3-9-2009

Exploitation of Intra-Spectral Band Correlation for Rapid Feacture Selection and Target Identification in Hyperspectral Imagery

Michael K. Miller

EXPLOITATION OF INTRA-SPECTRAL BAND CORRELATION FOR RAPID FEATURE SELECTION, AND TARGET IDENTIFICATION IN HYPERSPECTRAL IMAGERY

THESIS

Michael K. Miller, Major, USAF

AFIT/GOR/ENS/09-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/09-10

EXPLOITATION OF INTRA-SPECTRAL BAND CORRELATION FOR RAPID FEATURE SELECTION, AND TARGET IDENTIFICATION IN HYPERSPECTRAL IMAGERY

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Michael K. Miller, MS

Major, USAF

March 2009

AFIT/GOR/ENS/09-10

EXPLOITATION OF INTRA-SPECTRAL BAND CORRELATION FOR RAPID FEATURE SELECTION, AND TARGET IDENTIFICATION IN HYPERSPECTRAL IMAGERY

Michael K. Miller, MS
Major, USAF

Approved:

_____	_____
Dr. Kenneth. W. Bauer (Chairman)	date
_____	_____
Dr. John O. Miller (Member)	date

Abstract

The application of statistical techniques to hyperspectral imagery for the purpose of detecting anomalies or "targets" has drawn significant attention over the last decade. Multivariate techniques such as principal components analysis and factor analysis have been demonstrated effective at reducing hyperspectral image dimensionality from hundreds of spectral bands to a handful of bands capable of explaining the vast majority of the variance contained in the image. Furthermore the development of techniques such as Independent Components Analysis (ICA), Vertex Components Analysis (VCA), and Non-negative Matrix Factorization (NNMF) have provided improved means to more clearly separate spectra from one another. One common problem with such techniques is that typically an iterative algorithm must meet some predefined convergence criteria before terminating. High dimensional hyperspectral data sets result in prohibitively high computational expense for any real time application. As such some method for dimensionality reduction is desirable so as to speed the process of target location.

Clustering of individual pixels in the spatial dimension has been applied by mechanisms such as K-means, X-means, and ISODATA clustering algorithms (Williams: 2007). Exploitation of these algorithms is hindered by the fact that they begin with the assumption of no a priori knowledge of how the image will be clustered. This requires comparison of pixels and estimation of optimal clustering based on some minimization of the distance between pixels within the clusters, which in turn dictates a computationally expensive iterative process. Furthermore these approaches require a target number of clusters be provided as an input to the algorithm which may not be entirely practical in

many cases, and generally hampers any implementation of the common clustering algorithms into automated anomaly detection schemes. As an alternative to spatial clustering this paper proposes and demonstrates an implementation of spectral based clustering to determine and reduce a hyperspectral image dimensionality. While no a priori knowledge is likely in the spatial dimension, along the spectral dimensions, a certain amount of knowledge may be expected. In particular it may be expected that adjacent spectral bands are most highly correlated and that well separated bands are generally unlike one another. As a result very few comparisons are required to cluster spectral bands, and reasonable anomaly detection may be achieved by retaining the average spectra for only those bands which meet a predefined level of correlation between one another. An algorithm is provided which performs spectral clustering on eight real images and results are compared to Johnson's AutoGAD algorithm [2007].

To my wife, daughter, and son

A very special thanks goes to the three people who gave up the most during this endeavor. Your unceasing love and support through the entire process was more valuable and mean more to me than I can fully express in words.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Kenneth Bauer, for his guidance and support throughout the course of this thesis effort. The insight and experience was instrumental in this process. I would, also, like to thank my reader, Dr. J.O. Miller for his patient yet timely efforts in identifying ways to improve the end written product.

I would finally like to thank my fellow graduate, Capt. Matt Davis, who provided advice, support, and friendship. Your contributions to this work could not be overstated. Thank you.

Michael K. Miller, Major, USAF

Table of Contents

Page

List of Figures

List of Tables

NONNEGATIVE MATRIX FACTORIZATION AND EXPLOITATION OF -SPECTRAL BAND CORRELATION FOR RAPID FEATURE SELECTION, AND TARGET IDENTIFICATION IN HYPERSPECTRAL IMAGERY

I. Introduction

1.1 Background

The application of Hyperspectral Imagery (HSI) to Intelligence Surveillance and Reconnaissance has steadily progressed over the past decade, both in commercial-medical arenas and in military applications. Of particular interest to the Department of Defense is the capability of HSI to isolate spectrally distinct objects from surrounding detail, thereby locating potential targets by exploiting their distinct electromagnetic reflectance. This sets HSI apart from many other means of target detection as a completely passive scheme as opposed to more active means such as Synthetic Aperture Radar which can be detected when in use. The advantage presented by HSI was substantially advanced on 19 July 2000 with the launch of MightySat II, the first USAF satellite primarily dedicated to hyperspectral remote sensing (Brownlee: 2000:1).

Before proceeding further a brief explanation of the portion of the electromagnetic spectrum involved in HSI and the operations involved in handling hyperspectral data is in order. While the visual portion of the electromagnetic spectrum is familiar, the bands beyond the visual spectrum are only recently becoming commonly utilized in a variety of applications including the medical field, geologic surveys, and military applications. These bands include both the three visible bands mentioned earlier, but also bands beyond both ends of the visible spectrum, including the ultraviolet (UV), and infrared (IR) portions of the spectrum. These bands are of great practical use in passive remote sensing as while the frequencies are invisible to the human eye they are present in natural light and thus require no artificial illumination be applied.

Furthermore objects can be detected and identified by the frequencies they reflect much in the same way we visually recognize the difference between red, blue, and green. For example materials with high water (like foliage) content tend to absorb infrared light differently than materials with low water content (like metals). It is these differences in absorption and reflectance we exploit over a wide range of the spectrum, when applying hyperspectroscopy.

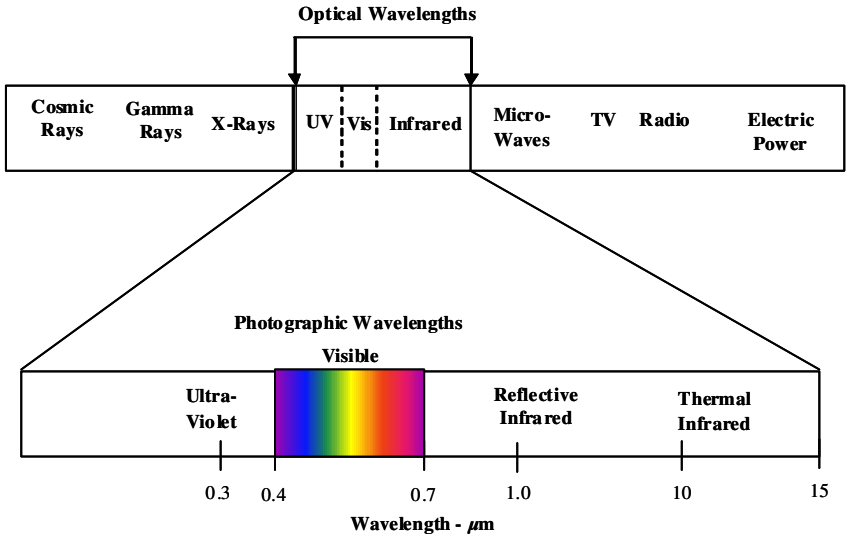


Figure 1-1. Hyperspectral Electromagnetic Range (Landgrebe, 2003:14)

An ongoing challenge to the application of hyperspectral imagery for target detection is simply the volume of data produced by the application. While ordinary photographic imagery presents data only across three components of the visual band (red, green, and blue) over an area consisting of some predefined number of pixels, HSI scans a predefined number of pixels over a substantially wider range of narrow bands within the electromagnetic spectrum. So while ordinary color photographic processes produce three layers (or matrices) of data, indicating the reflectance of light across the red, green, and blue bands, hyperspctral images contain hundreds of layers of reflectance data, each indicating the reflectance within separate narrow portions of

the electromagnetic spectrum. The resultant data set is frequently referred to as a "data cube" as the image size may be on the order of 200 by 200 pixels spatially, and may contain upwards of 200 frequency bands, or layers to the image.

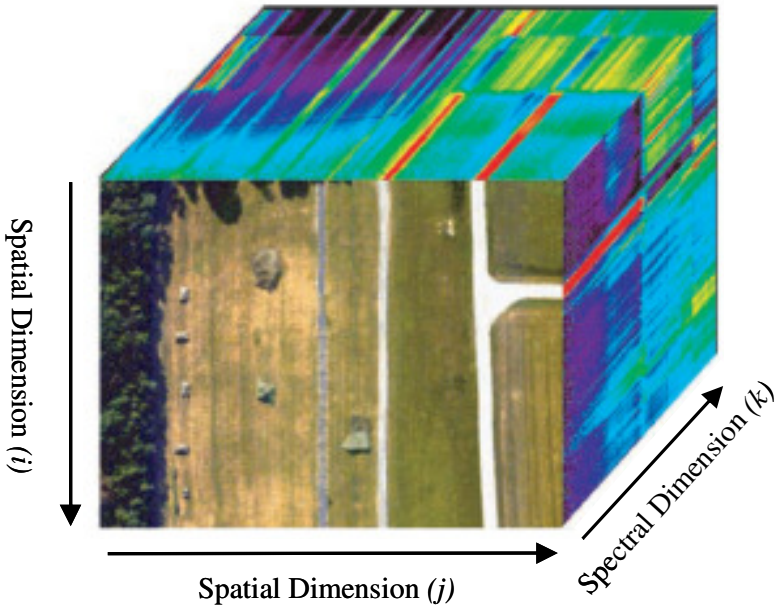


Figure 1-2. Hyperspectral Image/Data Cube (Shaw and Manolakis, 2003:13)

Given this volume of data a typical image can contain 8 million individual pixels. The challenge arises from the fact that targets of interest may reflect in only a small number of bands and consist of a very small number of pixels relative to the background environment. As a result any automated target recognition algorithm must be capable of isolating a small sample of target pixels from a much larger collection of differing signals and noise. The sheer volume of data contained in the image and the fact that while hundreds of bands may be sampled by the sensor, the underlying features producing a response in each band may be significantly fewer, demands dimensionality reduction. But the limited number of the number of target pixels available for detection versus the volume of data contained in the image makes the method of dimension

reduction and feature selection critical in retention of the desired target pixel data. These seemingly opposing goals will be the focus of this research.

1.2 Handling Hyperspectral Data

Management of this data is simplified by manipulating the "cube" of pixels into a single matrix of signal strengths. Spatially each pixel represents a small area within the geographic limits of the image, while each "layer" of the cube contains the reflectance, or signal strength in a specific portion of the spectrum for each pixel. Our interest is in the entire spectra for each pixel, so the cube of pixel returns is restructured so that each column of the rearranged matrix represents a single pixels entire spectral return (Figure 1-3). In this way a hyperspectral image with spatial dimensions i by j pixels and k spectral layers is rearranged into a j by n matrix where the n columns are the result of all i·j pixels.

