2007

Anomaly detection in hyperspectral imagery: a comparison of methods using diurnal and seasonal data

Patrick C. Hytla
University of Dayton

Anomaly Detection in Hyperspectral Imagery: A Comparison of Methods Using Diurnal and Seasonal Data

Thesis

Submitted To

School of Engineering

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree

Master of Science in Electrical Engineering

By

Patrick C. Hytla

UNIVERSITY OF DAYTON

Dayton, Ohio

December 2007

Anomaly Detection in Hyperspectral Imagery: A Comparison of Methods Using Diurnal and Seasonal Data

APPROVED BY:

[redacted]

Russell Hardie, Ph.D.
Advisory Committee Chairman
Professor Electrical Engineering
University of Dayton

[redacted]

Michael Eismann, Ph.D.
Committee Member
Research Engineer
Air Force Research Lab

[redacted]

Raul Ordoñez, Ph.D.
Committee Member
Professor Electrical Engineering
University of Dayton

[redacted]

Malcolm W. Daniels, Ph.D.
Associate Dean
School of Engineering

[redacted]

Joseph E. Saliba, Ph.D., P.E.
Dean, School of Engineering

ABSTRACT

Anomaly Detection in Hyperspectral Imagery: A Comparison of Methods Using Diurnal and Seasonal Data

Patrick Hytla
University of Dayton, 2007

Advisor: Dr. Russell C. Hardie

Hyperspectral Imaging (HSI) is an area of growing interest due its rich spectral content. Hyperspectral images contain a large number of spectral bands that represent electromagnetic radiation in the visible and infrared regions of the electromagnetic spectrum. Remotely sensed hyperspectral images, or images collected by a sensor some distance from a target, have applications in surveillance, search and rescue operations and military target detection and tracking. A particular area of interest within hyperspectral remote sensing is known as anomaly detection. Anomaly detection is a processing technique capable of finding spectral outliers, or anomalies, in hyperspectral data without *a priori* information about the scene or its contents. Generally, this is accomplished by creating a statistical model of the scene's background and detecting image pixels that do not conform to that given model. Many anomaly detection methods have been presented in literature. In this study we compare the performance of several of these detectors when applied to hyperspectral data sets that feature both diurnal and seasonal changes. These methods include a Mahalanobis distance (M-dist) anomaly detector, a locally adaptive Reed-Xiaoli (RX) anomaly detector, a Gaussian mixture model (GMM) anomaly detector, a Gaussian mixture RX (GMRX) anomaly detector and a cluster-based anomaly detector (CBAD).

Additionally, a new anomaly detector referred to as the fuzzy cluster-based anomaly detector (FCBAD) is introduced and its performance is compared to existing methods.

The use of anomaly detectors in change detection algorithms is another area of interest in the field of hyperspectral imaging. The goal of change detection is to identify differences, often very subtle, that are present between two images of the same scene taken at different times. In many cases the time elapsed between the two images is significant and natural changes will be present. It is preferable to suppress these natural changes, such as illumination conditions and seasonal changes, and exploit man-made changes. To accomplish this, change detection algorithms rely on two main components: a predictor and an anomaly detector. This study investigates global and cluster-based prediction and detection techniques and evaluates overall change detection performance.

ACKNOWLEDGMENTS

There are a number of people who deserve thanks and recognition for their assistance in the completion of this research. I would like to thank Russell Hardie for acting as my advisor at the University of Dayton. His guidance, assistance and expertise with regard to signal and image processing made this research possible. I would like to thank Michael Eismann and Joseph Meola of the Air Force Research Labs. Mike was kind enough to take time from his very busy schedule to serve on my committee and to answer any and all of my numerous questions. Joe was also a valuable resource in answering my questions. Additionally, I, as well as many others, owe Joe a debt of gratitude for his work collecting the data used in this study and numerous other anomaly and change detection studies. I would also like to thank Raúl Ordóñez for taking the time to serve on my committee.

I would also like to thank the staff of the Electrical and Computer Engineering department at the University of Dayton. The professors have been fantastic in both their teaching and assistance during my stay at the university. I would like to give a special thanks to Loretta Christon and Marilyn Knisley for their assistance. Their knowledge and support allowed me to proceed in a timely manner with this thesis.

Last, but most certainly not least I would like to thank my family and friends. My parents have provided me with love and support throughout my entire life. I thank them so much for that, and for the opportunity to attend the schooling that made this work possible. I would like to thank my friends for their support and for the much needed diversion from technical and research related work that they provide.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

This section provides an overview of important concepts relevant to this research. Additionally, the experimental setup and a set of research goals are presented.

1.1 Hyperspectral Imaging

The electromagnetic spectrum covers the frequency, or wavelength, range of all electromagnetic radiation. Hyperspectral imaging (HSI) involves the collection of a large number of wavelength bands, generally in some range of visible to infrared radiation (0.4-14 μm). The radiation is due to reflected radiation from the Sun as well as thermal emissions from sources on Earth. The electromagnetic spectrum generally associated with hyperspectral imaging and the sources of the radiation can be seen in Figure 1.1.1.

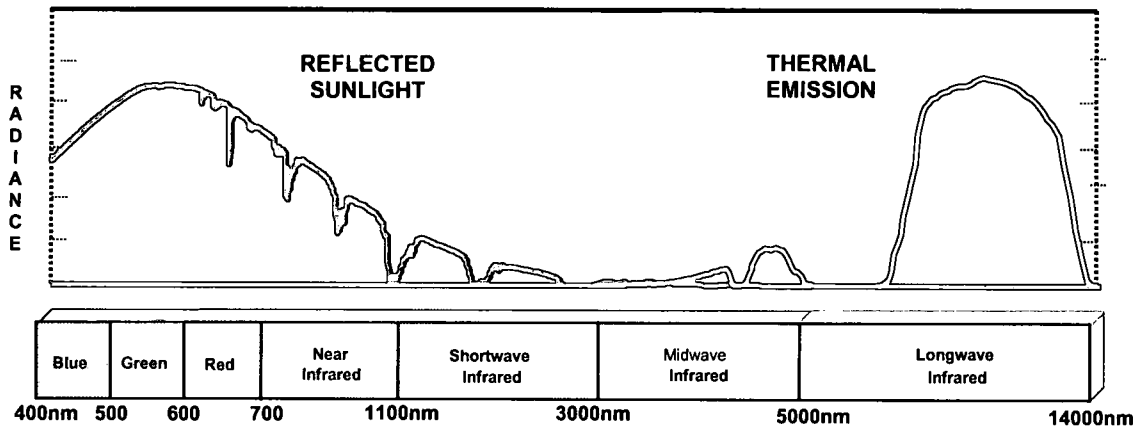

Figure 1.1.1: Electromagnetic spectrum. Notice that the visible through short-wave infrared regions are dominated by reflected sunlight and thermal emission can be ignored, while the long-wave infrared is dominated by thermal emissions and reflected sunlight can be ignored. Both reflected sunlight and thermal emission are present in the mid-wave infrared region and must be accounted for in calculations.

The spectral information collected by a hyperspectral imaging system provides a wealth of information about an object. Objects reflect or emit radiation at various wavelengths in different ways due to their material composition. This is clearly observable in the visible portion of the electromagnetic spectrum; an object appearing blue in color reflects more radiation near 0.4 μm than at other wavelengths in the visible spectrum. Objects also reflect or emit radiation in other portions of the spectrum, such as the infrared, that are not observable by the human eye. Sampling a portion of the reflected or emitted radiation from an object at a number of finely spaced band measurements can potentially provide a 'spectral signature' of an object[1]. This signature grants an insight into the composition of an object and can act as a means of identification.

A wide variety of imaging systems can be employed to capture hyperspectral data. One common system is known as a dispersive imaging spectrometer. These systems contain some sort of imaging optics, normally a lens system, a slit, a dispersive material and a focal place array (FPA) that converts the intensity of the captured radiation to an electric signal. The dispersive material, often a prism or diffraction grating, diffracts captured radiation in a wavelength dependant fashion onto the FPA. For example, radiation at 0.4 μm will be diffracted at a different angle than radiation at 0.7 μm. A limitation of dispersive spectrometers is that they may only capture two dimensions at a time (usually one spatial dimension, often called swath width and one spectral dimension). To capture the third dimension (spatial) the instrument must be physically moved or scanned. This procedure for dispersive spectrometers is referred to as a pushbroom method. With both spatial and spectral information present, the image acquired is sometimes called a 'hypercube.' In Cartesian coordinates, the spatial positions represent the x-direction and the y-direction,

while the spectral information is represented in the z-direction. All of the spectral measurements at a single spatial coordinate are often referred to as a 'hyperpixel.' These concepts are illustrated in Figure 1.1.2.

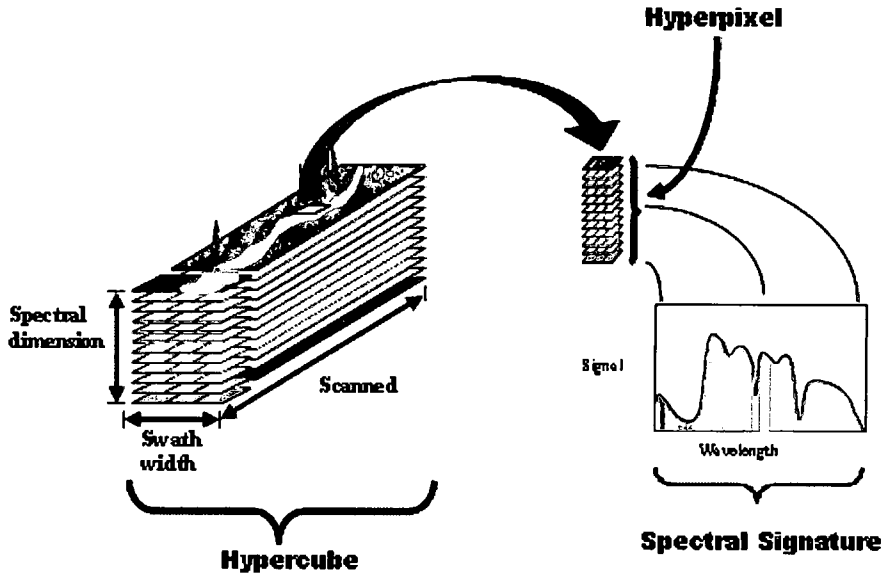

Figure 1.1.2: Concepts of Hyperspectral Imagery. This figure visually presents some hyperspectral concepts. A hyperpixel consists of all the spectral bands at a single spatial location. These spectral bands form a spectral signature, which can aid in identification of the pixel's contents. A hypercube is a collection of hyperpixels at various spatial locations that create a scene. With most imaging spectrometers it is only possible to collect two dimensions of a hypercube at any given time. The third dimension, referenced in the figure as the 'Scanned' dimension, is collected by physically moving the instrument.

The collection of spectral information is ideally done at a small distance from an object in a laboratory environment where all variables can be controlled. In most cases, however, this sort of procedure is completely unrealistic or impossible. A much more common and realistic method for the collection of hyperspectral data is known as remote sensing. Remote sensing involves collecting data over a wide area at some distance[1]. The remotely sensed scene often contains objects of interest, or anomalies, in a complex background. These anomalies are often very small in comparison to the number of total image pixels and may be undetectable by the human eye. Anomaly detection is an image

processing technique capable of detecting the object of interest within the background pixels without *a priori* knowledge of the scene. This technique creates a statistical background model for an image and detects pixels that do not fit that background model. Many methods have been developed to calculate the background model's probability density function (pdf) and these are covered thoroughly in existing literature[2,3,4]. A similar concept, known as change detection, attempts to identify man-made changes between two images of the same scene captured at different times. Change detection algorithms also attempt to suppress minor natural changes in the background of the images through a prediction process. However, before anomaly or change detection can take place, calibrations must be done to determine the imaging spectrometer's specific response characteristics.

1.2 Research Goals

This research uses several hyperspectral anomaly detection methods to detect targets in a remotely sensed scene. Images of the scene, containing four painted aluminum panels and a naturally wooded background, are collected over a period from late August 2005 through early May 2006. Data is collected during this period with a constant solar azimuth and a variety of natural seasonal changes. A diurnal data set is also collected accounting for the illumination changes experienced from 0800 hrs to 1900 hrs on May 8, 2006. Chapter 2 further describes the data collection procedure as well as calibrations and corrections employed to ensure the accuracy of the data. Several anomaly detection methods are then applied to both the seasonal and diurnal data sets to experimentally measure their detection performance. The anomaly detection algorithms are explained in detail in Chapter 3. Experimentation using several change detection algorithms is also conducted in this study.

Change detection theory is addressed in Chapter 4. Chapter 5 describes the experimental procedures used to measure each anomaly and change detection algorithm's performance and details the experimental results. Chapter 6 provides a summary of conclusions reached during this research process and addresses future research considerations.

CHAPTER 2

Hyperspectral Data Collection

This section provides an overview of the instruments and processes associated with the collection of hyperspectral data used in this research. It contains information on the imaging system as well as the calibration procedures employed to ensure accurate hyperspectral data is recovered. The scene setup and the collection procedures for both the seasonal and diurnal data collections are discussed. Additionally, due to the very large size of hyperspectral image files, a dimensionality reduction procedure is described.

2.1 Instrument Specifications

The data is collected using a Hyperspec VS-25 Imaging Spectrograph produced by Headwall Photonics[5]. It features a holographic diffraction grating to produce a spectral range from 400nm – 1000nm distributed into 128 wavelength bands. Using a 12μm slit width, the approximate full-width-half-maximum (FWHM) spectral resolution of the system is 2nm. The system uses a 50mm Navitar lens and a Dalsa Pantera TF 1M60 monochrome area scan CCD[6] in conjunction with the Hyperspec spectrometer. The square focal plane array (FPA) consists of 1024x1024 pixels, with each pixel having a side length of 12um. As configured for this research, the system operates with an exposure time of 70ms. The responsivity of the FPA is provided by the manufacturer and shown in Figure 2.1.1.

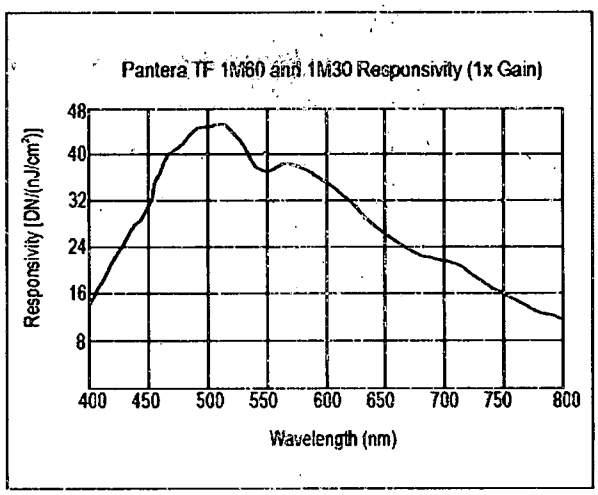

Figure 2.1.1: FPA Responsivity. The FPA is designed to primarily handle radiation in the visible section of the electromagnetic spectrum.

As is evident from Figure 2.1.1, the FPA is primarily designed to capture radiation in the visible section of the electromagnetic spectrum. The manufacturer's data is only represented from 400nm to 800nm, but the response of the FPA from 800nm-1000nm can be assumed to continue to fall at a fairly linear rate. The analog gain of the system is set to 4x for the research presented here. The camera and spectrometer are mounted on a pan and tilt system driven by two rotary motors and controlled by the Aerotech NDrive motion controller[7].

2.2 Spectral Calibration

While the manufacturer's specifications for each component in the imaging system are known when purchased, the components cannot be strictly counted upon to perform in a known fashion once the system is constructed and ready to collect data. To fully understand the operation of the entire system, and to ensure the collection of accurate data, several calibration and correction procedures must be completed. When an image is captured, the signal enters the spectrometer system and is dispersed onto the FPA. Each pixel on the FPA

corresponds to part of the image at a different wavelength. The individual response of each of pixel is unknown at the time of system construction, so a spectral calibration is done to determine this response. The goal of the spectral calibration is to map the center wavelength value to each pixel on the FPA. In a perfect spectrometer system the center wavelength should only vary from column to column, as that is the direction in which the signal is dispersed. However, errors often occur during manufacturing processes that manifest themselves in the form of smile, keystone, non-linear dispersion and misalignment. The overall equation for wavelength mapping takes these errors into account and is given as

$$\lambda(m,n) = am^2 + bn^2 + cmn + dm + en + f \tag{2.2.1}$$

where, m is the focal plane row, n is the focal place column, a is the smile coefficient, b is the non-linear dispersion coefficient, c is the coefficient of tilt, d is the rotation coefficient, e is the linear dispersion coefficient, and f is the spectral offset[8]. To determine these coefficients sources with known wavelengths are used. Specifically, a R-30025 He-Ne laser[9] and several Oriel Pencil Style Calibration Lamps[10] from Newport are used in this procedure. The gas lamps include a 6030 Argon lamp, a 6031 Krypton lamp, and a 6034 Mercury-Neon lamp. The known spectral lines of these sources are compared to the captured spectral response of the spectrometer system. The coefficients are determined through a least-squares fitting. Here, the coefficients are estimated as

$$a = b = c = 0$$
$$d = 0.0016$$
$$e = 1.1933$$
$$f = 105.7009$$

where a, b and c are sufficiently small (on the order of 10^{-6}) to be set to zero. Additionally, the rotational error can be determined and corrected, eliminating the d term. With the coefficients known or corrected for, the wavelength mapping equation becomes

$$\lambda(n) = 1.1933n + 106.5198. \tag{2.2.2}$$

The mapping equation is shown to be only a function of linear dispersion and spectral offset with units of nanometers (nm).

2.3 Noise Characterization

All data acquired through an imaging system is subject to noise. The term noise is used in a very general sense and accounts for many different types of noise present in the system. Generally, noise present in the system manifests itself in two main types: temporal noise and pattern noise. Temporal noise is the frame-to-frame changes incurred by a pixel while viewing a uniform source[11], while pattern noise is a spatial pattern that does not change from frame-to-frame[12]. One main noise present in this system consists of both temporal and pattern noise and is referred to as dark current. Dark current is due to the thermal output of the imaging system itself, so care is taken to keep the system as cooled as possible. The pattern noise can be removed by the use of an integrating sphere and an averaging process. A Labsphere USS-600 integrating sphere[13] provides uniform illumination and 1500 frames are collected by the imaging system. The mean dark current can be calculated and subtracted from this data, which removes the pattern noise from the system. The rest of the noise is temporal noise, which can only be attenuated, not removed, through the averaging process.

With the major source of noise reduced as much as possible, a signal-to-noise ratio (SNR) for each pixel in the FPA can be determined. The 1500 sampled frames of data

collected using the integrating sphere are broken down on a pixel-by-pixel basis into signal, s, and system noise, η_{SYS}. For a given pixel, they are

$$s = \hat{\mu}(m,n) \tag{2.3.1}$$

and

$$\eta_{SYS} = \hat{\sigma}(m,n) \tag{2.3.2}$$

where $\hat{\mu}(m,n)$ represents the sample mean and $\hat{\sigma}(m,n)$ represents the sample standard deviation. The SNR for a given pixel is defined as

$$SNR = \frac{s}{\eta_{SYS}}. \tag{2.3.3}$$

It is then compared to a threshold to determine which pixels provide reliable data. The SNR for a single sample row of the FPA is shown in Figure 2.3.1.

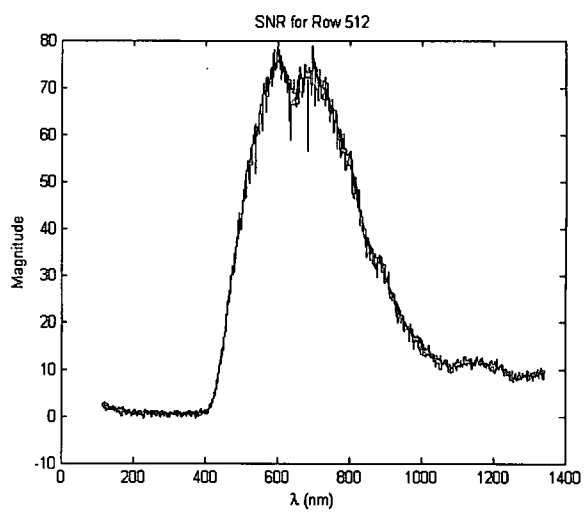

Figure 2.3.1: SNR across Row 512 of the FPA.

The SNR threshold is chosen to be 18, and as can be seen from Figure 2.3.1 portions of the FPA fall below the SNR ratio making them unusable. The data set is next spectrally filtered using a low-pass finite impulse response (FIR) filter with a Kaiser window to increase the

SNR. The overall spectral range of the FPA is found to be approximately 450nm to 900 nm. The useable range of rows on the FPA that meet the SNR threshold is found to be 1 to 800.

2.4 Radiometric Calibration

Once the signal reaches the FPA, it is converted to a digital number. It is often difficult to determine illumination levels from this digital number alone, so it is useful to convert it to spectral radiance. Spectral radiance is similar to radiance, or the power per unit projected area per unit solid angle of the radiation, but with a wavelength dependency. The process of converting the digital number outputted by the imaging system to a spectral radiance value is known as absolute radiometric calibration. Ideally, a single radiometric calibration could be performed on the imaging system and used throughout its lifecycle. However, due to the non-uniformity of the pixel response, in practice a radiometric calibration must be performed after every data collection.

A two-point calibration is a common radiometric calibration procedure used when the response of the FPA is fairly linear. The non-linear response error of the FPA is found to be less than 4% in this case, and thus meets the linearity condition for the purposes of this study. The two-point radiometric calibration typically uses a light frame and a dark frame to produce a gain and an offset for each digital number produced by the FPA. The spectral radiance for each FPA pixel is then computed using

$$L\ (m,n) = a(m,n)D(m,n) + b(m,n) \tag{2.4.1}$$

where a is the gain, b is the offset and D is the digital number. In general, the gain and offset are determined by

$$a(m,n) = \frac{L_L(m,n) - L_D(m,n)}{D_L(m,n) - D_D(m,n)} \tag{2.4.2}$$

and

$$b(m,n) = \frac{L_D(m,n)D_L(m,n) - L_L(m,n)D_D(m,n)}{D_L(m,n) - D_D(m,n)} \tag{2.4.3}$$

where L_D is the spectral radiance of the dark frame, L_L is the spectral radiance of the light frame, D_D is the raw digital number of the dark frame and D_L is the raw digital number of the light frame.

A specific type of two-point radiometric calibration implemented for here requires the collection of a light frame that is at or near the saturation point of the FPA and a completely dark frame. The integrating sphere is once again used as the source and 100 measurements are taken and averaged to produce a light frame. The dark frame is acquired by placing the lens cap over the lens and collecting 100 frames. The dark frame is ideally zero; however, in practice it will have some value greater than zero. This value is the dark current of the system and is due to the heat produced by the imaging system. In the radiometric calibration performed in this study, the dark current is in itself the offset, or *b* term. The dark current offset is then subtracted from the light frame. The offset is eliminated and the gain simplifies to

$$a(m,n) = \frac{L_L(m,n)}{D_{L-D}(m,n)} \tag{2.4.4}$$

where D_{L-D} represents the light frame with the dark current removed.

2.5 Second Order Correction

An inherent issue associated with diffraction gratings is multiple order diffraction. Diffraction gratings naturally disperse incident light into multiple orders with varying power. In general, diffraction gratings are described by the equation

$$a(\sin\theta_i + \sin\theta_m) = m\lambda, \qquad m = 0, \pm 1, \pm 2, \ldots \tag{2.5.1}$$

where θ_i is the incident angle, θ_m is the diffracted angle for a specific order m and a is a grating period constant. The problem of multiple order diffraction presents itself when the various diffraction orders overlap on the FPA. For example, in our imaging system the second order begins to overlap the first order at 800nm. Due to this overlap, 800nm of the first diffraction order and 400nm of the second diffraction order are both present at the same point on the FPA. A third order is also present, but it begins at 1200nm which is outside of the spectral range of the imaging system allowing it to be ignored. The grating used in this imaging system is known as a blazed grating, which is designed to place maximum power into the first diffraction order[14]. Other orders do still exist, however, and must be dealt with.

The simplest approach to counter the effects of multiple order diffraction is to place an order sorting filter into the system to block the higher orders from reaching the FPA. However, our imaging system does not have any type of order sorting filter so the problem must be dealt with in data processing phase. Since only the first diffraction order is present from 400nm-800nm, that data may remain unchanged. The data in the spectral range from 800nm-1000nm, however, requires the removal of the second order effects. This is done by determining the relationship between the efficiency (power) of the first and second orders. Due to the non-uniform response of the FPA, this must be done by finding the second order

gain for each affected pixel. After finding these gains, the second order correction is described by

$$D_C(m,n) = \begin{cases} D(m,n) & \lambda < 800nm \\ D(m,n) - \alpha(m',n')D(m',n') & \lambda \geq 800nm \end{cases} \tag{2.5.2}$$

where D_C is the corrected pixel value, α is the second order gain term and the prime notation indicates the locations of the pixels producing the second order gain. For a detailed insight into the process of determining the second order gain term, α, please see the work of Meola[15].

2.6 Data Collection

The goal of the data collection process is to collect images of a scene portraying both seasonal changes over many months and illumination changes during a diurnal period. The data used in this study is part of a larger data collection at Wright Patterson Air Force Base in Dayton, OH from August 2005 to May 2006. Over 100 data collections are made with variations in target content, vegetation conditions, solar position and time of day. One limitati0on of the data collection process is the need for the scene to be highly illuminated; collections can only occur on sunny days with limited cloud cover. This issue is related to the low overall SNR of the imaging system. In this experimentation, thirty-six seasonal data sets with a fixed solar azimuth position of 180 degrees and twelve diurnal data sets captured between 0800 and 1900 hrs are used. Solar position information for the specific coordinates of the scene is gathered from the National Oceanic and Atmospheric Administration (NOAA) website[16].

The scene consists of four painted aluminum panels, which are black, green, beige and silver in color, respectively, as shown in Figure 2.6.1. These panels are designed to act as man-made anomalies and are placed in a field with a naturally vegetative background.

The black panel is painted with off-the-shelf black spray paint, the green and beige are painted with Chemical Agent Resistant Coating (CARC) and the silver panel is coated with fire sprayed aluminum. The panels are mounted on a platform capable of tilting; however for this study the tilt and orientation of the panels remains constant. The panels face northeast toward a building located across the street from the scene. The hyperspectral imaging system is located on the fifth story of this building. The scene's orientation is shown in Figure 2.6.2.

