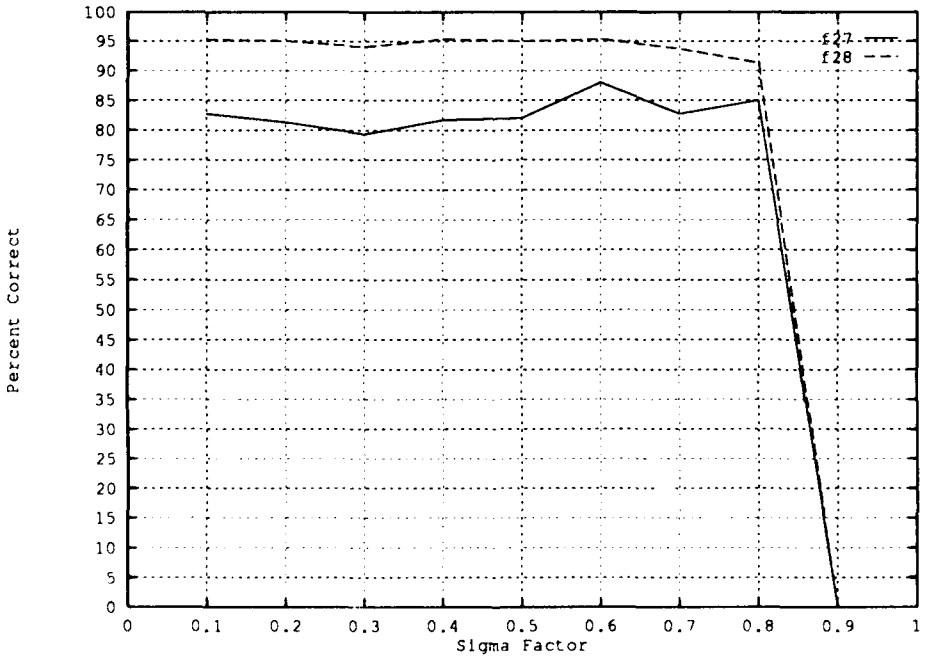

Figure 50. Percent Agree vs Sigma Factor

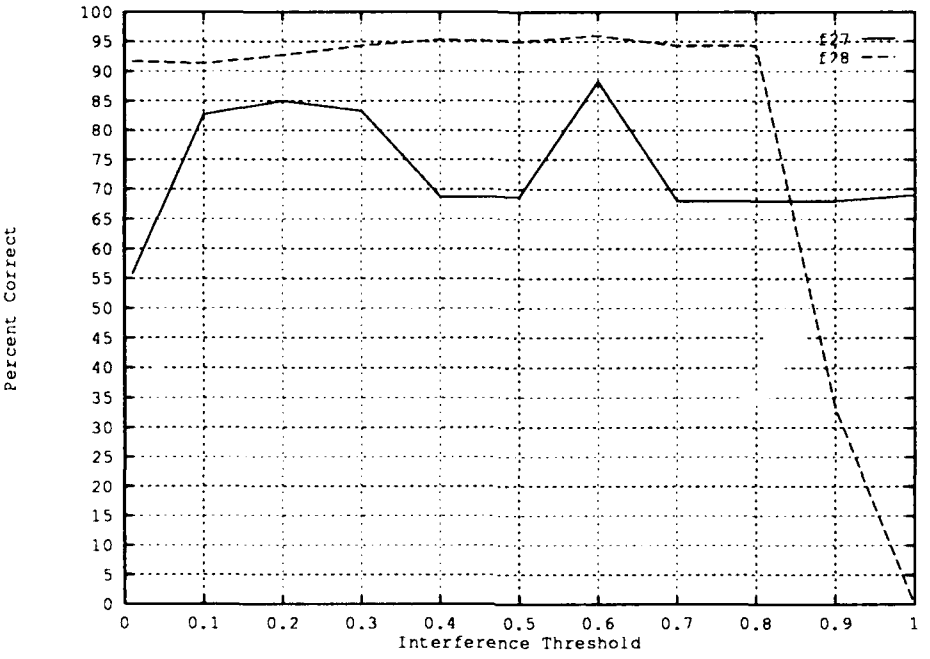

Figure 51. Percent Agree vs Interference Threshold

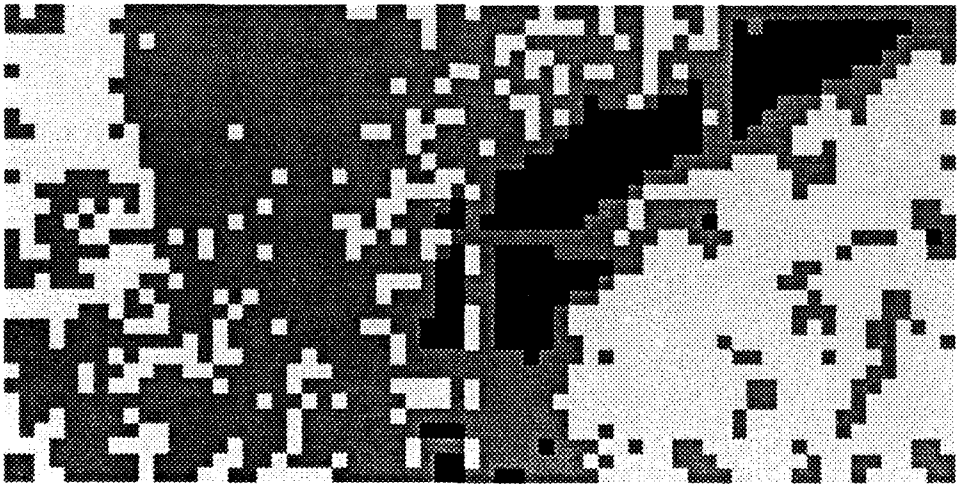

Figure 52. Average Threshold of 0.6, Trn F28 Test F27

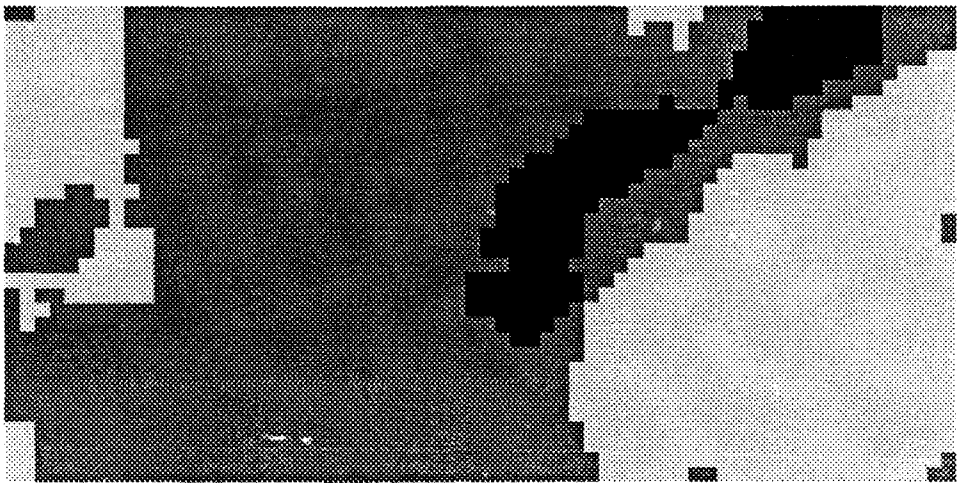

Figure 53. Average Threshold of 0.6, Trn F28 Test F27, 5 x 5 Median Filter

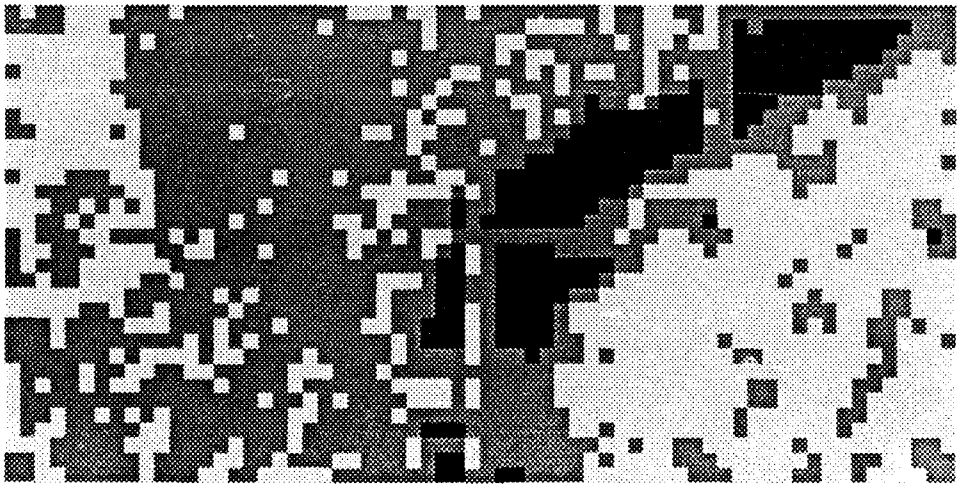

Figure 54. RBF Segmentation of F27, Trained on F28

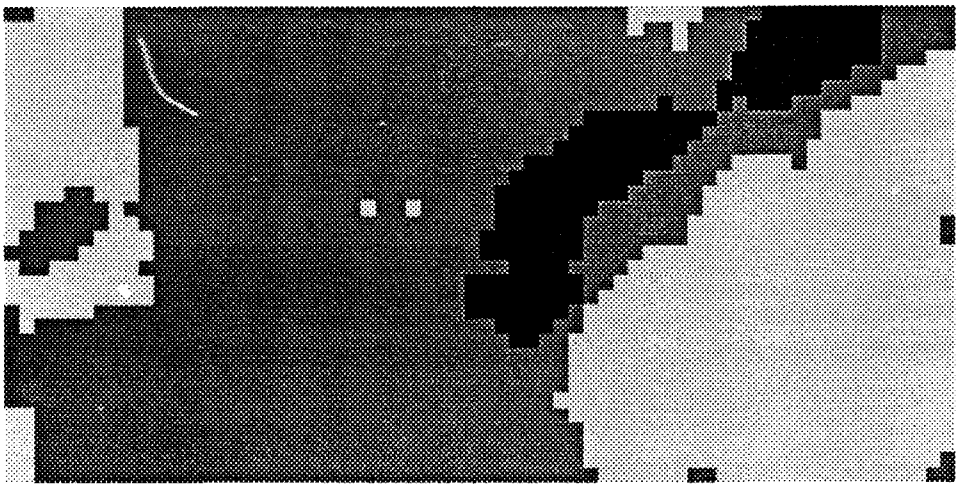

Figure 55. RBF Segmentation of F27, Trained on F28, 5 x 5 Median Filter

V. Conclusions and Recommendations

5.1 Introduction

This research investigated the use of Gabor functions and neural networks for segmentation of high resolution (1 foot by 1 foot) polarimetric SAR imagery. The interest was to determine methods suitable for separating image regions such as forest, field, shadow, road, water, and culture. The following questions were those to be answered during this work.

- Is there a particular orientation, pitch or combination of the Gabor filter(s) that will correlate well with trees, and still others with roads?
- How are these frequencies, rotations, and combinations selected?
- What size Gabor filter provides best segmentation of this imagery?
- What is the best method to train the Kohonen network for maximum separation?
- Is there a particular Kohonen neighbor and conscience rule which combines to quicken or increase the separation provided by this type net?
- How is the Kohonen network calibrated?
- Does a Kohonen / radial basis function hybrid network provide automatic calibration of the Kohonen and data classification?
- What are the proper training features for the network?

Frequency and rotation selections were made using FFTs of image regions. Frequency analysis showed coefficients were somewhat overlapping with some frequencies being more strongly oriented to certain classes. Processing demonstrated that frequencies of below approximately 1.5 cycles per window (32 pixels) provided

the best Gabor results. Results also seem to indicate that filters of 16 pixels are better for culture selection; however, more investigation is required here. Segmentation of naturally occurring areas was provided using only four cosine Gabor frequency and rotation combinations operating on only a single SAR polarization. Using various rotations of circularly symmetric Gabor filters had little effect on Gabor image results tested in early processing. This was determined visually prior to defining the final frequency and rotation values shown in Table 2.

Results obtained from Kohonen training demonstrate potential for their data clustering. These networks are difficult to train by the techniques used here. Unless a better method is available, this type network should not be used.

Results obtained from the RBF were significant. Percentages of over 80 percent were found across images. Results as high as 76 percent were obtained using simple scaling on data well out of range from the training data see Table 5. Using RBF network control parameters allowed selection of small average thresholds (a vigilance type parameter) to obtained RBF testing results above 80 percent while using only 18 RBF clusters. These results demonstrate effective results are possible using Gabor preprocessing and RBF classification.

5.2 Further Research

Below are listed as suggestions for further research.

- First step is to perform localized normalization such as Lambertization to allow consistent scaling across images. This was shown effective by Joe Brickey when operating on the imagery using a fractal based, correlation dimension estimation process.(4)
- Continue testing Gabor functions on a reduced data set as with the RBF training performed here. This data reduction seems to provide good results

and reduces processing time. The best technique might be sampling the image and filter prior to Gabor correlation.

- Test other filter sizes (4 by 4 or 8 by 8). Larger filters provided good results for naturally occurring regions, but smaller filters (16 by 16 pixels) seemed better for finding cultural borders.
- Continue using the RBF network. Once the normalization problem is solved, the RBF should provide high percentage segmentation results across images.
- Add other polarizations to the Gabor preprocessing as features for neural net processing.

Appendix A. *Other RBF Results*

These results are a continuation from the hand segmentation comparison section.

