A RADIAL BASIS FUNCTION NEURAL NETWORK APPROACH TO TWO-COLOR INFRARED MISSILE DETECTION

THESIS

Kin-Weng Chan, Flight Lieutenant, RAAF

AFIT/GE/ENG/01M-05

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/01M-05

A RADIAL BASIS FUNCTION NEURAL NETWORK APPROACH TO TWO-COLOR INFRARED MISSILE DETECTION

THESIS

Presented to the Faculty of the Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the

Degree of Master of Science (Electrical Engineering)

Kin-Weng Chan, B.Eng.

Flight Lieutenant, Royal Australian Air Force

March 2001

AFIT/GE/ENG/01M-05

A RADIAL BASIS FUNCTION NEURAL NETWORK APPROACH TO TWO-COLOR INFRARED MISSILE DETECTION

Kin-Weng Chan, B.Eng.

Flight Lieutenant, Royal Australian Air Force

Approved:

Steven C. Gustafson	12 Feb 01
Dr. Steven C. Gustafson Committee Chairman	Date
[illegible]	12 FEB 01
Maj. Eric P. Magee Committee Member	Date
[illegible]	12 FEB 01
1Lt. Daniel J. Petrovich Committee Member and Sponsor	Date

Acknowledgments

I would like to thank Dr. Steven Gustafson for his advice and encouragement during this research. Dr. Gustafson's classes introduced and explained (in practical terms) the pattern recognition and neural network theory necessary for conducting this research. I would also like to thank 1Lt. Daniel Petrovich for sponsoring this thesis. Experience gained from research on new missile-warning techniques and technologies for this thesis will be valuable in my follow-on (electro-optical warfare research and development) assignment. Finally, I would like to thank the Royal Australian Air Force for sending me to AFIT to obtain a Masters degree in Electrical Engineering and conduct research that benefits both the Royal Australian and United States Air Forces.

Table of Contents

List of Figures

List of Tables

AFIT/GE/ENG/01M-05

Abstract

Multi-color infrared imaging missile-warning systems require real-time detection techniques that can process the wide instantaneous field of regard of focal plane array sensors with a low false alarm rate. Current technology applies classical statistical methods to this problem and ignores neural network techniques. Thus the research reported here is novel in that it investigates the use of radial basis function (RBF) neural networks to detect sub-pixel missile signatures. An RBF neural network is designed and trained to detect targets in two-color infrared imagery using a recently developed regression tree algorithm. Features are calculated for 3 by 3 pixel sub-images in each color band and concatenated into a vector as input to the network. The RBF network responds with a value of unity to feature vectors representing missiles and with zero to vectors representing background. Images are thresholded prior to application to the trained RBF network to narrow the field of interest of the RBF network and increase missile detection speed. The RBF network-based technique then generates potential target locations and probabilities that the locations correspond to missiles. Results show that the RBF network-based technique operates in near real-time and detects 100% of the missiles in data that was not used in training. Receiver operating characteristic (ROC) curves show that overly high classification thresholds can exceed the RBF network response for a true missile and result in non-detection. However, these ROC curves also show that adaptive control of the classification threshold on the RBF network output can reduce the number of false alarms to zero.

Keywords: Radial basis function neural network, two-color infrared, sub-pixel missile signature, regression tree, real-time.

A RADIAL BASIS FUNCTION NEURAL NETWORK APPROACH TO TWO-COLOR INFRARED MISSILE DETECTION

1. Introduction

1.1 Background

Surface-to-Air Missiles (SAMs) have been a significant threat to aircraft since the Soviet-made SA-2 *Guideline* and SA-3 *Goa* systems first appeared during the Vietnam War. These radio frequency (RF)-guided SAMs and their successors can be detected from their active RF emitting Target Acquisition (TA), Target Tracking (TT), Target Illumination (TI), or Missile Guidance (MG) radars. However, related missiles that operate in the infrared (IR) portion of the electromagnetic spectrum are difficult to detect due to the absence of such emissions. These missiles passively home onto aircraft heated surfaces or engine emissions and thus do not provide aircraft with an indication that they are being attacked. They present the greatest *unseen* threat to civilian and military aircraft in areas of unrest around the world because they are too numerous and easily hidden for accurate accounting by intelligence agencies.

The exhaust plume from the propulsion system is the only visible indication that an IR-guided missile is in flight. The exhaust plume is brightest at launch during missile *boost* phase and reduces considerably in the *coast-to-intercept* phase. Therefore, missile-warning systems have the best chance of detecting such a missile at launch. Existing missile-warning systems use either Pulse Doppler micro/millimeter-wave radar or ultraviolet (UV) sensors. Both technologies are fairly mature but are effective only at short ranges.

1.2 Research Motivation

Fighter aircraft (especially single-seat fighters) require missile-warning systems that occupy a small amount of space and that autonomously detect threats, declare them to the pilot, and initiate countermeasures while maintaining a low false alarm rate. Cavity-backed spiral UV sensors are typically too bulky to be fitted to fighters. However, the benefits of infrared sensing are being investigated now that advances in focal plane array (FPA) technology have led to the greater availability of *staring* infrared detectors. An FPA sensor typically consists of a two-dimensional mosaic of photo-detectors placed in the focal plane of an optical system. FPA sensors occupy less space on an aircraft while providing longer-range performance than UV sensors and potentially providing greater reliability than their predecessors through elimination of mechanical scanning. However, staring sensors impose a higher processing burden on threat detection algorithms, and the elimination of scanning means that a sensor must respond over its entire field-of-regard (Sanderson, 1996).

Infrared sensors are typically limited in their detection capability by the presence of heavy background clutter, sun glints, and inherent sensor noise. However, typical threat environments also include false alarm generators such as burning fuels, flares, exploding ordnance, and industrial sources. UV-based missile warning systems have proven to be highly susceptible to these false alarms. Imaging infrared sensors that offer multi-spectral detection are becoming more readily available for use in missile-warning systems. Multi-spectral discrimination is potentially one of the most effective ways to improve the performance of infrared missile-warning sensors, since for combustion sources such as a missile exhaust plume the intensity in one band is significantly different than for a hot black body source such as a sun glint. Thus false alarms can be reduced and threats can be identified by simultaneously comparing images from different spectral bands in real-time.

Images collected by an FPA are similar to what a human eye might see at the selected wavelength. Humans recognize objects based on a-priori knowledge and intuition plus a potentially large amount of visual, auditory, and other data. Computers cannot replicate human processing power and pattern recognition capabilities. However, neural networks use 'brain-like' algorithms that can be trained to recognize patterns and objects, and are thus a promising technology for detecting targets.

Radial basis function (RBF) neural networks constitute one such methodology. RBF neural networks are often motivated by the need to perform exact interpolation of a set of data points in a multi-dimensional space. Exact interpolation requires every input vector to be mapped exactly onto a corresponding target vector. The radial basis function approach introduces a set of *N basis functions,* one for each data point, which take the form $\phi(\|\mathbf{x}-\mathbf{x}^n\|)$ where $\phi(.)$ is some non-linear function. The *n*th such function thus depends on the distance $\|\mathbf{x}-\mathbf{x}^n\|$, usually taken to be Euclidean between $\mathbf{x}$ and $\mathbf{x}^n$. The output of the mapping is then taken to be a linear combination of the basis functions

$$h(\mathbf{x}) = \Sigma\, \omega_n \phi(\|\mathbf{x}-\mathbf{x}^n\|). \tag{1.1}$$

Both theoretical and empirical studies have shown that, in the context of the exact interpolation problem, many properties of the interpolating function are relatively insensitive to the precise form of the non-linear function $\phi(.)$. However, the most common form of basis function is the Gaussian (Bishop, 1995)

$$\phi(x) = \exp(-x^2/2\sigma^2), \tag{1.2}$$

where σ is a parameter whose value controls the smoothness of the interpolating function.

1.3 Research Objectives

The objectives of this research are as follows:

1. Propose a method for detecting sub-pixel missile signatures in two-color infrared images using a Gaussian Radial Basis Function (RBF) neural network.
2. Evaluate the performance of this detection technique by training and testing the neural network with data containing *real* missile and background signatures.
3. Determine the near real-time effectiveness of the neural network in a *real-world* missile warning system by applying previously unseen images to the network and obtaining Receiver Operator Characteristic (ROC) curves.

1.4 Thesis Organization

Chapter 2 reviews background material necessary to understand the basic concepts and results of this thesis. The concepts of target detection and recognition are reviewed, followed by a review of neural networks in general, radial basis function neural networks, and multi-layer perceptron neural networks. The theory behind the paper that inspired this line of research is presented along with a review of similarities and differences between radial basis function networks and multi-layer perceptron networks. Finally, there is a brief description of the sensor system and data collection that provided the input training and performance testing data for this thesis. Chapter 3 explains the radial basis function neural network design. Also, the concept of the 'moving window transform' method described in Chapter 2 is extended and its relationship to the missile detection technique is explained. Chapter 4 presents the results of training and testing the RBF network and describes a near real-time algorithm that significantly improves detection effectiveness. Finally, Chapter 5 summarizes the research effort, provides conclusions, and offers recommendations for further research.

2. Literature Review

2.1 Basic Concepts of Target Detection and Recognition

Pattern recognition is the scientific discipline whose goal is the classification of *objects* into a number of categories or *classes*. These objects can be images or signal waveforms or any type of measurement. In the case of an imaging infrared FPA sensor, each pixel in the image is an object. The process of performing target detection or recognition in its basic form on such an image generally consists of three stages: segmentation, feature extraction and classification.

Segmentation is the process of assigning a label to each pixel in an image. For example, a set of labels may be {background, target} or {mountains, rivers, trees, roads, ... etc}. The purpose of segmentation is to reduce the number of pixels for further processing as well as to identify multiple targets in an image and their locations. After an image has been segmented into potential targets and background, the contiguous groups of target pixels are further processed by the feature extraction stage.

The feature extraction stage computes a number of features. The selection of features for any pattern recognition technique greatly influences the performance of the detection system. The desirability of minimizing the number of features to avoid the *curse of dimensionality* is well known and will be discussed further in Section 2.7. Some example features are length-to-width ratio, average temperature (in infrared) or complexity (ratio of border pixels to total blob pixels). Once computed, the features are concatenated into a vector of numbers, which is then sent to a classification stage.

The final stage in basic pattern recognition is the classifier, which assigns a label to each input feature vector. The labels could be {target, non-target} or {x% confidence target, non-target} for target *detection*, and {tank, truck, aircraft, ... etc} for target *recognition*.

2.2 Neural Networks

Neural networks are powerful tools in non-linear statistical analysis. Artificial neural networks (ANN) are collections of mathematical models that emulate some of the observed properties of biological nervous systems and draw on analogies with adaptive biological learning. The key element of the ANN-paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements that are analogous to neurons, which are tied together with weighted-connections that are analogous to synapses (Batelle, 1997).

Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. Such adjustments are true for ANNs as well, where learning typically occurs by example through training or exposure to a truthed set of input/output data, where the training algorithm iteratively adjusts the connection weights. These connection weights store the knowledge necessary to solve specific problems (Batelle, 1997).

Although ANNs have been studied since the late 1950s, it was not until the mid-1980s that algorithms became sophisticated enough for general applications. The advantages of ANNs lie in their resilience to distortions in the input data and their capability for learning. ANNs often excel at solving problems that are too complex for conventional technologies (e.g. problems that do not have an algorithmic solution or for which such a solution is too difficult to find). Some of the more popular ANNs include the multi-layer perceptron network (which is generally trained with the back-propagation-of-error algorithm), learning vector quantization, the radial basis function network, as well as Hopfield and Kohonen networks (Batelle, 1997).

2.3 Radial Basis Function Neural Networks

The radial basis function methods introduced in Chapter 1 have their origins in techniques for performing exact interpolation of a set of data points in a multi-dimensional space (Powell, 1987). However, an exact interpolating function for noisy data is typically highly oscillatory (which is undesirable), and since the number of basis functions is equal to the number of patterns in the data set, the mapping function may be very complex and costly to evaluate for large data sets.

In contrast, radial basis function neural network (RBFNN) models (Broomhead, 1988) provide a smooth interpolating function in which the number of basis functions is determined by the complexity of the mapping to be represented rather than by the size of the data set. Radial basis function networks are non-parametric models in that they do not have a-priori knowledge of the underlying function that fits the data. Instead, the determination of suitable centres for the basis functions becomes part of the training process, and each basis function is given its own width parameter σ_j, whose value is also determined during training (i.e. the basis functions do not all have the same σ). Finally, bias parameters are included in the linear sum that compensate for the difference between the average value over the data set of the basis function activations and the corresponding average value of the targets. With these changes to the exact interpolation formula, the form of the RBFNN mapping is

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} \omega_{kj}\phi_j(\mathrm{x}) + \omega_{k0}\,. \tag{2.1}$$

The ω_{k0} biases can be absorbed into the summation by including an extra basis function ϕ_0 whose activation is set to 1 (Bishop, 1995).

For the case of Gaussian basis functions

$$\phi_j(\mathbf{x}) = \exp\left(-\|\mathbf{x} - \mu_j\|^2/2\sigma_j^2\right), \tag{2.2}$$

where **x** is the *d*-dimensional input vector with elements x_i to x_d,

and μ_j is the vector that determines the center of basis function ϕ_j, with μ_{ji} to μ_{jd}.

A neural network diagram as shown in Figure 1 can represent this mapping function.

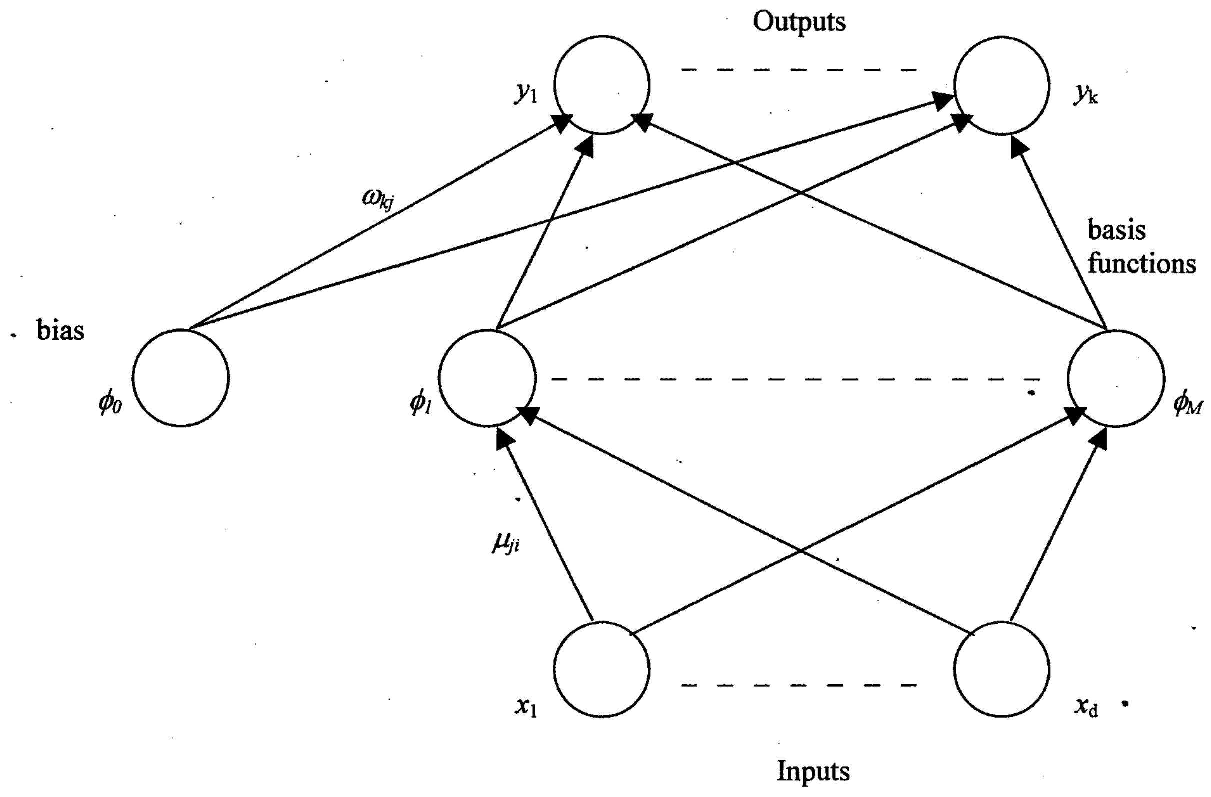


Figure 1. Architecture of a radial basis function neural network corresponding to Equation 2.1. There is only one hidden layer of neurons and each basis function acts like a hidden neuron. The hidden neurons compute the Euclidean distance between an input pattern and the vector represented by the links leading to each neuron. The lines connecting the inputs to basis function ϕ_j represent the corresponding elements, μ_{ji} to μ_{jd}, of the vector μ_j. The weights ω_{kj} are shown as lines from the basis functions to the output neurons. The activation of each output neuron is determined by a weighted sum of inputs from all hidden neurons. Biases are shown as weights from an extra basis function ϕ_0 whose output is fixed at 1 (Bishop, 1995).

Algorithms for building RBF networks often consist of two stages. The first stage selects the basis function centers μ_j and radii σ_j, and the second stage estimates the weights ω_{kj}. An RBF center could be allocated to each input point in a training set, but without further modification this scheme usually produces an overly complex model that over-fits peculiarities such as noise and training point choice. In a linear model with fixed basis functions and weights, one method for controlling the complexity of an RBF network is to add a penalty term to the sum-squared-error over the training set so that

$$E = \sum_{i=1}^{k} (t_i - y_i(\mathbf{x}_i))^2 + \lambda \sum_{j=1}^{m} \omega_j^2 , \tag{2.3}$$

where t_i is the target value for output neuron i when the network is presented with input vector $\mathbf{x}_i$.

When this combined error is optimized, large components in the weight vector are inhibited. This procedure is ridge regression or weight decay, and λ, which controls the balance between fitting the data and avoiding the penalty, is the regularization parameter. A small value for λ allows the model to fit the data closely without causing a large penalty, while a large value for λ means that a close fit is sacrificed in favor of larger weights. The parameter λ has a Bayesian interpretation, as it is the ratio of the noise corrupting the training data to the a-priori variance of the weights. However, this ratio may not be available in a practical situation, and thus it is usually necessary to establish an effective value for λ in parallel with optimizing the weights. This determination can be accomplished by using a model selection criterion such as Bayesian information criterion, generalized cross-validation, leave-one-out cross-validation, or maximum marginalized likelihood (Orr, 1999).

2.4 Multi-layer Perceptron Neural Networks

The multi-layer perceptron architecture is an extension of the *perceptron* developed by Rosenblatt (1959) to cover a variety of architectures designed to model the human brain. Use of the term perceptron generally refers to a single node. Multi-layer perceptrons have more than one layer of nodes with the nodes fully interconnected between layers. To teach the multi-layer perceptron neural network to recognize a pattern, the weights and biases in the network are adjusted so that application of a set of inputs produces the desired set of outputs. The most popular rule for training a multi-layer perceptron is the back-propagation algorithm in which an initial guess is selected for the weight vector that is then iteratively updated in the direction of the largest rate of decrease in the output-to-input error (Bishop, 1995).

A multi-layer perceptron has three distinctive characteristics (Haykin, 1999);

1. The model of each neuron in the network includes a *nonlinear activation function* that is smooth (i.e. differentiable everywhere). A commonly used form of nonlinearity that satisfies this requirement is the *sigmoidal nonlinearity* defined by the *logistic function*

$$y_j = 1 / [1+\exp(-v_j)] \, , \tag{2.4}$$

where v_j is the weighted sum of all synaptic inputs plus the bias of neuron j and y_j is the neuron output.

2. The network contains one or more layers of *hidden* neurons that are not part of the input or output of the network but that enable the network to learn complex patterns by extracting progressively more meaningful features from the input vectors.
3. The network exhibits high degrees of *connectivity* (determined by the synapses of the network). A change in the connectivity of the network requires a change in the population of synaptic connections or weights.

2.5 Automatic Target Recognition using a Multi-layer Perceptron Neural Network

Shirvaikar *et al.* (1993) showed that a back-propagation-trained two-layer perceptron with 45 hidden layer neurons was effective at automatic target recognition in high clutter thermal infrared imagery. The feature extraction stage was eliminated and raw gray-levels were utilized as inputs to the network. However, unlike the usual approach in which an entire image is the input to the neural network, this method used the neural network as a *moving window transform.* Although the authors used the word *convolution* to describe their technique, the moving window transform method was (in effect) a *sliding* of the neural network input layer over the entire image (Shirvaikar, 1993).

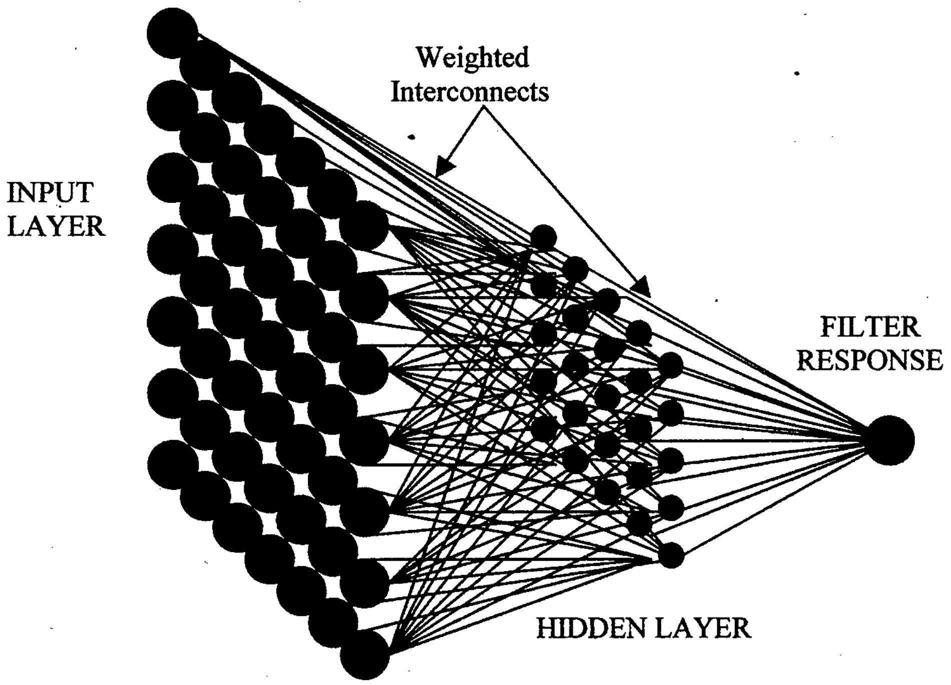


Figure 2. Depiction of the moving-window neural network concept (Shirvaikar, 1993). The input layer of the neural network is *slid* over an entire image, 128x128, 256x256, 512x512 ... etc, such that the image is divided into image *chips*, with each chip corresponding to the input layer of an individual neural network. The outputs of the neural networks are then combined produce a response that maps object locations from the spatial domain to the *probability density* domain. The outputs are high for target pixels and low for background pixels. Thus the response maps can then be thresholded to various degrees to mitigate false alarms in classifying the pixels as targets or background.

2.6 Relationship between Multi-layer Perceptron and Radial Basis Function Networks

Multi-layer perceptron (MLP) and radial basis function (RBF) neural networks are the two most commonly used types of feed-forward networks and they have more in common than most neural network literature suggests. Their fundamental difference is the way in which their hidden units combine values coming from preceding layers in the network: MLPs use *inner products*, while RBFs use *Euclidean distance*. There are also differences in the customary methods for training MLPs and RBF networks. However, most methods for training MLPs can also be applied to RBF networks (Sarle, 2000).