Figure 2.6.1: Scene Setup

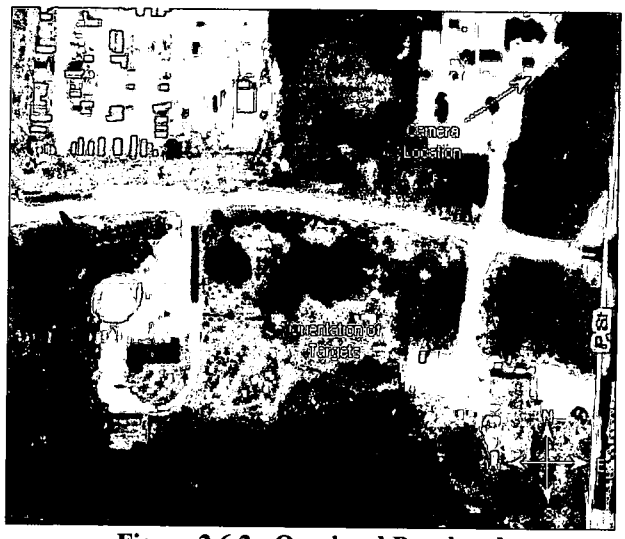

Figure 2.6.2: Overhead Panel and Camera Locations

The diurnal period represents the changes in the scene over a period from 0800-1900 hrs on May 8, 2006. This collection is designed to study the effects of solar position and solar illumination on anomaly detection performance. During this collection it is assumed that the vegetation in the background experiences only very minor changes that do not adversely influence anomaly detection performance. Major changes in illumination take place between 1700-1900 hrs. During this period, the sun begins to set behind the target panels in the west causing shadowing effects on the targets and lower overall illumination on the scene. This is most evident when viewing the black and green panels in Figure 2.6.6. Figures 2.6.3-2.6.6 show example color images of scene changes during the diurnal period.

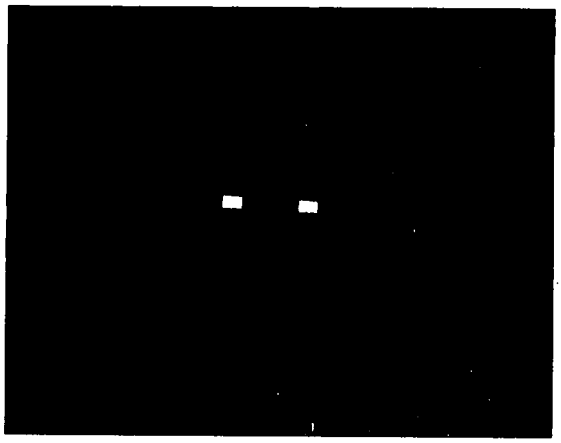

Figure 2.6.3: May 8, 2006 0800 hrs

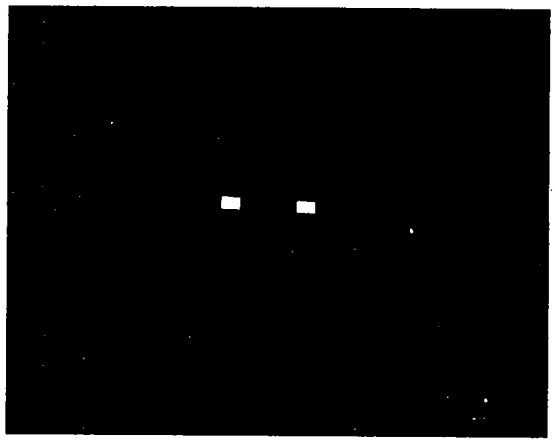

Figure 2.6.4: May 8, 2006 1200 hrs

Figure 2.6.5: May 8, 2006 1600 hrs

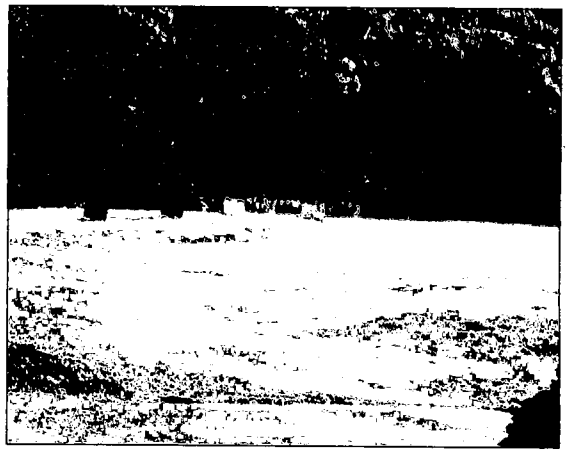

Figure 2.6.6: May 8, 2006 1900 hrs

During the seasonal period, from late August 2005 until late April 2006, the scene undergoes a wide variety of natural changes. The leaves on the trees senesce and eventually die, providing a naturally changing background. Beginning in mid-October 2005 and continuing until early March 2006 distinct shadowing features enter the scene, eventually engulfing the target panels. Additionally, during the December data collections a light snow cover is present on the natural elements of the scene, but not the targets. The overall goal of the anomaly detectors is to ignore these natural changes and only detect the target panels as anomalies. The detection performance results are explained in Chapter 5. Figures 2.6.7-

2.6.10 show example color images of scene changes during the seasonal period. Color images for all of the data used in this study are available in Appendix A.

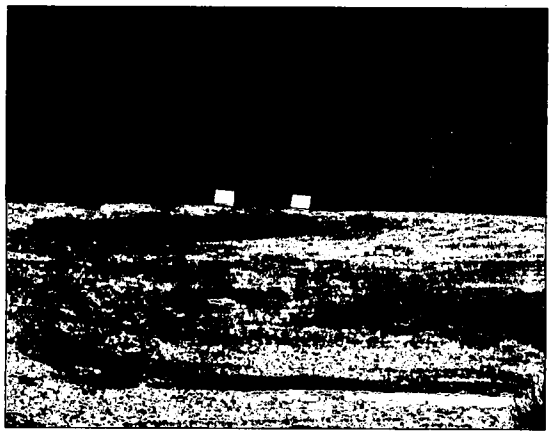

Figure 2.6.7: Aug 25, 2005

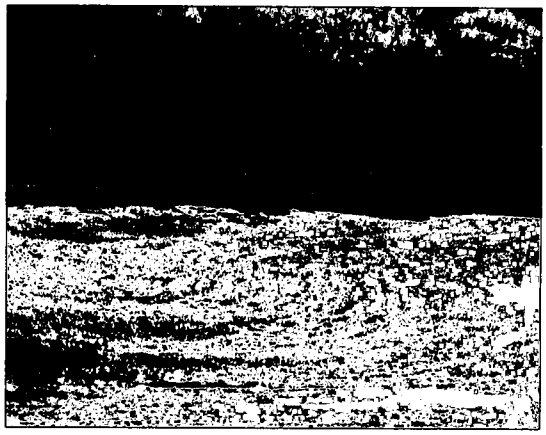

Figure 2.6.8: Oct 18, 2005

Figure 2.6.9: Dec 20, 2005

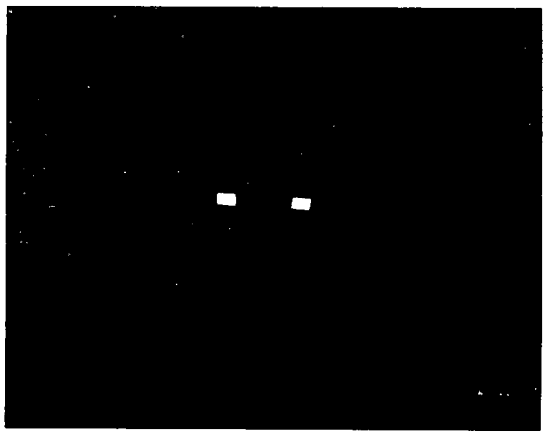

Figure 2.6.10: April 18, 2006

2.7 Principal Component Transform

The file size of each data set collected from the hyperspectral imaging system is on the order of 2 GB. This large data size makes the data difficult to transfer and extremely computationally intensive to process. Thus, there is a need to perform some sort of dimensionality reduction on the data sets. Dimensionality reduction methods typically result in some loss of information, however, it is possible to choose a method that keeps this

information loss to a minimum. One common dimensionality reduction technique often used in conjunction with hyperspectral data is known as the principal component (PC) transform. This approach is useful because hyperspectral data is over-sampled in the spectral dimension. Each spectral band in hyperspectral imagery often shares some degree of correlation with other bands. This correlation signifies that two or more bands contain some duplicated spectral information. Correlation manifests itself in the off-diagonal values of the covariance matrix of the hyperspectral data. The PC transform eliminates correlation between bands by diagonalizing the covariance matrix[17]. During this process, a new coordinate system for the data, known as PC space, is designated. The eigenvalues and eigenvectors of the covariance matrix can now be used to represent the overall variance found in each of the PC bands.

A plot of the eigenvalues for a sample case of the hyperspectral data collected in this study is shown on a logarithmic scale in Figure 2.7.1.

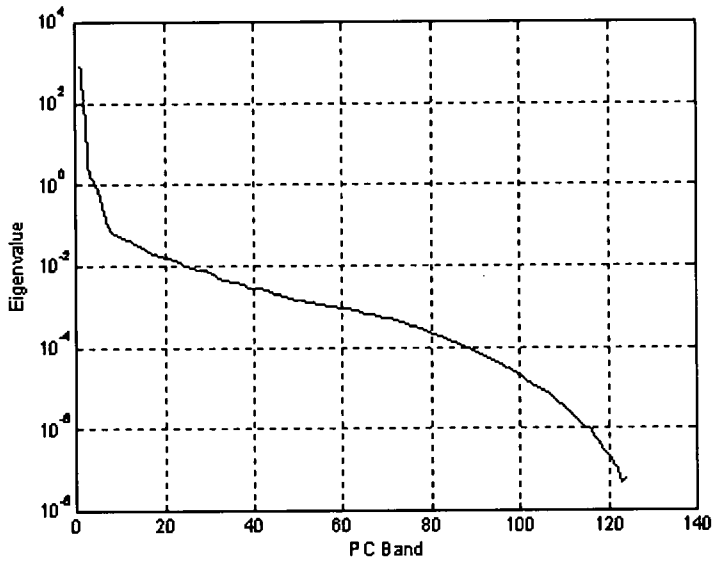

Figure 2.7.1: Eigenvalues for a sample hyperspectral covariance matrix. Notice the large drop off in eigenvalues after only a few bands.

The figure shows that only a few PC bands have large eigenvalues, while the majority of the bands have very small eigenvalues. This signifies that the vast majority of the variance in the data is contained in only a few bands. In fact, 99.4% of the variance in the sample hyperspectral data is contained in the first 10 PC bands. The rest of the PC bands contain mostly noise and can be discarded with some loss of useful information. In this study, the data is reduced by reserving the 10 PC bands with the largest eigenvalues for further analysis and discarding the rest. These 10 PC bands provide a very significant reduction in computational complexity during the data analysis phase. Figures 2.7.2 and 2.7.3 show the differences in the information contained in PC bands 1 and 10. While PC band 10 is retained for further analysis, it is clear from the figure that it contains much less spatial structure than PC band 1, a drop-off suggested by the eigenvalues in Figure 2.7.1.

Figure 2.7.2: PC Band 1

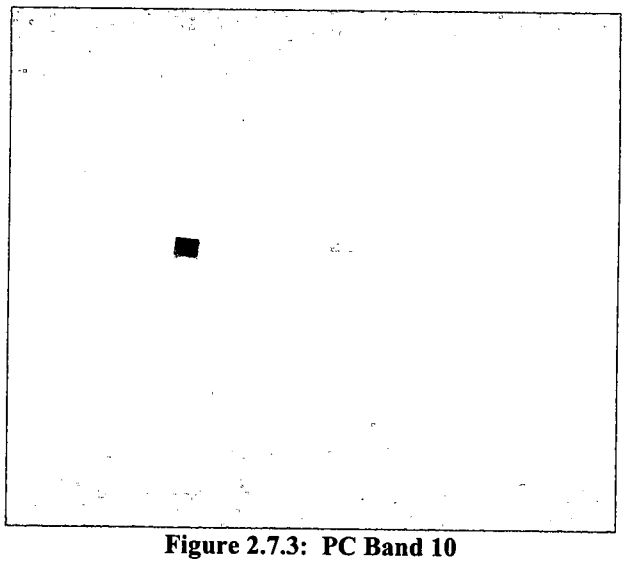

Figure 2.7.3: PC Band 10

CHAPTER 3

Hyperspectral Anomaly Detection Theory

This chapter describes how the background pixels of the hyperspectral data are statistically modeled and explains the theory associated with the anomaly detection algorithms used in this study.

3.1 Hyperspectral Data Modeling

To begin, a hyperpixel is defined as a vector corresponding to all the spectral measurements for a single pixel denoted as

$$\mathbf{z} = [z_1, z_2, \ldots, z_N]^T, \tag{3.1.1}$$

where N is the number of hyperspectral bands present in the image. Anomaly detectors are generally based on a distance metric to a known background class. Some are derived from a likelihood ratio test (LRT) given by

$$L(\mathbf{z}) = \frac{p_1(\mathbf{z})}{p_0(\mathbf{z})} > T, \tag{3.1.2}$$

where p_0 is the assumption that only background and no target is present in the image, p_1 is the case in which a target is present and T is some threshold. The discriminant function, also known as the decision statistic, for the LRT can be given as

$$d(\mathbf{z}) = \log(p_1(\mathbf{z})) - \log(p_0(\mathbf{z})). \tag{3.1.3}$$

In the anomaly detectors used here, p_1 is assumed constant over the observable range of data $\mathbf{z}$ since no target information is available. The background model, p_0, is handled based on

assumptions inherent to each anomaly detector, as described later in this chapter. Under these conditions, the decision statistic in (3.1.3) simplifies to

$$d(\mathbf{z}) = -\log(p_0(\mathbf{z})). \tag{3.1.4}$$

The assumptions for the background model can be broken into three basic types: global, locally adaptive, and cluster-based. Global models process the entire hyperspectral data set as a single entity, calculating a single set of background statistics assumed to be Gaussian in nature. An extension of the global model assumes the background consists of a superposition of several Gaussians. Locally adaptive methods calculate scene statistics utilizing a sliding window. Cluster-based models attempt to break the image pixels into clusters, or classes, based on similarities between pixels and compute background statistics for each cluster. The cluster-based models do not necessarily assume the background is strictly Gaussian.

The background models associated with each anomaly detection method addressed in this study are computed in a statistical fashion. The main statistics of interest are the mean vector and the covariance matrix. The mean vector, $\mathbf{m}_z$, and the covariance matrix, $\mathbf{C}_z$, are given as

$$\mathbf{m}_z = \frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i \tag{3.1.5}$$

and

$$\mathbf{C}_z = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{z}_i - \mathbf{m}_z)(\mathbf{z}_i - \mathbf{m}_z)^T, \tag{3.1.6}$$

where N is the vector length and i is the pixel index. In practice, the statistics are sample estimates computed from background training data in which no target is present. If the targets are sufficiently small, the statistics can be estimated from the full scene with some

loss in performance due to target contamination. The mean vector and covariance matrix are valid as given in (3.1.5) and (3.1.6) for the simple global case. In the locally adaptive case a mean and covariance are computed for each image pixel based on the pixels surrounding it. In the cluster-based case a mean and covariance are developed for each individual class.

Two separate types of clustering methods are implemented in this study. The first is a fairly simplistic method known a k-means clustering. The first step in the k-means algorithm is to initialize K classes. Image pixels are assigned to the class based on the smallest Euclidean Distance. Euclidean distance is given by

$$d_k = \left\| \mathbf{z}_i - \mathbf{m}_{\mathbf{z}|k} \right\|_2, \tag{3.1.7}$$

where k=1,...,K. Once this is complete, the statistics for each class are recomputed and the pixels are once again assigned to classes based on (3.1.7) with updated statistics. This process is continued until convergence is reached and results in a hard partition where each image pixel is a member of only one class.

The second clustering method is known as Stochastic Expectation Maximization (SEM)[18,19]. In this method, the K classes are initialized using global image statistics. A *posteriori* probability is then estimated, given by

$$P(k \mid \mathbf{z}) = \frac{p_0(\mathbf{z} \mid k) P_{\mathbf{z}}(k)}{\sum_{k=1}^{M} p_0(\mathbf{z} \mid k) P_{\mathbf{z}}(k)}, \tag{3.1.8}$$

where $P_{\mathbf{z}}(k)$ is the probability estimated from the data. Image pixels are assigned to a class in a Monte Carlo nature based on the *posteriori* probability. Updated class statistics are then estimated and the process is repeated until convergence. Following convergence, a final classification is made for each pixel based on the maximum *posteriori* probability. The SEM algorithm is more computationally complex than the k-means algorithm, but can be useful

due to the nature of it softer partition of classes. This soft partition allows for overlapping classes, in which an image pixel may belong to two or more classes.

An example of the differences between k-means and SEM clustering can be seen in Figures 3.1.1 and 3.1.2. Each method uses the same sample data, with 4 classes and 5 iterations.

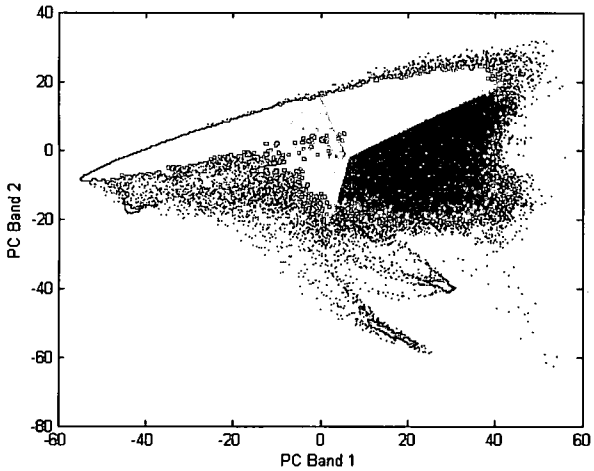

Figure 3.1.1: K-means clustering

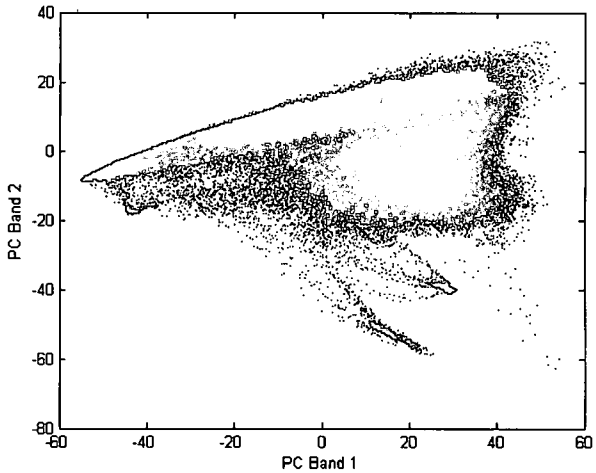

Figure 3.1.2: SEM Clustering

3.2 Mahalanobis Distance Anomaly Detection

The Mahalanobis distance (M-dist) anomaly detector[20] assumes the background of the entire hyperspectral image can be well represented by a single Gaussian pdf. The likelihood function for the background is given by

$$p_0(\mathbf{z}) = \frac{1}{\sqrt{(2\pi)^2 |\mathbf{C}_z|}} \exp\left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{m}_z)^T \mathbf{C}_z^{-1} (\mathbf{z} - \mathbf{m}_z) \right\}, \tag{3.2.1}$$

where $\mathbf{m}_z$ is the global mean and $\mathbf{C}_z$ is the global covariance. The decision statistic for the M-dist anomaly detector comes directly from (3.1.4) when (3.2.1) is used, and is given by

$$d_M(\mathbf{z}) = (\mathbf{z} - \mathbf{m}_z)^T \mathbf{C}_z^{-1} (\mathbf{z} - \mathbf{m}_z). \tag{3.2.2}$$

In (3.2.2), the pixels with the largest M-distance represent the most anomalous pixels. Figure 3.2.1 shows the distribution for a two band, simulated Gaussian distribution of image pixels. The pixels are represented by the blue dots and are assumed to have a mean, $\mathbf{m}_z$, which is located at the center of the pixel distribution. The contour superimposed on the plot represents the isocontours of the decision statistic found in (3.2.2). Pixels near the mean in the blue regions of the contour are considered to be part of the Gaussian background model. Pixels located further from the mean in the red regions of the contour would be considered increasingly anomalous.

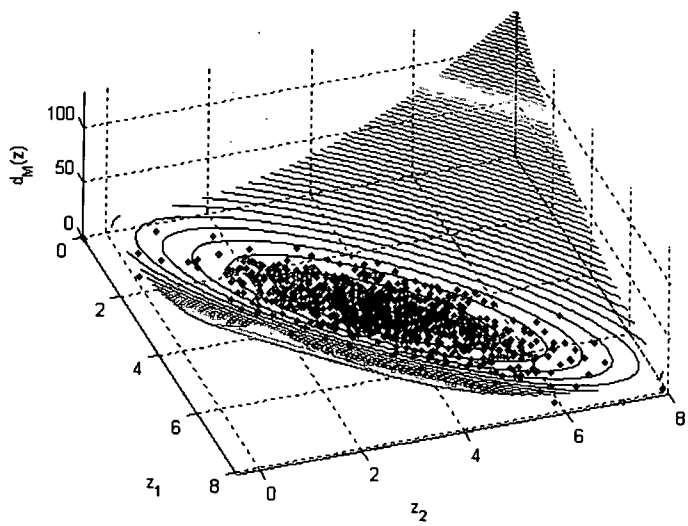

Figure 3.2.1: Scatter plot for a single Gaussian background model. The dots represent pixels in a simulated two band Gaussian distribution. The surface height is the M-dist detection statistic for an observation at $z=[z_1,z_2]$. The height is shown by the color coded contour. Pixels located close to the global mean (blue regions) are classified as background pixels, while pixels falling far from the global mean (yellow, orange and red regions) are classified as anomalous.

3.3 Gaussian Mixture Model Anomaly Detection

The Gaussian mixture model (GMM) anomaly detector[20] is an extension of the M-dist anomaly detector. The GMM model uses a superposition of K Gaussians, each with its own mean and covariance, as the background likelihood function. In remote sensing applications it is often appropriate to use such a model since the scene is likely to contain more than one type of background. The GMM background likelihood function is given by

$$p_0(\mathbf{z}) = \sum_{k=1}^{K} P_z(k) \frac{1}{\sqrt{(2\pi)^2 |\mathbf{C}_z(k)|}} \exp\left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{m}_z(k))^T \mathbf{C}_z^{-1}(k) (\mathbf{z} - \mathbf{m}_z(k)) \right\}, \qquad (3.3.1)$$

where $\mathbf{m}_z(k)$, $\mathbf{C}_z(k)$, and $P_z(k)$ are the mean, covariance and probability, respectively, of each of the k=1,2,...,K Gaussians. These model parameters are estimated from the data using SEM. The decision statistic for the GMM anomaly detector uses a minus log likelihood, given by

$$d_{GMM}(\mathbf{z}) = -\log\left\{ \sum_{k=1}^{K} P_z(k) \frac{1}{\sqrt{(2\pi)^2 |\mathbf{C}_z(k)|}} \exp\left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{m}_z(k))^T \mathbf{C}_z^{-1}(k) (\mathbf{z} - \mathbf{m}_z(k)) \right\} \right\}. \qquad (3.3.2)$$

This decision statistic is shown in Figure 3.3.1 for a similar scenario as shown in Figure 3.2.1, except that the background pixels are distributed among two Gaussian clusters. The method produces a smooth increasing decision statistic moving away from the center of the Gaussian clusters.

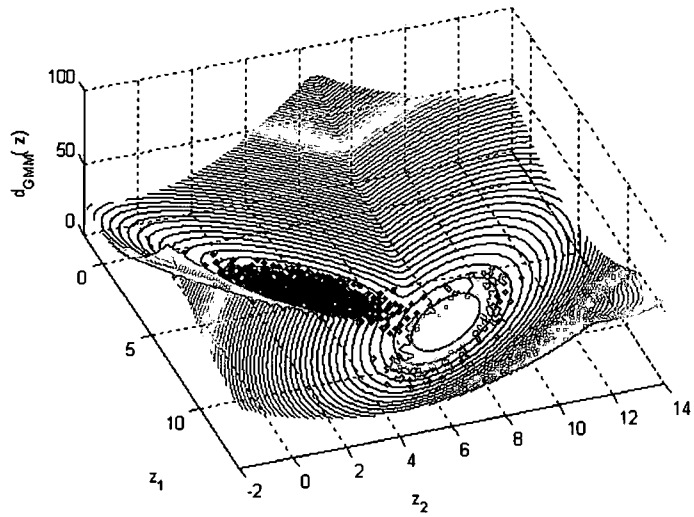

Figure 3.3.1: Scatter plots for a two class GMM. The surface height is the GMM detection statistic for an observation at $z=[z_1,z_2]$. Notice how the detection statistic rises as z moves away from the clusters.

3.4 Reed-Xiaoli Anomaly Detection

The Reed-Xiaoli (RX) anomaly detector[21] is a locally adaptive method. The decision statistic is similar to the one given in (3.2.1); however, $\mathbf{m}_z$ and $\mathbf{C}_z$ are estimated locally. Here, these localized statistics are governed by two user controlled windows, the guard window and the background window. The guard window should be specified to be at least as large as the targets found in the scene. The size of the background window is somewhat subjective based on the scene in question, but should be larger than the guard window. A background window that is too small could cause problems computing $\mathbf{C}_z$, while a window that is too large essentially eliminates the locally adaptive nature of the detector. Here, the guard window is 81x81 pixels and the background window is 141x141 pixels. The statistics are computed from the pixels falling in between the guard and background windows. The relatively large nature of the targets in the scene used in this study makes the RX detector very computationally complex. This tendency is addressed in a later section.

3.5 Gaussian Mixture RX Anomaly Detection

The Gaussian mixture RX (GMRX) anomaly detector[20] also assumes that the background hyperpixels follow a Gaussian mixture model. The method requires a class *posteriori* probability, presented in (3.1.8). The decision statistic for the GMRX detector is given by

$$d_{GMRX}(\mathbf{z}) = \left(\mathbf{z} - \mathbf{m}_z(p(\mathbf{z}))\right)^T \mathbf{C}_z^{-1}(p(\mathbf{z}))\left(\mathbf{z} - \mathbf{m}_z(p(\mathbf{z}))\right), \tag{3.5.1}$$

where $\mathbf{m}_z(p(\mathbf{z}))$ and $\mathbf{C}_z(p(\mathbf{z}))$ are once again the class mean and class covariance, respectively, estimated using the SEM algorithm. The function $p(\mathbf{z})$ is a maximum a posterior (MAP) classifier of the form

$$p(\mathbf{z}) = \underset{k}{\arg\max}\, P(k \mid \mathbf{z}). \tag{3.5.2}$$

The decision statistic for the GMRX detector is shown in Figure 3.5.1. The thick blue line superimposed on the figure represents the MAP classifier decision boundary.