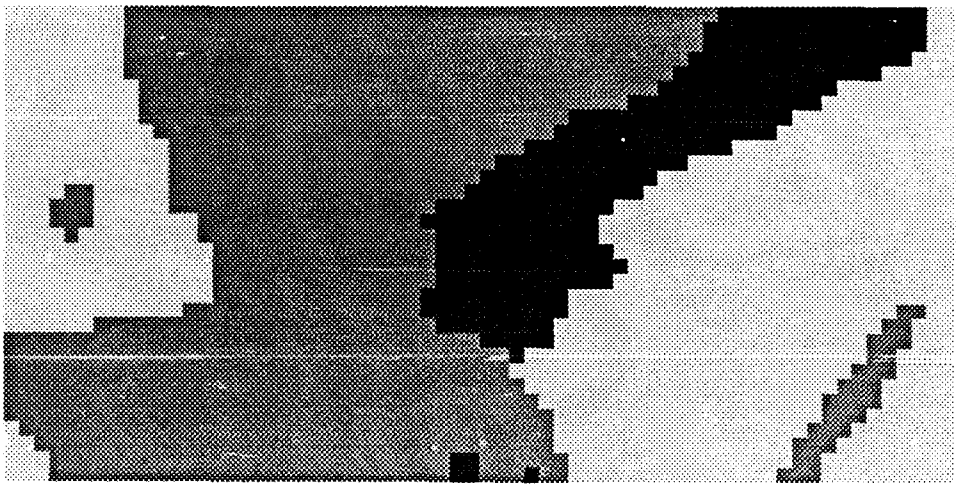

Figure 56. Hand Segmented Image m85f27

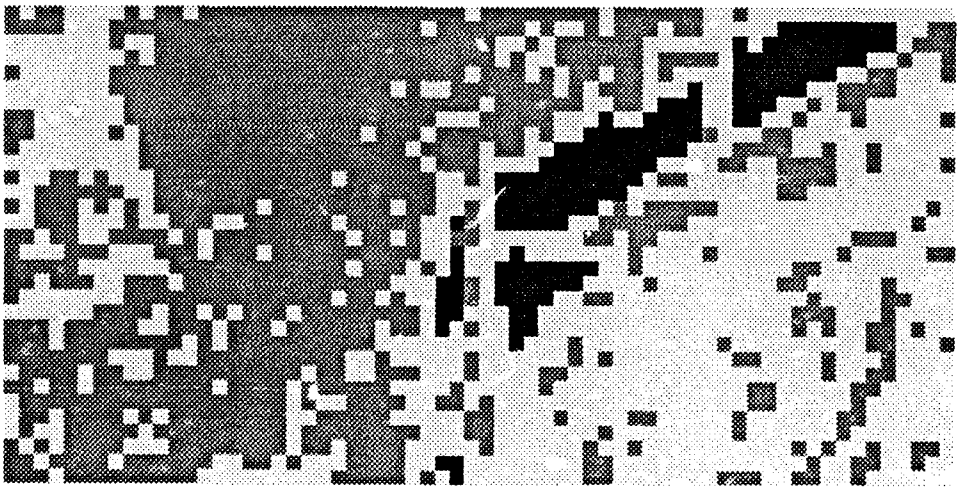

Figure 57. RBF Segmentation of F27, Trained on F28

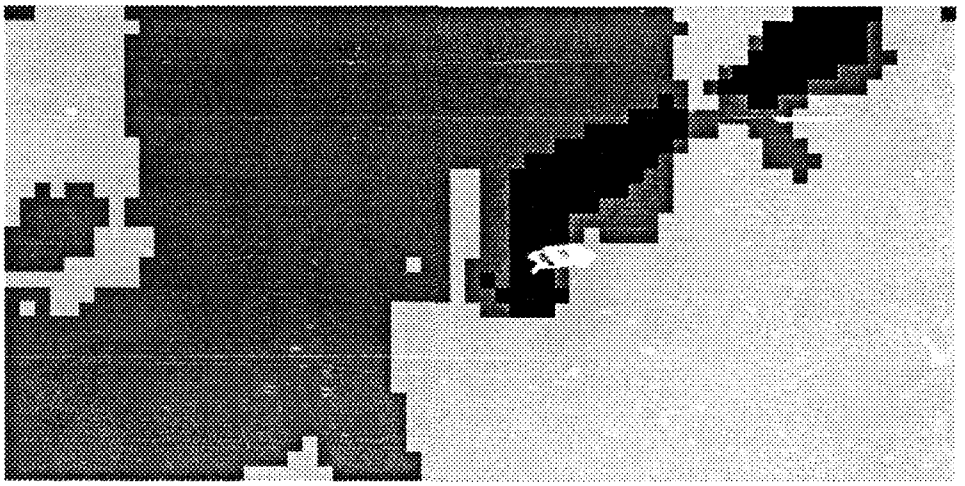

Figure 58. RBF Segmentation of F27, 5 x 5 Median Filter

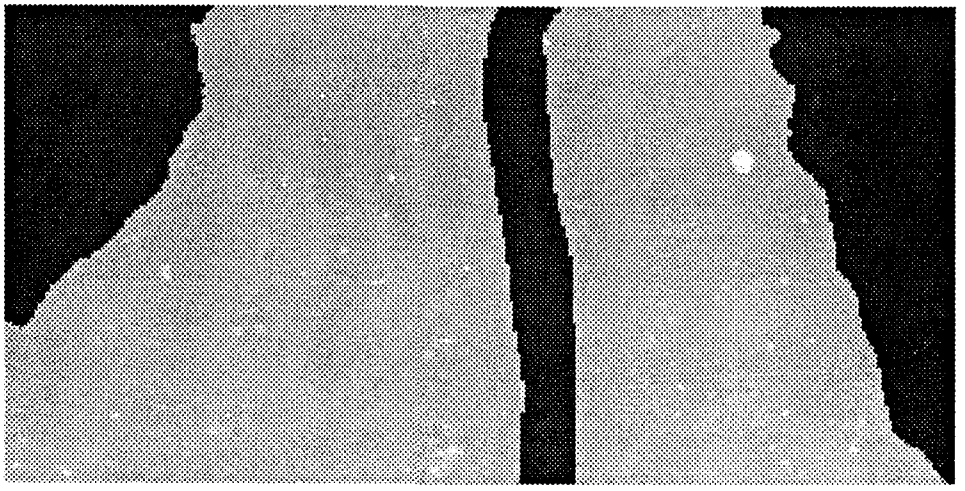

Figure 59. Hand Segmented Image m98f11

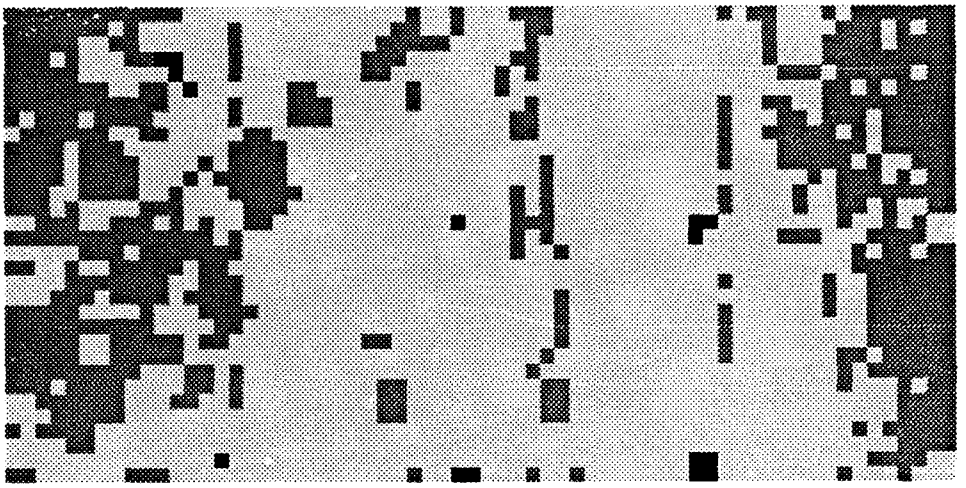

Figure 60. RBF Segmentation of F11, Trained on F28

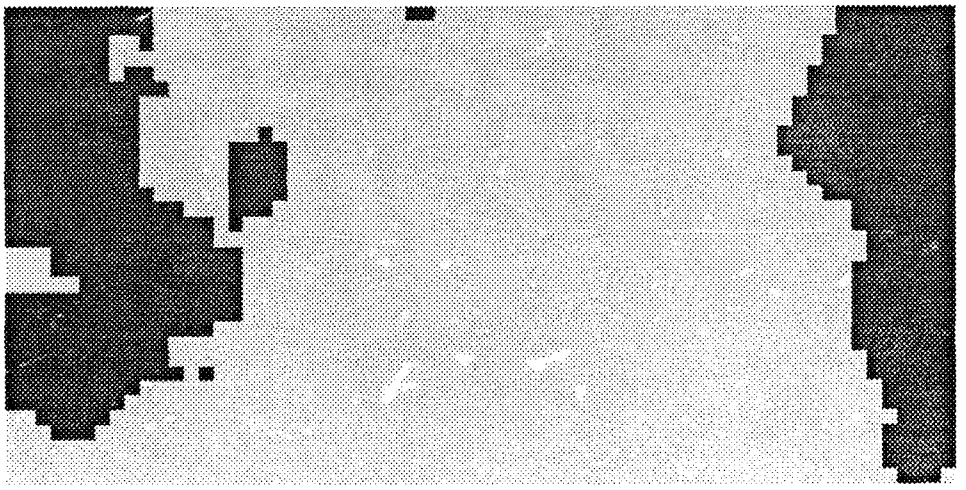

Figure 61. RBF Segmentation of F11, 5 x 5 Median Filter

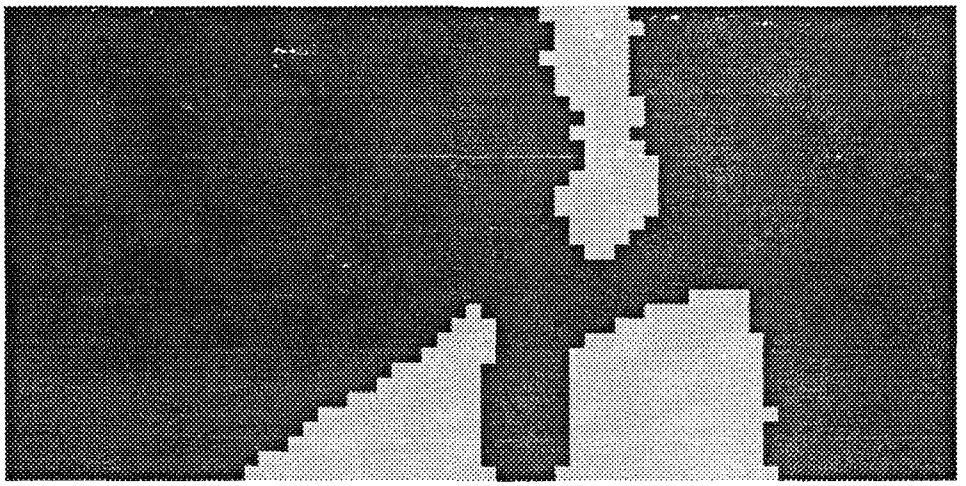

Figure 62. Hand Segmented Image m98f12

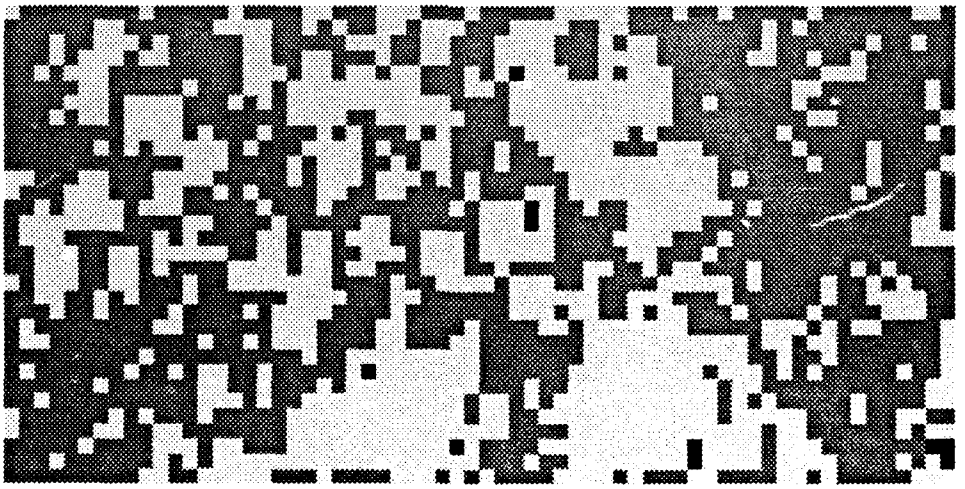

Figure 63. RBF Segmentation of F12, Trained on F28

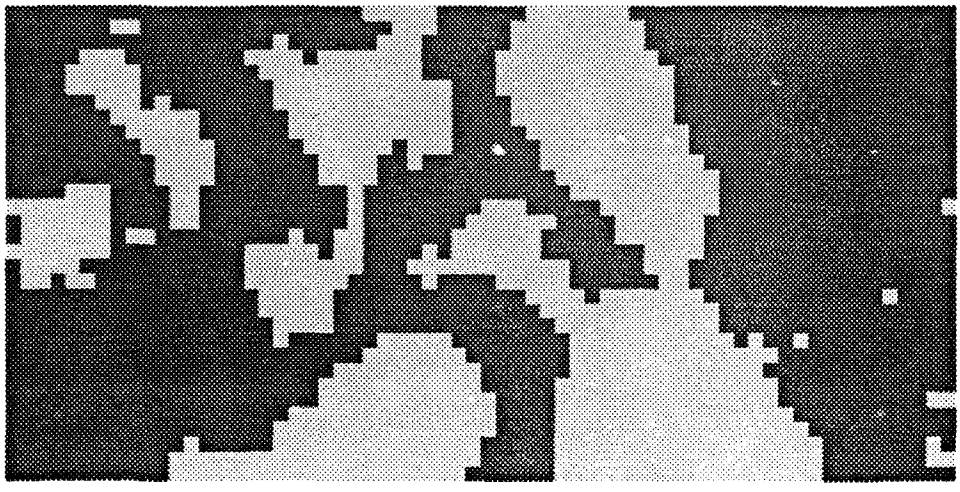

Figure 64. RBF Segmentation of F12, 5 x 5 Median Filter

Appendix B. *Software*

B.1 Gabor Code

```
/*********************************************************************
*  Program: max_gbr_slct_2048.c                                      *
*  Author:  Al L'Homme                                               *
*  Modified by :                                                     *
*  Revision: 18 September 1990                                       *
*  Variation of the gabortf1.c program where the wanted freq         *
*  and rotation are read from a datafile.                            *
*********************************************************************/

/******Modified 15 October to find the maximum value for each window
correlation and output that value for each pixel within that window.*/

#include <stdio.h>
#include <math.h>
#include <string.h>
#include <fcntl.h>
#include <ctype.h>
#include "fft_256.c"

#define FORWARD 1
#define BACKWARD -1
#define PI 3.141592653589793
#define MAX_PIC 256
#define MAX_WIDTH 4096
#define MAX_HEIGHT 512
#define CELL_OFFSET (MAX_WIDTH - 2 * MAX_PIC)
#define SEEK_SET 0
#define SEEK_CUR 1


/*****************************************************************************
*Reading in the data file.  Here wide and high are the filter dimensions. They *
* are used to zero out the bottom and right edges (aliasing)                   *
*****************************************************************************/
void Read_block(sar_file, seek_cell, real, imag, wide, high)
int sar_file;
long seek_cell;
double **real;
double **imag;
int wide;
int high;

{
    int i, j, row_bytes = 2 * sizeof(float);
```

```
int next_row_offset = CELL_OFFSET * sizeof(float);
int bytes_read, real_bytes_read, imag_bytes_read;
float foo;

if (lseek(sar_file, seek_cell, SEEK_SET) == -1L)
    {
        fprintf(stderr, "Seek error in Read_block\n");
        exit(1);
    }

bytes_read = 0;

for (i = 0; i < MAX_PIC; i++)
{
    for(j=0; j < MAX_PIC; j++)
    {
        real_bytes_read = read(sar_file, &foo, sizeof(float));
        real[i][j] = (double)foo;

        imag_bytes_read = read(sar_file, &foo, sizeof(float));
        imag[i][j] = (double)foo;
        bytes_read = real_bytes_read + imag_bytes_read;

        if ( bytes_read != row_bytes)
        {
            fprintf(stderr, "Read error in Read_block\n");
            printf("bytes_read = %d  row_bytes = %d\n",bytes_read, row_bytes);
            printf("i = %d  j = %d  \n", i, j);
            exit(1);
        }
    }

    if(lseek(sar_file, next_row_offset, SEEK_CUR) == -1L)
    {
        fprintf(stderr, "Seek error in Read_block\n");
        exit(1);
    }
}

for(i = 0 ; i < MAX_PIC; i++)
{
    for (j = 0; j < MAX_PIC; j++)
    {
       real[i][j] = sqrt(real[i][j] * real[i][j] + imag[i][j] * imag[i][j]);
       imag[i][j] = 0.0;
    }
}

for (j = MAX_PIC - high; j < MAX_PIC; j++)
{
    for (i = MAX_PIC - wide; i < MAX_PIC; i++)
```

```
        real[i][j] = 0.0;
    }

    fft2(real,imag,MAX_PIC,FORWARD);

    return;
}

void rotate(gab_array,degrees,x_size,y_size,omega,ftype,sig_x,sig_y,count)
double **gab_array;
float degrees;
int x_size, y_size;
float omega;
int ftype;
float sig_x;
float sig_y;
int count;
{
    int i,j;
    int l,m;


    double theta;
    double x,y,tx,ty;
    double x_prime,y_prime;
    double gtf();

    theta = (((double) degrees) /180)  * PI;