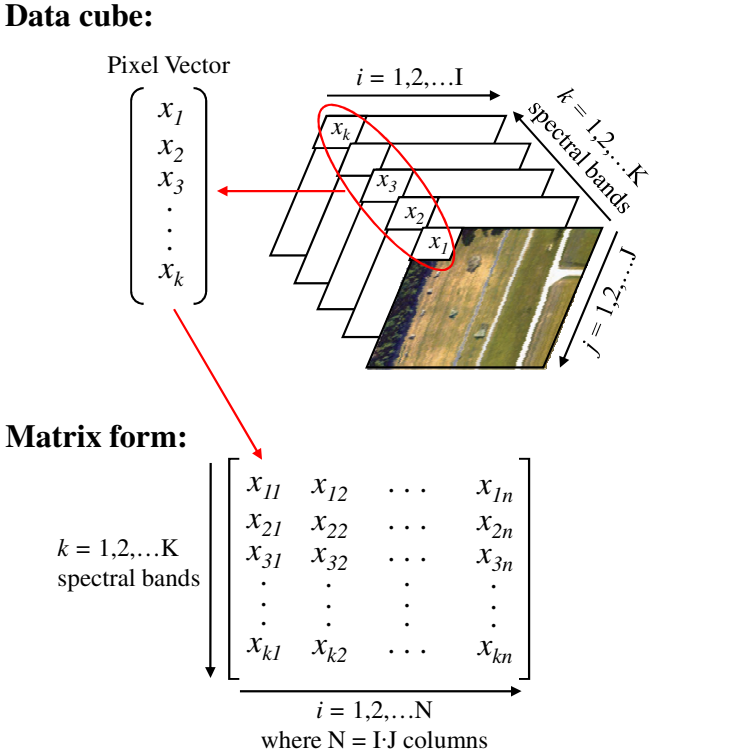


Figure 1-3. Restructuring a Hyperspectral Image Cube

1.3 Locating Targets of Interest

Hyperspectral target detection algorithms apply two general approaches to locating their targets, signature matching and anomaly detection. Signature matching techniques attempt to locate targets by comparing target pixels (columns of the matrix in Figure 1-3) against a library of known target reflectances for matches. As different materials produce dramatically different reflectances, each pixel's reflectance produces a characteristic "signature" to match against the materials of which targets are composed. Figure 1-4 shows how a vector of pixels results in a spectral signature across all bands of the hyperspectral image.

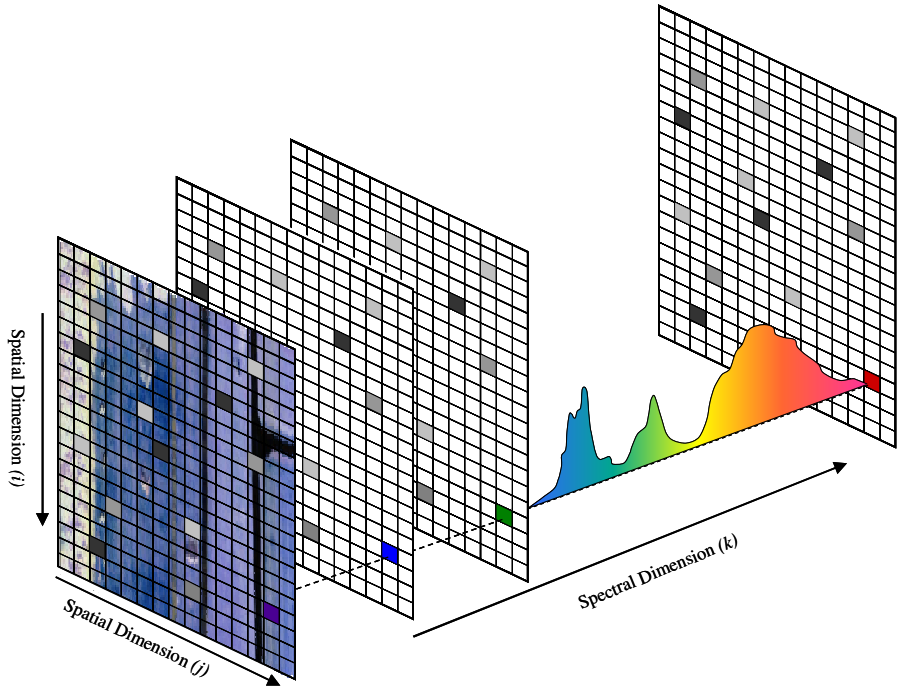


Figure 1-4. Single Pixel Spectra

At each band the pixel intensity results in a return, or reflectance value on the spectrograph. If this spectrograph can be matched to that of a material within a library of material spectra, then the primary material component within that pixel can be identified. Figure

1-5 shows how different the spectra of many naturally occurring materials can be. While signature matching is appealing based on its ability to not only identify target pixels, but also isolate the material composition of target and thereby provide target classification, matching pixels to any single signature poses a significant challenge. While materials may be distinguished from one another by their spectral characteristics, it is naïve to expect that pixels within an image will present a consistent, unique spectral shape when observed under remote sensing conditions. Sub-pixel material mixtures, atmospheric affects, and even the angle of illumination and view relative to the surface all result in variations in the spectral characteristics recorded by the sensor.

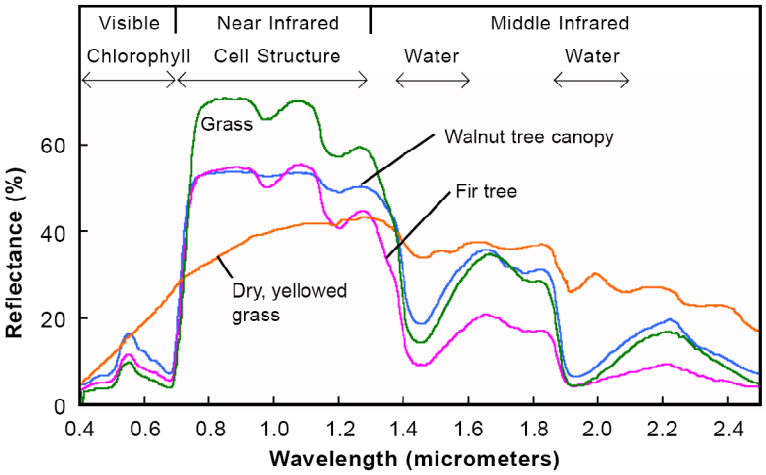


Figure 1-5. Spectral Signatures of Materials (Smith, 2006:7)

Anomaly Detection is the second technique applied to location of targets within a hyperspectral image. While multiple variations on this approach exist, the typical approach is comparison of the reflectance value of each individual pixel against the mean and variance of all other pixels (or within some defined surrounding neighborhood). If the reflectance of a pixel is found to exceed a target threshold, the pixel is identified as an outlier.

The popular category of techniques known as Global Linear Mixture Model Detectors assumes that each of the pixel vectors, or columns, is a convex combination of a finite number of endmembers. An endmember is a single spectra produced by an independent source or material reflectance. Although the number of endmembers present in the image and the coefficients required to separate their independent reflectance components is unknown, a powerful technique known as Independent Component Analysis (ICA) effectively isolates these underlying features, by imposing the requirement that the mixed components are a product of statistically independent signals (Stone: 8) This technique is fundamental to the algorithm applied in this thesis, and will be fully described as part of Chapter 2.

1.4 Research Objectives

As an extension of the thesis entitled "Improved Feature Extraction, Feature Selection, and Identification Techniques that Create a Fast Unsupervised Hyperspectral Target Detection Algorithm" by Captain Robert J. Johnson, the goal of this work is to further improve that methodology. Johnson pointed towards an investigation of methods that conform to non-negativity and sum-to-one constraints, which leads to replacing PCA and ICA with non-negative matrix factorization (NMF) as the mechanism of dimensionality reduction. This leads to the first objective of this thesis, application of NMF for dimension reduction. As the results will demonstrate, NMF is neither as effective nor as fast as ICA, despite the fact that ICA violates theory regarding the mixing of material reflectances to create their resultant pixel returns.

One goal of any automated target detection algorithm is that it be fast enough to process large volumes of data rapidly. This is especially true for target detection based on hyperspectral imagery. The addition of hyperspectral sensors to both manned and unmanned aircraft and

satellite platforms depends in part upon improving the rate at which imagery can be processed for further analysis. With that in mind a screening algorithm will be proposed which exploits correlation between spectral bands within an image to rapidly reduce dimensionality for processing. The goal is to improve algorithm run time while limiting performance loss.

In addition to reducing dimensionality by exploiting correlation, several other techniques will be investigated as candidates for enhancing algorithm performance and run time. Scott's rule estimating optimal histogram bin size is adopted in place of user estimated bin width. Johnson's Maximum Distance Secant Line heuristic is applied to estimate the location of a signal's noise baseline. A new measure of the contribution to kurtosis by one half of a distribution, Left Partial Kurtosis (LPK), is introduced and applied to determine when target pixels are likely to be found both in the left and right tails of signal histograms. Finally, an approach allowing for only two levels of iteratively filtering noise from target images is replaced with an approach that decreases the number of noise reducing iterations as the signal to noise ratio increases.

II. Literature Review and Critical Analysis of Current Practices

2.1. Linear Mixture Model (LMM) for HSI

In Chapter 1 the concept of a pixel's spectral signature was introduced, along with the recognition that recording any single pixel's spectra as that of some pure material within the context of remote sensing was naïve. While several effects lead to variations in the spectra recorded by sensors, sub-pixel mixing of materials as a linear combination of reflectances merits further discussion. Chang provides a straightforward description of the most commonly accepted mathematical explanation of how material reflectances mix to result in a single pixel vector, the Linear Mixing Model (LMM) (2007:108).

The model is based on some number of spectra of pure materials contained within the image scene. These pure material spectra are often referred to as endmembers. All pixel vectors are then a linear (additive) combination of the pure spectra. The number of endmembers (*M*) present within the image then limits the dimensionality of the data which can ascribed to a specific source to a maximum of *M*-1. Mathematically the LMM can be represented as:

$$\underline{x}_i = \sum_{m=1}^{M} a_{i,m}\boldsymbol{\varepsilon}_m + o_i + \mathrm{n}_i \tag{0.1}$$

Where:

$$\sum_{m=1}^{M} a_{i,m} = 1 \text{ for } i = 1,...,\mathrm{N}$$

$$a_{i,m} \geq 0 \text{ for } i = 1,...,\mathrm{N} \text{ and } m = 1,...,\mathrm{M}$$

$\underline{x}_i \equiv$ A pixel vector where each element is the intensity value of the reflectance in the k^{th} spectral band and i is the spatial dimension representing pixel location: $i = 1,...,\mathrm{N}$.

$\varepsilon_m \equiv$ Spectral signal of the m^{th} endmember in the image, where $m = 1,...,\mathrm{M}$ total endmembers. Each element of ε_m is the intensity value for the k^{th} spectral band of the endmember.

$a_{i,m} \equiv$ The abundance fraction of the m^{th} endmember in $\underline{x}_i$.\

$o_i \equiv$ The upwelling light from thermal radiation and atmospheric scattering

$\mathrm{n}_i \equiv$ Random Gaussian noise vector with zero mean

The LMM depends upon several assumptions which should be highlighted before proceeding. First, as indicated above the linear model is additive only, and as such only positive abundance fractions are permitted. In other words, a material can only contribute a positive reflectance, and cannot subtract from a pixel's resulting spectra. Second, each pixel vector (spectra) is a result of fractional abundances of the *M* pure contributors and thus the abundance fractions $a_{i,m}$ must sum to one. Third, the process of endmember identification depends upon the assumption that all pixel vectors are composed of combinations of the *M* pure material spectra, all of which are a priori unknown, but in the absence of noise are deterministic. Real hyperspectral data of course contains noise; so any endmember determination technique results in endmembers with error of the same magnitude as the noise. Finally, it is common to assume that the number of endmembers present in the scene is less than the number of bands recorded. As a result the dimensionality may be reduced to reflect the number of endmembers (materials)

present in the scene, while simultaneously retaining all information contained about the retained dimensions. This final assumption is essential to any method of hyperspectral data manipulation which reduces dimensionality to improve processing time.

2.2. Dimensionality Reduction

The dense nature of the information obtained by HSI allows detection of targets when traditional imagery would prove unsuitable, but the complexity of the information usually requires application of some scheme to reduce the dimensionality, while retaining as much content for anomaly detection as possible. The process of reducing dimensionality without loss of information is referred to as feature extraction. Johnson's AutoGAD (2008) employed one of the most powerful and popular forms of feature extraction, Principal Component Analysis (PCA).