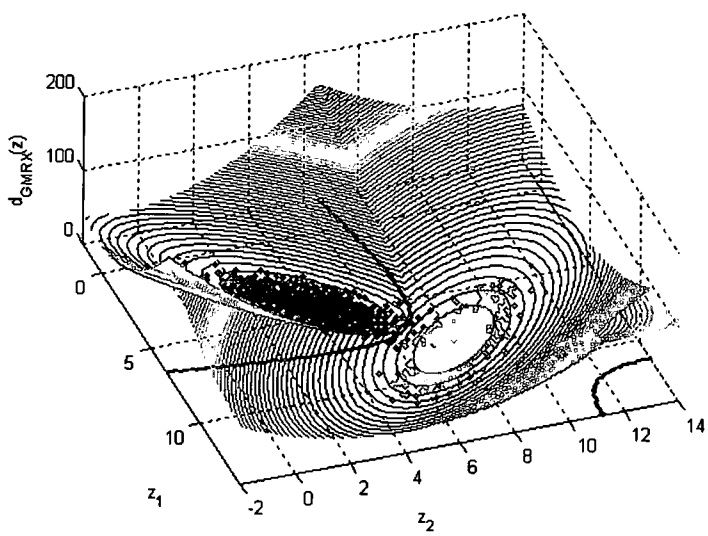

Figure 3.5.1: Scatter plots for a two class GMRX. The surface height is the GMRX detection statistic for an observation at $z=[z_1,z_2]$. The thick blue line represents MAP classifier decision boundary. Notice the smooth decision statistics, similar to the GMM method.

3.6 Cluster-Based Anomaly Detection

The cluster-based anomaly detector (CBAD)[22] is very similar to the GMRX detector. The difference is that CBAD uses a k-means clustering algorithm to estimate the component mean and covariance matrices and to segment the image. Unlike the previous methods, CBAD does not assume the background model is Gaussian in nature. The decision statistic is a class based M-distance given by

$$d_{CBAD}(\mathbf{z}) = \left(\mathbf{z} - \mathbf{m}_{\mathbf{z}}(q(\mathbf{z}))\right)^T \mathbf{C}_{\mathbf{z}}^{-1}(q(\mathbf{z}))\left(\mathbf{z} - \mathbf{m}_{\mathbf{z}}(q(\mathbf{z}))\right), \tag{3.6.1}$$

where $\mathbf{m}_{\mathbf{z}}(q(\mathbf{z}))$ represents a class mean and $\mathbf{C}_{\mathbf{z}}(q(\mathbf{z}))$ a class covariance. Instead of using a MAP classifier, CBAD determines a hyperpixel's class membership based on the minimum Euclidean distance measurement where

$$q(\mathbf{z}) : R^P \rightarrow \{1, 2, \ldots, K\} \tag{3.6.2}$$

is the partition function given by

$$q(\mathbf{z}) = \underset{k}{\arg\min} \left\|\mathbf{z} - \mathbf{m}_k(\mathbf{z})\right\|^2 . \tag{3.6.3}$$

Figure 3.6.1 shows the decision statistic for a two class CBAD anomaly detector. Similar to previous methods discussed, the further a pixel is located from the nearest class the more anomalous it is. However, due to the Euclidean distance measure in (3.6.3), there is a very strong discontinuity in the decision statistic between the clusters. This is denoted by the very sharp edge between the two classes in Figure 3.6.1.

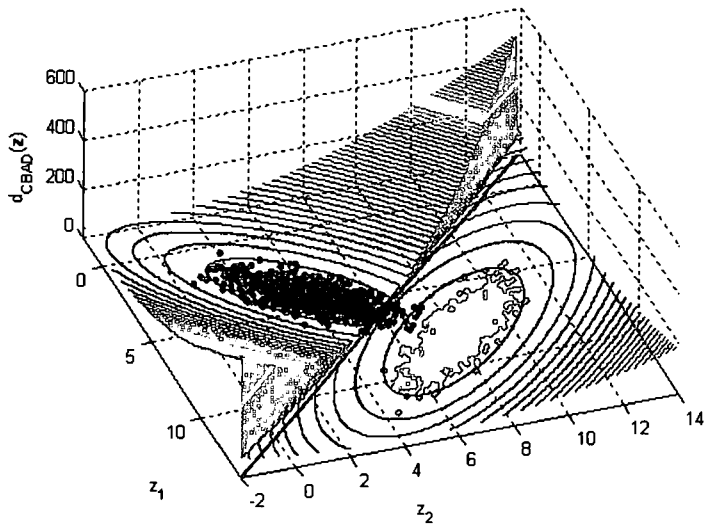

Figure 3.6.1: Scatter plots for a two class CBAD. The surface height is the CBAD detection statistic for an observation at $z=[z_1,z_2]$. Notice how the detection statistic has a discontinuity at the cluster boundary, denoted by the very sharp edge between clusters.

3.7 Fuzzy Cluster-Based Anomaly Detection

The fuzzy cluster-based anomaly detector (FCBAD) is a novel extension of the CBAD and GMRX algorithms in that it can include several possible membership functions. Like the CBAD method, FCBAD is not specifically linked to the assumption of a Gaussian background. The particular membership function used in this study is given as

$$p_k(\mathbf{z}) = \frac{1}{\left((\mathbf{z}-\mathbf{m}_z(k))^T \mathbf{C}_z^{-1}(k)(\mathbf{z}-\mathbf{m}_z(k))^T\right)^A + 1}, \tag{3.7.1}$$

where A is a tuning parameter and $\mathbf{m}_z(k)$ and $\mathbf{C}_z(k)$ are class mean and class covariance, respectively. For this research the tuning parameter is set to A=2. It should be noted that, among others, (3.5.2) or (3.6.3) can be substituted for (3.7.1), yielding the GMRX and CBAD detectors, respectively. The membership function is then used to compute the decision statistic,

$$d_{FCBAD}(\mathbf{z}) = \sum_{k=1}^{K} p_k(\mathbf{z})(\mathbf{z}-\mathbf{m}_z(k))^T \mathbf{C}_z^{-1}(k)(\mathbf{z}-\mathbf{m}_z(k)) . \tag{3.7.2}$$

As with the CBAD detector, the model parameters of the FCBAD detector are estimates using a k-means algorithm. FCBAD does not have a strict decision statistic like the CBAD or GMRX detectors, giving it more flexibility. Additionally, this method may have performance benefits for non-Gaussian data in which the background has heavy tails. Figure 3.7.1 shows the decision statistic for the two class FCBAD method. The blue lines represent the isocontours of the membership function given in (3.7.1) as it relates to the red cluster of pixels.

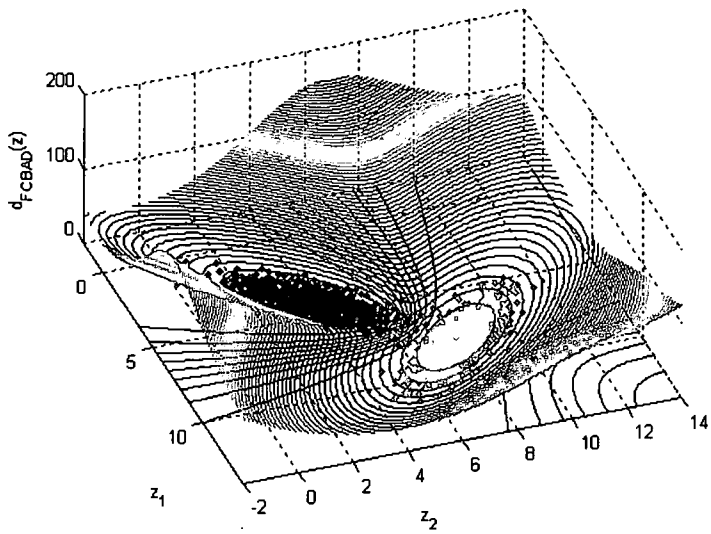

Figure 3.7.1: Scatter plots for a two class FCBAD. The surface height is the FCBAD detection statistic for an observation at $z=[z_1,z_2]$. Note the smooth decision statistic contour and the membership function (blue lines) related to the red class of pixels.

CHAPTER 4

Change Detection Theory

Change detection is an extension of anomaly detection using two similar data sets captured some time apart. The process uses a predictor and an anomaly detector to detect man-made changes between the data sets.

4.1 Change Detection Process

The two data sets used in change detection analysis are often referred to as the reference image (time 1), **x**, and the test image (time 2), **y**. The goal of change detection is to detect man-made changes present between the images. Man-made changes are those changes in the scene that involve the addition, removal, movement, or alteration of a target of interest. However, if the two scenes are captured some sizeable time interval apart, natural changes in the background of the scene will also be introduced. These natural changes may include illumination changes, shadowing and seasonal changes. It is desirable to eliminate or suppress these natural changes through a prediction process. The predictor generates a transformation from the time reference and test data sets. The transform is applied to create a predicted test image, which is then subtracted from the actual test image creating a residual error image. Ideally, the prediction and subtraction process removes the natural changes from the residual error image. The residual error image is then fed into an anomaly detector allowing for the extraction of man-made changes. The change detection process is shown in Figure 4.1.

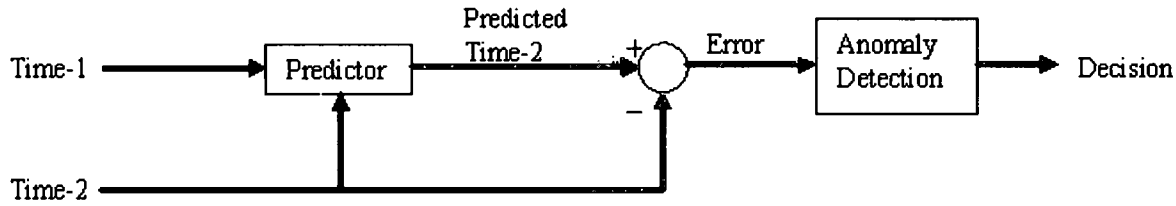

Figure 4.1.1: Change Detection Process. The main components of interest are a predictor and an anomaly detector.

4.2 Chronochrome Algorithm

The Chronochrome (CC) algorithm[23] is a prediction method used to create an estimated test image. The predicted test image can be expressed as

$$\hat{\mathbf{y}}_i = \mathbf{T}_{CC}\mathbf{x}_i + \mathbf{d}_{CC} \tag{4.2.1}$$

where $\mathbf{T}_{CC}$ is a linear gain matrix, $\mathbf{d}_{CC}$ is the offset vector, $\mathbf{x}_i$ is reference (time 1) data, and $\hat{\mathbf{y}}_i$ is the predicted test (time 2) data. In the chronochrome algorithm, the linear gain matrix and offset vector are defined as

$$\mathbf{T}_{CC} = \mathbf{C}_{\mathbf{yx}}\mathbf{C}_{\mathbf{x}}^{-1} \tag{4.2.2}$$

and

$$\mathbf{d}_{CC} = \mathbf{m}_{\mathbf{y}} - \mathbf{C}_{\mathbf{yx}}\mathbf{C}_{\mathbf{x}}^{-1}\mathbf{m}_{\mathbf{x}} \tag{4.2.3}$$

where $\mathbf{C}_{\mathbf{x}}$ is the covariance matrix of the reference image, $\mathbf{C}_{\mathbf{yx}}$ is the cross covariance, $\mathbf{m}_{\mathbf{x}}$ and $\mathbf{m}_{\mathbf{y}}$ are the mean vectors of the reference and test data, respectively. The cross covariance is a very important term in the CC algorithm and is defined as

$$\mathbf{C}_{\mathbf{yx}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{y}_i - \mathbf{m}_{\mathbf{y}})(\mathbf{x}_i - \mathbf{m}_{\mathbf{x}})'. \tag{4.2.4}$$

This term is dependant on the cross-covariance of the test data and reference data, and is extremely dependant on precise image registration.

The CC algorithm as described here is a global predictor. Another approach to prediction is a segmented or cluster-based predictor. This method uses a clustering algorithm, such as k-means or SEM, to create a clustered prediction of the test data. This method is well covered by Eismann, et.al[24].

4.3 Anomaly Detection

The final step of change detection involves a residual error statistic created from subtracting the test data from the predicted test data. This error statistic is given as

$$\boldsymbol{\varepsilon}_i = \hat{\mathbf{y}}_i - \mathbf{y}_i \tag{4.3.1}$$

where $\mathbf{y}_i$ is the test data. The error statistic is then fed into an anomaly detector. This step can incorporate a variety of anomaly detectors, including all of the detectors outlined in Chapter 3. Specifically, the M-dist, GMRX and CBAD anomaly detectors are applied in this change detection study.

CHAPTER 5

Experimental Results

This chapter explains the experimentation used to test anomaly detection performance. The experimental results for each anomaly detector's performance for both the diurnal and seasonal data sets are presented. Experimental results for the change detection study are also presented. Additionally, the overall processing time and computational complexity of each anomaly detection algorithm is discussed.

5.1 Diurnal Data

Twelve hyperspectral data sets collected at Wright-Patterson Air Force Base over the period of 0800-1900 hrs on May 8, 2006 are selected for this diurnal data set. Color image examples of selected diurnal data sets are shown in Figures 2.6.3-2.6.6. The overall diurnal period allows for an examination of the effects of a wide range of solar positions and solar illuminations on the scene. Three separate experiments are used to evaluate the performance of each of the six anomaly detection methods. These experiments are explained in the next three sections and experimental results are given for each.

5.1.1 Performance vs. Time of Day

The first experiment measures and compares the area under the curve (AUC) calculated from the receiver operating characteristic (ROC) curves for each hyperspectral

data set. Figure 5.1.1 shows the performance of all six anomaly detection methods. Two classes are used where applicable.

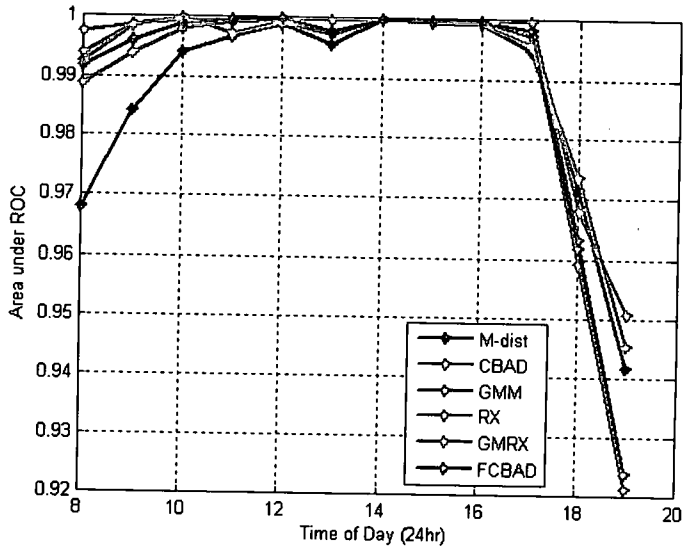

Figure 5.1.1: Performance vs. Time of Day – 0800-1900 hrs. Two classes are used where applicable.

All detectors perform well from 0800-1700 hrs and have similar AUC values, as can be seen in Figure 5.1.1. It is also noteworthy that all detectors have reduced performance between 1700-1900 hrs. This is due to illumination changes related to the solar angle with respect to the target panels. Between 1700-1900 hrs the sun slowly begins to set in the west, which is behind the northeast facing target panels. This causes a reduction in overall illumination on the panels and the scene in general, making anomaly detection more difficult.

Figure 5.1.2 displays the anomaly detection images for each of the methods for May 8, 2006 at 1400 hours. In these images brightness represents the decision statistic (i.e., bright pixels are more anomalous). All six methods are successfully able to detect and classify the four target panels as anomalies while raising few false alarms in the background. Figure 5.1.3 shows a portion of the ROC curves for each of the methods at 1400 hours. While CBAD has the best overall performance at this time, the overall difference between all six detection methods is minimal. Note the very small scaling on the x-axis of Figure 5.1.3.

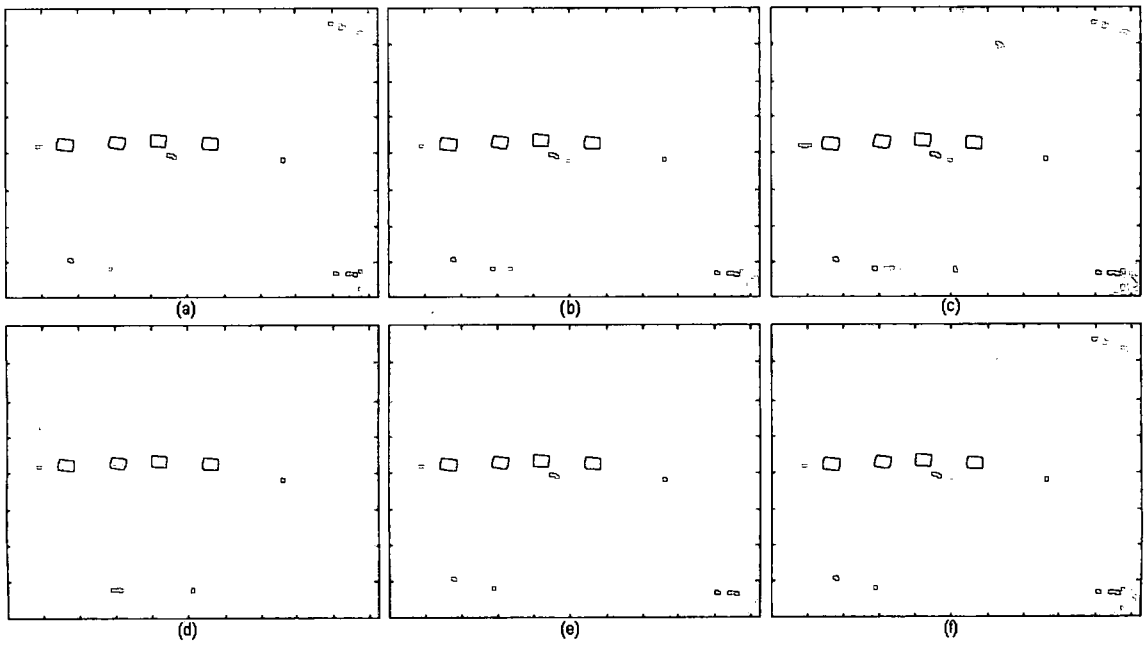

Figure 5.1.2: Anomaly Detection Images May 8, 2006 at 1400 hrs – (a) M-dist, (b) CBAD, (c) GMM, (d) RX, (e) GMRX, (f) FCBAD. All methods detect all four target panels as anomalous.

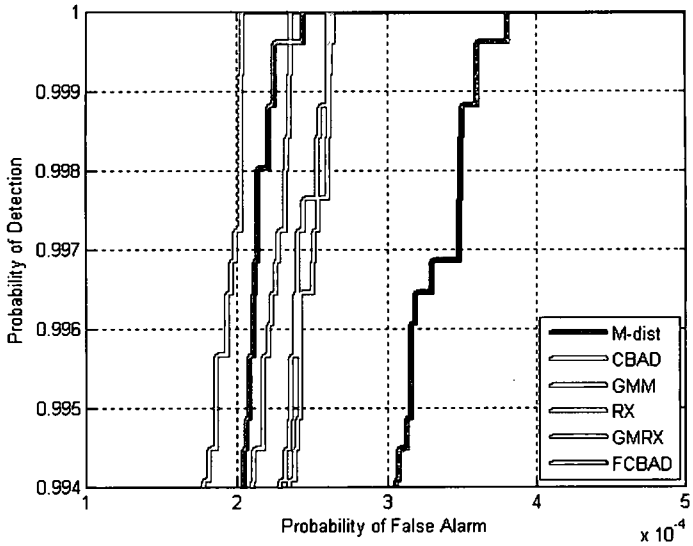

Figure 5.1.3: ROC performance for May 8, 2006 at 1400 hrs. Note the very small probability of false alarms (x-axis) and the very high probability of detection (y-axis).

5.1.2 Performance vs. Number of Classes

Let us now consider performance as a function of the number of classes used in the cluster-based methods (and superimposed K Gaussian methods GMM and GMRX, referenced as cluster-based methods for simplicity from here on) varied between two and ten. The AUC values for the twelve diurnal data sets are averaged for each method and number of classes used, as shown in Figure 5.1.4. This provides a way to analyze the general performance tendencies among the six different anomaly detection methods as well as allowing an analysis of the effect of the number of classes in the cluster-based methods. No changes are made to the M-distance and RX detectors. Their performance is given for comparison and is represented as a flat line.

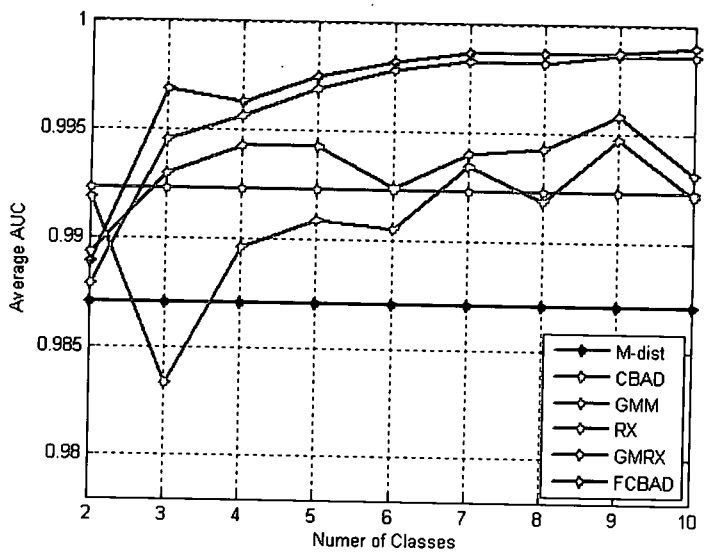

Figure 5.1.4: Performance vs. Number of Classes.

Figure 5.1.4 shows that the addition of more classes generally increases the overall performance of the cluster-based methods. The FCBAD detector has the overall best performance, claiming a small advantage over the CBAD detector. Both increase steadily in performance with the addition of more classes, until a point of diminishing returns around six or seven classes. This diminishing returns nature is likely due to the relative simplicity of the

target scene used in the data collection process. A more complicated and populated scene would have a more diverse set of pixel values and would likely support more classes. The GMM and GMRX methods also show some performance improvement with additional classes, but the gains are neither as large nor as consistent as with the CBAD and FCBAD methods. It is worth noting that the top performing methods, CBAD and FCBAD, use a relatively simple k-means clustering algorithm.

5.1.3 Performance with Target Contamination

In the previous two experiments, the statistics of the scene background are computed with the target panels completely masked out. This corresponds to a scenario where representative background training data is available and known to be free of targets. However, in many situations this type of training data is not available. In these situations it is sometimes necessary to use data where targets are present, provided that they are relatively small compared to the overall scene. In this experiment, the statistics of the background are degraded, or contaminated, by allowing a percentage of target panel pixels into the calculation. Target pixels are allowed in at increments of ten percent and when all of the target panel pixels are included they account for roughly 0.3% of the total image pixels. This represents a more realistic analysis of each detection algorithm's performance. Here, as in the second performance experiment, the AUC is averaged over all twelve data sets. The results are shown in Figure 5.1.5. The RX anomaly detector is not shown in this performance experiment since it uses windows designed to avoid target contamination.

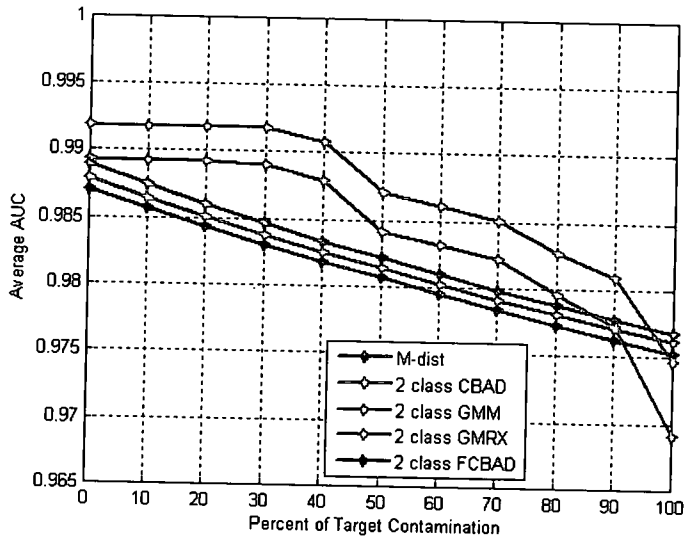

Figure 5.1.5: Performance with Target Contamination. Two classes are used in the clustering methods.

Figure 5.1.5 shows the performance of the anomaly detection methods with respect to the percentage of target contamination present. All of the clustering methods are shown for the simple two class case. As seen in Figure 5.1.5, the performance of all the methods is degraded as more target pixels are included. The performance of the M-distance, CBAD and FCBAD detectors fall at a fairly constant, linear rate throughout. The GMM and GMRX detectors exhibit less uniform behavior and some resistance to small amounts of target contamination; however, they are generally negatively affected by target contamination as well.

5.2 Seasonal Data

The seasonal data set consists of thirty six hyperspectral data sets collected at Wright-Patterson Air Force Base between August 24, 2005 and April 18, 2006. These data sets represent a wide range of seasonal conditions, including full illumination, distinct shadowing and light snow cover. All sets have a constant solar azimuth position fixed at 180 degrees. Example color images of selected seasonal data sets are shown in Figures 2.6.7-2.6.10. The

same three performance experiments used to analyze the diurnal data are also applied to the seasonal data.

5.2.1 Performance vs. Season

The AUC is computed for the thirty six seasonal data sets. The results are shown in Figure 5.2.1 and two classes are used in the clustering methods throughout section 5.2.1.

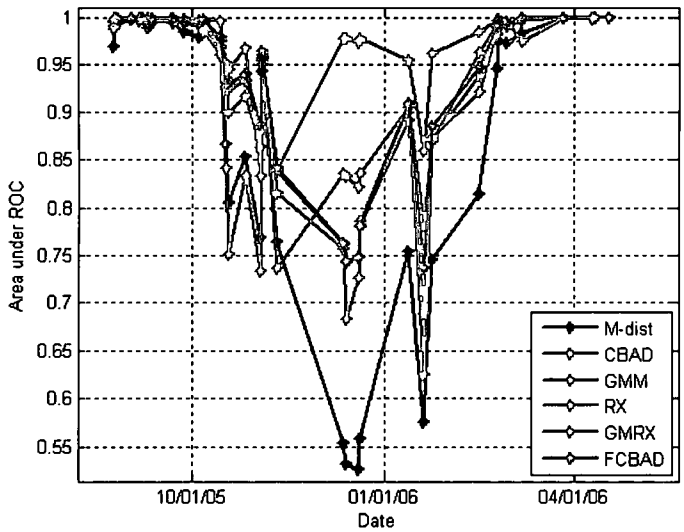

Figure 5.2.1: Performance vs. Season. Two clustering classes are used where applicable.

Clearly visible in Figure 5.2.1 are three distinct regions of performance. Two regions in which all the detection methods exhibit excellent performance are seen from the beginning of the collection (August 24, 2005) through mid-October 2005 and from March 2006 through the end of the collection on April 18, 2006. In the third region between mid-October 2005 and early March 2006 the performance of all detectors is highly degraded. This results from distinct shadowing features that enter the scene and engulf all four target panels. In addition, the background of the December data sets is covered in a light layer of snow (the target panels are not covered). Examples of the shadowing effects and snow cover are shown as color images in Figures 2.6.8 and 2.6.9. The anomaly detection images are shown in Figures

5.2.2 and 5.2.3 and the ROC curves of all six methods can be seen in Figures 5.2.4 and 5.2.5. Note the large difference in Probability of False Alarms (x-axis) between Figures 5.1.3, 5.2.4 and 5.2.5.