    for (j = 0; j < MAX_PIC; j++)
    for (i = 0; i < MAX_PIC; i++)
    gab_array[i][j] = 0;

    tx = x_size;
    ty = y_size;

    printf("gabor_array\n");
    for (j = y_size/2; j < y_size + y_size/2; j++)
    for (i = x_size/2; i < x_size + x_size/2; i++)
    {
        l = x_size - i;
        if (l < 0) l += MAX_PIC;
        m = y_size - j;
        if (m < 0) m += MAX_PIC;
        x = i; y = j;
        x_prime = x*cos(theta) - y*sin(theta) - tx*cos(theta) + ty*sin(theta);
        y_prime = x*sin(theta) + y*cos(theta) - tx*sin(theta) - ty*cos(theta);

        gab_array[l][m] = 255 / (2 * PI * sig_x * sig_x) *
                          gtf(x_prime,y_prime,sig_x,sig_y,omega,ftype);
```

```
        if(count == 0)
        {
             printf("%f\n",gab_array[l][m]);
        }
    }

}

double gtf(x,y,sigma_x,sigma_y,cycles,gtype)
double x,y;
float sigma_x, sigma_y;
float cycles;
int gtype;
{
    double f_of_x;
    f_of_x = (double)exp(-0.5*((x/sigma_x)*(x/sigma_x)+(y/sigma_y)*(y/sigma_y)));

/* If gtype = 1 then f_of_x is a cosine if 0 then is sine. */
    f_of_x *=(double) sin((x*PI* cycles)/(2*sigma_x) + (PI/2 * gtype));
    return (f_of_x);
}

/*************************************************************************
                          filter_max
*************************************************************************/
void filter_max(array, wide, high)
double **array;
int wide;
int high;
{
    int num_filter_blocks,  num_filter_row, num_filter_col;
    int filter_row, filter_col;
    long temp, filter_block_pointer;
    double max;
    int i, j;

    num_filter_row = MAX_PIC / high;
    num_filter_col = MAX_PIC / wide;
    num_filter_blocks = (MAX_PIC * MAX_PIC) / (wide * wide);

    for(filter_row = 0;filter_row < num_filter_row; filter_row++)
    {
      for(filter_col = 0;filter_col < num_filter_col; filter_col++)
      {
         filter_block_pointer = (filter_col * wide + filter_row * high *
                                 MAX_PIC);
         temp = filter_block_pointer;

         max = -1000.0;
         for(j = 0; j < high; j++)
         {
```

```
            for(i = 0;i < wide; i++)
            {
               if( array[temp] > max) max = array[];
               temp++;
            }
            temp+= MAX_PIC - wide;
          }

          temp = filter_block_pointer;
          for(j = 0; j < high; j++)
          {
            for(i = 0;i < wide; i++)
            {
               array[temp] = max;
               temp++;
            }
            temp += MAX_PIC - wide;
          }
        }
      }
}

/*************************************************************************
*                        MAIN FUNCTION                                   *
*************************************************************************/
void main(argc, argv)
int argc;
char *argv[];
{

    int k;
    char in_fname[60], out_name[40], dat_file[40];
    int num_freq_rotation_combos;
    int freq_rot_cnt,
    int in_file, out_file[64];
    int file_cntr, cnt_files;
    int width, length, type;
    int i,j;
    int sign, row_offset, output;

    long write_cnt = 0;
    long block_pointer;
    long out_pointer;
    long output_size;

    unsigned block_row, block_col, num_block_col, num_block_row;

    double **a;
    double **b;
    double **c;
    double **d;
```

```
double *xtra;

float freq, rot, al;
float **e;
float *also;

float sigma_x;
float sigma_y;
double max, min;
double rtemp, itemp;

char file_name[40];

FILE *gd;

if (argc == 3)
{
    strcpy(in_fname,argv[1]);
    strcpy(dat_file,argv[2]);
}

else
{
    printf("Usage: gabor_select input_file1 data_file\n");
    exit(1);
}

j=0;
a=(double **)calloc(MAX_PIC,sizeof(xtra));
b=(double **)calloc(MAX_PIC,sizeof(xtra));
c=(double **)calloc(MAX_PIC,sizeof(xtra));
d=(double **)calloc(MAX_PIC,sizeof(xtra));
e=(float **)calloc(MAX_PIC,sizeof(also));

if(!a || !b || !c || !d || !e)  j=1;
for(i=0;i<MAX_PIC;i++)
{
    a[i]=(double *)calloc(MAX_PIC,sizeof(double));
    b[i]=(double *)calloc(MAX_PIC,sizeof(double));
    c[i]=(double *)calloc(MAX_PIC,sizeof(double));
    d[i]=(double *)calloc(MAX_PIC,sizeof(double));