The MLP architecture has generally been the more popular for applications involving a large number of dimensions. The inputs are typically fully connected to the first hidden layer and each hidden layer is then fully connected to the next, with the last hidden layer fully connected to the outputs. Each layer typically uses a linear combination function. MLPs can also have *skip-layer* connections and direct connections from inputs to outputs. RBF networks usually have only *one hidden layer* for which the combination function is based on the Euclidean distance between the input vector and the weight vector (Sarle, 2000).

RBF networks usually do not have a term equivalent to the *bias* term in an MLP. However, some types of RBFs have a *width* associated with each hidden unit or with the entire hidden layer, which instead of being added into the combination function (like a bias), is divided into the Euclidean distance. A similarity between RBF networks and MLPs is apparent if the combination function is treated as the square-of-distance divided by the width, in which case the familiar *exp* or *softmax* activation functions produce members of the popular class of *Gaussian* RBF networks (Sarle, 2000).

Some important differences are as follows (Bishop, 1995):

1. The hidden unit representations of the MLP depend on weighted linear summations of the inputs transformed by monotonic activation functions. Thus the activation of a hidden unit in an MLP is constant on surfaces that consist of parallel (d-1)-dimensional hyper-planes in d-dimensional input space. In contrast, the hidden units in an RBF use distance to a prototype vector followed by transformation with a (usually) localized function. The activation of a radial basis function is therefore constant on concentric (d-1)-dimensional hyper-spheres (or more generally on (d-1)-dimensional hyper-ellipsoids).
2. An MLP forms a *distributed representation* in the space of activation values for the hidden units, since for a given input vector many hidden values typically contribute to the determination of the output value. During training, the functions represented by the hidden units must be such that when linearly combined by the final layer of weights, they generate the correct outputs for a range of possible inputs. The required interference and cross-coupling between the hidden units results in a highly nonlinear network training process with problems of local minima or nearly flat regions in the error function, which arise from near cancellations in the effects of different weights. Such cancellation can lead to very slow convergence of the training procedure, even with advanced optimization strategies. In contrast, an RBF network with localized basis functions forms a representation in the space of hidden units that is *local* with respect to the input space, because for a given input vector only a few hidden units typically have significant activations.

3. An MLP often has many layers of weights and a complex pattern of connectivity such that not all weights in any given layer are present. A variety of different activation functions may also be used within the same network. An RBF network, on the other hand, generally has a simpler architecture consisting of two layers of weights in which the first layer contains the parameters of the basis functions and the second layer forms the linear combinations of the activations of the basis functions that generates the outputs.
4. All the parameters in an MLP are usually determined at the same time as part of a single (global) training strategy involving supervised training. However, an RBF network is typically trained in two stages: the basis functions are determined first by unsupervised techniques using input data alone, and the second layer weights are subsequently found by fast linear supervised methods.

2.7 Network Selection

A Gaussian RBF network was selected for analysis in this thesis, as RBF networks have several advantages over MLPs. First, RBF networks can model any non-linear function using a single hidden layer, which removes design decisions regarding the number of layers needed. Second, the simple linear transformation in the output layer can be optimized fully using traditional linear modeling techniques, which are fast and do not suffer from problems such as local minima, which affect MLP training (StatSoft, 2000).

The radial functions used by RBF networks are also preferable to the logistic or polynomial functions used by other methods, as their response decreases (or increases) monotonically with distance and *radially* in all dimensions from a central point. The center, distance scale, and precise shape (Gaussian in our case) of the radial function are all parameters of the model and are all fixed after the first stage of training.

An RBF network is non-linear if its basis functions can move or change size or if there is more than one hidden layer. This thesis focuses on a single-layer network with Gaussian radial functions that are fixed in position and size, thus avoiding the computationally expensive non-linear *gradient descent* methods typically employed in explicitly non-linear networks (Orr, 1996).

RBFs are more sensitive than MLPs to the *curse of dimensionality* and have greater difficulties if the number of input features is large, since each additional input feature in a network adds another dimension to the space in which the training data cases reside. Thus, there must be sufficient training points to populate an N-dimensional space densely enough to determine its structure. The number of points needed for proper population grows very rapidly with dimensionality. For example, if an input variable is divided into M divisions, then the total number of cells is M^d, and this factor grows *exponentially* with the dimensionality of the input space (Bishop, 1995). Since each cell must contain at least one data point, this result implies that the quantity of training data needed to specify the mapping also grows exponentially. However, the number of features is small for this thesis, whereas the amount of data is large.

2.8 Regression Trees

Regression trees can both estimate a model and indicate which components of the input vector are most relevant for the modeled relationship. The basic idea of a regression tree is the recursive partitioning of an input space in half, and approximating the function in each half by taking the average of the output value of the data in each half. Each partition is along one of the dimensions of the input space. Thus dimensions that carry the most information about the output tend to be split earliest and most often.

The input space is recursively divided into hyper-rectangles (as it may involve more than three dimensions) that enclose all the patterns in a particular node. The nodes are organized in a binary tree, where each branch is determined by the dimension and boundary that together minimize the residual error between model and data. Thus a regression tree creates a hierarchical structure where the higher the node the coarser the feature captured by the node.

Using the tree analogy, the apex node corresponds to capturing the coarsest feature, which means that it contains all the input patterns in the data set. Progressing down the tree, each *child* node then has a subset of the input patterns of its *parent*, thus capturing finer and finer features until a terminal node (which contains a predefined minimum number of input patterns) is reached and cannot be split further (Orr, 1999).

Combining trees and RBF networks was first suggested by Kubat *et al.* (1995) in the context of classification rather than regression. Essentially, each terminal node of a regression tree contributes one hidden neuron to the RBF network. The center of the basis function is the center of the hyper-rectangle associated with the node, while the radius is the product of the half-width of the hyper-rectangle and a predefined scaling factor. Thus the tree sets the number, positions, and sizes of all potential RBFs in the network (Orr, 1999).

Using this method, model complexity is controlled by the amount of tree *pruning* and scaling of the RBFs relative to the hyper-rectangles. There is no discussion by Kubat (1995) about how to control scaling and pruning to optimize model complexity for a given data set. However, an alternative to treating every terminal node of the tree as an RBF is to have the regression tree generate a set of RBFs from which the final network is selected. The burden of controlling model complexity is thus shifted from the tree regression to the model selection criterion introduced in Section 2.3 (Orr, 1999).

2.9 Air Force Research Laboratory's Spectral Infrared Detection System

Dual-band infrared passive missile warning sensors have been under development at the Air Force Research Laboratory (AFRL) for many years where the objective has been to provide aircraft with cost-effective and robust detection and tracking of surface-to-air threats. The Spectral Infrared Detection System (SIRDS) test bed is the latest sensor developed by AFRL to evaluate and compare various spectral threat detection algorithms. The SIRDS optical sensor provides a 90° by 90° field of view to a 256 by 256-element FPA together with an integrated two-color filter wheel. The filter wheel allows the sensor to collect images that rapidly alternate between two bands in the infrared spectrum. The value of multi-color discrimination has been demonstrated for scanning sensors, particularly in heavy clutter at short ranges (Sanderson, 1996). Figure 3 shows a simplified block diagram of the sensor architecture.

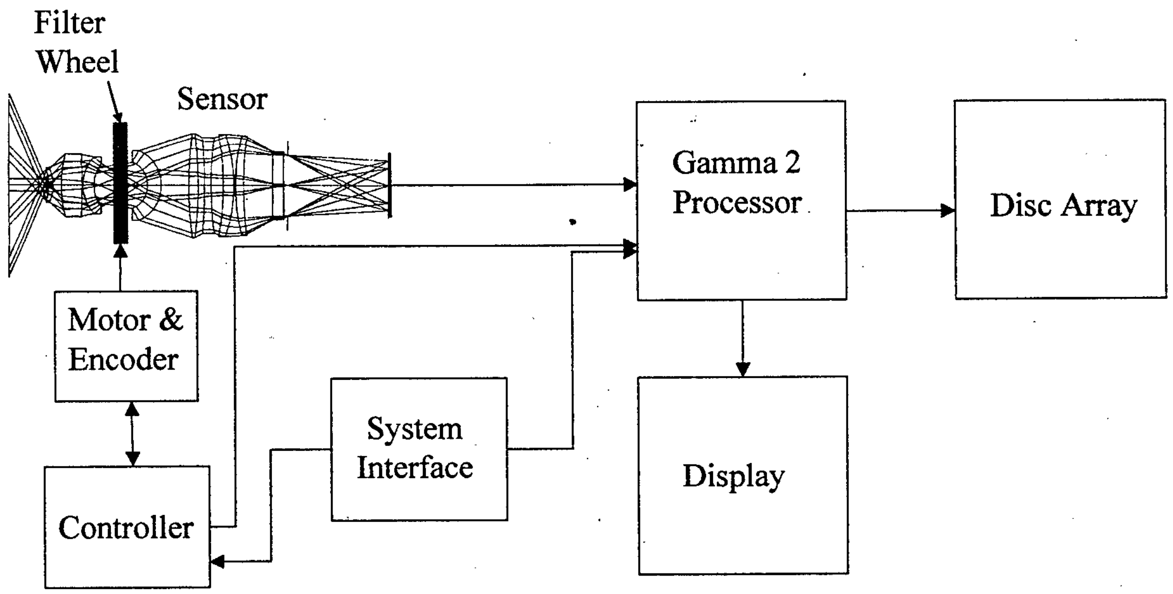


Figure 3. SIRDS Test-bed block diagram (Montgomery , 2000).

The two spectral bands used by the SIRDS and the subsequent target/background data collections are designated simply *red* and *blue* as shown in Figure 4. The widths of the bands are designed so that the photon flux is approximately equal in each band. For combustion sources such as a missile exhaust plume, the intensity in the blue band is significantly lower than in the red band. Conversely, for a hot black body source such as a sun glint, the reverse intensity relationship is true, thus permitting target discrimination by comparison of the intensities and intensity differences in each band (Montgomery, 2000).

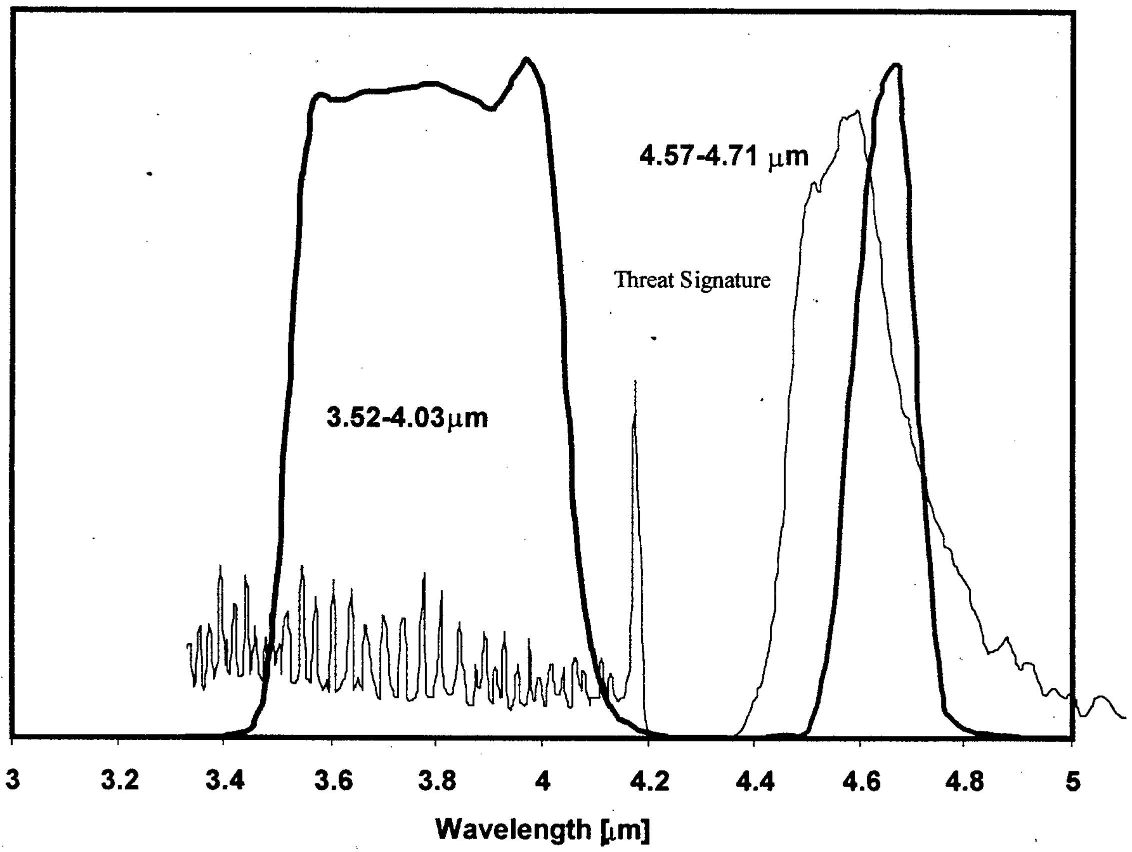


Figure 4. Pass-bands of the SIRDS sensor, together with target and background spectra. The red band lies in the region of the CO_2 ν_3 band, and the infrared signature of a missile in the powered phase is brightest in the neighborhood of this band at 4.3µm. The blue band lies in the atmospheric window just below the CO_2 ν_3 band (Montgomery, 2000).

The bands occupy four quadrants on the filter wheel such that in a single rotation of the wheel the FPA collects images in the order – *red1 / blue1 / red2 / blue2*. The responses in the red1 and red2 bands are not identical, and neither are they in the blue1 and blue2 bands due to material defects such as scratches and dust that accumulate on the different quadrants over time. The SIRDS data is organized in the same way as collected by the FPA. However, because the filter wheel to be used with a future version of the sensor consists of two regions, one for red and the other for blue, data from the red1 and red2 bands are treated as coming from the same band, as is data from blue1 and blue2. The current sensor can collect data at up to a 140 Hz frame rate. However, the background data used for training was collected at a 10 Hz frame rate during a series of test flights on 28 Aug 99. The map in Figure 5 shows the flight paths of the SIRDS sensor over populated areas and water and coastal regions along the Gulf coast and South Atlantic coast (DL1 and DL2), and inland over South Carolina (DL3), North Carolina (DL4), and Tennessee, Virginia and Kentucky (DL5). Background data for training the RBF network is from DL1 and from the first legs of the DL3, DL4, and DL5 data collection flights. Missile data is from separate flight tests (SD6 and SD8) over a static missile launch at the Eglin AFB weapons testing range. SD6 data is used to train the RBF network, while SD8 data is used to test the network and the overall detection algorithm.

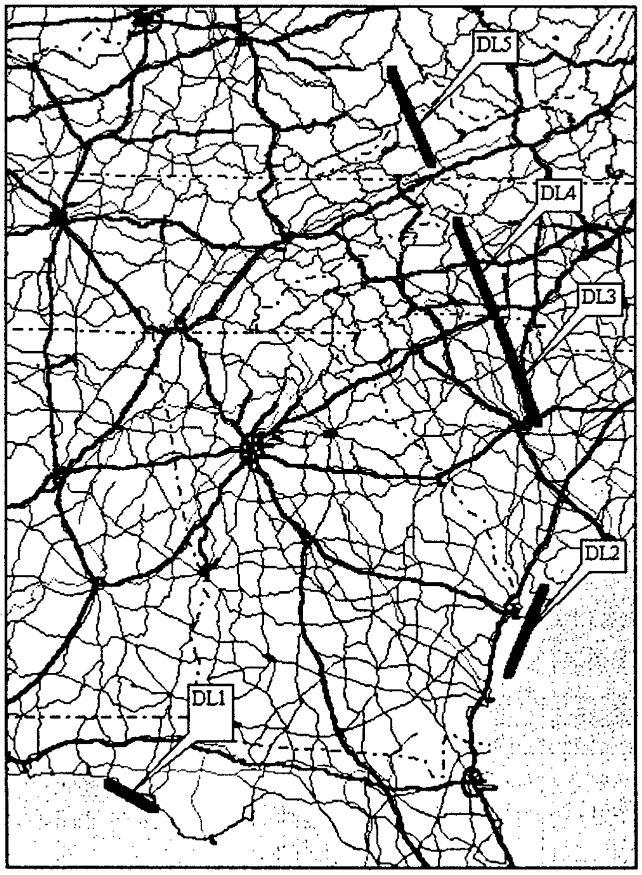


Figure 5. SIRDS data collection flight paths (Sanderson 1999).

3. Neural Network Design and Modeling

3.1 Introduction

This chapter presents the design of the Gaussian radial basis function neural network and the methodology used to analyze its effectiveness. Section 3.2 discusses the overall RBF network design and links this design to the radial basis function neural network and regression tree methods introduced in Chapter 2. Section 3.3 discusses the scope of the research, including the research objectives, and time and resource limitations. Section 3.4 discusses the design and optimization of the RBF network. Section 3.5 discusses the methods used to generate responses to input stimuli using the RBF network. Finally, Section 3.6 discusses the overall missile detection system concept, links the RBF network training to its prediction capability, and considers an algorithm that applies images to the neural network and manipulates the output to determine probable locations of missiles.

3.2 Radial Basis Function Network Overview

The RBF network is designed using functions discussed by Orr (1999) for non-parametric regression using radial basis function networks. The methods included in the toolbox employ various model selection criterion and techniques such as forward selection, ridge regression, and regression trees to control model complexity and generate RBF centers and radii. This section concentrates on *regression trees* and *leave-one-out cross-validation* model selection criterion, as these techniques were chosen for this thesis. For descriptions of the other methods and alternative model selection criterion, see Orr (1996).

3.2.1 Generating the Regression Tree

The apex node of the tree is the smallest hyper-rectangle that contains all p cases of the training input vector $\{\mathbf{x}_i\}^{p}_{i=1}$. Its size s_k (the half-width) and center c_k in each dimension k are

$$s_k = 0.5\ (\max_{i\in S}(x_{ik})\ -\ \min_{i\in S}(x_{ik})) \tag{3.1}$$

$$c_k = 0.5\ (\max_{i\in S}(x_{ik})\ +\min_{i\in S}(x_{ik})). \tag{3.2}$$

The apex node is then split into a left and right subset (S_L and S_R) on either side of a boundary (b) in one of the dimensions such that

$$S_L = \{\ i : x_{ik} \leq b\ \} \tag{3.3}$$

$$S_R = \{\ i : x_{ik} > b\ \}. \tag{3.4}$$

The mean output value on either side of the split is

$$<y_L> = 1/p_L \sum_{i\in S_L} y_i \tag{3.5}$$

$$<y_R> = 1/p_R \sum_{i\in S_R} y_i\ , \tag{3.6}$$

where p_L and p_R are the number of patterns in each subset. The residual square error between model and data is then

$$E(k,\ b) = 1/p\ (\sum_{i\in S_L}(y_i - <y_L>)^2\ +\ \sum_{i\in S_R}(y_i - <y_R>)^2\ ). \tag{3.7}$$

The split that minimizes this value over all possible k dimensions and b boundaries is used to create the child nodes. These children then become the apex for their own trees and are split recursively in the same manner until a node cannot be split without creating a child containing fewer patterns than a predefined minimum number p_{min}, which is a parameter of the method. Since the size of the regression tree does not determine the model complexity, there is no need to prune the tree as is normally required in recursive splitting methods (Orr, 1999).

3.2.2 Generating RBFs

Each node in the tree is associated with a hyper-rectangle of input space having a center $\mathbf{c}$ and size $\mathbf{s}$. To convert a hyper-rectangle into a Gaussian RBF, the center of the hyper-rectangle is used as the center of the RBF, and its size is multiplied by a scaling constant α to make the RBF radius $\mathbf{r} = \alpha\mathbf{s}$. The scaling constant is also a parameter of the method (like minimum-members) and is the same value for all nodes (Orr, 1999).

3.2.3 Selecting RBFs

In the standard methods for subset selection, RBFs generated by the regression tree are treated as an unstructured collection with no distinction between RBFs associated with different nodes. However, intuition suggests that the best order to consider RBFs for inclusion in the model is large ones first and small ones last (to synthesize coarse structure before fine details). This intuition suggests searching for suitable RBFs by traversing the tree from top to bottom in some form of breadth-first search. However, the size of a hyper-rectangle (in terms of volume) on one level is not guaranteed to be smaller than the size of all the hyper-rectangles in the level above (besides its parent). Thus the algorithm has a measure of backward elimination as well as forward selection in order to dynamically adjust the set of suitable RBFs by replacing selected RBFs with their children. This procedure avoids the situation where a parent RBF blocks selection of any of its children who would have been chosen in preference had they been considered first (Orr, 1999).

The algorithm depends on the concept of an *active list* of nodes. At any given moment in the selection process, only these nodes and their children are considered for inclusion or exclusion from the model. Every time RBFs are added or deleted from the model, the active list expands by replacing a node with its children. Eventually, the active list reaches the terminal nodes and the search is completed.

The steps of the algorithm are described in greater detail as follows (Orr, 1999):

1. Initialize the active list with the apex node and the model with the RBF associated with this node.
2. Consider, for all non-terminal nodes on the active list, the effect (on the model selection criterion) of adding both or just one of the children RBFs (three possible modifications to the model). If the parent RBF is already in the model, consider the effect of removing it before adding one or both children RBFs, or of just removing it (a further four possible modifications).
3. Choose the modification that most decreases the model selection criterion. The total number of possible modifications to the model is somewhere between three and seven times the number of active non-terminal nodes, depending on how many of their RBFs are already in the model. The choice then updates the current model and removes the node involved from the active list, replacing it with its children. If none of the modifications result in a decrease in the model selection criterion, then the algorithm chooses one of the active nodes at random and replaces it with its children, but does not alter the model.
4. Return to step 2 until all the active nodes are terminal nodes.

3.2.4 Model Selection Criteria

Model selection criteria are estimates of prediction error, which is an estimate of how well the trained model performs on future (unseen) inputs. The best model is the one whose estimated prediction error is least. The methods in the software package by Orr (1999) can be configured to use a variety of different model selection criteria. Four of these criteria are based on modifying the training set sum-squared-error to take into account the effective number of parameters in the model. They are: Unbiased Estimate of Variance (UEV), Final Prediction Error (FPE), Generalized Cross-Validation (GCV), and Bayesian Information Criterion (BIC), and are available to all methods. Two other model selection criteria are also offered as alternatives in certain methods: Leave-One-Out cross-validation (LOO) and Maximum Marginalized Likelihood (MML). Only LOO is discussed in this chapter; see (Orr, 1996) for descriptions of the other model selection criteria.

If data points are numerous, the data set can be partitioned in several different ways and the prediction error averaged over the different partitions. This procedure is the basis of leave-one-out cross-validation, where p patterns are split into a training set of p - 1 and a test set of 1, and the squared-error on the *left-out* pattern is averaged over the p possible ways of partitioning the set. The advantage of this criterion is that all the data can be used for training; none has to be held back for testing. An advantage of LOO for linear models such as RBF networks with fixed centres is that the prediction error can calculated *analytically* (Orr, 1996) as

$$<\sigma^2_{\mathrm{LOO}}> = \mathbf{y}^{\mathrm{T}}\,\mathbf{P}\,(\mathrm{diag}(\mathbf{P}))^{-2}\,\mathbf{P}\,\mathbf{y}\,/\,p\,, \qquad (3.8)$$

where **P** is the *projection matrix*,

$$\mathbf{P} = \mathbf{I}_p - \mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^{\mathrm{T}}\,, \qquad (3.9)$$

H is the *design matrix*,

$$\mathbf{H} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_m(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_m(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_p) & \phi_1(\mathbf{x}_p) & \phi_2(\mathbf{x}_p) & \cdots & \phi_m(\mathbf{x}_p) \end{bmatrix}, \tag{3.10}$$

$\mathbf{A}^{-1}$ is the variance matrix,

$$\mathbf{A}^{-1} = (\mathbf{H}^{\mathrm{T}}\mathbf{H} + \Lambda)^{-1}, \tag{3.11}$$

and $\mathbf{y} = [y_1\, y_2 \ldots y_p]^{\mathrm{T}}$ is the vector of training outputs.