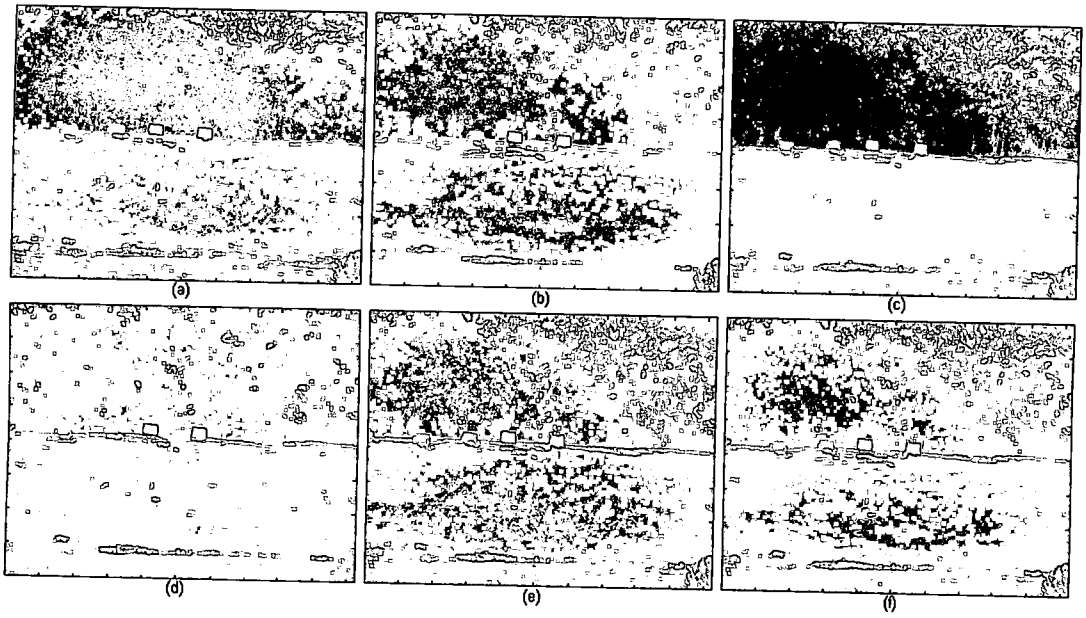

Figure 5.2.2: Anomaly Detection Images for October 18, 2005 – (a) M-distance, (b) CBAD, (c) GMM, (d) RX, (e) GMRX, (f) FCBAD. Notice all methods struggle to detect the black and green panels as anomalies.

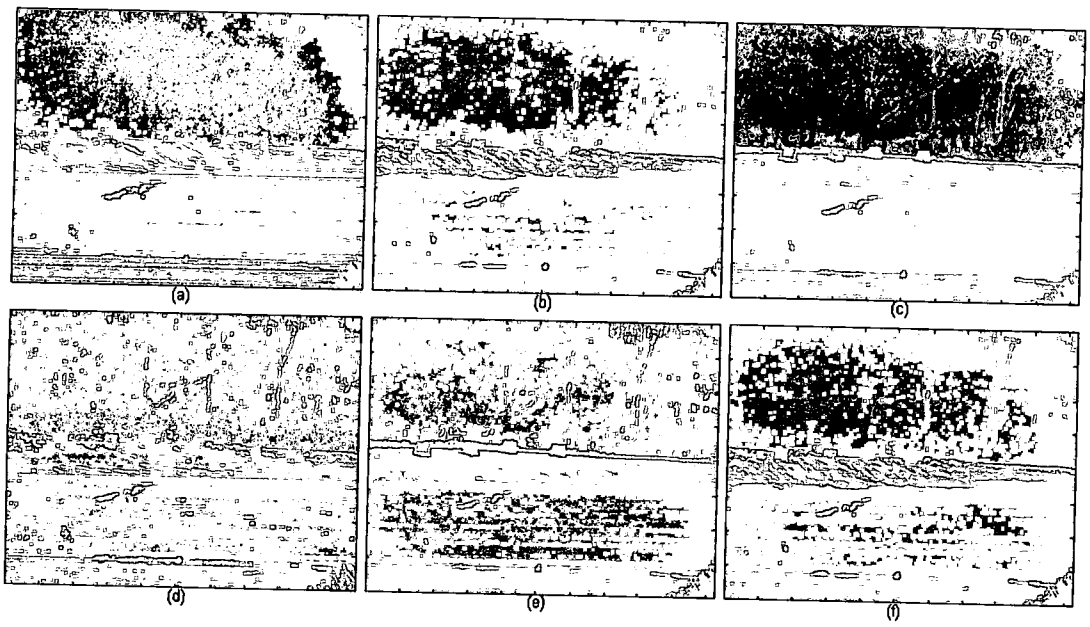

Figure 5.2.3: Anomaly Detection Images December 20, 2005 – (a) M-distance, (b) CBAD, (c) GMM, (d) RX, (e) GMRX, (f) FCBAD. With the exception of GMRX, all methods show poor detection performance for this date.

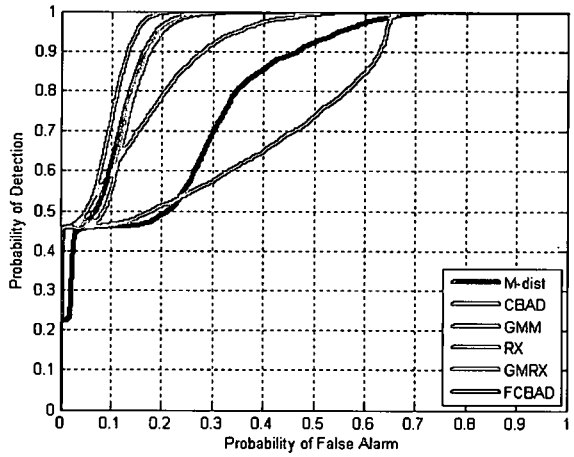

Figure 5.2.4: ROC performance for Oct 18, 2005

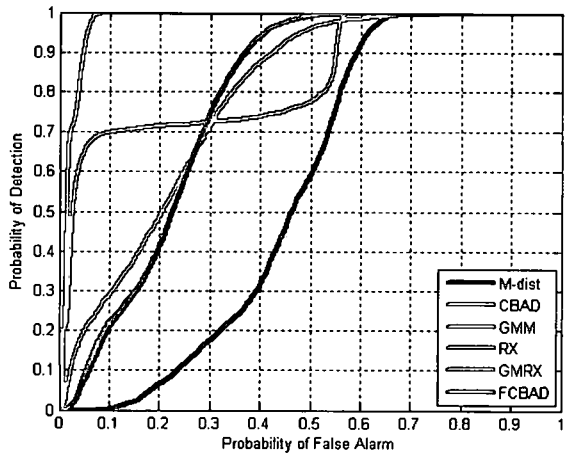

Figure 5.2.5: ROC performance for Dec 20, 2005

As can be seen from Figures 5.2.2-5.2.5, the detection performance of all six methods is seriously negatively impacted by the shadowing effects and snow cover in the scene. During the shadowed period, Figure 5.2.2 shows that the two left most targets, black and green respectively, are the most difficult for the detectors to properly classify as anomalies. There is also a very significant amount of false alarms raised in the background. This decreased performance is also seen in the ROC curve in Figure 5.2.4. As is shown in Figures 5.2.3 and 5.2.5, the performance of the methods degrades further in the presence of snow cover. In general, the methods fail to classify the target panels as anomalies. The exceptions here are the GMM and GMRX methods. The GMM method shows some ability to successfully operate under adverse conditions and is able to detect three target panels as seen in Figure 5.2.3 (c). The GMRX method exhibits good detection performance despite the shadowing and snow cover. Figure 5.2.3 (e) shows that the GMRX method is able to classify all four target panels as anomalies, despite the adverse conditions. This performance advantage is also seen in the ROC curve in Figure 5.2.5. It is worth noting that both the GMM and GMRX methods use stochastic expectation maximization as their clustering algorithms.

5.2.2 Performance vs. Number of Classes

The performance experimentation in this section is similar to that of section 5.1.2. In that previous section, the general behavior of the detection methods as a function of the number of classes is discovered for highly illuminated, non-shadowed data sets. Instead of repeating that analysis in this section, only the data sets that feature a highly shadowed scene are used. This allows a more thorough breakdown of each of detector's ability to classify anomalies in the presence of the adverse conditions. The results are shown in Figure 5.2.6.

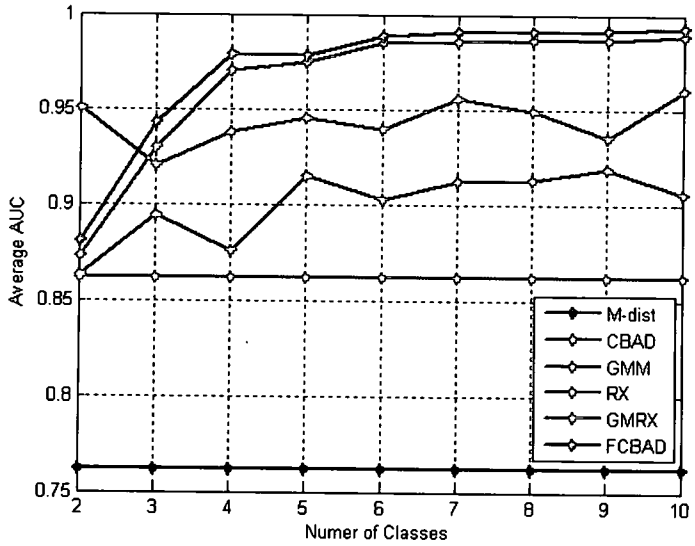

Figure 5.2.6: Performance vs. Number of Classes. The results for the seventeen shadowed data sets from mid October 2005 – early March 2006.

As seen in Figure 5.2.6, with the exception of the GMRX method which has near peak performance in the two class case, the overall performance of the cluster-based methods increases with the addition of more classes until a point of diminishing returns. This is similar to the behavior observed for the diurnal data sets in Figure 5.1.4. In Figure 5.2.6, however, the difference between the minimum and maximum achieved performance is much greater. Once again, the performance of the CBAD and FCBAD detectors are closely related. These methods have the best overall performance during the shadowed seasonal

period, with a diminishing returns behavior beginning at six classes. The GMM detector's performance also benefits from the addition of classes, although the performance gains are not as strong as with CBAD and FCBAD. Here, the GMM detector has optimum performance in the five class case. The GMRX method also exhibits similar behavior as in Figure 5.1.4. The GMRX method features very good performance with only two classes, but does not benefit appreciably from the addition of more classes. In the shadowed seasonal period the cluster-based methods significantly outperform the global and locally adaptive anomaly detection methods.

5.2.3 Performance with Target Contamination

The performance experiment used in this section is similar to the third performance experiment used in the diurnal case. Here, similar to section 5.2.2, only the shadowed data sets are used. The results are shown in Figure 5.2.7.

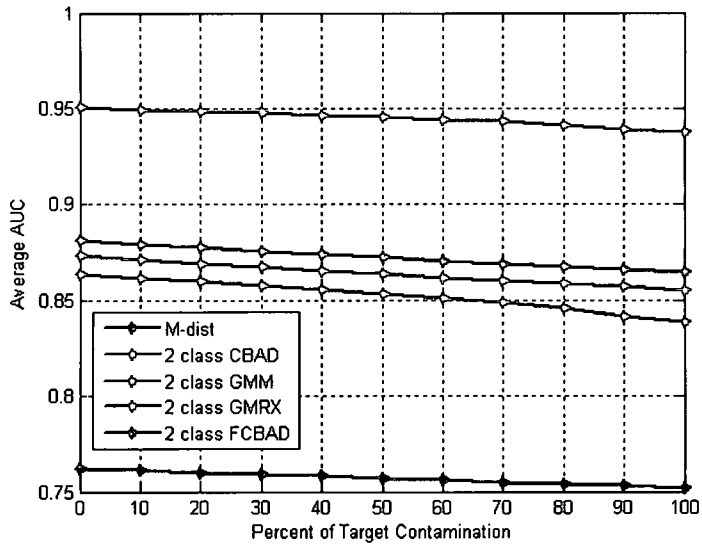

Figure 5.2.7: Performance with Target Contamination. The results are shown for the seventeen shadowed data sets from mid October 2005 – early March 2006.

The performance in Figure 5.2.7 is as expected, with all detectors having reduced performance as target pixel contamination increases. Unlike the somewhat varying nature seen for the GMM and GMRX detectors in the diurnal data in Figure 5.1.5, the relationship between performance and target contamination is fairly linear for all detectors.

5.3 Change Detection Performance

The experimentation conducted on change detection follows closely with the work of Eismann, et.al[24]. In that work, the idea of a segmented (cluster-based) prediction algorithm paired with an M-dist or RX anomaly detector is investigated. The cluster-based predictor can employ either the k-means or SEM clustering methods. In this study, that prediction-anomaly detection combination is compared to a change detection combination where the CC and cluster-based predictors are paired with cluster-based anomaly detectors. Six classes are used in cluster-based prediction and anomaly detection throughout this experiment. The prediction-anomaly detection combinations are outlined in Table 5.3.1.

Table 5.3.1: Change Detection Combinations

Predictor	Anomaly Detector
CC (global)	CBAD
CC (global)	GMRX
Segmented (K-means)	M-dist
Segmented (SEM)	M-dist
Segmented (K-means)	CBAD
Segmented (SEM)	GMRX

The reference and test images are chosen to be October 26, 2005 and October 14, 2005, respectively. It should be pointed out that these are in reverse chronological order, as only a limited number of appropriate test images are available. October 14 happens to fall just inside the time period where shadowing becomes a serious issue in the scene, as

described in previous sections. With the test image selected, it is then desired to choose an appropriately shadowed reference image that is collected a reasonable period of time from the test image. October 26 is judged to be a reasonable selection despite the reverse chronological order. The reference and test images are shown in Figures 5.3.1 and 5.3.2.

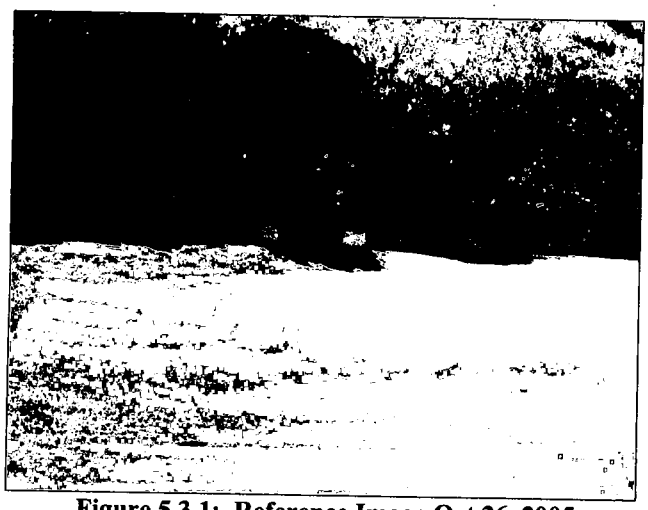

Figure 5.3.1: Reference Image Oct 26, 2005

Figure 5.3.2: Test Image Oct 14, 2005

Differing from previous experimentation, the four aluminum target panels are considered part of the background and are not desired to be detected as anomalies. Instead, two bundled green tarps (circled) are added to the test scene as shown in Figure 5.3.2. These are the targets we desire to identify through the change detection process while suppressing the natural changes from scene to scene. The change detection performance of each predictor-anomaly detector combination is shown in Figure 5.3.3. Background statistics for the scene are computed from the reference image and applied throughout the process, where applicable. No target masks are used in the statistical calculations or in the prediction process.

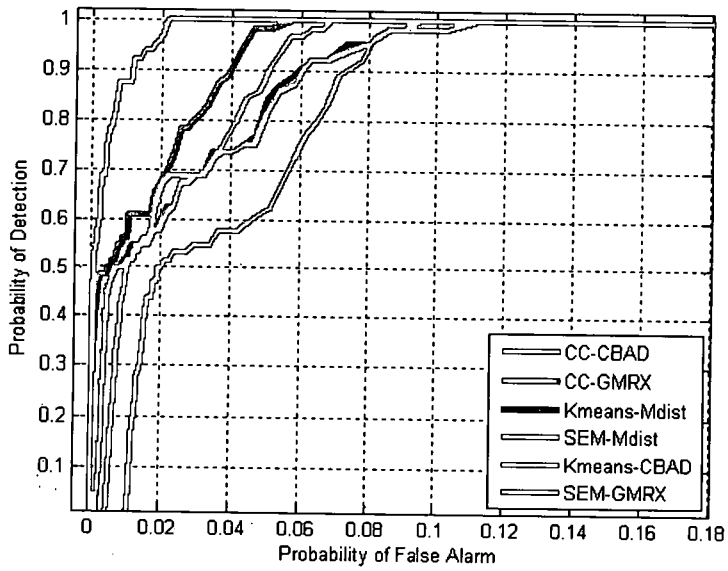

Figure 5.3.3: ROC Performance for Change Detection Combinations.

Figure 5.3.3 shows all predictior-detector combinations have relatively strong change detection performance. The combination of SEM prediction and GMRX anomaly detection has the best overall performance. This performance, coupled with the strong performance of the CC predictor and GMRX anomaly detector combination suggest the use of GMRX anomaly detection with statistics computed from the reference image is preferred for change detection.

5.4 Computational Complexity

In this section an attempt is made to quantify the computational complexity of each anomaly detection algorithm in terms of processing time. The computations are performed using Matlab Version 7.2.0.232 (R2006a) from The Mathworks[25]. Care is taken to limit the number of background processes running along with Matlab and to keep the workspace as consistent as possible during the entire process.

The computation is done for a single hyperspectral data set, August 25, 2005, which falls during the period of non-shadowed illumination. The processing time is computed for a varying number of classes where applicable and with no target contamination. The hyperspectral data set along with the appropriate target masks are preloaded into the Matlab workspace and are not included in the computation time of the algorithms. The k-means and SEM clustering algorithms are included in the computation time where applicable. The processing time for the cluster-based methods is directly related to the clustering algorithm used. The GMM and GMRX algorithms, which employ the more complicated SEM algorithm, generally require 2.5 to 3 times more processing time than the CBAD and FCBAD algorithms. The number of classes also impacts the processing time needed for each algorithm. As a general rule, the addition of new classes increases the algorithm's processing time by 10-20%. In the simple two class case, the CBAD and FCBAD detectors require 10.57 seconds and 12.39 seconds processing time, respectively. The two class GMM detector requires 29.23 seconds while the two class GMRX detector requires 29.31 seconds.

The M-dist anomaly detector has the least computational complexity and the shortest processing time, requiring only 2.07 seconds. However, as shown in previous sections, it only provides strong detection performance during conditions in which the target panels are highly illuminated and no shadowing effects are present. The RX detector requires a larger amount of processing time, on the order of 100 times more than the cluster-based detectors. The lengthy processing time is due to two main factors: the relatively large targets present in the scene and the use of loop structures in the programming of the algorithm. The large targets require that the guard and background windows be large, increasing the overall complexity of the algorithm. The programming of the RX algorithm is also an issue in this

study due to the way Matlab handles loop structures. The other anomaly detection algorithms are programmed to take advantage matrix operations, which is one of Matlab's strengths. The RX algorithm as programmed here, however, handles the window calculations with a loop structure. In Matlab, calculations performed in loop structures generally require significantly more processing time than equivalent matrix calculations. The RX detector's value has been proven in many real-time processing applications, such as the Civilian Air Patrol's (CAP) Airborne Real-time Cueing Hyperspectral Enhanced Reconnaissance (ARCHER) system. The algorithm would likely see a dramatic improvement in processing time if translated into a matrix based algorithm or a more hardware friendly programming language, such as C/C++ or VHDL.

CHAPTER 6

Conclusions

A summary of the the experimental results of this study is provided in this chapter. The final conclusions based on the experimental results and suggestions for future research in this field are also presented.

6.1 Summary

During the diurnal study, all six anomaly detection methods have very strong detection performance. The overall difference between the best and worst performing detectors is quite small. During this study all methods exhibit a degraded detection performance in the late afternoon and early evening caused by the reduction of overall scene illumination due the setting sun.

During the seasonal study, it is observed that scenes in which the target panels are highly illuminated with no shadowing effects (Aug 24, 2005 – Oct 14, 2005 and Mar 7, 2006 – April 18, 2006) exhibit performance similar to that of the diurnal data sets. With this in mind, special care is given to observing the operation of each detector under adverse conditions (i.e. in the presence of shadowing and snow cover). During these conditions, the performance of the detection methods separate distinctly. Here, the cluster-based methods exhibit significantly improved performance compared to the global and locally adaptive methods. In general, the performance of the cluster-based methods benefits from the addition of classes, up until a point of diminishing returns. The additional clusters provide a

level of robustness to the detectors, allowing them to operate with improved performance in adverse conditions. The exception is the GMRX detector, which displays nearly its best performance with only two clustering classes. FCBAD with six classes features the best overall performance during shadowed conditions, holding a slight advantage over the CBAD.

As expected, target pixel contamination has a negative effect on the performance of all anomaly detection methods. During the diurnal study, the GMM and GMRX detectors show some resistance to target pixel contamination. This resistance is not present in the seasonal study where performance degredation is generally linear. The detectors experience a 1-3% detection performance reduction when all of the target pixels are allowed to contaminate the background statistics.

In the change detection study, the segmented SEM based predictor coupled with a GMRX anomaly detector yields the best overall change detection performance. Additionally, the second best performing combination is a CC predictor coupled with a GMRX anomaly detector. These results support the use of a GMRX anomaly detector to achieve optimum results during the change detection process.

The M-dist detector requires the shortest processing time and is the least computationally complex. However, the M-dist detector's performance is only acceptable in situations where the data does not have significant shadowing effects. The processing time required for the cluster-based methods is directly related to the clustering algorithm employed. Detectors utilizing the SEM clustering algorithm require 2.5 to 3 times more processing time than detectors using k-means clustering. The addition of each new class also adds 10-20% to the overall processing time of both k-means and SEM clustering algorithms. The RX detector requires roughly 100 times more processing time than the cluster-based

methods. This is due to the large window sizes required by the scene and the use of loop structures in the programming of the RX detector. If the RX detector could be programmed to use matrix operations, similar to the other detectors investigated in this study, or translated into a more hardware friendly programming language, it is likely the RX detector would see a significant reduction in its processing time.

6.2 Future Considerations

In this study, clustering is performed on a relatively simple scene. Further testing with a more populated scene, such as an urban environment, would provide a more diverse range of pixel values. In such a scene, it is likely that the cluster-based methods would provide a larger performance advantage over the global and locally adaptive methods. Additionally, with a more diverse range of pixels, it is likely that a larger numbers of classes could be initiated with continued anomaly detection performance improvement.

The change detection study performed here is quite small and more than somewhat lacking when compared to the robustness of the anomaly detection studies. This is due to time constraints as well as a lack of available data to serve as suitable test images. A more extensive study of change detection predictor-anomaly detector combinations is certainly in order. The study should include more reference and test images, accounting for a wide variety of natural and man-made changes.

The computational complexity analysis performed here is somewhat flawed in its general approach. Most of the methods, with the exception of the RX detector, have been programmed in such as way as to improve their performance in the Matlab environment (i.e., matrix math is used and looping structures are eliminated whenever possible). Some attempt

should be made to develop a more Matlab-friendly version of the RX detector. The RX detector has been proven as practical detector and has been successfully applied in numerous real-world anomaly detection applications. Its application here is hampered by Matlab's slow computation ability with regard to looping structures. It may also be advantageous to program the other anomaly detection algorithms discussed in this study in a more hardware friendly programming language, such as C++ or VHDL. These steps would provide a more balanced analysis of the true computational complexity and processing time associated with each method.

APPENDIX A

Color Images

This appendix contains color images for all of the hyperspectral data used in the seasonal, diurnal and change detection data studies.