2.2.1. Principal Components Analysis

Dillon defines PCA a multivariate technique which,

> …transforms the original set of variables into a smaller set of linear combinations that account for most of the variance of the original set. The purpose of principal components analysis is to determine factors (i.e. principal components) in order to explain as much of the total variation in the data as possible with as few of these factors as possible. (Dillon and Goldstein, 1984:24)

The definition continues by explaining that successive principal components are weighted linear combinations of the observed variables such that they are uncorrelated with one another and simultaneously account for the maximum amount of the remaining total variance.

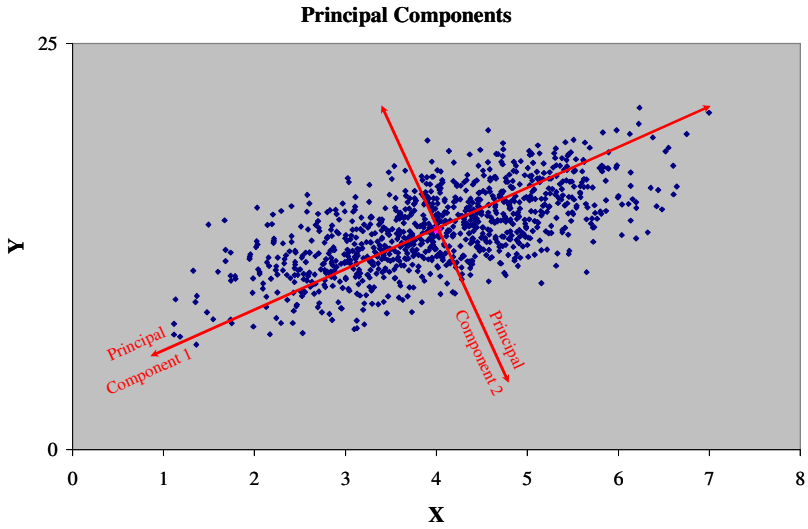


Figure 2-1. Two Dimensional PCA example

PCA is well suited to multivariate datasets where the majority of the variance present in the data is can be found along uncorrelated or orthogonal dimensions. The approach reduces the dimensionality by transforming the original data into a new orthogonal orientation such that the largest variance found within the data is aligned with the first principal component. Likewise, subsequent principal components align with orthogonal dimensions with the highest remaining variance. Figure 2-1 shows a simple example of this in two dimensions. In high dimensional data sets can commonly be oriented in such a way that a smaller set of dimensions explain a significant portion of the variance within the data, while remaining dimensions provide little further improvement, and can be removed without detracting from the information contained within.

Aside from the advantage of improving processing time, PCA provides some level noise elimination. Recall that the LMM assumes that all pixel vectors consist of a linear combination of pure spectra mixed with random noise. So by transforming the data into a space where subsequent dimensions contain the maximum remaining variance, and assuming that signal induced variation is of greater magnitude than noise, we expect that those dimensions containing

the most variance contain the highest signal to noise ratio. Thus retention of only those components consisting primarily of signal induced variation increases the overall signal to noise ratio of the data set (Johnson, 2008:2-12)

Classical PCA finds linear combinations $X_i = e_i^T Y$, for $i = 1,\ldots p$ which maximize variance. If X is a pixel vector across K bands then the first principal component is found by projecting X into a subspace (or principal component space) by

$$y_1 = e^T{}_1 X = e_{11}x_1 + e_{21}x_2 + \ldots + e_{K1}x_K \tag{0.2}$$

where:

$X \equiv$ pixel vector across K bands

$e_i \equiv$ projection matrix which rotate X into decorrelated dimensions

$e_{ij} \equiv$ rows of e_i describing rotation of x_i into component space y_{ij}

$y_i \equiv$ projections of X where the variance of y_1 is maximized, and each subsequent y_i maximizes the remaining variance

in which the variance of y_1, cov(y_1), is maximized subject to the projection vector e_1, being of unit norm. This can be written as the maximization problem:

$$\text{MAX}: \operatorname{cov}\left(e^T{}_1 X\right) = e^T{}_1 \Sigma_X e_1 \tag{0.3}$$

$$\text{s.t. } e^T{}_1 e_1 = 1$$

Dillon and Goldstein demonstrate the solution to the above maximization by sequential application and solution of the Lagrangian equation:

$$L\left(\underline{e}_1, \lambda_1\right) = \underline{e}_1^T \Sigma_{\underline{x}} \underline{e}_1 - \lambda_1\left(\underline{e}_1^T \underline{e}_1 - 1\right) \tag{0.4}$$

$$\nabla L = \begin{bmatrix} 2\Sigma_{\underline{x}}\underline{e}_1 - 2\lambda_1\underline{e}_1 \\ \underline{e}_1^T\underline{e}_1 - 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{0.5}$$

Despite its obvious utility, the use of PCA for hyperspectral data dimensionality reduction violates the basic assumption under the LMM that all pixel vectors are composed of non-negative combinations of endmember spectra. Hoyer (2004:1457) presented a methodology to overcome this limitation known as Non-negative Matrix Factorization.

2.2.2. Non-Negative Matrix Factorization (NMF)

Non-negative matrix factorization bears a striking resemblance to the Linear Mixture Model's first term and for that reason is an appealing technique. Just as the Linear Mixture Model represents an image as the sum of linear, non-negative constituent reflectances; NMF describes a data set as an additive only linear mixture of non-negative components. The technique begins with by denoting a set of N-dimensional measurement vectors $\mathbf{v}^t$ as a linear combination of basis vectors $\mathbf{w}_i$ according to a non-negative linear combination contained within the coefficient vector $\mathbf{h}^t$.

$$\mathbf{v}^t \approx \sum_{i=1}^{M} \mathbf{w}_i h_i^t = \mathbf{W}\mathbf{h}^t \tag{0.6}$$

When put into the same nomenclature as the Linear Mixing Model from equation 2.1, NMF can be written as,

$$\underline{x}_i = \sum_{m=1}^{M} a_{i,m}\varepsilon_m = \boldsymbol{E}\boldsymbol{a} \tag{0.7}$$

where $\boldsymbol{\alpha}_i$ is the abundance or coefficient vector for each endmember vector ε_i. In other words, each endmember vector (ε_i) represents the collection of non-negative reflectance values across

the sampled spectral bands, while each coefficient vector ($\boldsymbol{\alpha}_i$) describes the linear, non-negative contribution made by each end member spectra to each pixel.

Typically no combination of endmember vectors, ε_m, and abundance vectors, $\boldsymbol{\alpha}_i$, can perfectly reconstruct the original matrix of reflectances, $\underline{x}_i$, but the optimal pair of matrices $\boldsymbol{\varepsilon}$ and $\boldsymbol{\alpha}$ is defined as the one which minimizes the difference between $\underline{x}_i$ and $\boldsymbol{E\alpha}$. While a number of error functions have been proposed to provide a comparison between the original data set and the product of nonnegative components $\boldsymbol{E}$ and $\boldsymbol{\alpha}$, Hoyer (2004:1459) demonstrated an algorithm based on a Euclidian error function.

$$E(\mathbf{E},\alpha) = \left\| x - \mathbf{E}\alpha \right\|^2 = \sum_{i,m} \left(x_{i,m} - \left(\varepsilon_{i,m} \alpha_{i,m} \right) \right)^2 \tag{0.8}$$

Hoyer observed and went on to exploit the fact that NMF tends to produce sparse, or parts based, representations of the original data set. Ding, and others [2005] extended this observation with a demonstration that not only does NMF produce a sparse representation of data, but that the nonnegative factorization $\mathbf{V} = \mathbf{W}\mathbf{h}^t$ is equivalent to Kernel K-means clustering in the spectral dimension. This observation is important as in section three NMF will be replaced with a spectral clustering algorithm. The algorithm exploited in this work will not be a K-Means type of algorithm, but an altogether different mechanism for associating spectra into groups, based on intra-band correlation.

2.2.3. K-Means Clustering

K-means clustering is a widely applied clustering method, and was applied in the spatial dimension by Williams on the same ARES data used in this thesis. The technique attempts to partition the original data into K groupings, or clusters, where the difference between the mean value for each cluster and each point contained within the cluster is minimized. Just as a number of distance measures may be adopted to measure the difference between each cluster's mean and its associated original data, a Euclidian error function as in equation 2.8 is one of the most commonly applied measures.

Although K-means clustering will not be applied in this work, Ding provides a demonstration that NMF and K-means spectral clustering are "different prescriptions of the same problem" [2005:4]. In effect K-means clustering seeks pixels of similar spectral signatures. The average spectra of each of these "clusters" of similar pixels are used to "factor" the original data into a set of clustered components plus a remainder or error term. By design the average spectra must be positive. NMF approaches the problem somewhat differently in that it iteratively attempts to factor the original data using only nonnegative values within both the matrix of included spectra and the matrix describing how they are mixed. While the approaches are somewhat different Ding indicates they are closely related. Recognition of the similarity between these two methods is significant in that an intra-spectral band correlation clustering approach will be applied in Section 3.

2.3. Dimensionality Estimation

Determination of the number of spectrally distinct materials present within the image is of great concern when attempting to identify target pixels within the larger image frame for

several reasons. The Fast ICA algorithm applied following dimensionality reduction is an iterative technique placing significant computational demands on the processor utilized in the system, so appropriate dimensionality reduction will substantially improve processing time. However, if an insufficient number of dimensions are retained, two or more endmembers must be "mixed" with one another or mixed into several independent components by ICA. If any of these mixed endmembers happens to be that of a target signal, the likelihood of detecting the target is significantly reduced. The goal then is balancing the requirement to explain enough of the variance in the data with reducing the dimensionality sufficiently to reduce computational expense.

One common approach for selecting the number of endmembers to retain is simply retaining enough components to explain a predefined percentage variability found in the data. The pitfall associated with this approach rest in how dramatically the number of retained components varies with only small changes in percentage variability explained. This fact leads to a problem when attempting to select a single percentage variability to be retained for multiple images. As demonstrated by Johnson [2008:3-8] setting a single threshold of 99.78% variance retained on ARES 2F results in 33 retained components, or an estimation of 33 distinct spectral endmembers. But the same threshold leads to only 9 retained components when applied to ARES 1F. A number of approaches to estimate dimensionality from the eigenvalues of have been demonstrated including Kaiser's criterion, Cattell's test, and Horn's test [Bauer: 54].

Johnson proposes a simple and effective approach to identifying the number of spectrally distinct endmembers present in the image. The technique, known as the Maximum Distance Secant Line (MDSL), is based on the assumption that if an image is comprised of M distinct endmember signatures and noise. Stocker [2003:652] provides the objective model for the

spectral covariance matrix of the observed data, based on the same LMM discussed in section 2.1.

$$\mathbf{K}_x = E(\underline{\mathbf{x}} - E\underline{\mathbf{x}})(\underline{\mathbf{x}} - E\underline{\mathbf{x}})^T = E\left\{E(\underline{\mathbf{c}} - \overline{\underline{\mathbf{c}}})(\underline{\mathbf{c}} - \overline{\underline{\mathbf{c}}})^T\right\}\mathbf{E}^T + E\underline{\mathbf{n}}\underline{\mathbf{n}}^T = \mathbf{E}\Sigma\mathbf{E}^T + \mathbf{K}_n \quad (0.9)$$

Where E denotes expected value, $\mathbf{C}$ is the covariance of the abundance fraction (a from equation 2.1), $\underline{\mathbf{n}}$ represents random Gaussian noise as in equation 2.1, and K_n is the covariance from the sensor noise. That part of the covariance matrix generated by actual signal, $\mathbf{E}\Sigma\mathbf{E}^T$, is of rank M and will generate M non-zero eigenvalues. In the absence of noise all remaining K-M eigenvalues would be zero, while the inclusion of additive white noise ($\mathbf{K}_n = \sigma^2 I$) simply increases all remaining K-M eigenvalues by the noise variance σ^2. In such a case, σ^2 is equal to the constant valued portion of the eigenvalue distribution, and the correct number of endmembers M easily arrived at by counting the number of eigenvalues greater than σ^2. Figure 2-2 reproduced from Stocker [2003:653] illustrates the theoretically ideal example with $K = 64$ bands, $M = 8$ endmembers, and the white noise level, $\sigma^2 = 1$.