All the regularization parameter elements of the diagonal matrix Λ are zero for our case, since the selection process limits model complexity. Once model selection is complete, the network weights are calculated by solving the equation

$$\mathbf{w} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\,\mathbf{H}^{\mathrm{T}}\,\mathbf{y}. \tag{3.12}$$

3.3 Scope of Research

The objectives of this research are as follows:

1. Propose a method for detecting sub-pixel missile signatures in two-color infrared images using a Gaussian Radial Basis Function (RBF) neural network.
2. Evaluate the performance of this detection technique by training and testing the neural network with data containing *real* missile and background signatures.
3. Determine the near real-time effectiveness of the neural network in a *real-world* missile warning system by applying previously unseen images to the network and obtaining Receiver Operator Characteristic (ROC) curves.

To meet the research objectives and stay within time and resource constraints, the scope of the research was limited to the areas detailed in the following paragraphs.

Eighteen gray-level intensities were initially used as the baseline feature set to cover the nine pixels from each of the two color bands of the SIRDS sensor (i.e., 3 by 3 element square windows encompassing the pixel containing the missile in the red and blue bands). However, a smaller set of features for training the RBF network was produced from combinations of these eighteen pixel values to avert the *curse of dimensionality*.

Only a subset of the data collected by the SIRDS sensor was used to train the RBF network due to the immense amount of data involved and the processing and memory limitations of computers. There are more than 7,000 *distinct* 3x3 windows in a single frame of SIRDS imagery. Therefore, a decision was made to only consider those 3x3 cases obtained from selected 150x150 regions in two frames (consecutive red/blue pairs) from each of the background data sets.

Each pair of red and blue band 150x150 regions provided 2500 training cases because there were 50x50 *distinct* 3 by 3 image chips in each region. The training set was hand-selected by viewing frames from each data set and identifying 150x150 regions that encompassed significant background textures in the images. The background training cases obtained this way numbered 10,000, which when combined with the 2,000 cases already obtained from the missile data set produced a 12,000x4-element input to the RBF network training algorithm. An overview of the data extraction methodology is shown in Figure 6.

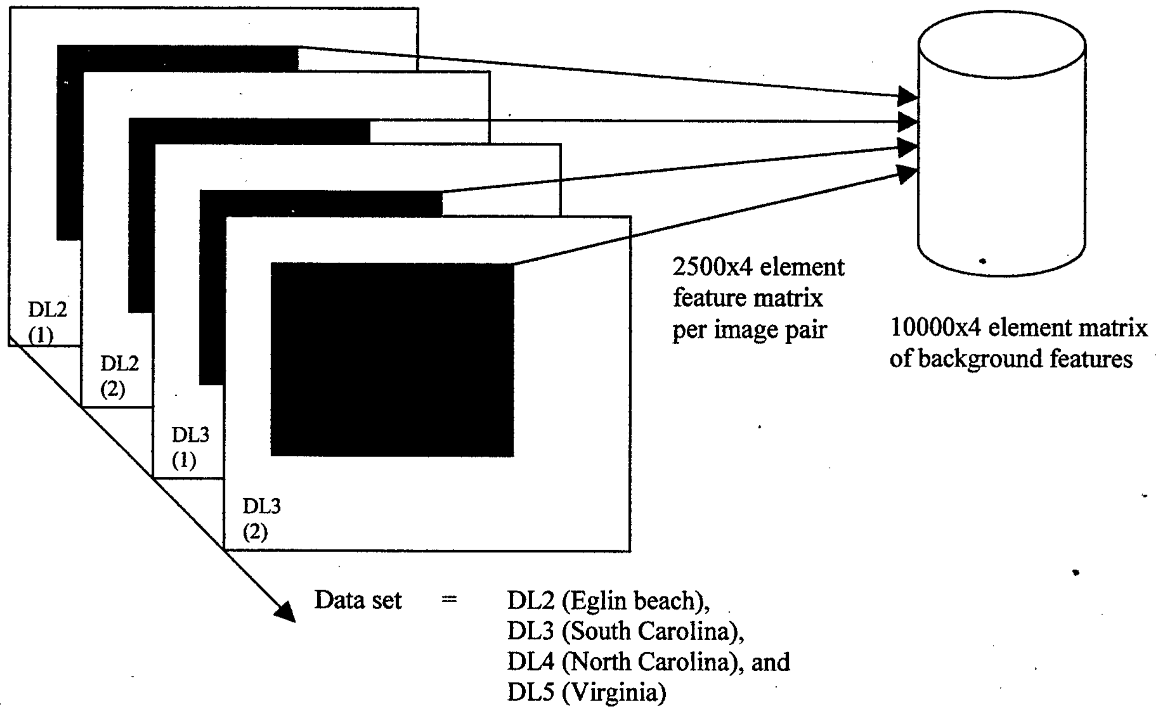

DL2 (1)
DL2 (2)
DL3 (1)
DL3 (2)
2500x4 element feature matrix per image pair
10000x4 element matrix of background features
Data set = DL2 (Eglin beach), DL3 (South Carolina), DL4 (North Carolina), and DL5 (Virginia)

Figure 6. Data extraction method for the feature set.

3.4 RBF Network Design Parameters

There are three main parameters in the design of an RBF network using the Orr (1999) Matlab functions. They are the *model selection criterion*, the minimum number of members allowed in a node p_{min} (which controls the depth of the regression tree), and the scaling parameter α (which determines the relative size between hyper-rectangles and RBFs).

Leave-One-Out cross-validation (LOO) was chosen as the model selection criterion and was used with the regression tree method (*rbf_rt_1*) for selecting suitable RBFs. This method first models the data with a regression tree, then uses the nodes in the tree to determine the centers and radii of a set of RBFs. A subset of these RBFs is then selected by considering large RBFs before smaller ones and minimizing the prediction error through LOO.

The minimum number of cases allowed in a node p_{min} has some effect on performance, and experimentation must be performed with different sets of trial values to find one that works well on a given data set (Orr, 1999). The default is a single value of 5, but any value or collection of values down to 1 may be selected. The program grows a separate regression tree for each value of p_{min} entered, and each tree gives rise to a separate set of unscaled RBFs. A regression tree for $p_{min} = 1$ takes much time to grow if presented with a large data set because the tree must keep splitting until there is a minimum of one case of any dimension of the feature vector in a node.

The scaling parameter α has a significant effect on method performance, and experimentation must also be performed to find a value that works well. The default has two trial values, one and two. However, any range of scale values can be entered, leaving the program to choose the winning network with the lowest model selection criterion score. Experience shows that if the input space has a large number of dimensions, then the best scale values are usually larger than these (Orr, 1999).

3.5 Feature Saliency

Feature saliency involves finding the features or combinations of features that carry the most information and which are thus most relevant to the target recognition solution. Feature saliency plays an important role in reducing complexity by reducing the dimensionality of the input data.

3.5.1 Input Pre-processing

Each training case is extracted from the flight-collected data sets such that there are eighteen input values per case which correspond to the pixel intensities in consecutive color bands as follows:

$$\text{Red chip} = \begin{matrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{matrix} \qquad \text{Blue chip} = \begin{matrix} b11 & b12 & b13 \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{matrix} \qquad (3.13)$$

Initially, all eighteen values were interleaved into a row such that the raw red and blue band intensities from consecutive frames are the components of a vector

$$v = [r11\ b11\ r33\ b33\ r12\ b12\ r32\ b32\ r13\ b13\ r31\ b31\ r21\ b12\ r23\ b23\ r22\ b22] \quad (3.14)$$

The mean intensities in each band are then subtracted from the raw intensities, and the resulting value is divided by the maximum intensity in each band to normalize the data.

This eighteen-dimensional vector was difficult to handle computationally, therefore the number of dimensions needed to be reduced. One of the simplest techniques for dimensionality reduction is to select a subset of the inputs and discard the remainder. However, all the inputs carried useful information, so a better method was to find combinations of the inputs that distinguished the *point-source* characteristics of sub-pixel missile signatures from the more uniform characteristics of background signatures in small (local) regions of an image.

3.5.2 Z scores for Dimensionality Reduction

Z scores are a special application of transformation rules and one method of combining multiple inputs into a single statistic. The Z score of an item is a statistical measure that quantifies how far and in which direction that item deviates from its distribution mean, expressed in units of its distribution standard deviation. The mathematics of the Z score transformation are such that if every item in a distribution is converted into its Z score, the scores have a mean of zero and a standard deviation of one. Z scores are especially informative when the distribution to which they refer is normal, as the distance between the mean and Z score is a fixed proportion of the area under the curve. The formula for converting a given value X into its corresponding Z score in a distribution is (Hoffman, 2000)

$$Z_x = (X - \mu_x) / \sigma_x . \tag{3.15}$$

Two variations of the Z score were used for reducing the number of dimensions in the input data. The Z score used by Baxley *et al.* (2000) was applied to the data from red band image chips as follows:

$$Z_B = (8 * r22) / (r11+r12+r13+r21+r23+r31+r32+r33) \tag{3.16}$$

The same formula was applied to blue band image chips.

3.5.3 Z score from Double-Gated Filtering Methods

The second Z-score-like statistic came from double-gated filtering methodology (Sevigny, 1994). In double-gated filtering, an image is scanned with a moving window that consists of two concentric sub-windows. The inner sub-window (the target gate) includes the center element plus an optional number of rings of pixels surrounding it. The outer sub-window (the background gate) incorporates pixels that lie on the perimeter of the moving window plus an optional number of inward rings. Using the Holmes method (Morin, 2000), the means and standard deviations of the pixel gray levels are evaluated for the two gates such that the output

$$O(x, y) = [\mu_T - \mu_B] / \sigma_B \tag{3.17}$$

corresponds to the center of the moving window, where μ_T is the mean of the pixels in the target gate, μ_B is the mean of the pixels in the background gate, and σ_B is the standard deviation of the pixels in the background gate. In order to use this method, each image chip is considered to have an inner sub-window with only the center pixel inside it and an outer sub-window with the remaining eight pixels surrounding the center as follows:

$$\text{Red chip} = \boxed{\begin{array}{ccc} r11 & r12 & r13 \\ r21 & \boxed{r22} & r23 \\ r31 & r32 & r33 \end{array}} \qquad \text{Blue chip} = \boxed{\begin{array}{ccc} b11 & b12 & b13 \\ b21 & \boxed{b22} & b23 \\ b31 & b32 & b33 \end{array}} \tag{3.18}$$

The second version of the Z score is then calculated for the red and blue band chips using the *Holmes parameter* as follows:

$$Z_H\,(\text{red}) = \frac{r22 - \text{mean}(r11\ r12\ r13\ r21\ r23\ r31\ r32\ r33)}{\text{standard deviation}(r11\ r12\ r13\ r21\ r23\ r31\ r32\ r33)} \tag{3.19}$$

3.5.4 Principal Component Analysis

Principal component analysis (related to the *Karhunen-Loéve transformation*) is one popular linear dimensionality reduction procedure for visualizing a multi-dimensional data space. In practice, it proceeds by first computing the means of the data values in each dimension, then subtracting off the means from the values. Next, the covariance matrix is calculated, and its eigenvectors and eigenvalues are found. Each of the eigenvectors is a *principal component.* Dimensionality reduction is then obtained by retaining the eigenvectors corresponding to the *M largest* eigenvalues and projecting the data set onto these eigenvectors to get the components of the transformed vectors in the new *M*-dimensional space (Bishop, 1995).

The technique is illustrated schematically in Figure 7 for the case of reducing data in two-dimensions to one dimension.

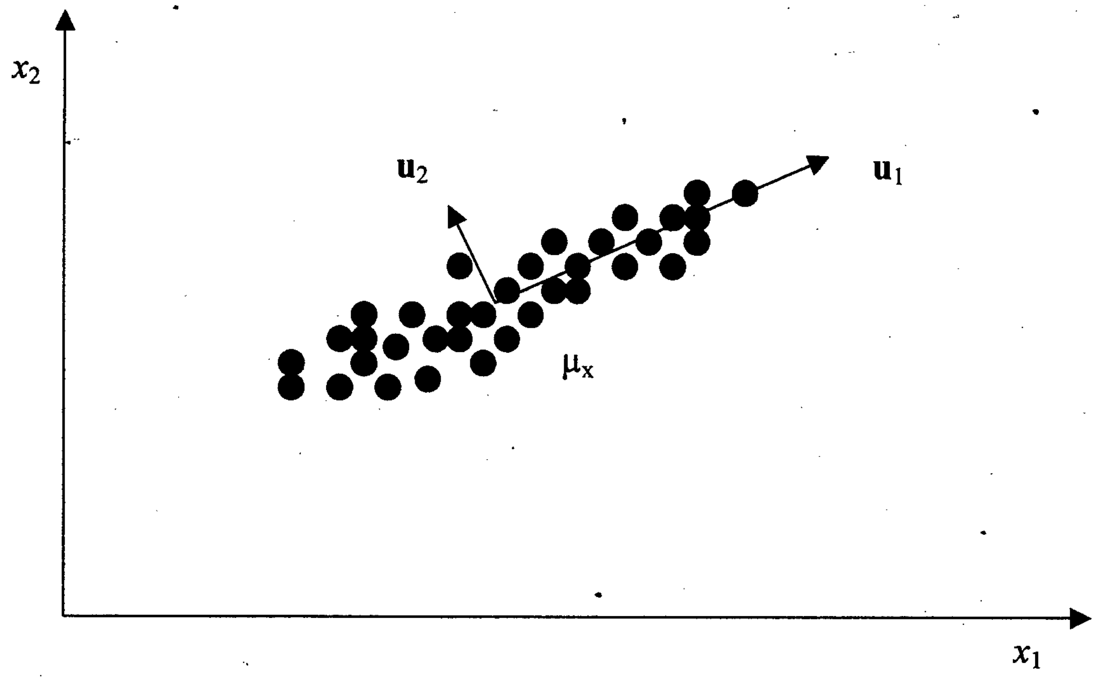


Figure 7. Schematic illustration of principal component analysis applied to data in two dimensions. In a linear projection down to one dimension, the optimum choice of projection, in the sense of minimizing the sum-of-squares error, is obtained by first subtracting off the mean, μ_x, of the data set, then projecting onto the first eigenvector $\mathbf{u}_1$, of the covariance matrix (Bishop, 1995).

This method can be regarded as a form of unsupervised learning since it relies on the input data itself without reference to the corresponding target data. However, this neglect of target information implies that the result can also be significantly sub-optimal in preserving the discriminatory capabilities of the data as shown in Figure 8.

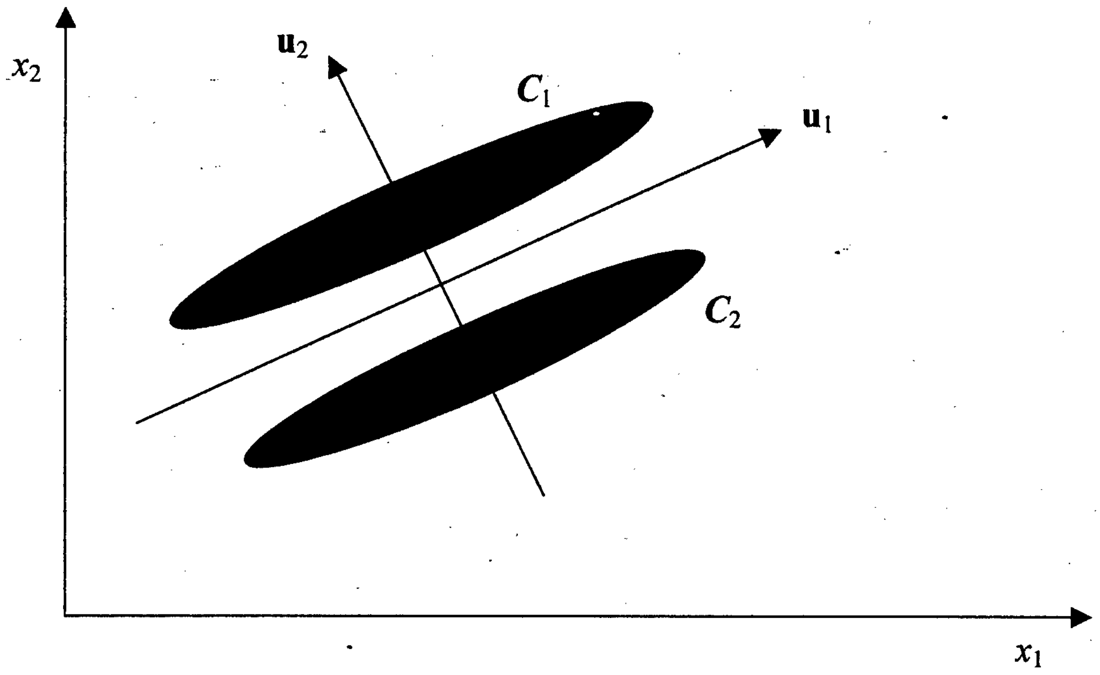


Figure 8. Example of principal component analysis resulting in discriminatory information being discarded. Dimensionality reduction to one dimension using principal component analysis projects the data onto the vector $\mathbf{u}_1$, which would remove all ability to discriminate the two classes C_1 and C_2. Full discrimination capability is preserved, however, if the data is projected onto the vector $\mathbf{u}_2$ instead (Bishop, 1995).

3.5.5 Automatic Relevance Determination

The program *rbf_rt_1* by Orr (1996) has a feature that is not shared by any of his other RBF network training methods, but which is similar to performing a principal component analysis. The *rbf_rt_1* method monitors which dimensions of the input data are first to be split and how often each dimension is split during tree growth. These (tree) splitting statistics provide a form of automatic relevance determination, since they identify dimensions that are seldom split (or not among the first to be split) and which thus carry less information than the dimensions that are frequently split.

3.6 RBF Network Prediction

Once the RBF network is trained and its centers, radii, and weights are returned, the network is used to make predictions, i.e. to calculate the probability of a missile in any part of an image. The first task is to build a *design matrix* **H**. The term design matrix implies deliberate choice or design of the inputs of the training set; however, the inputs are usually not controlled in practical applications of neural networks (Orr, 1999).

For models that are linear with respect to the weights w_j,

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_j\, h_j(\mathbf{x})\,, \tag{3.13}$$

where for our case of Gaussian basis functions,

$$h_j(\mathbf{x}) = \exp(-(\mathbf{x} - \mathbf{c}_j)^2 / r_j^2)\,. \tag{3.14}$$

The system of linear equations to be solved (in a least squares sense) is

$$h_1(\mathbf{x}_1)w_1 + h_2(\mathbf{x}_1)w_2 + \ldots + h_m(\mathbf{x}_1)w_m = y_1\,,$$

$$h_1(\mathbf{x}_2)w_1 + h_2(\mathbf{x}_2)w_2 + \ldots + h_m(\mathbf{x}_2)w_m = y_2\,,$$

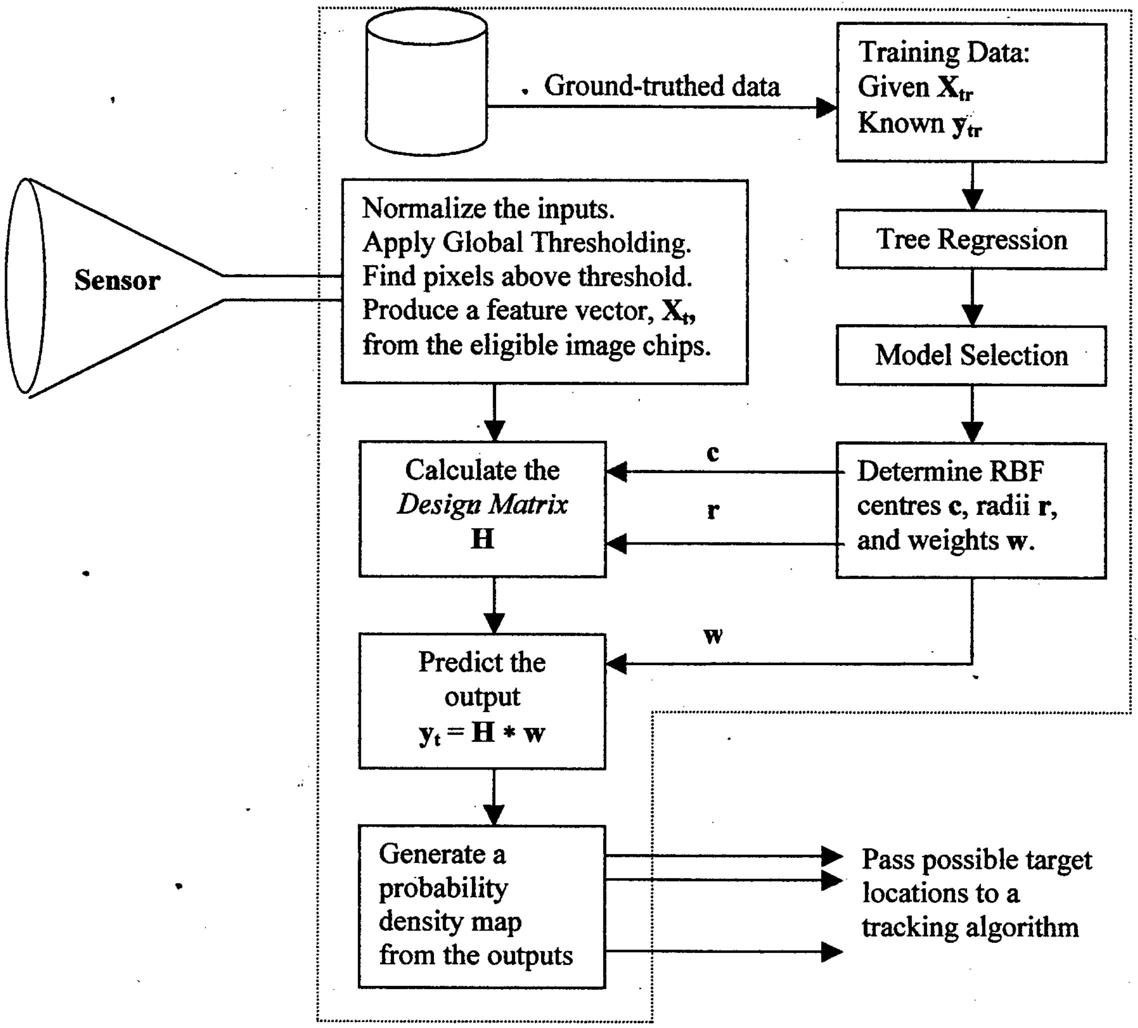
$$\ldots = \ldots\,,$$

$$h_1(\mathbf{x}_p)w_1 + h_2(\mathbf{x}_p)w_2 + \ldots + h_m(\mathbf{x}_p)w_m = y_p\,. \tag{3.15}$$

Here the design matrix **H** consists of the coefficients on the left-hand side of the system of equations, i.e., $H_{ij} = h_j(\mathbf{x}_i)$. Orr's (1999) *rbf_dm* Matlab function calculates **H** using input data, the RBF centers, and the RBF radii. The default configuration for *rbf_dm* implements Gaussian radial functions of the form $\exp(-z^2)$, where z is the distance vector. However, alternative radial functions include Cauchy, multi-quadratic, and inverse functions. Matrix-multiplying **H** with the weight vector **w** (**f** = **H** * **w**) then yields a predicted output from the RBF network.

3.7 Overall Missile Detection Concept

The parameters of the RBF network combined with the *rbf_dm* function enable prediction of the presence of missile-like signatures as shown in Figure 9.