A.1 Diurnal Images

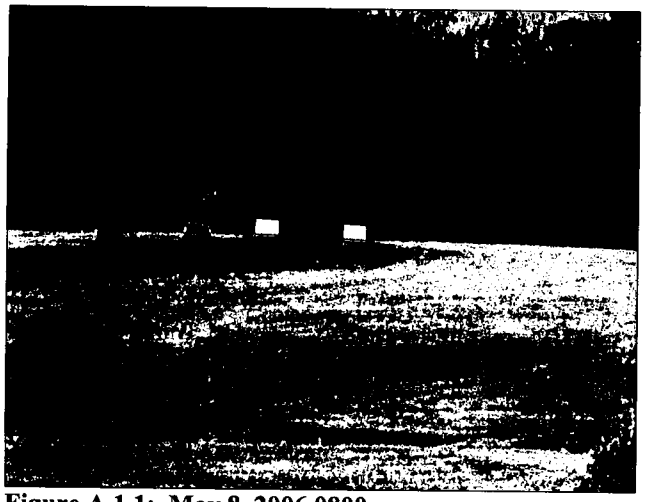

Figure A.1.1: May 8, 2006 0800

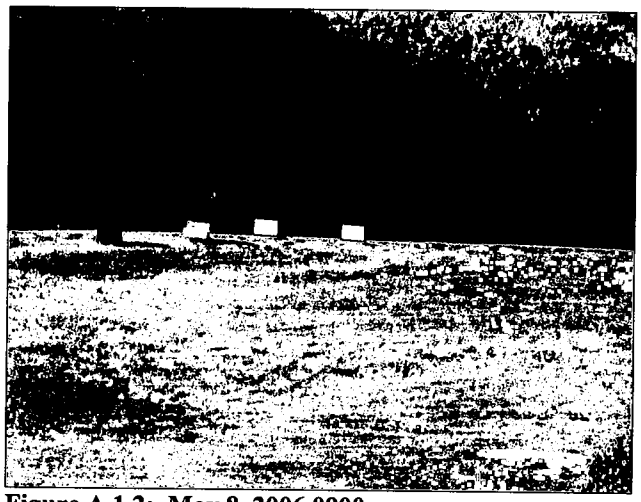

Figure A.1.2: May 8, 2006 0900

Figure A.1.3: May 8, 2006 1000

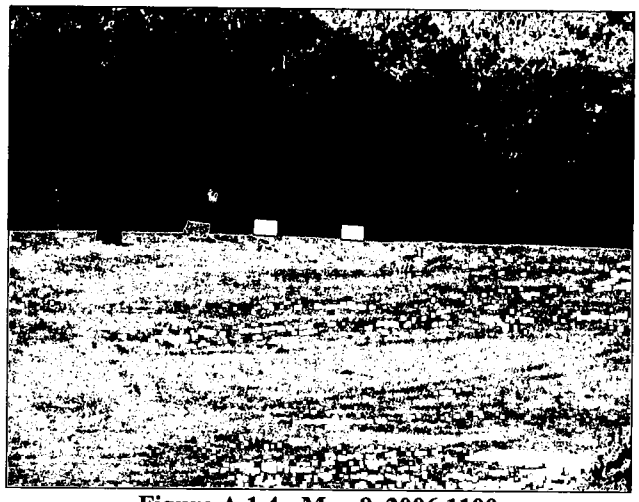

Figure A.1.4: May 8, 2006 1100

Figure A.1.5: May 8, 2006 1200

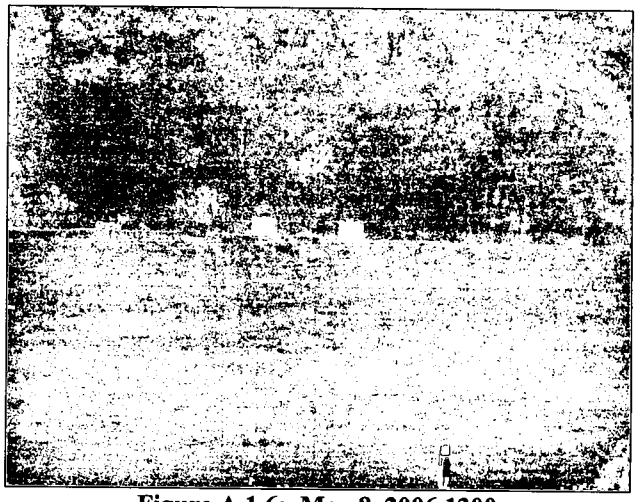

Figure A.1.6: May 8, 2006 1300

Figure A.1.7: May 8, 2006 1400

Figure A.1.8: May 8, 2006 1500

Figure A.1.9: May 8, 2006 1600

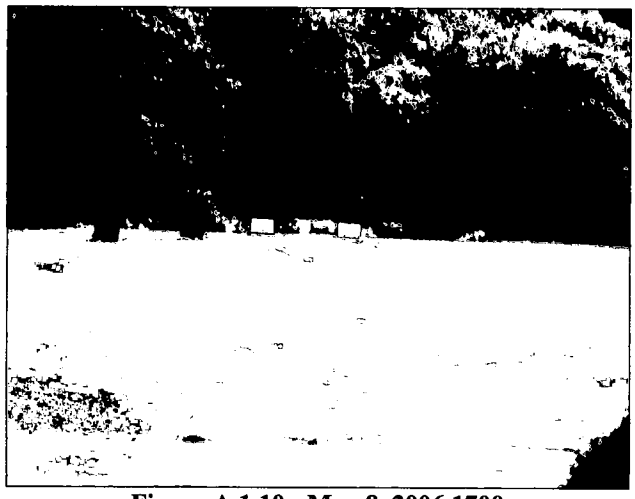

Figure A.1.10: May 8, 2006 1700

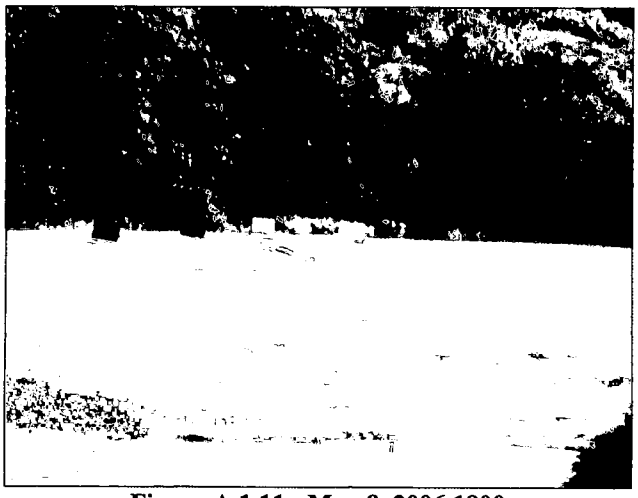

Figure A.1.11: May 8, 2006 1800

Figure A.1.12: May 8, 2006 1900

A.2 Seasonal Images

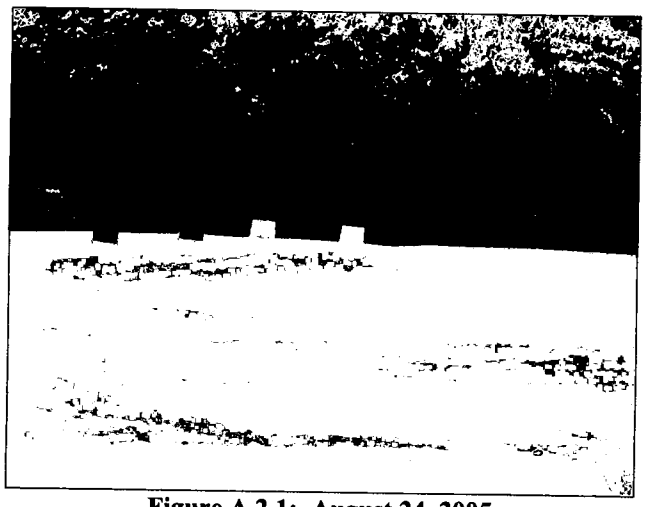

Figure A.2.1: August 24, 2005

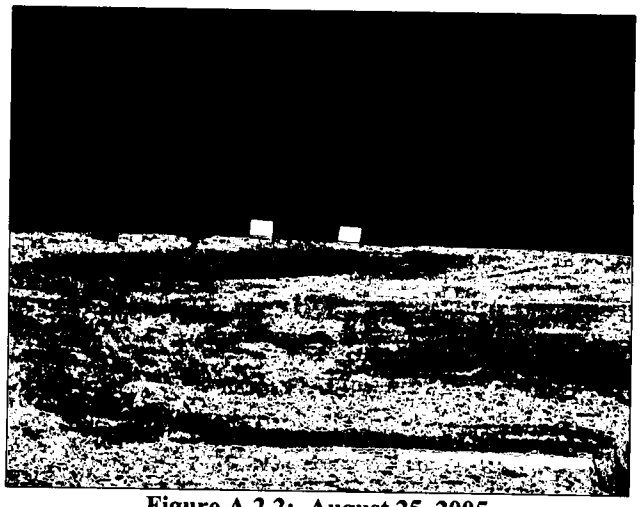

Figure A.2.2: August 25, 2005

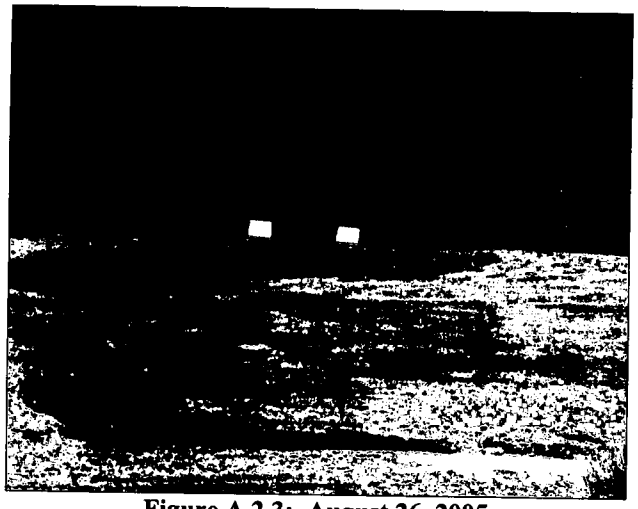

Figure A.2.3: August 26, 2005

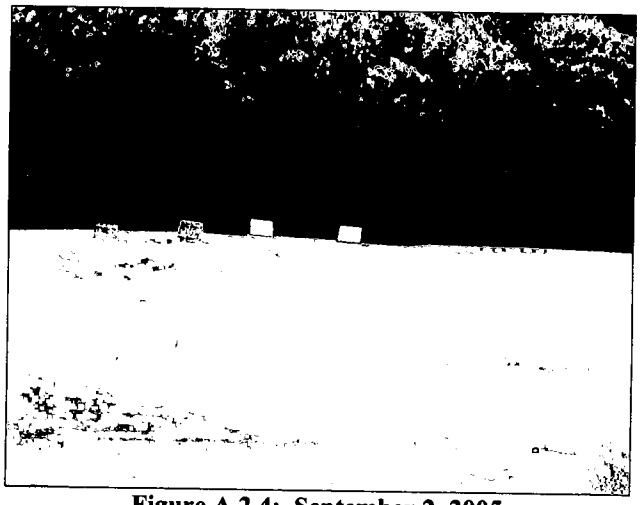

Figure A.2.4: September 2, 2005

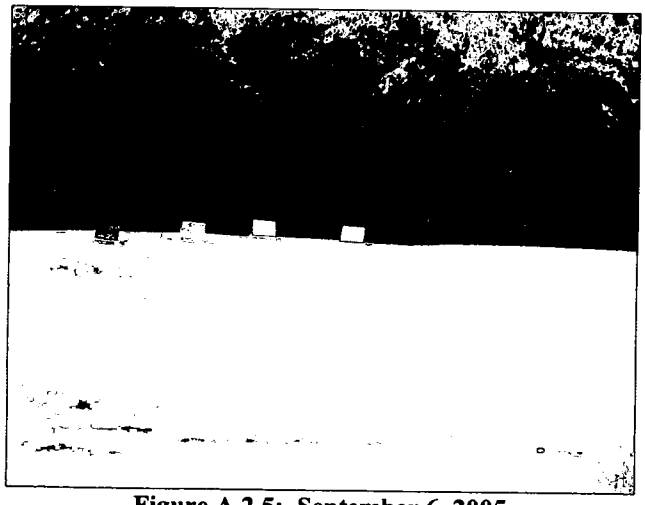

Figure A.2.5: September 6, 2005

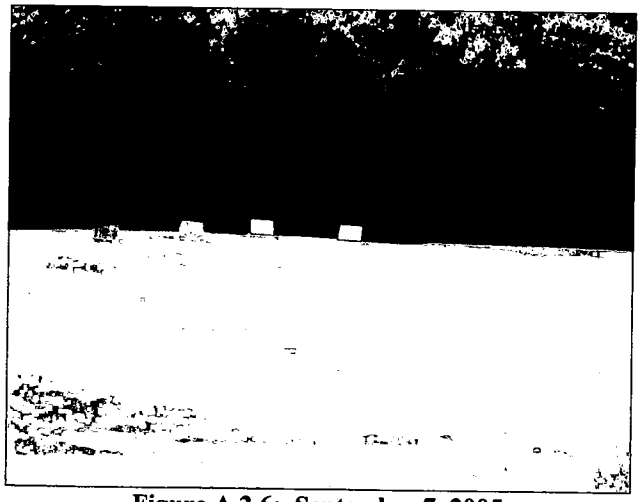

Figure A.2.6: September 7, 2005

Figure A.2.7: September 10, 2005

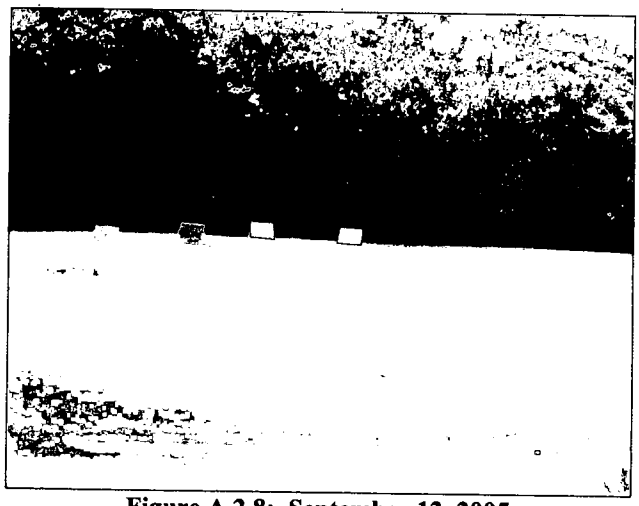

Figure A.2.8: September 12, 2005

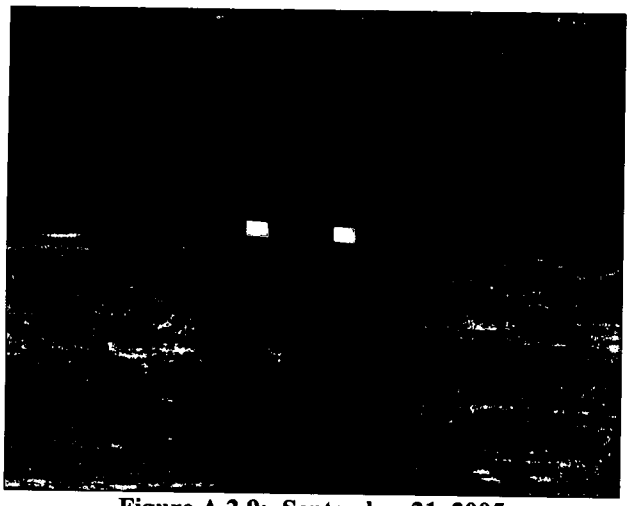

Figure A.2.9: September 21, 2005

Figure A.2.10: September 22, 2005

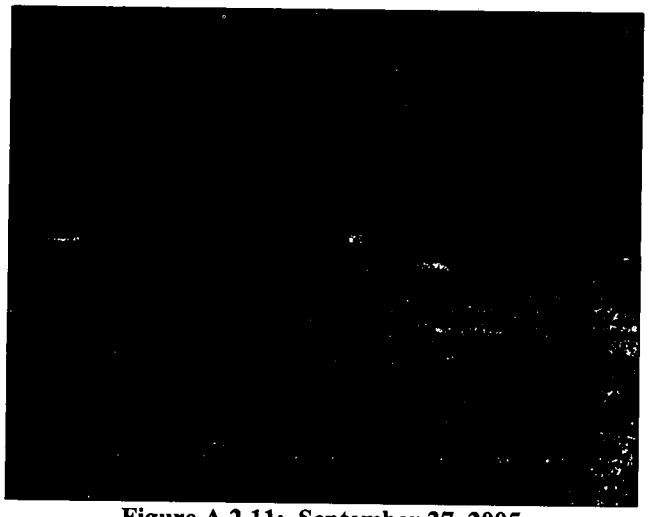

Figure A.2.11: September 27, 2005

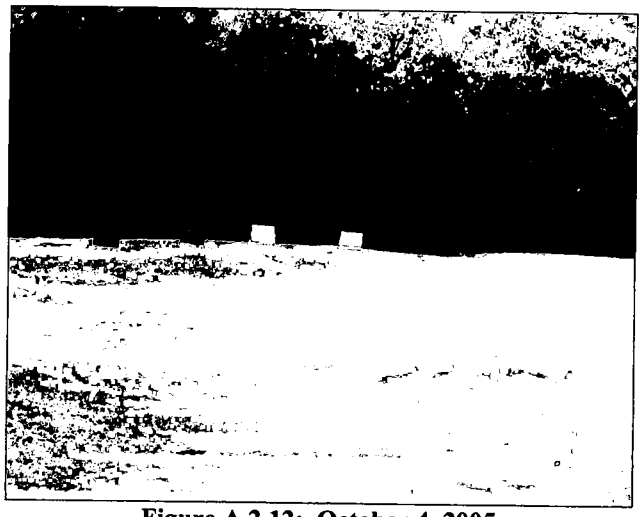

Figure A.2.12: October 4, 2005

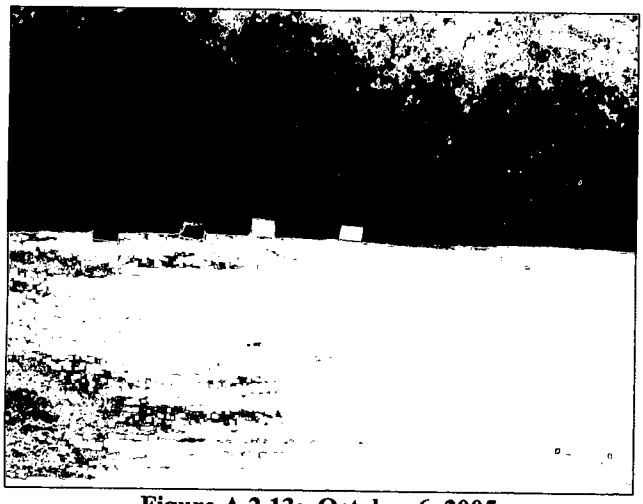

Figure A.2.13: October 6, 2005

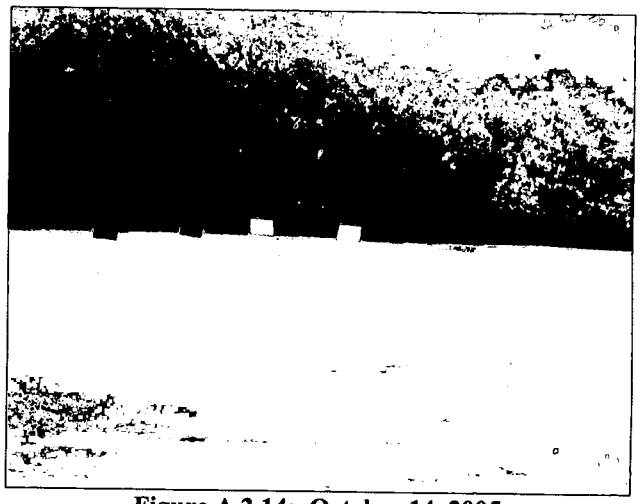

Figure A.2.14: October 14, 2005

Figure A.2.15: October 17, 2005

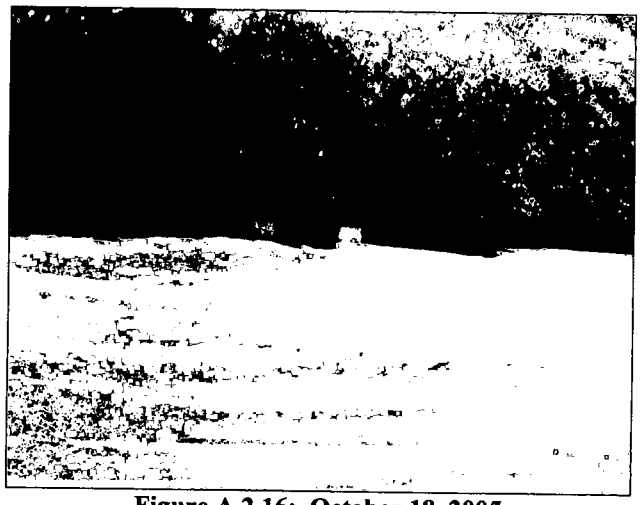

Figure A.2.16: October 18, 2005

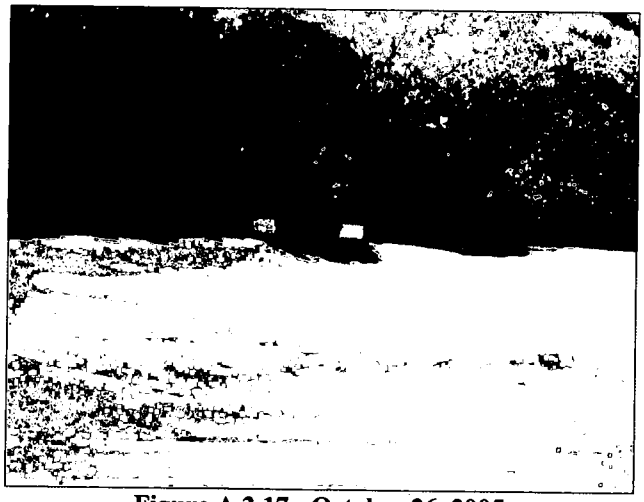

Figure A.2.17: October 26, 2005

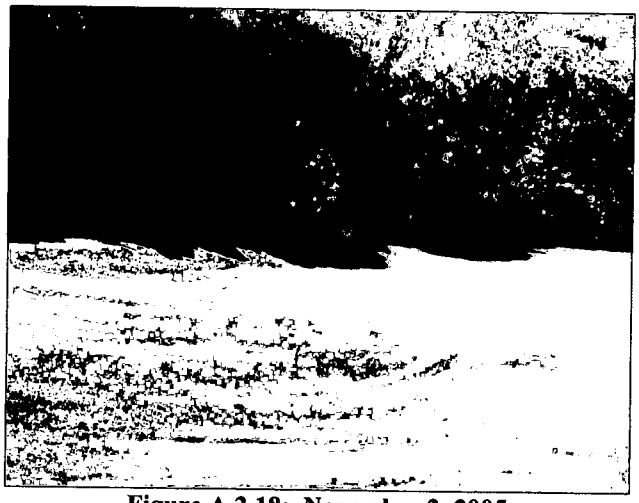

Figure A.2.18: November 2, 2005

Figure A.2.19: November 3, 2005

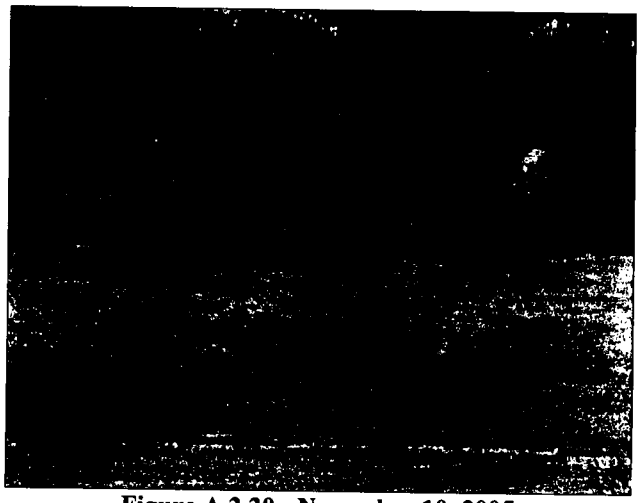

Figure A.2.20: November 10, 2005

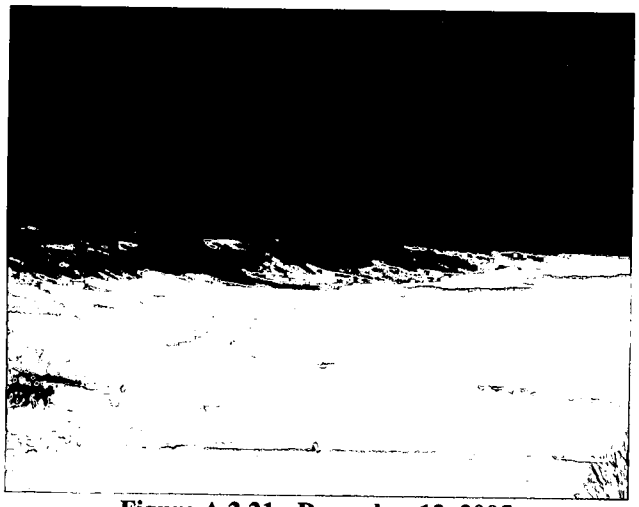

Figure A.2.21: December 12, 2005

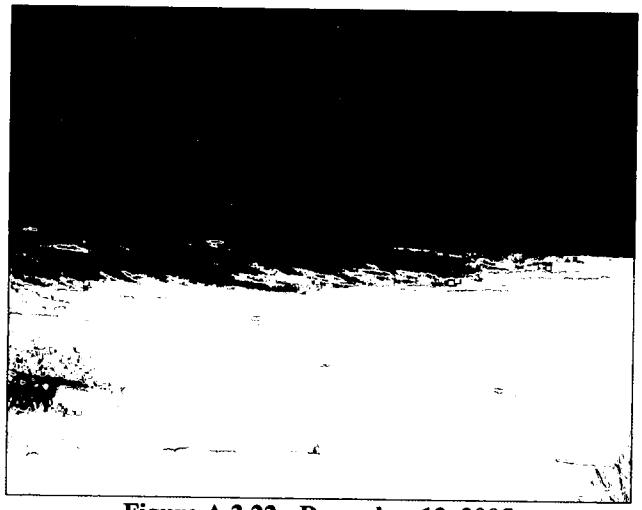

Figure A.2.22: December 13, 2005

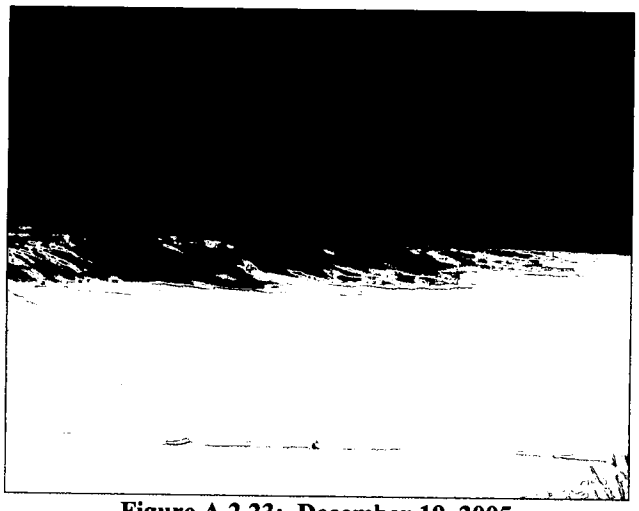

Figure A.2.23: December 19, 2005

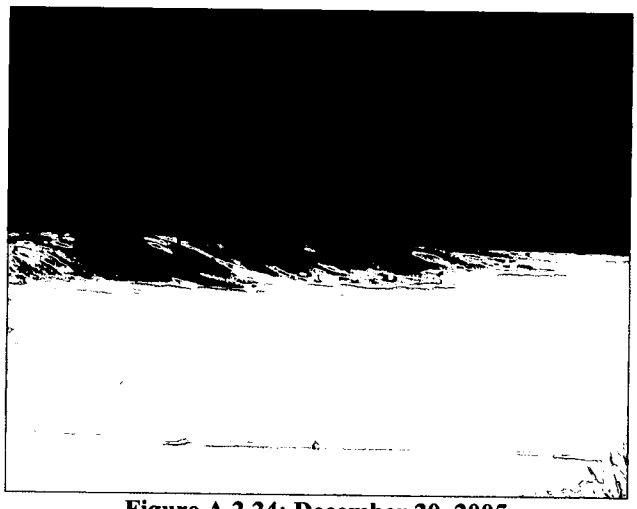

Figure A.2.24: December 20, 2005

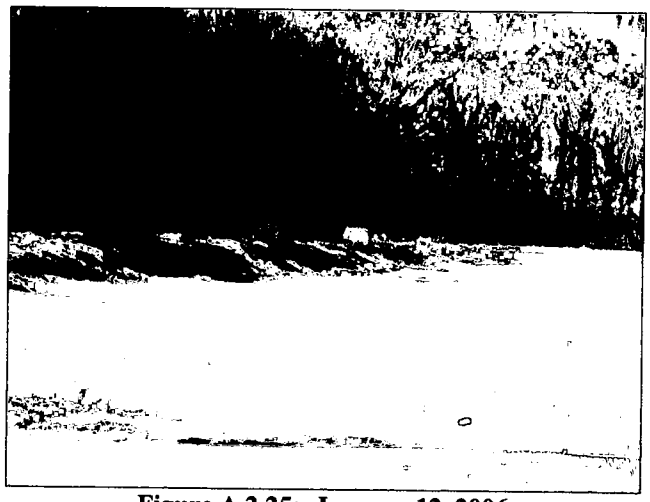

Figure A.2.25: January 12, 2006

Figure A.2.26: January 19, 2006

Figure A.2.27 January 23, 2006

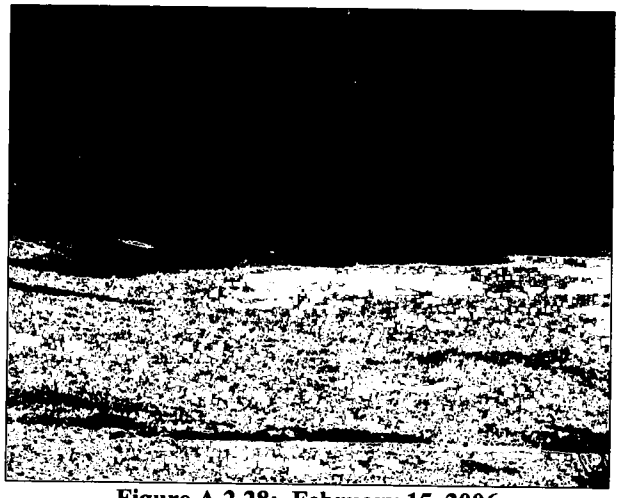

Figure A.2.28: February 15, 2006

Figure A.2.29: February 23, 2006

Figure A.2.30: February 24, 2006

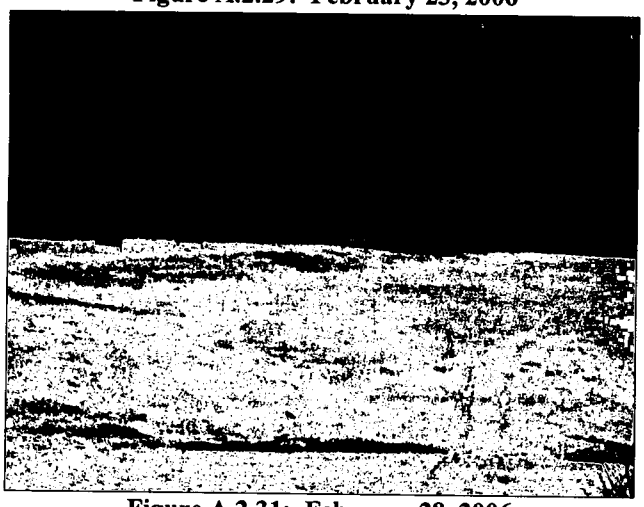

Figure A.2.31: February 28, 2006

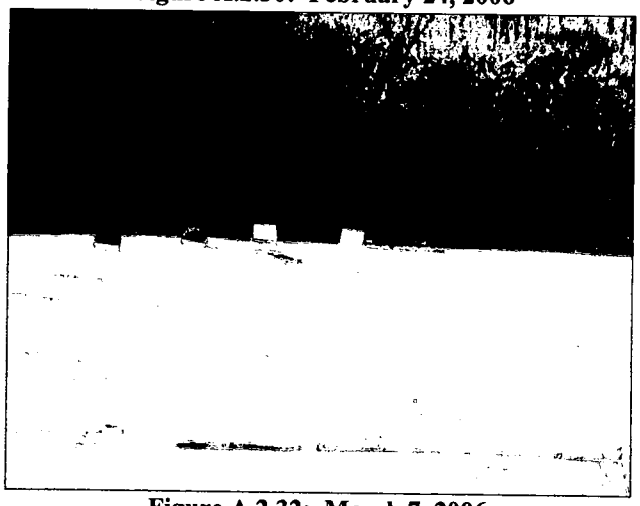

Figure A.2.32: March 7, 2006

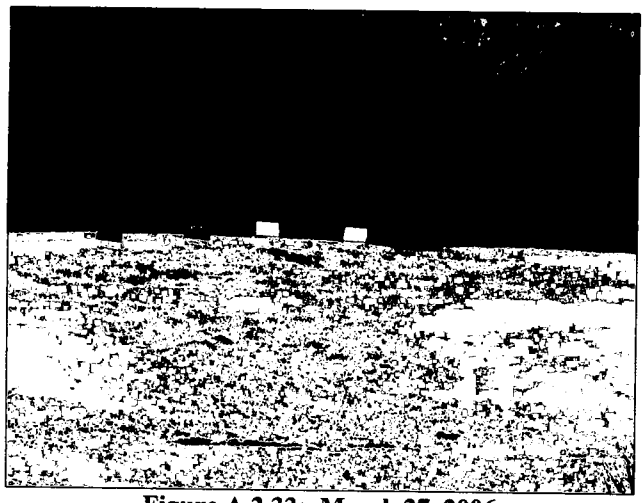

Figure A.2.33: March 27, 2006

Figure A.2.34: April 10, 2006 SA

Figure A.2.35: April 11, 2006

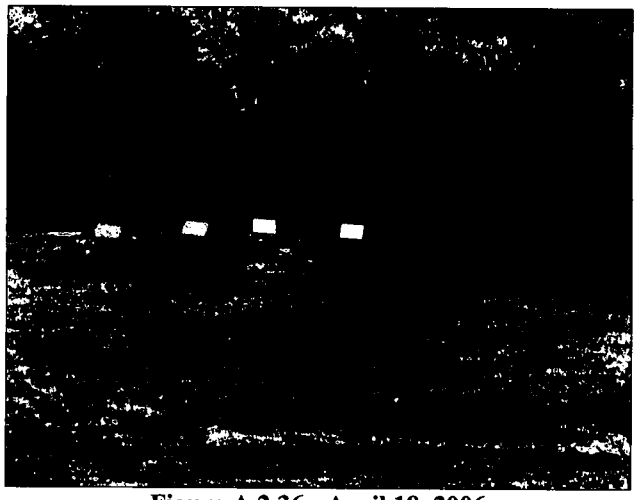

Figure A.2.36: April 18, 2006

A.3 Change Detection Images

Figure A.3.1: Reference Image Oct 26, 2005

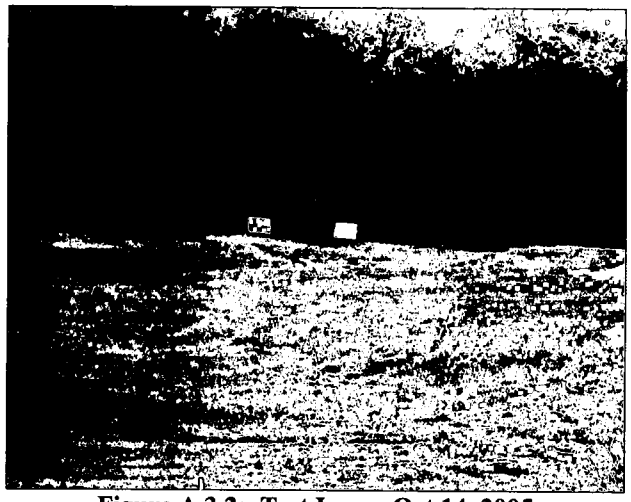

Figure A.3.2: Test Image Oct 14, 2005

APPENDIX B

Matlab Code

This appendix contains relevant Matlab code used to produce the results for this work. The code for the diurnal and seasonal code is roughly the same, with only naming convention changes. This appendix only presents code used in the seasonal study, denoted by its solar position (180az).

B.1 Anomaly Detection Code

anomaly_detection_methods_180az

```
% This program calculates and outputs the anomaly detection statistics for
% Global Anomaly Detection, Cluster Based Anomaly Detection (CBAD), and
% Gaussian Mixture Model (GMM) Anomaly Detection with kmeans clustering
% stats and SEM clustering stats.  The algorithms are preformed on 36
% hyperspectral data sets from August 2005 - May 2006 with solar position
% of 180az.


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%% INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% set_start             specify first data set to be used

% set_end               specify last data set to be used

% num_class             the number of classes

% disp_pca              1 - display pca image (band 1)
%                       0 - do not display pca image

% disp_mask             1 - display target and ignore mask
%                       0 - do not display target and ignore mask

% target                allows for the modification of the mask
%                       to include parts of the targets into each anomaly
%                       detection algorithm's statistics calculation
%                       0 - no targets
%                       1 - all targets
```

```
% thresh                number of thresholds for ROC curve generation

% tfa                    target_fraction_allowed

% kmeans_its             number of iterations used in k-means clustering

% sem_its                number of iterations used in SEM clustering

% save                   1 - save varaibles in workspace to .mat file
%                        0 - do not save variables


clear all
close all
clc
tic

kmeans_its1 = 1;
kmeans_its5 = 5;
sem_its5 = 5;

set_start = 1;
set_end = 36;

thresh = 200;
save_data = 1;
disp_pca = 0;
disp_mask = 0;

%%%%%% Date Inputs %%%%%%
% date1 = '24 Aug 2005';
% date2 = '25 Aug 2005';
% date3 = '26 Aug 2005';
% date4 = '2 Sept 2005';
% date5 = '6 Sept 2005';
% date6 = '7 Sept 2005';
% date7 = '10 Sept 2005';
% date8 = '12 Sept 2005';
% date9 = '21 Sept 2005';
% date10 = '22 Sept 2005';
% date11 = '27 Sept 2005';
% date12 = '4 Oct 2005';
% date13 = '6 Oct 2005';
% date14 = '14 Oct 2005';
% date15 = '17 Oct 2005';
% date16 = '18 Oct 2005';
% date17 = '26 Oct 2005';
% date18 = '2 Nov 2005';
% date19 = '3 Nov 2005';
% date20 = '10 Nov 2005';
% date21 = '12 Dec 2005';
% date22 = '13 Dec 2005';
% date23 = '19 Dec 2005';
% date24 = '20 Dec 2005';
```

```
% date25 = '12 Jan 2006';
% date26 = '19 Jan 2006';
% date27 = '23 Jan 2006';
% date28 = '15 Feb 2006';
% date29 = '23 Feb 2006';
% date30 = '24 Feb 2006';
% date31 = '28 Feb 2006';
% date32 = '7 Mar 2006';
% date33 = '27 Mar 2006';
% date34 = '10 Apr 2006';
% date35 = '11 Apr 2006';
% date36 = '18 Apr 2006';

% All dates are with sun at 180 az.

% Author: Patrick Hytla
% University of Dayton
%
% 5/23/2007
%
% COPYRIGHT © 2007 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Begin Prodecure
for num_class = 2
  for target = 0
    for tfa = 0           % target_fraction_allowed


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%% Mask Adjustment (tfa) %%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Allow target pixels into statistics calculation
        if target == 1
          tai = all_targets;
          I  = find(tai);
          tai2 = zeros(size(tai));
          N = length(I);
          r = round((1-tfa)*N);
          subset = round( linspace(1,N,r) );
          subset = unique(subset);
          tai2( I(subset) ) = 1;
          target_and_ignore = tai2 | all_ignore;
        end
        if disp_mask == 1          % display mask
          im(target_and_ignore)
          title(date)
        end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%% Clustering %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % ******** K-means Clustering *************
        % k-means clustering
        stats_kmeans1 = kmeans_hsi(pca, num_class, 100, kmeans_its1,
target_and_ignore);
        stats_kmeans5 = kmeans_hsi(pca, num_class, 100, kmeans_its5,
target_and_ignore);
        % *****************************************

        % ******** SEM Clustering *****************
        % random initialization for SEM
        [nr,nc,nb]=size(pca);
        rand('state',sum(100*clock));
        cmap=ceil(num_class*rand(nr,nc));
        cmap(find(target_and_ignore>0))=0;
        stats_random = get_stats_class(pca,cmap,target_and_ignore);

        % Feed the stats from kmeans and random to initialize the SEM
        % 5 SEM it
        [stats_sem_k1_5it, class_sem_k1_5it, P_sem_k_5it] =
stochastic_expectation_maximization(pca, sem_its5, stats_kmeans1,
target_and_ignore);
        [stats_sem_r_5it, class_sem_r_5it, P_sem_r_5it] =
stochastic_expectation_maximization(pca, sem_its5, stats_random,
target_and_ignore);
        % ******************************************
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%% Anomaly Detection Methods %%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % ******** Global Mahalanobis Distance (M- distance) ********
        [outM, statsM] = anomaly_detection(pca, target_and_ignore);
        [pd_M(n,:),pf_M(n,:),T_M(n,:),auc_M(n,:)] =
generate_ROC_curve(outM,all_targets,thresh,all_ignore,1);
        % ***********************************************************

        % ******** GMM Anomaly detection **************
        % GMM w/k-means clustering
        out_gmm_kmeans5 = anomaly_detection_likelihood(pca,
stats_kmeans5);

[pd_gmm_kmeans5(n,:),pf_gmm_kmeans5(n,:),T_gmm_kmeans5(n,:),auc_gmm_kmeans
5(n,:)] =
generate_ROC_curve(out_gmm_kmeans5,all_targets,thresh,all_ignore,1);

        % GMM w/SEM
        % 5 SEM iterations
        % 1 kmeans initialization iteration
```

```
        out_gmm_sem_k1_5it = anomaly_detection_likelihood(pca,
stats_sem_k1_5it);

[pd_gmm_sem_k1_5it(n,:),pf_gmm_sem_k1_5it(n,:),T_gmm_sem_k1_5it(n,:),auc_g
mm_sem_k1_5it(n,:)] =
generate_ROC_curve(out_gmm_sem_k1_5it,all_targets,thresh,all_ignore,1);

        % GMM w/SEM (random initialization)
        out_gmm_sem_r_5it = anomaly_detection_likelihood(pca,
stats_sem_r_5it);

[pd_gmm_sem_r_5it(n,:),pf_gmm_sem_r_5it(n,:),T_gmm_sem_r_5it(n,:),auc_gmm_
sem_r_5it(n,:)] =
generate_ROC_curve(out_gmm_sem_r_5it,all_targets,thresh,all_ignore,1);
        % *********************************************

        % ******** Cluster Based Anomaly Detection (CBAD) ********
        % 5 kmeans iteration
        class_vq_pca2_5 = classify_monolithic(pca, stats_kmeans5,
'identity', 'dist');
        out_cbad5 = anomaly_detection_class(pca, stats_kmeans5,
class_vq_pca2_5 );
        [pd_cbad5(n,:),pf_cbad5(n,:),T_cbad5(n,:),auc_cbad5(n,:)] =
generate_ROC_curve(out_cbad5,all_targets,thresh,all_ignore,1);
        % *******************************************************

        % ***** GMRX w/ SEM *****
        stats_kmeans1 = kmeans_hsi(pca, num_class, 100, kmeans_its1,
target_and_ignore);
        [stats_gmrx, class_gmrx, P_gmrx] =
stochastic_expectation_maximization(pca, sem_its, stats_kmeans1,
target_and_ignore);
        P = generate_membership_monolithic(pca, stats_gmrx, 'full',
'dist', 2 );
        out_gmrx = min(P,[],3);
        [pd_gmrx(n,:),pf_gmrx(n,:),T_gmrx(n,:),auc_gmrx(n,:)] =
generate_ROC_curve(out_gmrx,all_targets,thresh,all_ignore,1);
        % ********************

        % ***** FCBAD (Soft Partition) *****
        P2 = generate_membership_monolithic(pca, stats_kmeans5, 'full' ,
'butterworth' , 2 );
        % Try anomaly detection class with M-dist class and M-dist
statistoc
        MD = generate_membership_monolithic(pca, stats_kmeans5, 'full' ,
'dist' );
        out_MD = sum( P2.*MD, 3 );
        [pd_fcbad(n,:),pf_fcbad(n,:),T_fcbad(n,:),auc_fcbad(n,:)] =
generate_ROC_curve(out_MD,all_targets,thresh,all_ignore,1);
        % *****************
    end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%% Save Data %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
        if save_data == 1

save(filename1,'auc_M','auc_cbad5','auc_gmm_kmeans5','auc_gmm_sem_k1_5it',
'auc_gmm_sem_r_5it',...

'pd_M','pd_cbad5','pd_gmm_kmeans5','pd_gmm_sem_k1_5it','pd_gmm_sem_r_5it',
...

'pf_M','pf_cbad5','pf_gmm_kmeans5','pf_gmm_sem_k1_5it','pf_gmm_sem_r_5it',
...

'T_M','T_cbad5','T_gmm_kmeans5','T_gmm_sem_k1_5it','T_gmm_sem_r_5it',...

'xx','target_and_ignore','num_class','target','tfa','thresh');
        end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    end
  end
end

toc
```

kmeans_hsi

```
function stats = kmeans_hsi( in, num_class, min_class, num_its, ignore,
stats );
%
% stats = kmeans_hsi( in, num_class, min_class, num_its, ignore, stats );
%
% K-means clustering algorithm for hyperspectral imagery.
%
% REQUIRED INPUTS
% in             3D HSI/MSI image spectral simension is 3rd dimension
% num_class      Number of classes
% min_class      Minimum number of samples in a class
% num_its        Number of iterations
%
% OPTIONAL INPUTS
% ignore    2D binary mask indicating what to exclude from statistics (can
%                be empty [] or not included)
% stats.M   2D array, rows are initial mean vectors of classes
%
% OUTPUTS
% stats     Statistical parameters to use if previsouly computed
%   stats.M      2D array, rows are mean vectors of background classes