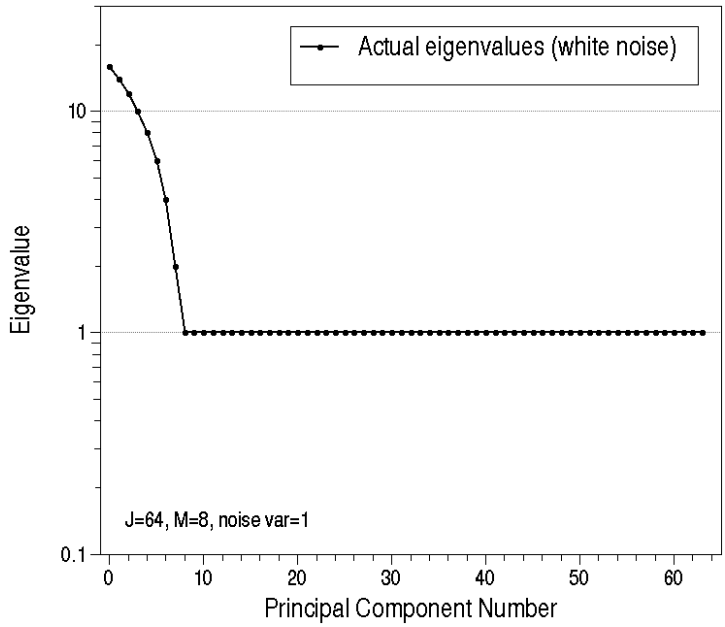


Figure 2-2. Ideal Model Signal Eigenvalues with White Noise Eigenvalues (*M* = 8) [Stocker, 2003:653]

When working with real images the a covariance matrix will have to be estimated from the recorded data, which given large sample sizes cross terms may be assumed negligible, yields

$$\hat{\mathbf{K}}_x = \mathbf{E}\hat{\Sigma}\mathbf{E}^T + \hat{\mathbf{K}}_v \tag{0.10}$$

Where $\hat{\Sigma}$ is the sample covariance of the abundance fractions, *a*, and $\hat{\mathbf{K}}_v$ is the noise sample covariance. While the signal portion of 2.10 will be of rank M, the eigenvalues of the white noise covariance will be non-constant based on sample estimation. As a result the flat section of eigenvalues at σ^2 from additive noise will become an inclined section of values in the vicinity of σ^2 on the eigenvalue distribution. The inclusion of non-constant sensor noise (non-white noise) further complicates the eigenvalue distribution by increasing the slope of the noise portion of the eigenvalue distribution. Figure 2-3 also from Stocker, shows these two situations.

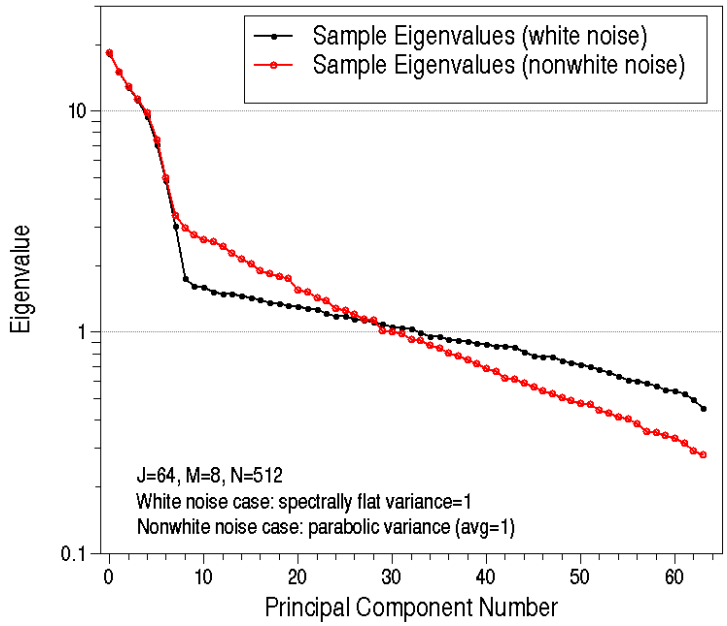


Figure 2-3. Sample Signal Eigenvalues with White & Non-White Noise Eigenvalues (M = 8) [Stocker, 2003:653]

Johnson exploits the fact that ordered eigenvalues of $\Sigma_{\underline{x}}$ may be used to identify the breakpoint between true signal and dimensions consisting primarily of noise. MDSL identifies the "knee in the curve" between signal eigenvalues and those of noise by locating the eigenvalue which is the greatest Euclidean distance from a "secant line" connecting the first and last eigenvalues when plotted on a logarithmic scale. Figure 2-4 shows graphically what this technique looks like on Stocker's data and how in both situations MDSL locates the first eigenvalue primarily comprised of noise, correctly producing a signal dimensionality of M = 8.

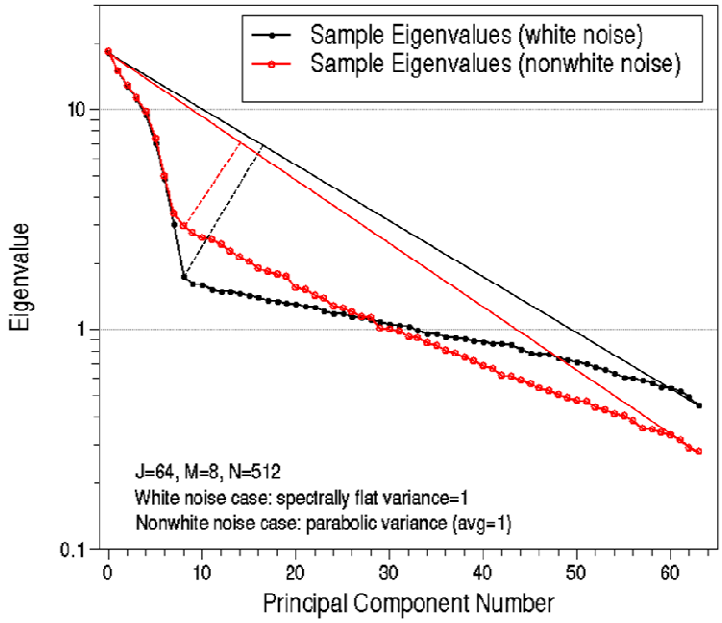


Figure 2-4. MDSL Technique for Locating Breakpoint between Signal and Noise on Stocker's Simulated Data (M = 8)

2.4. Independent Component Analysis (ICA)

Independent Component Analysis (ICA) can be seen as an extension to PCA, but rather than transform observed data into a rotation of decorrelated variables by application of second-order statistics, ICA includes an assumption of nongaussianity and independence of the latent variables. Though the application of higher order statistics these independent latent variables, or independent components, can be isolated. The isolation of independent components as opposed to merely uncorrelated components makes ICA a substantially more powerful technique than PCA and enables the approach to locate underlying factors where classical methods may fail [Hyvärinen, 2001:xvii].

ICA is a relatively new technique, having been introduced as recently as the early 1980s. A number of improved algorithms were introduced throughout the 1990s, culminating with the introduction of FastICA by A. Hyvärinen and E. Oja in 1997 [Hyvärinen, 1997:1483]. Their

algorithm demonstrated the ability to separate a data set into a collection of non-Gaussian independent components at significantly faster rates than previous ICA approaches. They went on to demonstrate that maximizing the non-Gaussianity of the information is equivalent to minimizing mutual information, or dependence between variables. Hyvärinen, Karhunen and Oja provide a complete development of the theory underlying ICA in their 2001 text *Independent Component Analysis*. The following sections provide a brief development of the theory behind ICA and its application.

The classical application of ICA is to the so called cocktail party problem, in which a number of "independent" voices (3 in equation 2.17 below) are speaking simultaneously, all of which is recorded by at least the same number of microphones placed about the room. Just as in hyperspectral reflectance data each recording is a mixture or weighted sum of all the sounds in the room at that moment. This mixture can be expressed as a simple linear equation:

$$\begin{aligned} x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) + a_{13}s_3(t) \\ x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t) + a_{23}s_3(t) \\ x_3(t) &= a_{31}s_1(t) + a_{32}s_2(t) + a_{33}s_3(t) \end{aligned} \tag{0.11}$$

Where x_i is the recorded mixture of signals at time, t, s_i is the independent source signal, and a_{ij} is the mixing coefficients (or abundance frequencies) which in this case are dependent on the distances between each microphone and the speakers being recorded. Just as in the hyperspectral signal separation problem, the mixed signal are the only information available, and neither the mixing parameters a_{ij}, nor the original signals $s_i(t)$ are known. For convenience equation 2.11 is typically written in vector-matrix notation.

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{0.12}$$

2.4.1. ICA Assumptions

Fortunately it is possible to separate the original signals, by making the assumption that all $s_i(t)$ are statistically independent at each time instant [Hyvärinen, 2001:150]. This is not only true for separation of mixed sound signals, but in the case of mixed spectral reflectances, ICA may be applied to unmix source spectral reflectances into their independent components. The assumption of independence is the first of three primary assumptions underlying the basic theory of ICA [Hyvärinen, 2001:152]. By independence we expect that the value of each s_i gives no information regarding the value of s_j for $s_i \neq s_j$. Or in terms of the joint probability density function (pdf) of our random variable s_i all random variables are considered independent if the joint pdf is factorizable.

$$p(s_1, s_2, s_3, ..., s_n) = p_1(s_1)p_2(s_2)p_3(s_3)...p_n(s_n) \tag{0.13}$$

The second of the basic assumptions is the nongaussian nature of the independent components. While the first assumption is foundational to the process of ICA, the second assumption is critical to estimation of the model. The requirement for this assumption stems from the dependence of ICA on higher order statistical information, primarily kurtosis, which is essentially a measure of how outlier prone a distribution is. Kurtosis is typically defined in one of two ways. ICA theory relies on the more common of the classical definitions for kurtosis.

$$\text{kurt}(x) = E\{x^4\} - 3\left(E\{x^2\}\right)^2 \tag{0.14}$$

however since whitening established unitary variance, $E\{x^2\} = 1$, so equation 2.14 reduces to:

$$\text{kurt}(x) = E\{x^4\} - 3 \tag{0.15}$$

By this definition Gaussian distributions produce kurtosis values of zero, while supergaussian distributions result in kurtosis values larger than zero.

The alternative definition for kurtosis does not apply the correction factor establishing that Gaussian distributions produce kurtosis values of zero.

$$\text{kurt}(x) = \frac{E\{x-\mu\}^4}{\sigma^4} \tag{0.16}$$

where:

$x \equiv$ sample distribution
$\mu \equiv$ population mean
$\sigma \equiv$ population standard deviation

By this definition a Gaussian distribution produces a kurtosis value of 3, while supergaussian distributions result in kurtosis values larger than three. The difference between the two definitions is subtle, but significant in that ICA employs the kurtosis as defined by equation 2.14 while other portions of this thesis apply the definition from equation 2.16.