Ground-truthed data
Training Data: Given $\mathbf{X}_{tr}$ Known $\mathbf{y}_{tr}$
Sensor
Normalize the inputs. Apply Global Thresholding. Find pixels above threshold. Produce a feature vector, $\mathbf{X}_t$, from the eligible image chips.
Tree Regression
Model Selection
Calculate the *Design Matrix* **H**
c
r
Determine RBF centres **c**, radii **r**, and weights **w**.
w
Predict the output $\mathbf{y}_t = \mathbf{H} * \mathbf{w}$
Generate a probability density map from the outputs
Pass possible target locations to a tracking algorithm

Figure 9. Missile detection algorithm block diagram. First, ground-truthed data is used to train the RBF network and obtain centers, radii, and weights in the feature space. Then red and blue band image chips are input to the RBF network in the same form as used for training. Global thresholding of the normalized inputs enables the algorithm to run in near real-time by narrowing the field of interest of the RBF network to only those image chips around pixels above the predetermined threshold. The output of the RBF network is the center pixel of each 3 by 3 image under test, but in a new (probability density) mapping in which the pixel values may vary between zero and one depending on how much (or little) they resemble background or missiles.

4. Results and Analysis

4.1 Introduction

This Chapter discusses the simulations and results of testing the Gaussian RBF neural network and missile detector configuration described in Chapter 3. Section 4.2 presents plots of some of the raw images that were used to train and test the RBF network and discusses the characteristics of the features selected for training the RBF network. Also presented are sample plots of the data set after principal component analysis. Section 4.3 presents the RBF network parameters obtained from the regression-tree/leave-one-out cross-validation training process and discusses their significance. Section 4.4 discusses results from testing the RBF network with previously 'unseen' missile data. Section 4.5 describes the design of a near real-time missile detection algorithm to complement the RBF neural network, and it discusses the results of applying this algorithm to the test data. Using Receiver Operating Characteristic (ROC) curves, Section 4.6 assesses the performance of the RBF network-based detection algorithm. Finally, Section 4.7 summarizes the results presented in Chapter 4.

4.2 Training Data

Several pre-processing procedures were used on the training data to find one that could be performed in a timely fashion by the regression tree building and RBF network selection algorithms. Although the computer used was a 128-megabyte RAM Pentium III 450MHz PC, the problem lay in the inherently memory-intensive tree building process. The depth of the tree was dependent on the parameter entered for the minimum number of cases allowed in a node. Experimentation determined that small values of this parameter (i.e., one to ten) could only be used with relatively small data sets of a few hundred cases. Large data sets required (minimum case) parameters of 100 or greater, because otherwise the computer had insufficient memory to complete the tree regression.

The scaling parameter did not affect the depth of the tree, but greatly influenced the number and locations of RBFs selected. When the number of dimensions was large, such as when all eighteen raw pixel intensities were considered as features, the optimal RBF scaling parameter. was found to be ten to twenty times the size of the hyper-rectangles in the regression tree. This result was only obtained for a small subset of the data, however, since the computer was never able to build a complete tree for the entire eighteen-dimensional data set. The scaling parameters that worked with the available computing resources changed again when the eighteen dimensions were reduced to the four chosen features. Scaling factors of two to five were found to work better when the number of dimensions was small. The regression tree algorithm requires setting applicable values for these two parameters before initiating the process. Here 100 minimum members and a scaling factor of two were used.

4.2.1 SIRDS Imagery

The imagery used in this research was collected by the SIRDS 256 by 256 focal plane array sensor. The images were collected as a series of red1, blue1, red2, and blue2 frames because the color-wheel consisted of four quadrants. In the missile data set, the featureless desert environment of the test range resulted in the missile being the only significant IR energy source in the scenes. Also, there was only one missile in each scene, so the missile-bearing pixels were always the ones with maximum intensity in each frame. Image chips were then extracted by centering a 3 by3 window on the maximum intensity pixel and using the pixel values contained in the window for calculating the features. Figures 10 and 11 show frames in the red 1 and red 2 bands, and Figures 12 and 13 show frames in the blue 1 and blue 2 bands. The view in all four figures is straight toward the ground.

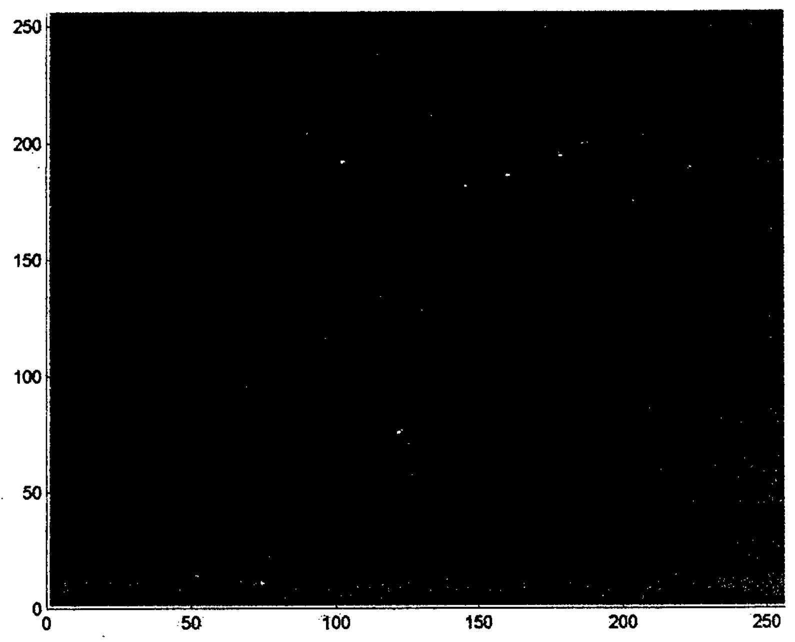


Figure 10. Missile scene in the red 1 band. The black arrow points to the missile location in the scene. The scene is mostly featureless except for the missile, since the image was collected over a desert test range.

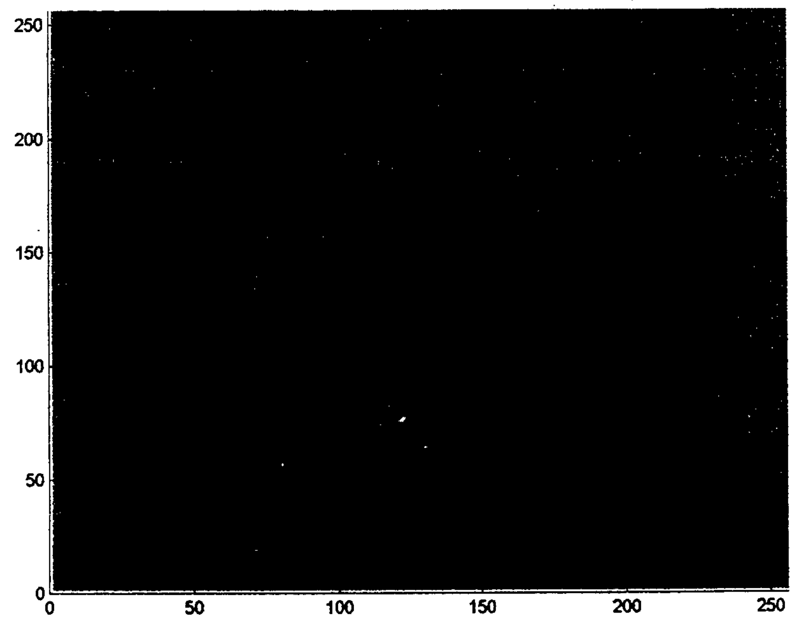


Figure 11. Missile scene in the red 2 band. The image from the second red quadrant is practically identical to the image seen in the previous figure from the first red quadrant. Therefore, the two quadrants were regarded as the same color band, which simplified the detection problem and allowed the use of fewer features.

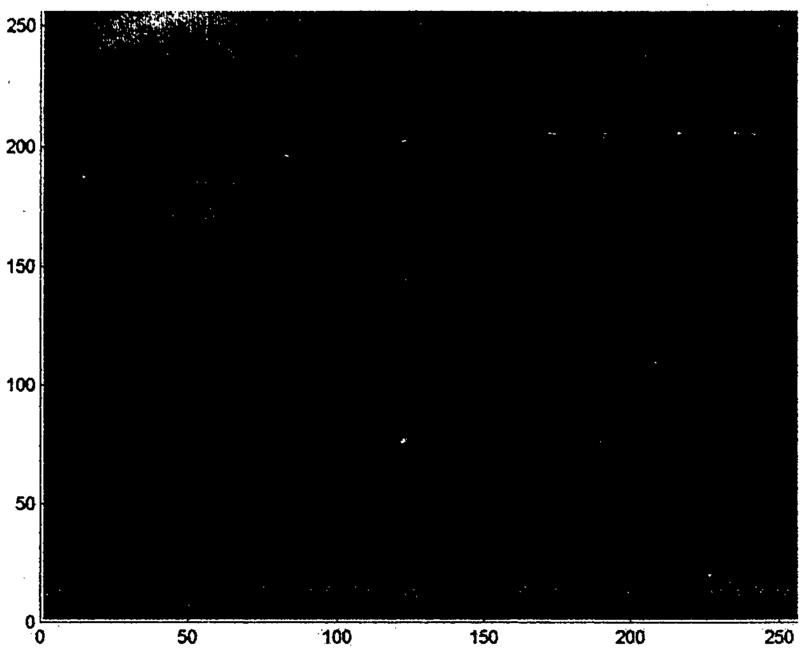


Figure 12. Missile scene in the blue 1 band. The image in the blue band displays more texture than in the red band due to the abundance of black-body sources around the missile such as sand, which absorbs and re-radiates energy from the Sun at this particular wavelength.

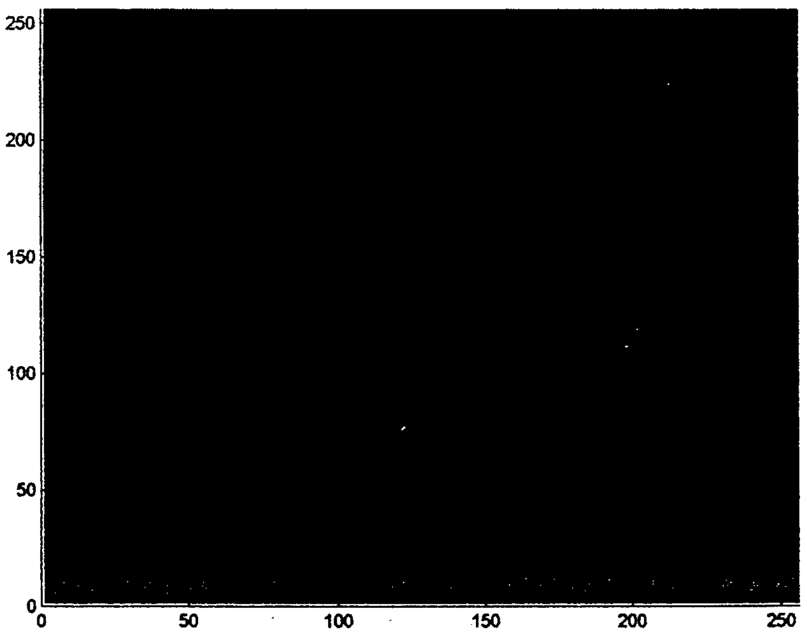


Figure 13. Missile scene in the blue 2 band.

As expected, images in the red bands were mostly featureless (except for the missile) due to the lack of any other combustion source on the test range. Images in the blue bands displayed more texture because black body sources such as clouds and hot sand were present in the scene. Figure 14 shows four consecutive intensity plots from one rotation of the color-wheel (i.e. consecutive red1, blue1, red2 and blue2 frames). These plots show that there is little difference between images from the red1 and red2 quadrants and blue1 and blue2 quadrants. Thus the four-quadrant nature of the color-wheel was ignored in favor of treating the two red quadrants as one red band and the two blue quadrants as one blue band, thus reducing the complexity of the problem and also reducing the number of features needed to characterize the data.

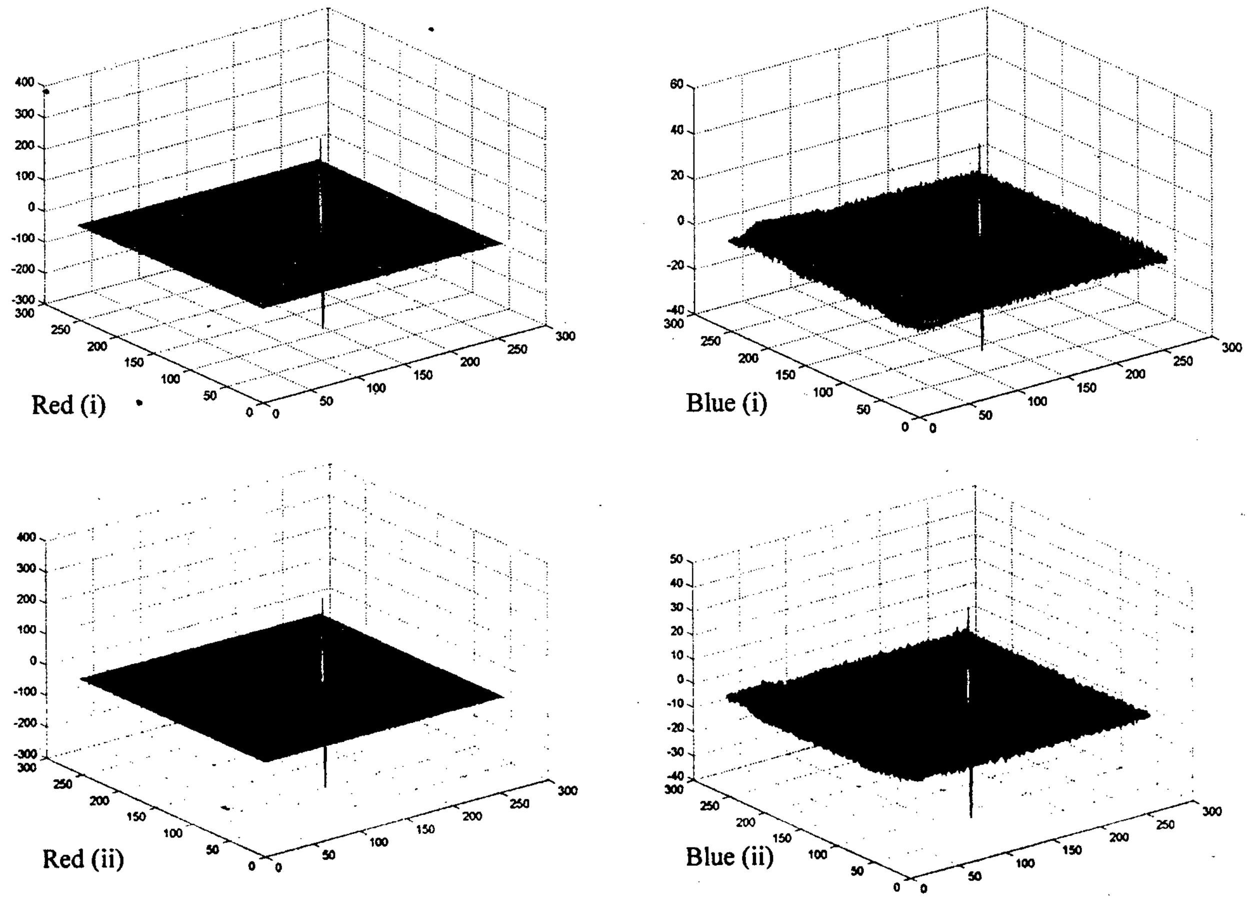


Figure 14. Intensity plots of a missile scene in each color band.

As discussed in Section 2.9, background features were obtained from a variety of clutter environments over-flown by the SIRDS sensor. Figures 15 and 16 show scenes of Eglin Beach in the red and blue bands, respectively.

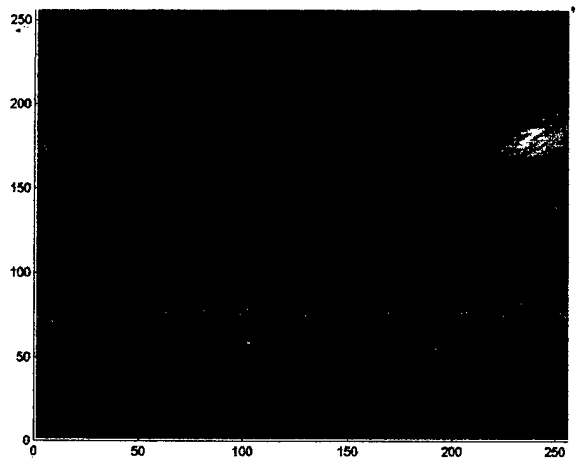


Figure 15. Image of Eglin beach in the red band. The portion of this scene inside the rectangular window provided training data to the RBF network. The window was chosen to highlight what appeared to be a strip of beach towards the lower right-hand corner of the scene. The window deliberately excluded the strong feature towards the upper right-hand corner of the scene. This feature did not correlate with any known object on the ground and appeared to be due to glare from the sensor optics.

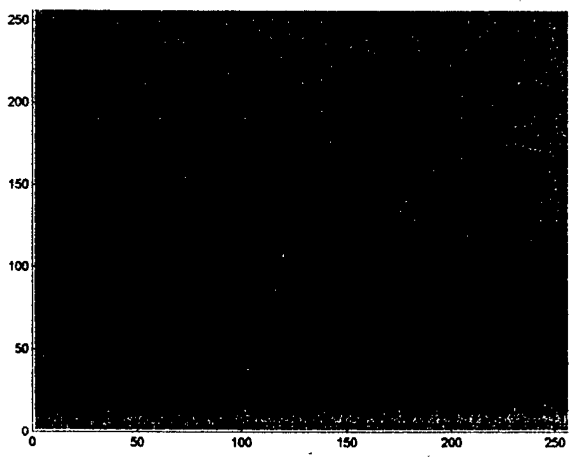


Figure 16. Image of Eglin beach in the blue band. The anomalous feature discussed in Figure 15 is less evident in this image, and the image displays more texture than in the red band.

Figures 17 and 18 show scenes of the South Carolina countryside in the red and blue bands, respectively.

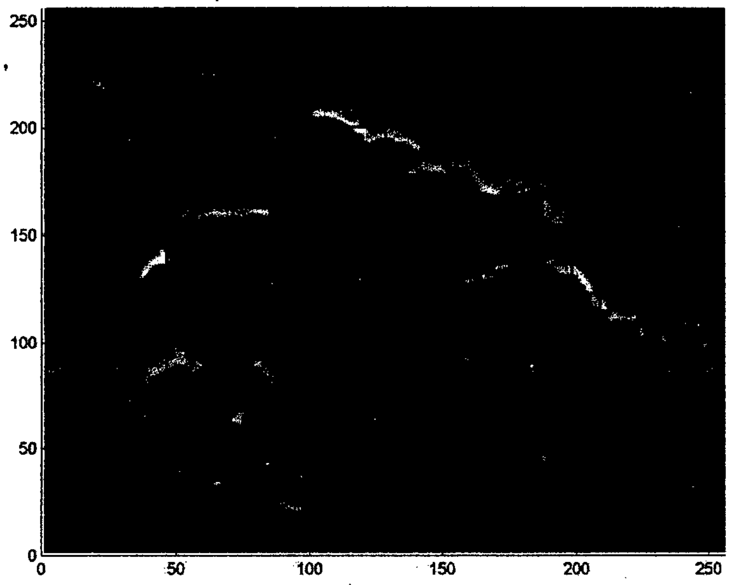


Figure 17. Image of South Carolina countryside in the red band. The pixels inside the rectangular window were chosen for the training data set.

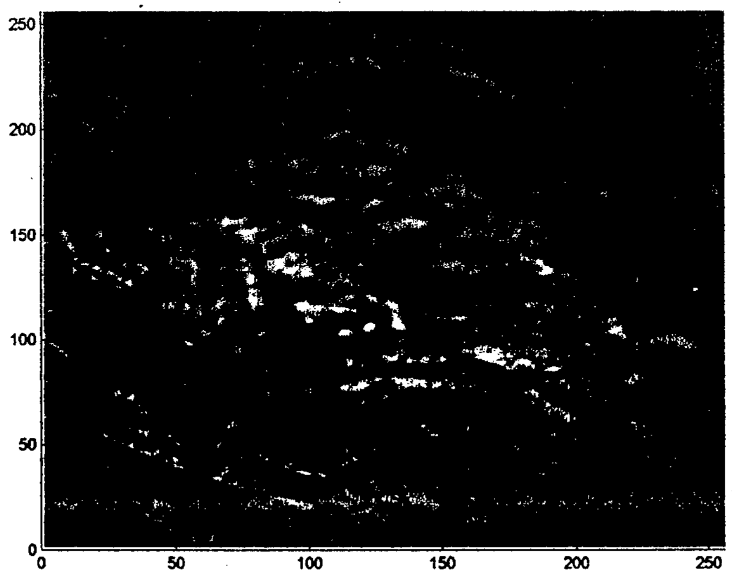


Figure 18. Image of South Carolina countryside in the blue band.

Figures 19 and 20 show scenes of the North Carolina countryside in the red and blue bands, respectively.

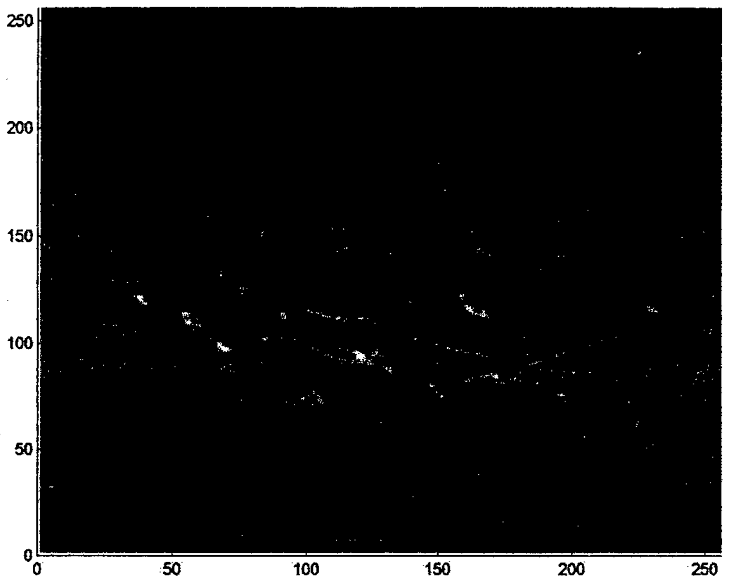


Figure 19. Image of North Carolina countryside in the red band.

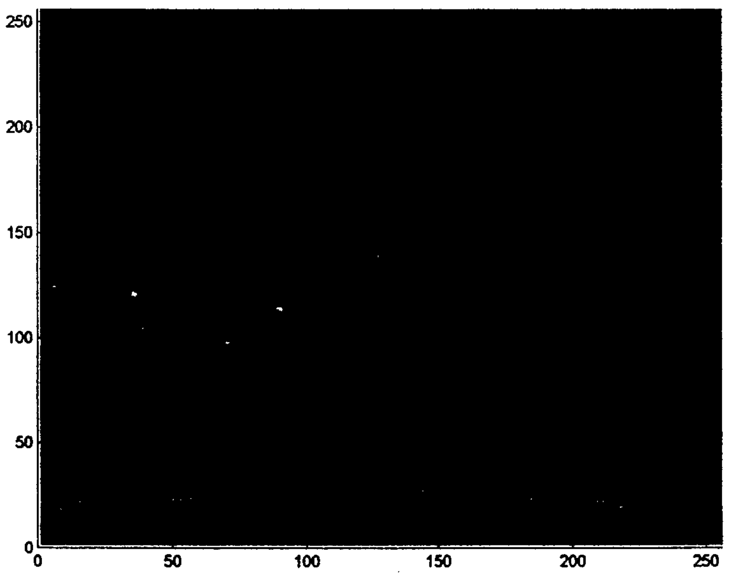


Figure 20. Image of North Carolina countryside in the blue band.