    e[i]=(float *)calloc(MAX_PIC,sizeof(float));
    if(!a[i] || !b[i] || !c[i] || !d[i] || !e[i])  j=1;
}

if(j)
{
    printf("Error! Calloc can't allocate enough memory\n");
    exit();
}
```

```
    if((gd=fopen(dat_file,"r")) == NULL)
    {
          printf("Can't open data_file");
  exit (-1);
    }
    fscanf(gd,"%d",&num_freq_rotation_combos);
    fscanf(gd,"%d",&width);
    fscanf(gd,"%d",&length);

    /*type needs to be 0 for odd symmetric gabor and 1*/
    /* for even symmetric gabor functions. */

    fscanf(gd,"%d",&type);

    fscanf(gd, "%s", out_name);  /* Reads the SAR file to be operated upon*/


/*****************************************************************************
*                 Start the READ loops                                       *
*****************************************************************************/
    k = 0;  /* Used for reference when opening output files. */

    if((in_file = open(in_fname, O_RDONLY ))== -1)
    {
        printf("Error opening input file \n");
        exit(1);
    }

/* These next two loops select the file block to operate upon.*/

    num_block_row = MAX_HEIGHT / MAX_PIC;
    num_block_col = MAX_WIDTH / (2 * MAX_PIC);

    for(block_row=0; block_row < num_block_row; block_row++)
    {
        for(block_col=0; block_col < num_block_col; block_col++)
        {
            block_pointer = (long)(block_col * MAX_PIC * 2 + block_row *
                            MAX_PIC * MAX_WIDTH) * sizeof(float);

            Read_block(in_file, block_pointer, a, b, width,length);
            file_cntr = 0;
/*****************************************************************************
                        freq_rot_cnt loop start
*****************************************************************************/
       for (freq_rot_cnt = 0; freq_rot_cnt < num_freq_rotation_combos;
                           freq_rot_cnt++)
            {
                  fscanf(gd,"%f",&freq);
                  fscanf(gd,"%f",&rot);
                  fscanf(gd,"%f",&sigma_x);
```

```
                fscanf(gd,"%f",&sigma_y);
              /*  printf("freq = %f \n  rot = %f\n sigma_x = %f sigma_y =
                      %f\n",freq, rot, sigma_x, sigma_y);*/

                al =0.0;
                sprintf(file_name,"%4.3f%s%4.3f",freq,out_name, rot);

                  if(k==0)
                  {
                if ((out_file[file_cntr] = open(file_name, O_CREAT |
                                                O_WRONLY ,0644)) == -1)
                      {
                          printf("Error opening output file \n");
                          exit(1);
                      }
                output_size = (MAX_HEIGHT * MAX_WIDTH )/ 2 * sizeof(float);
               /* printf("k = %d\n file_cntr = %d\n output_size = %ld\n", k ,
                           file_cntr, output_size);*/

                write_cnt = write(out_file[file_cntr], &al ,output_size);
                  }

              rotate(c,rot,width,length,freq,type,sigma_x,sigma_y,k);
              for(i = 0;i < MAX_PIC; i++)
              {
                 for(j = 0;j < MAX_PIC; j++)
                 {
                  d[i][j] = 0;
                 }
              }
/*****************************************************************************
Frequency correlation of the image with the filter
*****************************************************************************/

              fft2(c,d,MAX_PIC,FORWARD);/* Taking the FFT of the Filter*/
              for (j = 0; j < MAX_PIC; j++)
              {
                  for (i = 0; i < MAX_PIC; i++)
                  {

                      rtemp = c[i][j] * a[i][j] - b[i][j] * d[i][j];
                      itemp = d[i][j] * a[i][j] + c[i][j] * b[i][j];

                      c[i][j] = rtemp;
                      d[i][j] = itemp;
                  }
              }

              fft2(c,d,MAX_PIC,BACKWARD);

              filter_max(c,width, length);
```

```
 max = -100.0;
 min = 1000.0;


for (j = 0; j < MAX_PIC; j++)
    for (i = 0; i < MAX_PIC; i++)
    {
         c[i][j] /= max;
         c[i][j] *= 118;
       /*  c[i][j] += 128;*/
    }