The third assumption is that the mixing matrix states that the unknown mixing matrix is square and invertible. This assumption is made primarily for simplicity, but is based on the logical assertion that the number of independent components is equal to the number of observed mixtures. This greatly simplifies the solution strategy since once estimation for the mixing matrix (**A**) has been made, its inverse can be found (call it **B**) and the independent components can be obtained easily by solving equation 2.12 for s.

$$\mathbf{s = Bx} \tag{0.17}$$

2.4.2. ICA Ambiguities

Two main ambiguities, or indeterminacies, exist for the ICA model. The first of these is that it is not possible to determine the variances (or scale) of the independent components. This make logical sense as the model deals with two unknowns **s** and **A**, so any scalar multiplier α_i in any one of the sources s_i can be countered by dividing the corresponding column $\boldsymbol{a}_i$ of **A** by the same scalar.

$$\mathbf{x} = \sum_i \left(\frac{1}{\alpha_i} \mathbf{a}_i \right) \left(s_i \alpha_i \right) \tag{0.18}$$

The result of this is that the magnitudes of the independent components are generally fixed at unit variance, $E\left\{ s_i^2 \right\} = 1$. This leaves a sign ambiguity, effectively the model is unaffected by a positive or negative multiplier on any component.

The second ambiguity is that the specific order of independent components cannot be determined, which is again based on the fact that both **s** and **A** are unknown. In effect any permutation matrix and its inverse could be applied to **A** and **s** within the model to give

$$\mathbf{x} = \mathbf{A}\mathbf{P}^{-1}\mathbf{P}\mathbf{s} \tag{0.19}$$

The elements of **Ps** then are same independent components s_j, but in a different order. Likewise, $\mathbf{AP}^{-1}$ is simply the corresponding rearrangement of the original unknown mixing matrix, **P**.

2.4.3 An ICA Example

Hyvärinen [2001:155] provides an excellent graphical illustration of how ICA is able to separate a complicated collection of data into its most independent dimensions. He begins by considering two independent components taken from the following uniform distributions:

$$p(s_i) = \begin{cases} \frac{1}{2\sqrt{3}}, & \text{if } |s_i| \le \sqrt{3} \\ 0, & \text{otherwise} \end{cases} \tag{0.20}$$

The range of this distribution was chosen specifically so that it has zero mean and a variance equal to one. The joint density of 2000 samples of s_1 and s_2 then takes on the shape of a square as shown in figure 2-5.

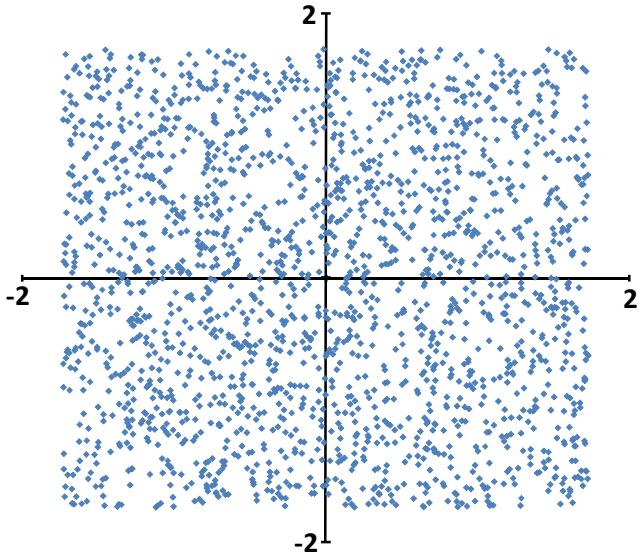


Figure 2-5 Joint Distribution of Independent Components

When the two components are mixed by an arbitrary mixing matrix, the shape of the joint density is changed. Using the following matrix:

$$\mathbf{A} = \begin{bmatrix} 5 & 10 \\ 10 & 2 \end{bmatrix} \tag{0.21}$$

The mixed variables x_1 and x_2 are obtained and when the joint distribution of the observed variables is graphed we see that the data has taken on a uniform distribution in the shape of a parallelogram (figure 2-6). More importantly the variables x_1 and x_2 are no longer independent of one another (i.e. knowledge of the value of one variable provides information on the value of the other). From figure 2-6 it can be seen that if one of the variables takes on its maximum value, the value of the other variable must also be its maximum value. In figure 2-5 however knowledge of the value of s_1 provides no useful information about the value of s_2.

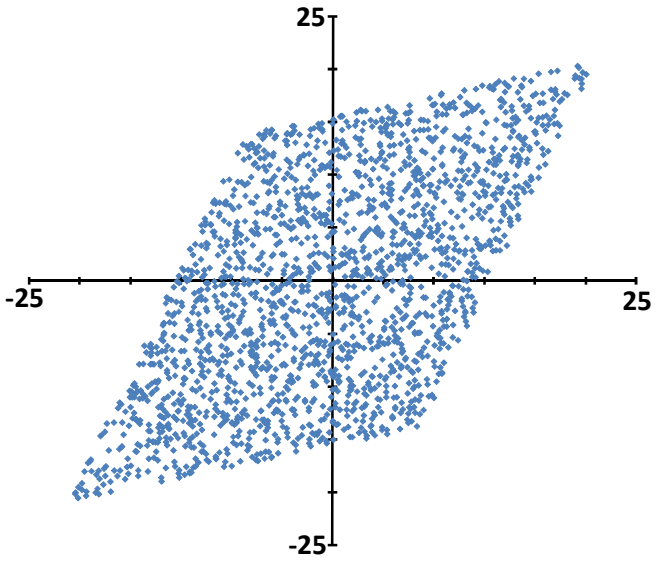


Figure 2-6 Joint Distribution of Mixtures x_1 and x_2

Given that this mixed data is the only information available, the problem becomes estimating the mixing matrix **A** from only this mixed data. Observation of figure 2-6 indicates that it would be possible to estimate **A** by recognizing that the edges of the parallelogram are in the directions of the columns of **A**. However in cases where the underlying distributions are

more complicated than a pair of uniform variables, location of the "edges" of the joint distribution becomes somewhat more difficult. For example figures 2-7 and 2-8 show the joint distribution of supergaussian independent components and the mixed joint distribution by the previously applied mixing matrix **A**. Locating the "edges", or at least the directions of the edges, in figure 2-8 while possible, is not nearly as simple as in the previous case. In practice, edge location is considered a poor technique because it works with variables derived from very specific distribution. Generally edges cannot be found; and edge location algorithms and other similar methods, tend to be computationally complex and unreliable [Hyvärinen 2001:156].

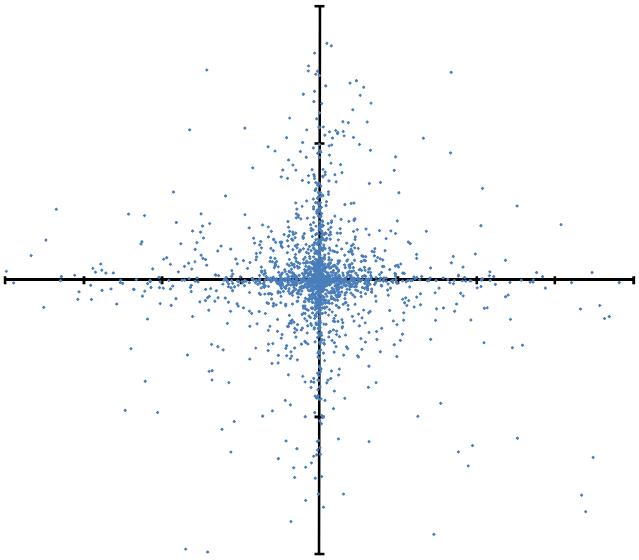


Figure 2-7 Joint Distribution of Supergaussian Independent Components s_1 and s_2

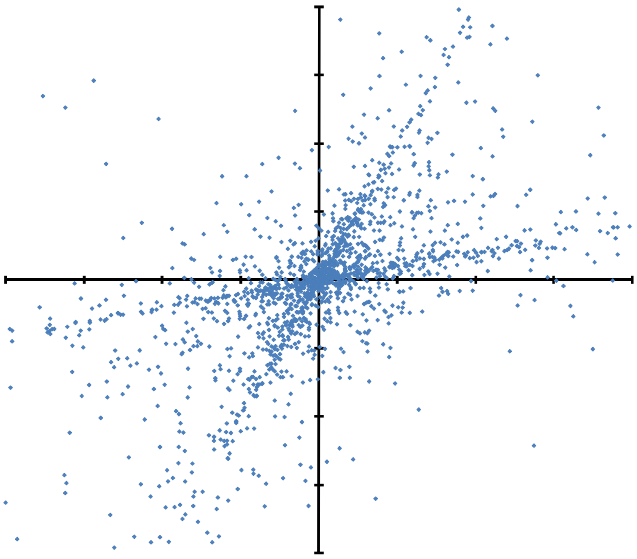


Figure 2-8 Joint Distribution of Mixtures x_1 and x_2 from Supergaussian Independent Components

Rather than attempt to identify the mixing matrix graphically, ICA provides a mechanism which works well for any distribution of independent components, and is both fast and reliable. The first step in this process is whitening of the data.

2.4.4 Uncorrelatedness verses Whitening and ICA

Recall from section 2.2.1 that principal components analysis orients data into a set of dimensions in which the variance along orthogonal or uncorrelated axes is maximized. This procedure is a simple linear transformation of the variables, which makes it a tempting technique for analyzing hyperspectral data. However simply identifying uncorrelated components, does not imply that independent components have been found. In other words, independence of variables is "stronger" than zero correlation between components. Recall that if two variables x_1 and x_2 are uncorrelated their covariance is zero:

$$\operatorname{cov}(x_1, x_2) = E\{x_1 x_2\} - E\{x_1\} E\{x_2\} = 0 \tag{0.22}$$

For two variables x_1 and x_2 to be considered independent, the joint density function, $f(x_1, x_2)$, must be a product of their marginal densities [Wackerly, 2002, 235].

$$f(x_1, x_2) = f_1(x_1) f_2(x_2) \tag{0.23}$$

Likewise any pair of functions, h_1 and h_2, containing the variables, x_1 and x_2, must also be factorizable into separate terms

$$E\{h_1(x_1) h_2(x_2)\} = E\{h_1(x_1)\} E\{h_2(x_2)\} = 0 \tag{0.24}$$

However in the case of zero covariance this is not necessarily true. Hyvärinen provides a discrete example of four equally likely values: (0,1), (0,-1), (1,0), and (-1,0) which can be demonstrated to be uncorrelated by equation 2.19. Using equation 2.21 where h_1 and h_2 simply square the variables x_1 and x_2 the following is obtained,

$$E\{x_1^2 x_2^2\} = 0 \neq 1/4 = (1/2)(1/2) = E\{x_1^2\} E\{x_2^2\} \tag{0.25}$$

which violates equation 2.21, indicating that although x_1 and x_2 are uncorrelated they are not independent.

Whitening adds to uncorrelated variables the requirement that their variances be equal to unity. Or in other words both the covariance matrix and the correlation matrix of a vector (**x**) of whitened random variables are equal to the identity matrix. Whitening requires a simple linear transformation of each observed data vector **x** by multiplying it with some matrix **V**

$$\mathbf{z} = \mathbf{V}\mathbf{x} \tag{0.26}$$

The FastICA algorithm adopted for this work calculates the whitening matrix V by one of the most common mechanisms for whitening eigenvalue decomposition of the covariance matrix

$$\mathbf{V} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^{T} \tag{0.27}$$

where **E** is the orthogonal matrix of eigenvectors of $E\{\mathrm{x}\mathrm{x}^{T}\}$ and **D** is the diagonal matrix of its eigenvalues, $\mathbf{D} = \mathrm{diag}(d_1,\ldots d_n)$.