Figures 21 and 22 show the Virginia countryside in the red and blue bands, respectively.

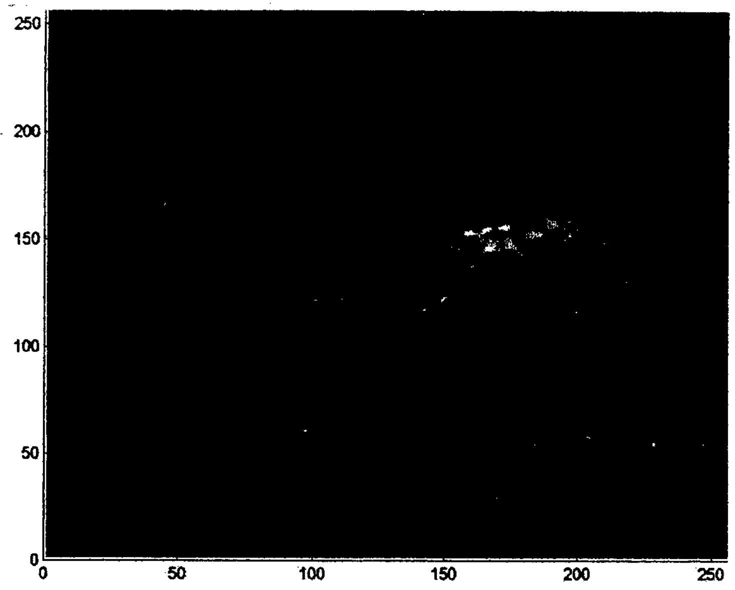


Figure 21. Image of Virginia countryside in the red band.

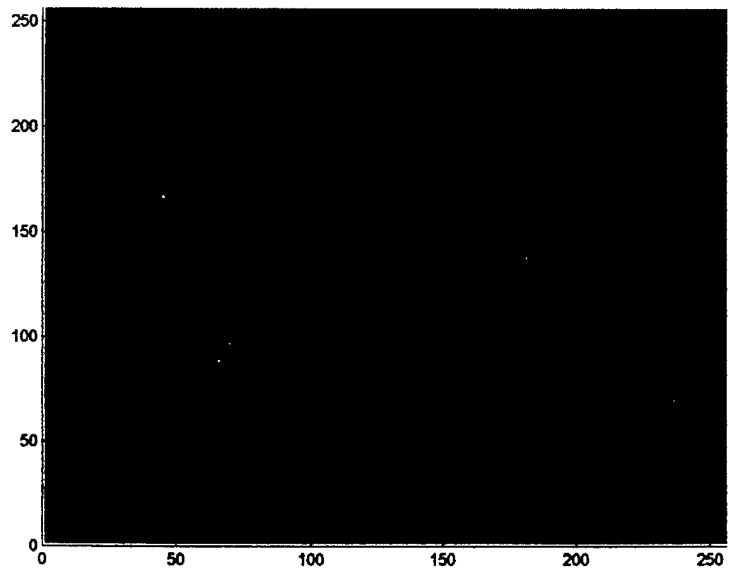


Figure 22. Image of Virginia countryside in the blue band.

4.2.2 Feature Selection

The eighteen raw pixel intensities from red/blue image chip pairs were the first features to be tested. However, the computer had insufficient memory to build the complete regression tree for this eighteen-dimensional feature space. Thus Z scores and Holmes parameters were investigated as a means of combining the inputs into more meaningful statistics and to reduce the dimensionality of the input space.

The Z score was taken from work already performed by Baxley *et al.* (2000) and designated as Z_B. The concept of this Z score, described in Section 3.5.2, is that the center pixel is usually much brighter than the average of the pixels immediately surrounding it in a target image chip. This finding is especially valid for sub-pixel missile detection, since if light from the missile is incident on a very small (sub-pixel) portion of a detector element, its energy is largely concentrated in that one image pixel. Conversely, background clutter is usually uniformly high or low in intensity over small regions. Thus the ratio of a center pixel intensity to the surrounding pixels average intensity should be high for target chips and low for background chips.

The second Z-score-like statistic was the *Holmes* parameter, which is normally associated with double-gated filtering techniques. The Holmes parameter was designated Z_H. The concept of the Holmes parameter, described in Section 3.5.3, is that two groups of pixels extracted from an image will differ significantly in mean intensity if one of the groups is mainly an aggregate of target pixels. Thus the Holmes parameter should also be high for target chips and low for background chips. The Holmes parameter has been shown effective for detecting *extended* sources such as in Synthetic Aperture Radar (SAR) imagery (Morin, 2000), but is adaptable to *point* sources (whereas the opposite is not usually true).

4.2.3 Principal Component Analysis of Features

A row of the feature vector was arranged as [Z_{B_RED}, Z_{B_BLUE}, Z_{H_RED}, Z_{H_BLUE}]. The covariance matrix of the 12,000-case feature set was

$$\text{Covariance}(X_{tr}) = \begin{matrix} 3.3996 & 0.1510 & 0.1960 & 0.0949 \\ 0.1510 & 2.4125 & 0.2004 & 0.1149 \\ 0.1960 & 0.2004 & 4.0751 & 3.2109 \\ 0.0949 & 0.1149 & 3.2109 & 3.7354\,. \end{matrix} \tag{4.1}$$

The eigenvectors and eigenvalues of the covariance matrix were

$$\text{Eigenvectors} = \begin{matrix} -0.1403 & -0.9882 & -0.0228 & 0.0573 \\ 0.9889 & -0.1369 & -0.0298 & 0.0491 \\ -0.0114 & 0.0277 & 0.6890 & 0.7242 \\ -0.0472 & 0.0631 & -0.7238 & 0.6855\,, \end{matrix} \tag{4.2}$$

$$\text{Eigenvalues} = \begin{matrix} 2.3833 & 0 & 0 & 0 \\ 0 & 3.4090 & 0 & 0 \\ 0 & 0 & 0.6868 & 0 \\ 0 & 0 & 0 & 7.1436\,. \end{matrix} \tag{4.3}$$

As shown in Equations 4.2 and 4.3, the eigenvectors corresponding to the two largest eigenvalues were $[0.0573, 0.0491, 0.7242, 0.6855]^T$ and $[-0.9882, -0.1369, 0.0277, 0.0631]^T$, which were the principal components of the Holmes parameter and Z score in the blue band, respectively. The projection x of any vector b onto the column space of these eigenvectors was computed using

$$x = (A^TA)^{-1}A^Tb\,, \tag{4.4}$$

where $A = \begin{matrix} 0.0573 & 0.9882 \\ 0.0491 & -0.1369 \\ 0.7242 & 0.0277 \\ 0.6855 & 0.0631\,, \end{matrix}$

and b = transpose(X_{tr}).

The goal was to project the 4-dimensional feature space onto a 2-dimensional space for better visualization. The new feature space is shown in Figure 23.

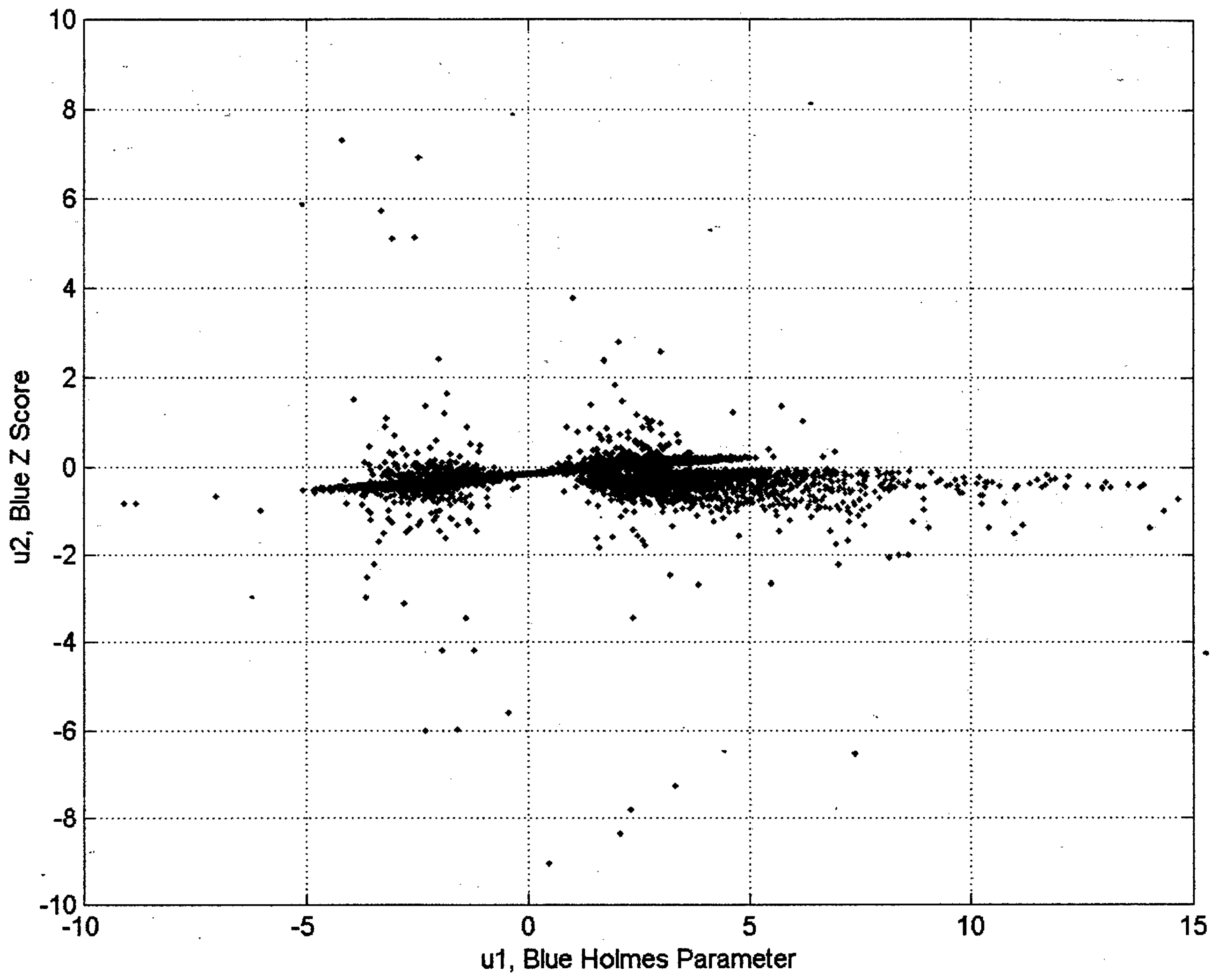


Figure 23. Projection of the input space into two dimensions by principal component analysis. The axes **u1** and **u2** correspond to the dimensions of the Holmes parameter and Z score for the blue bands respectively, as the eigenvectors of these two features corresponded to the two largest eigenvalues of the covariance matrix. The red data points represent components of the original missile features projected onto the new feature space and the black data points represent components of the background features projected onto the new space. There is a slight overlap between the two sets of data, but overall they appear to occupy different regions of the feature space. This result means that the features chosen have distinguishing characteristics that should provide a good discrimination capability to the RBF network.

Figure 24 shows the two-dimensional space of the red and blue band Z scores, and Figure 25 shows the two-dimensional space of the red and blue band Holmes parameters.

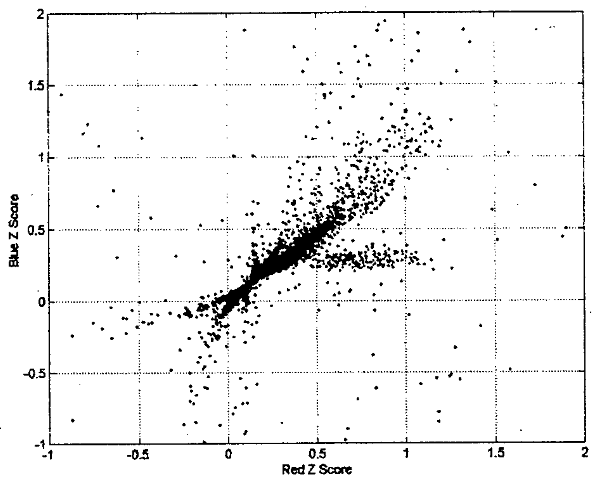


Figure 24. Two-dimensional space of the Z scores. Red data points represent Z scores of missiles and black data points represent the Z scores of background training cases.

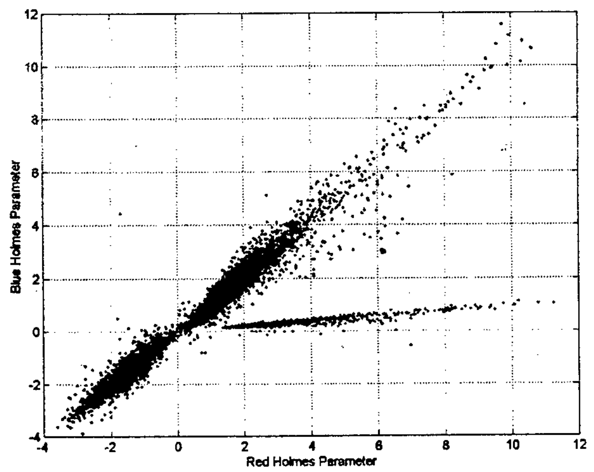


Figure 25. Two-dimensional space of the Holmes parameters. Red data points represent Holmes parameters of missiles and black data points represent Holmes parameters of the background training cases. Holmes parameters corresponding to missiles are generally large-valued and positive, whereas those corresponding to background are relatively small-valued and positive and negative. The distinct 'spike' of missile data points away from the generally linear distribution is more apparent here than in the Z score scatter plot. These spikes are the only immediately apparent difference between missile signatures in the boost and sustain phases of the missile firing.

Figure 26 shows the red and blue missile Z scores as a function of time, and Figure 27 shows the red and blue Holmes parameters as a function of time.

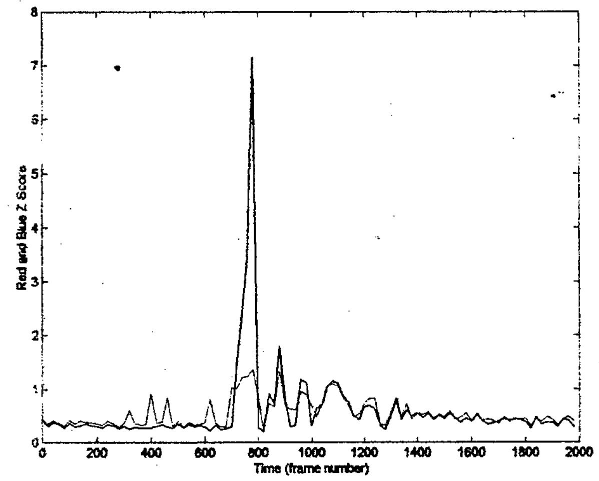


Figure 26. Red and blue missile Z scores versus time. Red Z scores are slightly higher than blue Z scores at the start of missile firing (the boost stage), which is when the exhaust plume is very hot. The blue Z score temporarily dominates after the motor cuts-out. However, the two features are mostly identical for the remainder of the training set, which indicates that the missile becomes less distinguishable from the background after the booster cuts-out.

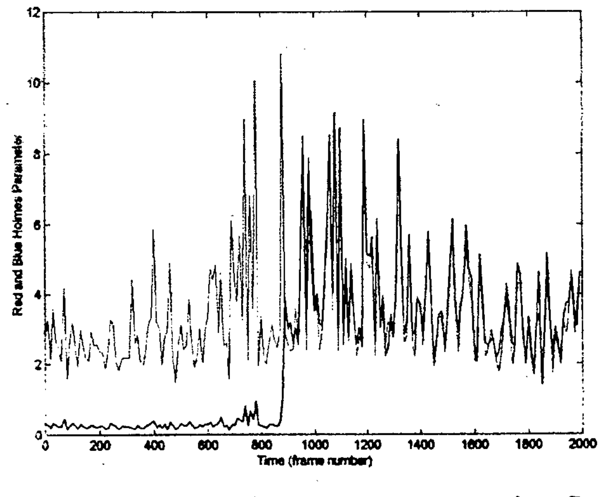


Figure 27. Red and blue Holmes parameters versus time. Red Holmes parameters are much higher than blue Holmes parameters during missile boost and this produces the distinct 'spike' of missile data points in Figure 25. The greater separation between red and blue Holmes parameters than red and blue Z scores during missile boost indicates that the Holmes parameter should be the more powerful feature in missile detection.

4.3 The RBF Network

The RBF neural network was produced by configuring the regression tree algorithm to grow to a minimum of 100 input cases in a node and by scaling the size of the hyper-rectangles associated with each node by a factor of two. The training vector contained 12,000 cases and the target vector contained 12,000 elements. Each element of the target vector corresponded to one case in the training vector and had a value of 1.0 if the case came from the missile data set and 0.0 if it came from a background data set. The resulting neural network consisted of 103 RBF centers distributed in the four-dimensional feature space with individual radii and weights associated with each center. The RBF centers, radii, and weights are listed in Appendix A.

Information returned from the regression tree method indicated that the red band Z score was the first feature to be split and was also the most often split: 107 times. The blue band Z score was the second feature to be split, but it was only split once. The red and blue band Holmes parameters were the third and fourth features to be split: five and two times, respectively. Orr (1999) suggests that the feature that is split first and/or most often split is the most relevant (and thus useful) feature for discrimination. However, Figure 26 shows that red Z scores are not significantly different from blue Z scores (even during missile boost). Therefore, the regression tree method needed to partition the red Z score data set much more than the other features' data sets in order to cluster sufficient red Z scores to discriminate between missiles and background. On the other hand, Figure 27 shows that red Holmes parameters are clearly higher than blue Holmes parameters during missile boost. Holmes parameters (alone) displayed sufficient discrimination between missiles and background (as found in the principal component analysis discussed in Section 4.2.3) and thus did not require much partitioning by the regression tree method. Therefore, the Holmes parameter is (logically) the more useful feature for discrimination (contrary to Orr's conclusions about the regression tree method's automatic relevance determination).

4.4 Test Results

Initially, the 3 by 3 window transform was moved over an entire test frame to apply the RBF network to every pixel in the scene. However, this method was slow due to the large number of operations needed for a 256 by 256 image. Figure 28 shows intensity plots for two frames from the 'unseen' missile data set used to test the RBF network.

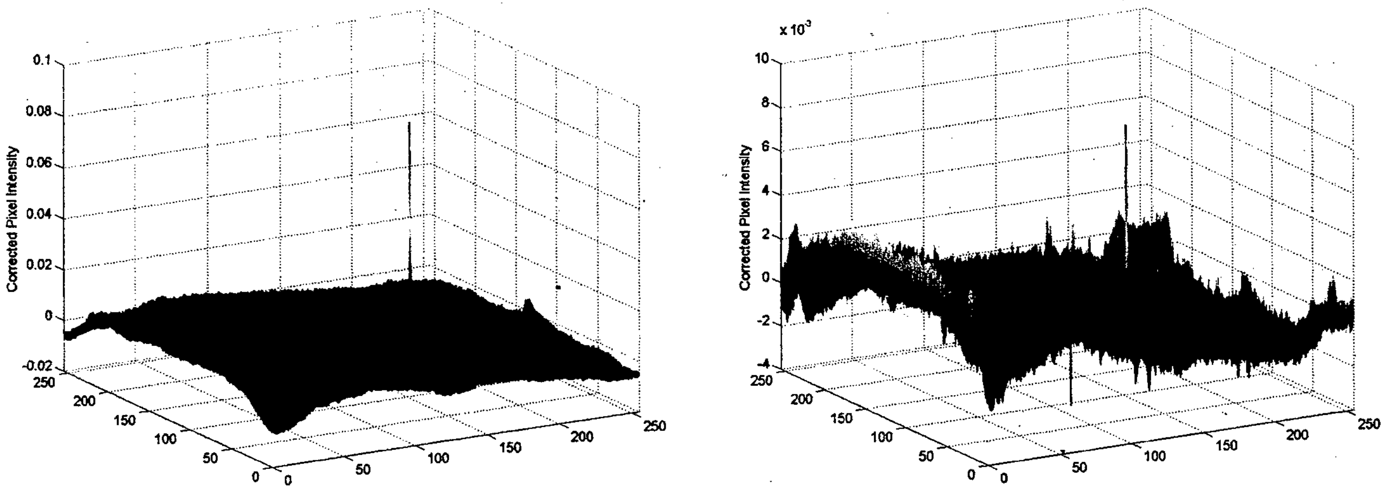


Figure 28. Missile test frames. The intensity plot on the left is from the red band, and the plot on the right is the frame that immediately followed in the blue band. The missile is the sharp peak located at coordinate {51, 125}.

Figures 29 and 30 show the probability density and target location maps, respectively, obtained when the 3 by 3 window was slid over the entire image in each band and applied to the RBF network.

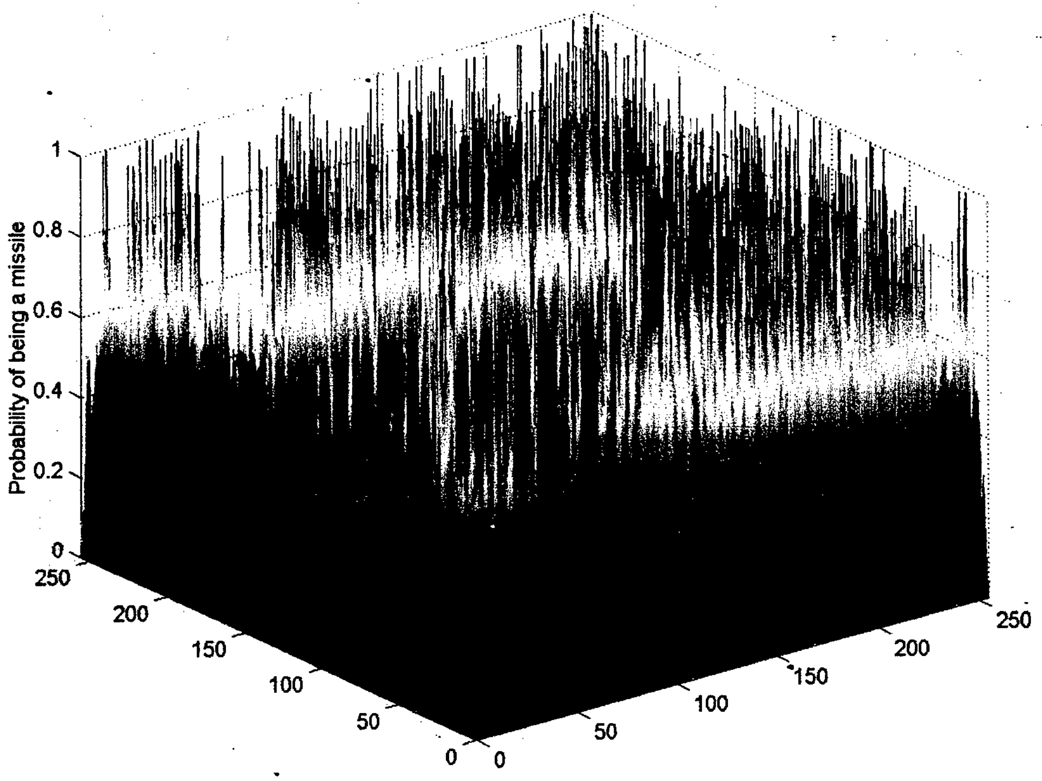


Figure 29. Probability density mapping from analyzing an entire test frame. The RBF network determined that many parts of the scene contained missiles with a high probability, even though the test scene contained one actual missile. The red spikes in the map indicate pixels that had an 80% or greater chance of containing a missile. The large number of false alarms is probably due to irregularities in the noise floor of the data. Pixels in noise can appear to be missiles if the mean of the noise around them is low enough to enhance the features calculated for these pixels.