 /* Converting to float for output */
 for(i = 0;i < MAX_PIC; i++)
{
     for(j = 0;j < MAX_PIC;j++)
      {
        e[i][j] = (float)c[i][j];
      }
}

 out_pointer = (long)(block_col * MAX_PIC + block_row *
               MAX_PIC * MAX_WIDTH/2) * sizeof(float);

 /*printf("e[i][j] values\n");*/

 for(i = 0; i < MAX_PIC;i++)
 {
   for(j = 0;j < MAX_PIC;j++)
   {
     /*printf("%f\n", e[i][j]);*/

     if(e[i][j] > max) max = e[i][j];
     if(e[i][j] < min) min = e[i][j];
   }
 }

   printf("maximum = %f\n", max);
   printf("minimum = %f\n", min);

 lseek(out_file[file_cntr], out_pointer, SEEK_SET);

 row_offset = (MAX_WIDTH / 2 - MAX_PIC) * sizeof(float);

 output = MAX_PIC * sizeof(float);

for(i = 0; i < MAX_PIC; i++)
{
 write_cnt = write(out_file[file_cntr],&e[i][0],output);
```

```
                    if(write_cnt != MAX_PIC    izeof(float))
                    {
                      printf("Error! Writing wrong amount of data.\n");
                      printf("write_cnt = %d\n", write_cnt);
                    }

                    if(lseek(out_file[file_cntr], row_offset, SEEK_CUR) == -1L)
                    {
                        fprintf(stderr, "Seek error in Write_block\n");
                        exit(1);
                    }
                }
                   file_cntr++;

               }  /* Ends the freq_rot_cnt loop */
               k = 1;
           }     /* Ends the inner read file loop */
       }         /* Ends the read file loop */

fclose(gd);
close(in_fname);
}   /* Ends the main function */
```

```
/*****************************************************************************
*  Program :   fft.c                                                         *
*  Written by: Eric Fretheim                                                 *
*                                                                            *
* The user needs to define MAX_PIC to be the size of the FFT.  For example   *
*  for a 256 x 256 FFT, MAX_PIC is 256.   Same as for a one dimensional FFT.*
*  When wanting a 2-D FFT call the fft2 routine.  One dimensional uses the   *
*  fft routine.  The picc and ipicc elements of the fft2 routine are for     *
*  real and imaginary processing.  Similar for the fft routine.  The n       *
*  defines the size of the fft.   The dir is used to tell                    *
*  fft.c which FFT direction, forward or reverse. Refer to the max_gbr_slct *
*  program for an example.                                                   *
*****************************************************************************/
#define MAX_PIC 128  /* MAX_PIC must equal n */

fft2(picc,ipicc,n,dir)
double **picc;
double **ipicc;
int n;              /* image width */
int dir;
{
double pic[MAX_PIC+1];
double ipic[MAX_PIC+1];
int i,j;

/* printf("Start Fourier transform --- rows.\n"); */

for (i = 0; i < n; i++)
{
for (j = 0; j < n; j++)
{
pic[j+1] = picc[i][j];
ipic[j+1] = ipicc[i][j];
}
fft(pic,ipic,n,dir);

for (j = 0; j < n; j++)
{
picc[i][j] = pic[j+1];
ipicc[i][j] = ipic[j+1];
}
}

for (j = 0; j < n; j++)
{
for (i = 0; i < n; i++)
{
pic[i+1] = picc[i][j];
ipic[i+1] = ipicc[i][j];
}
fft(pic,ipic,n,dir);
```

```
for (i = 0; i < n; i++)
{
picc[i][j] = pic[i+1];
ipicc[i][j] = ipic[i+1];
}
}
return;
}

fft(fr,fi,n,dir)
double *fr;
double *fi;
int n;
int dir;
{
double tr=0, ti=0;
double wr=0, wi=0;
double el=0, a=0;
int i=0, j=0;
int mr=0;
int l=0;
int k=0;
int m=0, nn=0;
int step=0;

if (dir < 0 )
for (i = 1; i < n+1; i++)
fi[i] = -fi[i];
mr = 0;
nn = n - 1;
for (m = 1; m <= nn; m++)
{
l = n/2;
while (l + mr > nn) l = l/2;
mr = mr%l + l;
if (mr > m )
{
tr = fr[m+1];
fr[m+1] = fr[mr+1];
fr[mr+1] = tr;

ti = fi[m+1];
fi[m+1] = fi[mr+1];
fi[mr+1] = ti;
}
}
l = 1;
while (l < n)
{
step = 2*l;
```

```
el = l;
for (m = 1; m <= l; m++)
{
a = 3.1415926535 * (double) (1-m)/ el;
wr = cos(a);
wi = sin(a);
for (i = m; i <= n; i += step)
{
j = i + l;
k = i;
tr = wr*fr[j] - wi*fi[j];
ti = wr*fi[j] + wi*fr[j];
fr[j] = fr[k] - tr;
fi[j] = fi[k] - ti;
fr[k] = fr[k] + tr;
fi[k] = fi[k] + ti;
}
}
l = step;
}

if ( dir < 0)
for (i = 1; i < n+1; i++)
{
fr[i] = fr[i]/n;
fi[i] = -fi[i]/n;
}

return;
}
```