2.5. Automated Global Anomaly Detection (AutoGAD) Basics

Captain Robert Johnson provided an effective anomaly detection algorithm technique in his 2007 thesis proposal "Improved Feature Extraction, Feature Selection, and Identification Techniques that Create a Fast Unsupervised Hyperspectral Target Detection Algorithm". The Automated Global Anomaly Detector, or AutoGAD, performs multiple operations on a hyperspectral data cube, to isolate those pixels which are identified as especially different from the background pixels in the image. The process begins by restructuring the data cube into a matrix where each row contains a single pixels entire spectral signature across all 210 bands. removing those bands which are known to absorb signal from the dataset. For the ARES images studied both in Johnson's thesis and this study, this step reduced the dimensionality from a raw set of 210 spectral bands to 145 bands. A graphical depiction of the steps contained in the preprocessing portion of the algorithm is displayed in figure 2-9.

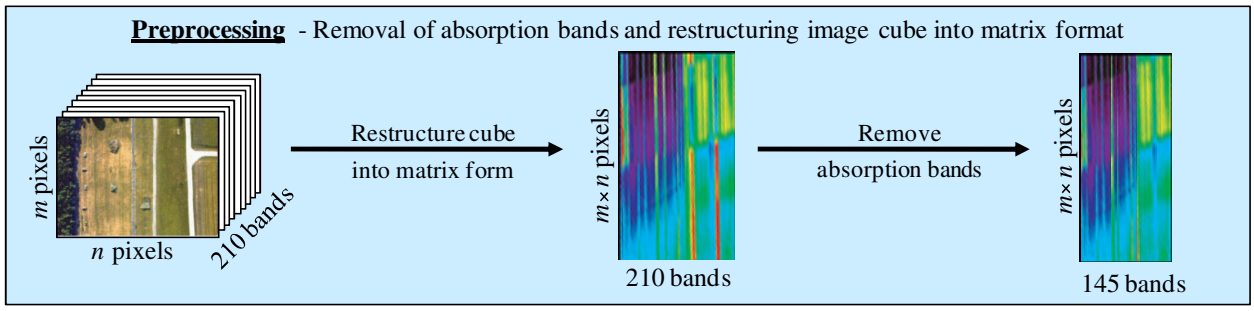


Figure 2-9 AutoGAD Preprocessing of Hyperspectral Data Cube

AutoGAD then extracts the primary features contained in the data by performing Principal Components Analysis and producing an ordered log scale eigenvalue curve from the results. Dimensionality is assessed using the MDSL technique described in section 2.2.3. This novel approach at finding the dimensionality rapidly reduces the spectral dimensions from 145 to approximately the 10 to 15 dimension containing the majority of the variation. These uncorrelated dimensions are then passed into the FastICA [Hyvärinen, 2001], algorithm which "whitens" the remaining components and then rotates the remaining dimensions into their most independent orientation. It is from these independent signals that the outlier pixels are found. Figure 2-10 provides a depiction of the process of feature extraction in AutoGAD, while Figure 2-11depicts the second portion of AutoGAD feature extraction, ICA.

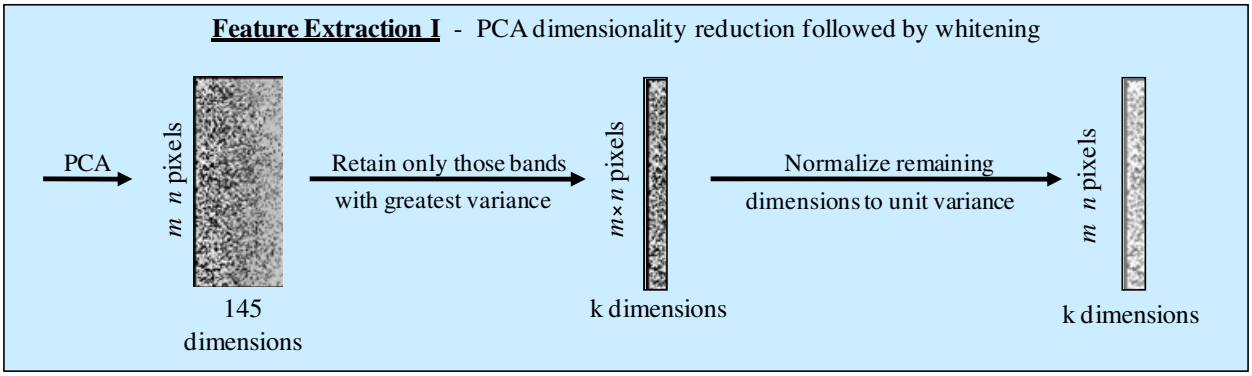


Figure 2-10 AutoGAD Feature Extraction I (PCA, Dimensionality Reduction, and Whitening)

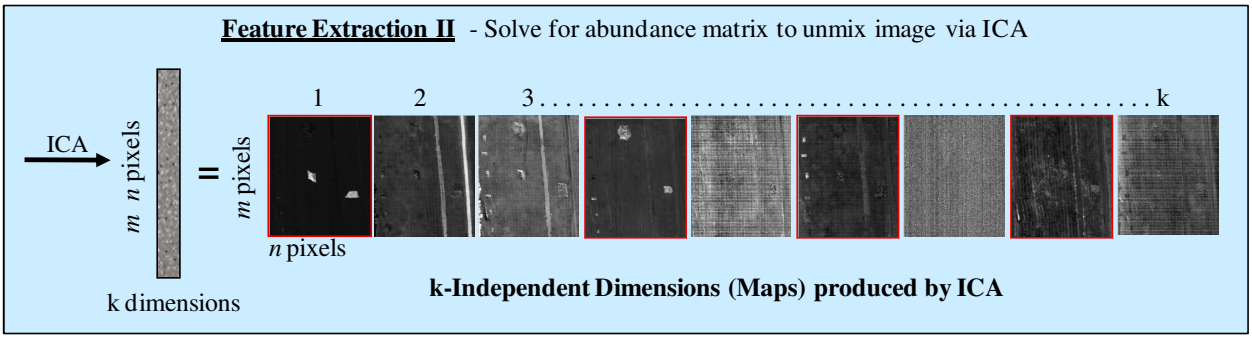


Figure 2-11 AutoGAD Feature Extraction II (ICA with sample of resulting independent components)

At this point a set of independent dimensions has been produced, some of which potentially contain identifiable targets, while others contain primarily noise. The process of discriminating between these two classes of components, or maps, is known as feature selection. AutoGAD measures a maximum signal score and calculates a sample signal to noise ratio (SNR) for each remaining component. By convention all signals are ordered so that the largest absolute value IC score is positive valued. Maximum signal score is simply the pixel value furthest into the tail in the positive direction. Measurement of SNR requires identification of a threshold between potential target pixels and all other "background" pixels.

A histogram is then produced from the remaining independent signals by binning each pixel across the range of signal values. The whitening accomplished during ICA automatically scales each dimension such that the mean is equal to zero and the variance is one, so the vast majority of the pixel values fall into bins near zero. As required given the assumptions underlying ICA, each of the signal distributions is extremely non-gaussian, containing a handful of bins representing the vast majority of pixels, and a large number of bins containing only a small number of pixels each in the tails of the distribution. It is from these "tails" that target pixels are isolated. AutoGAD distinguishes between potential target pixels and background noise by locating the first bin with zero pixels occurring, and setting that value as the threshold between noise and signal.

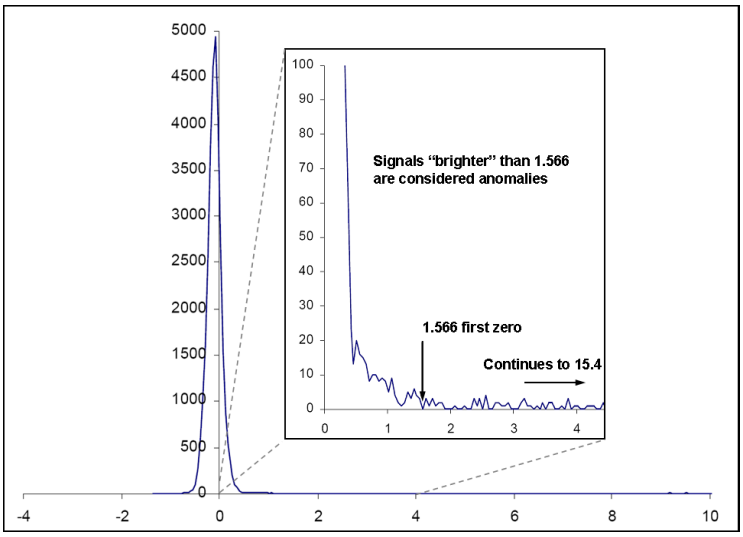


Figure 2-12. First Zero Bin Histogram Identification of Noise Threshold [Johnson:153]

Once the threshold between signal and background pixels has been identified, Potential Target Signal to Noise Ratio (PT SNR) of each map is found by the equation below.

$$PT\ SNR_{dB} = 10 \bullet \log_{10}\left(\frac{\text{power(potential target signal)}}{\text{power(background)}}\right)$$
$$= 10 \bullet \log_{10}\left(\frac{\text{var(potential target signal)}}{\text{var(background)}}\right) \qquad (0.28)$$
$$10 \bullet \log_{10}\left(\frac{\sigma_i^2}{\sigma_b^2}\right)$$

Maps exceeding both the maximum pixel value and the signal to noise ratio expected for target maps, are retained for further processing while all other maps are discarded. Figure 2-13 depicts AutoGAD's feature selection process for two of the independent components from ARES 1F.

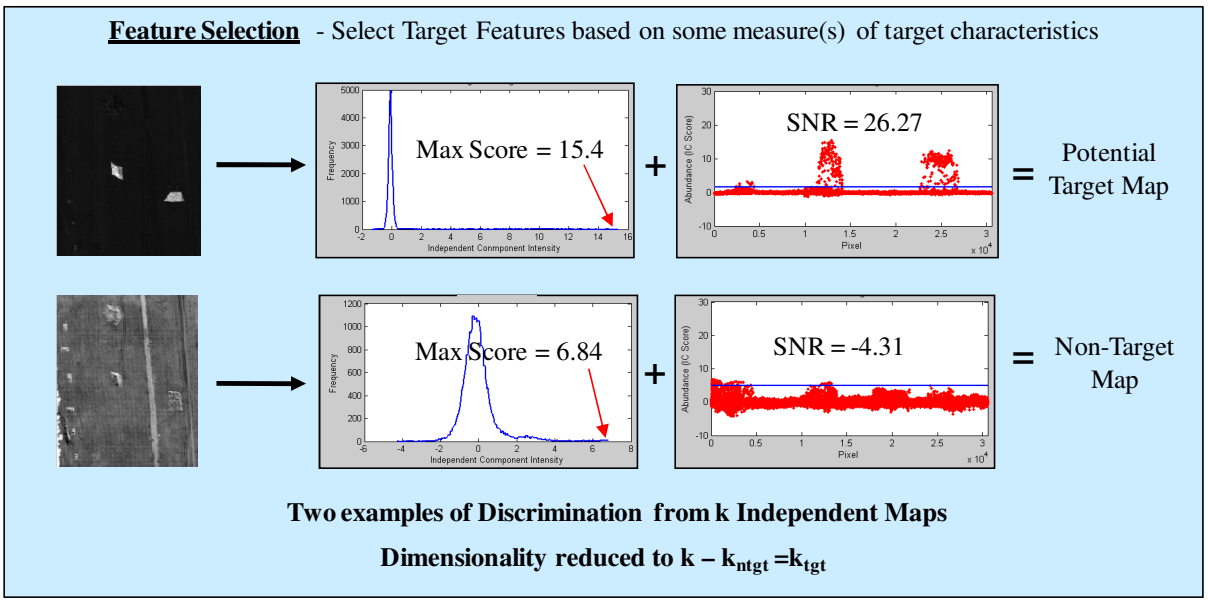


Figure 2-13. AutoGAD Feature Feature Selection (Max Signal Score and PT SNR)

The next step in processing the remaining signals is smoothing, or Iterative Adaptive Noise (IAN) filtering. IAN filtering reduces the noise present in the signal by comparing each pixel value and the variance in the immediate neighborhood of that pixel to the overall system variance. The filter more heavily smoothes those pixels with variance near the overall system noise, and applies less smoothing to those pixels in neighborhoods where the variance is greater than that of the system as a whole [Johnson, 2007:3-49]. A second signal histogram is then generated with the smoothed signals and the baseline noise threshold is again estimated based on the value of the first histogram bin containing zero pixels. All pixels with values greater than this threshold are deemed target pixels, while those pixels less than the threshold are deemed non-target pixels. It is possible to estimate a threshold both above and below the background noise band, in which case pixels with values less than the negative threshold are also labeled

targets. Finally, a single map representing the collection of target pixels is constructed from the remaining independent maps.