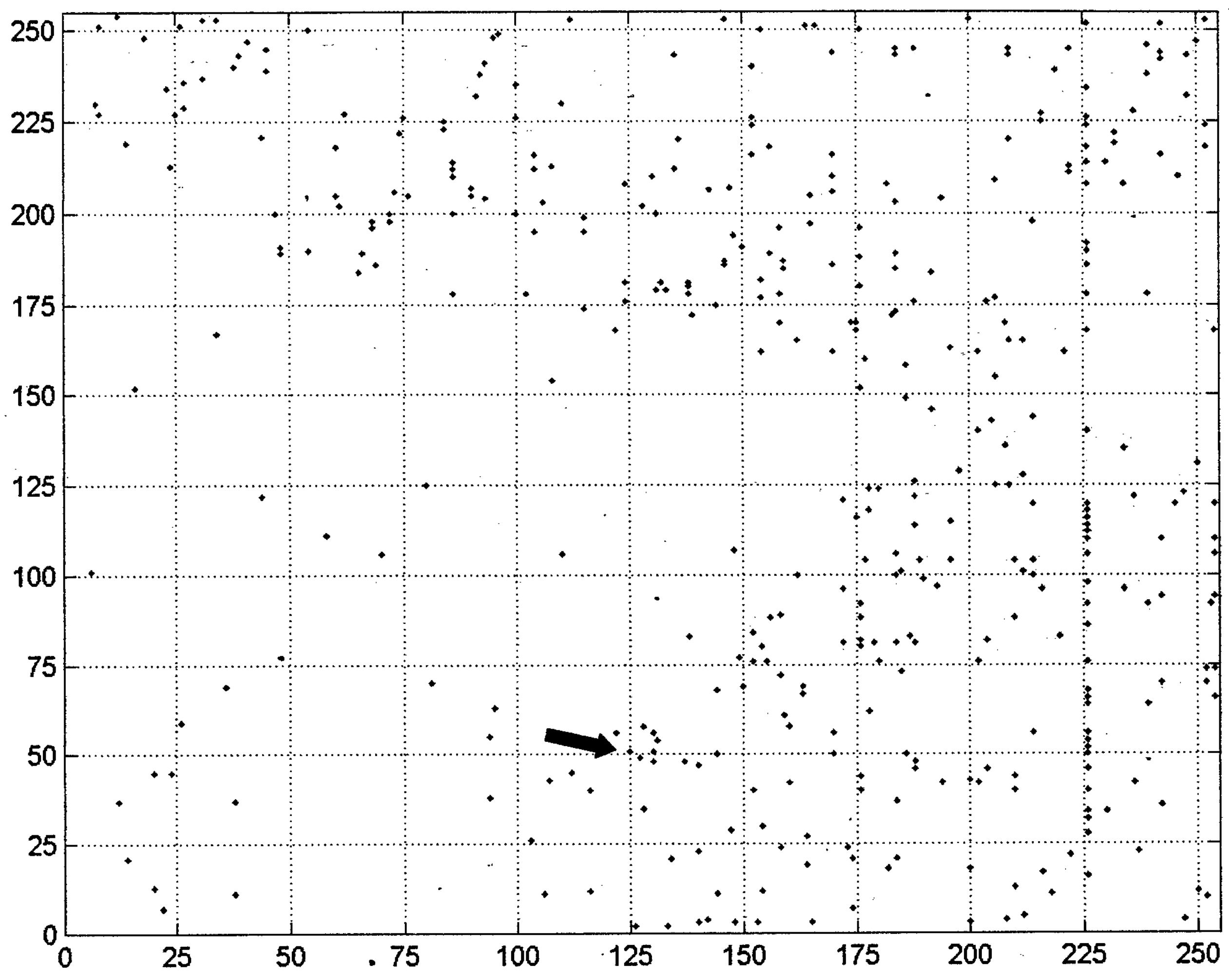


Figure 30. Target location map from analyzing entire test frames. The red dots indicate pixels that were given a 95% or greater probability of containing a missile by the RBF network. There were 401 of these potential targets in this scene alone, of which only one, at coordinate {51, 125} and marked by the black arrow, was the true missile.

A technique for eliminating spurious detections compares results from successive frames with a logical AND operation. Therefore, the next two red/blue frames in the data set were analyzed and their (>95%) target locations 'ANDed' with the previous target location map to produce the comparison mapping shown in Figure 31.

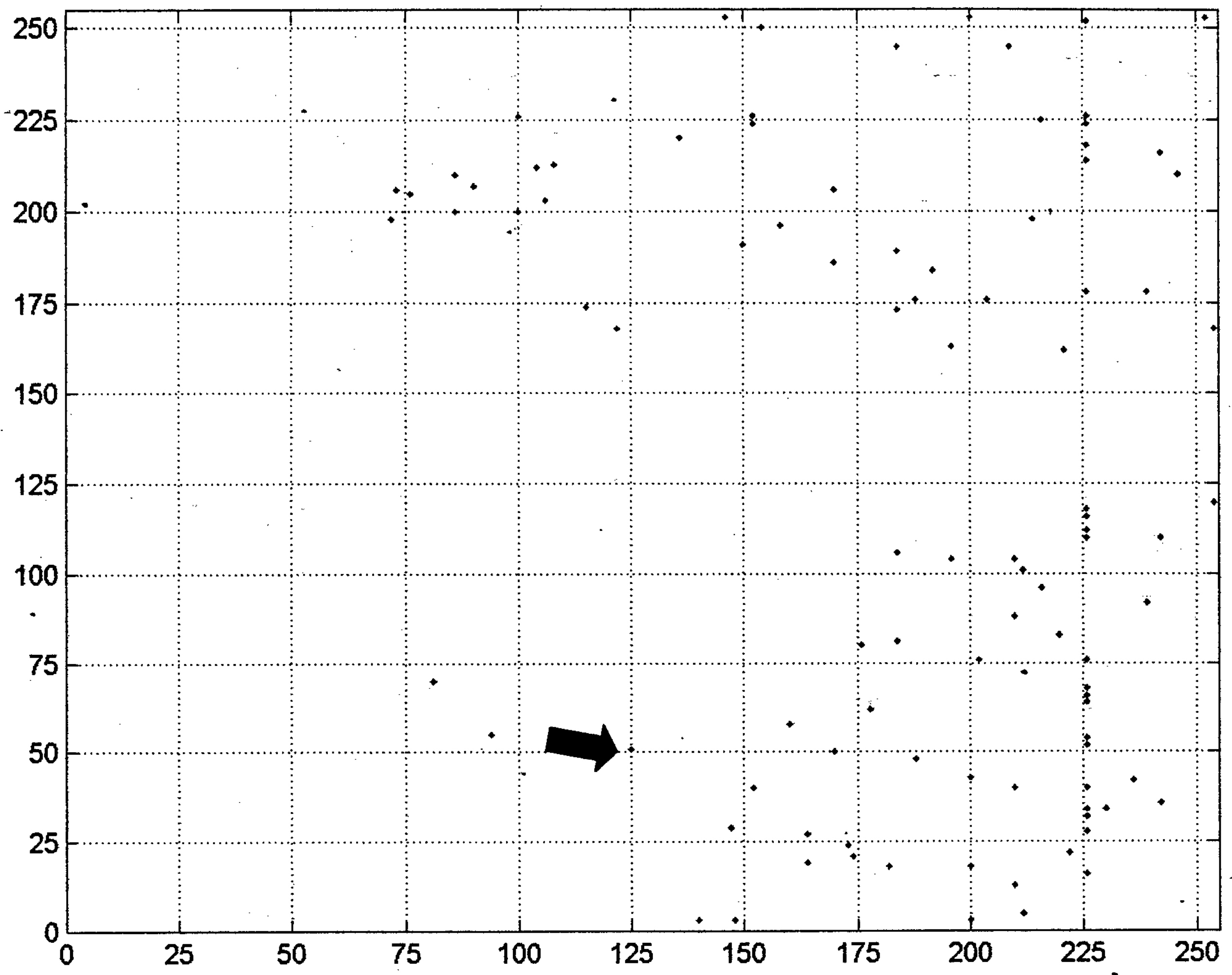


Figure 31. Result after two-frame registration and comparison. Target location maps for pixels with greater than 95% probability of containing missiles were compared using a logical AND operation. Only targets that appeared in the same locations in both maps were retained. The number of potential targets was reduced to 99, much less than the original 401 but still including too many false alarms. The actual missile was one of the retained targets as indicated by the black arrow.

BLANK PAGE

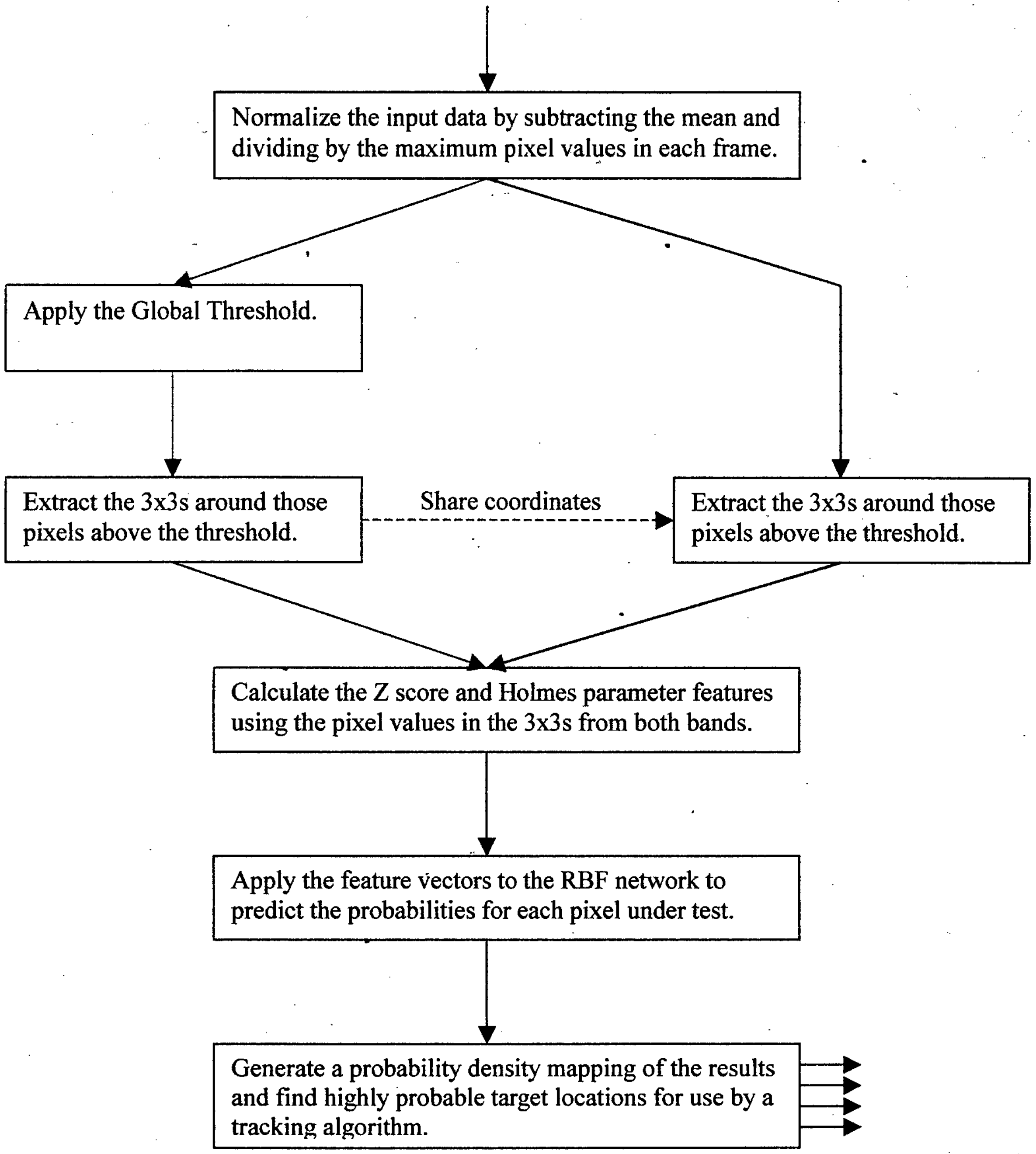

Normalize the input data by subtracting the mean and dividing by the maximum pixel values in each frame.
Apply the Global Threshold.
Extract the 3x3s around those pixels above the threshold.
Share coordinates
Extract the 3x3s around those pixels above the threshold.
Calculate the Z score and Holmes parameter features using the pixel values in the 3x3s from both bands.
Apply the feature vectors to the RBF network to predict the probabilities for each pixel under test.
Generate a probability density mapping of the results and find highly probable target locations for use by a tracking algorithm.

Figure 32. Near real-time missile detection algorithm block diagram.

The threshold level must be adaptable to particular situations that depend on the strength of noise in the data. A missile in its initial boost phase is always brighter than the *average* intensity of the image, but the missile signature may fall to near the background because of the plume becoming cooler once the boost phase is complete. Therefore, the threshold level must be positive and higher than the average intensity in an image, but not too high above the average intensity as to completely ignore potential targets that may be near the noise floor. Figure 33 shows that the noise floor rarely exceeded 0.01 in a profile of the red band test image.

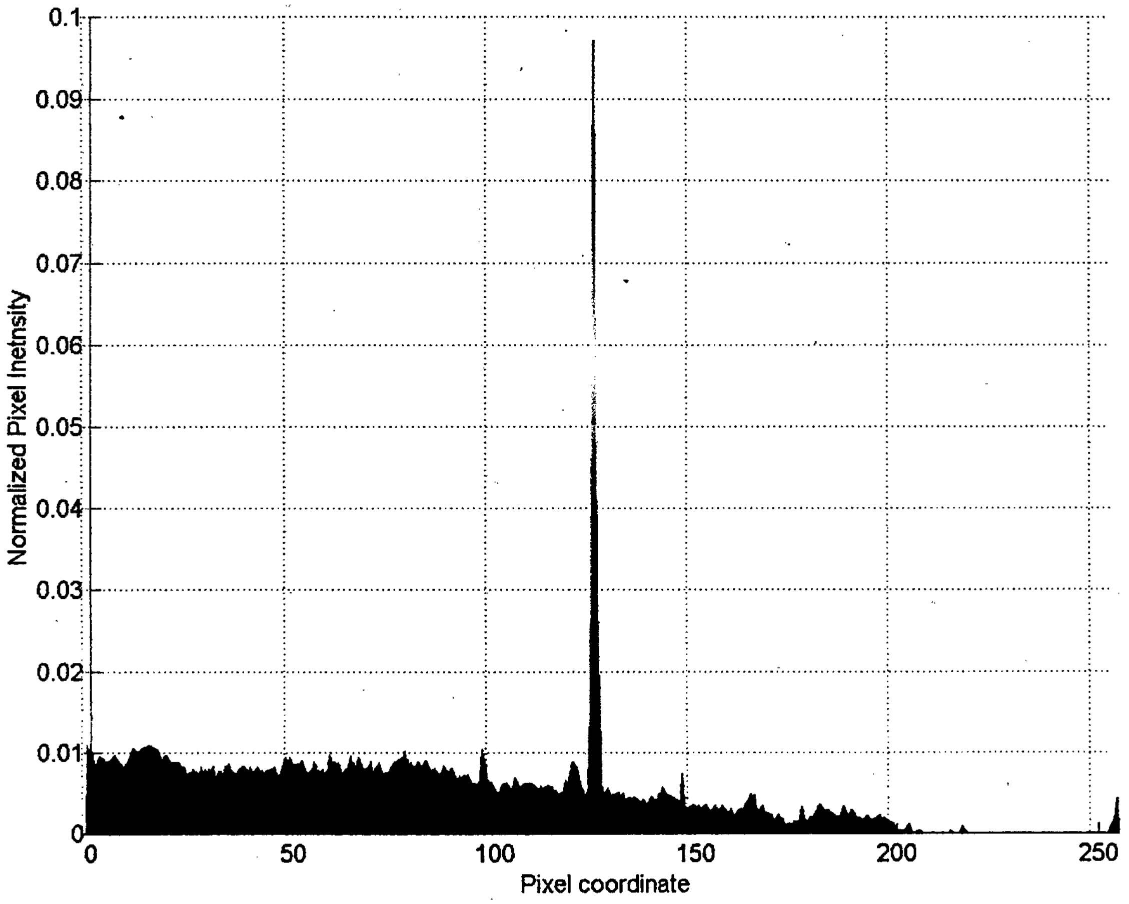


Figure 33. Red band test image profile. In profile, the normalized intensity of the missile-bearing pixel is clearly much greater than the other pixels, while the normalized intensities of the background pixels rarely exceed 0.01.

Therefore, a global threshold of 0.01 (normalized intensity) was applied to the red band test image to remove from consideration pixels that were clearly too low in normalized intensity to be potential missiles. Figure 34 shows the image profile after this thresholding.

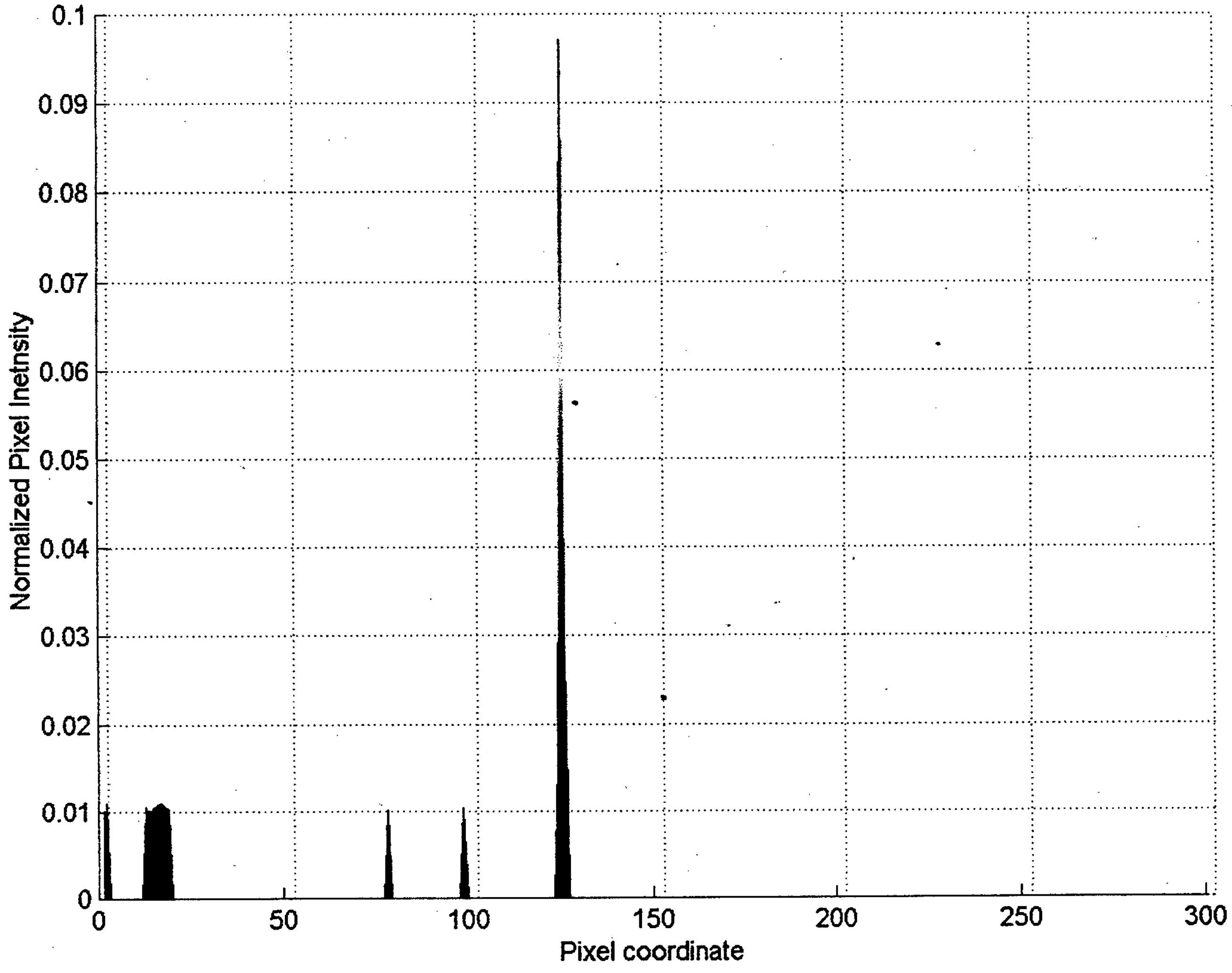


Figure 34. Thresholded red band test image profile. The smaller peaks to the left of the actual missile indicate those pixels that had normalized intensities that exceeded 0.01. Only these pixels and the missile-bearing pixel were retained for further processing to concentrate detection resources and increase the speed of the detection process.

After thresholding, only 26 pixels from the test image remained in consideration. As expected, the missile-bearing pixel probability was near unity, while the other pixel probabilities were 0.4 or lower. The resulting probability density map is shown in Figure 35.

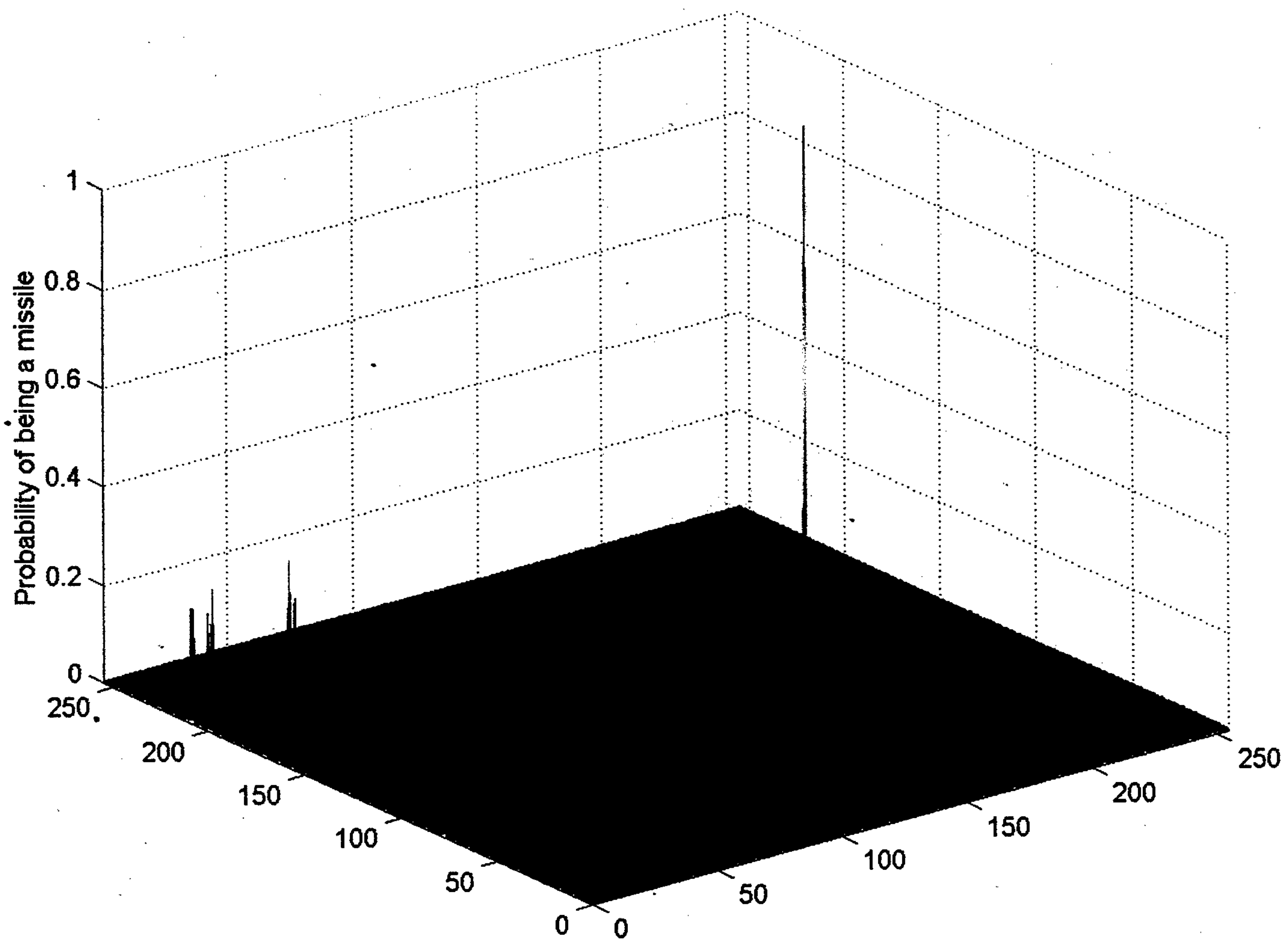


Figure 35. Result after global thresholding at 0.01 normalized intensity. The only pixel with a probability near unity exactly corresponded to the missile-bearing pixel in the test image, and the RBF network operated in near real-time as a result of having fewer potential targets to analyze.