%   stats.C      3D array, Covariances of background classes
%   stats.CI     3D array, Inverse covariances of background classes
%   stats.P
%   stats.S
%   stats.nobs   Number of observations associate with each class
```

```
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/12/06, 5/17/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


%---------------%
% Get size info %
%---------------%

[sy,sx,sz] = size(in);
nov = sx*sy;

if nargin > 4 % ignore mask provided

    % Find idices of hyperpixels to use in clustering
    I_use = setdiff( [1:nov], find(ignore(:)) );

    % Extract only the hyperpixels we want into X
    % Each hyperpixel is a column
    in = reshape( in, nov, sz );
    nov = length( I_use );
    in = in( I_use, : );

else

    % Put all the hyperpixels in the matrix as columns
    in = reshape( in, nov, sz );
    % in = reshape( shiftdim( in, 2 ), sz, nov );

end

%-----------------------------------%
% Pick initial codebook if necessary %
%-----------------------------------%

if nargin == 5 || ( nargin == 6 && ( isempty(stats) || ~isfield(stats,M) )
) % no initial info provided

    % pick evenly spaced vectors to initialize means
    pick = round( linspace( 1, nov, num_class ) );
    stats.M = in( pick, : );

end

% Initialize paramaters
stats.nobs = zeros( 1, num_class );
stats.C = zeros( sz, sz, num_class );
stats.CI = zeros( sz, sz, num_class );
stats.S = zeros( num_class, sz );
```

```
for m = 1 : num_its

    disp(sprintf('Beginning clustering iteration %d of %d', m, num_its ));

    %---------------------------------------------%
    % Assign each observation vector to a cluster %
    %---------------------------------------------%

    % Find distance from each obs vec to one code word at a time
    D = zeros( nov, num_class );

    for n = 1 : num_class  % loop over classes

        % Distance from each obs to  code word n
        D( :, n ) = sum( ( in - repmat( stats.M(n,:), nov, 1 ) ).^2, 2 );

    end

    % Assign each obs to codeword with min distance
    [ min_D, min_idx ]=min( D, [], 2 ); % minind = set of codeword
assignment indices.

    %-----------------------------------------------%
    % Combine all obs vecs assigned to same codeword %
    %-----------------------------------------------%

    for n = 1 : num_class  % loop over codewords

        nI = find( min_idx == n ); % indices of all obs vecs assigned to n
        stats.nobs(n) = length(nI);  % number of obs vecs assigned to n

        if m == num_its

            % Last time, so comput all output parameters
            stats.S(n,:) = var( in( nI, : ) );
            stats.C(:,:,n) = cov( in( nI, : ) );
            stats.CI(:,:,n) = inv( stats.C(:,:,n) );
            stats.P(n) =  stats.nobs(n) / nov;

        else

            % Not the last time, check sizes of clusters and update means
            if stats.nobs(n) < min_class

                rdobs=round(rand*nov-1)+1;
                stats.M(n,:)=in(rdobs,:);
                disp(sprintf('Abandoning codeword %d, using obs vec
%d',n,rdobs));

            else

                stats.M(n,:) = mean( in( nI, : ) );
```

```
            end

        end

    end

end
```

get_stats_class

```
function stats = get_stats_class( in, class, ignore );
%
% stats = get_stats_class( in, class, ignore );
%
% Computes the class statstics from the samples in each class
% in teh class map ('class').  Uses sample estimates.
%
% REQUIRED INPUTS
% in            3D HSI/MSI image spectral simension is 3rd dimension
% class         2D class map
%
% OPTIONAL INPUTS
% ignore    2D binary mask indicating what to exclude from statistics (can
%               be empty [] or not included)
%
% OUTPUTS
% stats     Statistical parameters to use if previsouly computed
%   stats.M     2D array, rows are mean vectors of background classes
%   stats.C     3D array, Covariances of background classes
%   stats.CI    3D array, Inverse covariances of background classes
%   stats.P
%   stats.S
%   stats.nobs  Number of observations associate with each class
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 7-20-06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


%---------------%
% Get size info %
%---------------%

[sy,sx,sz] = size(in);
nov = sx*sy;

% Put all the hyperpixels in the matrix as columns
in = reshape( in, nov, sz );

% Number of classes
```

```
num_class = max(class(:));

if nargin > 2 % ignore mask provided

    % Find indices of hyperpixels to ignore
    Iignore =  find(ignore(:));

    % Make ignore pixels a 0 class to be ignored
    class(Iignore) = 0;

    % Number of hyperpixels excluding ignores
    nov_after_ignore = nov - length(Iignore);

else

     % Number of hyperpixels excluding ignores
    nov_after_ignore = nov;

end


% Initialize paramaters
stats.nobs = zeros( 1, num_class );
stats.C = zeros( sz, sz, num_class );
stats.CI = zeros( sz, sz, num_class );
stats.S = zeros( num_class, sz );


for n = 1 : num_class  % loop over classes

    nI = find( class == n ); % indices of all obs vecs assigned to n
    stats.nobs(n) = length(nI);  % number of obs vecs assigned to n

    % Compute all output parameters
    stats.S(n,:) = var( in( nI, : ) );
    stats.C(:,:,n) = cov( in( nI, : ) );
    stats.CI(:,:,n) = inv( stats.C(:,:,n) );
    stats.P(n) =  stats.nobs(n) / nov_after_ignore;
    stats.M(n,:) = mean( in( nI, : ) );

end
```

stochastic_expectation_maximization

```
function [ stats, class, P ] = stochastic_expectation_maximization( in,
num_its, stats, ignore, seed );
%
% [ stats, class, P ] = stochastic_expectation_maximization( in, num_its,
stats, ignore, seed );
%
% Stochastic expectation maximization algorithm.
%
```

```
%
% REQUIRED INPUTS
% in            3D HSI/MSI image spectral simension is 3rd dimension
% num_its       Number of iterations
%
% stats     Initialization statistical parameters (recommend using
kmeans_hsi.m)
%   stats.M   2D array with class mean vectors for background as rows
%   stats.C   3D array of class covariances for background
%   stats.P   1D array of class prior probabilities
%
% OPTIONAL INPUTS
% ignore    2D binary mask indicating what to exclude from statistics (can
%               be empty [] or not included)
%
% seed      State for rand().  Used to get repeatable results.
%
% OUTPUTS
% stats     Statistical parameters to use if previsouly computed
%   stats.M     2D array, rows are mean vectors of background classes
%   stats.C     3D array, Covariances of background classes
%   stats.CI    3D array, Inverse covariances of background classes
%   stats.P
%   stats.S
%   stats.nobs  Number of observations associate with each class
%
% Author: Patrick Hytla and Russell Hardie, Ph.D.
% University of Dayton
%
% 7-26-06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.

[sy,sx,sz]=size(in);

if nargin < 4 | isempty( ignore )
    ignore = zeros(sy,sx);
end

if nargin < 5 || isempty(seed)
    seed = sum(100*clock);
end

for k = 1 : num_its

    % Get posteriori class assignment probabilities based on the initial
stats
    P = generate_membership_monolithic( in, stats, 'full', 'post', [] );

    % Identify and treat bad pixels with no membership within working
    % precision
    ignore_bad = P(:,:,1)==-1;
    I_ignore = find( P == -1 );
    P(I_ignore) = 1/size(stats.M,1);
```

```
    % Perform stochastic class assignment
    class = stochastic_class_assignment( P, k*seed );

    % Compute statistics
    stats = get_stats_class( in, class, ignore | ignore_bad );

end
```

generate_membership_monolithic

```
function P = generate_membership_monolithic( in, stats, cov_type,
func_type, K )
%
% P = generate_membership_monolithic( in, stats, cov_type, func_type, K )
%
% Generates membership function (or M-dist) for each class at each pixel
in input
% "in" using class statistics provided in "stats".  Various membership
% function options are available and are determined by "flag".
% Some pixels will have no membership to within the working
% prescision.  These pixels are flagged with all -1 values.
%
% REQUIRED INPUTS
% in        3D HSI/MSI image (spectral dimension is 3rd dimension)
%
% stats     Statistical parameters to use if previsouly computed
%   stats.M   2D array with class mean vectors for background as rows
%   stats.C   3D array of class covariances for background
%   stats.P   1D array of class prior probabilities (only needed for
%               posteriori membership function)
%
% OPTIONAL INPUTS
% cov_type     Type of covariance to use for membership function.  It is
%              assumed that the full covariance is provided in stats.  We
can simplify
%              this as follows:
% 'identity'    Use identiy matrices for covariances
% 'spherical'   Use the average band variance and scale identity mx
% 'diagonal'    Use only the band variances (take diagonal)
% 'full'        (default) Uses the full provided covariances
%
% func_type      Type of membership function to use (all normalized to sum
to 1)
%   'gaussian '    = (default) Normalized Gaussian membership function
%                        P_i(x) = exp( -.5*(x-mu_i)'*CI_i(x-mu_i) )
%   'posteriori'     = Gaussian a posteriori probability
%                        P_i(x) = pr( class=i | x ) for Gaussian Mixture
Model
%   'butterworth' = Butterwork membership
%                        P_i(x) = 1 / ( 1 + ((x-mu_i)'*CI_i(x-mu_i))^K )
%   'dist'         = M-distance to each class (can be used for
%                        classification and anomaly detection)
%
```

```
% K     Parameter used in Butterworth membership function above.  Default
= 1
%
% OUTPUT
% P       Membership map same spatial dimensions as in, but number of
%           classes deep.  For hard partitioing these would be binary with
one one
%           in each pixel position (for one and only one class).
%           If func_type = 'dist' the output is not the membership
function
%           but rather the M-distances to each class.
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/13/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.

% Later give option to modifiy covariance matrices here (rather than
% externally for identity, spherical, diagonal, full)
% Explore use of const.  We did not do in soft-PWS or in GMMEM, that
% was probably a mistake.  Normalized Gaussian membership function.
% flag the use of the const term. - normal Gausss membership versus a
% posteriori probablity.  Explore effect of priors too.

% Size info
[ sy, sx, sz ]=size( in );
NC = size( stats.C, 3 );
P = zeros( sy*sx, NC );

% Set up defaults
if nargin == 4

    K = 1;

elseif nargin == 3

    func_type = 'gaussian';
    K=1;

elseif nargin == 2

    cov_type = 'full'
    func_type = 'gaussian';
    K=1;

end

% Reshape input
in =  reshape( in, sx*sy, sz );

% Pre-invert matrices
CI = stats.C;
for k = 1 : NC
```

```
    switch cov_type
        case 'identity'
            C = eye( sz );
        case 'spherical'
            C = eye( sz ) * mean( diag( stats.C(:,:,k) ) );
        case 'diagonal'
            C = diag( diag( stats.C(:,:,k) ) );
        case 'full'
            C =  stats.C(:,:,k);
    end

    CI(:,:,k) = inv( C );

end

% Break image into "chunks" to keep from memory overload
chunk = 40000;
M=ceil( sy*sx/chunk );

% Loop over "chunks"
for m = 1 : M

    % Subset of hyperpixels to consider
    I = [ (m-1)*chunk+1: min( sy*sx, m*chunk ) ];

    % Initialize the partition membership function
    nov = length( I );

    for n = 1 : NC  % loop over classes

        % Compute M-distances
        temp = in(I,:) - repmat( stats.M(n,:), nov, 1 );
        D = sum ( ( ( temp * CI(:,:,n) ) .* temp )' );

        if strcmp( func_type(1:4), 'gaus' )

            P(I,n) = exp( -.5*D(:) )';

        elseif strcmp( func_type(1:4), 'post' )

            % pdf constant
            const = 1 / sqrt( ((2*pi)^2)*det( stats.C(:,:,n) ) );
            P(I,n) = stats.P(n) * const * exp( -.5*D(:) )';

        elseif strcmp( func_type(1:4), 'dist' )

            P(I,n) = D';

        else % Butterworth

            P(I,n) = ( 1 ./ ( 1 + D.^K ) )';
```

```
        end

    end

    m

end

% If membership function, normalize to sum to one.
% Do not do this if func_type = 'distance', although
% that would probably not hurt our final applications
P = P';

if ~strcmp( func_type(1:4), 'dist' )

    % Normalize (membership should sum to 1)
    P_sum = sum(P);
    I_good = find( P_sum > 0 );
    P(:,I_good) = P(:,I_good) ./ repmat( P_sum(I_good), [NC,1] );

    % Flag pixels with no membership within working precision
    I_bad = find( P_sum == 0 );
    P(:,I_bad ) = -1;

end

% Reshape P
P = reshape( P', sy, sx, NC );
```