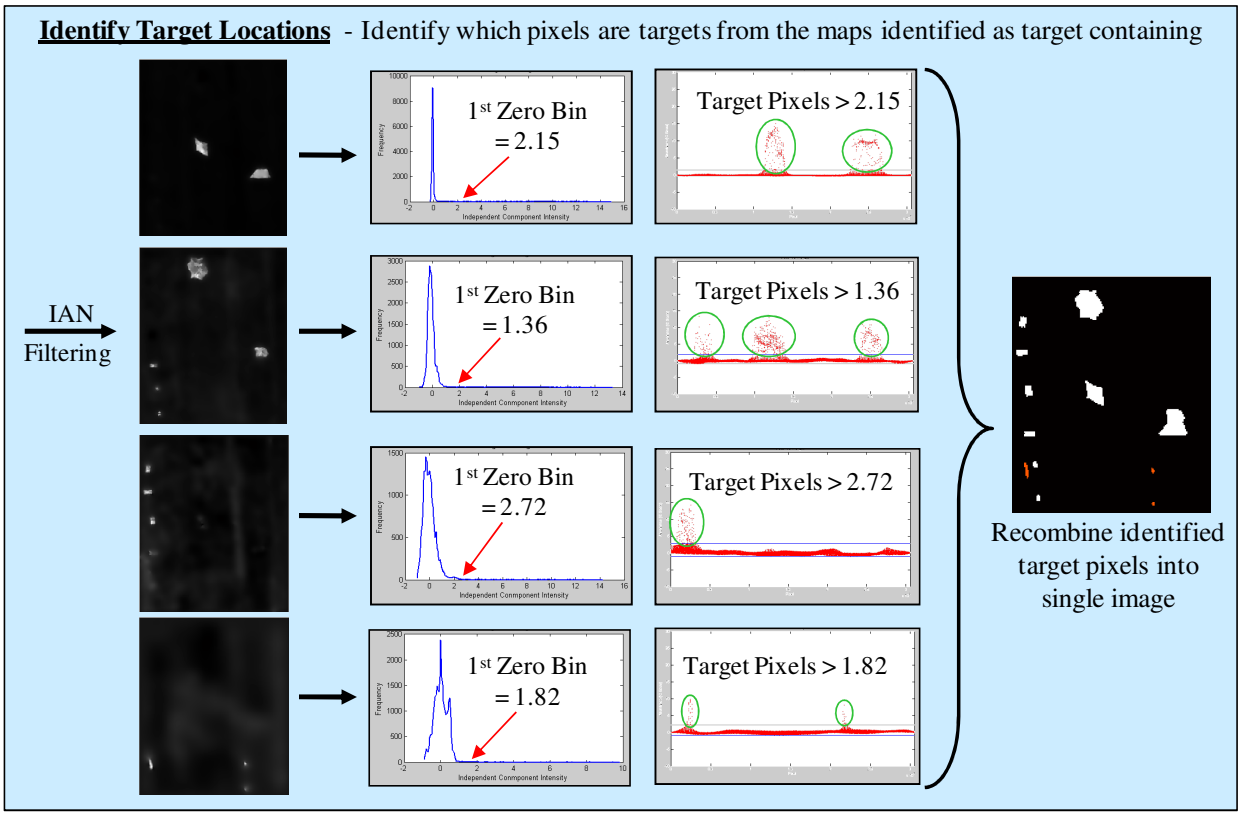


Figure 2-14. AutoGAD Target Pixel Identification and Image Reconstruction

The entire AutoGAD process typically requires less than 10 seconds to locate and report target pixels to the user. Johnson reported a 0.84 True Positive Fraction (TPF), and a 0.0025 False Positive Fraction (FPF), with an average processing time of 6.64 seconds for the eight images. While these results are extraordinary, the customer desires an even faster method for isolating images containing targets. Figure 2-15 shows the result of ICA for one of the eight images. Although PCA has reduced the dimensionality from its 145 original dimensions, the remaining 15 dimensions still contain a certain amount of noise, as seen in the four independent signals on the left side of the figure. While this does not present a problem with regard to target

detection, it does in situations where speed is considered an important factor. Each additional dimension processed via ICA complicates the iterative process of producing independent signals and thereby adds time to the entire process. Further, each retained dimension requires creation of at least one signal histogram, and calculation of signal to noise ratio, prior to elimination as a non-target map. As a result, retention of more bands than required to identify independent components becomes vitally important to minimizing overall algorithm run time.

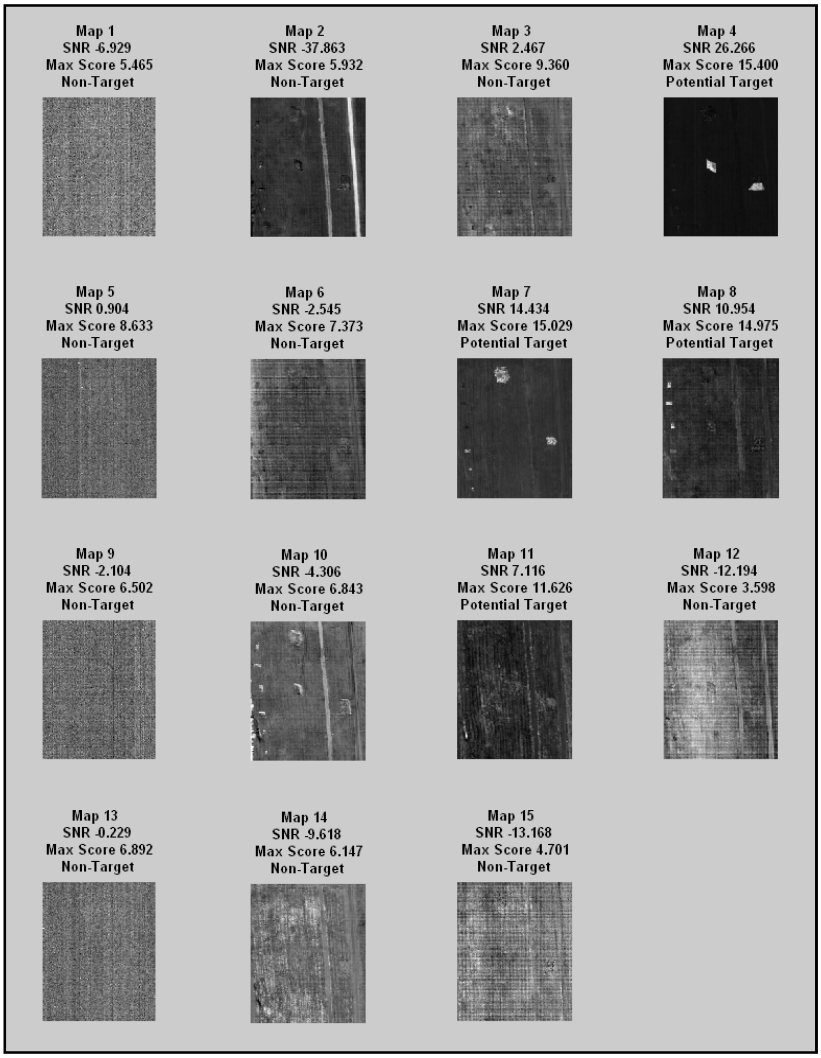


Figure 2-15. 15 Independent Signal Components produced by AutoGAD

2.5.1. Histogram Bin Width and Scott's Rule

Rather than employing a user defined bin width for formation of the signal histogram, Scott's Rule (equation 2.18) was applied to determine the histogram bin width for each independent signal resulting from ICA[Scott:1979:608].

$$h_n = 3.49sn^{-1/3} \tag{0.29}$$

Where:

$h_n \equiv$ Histogram Bin Width
$s \equiv$ standard deviation
$n \equiv$ number of observations

Although Scott's rule makes the assumption that the underlying distribution is Gaussian, this assumption has no significant impact on the shape of the resulting histogram. Figure 2-16 below displays the independent components of ARES 1F and the associated histograms for Johnson's original AutoGAD using a histogram bin width of 0.05 (Johnson's recommended setting), while Figure 2-17 shows the same set of independent components with histograms created according to Scott's rule. The main difference in the histograms produced by the two techniques is simply that application of Scott's rule tends to produce slightly larger bin sizes and therefore generate somewhat more smooth distributions, while the fixed bin width of 0.05 applied in AutoGAD produces more ragged distributions. This would be significant if the use of first zero bin were to be employed as the discriminator between background noise and signal, but this particular technique will be replaced with a graphical measurement which locates the "knee in the curve" where background transitions to signal. This technique will be described fully in section three.

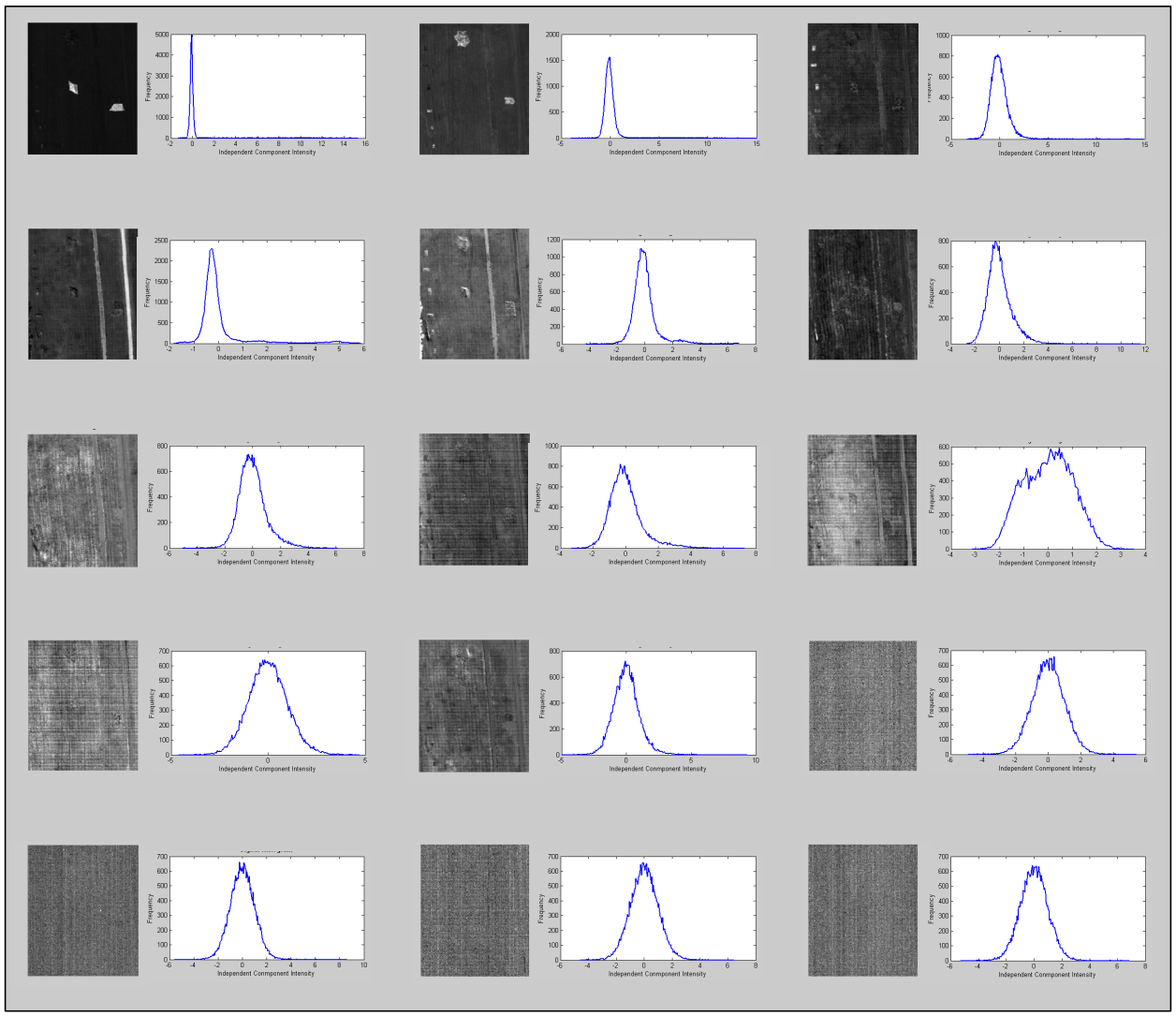


Figure 2-16. Independent Components with associated histograms (bin width = 0.05)