4.6 Receiver Operating Characteristic (ROC) Curves

The performance of the RBF network was characterized using Receiver-Operating-Characteristic (ROC) curves, which plot the probability of correct detection versus the number of false alarms for classification thresholds from 0.05 to 1.0. These parameters were chosen since the RBF network had been trained to respond to missile-bearing pixels with the value 1.0, and to respond to background pixels with the value 0.0, where pixels with probabilities above the threshold were classified as missiles. Intuition suggests that probabilities above 0.8 (i.e., pixels with greater than an 80% chance of being a missile) were best for detecting missiles. However, the optimum threshold is situation specific, since missile IR signatures change with the engagement environment and the probability of the target pixel matching the training examples may change accordingly.

The probability of correct detection can only be one or zero because there was only ever one missile in a scene. However, the number of false alarms associated with a correct detection can vary greatly depending on the classification threshold used to declare the missile. The optimum classification threshold occurs when the ROC curve simultaneously achieves zero false alarms and unity probability of correct detection. Figures 36 and 37 show ROC curves at different stages of the missile firing. Figure 36 shows the ROC curve for an image at the start of the missile firing, where the exhaust plume is very hot and the missile signature is easily distinguishable from the background. Figure 37 shows the ROC curve for an image after the missile motor has cut out. Although the two curves look identical, the positions and values of classification threshold with respect to false alarms is very different.

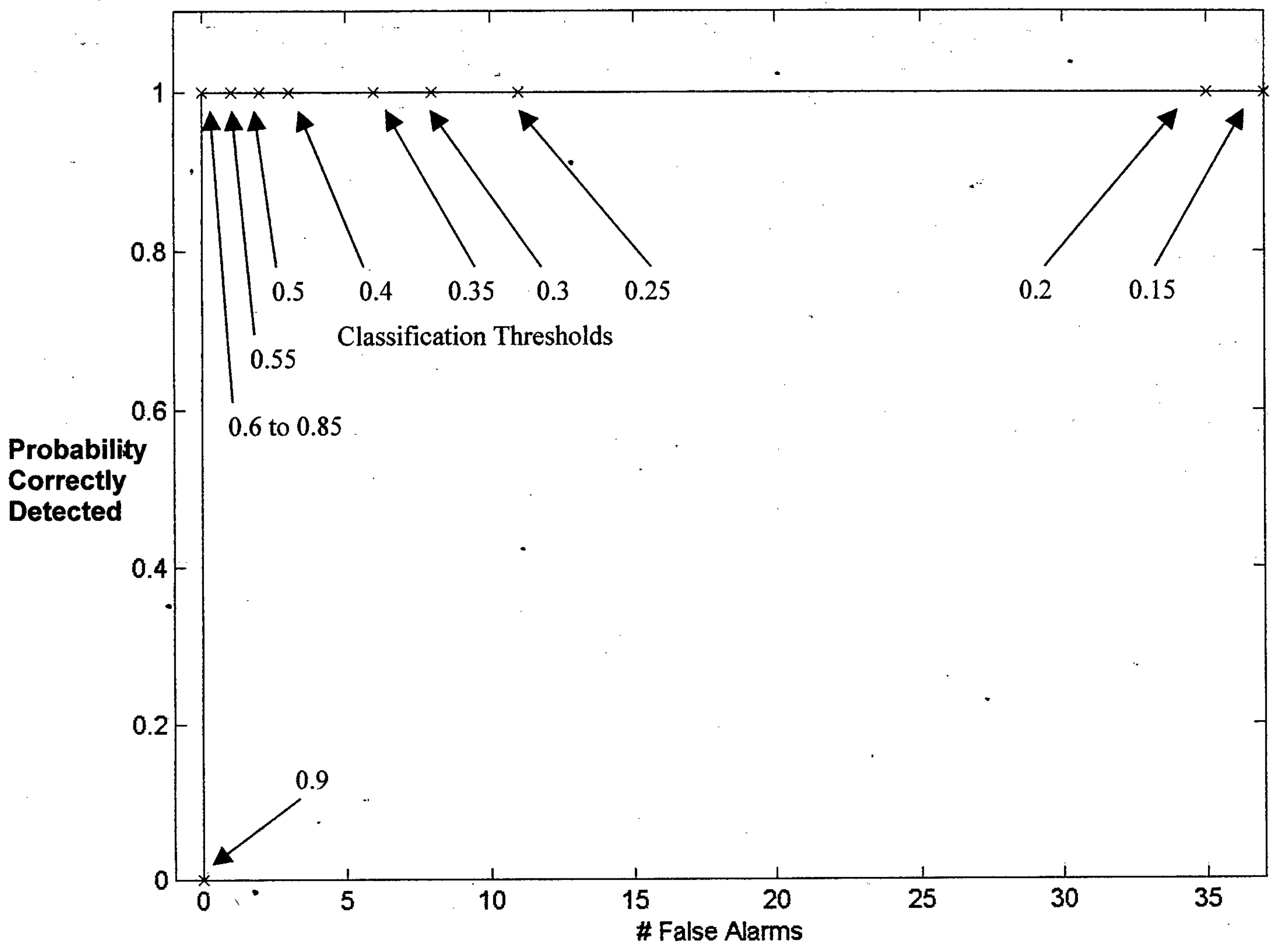


Figure 36. Receiver Operating Characteristic (ROC) curve during missile boost phase. The missile is correctly detected for classification thresholds up to 0.85, indicating that the RBF neural network predicted the presence of the missile with a high probability in this phase of the missile firing. The penalty for using lower classification thresholds than 0.6 is an increase in the number of false alarms. The penalty for using a classification threshold higher than 0.85 is the non-detection of the missile. Therefore, there is a trade-off between maintaining a 100% probability of detection and the number of false alarms that can be tolerated.

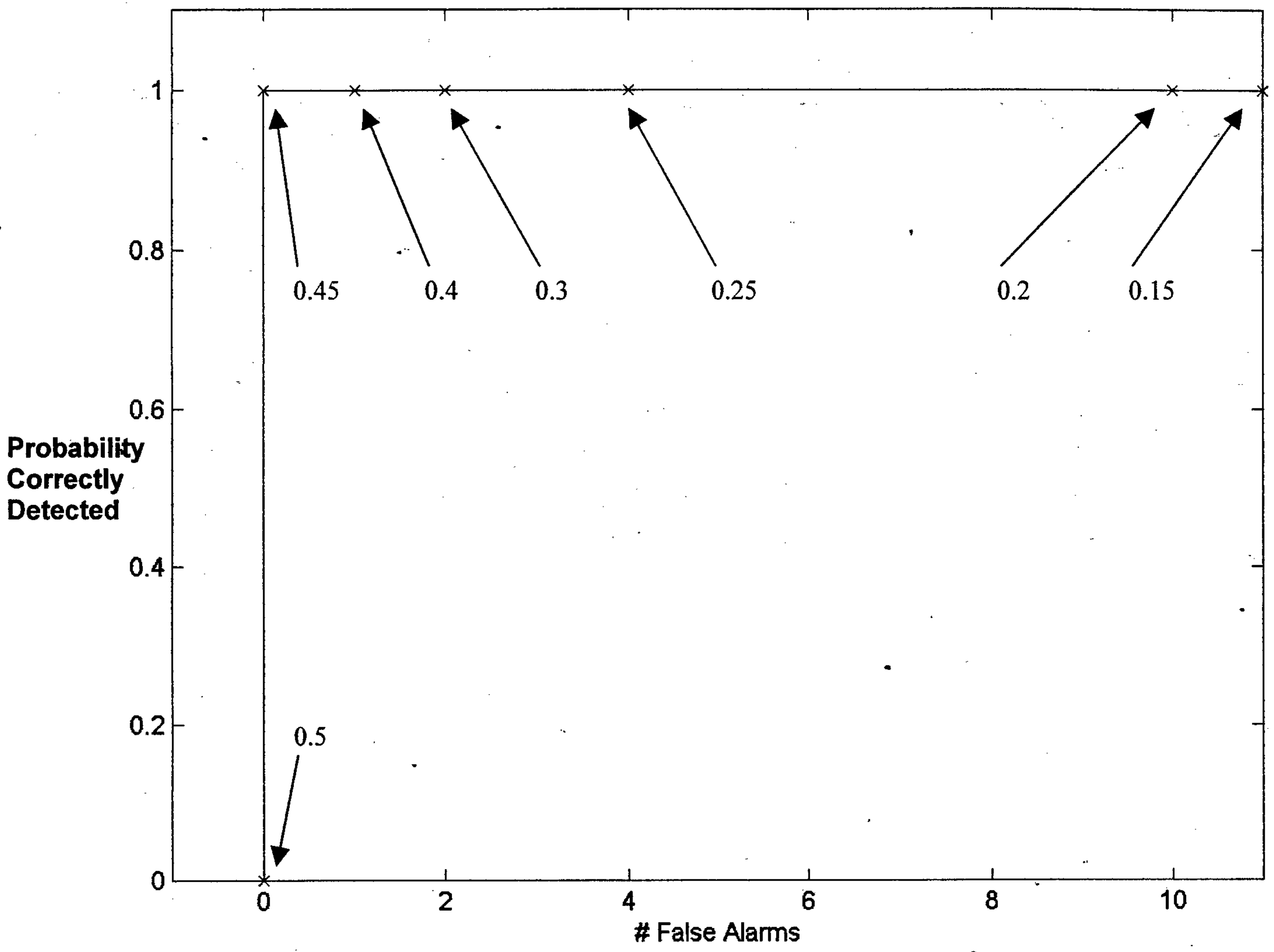


Figure 37. ROC curve after the missile motor has cut out. The classification threshold at which the missile drops out of the picture is much lower than earlier in the missile firing because the missile signature has diminished so greatly that the RBF neural network no longer predicts the presence of the missile with high probability. A classification threshold higher than 0.45 at this stage of the engagement would result in non-detection of the missile.

The ROC curves showed that the optimum threshold for detecting the actual missile varied with time. Setting the threshold too low invited detecting many false alarms, whereas setting it too high resulted in omission of the actual missile. In general, missiles could be reliably detected using probability thresholds of 0.4 or higher at a small cost in false alarms (since the majority of background pixels had probabilities lower than 0.4). However, the threshold must be adaptable to compensate for variations in the RBF network probability estimations.

An adaptive method could be based on the fly-out characteristics of missiles. The missile plume is usually very hot during launch as the motor *boosts* the missile off the rail and imparts a rapid acceleration. This boost phase may last two to three seconds, after which the motor *coasts* the missile for the remainder of the engagement along its intercept trajectory (sometimes with a final boost at the end of the intercept to give the missile extra impetus in the end-game). It is during the coast phase that the missile signature may fall to values at or below the background clutter where the neural network may not recognize the missile. More tests are needed to obtain data for training an RBF network to recognize missiles in this phase.

4.7 Summary of Results

RBF network training took longer to complete than expected due to the complicated configuration rules for the regression tree method by Orr (1999). It is unknown whether the network parameters obtained were the best solution to the problem (available computer resources limited the depth to which the regression tree could grow). The RBFs used here were developed by configuring the regression tree to split the input data until there were a minimum of 100 input cases in a single node and by scaling the hyper-rectangles associated with each node by the factor of two. The optimum RBF network was determined by selecting RBFs in the tree that most decreased the prediction error using leave-one-out cross-validation. The resulting RBF network consisted of 103 RBF centers representing a 12,000-case training set that consisted of 2,000 missile cases and 10,000 background cases from infrared images of Eglin beach and South Carolina, North Carolina, and Virginia countryside.

The RBF network was slow to predict responses for entire image frames due to the large amount of processing required. The results also contained many false alarms due to noise in the data. A global thresholding stage was applied (prior to the RBF network) to red band inputs to suppress pixels whose normalized intensities were below that expected of actual missiles, and the RBF network then performed faster and with fewer false alarms. ROC curves showed that the optimum probability threshold for detecting the actual missile varied with time.

Overall, the RBF network, once designed and implemented in a near real-time multi-stage algorithm, correctly recognized missiles in two-color infrared imagery while producing a low number of false alarms.

5. Conclusions and Recommendations

5.1 Restatement of Research Objectives

The objectives of this research were to:

1. Propose a method for detecting sub-pixel missile signatures in two-color infrared images using Gaussian Radial Basis Function (RBF) neural networks.
2. Evaluate the performance of this detection technique by training and testing the neural network with data containing *real* missile and background signatures.
3. Determine the near real-time effectiveness of the neural network in a *real-world* missile warning system by applying previously unseen images to the network and obtaining Receiver Operator Characteristic (ROC) curves.

5.2 Conclusions

5.2.1 Combining Regression Trees and Radial Basis Function Neural Networks

The use of regression trees for generating radial basis function neural networks is innovative. However, the tree building process is also memory intensive and limits the amount of training data. The resulting RBF network is also dependent on the initial configuration of the regression technique; the 'minimum members per node' and 'hyper-rectangle RBF scaling factors' greatly influence the final design. An optimal combination of these parameters is not intuitive, and only trial-and-error finds a workable solution for particular situations. Nevertheless, the techniques of Orr (1999) are consistent with the development of radial basis function neural networks and address the research objectives.

5.2.2 Z Score and Holmes Parameter

The Z score and Holmes parameters were effective in differentiating pixels that represented missiles from those which contained only background; a pixel containing energy from a missile is significantly higher in intensity than its surrounding neighbors, whereas pixels from background scenes are usually uniformly high or low in intensity over a local region. The Holmes parameters were much larger in the red band than in the blue band during missile boost, which (probably) enabled the RBF network to operate as effectively as it did. The red and blue Z scores did not display as large a separation as the Holmes parameters during the same boost phase of the missile firing. Automatic relevance determination by regression trees found that the red band Z scores required the most partitioning to enable their use in the neural network. The red and blue band Holmes parameters were the last of the four features to be split by the regression tree method, indicating that these features were sufficiently distinct on their own for discriminating between missiles and background. The use of only four features (and not one feature for each of the original eighteen raw pixel intensities) averted the *curse of dimensionality*.

5.2.3 Data Normalization

Instead of compensating for variations between data sets, all data was normalized by subtracting the mean and dividing by the maximum pixel intensity in each image before neural network processing. As a result, the neural network algorithm only processes intensity ratios instead of high-value absolute intensities.

5.2.4 Global Thresholding

Global thresholding is a pre-processing stage that narrows the field of interest for the RBF network and concentrates detection resources in only those areas of the image that intuitively contain missiles. The use of normalized data also means that the level of thresholding is uniformly applicable to a variety of data sets even if they represent different environments.

5.2.5 Overall Performance

The RBF network-based missile detection algorithm performs very well in recognizing the correct target in the tested images. Intentional design of the RBF network response to mimic probability estimation allows a mapping of the feature space to a corresponding probability density space from which individual pixels are clearly likely (or unlikely) to be missiles depending on their values between zero and one. ROC curves of the results show that the probability thresholds that best eliminate false alarms vary as a function of time, and thus these thresholds need to be adaptive to compensate for variations in missile IR signatures during typical engagements. Nevertheless, this research demonstrates that RBF neural networks are effective at two-color IR missile detection. The algorithms that performed data collection, neural network training and testing, and missile detection are listed in Appendix B.

5.3 Recommendations for Future Research

The following are recommendations for further research:

1. The Matlab toolboxes by Orr (1999) enable 'forward selection' and 'ridge regression' techniques to be used as alternatives to the regression tree method considered in this thesis. These techniques should be investigated.
2. The Holmes parameter clearly differentiates between missile boost and sustain phases. Therefore, this difference could be used to tailor the training data set to optimize the RBF network to detect missiles in their boost phase (where the exhaust plume is at its hottest and thus most recognizable from the background). The Holmes parameter displayed a significantly better discrimination capability than the Z score (mostly during the missile boost phase). Therefore, the Z score could be replaced by other features that characterize the plume in this phase of the missile firing as effectively as the Holmes parameter. However, the total number of input features to the RBF network should still be kept small to avert the curse of dimensionality.
3. A neural network is only as effective as the data with which it is trained. Currently, the RBF network is trained to detect missiles that remain at the same ground location. A typical missile-warning system would not consider such a target a threat, since the missile does not approach the host platform. Therefore, missile fly-out data is required to train the RBF network to recognize the characteristics of real threats.
4. Finally, if data with missile fly-out characteristics becomes available, a tracking algorithm should be designed that uses the target information provided by the detection algorithm developed here, which generates potential target coordinates and the probabilities that they correspond to missiles.

Appendix A

A.1 **RBF Centers:** Columns 1 through 7

82.3781	82.3781	0.3386	82.4454	82.3781	-38.1868	0.2568
55.0594	55.0594	55.0594	55.0594	55.0594	55.0594	55.0594
1.9500	6.3244	-2.9597	-2.9597	1.6214	1.9500	1.9500
32.5097	32.5097	32.5097	32.5097	32.5097	32.5097	32.5097

Columns 8 through 14

82.3781	82.3781	82.3781	82.3781	82.3781	-38.1868	-38.3718
55.0594	55.0594	55.0594	55.0594	55.0594	66.2234	-10.8908
2.0714	6.7744	2.0714	2.0714	2.4155	1.9500	1.9500
32.5097	32.5097	-2.8484	37.2187	32.5097	32.5097	32.5097

Columns 15 through 21

0.0573	-0.0700	0.1150	0.0073	0.1346	0.1433	0.0489
-10.8908	-10.8908	-10.8908	-10.8908	-10.8908	-10.8908	-10.8908
1.9500	1.9500	1.9500	1.9500	1.9500	1.9500	1.9500
32.5097	32.5097	32.5097	32.5097	32.5097	32.5097	32.5097

Columns 22 through 28

0.1479	82.3781	82.3781	0.3230	82.4298	82.3781	0.3913
-10.8908	55.0594	55.0594	55.0594	55.0594	55.0594	55.0594
1.9500	2.6599	7.0187	7.0187	7.0187	2.6599	7.0187
32.5097	32.5097	32.5097	32.5097	32.5097	37.5518	32.5097

```
Columns 29 through 35

  0.1514   0.0630   0.1540   0.0679   0.1563   0.0720   0.1581

-10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908

  1.9500   1.9500   1.9500   1.9500   1.9500   1.9500   1.9500

 32.5097  32.5097  32.5097  32.5097  32.5097  32.5097  32.5097


Columns 36 through 42

  0.0750   0.1592   0.0774   0.1605   0.0796   0.1614   0.0813

-10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908

  1.9500   1.9500   1.9500   1.9500   1.9500   1.9500   1.9500

 32.5097  32.5097  32.5097  32.5097  32.5097  32.5097  32.5097


Columns 43 through 49

  0.0829   0.1630   0.0844   0.0858   0.0871   0.1650   0.0883

-10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908

  1.9500   1.9500   1.9500   1.9500   1.9500   1.9500   1.9500

 32.5097  32.5097  32.5097  32.5097  32.5097  32.5097  32.5097


Columns 50 through 56

  0.0894   0.0903   0.1666   0.0913   0.1671   0.0923   0.0933

-10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908 -10.8908

  1.9500   1.9500   1.9500   1.9500   1.9500   1.9500   1.9500

 32.5097  32.5097  32.5097  32.5097  32.5097  32.5097  32.5097
```

```
Columns 57 through 63

  0.1680    0.0942    0.1685    0.0950    0.0958    0.0966    0.1696

-10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908

  1.9500    1.9500    1.9500    1.9500    1.9500    1.9500    1.9500

 32.5097   32.5097   32.5097   32.5097   32.5097   32.5097   32.5097


Columns 64 through 70

  0.0972    0.0979    0.0986    0.0992    0.1709    0.4234    0.4583

-10.8908  -10.8908  -10.8908  -10.8908  -10.8908   55.0594   55.0594

  1.9500    1.9500    1.9500    1.9500    1.9500    7.0187    7.0187

 32.5097   32.5097   32.5097   32.5097   32.5097   32.5097   32.5097


Columns 71 through 77

  0.4992    0.5479    0.6037    0.6818    0.7721    0.8687   82.7046

 55.0594   55.0594   55.0594   55.0594   55.0594   55.0594   55.0594

  7.0187    7.0187    7.0187    7.0187    7.0187    7.0187    7.0187

 32.5097   32.5097   32.5097   32.5097   32.5097   32.5097   32.5097


Columns 78 through 84

  0.1712    0.1716    0.1013    0.1720    0.1723    0.1026    0.1730

-10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908

  1.9500    1.9500    1.9500    1.9500    1.9500    1.9500    1.9500

 32.5097   32.5097   32.5097   32.5097   32.5097   32.5097   32.5097
```

```
Columns 85 through 91

  0.1041    0.1734    0.1054    0.1061    0.1744    0.1751    0.1092

-10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908

  1.9500    1.9500    1.9500    1.9500    1.9500    1.9500    1.9500

 32.5097   32.5097   32.5097   32.5097   32.5097   32.5097   32.5097


Columns 92 through 98

  0.1760    0.1770    0.1124    0.1141    0.1812    0.1269    0.1879

-10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908  -10.8908

  1.9500    1.9500    1.9500    1.9500    1.9500    1.9500    1.9500

 32.5097   32.5097   32.5097   32.5097   32.5097   32.5097   32.5097


Columns 99 through 103

  0.1354    0.1917    0.1948    0.1840    0.2031

-10.8908  -10.8908  -10.8908  -10.8908  -10.8908

  1.9500    1.9500    1.9500    1.9500    1.9500

 32.5097   32.5097   32.5097   32.5097   32.5097
```