B.2 Kohonen Calibration Code

```
/*****************************************************************************
* Program: build_koho_in.c                                                   *
* Written by: Al L'Homme and Joe Brickey                                     *
* Date: 6 September 1990                                                     *
* This program converts multiple data files into a single file of vectors.   *
* Each vector is composed of corresponding pixel elements from each file.    *
*****************************************************************************/
#include <stdio.h>
#include <sys/stat.h>
#include <math.h>
#include <fcntl.h>
#include <unistd.h>

FILE *outfile;
FILE *in_file;

void main(argc, argv)
int argc;
char *argv[];
{
        char in_fname[40], out_fname[40];

        char in_datnames[20][40]; /* Allows for input of 20 files. */

        int num_files, in_data_files[20], number_reads;
        long int filelength;
        float file[20][4096];
        float files_min[2048];
        long bytes_read, row_bytes, counter;
        float normal_fact, output;
        int i, j, k, l, x, y, z;
        float sum, min, max;

        if(argc != 3)
        {
                printf("Usage : build_koho_in in_data_file  out_file\n");
                exit(1);
        }
else
        /* in_data_file provides the information to the fscanf requests below */
strcpy(in_fname,argv[1]);
        strcpy(out_fname,argv[2]);
if((in_file = fopen(in_fname,"r")) == NULL)
{
printf("Error opening input file  \n");
exit(1);
}

if((outfile = fopen(out_fname,"w")) == NULL)
```

```
}

if((outfile = fopen(out_fname,"w")) == NULL)
{
printf("Error opening output file \n");
exit(1);
        }

/*****************************************************************************
* Reads information from data file: 1) # of files to be converted to vectors *
* 2) Length of each file (Assume all files equal length).                    *
* 3) Filenames                                                               *
*****************************************************************************/
        fscanf(in_file,"%d",&num_files);

/* filelength is the number of bytes in the file */
        fscanf(in_file,"%ld",&filelength);

/* Loops to open files for converting into input vectors */
        for(i = 0; i < num_files; i++)
        {
            fscanf(in_file,"%s",&in_datnames[i][0]);
            /*printf("%s\n",in_datnames[i]);*/
            if((in_data_files[i] = open(in_datnames[i], O_RDONLY )) == -1)
        {
    printf("Error opening input file %d \n",i);
    exit(1);
        }
        }

        row_bytes = 2048 * sizeof(float); /* image width times type size*/
        number_reads = (filelength / row_bytes);


        min = 3500;
        max = -25.000;


/* This loop finds the minimum value in all input files to be vectorized.
   This value will be used to scale and normalize the kohonen input data. */

       for(i = 0; i < num_files; i++)
        {

          printf("reading file %s",in_datnames[i]);
          for(j = 0; j < number_reads; j++)
          {
            read(in_data_files[i], &files_min[0], row_bytes);
            for (l = 0; l <  2048; l++)
            {
                if(files_min[l] < min ) min = files_min[l];
```

```
                if(files_min[l] > max ) max = files_min[l];
              }
            }

        }

    /*    printf("minimum value = %f\n", min);
        printf("maximum value = %f\n", max);

     */

/* This loop closes then opens files to reset pointer to file beginning. */
        for(i = 0; i < num_files; i++)
        {
                close(in_data_files[i]);
        }

        for(i = 0; i < num_files; i++)
        {
                if((in_data_files[i] = open(in_datnames[i], O_RDONLY )) == -1)
        {
    printf("Error opening input file %d \n",i);
    exit(1);
        }
        }

/* This starts the vectorization process of the input files. */

        /* Reading data_files a row at a time. */
        for(i = 0; i < number_reads; i++)
        {
            for(j = 0; j < num_files; j++)
            {
                read(in_data_files[j], file[j], row_bytes);
                /*for(k=0; k<512; k++)
                printf("filedata = %f\n", file[j][k]);*/
            }

        /* Sums and normalizes the vectors. Note that min was determined     *
        *  as the minimum value across all elements of the input data files. *
        *  Here looping on the number of vectors within an image row and     *
        *  finding a normalization factor for each vector.                   */

            for(y = 0; y < 2048; y++)
            {
                 sum = 0.0;

                 for(x = 0; x < num_files; x++)
                 {
                     /* file[x][y] = file[x][y] - min;*/
                      sum = (file[x][y]) * (file[x][y]) + sum;
```

```
                }


                normal_fact = 1 / sqrt(sum);
             /*   printf("%f\n", normal_fact);*/
                for(x = 0; x < num_files; x++)
                {
                        output = file[x][y];
                        output = normal_fact * file[x][y];
                     /*   printf("output value = %f",output);*/
                        fprintf(outfile,"%f\n",output);

                }
             }

            /* printf("i = %d\n",i); */
} /* Ends the reading loop. */

fclose(in_file);

close(outfile);
}
```

```
/*************************************************************************
* Build_cal.c                                                            *
* 20 August 1990                                                         *
* Builds calibration files for Kohonen net.                              *
* Written by: Al L'Homme and Joe Brickey                                 *
*************************************************************************/
# include <math.h>
# include <stdio.h>
# include <time.h>

#define SEEK_SET 0
#define SEEK_CUR 1

float input[512][1024][15];
float output[64][64][15];

void main(argc,argv)
int argc;
char *argv[];
{

FILE *fp1,*fp2;

int xstart,ystart,xend,yend, num_inputs;
int num_blocks_x, num_blocks_y, windowsize, num_total;
long starting_point, tempin, tempout;
long start_x_block, end_x_block, start_y_block, end_y_block;
int image_width, image_height, top_y_coordinate;
int i, j ,k, x, y, z, n;

char out_name_file[40], in_name_file[40];

if(argc == 3)
{
     strcpy(in_name_file,argv[1]);
     strcpy(out_name_file,argv[2]);
}
else
{
     printf("Usage: build_cal input_file.trn outputfile.asc \n");
     exit(1);
}


fp1 = fopen(in_name_file, "r");


printf("\n Input starting coordinates and ending coordinates:  x1 y1 x2 y2\n" );
scanf("%d %d %d %d",&xstart, &ystart, &xend, &yend);
printf("Input number of files (number of vector elements)\n");
scanf("%d", &num_inputs);
```

```
printf("Enter window size.\n");
scanf("%d", &windowsize);
printf("Enter image width and image height.\n");
scanf("%d %d", &image_width, &image_height);
/*printf("starting to do preliminary calculations\n");*/

start_x_block = (int) xstart / windowsize;
end_x_block = (int) xend / windowsize;
num_blocks_x = end_x_block - start_x_block;
top_y_coordinate = image_height -1;
start_y_block = (int) ((top_y_coordinate - ystart) / windowsize);
end_y_block = (int) ((top_y_coordinate - yend) / windowsize);
num_blocks_y = end_y_block - start_y_block;



/*printf("starting to read input\n");*/
for(i = 0; i < image_height; i++)
   {
   /*printf("entered first read loop\n");*/
   for(j = 0; j < image_width; j++)
      {
      /*printf("entered second read loop \n");*/

      for(k = 0; k < num_inputs; k ++)
         {
       /*  printf("entered third read loop");*/

         fscanf(fp1,"%f ", &input[i][j][k]);
         }
      }
   }
/*printf("finished reading input");*/
i = start_y_block;

while(i < start_y_block + num_blocks_y)
   {
  /* printf("start 1st while\n");*/
   j = start_x_block;
   while(j < start_x_block + num_blocks_x)
      {
      /*printf("start 2nd while\n");*/
      k = 0;
      while(k < num_inputs)
         {
         /*printf("start 3rd while %d %d %d %f\n",i,j,k,input[i][j][k]);*/
         output[i - start_y_block][j - start_x_block][k] = input[i][j][k];
         k++;
         }
      j++;
```

```
      }
   i++;
   }

/*printf("start writing out");*/

fp2 = fopen(out_name_file, "w");


for(i = 0; i < num_blocks_y; i++)
   {
   for(j = 0; j < num_blocks_x; j++)
      {
      for(k = 0; k < num_inputs; k ++)
         {
         fprintf(fp2,"%f \n", output[i][j][k]);
         }
      }
   }


fclose(fp1);
fclose(fp2);


}
```

```
/*****************************************************************************
* Program: Auto_net.c                                                        *
* Written by : Al L'Homme and Joe Brickey                                    *
*****************************************************************************/
# include <math.h>
# include <stdio.h>
# include <time.h>

float weights[256][16];
float input[16];

struct layer
   {
   float output;