stochastic_class_assignment

```
function class = stochastic_class_assignment( prob, seed );
%
% class = stochastic_class_assignment( prob, seed );
%
% Randomizes class assignments based on probabilities provided.  This is a
% key component of the stochastic expectation maximization algorithm.
%
% REQUIRED INPUTS
%
% prob = 3D class membership probability for each pixel in each class.
%        Dimensions 1 & 2 are spatial.  3rd dimension is the probability.
%        Sum over 3rd dim should be 1.
%
% OPTIONAL INPUTS
%
% seed   State for the rand() generator.  Allows one to generate
repeatable
%        results.
%
% OUTPUTS
% class  Class index map.  Same spatial dimensions as prob.
%
```

```
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 7-20-06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% Get size
[ sy, sx, num_class ] = size(prob);
prob = reshape( prob, sx*sy, num_class );

% Cumulative sum of probabilites
P = cumsum( prob, 2 );

% Set state for rand()
if nargin < 2
  rand('state',sum(100*clock))
else
    rand('state',seed)
end

% Generate uniform random #'s
R = repmat( rand( sx*sy, 1 ), 1, num_class );

% Compare with probabilities to see in what range they lie
compare = R > P;

% Assign to class corresponding to the probability range for that class
class = sum(compare,2)+1;

% Reshape to input size
class = reshape( class, sy, sx );
```

anomaly_detection

```
function [ out, stats ] = anomaly_detection( in, ignore, stats )
%
% [ out, stats ] = anomaly_detection( in, ignore, stats )
%
% Computed the Mahalanobis distance between each pixel
% and the mean (weighted by the inverse covariance matrix).
%
% REQUIRED INPUTS
% in      3D HSI/MSI image spectral simension is 3rd dimension
%
% OPTIONAL INPUTS
% ignore    2D binary mask indicating what to exclude from statistics (can
%               be empty [] or not included)
% stats     Statistical parameters to use if previsouly computed
%   stats.M   Mean of background
%   stats.CI  Inverse covariance of background
%
```

```
% OUTPUTS
% out       2D M-distance map
% stats     Statistical parameters used (either same as input or
estimated)
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/10/06, 5/12/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% Size and reshape so each row is hyperpixel
[ sy, sx, sz ]=size( in );
in = reshape( in, sy*sx, sz );

if nargin < 3 || isempty(stats) % No stats provided compute

    if nargin == 2 % Ignore mask provided

        % Get valid indices
        I_use = setdiff( [1:sx*sy], find(ignore(:)) );

        % Estimate mean
        stats.M = mean( in( I_use, : ) );

        % Get inverse covariance
        stats.CI = inv( cov( in( I_use, : ) ) );

    else % No ignore mask ptovided

        % Estimate mean
        stats.M = mean( in );

        % Get inverse covariance
        stats.CI = inv( cov(in) );

    end

end


% Compute M-distances
in = in - repmat( stats.M, sx*sy, 1 );
out = sum ( ( ( in * stats.CI ) .* in )' );

% Reshape into 2D image
out = reshape( out, sy, sx );
```

anomaly_detection_class

```
function out = anomaly_detection_class( in, stats, class )
%
% out = anomaly_detection_class( in, stats, class )
%
% Computed the Mahalanobis distance between each pixel
% and the mean (weighted by the inverse covariance matrix).
%
% REQUIRED INPUTS
% in      3D HSI/MSI image spectral simension is 3rd dimension
%
% stats     Statistical parameters to use if previsouly computed
%   stats.M   2D array with class mean vectors for background as rows
%   stats.C   3D array of class covariances for background
%
% OPTIONAL INPUTS
% class     2D class map (default uses Euclidean distance VQ based on
means
%           in stats structure).
%
% OUTPUTS
% out       2D M-distance map
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/12/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.

% Default class map based on Euclidean distance
if nargin < 3 || isempty( class )
    class = clusterhyper( in, stats.M' );
end

% Size and reshape so each row is hyperpixel
[ sy, sx, sz ]=size( in );
in = reshape( in, sy*sx, sz );
out = zeros( sy, sx );
NC = max( class(:) );

for k = 1 : NC

    % Class k indices
    I = find(class(:)==k);

    % Extract class covariance and invert
    CI = inv(stats.C(:,:,k));

    % Compute M-distances
    temp = in(I,:) - repmat( stats.M(k,:), length(I), 1 );
    out(I) = sum ( ( ( temp * CI ) .* temp )' );

end
```

clusterhyper

```
function [class]=clusterhyper(in,A1);
%
% [class]=clusterhyper(in,A1);
%
% VQ clustering program
% for multi- and hyper-spectral images.
% Complement to vqcodehyper.m
%
% in        Input hyperspectral image (3d array)
%
% A1        Final codebook
%
% class     Class map.  Value is index into A1.
%
% Author: Dr. Russell Hardie
% University of Dayton
% 7/3/02


%---------------%
% Get size info %
%---------------%

[sy,sx,nb]=size(in);
nov=sx*sy;
[nb,N]=size(A1);

% Put all the observation vectors in the matrix X as columns
X=reshape(shiftdim(in,2),nb,nov);

% Find distance from each obs vec to one code word at a time
D=zeros(N,nov);

for n=1:N  % loop over codewords
    CWMX=A1(:,n)*ones(1,nov);
    if nb > 1
        D(n,:)=sum((X-CWMX).^2); % distance from each obs to  code word n
    else
        D(n,:)=(X-CWMX).^2;
    end
    n
end

if N~=1
    % assign each obs to codeword with min distance
    [mind,minind]=min(D); % minind = set of codeword assignment indices.
else
    minind = ones(size(D));
end

% form 2d class map
class=reshape(minind,sy,sx);
```

anomaly_detection_likelihood

```
function out = anomaly_detection_likelihood( in, stats )
%
% out = anomaly_detection_likelihood( in, stats )
%
% Computes the likelihood of each hyperpixel in "in" based on the Gaussian
% Mixture Model with parameters specified in "stats".
%
% REQUIRED INPUTS
% in      3D HSI/MSI image spectral simension is 3rd dimension
%
% stats     Statistical parameters to use if previsouly computed
%   stats.M   2D array with class mean vectors for background as rows
%   stats.C   3D array of class covariances for background
%   stats.P   1D array with the a priori probabilities of each class
%
% OUTPUTS
% out       2D likelihood map
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/13/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% Size and reshape so each row is hyperpixel
[ sy, sx, sz ]=size( in );
in = reshape( in, sy*sx, sz );
out = zeros( sy*sx, 1 );
NC = length( stats.P );

for k = 1 : NC

    % pdf constant
    const = 1 / sqrt( ((2*pi)^2)*det( stats.C(:,:,k) ) );

    % Invert covariance matrix
    CI = inv( stats.C(:,:,k) );

    % Compute M-distances
    temp = in - repmat( stats.M(k,:), sx*sy, 1 );
    D = sum ( ( ( temp * CI ) .* temp )' );

    % Update likelihood by adding this component likelihood
    out = out + stats.P(k) * const * exp( -.5*D(:) );

    k

end

% Find zero likelihood pixels (due to precision)
```

```
I_zero = find( out == 0 );
out( I_zero ) = 1 / 1e300;  % very small finite likelihood

% Form log reciprocal
% out = log( 1./ out );

% Form -log likelihood
out = -log( out );

% Reshape output
out = reshape( out, sy, sx );
```

classify_monolithic

```
function class = classify_monolithic( in, stats, cov_type, func_type )
%
% class = classify_monolithic( in, stats, cov_type, func_type )
%
% Generates class assignment for each each pixel in input
% "in" using class statistics provided in "stats".  Various
% options for treating the covariance and the subsequent functions
% are provided to allow flexibility in generating the class boudaries.
%
% REQUIRED INPUTS
% in        3D HSI/MSI image (spectral dimension is 3rd dimension)
%
% stats     Statistical parameters to use if previsouly computed
%   stats.M   2D array with class mean vectors for background as rows
%   stats.C   3D array of class covariances for background
%   stats.P   1D array of class prior probabilities (only needed for
%               posteriori function)
%
% OPTIONAL INPUTS
% cov_type     Type of covariance to use for membership function.  It is
%              assumed that the full covariance is provided in stats.  We
can simplify
%              this as follows:
% 'identity'    Use identiy matrices for covariances (Euclidean distance)
% 'spherical'   Use the average band variance and scale identity mx
% 'diagonal'    Use only the band variances (take diagonal)
% 'full'        (default) Uses the full provided covariances (true M-
distance)
%
% func_type      Type of membership function to use (all normalized to sum
to 1)
%   'post'    = Gaussian a posteriori probability
%                       P_i(x) = pr( class=i | x ) for Gaussian Mixture
%                       Model.  Some outliers may take this metric beyond
%                       the precision limits.
%   'dist'        = (default) M-distance to each class (can be used for
%                       classification and anomaly detection)
%
% OUTPUT
% class        2D class assigments (class map)
```

```
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 6/10/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% Size info
[ sy, sx, sz ]=size( in );
NC = size( stats.C, 3 );
class = zeros( sy*sx, 1 );

% Set up defaults
if nargin == 3

    func_type = 'dist';

elseif nargin == 2

    cov_type = 'full'
    func_type = 'dist';

end

% Reshape input
in =  reshape( in, sx*sy, sz );

% Pre-invert matrices
CI = stats.C;
for k = 1 : NC

    switch cov_type
        case 'identity'
            C = eye( sz );
        case 'spherical'
            C = eye( sz ) * mean( diag( stats.C(:,:,k) ) );
        case 'diagonal'
            C = diag( diag( stats.C(:,:,k) ) );
        case 'full'
            C =  stats.C(:,:,k);
    end

    CI(:,:,k) = inv( C );

end

% Break image into "chunks" to keep from memory overload
chunk = 40000;
M=ceil( sy*sx/chunk );

% Loop over "chunks"
for m = 1 : M
```

```
    disp( sprintf('Processing block %d of %d',m,M) )
    drawnow

    % Subset of hyperpixels to consider
    I = [ (m-1)*chunk+1: min( sy*sx, m*chunk ) ];

    % Initialize the partition membership function
    nov = length( I );
    P = zeros( nov, NC );

    if strcmp( func_type(1:4), 'post' ) % Posteriori probability

        for n = 1 : NC  % loop over classes

            % Compute M-distances
            temp = in(I,:) - repmat( stats.M(n,:), nov, 1 );
            D = sum ( ( ( temp * CI(:,:,n) ) .* temp )' );

            % pdf constant
            const = 1 / sqrt( ((2*pi)^2)*det( stats.C(:,:,n) ) );
            P(:,n) = stats.P(n) * const * exp( -.5*D(:) )';

        end

        [ junk, class(I) ] = max( P' );

    else % M-distance

        for n = 1 : NC  % loop over classes

            % Compute M-distances
            temp = in(I,:) - repmat( stats.M(n,:), nov, 1 );
            D = sum ( ( ( temp * CI(:,:,n) ) .* temp )' );
            P(:,n) = D' ;

        end

        [ junk, class(I) ] = min( P' );

    end

end

% Reshape output
class = reshape( class, sy, sx );
```

RX

```
function D=RX(hsi, w, g)

%   D=RX(hsi, w, g)
%
%  This function performs the RX algorithm on a hyperspectral data cube
%  using a square window of size w x w and guard window of size g x g.
%
%   hsi:                Hyperspectral data cube [bands rows columns]
%
%   w:                  Window size (odd integer)
%
%   g:                  Guard window size (odd integer)
%
%   D:                  Detection statistic image [rows columns]
%
%
%   Author:  Joe Meola
%   Last update:  9 February 2007



% Determine size of hypercube
[bands rows columns]=size(hsi);
hsi=reshape(hsi, bands, rows*columns);

% Used for window indexing
L=(w-1)/2;
G=(g-1)/2;

% Initialize D
D=zeros(rows, columns);



%% Cycle through Data Applying M-dist to windowed area
for m=1:rows
    for n=1:columns

        % Initialize Mask
        H=zeros(rows, columns);

        % Indices of windowed area
        row_idx=m-L:m+L;
        col_idx=n-L:n+L;

        % Indices of guard area
        rg_idx=m-G:m+G;
        cg_idx=n-G:n+G;

        % Only use indices within bounds
        r=row_idx( (row_idx > 0)&(row_idx < (rows+1)) );
        c=col_idx( (col_idx > 0)&(col_idx < (columns+1)) );
```

```
        % Find single index locations
        R=r'*ones(1, length(c));
        C=ones(length(r), 1)*c;
        idx_w = R + (C-1)*rows;
        idx_w=reshape(idx_w, 1, length(r)*length(c));


        % Only use indices within bounds
        rg=rg_idx( (rg_idx > 0)&(rg_idx < (rows+1)) );
        cg=cg_idx( (cg_idx > 0)&(cg_idx < (columns+1)) );


        Rg=rg'*ones(1, length(cg));
        Cg=ones(length(rg), 1)*cg;
        idx_g = Rg + (Cg-1)*rows;
        idx_g=reshape(idx_g, 1, length(rg)*length(cg));

        % Set up mask
        H(idx_w)=1;

        % Remove guard window
        H(idx_g)=0;

        % Select spectra within window
        RW=hsi(:,(H==1));

        % Determine window statistics
        C=cov(RW');
        mn=mean(RW, 2);
        Cinv=inv(C);

        % Apply M-dist to pixel of interest to determine detection stat
        p=m + rows*(n-1);
        D(m,n)=(hsi(:,p) - mn)'*Cinv*(hsi(:,p) - mn);

    end

end
```

B.2 Change Detection Code

change_detection

```
% Change Detection

% Author: Patrick Hytla
% University of Dayton
%
% 11/13/2007
%
% COPYRIGHT © 2007 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all
clc

num_class = 6;
thresh = 200;
kmeans_its5 = 5;
sem_its = 5;

% %%%%%%%%%%%%%%%% Load Data %%%%%%%%%%%%%%%%%%%
load Data\Scene_180az_26_Oct_final_pca.mat
time1=hsi(:,:,1:end);
clear hsi
date1 = '26 Oct 2005';
im(time1(:,:,1))
title([date1]);

% Time 2
load Data\Scene_180az_14_Oct_change_final_pca.mat
time2=hsi(:,:,1:end);
clear hsi
load Masks\target_masks_pca_oct14_change_180az
mask = Ti1|Ti2;
ignore = ( Te1 & ~Ti1 ) | ( Te2 & ~Ti2 );
date2 = '14 Oct 2005 change';
im(time2(:,:,1))
title([date2]);

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%% Prediction %%%%%%%%%%%%%%%%%%%%
% Chronochrome
[linear_predict, L1]  = eq_chronochrome(time1, time2);
out_linear = linear_predict-time2;

stats_kmeans_init = kmeans_hsi(time1, num_class, 100, 1, []);
stats_kmeans_predict = kmeans_hsi(time1, num_class, 100, kmeans_its5, []);
[stats_sem_predict, class_sem_predict, P_sem_predict] =
stochastic_expectation_maximization(time1, sem_its, stats_kmeans_init,
[]);
```

```
% Segmented
% Kmeans
[class_cluster_kmeans]=clusterhyper(time1,stats_kmeans_predict.M');
[cluster_predict_kmeans, L2] = eq_class_chronochrome(time1, time2,
class_cluster_kmeans);
out_cluster_kmeans = cluster_predict_kmeans-time2;

% SEM
[class_cluster_sem]=clusterhyper(time1,stats_sem_predict.M');
[cluster_predict_sem, L3] = eq_class_chronochrome(time1, time2,
class_cluster_sem);
out_cluster_sem = cluster_predict_sem-time2;


% %%%%%%%%%%%% Anomaly Detection %%%%%%%%%%%%%%
% Linear-CBAD
class_vq_pca1 = classify_monolithic(out_linear, stats_kmeans_predict,
'identity', 'dist');
out_linear_cbad = anomaly_detection_class(out_linear,
stats_kmeans_predict, class_vq_pca1);
[pd_linear_cbad,pf_linear_cbad,T_linear_cbad,auc_linear_cbad] =
generate_ROC_curve(out_linear_cbad, mask ,thresh, ignore, 1);

% Linear-GMRX
P = generate_membership_monolithic(out_linear, stats_sem_predict, 'full',
'dist', 2 );
out_linear_gmrx = min(P,[],3);
[pd_linear_gmrx,pf_linear_gmrx,T_linear_gmrx,auc_linear_gmrx] =
generate_ROC_curve(out_linear_gmrx, mask ,thresh, ignore, 1);


% *** Using Cluster Prediction ***
% Kmeans-Mdist
[out_kmeans_mdist, stats_kmeans_mdist] =
anomaly_detection(out_cluster_kmeans,[]);
[pd_kmeans_mdist,pf_kmeans_mdist,T_kmeans_mdist,auc_kmeans_mdist] =
generate_ROC_curve(out_kmeans_mdist, mask, thresh, ignore, 1);

% SEM-Mdist
[out_sem_mdist, stats_sem_mdist] = anomaly_detection(out_cluster_sem,[]);
[pd_sem_mdist,pf_sem_mdist,T_sem_mdist,auc_sem_mdist] =
generate_ROC_curve(out_sem_mdist, mask, thresh, ignore, 1);

% K-means-CBAD
class_vq_pca3 = classify_monolithic(out_cluster_kmeans,
stats_kmeans_predict, 'identity', 'dist');
out_kmeans_cbad = anomaly_detection_class(out_cluster_kmeans,
stats_kmeans_predict, class_vq_pca3);
[pd_kmeans_cbad,pf_kmeans_cbad,T_kmeans_cbad,auc_kmeans_cbad] =
generate_ROC_curve(out_kmeans_cbad, mask ,thresh, ignore, 1);

% SEM-GMRX
P = generate_membership_monolithic(out_cluster_sem, stats_sem_predict,
'full', 'dist', 2 );
out_sem_gmrx = min(P,[],3);
```

```
[pd_sem_gmrx,pf_sem_gmrx,T_sem_gmrx,auc_sem_gmrx] =
generate_ROC_curve(out_sem_gmrx, mask ,thresh, ignore, 1);


figure
plot(pf_linear_cbad,pd_linear_cbad,'-r',pf_linear_gmrx,pd_linear_gmrx,'-
b',pf_kmeans_mdist,pd_kmeans_mdist,'-k',...
     pf_sem_mdist,pd_sem_mdist,'-g',pf_kmeans_cbad,pd_kmeans_cbad,'-
c',pf_sem_gmrx,pd_sem_gmrx,'-m','Linewidth',3)
xlabel('Probability of False Alarm');
ylabel('Probability of Detection');
title('Single clustering routine')
legend('CC-CBAD','CC-GMRX','Kmeans-Mdist','SEM-Mdist','Kmeans-CBAD','SEM-
GMRX','Location','Southeast')
grid
```

eq_chronochrome

```
function [ out, L ] = eq_chronochrome( in1, in2, ignore )
%
% [ out, L ] = eq_chronochrome( in1, in2, ignore )
%
% Background equalization creating "out" by applying a linear mapping of
% "in1" according to the chronochrome algorithm. Statistics from "ignore"
mask region
% are excluded in linear mapping computation.
%
% in1     3D HSI/MSI image (spectral dimension is 3rd dimension)
% in2     3D HSI/MSI image (spectral dimension is 3rd dimension)
% ignore  2D binary mask indicating what to exclude from statistics
% out     3D HSI/MSI formed by linearly transforming "in1" to
%           match "in2"
% L       Linear transformation matrix
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/11/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% Size and reshape so each row is hyperpixel
[ sy, sx, sz ]=size( in1 );

% Get indices of hyperpixels to use
if nargin == 3

    I_ignore = find( ignore(:) );
    I_use = setdiff( [1:sx*sy], I_ignore );
    N_use = length(I_use);

else
```

```
    I_use = [1:sx*sy];
    N_use = sx*sy;

end

% Get means
in1 =  reshape( in1, sx*sy, sz );
in2 =  reshape( in2, sx*sy, sz );

m1 = mean( in1( I_use, : ) );
m2 = mean( in2( I_use, : ) );

m1 = repmat( m1,[sy*sx,1] );
m2 = repmat( m2,[sy*sx,1] );

% Covariance
C1 = ( in1( I_use, : ) - m1( I_use, : ) )' * ...
    ( in1( I_use, : ) - m1( I_use, : ) );
C1=C1/N_use;

% Cross-covariance
C21 = ( in2( I_use, : ) - m2( I_use, : ) )' * ...
    ( in1( I_use, : ) - m1( I_use, : ) );
C21=C21/N_use;

% Compute linear mapping
% L=C21*inv(C1);
L=C21/C1;

% Apply linear mapping
out = L * (in1 - m1 )';
out = reshape( out' + m2 , sy,sx,sz );
```

eq_chronochrome_class

```
function [ out, L ] = eq_class_chronochrome( in1, in2, class, ignore )
%
% [ out, L ] = eq_class_chronochrome( in1, in2, class, ignore )
%
% Background equalization creating "out" by applying a linear mapping of
% "in1" according to the chronochrome algorithm. Statistics from "ignore"
mask region
% are excluded in linear mapping computation.
%
% in1     3D HSI/MSI image (spectral dimension is 3rd dimension)
% in2     3D HSI/MSI image (spectral dimension is 3rd dimension)
% class   2D class map
% ignore  2D binary mask indicating what to exclude from statistics
%
% out     3D HSI/MSI formed by linearly transforming "in1" to
%           match "in2"
%
```

```
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/11/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% Size info
[ sy, sx, sz ]=size( in1 );
NC = max( class (:) );
L = zeros( sz, sz, NC );

% Get indices of hyperpixels to ignore
if nargin == 4

    I_ignore = find( ignore(:) );

else

    I_ignore = [];

end


% Reshape input
in1 =  reshape( in1, sx*sy, sz );
in2 =  reshape( in2, sx*sy, sz );
out = in1; % initialize output

for k = 1 : NC

    % Class k indices
    I = find(class(:)==k);

    % The portion of class k we should not ignore
    I_use = setdiff( I, I_ignore );

    % Get means
    m1 = mean( in1( I_use, : ) );
    m2 = mean( in2( I_use, : ) );
    m1 = repmat( m1,[sy*sx,1] );
    m2 = repmat( m2,[sy*sx,1] );

    % Covariance
    C1 = ( in1( I_use, : ) - m1( I_use, : ) )' * ...
        ( in1( I_use, : ) - m1( I_use, : ) );
    C1=C1/length(I_use);

    % Cross-covariance
    C21 = ( in2( I_use, : ) - m2( I_use, : ) )' * ...
        ( in1( I_use, : ) - m1( I_use, : ) );
    C21=C21/length(I_use);
```

```
    % Compute linear mapping
    % L(:,:,k)=C21*inv(C1);
    L(:,:,k)=C21/C1;

    % Apply linear mapping
    out(I,:) = ( L(:,:,k) * ( in1(I,:) - m1(I,:) )' )' + m2(I,:);

    k

end

% Reshape output
out = reshape( out, sy,sx,sz );
```