A.2 RBF Radii:

```
Columns 1 through 7

 164.2136 164.2136   0.1346 164.0790 164.2136  76.8582   0.0290

 154.2284 154.2284 154.2284 154.2284 154.2284 154.2284 154.2284

  18.5681   9.8192   8.7488   8.7488   0.4134  18.5681  18.5681

  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342


Columns 8 through 14

 164.2136 164.2136 164.2136 164.2136 164.2136  76.8582  76.4881

 154.2284 154.2284 154.2284 154.2284 154.2284 131.9004  22.3280

   0.4867   8.9192   0.4867   0.4867   0.2016  18.5681  18.5681

  80.1342  80.1342   9.4182  70.7160  80.1342  80.1342  80.1342


Columns 15 through 21

   0.3701   0.1155   0.2546   0.0391   0.2155   0.1981   0.0093

  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280

  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681

  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342


Columns 22 through 28

   0.1888 164.2136 164.2136   0.1035 164.1101 164.2136   0.0331

  22.3280 154.2284 154.2284 154.2284 154.2284 154.2284 154.2284

  18.5681   0.2871   8.4305   8.4305   8.4305   0.2871   8.4305

  80.1342  80.1342  80.1342  80.1342  80.1342  70.0500  80.1342
```

```
Columns 29 through 35

 0.1818   0.0051   0.1767   0.0046   0.1721   0.0036   0.1685

22.3280  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280

18.5681  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681

80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342


Columns 36 through 42

 0.0023   0.1662   0.0026   0.1636   0.0017   0.1619   0.0017

22.3280  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280

18.5681  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681

80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342


Columns 43 through 49

 0.0015   0.1587   0.0015   0.0013   0.0013   0.1546   0.0012

22.3280  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280

18.5681  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681

80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342


Columns 50 through 56

 0.0009   0.0010   0.1515   0.0010   0.1505   0.0011   0.0009

22.3280  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280

18.5681  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681

80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342
```

```
Columns 57 through 63

  0.1486    0.0009    0.1477    0.0008    0.0009    0.0006    0.1454

 22.3280   22.3280   22.3280   22.3280   22.3280   22.3280   22.3280

 18.5681   18.5681   18.5681   18.5681   18.5681   18.5681   18.5681

 80.1342   80.1342   80.1342   80.1342   80.1342   80.1342   80.1342


Columns 64 through 70

  0.0007    0.0007    0.0006    0.0007    0.1428    0.0311    0.0385

 22.3280   22.3280   22.3280   22.3280   22.3280  154.2284  154.2284

 18.5681   18.5681   18.5681   18.5681   18.5681    8.4305    8.4305

 80.1342   80.1342   80.1342   80.1342   80.1342   80.1342   80.1342


Columns 71 through 77

  0.0434    0.0539    0.0577    0.0986    0.0820    0.1111  163.5606

154.2284  154.2284  154.2284  154.2284  154.2284  154.2284  154.2284

  8.4305    8.4305    8.4305    8.4305    8.4305    8.4305    8.4305

 80.1342   80.1342   80.1342   80.1342   80.1342   80.1342   80.1342


Columns 78 through 84

  0.1422    0.1414    0.0008    0.1407    0.1400    0.0005    0.1386

 22.3280   22.3280   22.3280   22.3280   22.3280   22.3280   22.3280

 18.5681   18.5681   18.5681   18.5681   18.5681   18.5681   18.5681

 80.1342   80.1342   80.1342   80.1342   80.1342   80.1342   80.1342
```

```
Columns 85 through 91

  0.0007   0.1379   0.0007   0.0007   0.1359   0.1343   0.0008

 22.3280  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280

 18.5681  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681

 80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342


Columns 92 through 98

  0.1327   0.1306   0.0014   0.0020   0.1222   0.0134   0.1087

 22.3280  22.3280  22.3280  22.3280  22.3280  22.3280  22.3280

 18.5681  18.5681  18.5681  18.5681  18.5681  18.5681  18.5681

 80.1342  80.1342  80.1342  80.1342  80.1342  80.1342  80.1342


Columns 99 through 103

  0.0037   0.1013   0.0951   0.0029   0.0074

 22.3280  22.3280  22.3280  22.3280  22.3280

 18.5681  18.5681  18.5681  18.5681  18.5681

 80.1342  80.1342  80.1342  80.1342  80.1342
```

A.3 RBF Weights: 1.0e+003 *

1 to 21	22 to 42	43 to 63	64 to 84	85 to 103
0.0008	-2.1481	0.0000	0.0000	-0.0000
0.0250	0.0156	2.7871	0.0000	0.0202
-0.0002	-1.0693	0.0000	0.0000	-0.0000
-0.0042	0.0021	0.0000	0.0000	0.0000
0.0000	-0.7011	0.0000	-0.9954	0.4093
0.0147	-0.0173	2.7355	0.0005	1.0767
-0.0001	0.0005	-0.0000	0.0009	-0.0000
1.2521	-3.3815	0.0000	0.0007	0.8953
-0.0543	0.0000	-0.0000	0.0006	0.8563
-0.0550	-5.1024	-0.0070	0.0004	-0.0000
-1.3340	0.0000	0.0000	0.0009	-0.0000
-0.0001	-2.3199	0.1945	0.0001	1.9783
-0.0120	0.0000	0.0000	0.0007	0.0000
-0.0009	0.2221	0.0000	1.8049	-2.6575
-0.0082	0.0000	-0.6141	-1.8438	-0.0000
0.0019	1.3877	0.0000	-1.6324	1.8304
0.2864	0.0000	-1.4453	0.0000	-0.4556
-0.0000	3.0710	-0.0000	-0.5817	0.0000
-3.3314	-0.0000	0.0000	-0.9028	0.0000
9.0486	2.2036	-0.0000	0.0000	
0.0000	0.0000	-0.9039	-0.6729	

Table 1. Weights associated with each RBF center

Appendix B

B.1 missile_data_collection.m

```
% Purpose: Read frames from the datafile one at a time, and find the
brightest pixel in each frame. Then extract the 3 by 3 array of pixels
immediately surrounding that pixel, calculate features from the pixels,
and arrange the features in a row vector.

clear all;
close all;
i = 1;
missile = zeros(2500,4);
for m = 0:2:5000
   % read the data for the red and blue bands
   [im_red] = ldgal('sdata6', m, 256);
   [im_blue] = ldgal('sdata6', m+1, 256);
   % find the brightest spot in each frame and its matrix coordinate
   [r1, c1] = find(im_red==max(max(im_red)));
   [r2, c2] = find(im_blue==max(max(im_blue)));
   % if the peak intensity occurs over two pixels, pick the larger of the
   two coordinates
   y1 = ceil(mean(r1));
   x1 = ceil(mean(c1));
   y2 = ceil(mean(r2));
   x2 = ceil(mean(c2));
   % extract the 3 x 3 matrix around each pixel
   red = im_red((y1-1):(y1+1),(x1-1):(x1+1));
   blue = im_blue((y2-1):(y2+1),(x2-1):(x2+1));
   % correct for variations between images and normalize the dataset by
   % subtracting the mean from each value and dividing by the maximum
   max_red = zeros(3) + max(max(im_red));
   max_blue = zeros(3) + max(max(im_blue));
   mean_red = zeros(3) + mean(mean(im_red));
   mean_blue = zeros(3) + mean(mean(im_blue));
   new_red = (red - mean_red)./max_red;
   new_blue = (blue - mean_blue)./max_blue;
   % define the outer ring of pixels
   red_outer = [new_red(1,1) new_red(1,2) new_red(1,3) new_red(2,1)
         new_red(2,3) new_red(3,1) new_red(3,2) new_red(3,3)];
   blue_outer = [new_blue(1,1) new_blue(1,2) new_blue(1,3) new_blue(2,1)
         new_blue(2,3) new_blue(3,1) new_blue(3,2) new_blue(3,3)];
   % calculate the Z score and Holmes parameter features for training the
     neural network
   f1 = new_red(2,2)/sum(red_outer);
   f2 = new_blue(2,2)/sum(blue_outer);
   f3 = (new_red(2,2) - mean(red_outer))/sqrt(var(red_outer));
   f4 = (new_blue(2,2) - mean(blue_outer))/sqrt(var(red_outer));
   missile(i,:) = [f1 f2 f3 f4];
   i = i+1;
end
```

B.2 background_data_collection.m

```
% Purpose: Read frames from background datafiles one at a time, and take
a 150x150 portion from a characteristic region of the image. Divide the
portion into 2500 (3 by 3) image chips, and arrange the data in a table.

clear all;
close all;
i = 1;                          % index value
background_data = zeros(2500,4); % initialize table of pixel values
m = 2;
   % read the data for the red and blue bands
   [im_red] = ldgal('eglinbeach1', m, 256);
   [im_blue] = ldgal('eglinbeach1', m+1, 256);
   % extract the 150x150 portion of the images
   red = im_red(1:150,101:250);
   blue = im_blue(1:150,101:250);
   % find the maximums and means of the 256x256 images
   max_red = zeros(150) + max(max(im_red));
   max_blue = zeros(150) + max(max(im_red));
   mean_red = zeros(150) + mean(mean(im_red));
   mean_blue = zeros(150) + mean(mean(im_blue));
   % subtract the mean and divide by the maximum pixel values
   new_red = (red - mean_red)./max_red;
   new_blue = (blue - mean_blue)./max_blue;
   % divide into 3 by 3 chips and rearrange into a row vector,
   for y = 1:3:150
      for x = 1:3:150
         % define the outer ring of pixels
         red_outer = [new_red(y,x) new_red(y,x+1) new_red(y,x+2)
               new_red(y+1,x) new_red(y+1,x+2) new_red(y+2,x)
               new_red(y+2,x+1) new_red(y+2,x+2)];
         blue_outer = [new_blue(y,x) new_blue(y,x+1) new_blue(y,x+2)
               new_blue(y+1,x) new_blue(y+1,x+2) new_blue(y+2,x)
               new_blue(y+2,x+1) new_blue(y+2,x+2)];
         % calculate the Z score and Holmes parameter features
        f1 = new_red(y+1,x+1)/sum(red_outer);
        f2 = new_blue(y+1,x+1)/sum(blue_outer);
        f3 = (new_red(y+1,x+1) - mean(red_outer))/sqrt(var(red_outer));
        f4 = (new_blue(y+1,x+1) - mean(blue_outer))/sqrt(var(red_outer));
        background_data(i,:) = [f1 f2 f3 f4];
        i = i+1;
      end
   end
eglinbeach1 = background_data;

%***********************************************************************
% repeat for images from the 13a, 14a and 15a data sets, then
% combine into a single combined vector together with the missile data
%***********************************************************************
background = [eglinbeach1;13a;14a;15a];
load missile
Xtr = transpose([missile;background]);
```

B.3 rbfnn_training.m

```
% Design and training of a gaussian radial basis function neural network
% using Matlab Functions for Radial Basis Function Networks by Mark J.L.
Orr, Institute for Adaptive and Neural Computation, Division of
Informatics, Edinburgh University, Scotland, UK

load input
load target

% Configure parameters
conf.lambda = .0000000001;
conf.msc = 'loo';
conf.minmem = 100;
conf.scales = 2;
conf.timer = 'optimization in progress'

% Start the function that determines the centres and radii of a set of
RBFs using the training data and the expected outputs

% function [C, R, w, info, conf] = rbf_rt_1(X, y, conf)
%
% Hybrid radial basis function network and regression tree.
%
% Solves a regression problem with inputs X and outputs y using a
regression tree and an RBF network selected using tree-guided forward and
backward subset selection. Returns the hidden unit centres C, their radii
R, the hidden-to-output weights w, some additional information info and a
fully instantiated configuration structure conf.
%
% X is an n-by-p matrix of inputs, where p specifies the number of cases,
and n specifies the number of features per case.
% y is a p-by-1 matrix of outputs.
% C is an n-by-m matrix, where m specifies the number of RBF units, and
each n column corresponds to
% one centre in the input space.
% R is also an n-by-m matrix, where each column corresponds to a set of
'n' scaling parameters, one for each feature, which determine the width
of the m-th RBF unit.
% w is either an m- or an (m+1)-dimensional vector depending on whether
the method has included a bias unit in the network.
% C, R, and w are used to make predictions from the network.

[C, R, w, info, conf] = rbf_rt_1(Xtr, y, conf)
```

B.4 rbfnn_testing.m

```
clear all;
close all;

% load the radial basis function network's centre, radii and weight
vectors
load centres
load radii
load weights

n = 1;

for m = 0:2:8;
im_response = zeros(256); % create a 256x256 matrix of zeros

% read data from consecutive red and blue bands
[im_red] = ldgal('m:\sdata8', m, 256);
[im_blue] = ldgal('m:\sdata8', m+1, 256);

% find the maximums and means of the images in each band
max_red = zeros(256)+max(max(im_red));
mean_red = zeros(256)+mean(mean(im_red));
max_blue = zeros(256)+max(max(im_blue));
mean_blue = zeros(256)+mean(mean(im_blue));

% subtract off the mean and divide by the maximum value in each image
im_red = (im_red - mean_red)./max_red;
im_blue = (im_blue - mean_blue)./max_blue;

% divide the image into overlapping 3 by 3 chips
new_im_red = transpose(im2col(im_red, [3 3], 'sliding'));
new_im_blue = transpose(im2col(im_blue, [3 3], 'sliding'));

for i = 1:64516
   % define the outer gate pixels in a row vector
   red_outergate = [new_im_red(i,1) new_im_red(i,2) new_im_red(i,3)
               new_im_red(i,4) new_im_red(i,6) new_im_red(i,7)
               new_im_red(i,8) new_im_red(i,9)];

   blue_outergate = [new_im_blue(i,1) new_im_blue(i,2) new_im_blue(i,3)
               new_im_blue(i,4) new_im_blue(i,6) new_im_blue(i,7)
               new_im_blue(i,8) new_im_blue(i,9)];

   % Derive Z-scores for each band
   f1 = (8*new_im_red(i,5))/sum(red_outergate);
   f2 = (8*new_im_blue(i,5))/sum(blue_outergate);

   % Derive Holmes parameters for each band
   f3 = (new_im_red(i,5)-mean(red_outergate))/sqrt(var(red_outergate));
   f4 = (new_im_blue(i,5)-
                        mean(blue_outergate))/sqrt(var(blue_outergate));
   % Generate a 4-dimensional vector from these features for input to the
rbf network
   Xt(i,:) = [f1 f2 f3 f4];
end

% use the RBF network centres, radii and weights to predict the outputs
from each 3 by 3 chip
Ht = rbf_dm(transpose(Xt), C, R);
ft = Ht * w;

% rearrange the response vector into a probability density map
im_response(2:255,2:255) = col2im(ft,[254 254],[254 254],'distinct');

% rescale the map with a range of 0 to 1
im_response(im_response<0) = 0;
im_response(im_response>1) = 1;

% find the indices of the map where its elements are above a certain
threshold
[tgt_r, tgt_c] = find(im_response > 0.95);

% plot the target's possible locations on a 2-dimensional map
figure; plot(tgt_c, tgt_r, 'r.'); axis([1 255 1 255]); grid on;
figure; surf(im_red); shading interp; axis([1 255 1 255 0 .1]); view(2);

% store the original image and probability density map for comparison
analysis
red(:,:,n) = im_red;
blue(:,:,n) = im_blue;
map(:,:,n) = im_response;

n = n + 1;

end
```

B.5 detection_algorithm.m

```
clear all;
close all;

load centres
load radii
load weights

threshold = 0.01;
temp_red = zeros(258);
temp_blue = zeros(258);
x = 0.05:.05:1;             % set up the axes for an ROC curve
y = zeros(1,10);
M = 3600;                   % set the number of the last frame to be analyzed
j = 1;

%for m = 0:2:M;
m = 700;
im_response = zeros(258); % create 256x256 matrices of zeros with an
                            additional border 1 pixel-wide
% read data from consecutive red and blue bands
[im_red] = ldgal('m:\sdata8', m, 256);
[im_blue] = ldgal('m:\sdata8', m+1, 256);

% find the maximums and means of the images in each band
max_red = zeros(256)+max(max(im_red));
mean_red = zeros(256)+mean(mean(im_red));
max_blue = zeros(256)+max(max(im_blue));
mean_blue = zeros(256)+mean(mean(im_blue));

% subtract off the mean and divide by the maximum value in each image
im_red = (im_red - mean_red)./max_red;
im_blue = (im_blue - mean_blue)./max_blue;

temp_red(2:257,2:257) = im_red;
temp_blue(2:257,2:257) = im_blue;

% apply global thresholding to each image
[r, c] = find(temp_red > threshold);

for n = 1:length(r)
   % define the outer gate pixels in a vector
   red_outergate = [temp_red(r(n)+1,c(n)-1) temp_red(r(n)+1,c(n))
temp_red(r(n)+1,c(n)+1) ...
         temp_red(r(n),c(n)-1) temp_red(r(n),c(n)+1) temp_red(r(n)-
1,c(n)-1) temp_red(r(n)-1,c(n)) ...
         temp_red(r(n)-1,c(n)+1)];

   blue_outergate = [temp_blue(r(n)+1,c(n)-1) temp_blue(r(n)+1,c(n))
temp_blue(r(n)+1,c(n)+1) ...
         temp_blue(r(n),c(n)-1) temp_blue(r(n),c(n)+1) temp_blue(r(n)-
1,c(n)-1) temp_blue(r(n)-1,c(n)) ...
         temp_blue(r(n)-1,c(n)+1)];
   % Derive Z-scores for each band
   f1 = (8*temp_red(r(n),c(n)))/sum(red_outergate);
   f2 = (8*temp_blue(r(n),c(n)))/sum(blue_outergate);

   % Derive Holmes parameters for each band
   f3 = (temp_red(r(n),c(n))-
mean(red_outergate))/sqrt(var(red_outergate));
   f4 = (temp_blue(r(n),c(n))-
mean(blue_outergate))/sqrt(var(blue_outergate));

   % Generate a 4-dimensional vector from these features for input to the
rbf network
   Xt = [f1;f2;f3;f4];

   % use the RBF network centres, radii and weights to predict the
outputs from each 3x3 chip
   Ht = rbf_dm(Xt, C, R);
   im_response(r(n), c(n)) = Ht * w;
end

% remove the border pixels to return to a 256x256 array again
im_response(1,:)=[]; im_response(:,1)=[]; im_response(257,:)=[];
im_response(:,257)=[];

%surf(im_response); shading interp; axis([0 255 0 255 0 1]); view(2);
hold on;
%surf(im_red); shading interp; colormap(gray); axis([0 255 0 255 0 0.1]);
view(2); figure;

% find the number of targets that are greater than or equal to each
probability
for i = 1:20
   y = sum(sum(im_response >= x(i)));
   if y-1 < 0
      num_correctly_detected(i) = 0;
      num_false_alarms(i) = 0;
   else
      num_correctly_detected(i) = 1;
      num_false_alarms(i) = y - 1;
   end
end
plot(num_false_alarms,num_correctly_detected,'kx-');
axis([-1 max(num_false_alarms) 0 1.1])
xlabel('# False Alarms'); ylabel('# Correctly Detected')
```

Bibliography

Batelle, 1997. Batelle Memorial Institute, "What is an Artificial Neural Network", Pacific Northwest National Laboratories, 1997, http://www.emsl.pnl.gov:2080/proj/neuron/neural/neural.ann.html.

Baxley, 2000. Baxley, Frank O., Sanderson, Richard B., Montgomery, Joel B., "Evaluation of two-color missile detection algorithms against real backgrounds", SPIE Aerosense Conference, April 2000, Orlando, Florida.

Bishop, 1995. Bishop, Christopher M., "Neural Networks for Pattern Recognition", Oxford University Press Inc., New York, 1995.

Broomhead, 1988. Broomhead, D. S., Lowe, D., "Multivariable functional interpolation and adaptive networks", *Complex Systems*, 1988, and Moody, J., Darken, C. J., "Fast learning in networks of locally-tuned processing units", *Neural Computation*, 1989.

Haykin, 1999. Haykin, Simon, "Neural Networks: A Comprehensive Foundation", 2nd Ed., pp. 156-157, Prentice-Hall Inc., 1999.

Hoffman, 2000. Hoffman, Howard S., "The Internet Glossary of Statistical Terms – Z Score", in The Animated Software Company, last modified January 2000, http://www.animatedsoftware.com/statglos/sgzscore.htm.

Kubat, 1995. Kubat M., Ivanova I., "Initialization of RBF networks with decision trees", in *Proceedings of the 5th Belgian-Dutch Conference of Machine Learning, BENELEARN'95*, pp. 61-70, 1995.

Montgomery, 2000. Montgomery, Joel B., Baxley, Frank O., Sanderson, Richard B., "Tactical Mid-Infrared Test-bed", Targets and Backgrounds VI: Characterization, Visualization, and the Detection Process, SPIE Aerosense Conference, April 2000, Orlando, Florida.

Morin, 2000. Morin A., "Adaptive spatial filtering techniques for the detection of targets in infrared imaging seekers", Acquisition Tracking and Pointing XIV, SPIE Aerosense Conference, April 2000, Orlando, Florida.

Orr, 1996. Orr, Mark, "Introduction to Radial Basis Function Networks", Institute for Adaptive and Neural Computation, Division of Informatics, Edinburgh University, Edinburgh EH8 9LW, Scotland, England, April 1996, www.anc.ed.ac.uk/~mjo/papers/intro.ps.

Orr, 1999. Orr, Mark, "Matlab Functions for Radial Basis Function Networks", Institute for Adaptive and Neural Computation, Division of Informatics, Edinburgh University, Edinburgh EH8 9LW, Scotland, England, June 30 1999, http://www.anc.ed.ac.uk/~mjo.

VITA

Flight Lieutenant (FLTLT) Kin-Weng Chan joined the Royal Australian Air Force (RAAF) in 1996 after graduating from the University of Adelaide with a Bachelor of Engineering degree (with Honours) in Electrical and Electronic Engineering. From October 1996 to August 1999 he served as the F/A-18 Electronic Countermeasures Engineer, Electronic Warfare Squadron (EWSQN), Aircraft Research and Development Unit (ARDU) located at RAAF Base Edinburgh, Adelaide, South Australia. In 1999 he was assigned to the United States Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton, Ohio, to pursue a Masters of Science degree in Electrical Engineering majoring in Electro-Optics and Automatic Target Recognition. Following graduation, FLTLT Chan will return to Australia to serve as the Electro-Optics Research and Development Engineer (EOR&D) at EWSQN.

Permanent Address: EOR&D
Research and Development Flight
EWSQN ARDU
RAAF Base
Edinburgh SA 5111
AUSTRALIA

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)* 02-12-2001	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* Jun 2000 – Feb 2001

4. TITLE AND SUBTITLE A RADIAL BASIS FUNCTION NEURAL NETWORK APPROACH TO TWO-COLOR INFRARED MISSILE DETECTION	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Chan, Kin-Weng, Flight Lieutenant, Royal Australian Air Force	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Building 640 WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/01M-05
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratories, Sensors Directorate, Infrared Technologies Branch Attn: 1Lt. Daniel J. Petrovich 3109 P St. Bldg 622, WPAFB, OH, 45433-7700 Tel: (937) 255-9614 x242	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/SNJM **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Multi-color infrared imaging missile-warning systems require real-time detection techniques that can process the wide instantaneous field of regard of focal plane array sensors with a low false alarm rate. Current technology applies classical statistical methods to this problem and ignores neural network techniques. Thus the research reported here is novel in that it investigates the use of radial basis function (RBF) neural networks to detect sub-pixel missile signatures. An RBF neural network is designed and trained to detect targets in two-color infrared imagery using a recently developed regression tree algorithm. Features are calculated for 3 by 3 pixel sub-images in each color band and concatenated into a vector as input to the network. The RBF network responds with a value of unity to feature vectors representing missiles and with zero to vectors representing background. Images are thresholded prior to application to the trained RBF network to narrow the field of interest of the RBF network and increase missile detection speed. The RBF network-based technique then generates potential target locations and probabilities that the locations correspond to missiles. Results show that the RBF network-based technique operates in near real-time and detects 100% of the missiles in data that was not used in training. Receiver operating characteristic (ROC) curves show that overly high classification thresholds can exceed the RBF network response for a true missile and result in non-detection. However, these ROC curves also show that adaptive control of the classification threshold on the RBF network output can reduce the number of false alarms to zero.

15. SUBJECT TERMS

Radial basis function neural network, two-color infrared, sub-pixel missile signature, regression tree, real-time.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Steven C. Gustafson, ENG
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)*
U	U	U	UU	101	(937) 255-3636, ext 4598

Standard Form 298 (Rev. 8-98)