   int numwins;
   } layerval[256];

void main(argc,argv)
int argc;
char *argv[];
{


FILE *fp1,*fp2,*fp3;

int inputloop;
float temp;
int yc,x,y,xsize,ysize, number_inputs,num_cal_vectors;
int ic, net_size,i,j;
float scalefactor,min,max;
char in_weight[60], in_data[60], out_asc[60];

if(argc == 5)
{
     strcpy(in_weight,argv[1]);
     strcpy(in_data,argv[2]);
     strcpy(out_asc,argv[3]);
}
else
{
     printf("Usage: auto_net input_file.net input_data output_file.cal
            number_of _cal_vectors \n");
     exit(1);
}

num_cal_vectors = atoi(argv[4]
/*printf("Enter number of calibration vectors\n");
scanf("%d",&num_cal_vectors);*/

fp1 = fopen(in_weight, "r");
```

```
fscanf(fp1,"%d %d %d", &xsize , &ysize, &number_inputs);
net_size = xsize * ysize;
 for( yc= 0;yc < net_size ; yc++)
     {

for(ic=0; ic < number_inputs; ic++)
   {
fscanf(fp1,"%f ", &weights[yc][ic]);
        /*printf("%f ", weights[yc][ic]);*/
   }
     }

fp2 = fopen(in_data, "r");
for(i=0;i < net_size;i++)
   {
   layerval[i].numwins = 0;
   }
for(inputloop=0 ; inputloop < num_cal_vectors ; inputloop++)
{

   for(j = 0; j < number_inputs; ++j)
      {
      fscanf(fp2,"%f ",&input[j]);
      }


   for( yc= 0;yc < net_size; yc++)
      {

temp = 0.0;

for(ic=0; ic < number_inputs; ic++)
   {
temp = temp + (weights[yc][ic] - input[ic])*(weights[yc][ic]
                                - input[ic]);
layerval[yc].output = temp;
   }
      }
    min = 30.0;
    max = 0.0;
    for(yc = 0; yc < net_size;yc++)
       {
       if(layerval[yc].output < min) min = layerval[yc].output;
       if(layerval[yc].output > max) max = layerval[yc].output;
       }
   scalefactor = 1 / (max - min);
   for(yc = 0; yc <net_size;yc++)
      {
      temp = layerval[yc].output;
      layerval[yc].output = ((temp - min) * scalefactor);
```

```
      }
   for(yc = 0; yc < net_size;yc++)
      {
      if(layerval[yc].output == 0) layerval[yc].numwins = layerval[yc].numwins + 1;
      }
}

   printf("\n\n Output Kohonen Surface Winning \n\n");
   for(x=0;x< ysize; x++)
      {
      for(i = 0; i < xsize; ++i)
          {
  printf("%d    ",layerval[x* xsize +i].numwins);
          }
          printf("\n");
      }

fp3 = fopen(out_asc, "w");

fprintf(fp3,"%s\n\n\n",out_asc);

for(x=0;x< ysize; x++)
      {
      for(i = 0; i < xsize; ++i)
          {
                  fprintf(fp3,"%d    ",layerval[x * xsize +i].numwins);
  }
  fprintf(fp3,"\n");
      }
fclose(fp1);
fclose(fp2);
fclose(fp3);
}
```

```
/*****************************************************************************
* compare_net_nodes.c                                                        *
* 10 October 1990                                                            *
* compares kohonen layers                                                    *
* Written by:  Al L'Homme and Joe Brickey                                    *
*****************************************************************************/
# include <math.h>
# include <stdio.h>
# include <time.h>

#define SEEK_SET 0
#define SEEK_CUR 1

int input[10][256];
int output[256];
int counter[5];
int class[5][256];
void main(argc,argv)
int argc;
char *argv[];
{
FILE  *fp0, *fp1,*fp2,*fp3,*fp4, *ifp[10];
int num_nodes, num_side, x, y, result, threshold;
int image_width, image_height, num_class, max;
char data_file[40], out_name_file1[40],out_name_file2[40], in_file_names[10][40];

if(argc == 5)
{
     strcpy(data_file,argv[1]);
     strcpy(out_name_file1,argv[2]);
     strcpy(out_name_file2,argv[3]);
}
else
{
     printf("Usage: compare data_file file.nodes file.region num_class\n");
     exit(1);
}

num_class = atoi(argv[4]);
fp0 = fopen(data_file, "r");
for(x = 0; x < num_class; x++)
{
    fscanf(fp0, "%s", &in_file_names[x][0]);
    if((ifp[x] = fopen( in_file_names[x], "r")) == NULL)
    {
  printf("Error opening input file %d \n",x);
  exit(1);
    }
}

/* Assumed square layer. */
```

```
printf("Input Dimension of Output Kohonen Layer (int)\n");
scanf("%d", &num_side);
printf("Threshold to be used for classifying node (int)\n");
scanf("%d", &threshold);
printf("Input Image Dimension width height (int)(int)\n");
scanf("%d %d", &image_height, &image_width);

num_nodes = num_side * num_side;
for(y=0; y < num_class; y++)
{
    for(x=0; x < num_nodes; x++)
    {
        fscanf(ifp[y],"%d", &input[y][x]);
    }
}

/* Finding which node wins most for a given class.*/
for(x=0; x < num_nodes; x++)
{
    max = input[0][x];
    for(y = 0; y < num_class; y++)
    {
        if(max < input[y][x] + threshold)
        {
            max = input[y][x];
            output[x] = y;
        }
    }
}

for(x = 0; x < num_class; x++)
    {
     counter[x] = 0;
    }

/* Counting number of nodes that win for each class, and finding its location.
   Location is the node number. From zero to num_nodes minus one.            */

for(y = 0; y < num_class; y++)
    {
    for(x=0;x < num_nodes; x++)
       {
          if(output[x] == y)
          {
             class[y][counter[y]] = x;
             counter[y] = counter[y] +1;
          }
       }
     }
fp3 = fopen(out_name_file1, "w");
fprintf(fp3,"%d\n%d\n%d\n", image_height, image_width, num_class);
```

```
for(x =0; x < num_class; x++)
    {
     fprintf(fp3,"%d\n",counter[x]);
    }
for(x =0; x < num_class; x++)
   {
   for(y = 0; y < counter[x]; y++)
      {
          fprintf(fp3,"%d\n", class[x][y]);
      }
   }
fp4 = fopen(out_name_file2, "w");

for(y = 0; y < num_side; y++)
   {
   for(x = 0; x < num_side; x++)
      {
      fprintf(fp4,"%d ", output[y * num_side + x]);
      printf("%d ", output[y * num_side + x]);
      }
    fprintf(fp4,"\n ");
    printf("\n");
    }
}
```

```
/*****************************************************************************
* final_net.c    17 Aug 1990                                                 *
* Written by: Joe Brickey  and Al L'Homme                                    *
*****************************************************************************/
# include <math.h>
# include <stdio.h>
# include <time.h>
#include <fcntl.h>
float weights[256][16];
float input[16];
unsigned char output[2048 * 512];
struct layer
   {
   float output;
   int numwins;
   } layerval[256];

void main(argc,argv)
int argc;
char *argv[];
{

FILE *fp1,*fp2,*fp3;

int inputloop, out_file;
int num_per_class[256];
float temp;
long image_size, count;
int yc,x,y,z,xsize,ysize, number_inputs, gray_scale_factor;
int ic, net_size,i,j,k, winner, num_class, image_height, image_width;
float scalefactor,min,max, grayscale_factor;
char in_weight[60], in_assign[60],in_data[60], out_usb[60];
int class[10][256];

if(argc == 5)
{
     strcpy(in_weight,argv[1]);
     strcpy(in_assign,argv[2]);
     strcpy(in_data,argv[3]);
     strcpy(out_usb,argv[4]);
}
else
{
     printf("Usage: final_net input_file.net assign.nodes data.asc
                    output_file.asc\n");
     exit(1);
}

fp1 = fopen(in_weight, "r");

fscanf(fp1,"%d %d %d", &xsize , &ysize, &number_inputs);
```

```
net_size = xsize * ysize;
 for( yc= 0;yc < net_size ; yc++)
     {
for(ic=0; ic < number_inputs; ic++)
   {
fscanf(fp1,"%f ", &weights[yc][ic]);
       /*printf("%f ", weights[yc][ic]);*/
   }
     }
fclose(fp1);

fp2 = fopen(in_assign, "r");
fscanf(fp2,"%d %d %d", &image_width, &image_height, &num_class);
for(x = 0; x < num_class; x++)
   {
   fscanf(fp2,"%d", &num_per_class[x]);
   printf("%d",num_per_class[x]);
   }
for(x = 0; x < num_class; x++)
   {
   for(y=0; y < num_per_class[x]; y++)
      {
      fscanf(fp2,"%d",&class[x][y]);
      }
   }
grayscale_factor = 255 / (num_class - 1);
printf("%d",grayscale_factor);

image_size = image_width * image_height;

fp3 = fopen(in_data, "r");

for(inputloop=0 ; inputloop < image_size ; inputloop++)
{

   for(j = 0; j < number_inputs; ++j)
      {
      fscanf(fp3,"%f ",&input[j]);
      }

   for( yc= 0;yc < net_size; yc++)
      {

temp = 0.0;

for(ic=0; ic < number_inputs; ic++)
   {
temp = temp + (weights[yc][ic] - input[ic])*(weights[yc][ic] - input[ic]);
layerval[yc].output = temp;
   }
      }
```

```
    min = 5000.0;
    max = 0.0;
    for(yc = 0; yc < net_size;yc++)
       {
       if(layerval[yc].output < min) min = layerval[yc].output;
       if(layerval[yc].output > max) max = layerval[yc].output;
       }
   scalefactor = 1 / (max - min);
   for(yc = 0; yc <net_size;yc++)
      {
      temp = layerval[yc].output;
      layerval[yc].output = ((temp - min) * scalefactor);
      }
   for(yc = 0; yc < net_size;yc++)
      {
      if(layerval[yc].output == 0)
{
winner = yc;
}
      }
   for(x = 0; x < num_class; x++)
      {
      for(y = 0; y < num_per_class[x]; y++)
 {
 if(winner == class[x][y])
    {
    output[inputloop] = (unsigned char) (x * grayscale_factor);

    }
 }
      }
}

if((out_file = open(out_usb, O_CREAT | O_WRONLY , 0644)) == -1)
{
       printf("error opening output file \n");
       exit(1);
}
count = write(out_file, output, image_size);
if(count  != image_size)
{
       printf("Error writing file \n");
       exit(1);
}
fclose(fp2);
fclose(fp3);
close(out_file);
}
```