B.3 Plotting and Visualization Code

generate_roc_curve

```
function [pd,pf,T,auc,shift] =
generate_ROC_curve(in,truth,num,ignore,log_flag);
%
% [pd,pf,T,auc,shift] = generate_ROC_curve(in,truth,num,ignore,log_flag);
%
% Generate a pixel based ROC curve by thresholding "in" with "num"
thresholds
% and comparing to "truth" except for where the ignore mask is 1.
%
% in            Feature image to threshold for detection
% truth         Truth mask
% num           Number of thresholds to test
% ignore        (OPTIONAL) Mask to ignore when computing stats
% log_flag      (OPTIONAL) 1 to use log spaces thresholds (default is
%                linear)
%
% Author: Russell Hardie, Ph.D.
% University of Dayton
%
% 5/10/06
%
% COPYRIGHT © 2006 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% Size info
S=size( in );

% Default
if nargin < 5 | isempty(log_flag)
    log_flag = 0;
end

% Create threshold array
if log_flag == 1
```

```
    % shift to be > 0 for log space
     shift =  - min(in(:)) + 1;
    in = in + shift;

    % Get thresholds
    T = logspace( log10( min(in(:)) ), log10( max(in(:)) ), num );

else

     shift = 0;
    T = linspace( min(in(:)), max(in(:)), num );

end


if nargin < 4 || isempty( ignore )
   ignore = repmat( logical(0), S );
end

% define the usable true and false masks
use_truth = truth(:) & ~ignore(:);
use_false = ~truth(:) & ~ignore(:);

% Count true and false pixels
num_true = sum( use_truth );
num_false = sum( use_false );

% Loop over threshold
for k = 1 : num

   % Threshold input
   thresh_in = in > T(k);

   % Compute the probability of detection and false alarm
   pd(k) = sum( thresh_in(:) & use_truth )/num_true;
   pf(k) = sum( thresh_in(:) & use_false )/num_false;

end

% Area under curve
W=-diff(pf);
H=.5*diff(pd)+pd(1:end-1);
auc = sum( W.*H );

% Compensate T for shift so threshold can be applied to raw input
T = T - shift;
```

faster_roc

```
function [fp_frac, tp_frac , thresh_val] = faster_roc( test_statistic,
actual_labels, plotflag )
%
% [fp_frac, tp_frac , thresh_val] = faster_roc( test_statistic,
actual_labels, plotflag )
%
% ROC curve from decision statistic and labels.
% Based on function by Marty Disimio
% Modified by Russell Hardie 9/1/06, 9/4/06
%
% INPUTS:
% test_statistic = vector of scores output from a classifier
% actual_labels = actual labels
%               1 is associated with a tp
%               0 is associated with an fp
% plotflag: if 1 (default), plot; if 0, don't plot
%
%  OUTPUTS:
%  fp_frac  =  fraction of fp's passing a given threshold
%  tp_frac =  fraction tp's passing a given threshold
%  thresh_val = vector of threshold values


% Plot flag default
if ~exist( 'plotflag' ,'var')
    plotflag = 1;
end

% Sort decision statistic and corresponding labels
[sorted_test_statistic, sorted_idx] = sort( test_statistic(:) , 'descend'
);
sorted_labels = actual_labels( sorted_idx );

% Save effective thresholds used for ROC
thresh_val = sorted_test_statistic;
thresh_val = [ inf; thresh_val; -inf ];

% Compute the number of TPs and FPs for each effective threshold
tp_frac = cumsum( sorted_labels(:) == 1 );
fp_frac = cumsum( sorted_labels(:) == 0 );

% Divide by the total number of TPs and FPs to get rates
num_tp =  sum( actual_labels == 1 );
num_fp = sum( actual_labels == 0 );

tp_frac = tp_frac  / num_tp ;
fp_frac = fp_frac / num_fp;

% Add the threshold = -infinity and +infinity results
tp_frac = [0; tp_frac ; 1 ];
fp_frac = [0; fp_frac ; 1 ];
```

```
% Plot
if plotflag
    figure
    plot( fp_frac, tp_frac, 'b-' )
    xlabel('FP Fraction '); ylabel('TP Fraction')
    axis([ 0 1 0 1]);
    grid
end
```

im

```
function [ h, out, gain, bias ] = im( in, new_fig, ...
    exclude_zero, nstd, x, y )
%
% [ h, out, gain, bias ] = im( in, new_fig, exclude_zero, nstd, x, y )
%
%   Image display function that can be used for displaying most binary,
%   grayscale or color image with auto-scaling
%
% h     Handle to image object
% out   Output image scaled for display
% gain  Effective gain applied to display data
% bias  Effective bias applied to display data
%
% in          Input image
% new_fig     New figure flag, generates new fig if set to 1 (defualt = 1)
%             -1 means do not display image (calculate output only).
% exclude_zero  Exclude 0 flag, if set to 1, it will ignore 0 values when
%               computing statistics.  This is helpful when an image
%               has a zero border for example.
% nstd        Number of standard deviations to map to display range
%             (default = 2).
%
% x           x axis values for display (default = [1:sx])
% y           y axis values for display (default = [1:sy])
%
% Author: Dr. Russell Hardie
% University of Dayton
% 5/2005
%
% COPYRIGHT © 2005 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.

% Convert to double
in = double(in);

% Get size info
[ sy, sx, sz ] = size( in );

% Is the logical (binary) pixel data?
if islogical(in) || (min(in(:))==0 && max(in(:))==1)
    binary_flag = 1;
else
    binary_flag = 0;
end
```

```
% Set up the default axes
if nargin < 6 || isempty(x) || isempty(y)
    x=[1:sx]; % default
    y=[1:sy]; % default
end

% Open new figure window if indicated
if nargin < 2 || isempty(new_fig)
    new_fig = 1;
end

if new_fig == 1;
    figure % default
end

if nargin < 3 || isempty(exclude_zero)
    exclude_zero = 0; % default
end

% initialize handle
h = [];

% number of pixels to use for statistics (mean and std)
max_num = min( [ 10000, sy*sx ] );

if binary_flag == 1

    % Treat by scaling 0 -> 0 and 1->255

    out = in(:,:,1)*255;
    gain = 255;
    bias = 0;

    if new_fig > -1
        h=image(x,y,out);
        % image(x,y,out);
        colormap(gray(256));
        axis('image');
    end

else

    % Treat by mapping nstd standard deviations
    % of input range into display range

    if nargin < 4 || isempty(nstd)
        nstd = 2; % default
    end

    if sz >= 3 % color data

        R = in(:,:,1);
        G = in(:,:,2);
```

```
        B = in(:,:,3);

        if exclude_zero == 1

            I = find( R~=0 | G~=0 | B~=0 );
            USE = round( linspace(1,length(I),max_num ) );

            mR = mean(R(I(USE)));
            mG = mean(G(I(USE)));
            mB = mean(B(I(USE)));

            sR = std(R(I(USE)));
            sG = std(G(I(USE)));
            sB = std(B(I(USE)));

        else

            USE = round( linspace(1,sy*sx,max_num ) );

            mR = mean(R(USE));
            mG = mean(G(USE));
            mB = mean(B(USE));

            sR = std(R(USE));
            sG = std(G(USE));
            sB = std(B(USE));

        end

        gain(1) = 128/(sR*nstd);
        gain(2) = 128/(sG*nstd);
        gain(3) = 128/(sB*nstd);

        bias(1) = 128-(mR*128)/(sR*nstd);
        bias(2) = 128-(mG*128)/(sG*nstd);
        bias(3) = 128-(mB*128)/(sB*nstd);

        R = in(:,:,1)*gain(1)+bias(1);
        G = in(:,:,2)*gain(2)+bias(2);
        B = in(:,:,3)*gain(3)+bias(3);

        out(:,:,1)=uint8(clip(round(R),0,255));
        out(:,:,2)=uint8(clip(round(G),0,255));
        out(:,:,3)=uint8(clip(round(B),0,255));

        if new_fig > -1
            h=image(x,y,out);
            % image(x,y,out);
            axis('image');
        end

    else % assume grayscale (use first band only)

        in = in(:,:,1);
```

```
        if exclude_zero == 1

            I = find(in~=0);
            USE = round( linspace(1,length(I),max_num ) );

            inmn=mean(in(I(USE)));
            instd=std(in(I(USE)));

        else

            USE = round( linspace(1,sy*sx,max_num ) );

            inmn=mean(in(USE));
            instd=std(in(USE));

        end

        gain = 128/(instd*nstd);
        bias = 128-(inmn*128)/(instd*nstd);
        out=in*gain+bias;

        if new_fig > -1
            h=image(x,y,out);
            % image(x,y,out);
            colormap(gray(256));
            axis('image');
        end

    end

end
```

scatter_plot

```
function scatter_plot(data,cmap,tmap,k1,k2)

%
% scatter_plot(data,cmap,tmap,k1,k2)
%
% This function produces a scatter plot of two selected bands of
% hyperspectral data for specified background and target classes
%
% Formats: data(nb,nr,nc)
%          cmap(nr,nc)
%          tmap(nr,nc)

[nb,nr,nc]=size(data);
nbkg=max(cmap(:));
ntgt=max(tmap(:));
ntot=nbkg+ntgt;
band1=band(data,k1);
band2=band(data,k2);
```

```
count=1;
bmark=['.r'; '.c'; '.b'; '.g'; '.y'; '.m'; '.r'; ];
tmark=['+c'; '+r'; '+b'; '+g'; '+m'; '+y'; '+k'];
if nbkg > 0
    if nbkg == 1
        x=band1(find(cmap==1));
        y=band2(find(cmap==1));
        plot(x,y,'k.','MarkerSize', 2);
        hold on;
    else
        for i=1:nbkg
            x=band1(find(cmap==i));
            y=band2(find(cmap==i));
            plot(x,y,bmark(i,:),'MarkerSize', 2);
            hold on;
        end
    end
end

if ntgt > 0
    for i=1:ntgt
        x=band1(find(tmap==i));
        y=band2(find(tmap==i));
        plot(x,y,tmark(i,:),'MarkerSize', 2);
        hold on;
    end
end
hold off;
```

anomaly_detection_methods_180az_avg_auc

```
% Create averaged AUC plots for each method w.r.t number of classes (2-10)
% for 180az data.

% Author: Patrick Hytla
% University of Dayton
%
% 6/18/2007
%
% COPYRIGHT © 2007 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.
clear all
%close all
clc

start =15;
stop = 31;
avg = stop-start+1;

load RX_180az_g81_w141_full_auc_only.mat      % load RX auc data


% %%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%% Load Class Data %%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% 2 classs
load 2class_180az_merged  % load merged data
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
auc_rx_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_2class = auc_M_temp/avg;            % Averaging
auc_cbad_avg_2class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class =  auc_gmrx_temp/avg;
auc_fcbad_2class = auc_fcbad_temp/avg;
auc_rx_avg_2class = auc_rx_temp/avg;
auc_fcbad_fuzzy_2class = auc_fcbad_fuzzy_temp/avg;

% 3 Class
load 3class_180az_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_3class = auc_M_temp/avg;            % Averaging
auc_cbad_avg_3class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_3class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_3class =  auc_gmrx_temp/avg;
auc_fcbad_3class = auc_fcbad_temp/avg;
auc_rx_avg_3class = auc_rx_temp/avg;
auc_fcbad_fuzzy_3class = auc_fcbad_fuzzy_temp/avg;

% 4 Class
load 4class_180az_merged
```

```
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_4class = auc_M_temp/avg;            % Averaging
auc_cbad_avg_4class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_4class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_4class =  auc_gmrx_temp/avg;
auc_fcbad_4class = auc_fcbad_temp/avg;
auc_rx_avg_4class = auc_rx_temp/avg;
auc_fcbad_fuzzy_4class = auc_fcbad_fuzzy_temp/avg;

% 5 Class
load 5class_180az_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_5class = auc_M_temp/avg;            % Averaging
auc_cbad_avg_5class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_5class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_5class =  auc_gmrx_temp/avg;
auc_fcbad_5class = auc_fcbad_temp/avg;
auc_rx_avg_5class = auc_rx_temp/avg;
auc_fcbad_fuzzy_5class = auc_fcbad_fuzzy_temp/avg;

% 6 Class
load 6class_180az_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
```

```
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_6class = auc_M_temp/avg;           % Averaging
auc_cbad_avg_6class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_6class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_6class =  auc_gmrx_temp/avg;
auc_fcbad_6class = auc_fcbad_temp/avg;
auc_rx_avg_6class = auc_rx_temp/avg;
auc_fcbad_fuzzy_6class = auc_fcbad_fuzzy_temp/avg;

% 7 Class
load 7class_180az_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_7class = auc_M_temp/avg;           % Averaging
auc_cbad_avg_7class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_7class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_7class =  auc_gmrx_temp/avg;
auc_fcbad_7class = auc_fcbad_temp/avg;
auc_rx_avg_7class = auc_rx_temp/avg;
auc_fcbad_fuzzy_7class = auc_fcbad_fuzzy_temp/avg;

% 8 Class
load 8class_180az_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;
```

```
auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_8class = auc_M_temp/avg;           % Averaging
auc_cbad_avg_8class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_8class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_8class =  auc_gmrx_temp/avg;
auc_fcbad_8class = auc_fcbad_temp/avg;
auc_rx_avg_8class = auc_rx_temp/avg;
auc_fcbad_fuzzy_8class = auc_fcbad_fuzzy_temp/avg;

% 9 Class
load 9class_180az_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_9class = auc_M_temp/avg;           % Averaging
auc_cbad_avg_9class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_9class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_9class =  auc_gmrx_temp/avg;
auc_fcbad_9class = auc_fcbad_temp/avg;
auc_rx_avg_9class = auc_rx_temp/avg;
auc_fcbad_fuzzy_9class = auc_fcbad_fuzzy_temp/avg;

% 10 Class
load 10class_180az_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_rx_temp = 0;
auc_fcbad_fuzzy_temp = 0;

auc_M_temp = sum(auc_M(start:stop));            % Sum AUC's
auc_cbad_temp = sum(auc_cbad5(start:stop));
```

```
auc_gmm_sem_k1_5it_temp = sum(auc_gmm_sem_k1_5it(start:stop));
auc_gmrx_temp = sum(auc_gmrx(start:stop));
auc_fcbad_temp = sum(auc_fcbad(start:stop));
auc_rx_temp = sum(auc_rx(start:stop));
auc_fcbad_fuzzy_temp = sum(auc_fcbad_fuzzy(start:stop));

auc_M_avg_10class = auc_M_temp/avg;            % Averaging
auc_cbad_avg_10class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_10class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_10class =  auc_gmrx_temp/avg;
auc_fcbad_10class = auc_fcbad_temp/avg;
auc_rx_avg_10class = auc_rx_temp/avg;
auc_fcbad_fuzzy_10class = auc_fcbad_fuzzy_temp/avg;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%
% %%%%% Averaging %%%%%
% %%%%%%%%%%%%%%%%%%%%%

t = [2:10];

auc_M_avg_all_class = [auc_M_avg_2class auc_M_avg_3class auc_M_avg_4class
auc_M_avg_5class auc_M_avg_6class auc_M_avg_7class ...
  auc_M_avg_8class auc_M_avg_9class auc_M_avg_10class];

auc_cbad_avg_all_class = [auc_cbad_avg_2class auc_cbad_avg_3class
auc_cbad_avg_4class auc_cbad_avg_5class auc_cbad_avg_6class
auc_cbad_avg_7class ...
  auc_cbad_avg_8class auc_cbad_avg_9class auc_cbad_avg_10class];

auc_gmm_sem_k1_5it_avg_all_class = [auc_gmm_sem_k1_5it_avg_2class
auc_gmm_sem_k1_5it_avg_3class auc_gmm_sem_k1_5it_avg_4class
auc_gmm_sem_k1_5it_avg_5class auc_gmm_sem_k1_5it_avg_6class
auc_gmm_sem_k1_5it_avg_7class ...
  auc_gmm_sem_k1_5it_avg_8class auc_gmm_sem_k1_5it_avg_9class
auc_gmm_sem_k1_5it_avg_10class];

auc_rx_avg_all_class = [auc_rx_avg_2class auc_rx_avg_3class
auc_rx_avg_4class auc_rx_avg_5class auc_rx_avg_6class auc_rx_avg_7class
auc_rx_avg_8class auc_rx_avg_9class auc_rx_avg_10class];

auc_gmrx_avg_all_class = [auc_gmrx_2class auc_gmrx_3class auc_gmrx_4class
auc_gmrx_5class auc_gmrx_6class auc_gmrx_7class auc_gmrx_8class
auc_gmrx_9class auc_gmrx_10class];

auc_fcbad_avg_all_class = [auc_fcbad_2class auc_fcbad_3class
auc_fcbad_4class auc_fcbad_5class auc_fcbad_6class auc_fcbad_7class
auc_fcbad_8class auc_fcbad_9class auc_fcbad_10class];

auc_fcbad_fuzzy_avg_all_class = [auc_fcbad_fuzzy_2class
auc_fcbad_fuzzy_3class auc_fcbad_fuzzy_4class auc_fcbad_fuzzy_5class
auc_fcbad_fuzzy_6class auc_fcbad_fuzzy_7class auc_fcbad_fuzzy_8class
auc_fcbad_fuzzy_9class auc_fcbad_fuzzy_10class];
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%
% %%%%% Plotting %%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%

figure
plot(t,auc_M_avg_all_class,'k-*','Linewidth',2)
hold on
plot(t,auc_cbad_avg_all_class,'r-*','Linewidth',2)
plot(t,auc_gmm_sem_k1_5it_avg_all_class,'g-*','Linewidth',2)
plot(t,auc_rx_avg_all_class,'c-*','Linewidth',2)
plot(t,auc_gmrx_avg_all_class,'m-*','Linewidth',2)
plot(t,auc_fcbad_avg_all_class,'b-*','Linewidth',2)
%plot(t,auc_fcbad_fuzzy_avg_all_class,'y-*')
hold off
grid
%legend('Global Anomaly','CBAD','GMM','RX', 'GMRX','FCBAD','FCBAD
fuzzy','Location','SouthEast')
legend('M-dist','CBAD','GMM','RX', 'GMRX','FCBAD','Location','SouthEast')
xlabel('Numer of Classes')
ylabel('Average AUC')
%title('180az data: Performance vs. Number of Classes (tfa = 0)');
```

anomaly_detection_methods_180az_avg_auc_tfa

```
% Create averaged AUC plots with varying target fraction allowed
% into the background statistics (tfa) for each method w.r.t number of
% classes (2-6).  For 180az data.

% Author: Patrick Hytla
% University of Dayton
%
% 6/18/2007
%
% COPYRIGHT © 2007 THE UNIVERSITY OF DAYTON.  ALL RIGHTS RESERVED.


% %%%%%%%%%%%%%%%%%
% %%%%% Inputs %%%%%
% %%%%%%%%%%%%%%%%%

% start            which data set to start on (1-36)

% stop             which data set to stop on (1-36).

% %%%%%%%%%%%%%%%%%

clear all
%close all
clc
```

```
start = 15;          % start 1 beginning, 14 shadow beginning
stop = 31;           % stop 13 last good, 31 last shadow
avg = stop+1-start;

% %%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%% Load Class Data %%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%

% 2 classs
load 2class_180az_merged
auc_M_temp = 0;                    % set temp variables to zero
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);            % sum AUC's
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class = auc_M_temp/avg;             % averaging
auc_cbad_avg_2class = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class =  auc_gmrx_temp/avg;
auc_fcbad_2class = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa10_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa10 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa10 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa10 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa10 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa10 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa10 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa20_merged
```

```
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa20 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa20 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa20 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa20 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa20 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa20 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa30_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa30 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa30 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa30 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa30 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa30 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa30 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa40_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
```

```
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa40 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa40 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa40 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa40 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa40 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa40 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa50_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa50 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa50 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa50 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa50 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa50 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa50 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa60_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa60 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa60 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa60 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa60 =  auc_gmrx_temp/avg;
```

```
auc_fcbad_2class_tfa60 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa60 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa70_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa70 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa70 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa70 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa70 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa70 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa70 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa80_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa80 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa80 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa80 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa80 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa80 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa80 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa90_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
```

```
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa90 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa90 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa90 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa90 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa90 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa90 = auc_fcbad_fuzzy_temp/avg;

load 2class_180az_all_target_tfa100_merged
auc_M_temp = 0;
auc_cbad_temp = 0;
auc_gmm_sem_k1_5it_temp = 0;
auc_gmrx_temp = 0;
auc_fcbad_temp = 0;
auc_fcbad_fuzzy_temp = 0;
for i = start:stop
  auc_M_temp = auc_M_temp + auc_M(i,1);
  auc_cbad_temp = auc_cbad_temp + auc_cbad5(i,1);
  auc_gmm_sem_k1_5it_temp = auc_gmm_sem_k1_5it_temp +
auc_gmm_sem_k1_5it(i,1);
  auc_gmrx_temp = auc_gmrx_temp + auc_gmrx(i,1);
  auc_fcbad_temp = auc_fcbad_temp + auc_fcbad(i,1);
  auc_fcbad_fuzzy_temp = auc_fcbad_fuzzy_temp + auc_fcbad_fuzzy(i,1);
end
auc_M_avg_2class_tfa100 = auc_M_temp/avg;
auc_cbad_avg_2class_tfa100 = auc_cbad_temp/avg;
auc_gmm_sem_k1_5it_avg_2class_tfa100 = auc_gmm_sem_k1_5it_temp/avg;
auc_gmrx_2class_tfa100 =  auc_gmrx_temp/avg;
auc_fcbad_2class_tfa100 = auc_fcbad_temp/avg;
auc_fcbad_fuzzy_2class_tfa100 = auc_fcbad_fuzzy_temp/avg;

% merge
auc_M_avg_2class_tfa = [auc_M_avg_2class auc_M_avg_2class_tfa10
auc_M_avg_2class_tfa20 auc_M_avg_2class_tfa30 auc_M_avg_2class_tfa40...
auc_M_avg_2class_tfa50 auc_M_avg_2class_tfa60 auc_M_avg_2class_tfa70
auc_M_avg_2class_tfa80 auc_M_avg_2class_tfa90 auc_M_avg_2class_tfa100];

auc_cbad_avg_2class_tfa = [auc_cbad_avg_2class auc_cbad_avg_2class_tfa10
auc_cbad_avg_2class_tfa20 auc_cbad_avg_2class_tfa30
auc_cbad_avg_2class_tfa40...
auc_cbad_avg_2class_tfa50 auc_cbad_avg_2class_tfa60
auc_cbad_avg_2class_tfa70 auc_cbad_avg_2class_tfa80...
auc_cbad_avg_2class_tfa90 auc_cbad_avg_2class_tfa100];

auc_gmm_sem_k1_5it_avg_2class_tfa = [auc_gmm_sem_k1_5it_avg_2class
auc_gmm_sem_k1_5it_avg_2class_tfa10 auc_gmm_sem_k1_5it_avg_2class_tfa20
auc_gmm_sem_k1_5it_avg_2class_tfa30 auc_gmm_sem_k1_5it_avg_2class_tfa40...
```

```
auc_gmm_sem_k1_5it_avg_2class_tfa50 auc_gmm_sem_k1_5it_avg_2class_tfa60
auc_gmm_sem_k1_5it_avg_2class_tfa70 auc_gmm_sem_k1_5it_avg_2class_tfa80...
auc_gmm_sem_k1_5it_avg_2class_tfa90 auc_gmm_sem_k1_5it_avg_2class_tfa100];

auc_gmrx_avg_2class_tfa = [auc_gmrx_2class auc_gmrx_2class_tfa10
auc_gmrx_2class_tfa20 auc_gmrx_2class_tfa30 auc_gmrx_2class_tfa40...
auc_gmrx_2class_tfa50 auc_gmrx_2class_tfa60 auc_gmrx_2class_tfa70
auc_gmrx_2class_tfa80 auc_gmrx_2class_tfa90 auc_gmrx_2class_tfa100];

auc_fcbad_avg_2class_tfa = [auc_fcbad_2class auc_fcbad_2class_tfa10
auc_fcbad_2class_tfa20 auc_fcbad_2class_tfa30 auc_fcbad_2class_tfa40...
auc_fcbad_2class_tfa50 auc_fcbad_2class_tfa60 auc_fcbad_2class_tfa70
auc_fcbad_2class_tfa80 auc_fcbad_2class_tfa90 auc_fcbad_2class_tfa100];

auc_fcbad_fuzzy_avg_2class_tfa = [auc_fcbad_fuzzy_2class
auc_fcbad_fuzzy_2class_tfa10 auc_fcbad_fuzzy_2class_tfa20
auc_fcbad_fuzzy_2class_tfa30 auc_fcbad_fuzzy_2class_tfa40...
auc_fcbad_fuzzy_2class_tfa50 auc_fcbad_fuzzy_2class_tfa60
auc_fcbad_fuzzy_2class_tfa70 auc_fcbad_fuzzy_2class_tfa80
auc_fcbad_fuzzy_2class_tfa90 auc_fcbad_fuzzy_2class_tfa100];


% Plot
t = linspace(0,100,11);
figure
plot(t,auc_M_avg_2class_tfa,'k-*','Linewidth',2)
hold on
plot(t,auc_cbad_avg_2class_tfa,'r-*','Linewidth',2)
plot(t,auc_gmm_sem_k1_5it_avg_2class_tfa,'g-*','Linewidth',2)
plot(t,auc_gmrx_avg_2class_tfa,'m-*','Linewidth',2)
plot(t,auc_fcbad_avg_2class_tfa,'b-*','Linewidth',2)
hold off
grid
xlabel('Percent of Target Contamination')
ylabel('Average AUC')
legend('M-dist','2 class CBAD', '2 class GMM','2 class GMRX','2 class
FCBAD','Location','Southeast')
```

REFERENCES

1) Shaw, Gary, and Dimitris Manolakis. "Signal Processing for Hyperspectral Image Exploitation." *IEEE Signal Processing Magazine* January 2002: 12-16.

2) D. Manolakis and G. Shaw, "Detection Algorithms for Hyperspectral Imaging Applications," *IEEE Signal Processing Magazine*, Jan 2002, pp. 29-43.

3) M. J. Carlotto, "Detection and Analysis of Change in Remotely Sensed Imagery with Application to Wide Area Surveillance," *IEEE Transactions on Image Processing*, Vol. 6 No. 1, Jan 1997, pp. 189-202.

4) A. Schaum and A. Stocker, "Subclutter Target Detection Using Sequences of Thermal Infrared Multispectral Imagery," in *SPIE, Algorithms for Multispectral and Hyperspectral Imagery*, vol. 3071, S. S. Shen, M. R. Descour, Eds., 1997, pp. 12-22.

5) Headwall Photonics. *Hyperspec VS Family of Imaging Spectrographs*. www.headwallphotonics.com .

6) DALSA Digital Imaging. *Pantera TF 1M60 Area Scan Cameras*. www.dalsa.com .

7) Aerotech. *ADRS Series Mechanical Bearing Rotary Stages*. www.aerotech.com .

8) Simi, Christopher, et al. "Night Vision Imaging Spectrometer (NVIS) Calibration and Configuration: Recent Developments." *SPIE* Vol. 4381 (2001): 109:115.

9) Newport. *HeNe Lasers*. www.newport.com .

10) Newport. *Oriel Pencil Style Calibration Lamps*. www.newport.com .

11) D'Agostino, John A., and Curtis Webb. "3-D Analysis and Measurement Methodology for Imaging System Noise." *SPIE*. Vol. 1488 (1991): 110-117.

12) Holst, Gerald. *CCD Arrays Cameras and Displays*. 2nd ed. Winter Park, FL: JCD Publishing, 1998.

13) Labsphere. "A Guide to Integrating Sphere Radiometry and Photometry." *Techguide*. www.labsphere.com .

14) Pedrotti, Frank L., and Leno S. Pedrotti. *Introduction to Optics*. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1993.

15) Meola, Joseph. *Analysis of Hyperspectral Change and Target Detection as Affected by Vegetation and Illumination Variations*. Thesis. University of Dayton, 2006.

16) Solar Position Calculator. *National Oceanic and Atmospheric Administration Surface Radiation Research Branch*. http://www.srrb.noaa.gov/highlights/sunrise/azel.html .

17) Richards, John A. *Remote Sensing Digital Image Analysis: An Introduction*. 2nd ed. New York, NY: Springer-Verlag, 1993.

18) D. W. Stein, *et al*, "Anomaly Detection from Hyperspectral Imagery," *IEEE Signal Proc. Magazine*, Jan 2002, pp. 58-69.

19) M. T. Eismann and R. C. Hardie, "Initialization and Convergence of the Stochastic Mixing Model," *Proceedings of the SPIE*, Vol. 5159, 2003.

20) M. T. Eismann and R. C. Hardie, "Application of the stochastic mixing model to hyperspectral resolution enhancement," *IEEE Trans. On Geoscience and Remote Sensing*, Vol. 42 pp. 1924-1933, Sept. 2004.

21) L.S. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution," IEEE *Trans. On Acoustics, Speech, Signal Processing*, vol. 38, pp. 1760-1770, Oct 1990.

22) M.J. Carlotto, "A Cluster Based Approach for Detecting Man-Made Objects and Changes in Imagery," *IEEE Trans. On Geoscience and Remote Sensing*, vol. 43, no. 2, pp. 374-387, Feb. 2005.

23) A. Schaum and A. Stocker, "Advanced Algorithms for Autonomous Hyperspectral Change Detection," *Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop (AIPR'04)*, 2004.

24) M.T. Eismann, J. Meola and R.C. Hardie, "Hyperspectral Change Detection in the Presence of Diurnal and Seasonal Variations," *IEEE Trans. On Geoscience and Remote Sensing*, vol. 46, Jan. 2008.

25) The MathWorks. *Matlab and Simulink for Technical Computing*. www.mathworks.com.

R002593606