Bibliography

1. Barmore, Gary D. *Speech Recognition Using Neural Nets and Dynamic Time Warping*. MS thesis, AFIT/GEO/GENG/88D-1, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1988.
2. Bastiaans, Martin J. "A Sampling Theorem for the Complex Spectogram, and Gabor's Expansion of a Signal in Gaussian Elementary Signals," *Optical Engineering*, *20*:594–598 (1981).
3. Bovik, Alan C. and others. "Multichannel Texture Analysis Using Localized Spatial Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(1):55–73 (January 1990).
4. Brickey, Joseph L. *Fractal Geometry Segmentation of High Resolution Polarimetric Synthetic Aperture Radar Data*. MS thesis, AFIT/GE/ENG/90D, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1990.
5. Daugman, John G. "Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-Dimentional Visual Cortical Filters," *Journal of the Optical Society of America*, *2*(7):1160–1169 (July 1985).
6. Fretheim, Eric J., "Discussions With Eric Fretheim."
7. Friedlander, Benjamin and Boaz Porat. "Detection of Transient Signals by the Gabor Representation," *IEEE Transactions on Acoustics and Signal Processing*, *37*(2):169–180 (February 1989).
8. Gabor, Dennis. "Theory of Communication," *Journal of IEEE*, *93*:429–457 (1946).
9. Jones, J. P. and other. "Information Management in the Visual Cortex," *Science* (1985).
10. Kabrisky, Matthew and Steven Rogers, "Lectures on Pattern Recognition: Scene Analysis," 1989. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, Fall 1989.
11. Lambert, L. C. *Evalution and Enhancement of the AFIT Autonomous Face Recognition Machine*. MS thesis, AFIT/GE/ENG/87, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, 1987.
12. Lippmann, Richard P. "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pages 4–22 (April 1987).
13. Martin, T. and others. "A Distortion-Invariant Pattern Recognition Algorithm," *Computer Vision, Graphics, and Image Processing*, *31*(7):50–66 (July 1985).

14. Moody, John and Christian Darken. "Learning with Localized Receptive Fields." In *Proceedings of the 1988 Connectionist Models Summer School*, pages 133–143, New Haven CT: Yale Computer Science, 1988.

15. Mueller, Michael and others. "Gabor Transforms to Preprocess Video Images for Back-Propagation," *Unknown* (Unknown).

16. Nowlan, Steven J. *Max Likelihood Competition in RBF Networks*. Technical Report Technical Report CRG-TR-90-2, Toronto Canada: Department of Computer Science, University of Toronto, February 1990.

17. Porat, Moshe and Yehoshua Zeevi. "The Generalized Gabor Scheme of Image Representation in Biological and Machine Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *10*(4):452–468 (July 1983).

18. Recla, Wayne F. *A Study in Speech Recognition Using a Kohonen Neural Net, Dynamic Programming, and Multi-Feature Fusion*. MS thesis, AFIT/GEO/GENG/88D-1, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1989.

19. Rogers, Steven K. "Tutorial Text Abstract: Introduction to Biological and Artificial Neural Networks for Pattern Recognition." Unpublished Notes for Neural Networks Class, School of Engineering, Air Force Institute of Technology, Fall 1989.

20. Ruck, Dennis. "Multisensor Target Detection and Classification," *Proceedings of SPIE Electro-Optics Conference*, *37*(2):23–31 (April 1988).

21. Stowe, Francis Scott. *Speech Recognition Using Kohonen Neural Networks, Dynamic Programming, and Multi-Feature Fusion*. MS thesis, AFIT/GE/ENG/90D, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1990.

22. Tarr, Gregory L. *Dynamic Analysis of Feedforward Neural Networks Using Simulated and Measured Data*. MS thesis, AFIT/GE/GENG/88D, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1988.

23. Thomas, Scott G. *Angle of Arrival Detection through RBF Artificial Neural Network Analysisi of Optical Fiber Intensity Patterns*. MS thesis, AFIT/GE/ENG/90D, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1990.

24. Turner, M. R. "Texture discrimination by Gabor Functions," *Biological Cybernetics*, *55*:71–82 (January 1986).

25. Webster, Michael A. and Russell L. De Valois. "Relationship Between Spatial-Frequency and Orientation Tuning of Striate-Cortex Cells," *Journal of the Optical Society of America*, *2*(7):1124–1132 (July 1985).

26. Zahirniak, Daniel R. *Characterization of Radar Signals Using Neural Networks.* MS thesis, AFIT/GE/ENG/90D-, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1990.

Vita

Captain Albert P. L'Homme was born on 28 Nov 1958 in Bremerhaven, Germany. He graduated from Leto High School in June 1976. He entered the United States Air Force in January 1977 and graduated from missile systems analyst technical school in August of 1977. Following training, he was assigned to the 19th Bomb Wing at Robins AFB where he served from September 1977 until April 1983 as a missile systems technician and munition operations controller. While being sponsored under the Airmen's Education and Commissioning Program (AECP), May 1983 - May 1986, he majored in electrical engineering at the University of Central Florida and received a Bachelor of Science in Engineering in May 1986. Following graduation, he received his commission through OTS in August 1986. Fort Meade, Maryland was his next assignment where he served with the 6940th Electronic Securtiy Wing as a project manager and project engineer for design of advanced avionic communications security equipment from September 1986 until April 1989. In December 1988, he was selected to attend the School of Engineering, Air Force Institute of Technology in the communications/pattern recognition field.

Permanent address:

6803 Wilshire Court
Tampa, Florida 33615

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1990	*Master's Thesis*

4. TITLE AND SUBTITLE
Gabor Filters and Neural Networks for Segmentation of Synthetic Aperture Radar Imagery

5. FUNDING NUMBERS

6. AUTHOR(S)
Albert P. L'Homme, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/90D-31

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
WRDC/AARA, WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This research investigates Gabor filters and artificial neural networks for autonomous segmentation of (1 foot by 1 foot) high resolution polarimetric synthetic aperture radar (SAR). Processing involved frequency correlation between the SAR imagery and biologically motivated Gabor functions. Methods for selecting the Gabor tuning parameters from the endless choices of frequency, rotation, standard deviation and bandwidth are discussed. Using these parameters, resulting Gabor correlation images were reduced in speckle, and more detailed. This research used cosine Gabor functions and operated on single polarization HH magnitude data. Following selection of the appropriate Gabor features, multiple Gabor representations were generated and converted for ANN training. Networks investigated were the Kohonen and radial basis function (RBF) algorithms. Provided are results demonstrating a Kohonen network calibration technique and how combination of Gabor processing and RBF networks provide scene segmentation. Keywords:

14. SUBJECT TERMS
Gabor Filters, Gabor Functions, Synthetic Aperture Radar, Polarimetric, Segmentation, Kohonen Neural Networks, Radial Basis Functions, Theses, (JHD)

15. NUMBER OF PAGES
117

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave Blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency. Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ..., To be published in .... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availablity Statement. Denote public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR)

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - DOD - Leave blank
DOE - DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports
NASA - NASA - Leave blank
NTIS - NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.