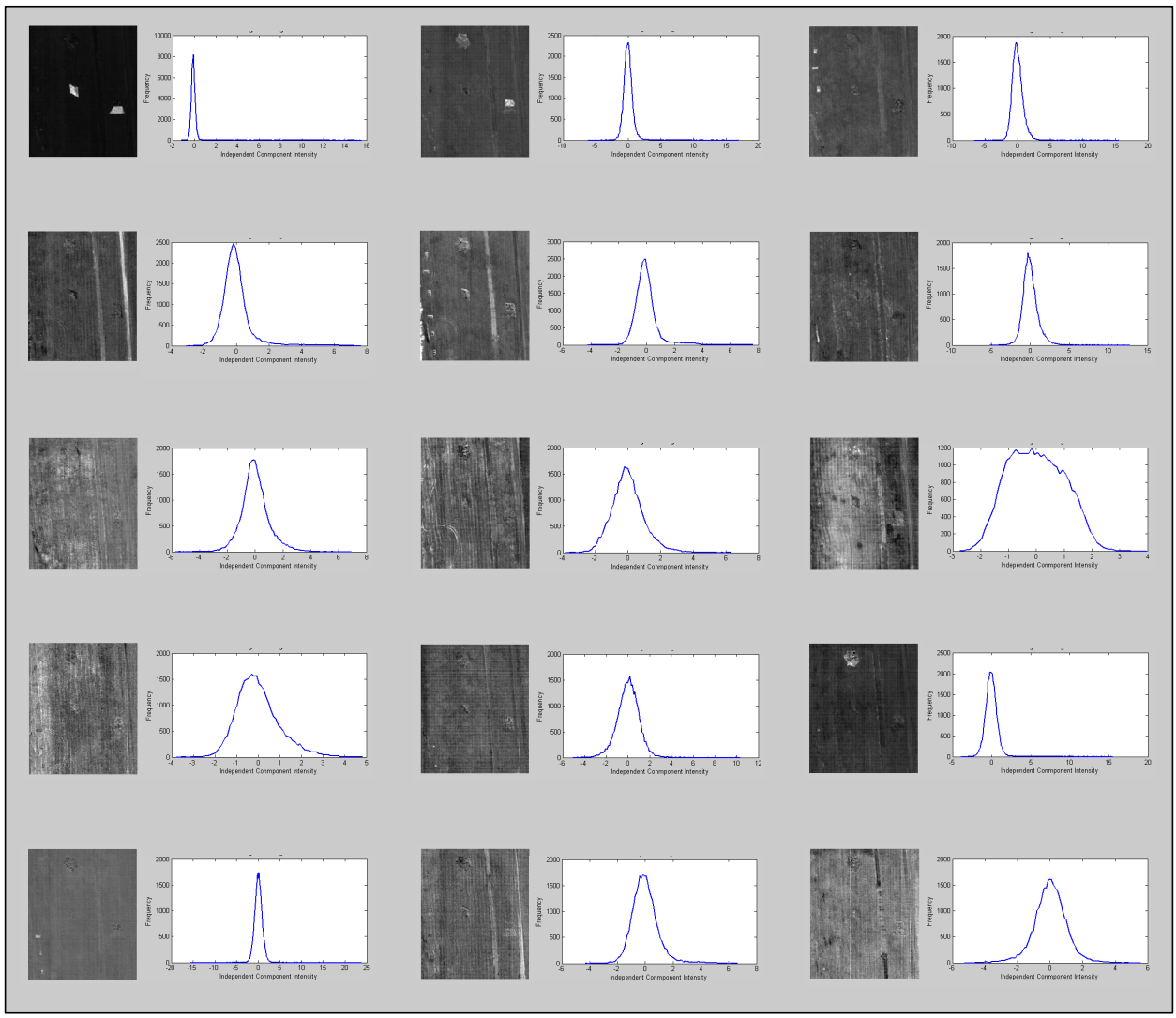


Figure 2-17. Independent Components with associated histograms (bin width determined by Scott's Rule)

III. Methodology and Test Image Experimentation

3.1. Detector Process Flow Comparison

Two separate adaptations of Capt Johnson's AutoGAD were developed as a part of this work. The first procedure is primarily focused on replacing PCA and ICA with a methodology which agrees with the Linear Mixture Model on how light reflected from sub-pixel sized objects of different materials mixes to produce a single pixel reflectance spectra. In this particular algorithm Non-negative matrix factorization replaces both PCA and ICA. The second procedure employs a clustering algorithm in place of PCA to perform simultaneous dimensionality assessment and reduction. Before delving further into this second algorithm, a brief discussion follows on the NMF algorithm and its unsuitability to situations where algorithm speed is considered a priority. Figure 3-1 shows a comparison of the AutoGAD algorithm to the AutoGAD – NMF algorithm. The only significant change (shown in bold) made to the algorithm is substitution of NMF in place of PCA and ICA for dimensionality assessment and estimation of the unmixing matrix. This single change to the algorithm provided such poor results that no further attempts were made to improve upon the algorithm and further use of NMF was abandoned.

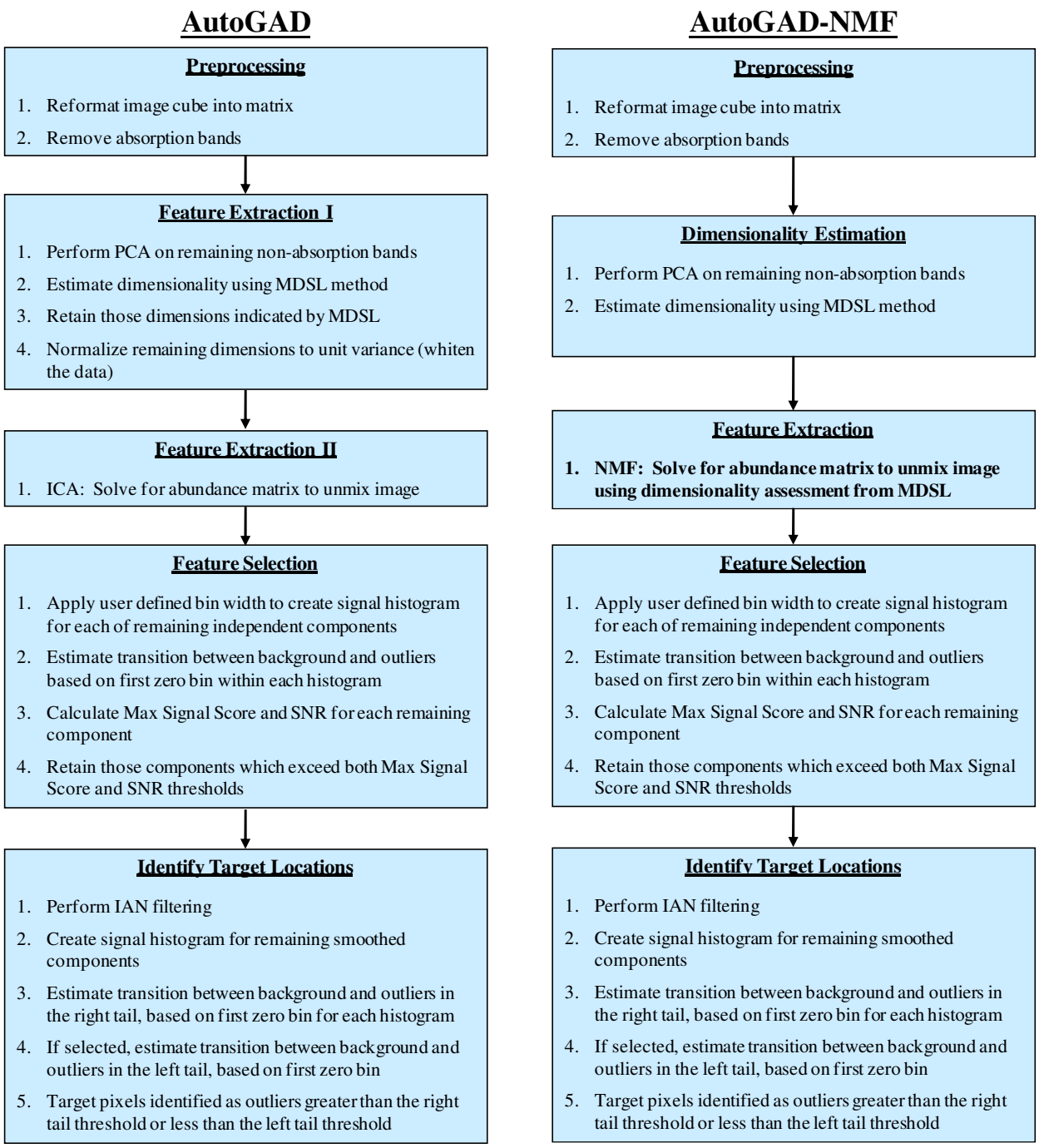


Figure 3-1. Process Comparison AutoGAD vs. AutoGAD by NMF

3.2. NMF Algorithm

As with AutoGAD the first issue is reduction of dimensionality. ARES data contains 210 spectral bands. A number of these are along atmospheric absorption bands and contain little or no useful information [Smetek: 2007]. Elimination of these absorption bands immediately reduces the data from 210 to 145 usable spectral bands. In the version of the algorithm which employs NMF, an estimate of the number of spectrally distinct endmembers is required so that the NMF algorithm may reduce the data from a single matrix of 145 dimensions into two matrices, one containing only the retained "primary" endmember spectra and one describing how the endmember spectra are mixed to create the original image. As in AutoGAD, estimation of the number of spectra to retain for further analysis is based on a graphical technique which locates the "knee in the curve" separating signal from noise on an eigenvalue curve.

Following endmember estimation, a non-negative matrix factorization [Ross:1999] is applied to the original dataset as a means of partitioning the original data into a set of basis vectors (underlying primary spectral components), a matrix of coefficients describing how the basis vectors are mixed with one another, and an error term. In its original version, unmixing of the endmember matrix and abundance matrix ended at this point in order to maintain non-negativity of all components of both matrices. As will be demonstrated in section 3.2 this approach was unable to separate target pixels sufficiently from background detail to isolate targets. So a second version of the algorithm followed NMF dimension reduction with ICA to further separate components into the most independent versions of the non-negative factors provided by NMF.

In the clustering algorithm, comparisons are made between adjacent spectral bands of the image for high intra-band correlation. If two or more adjacent bands exceed a threshold

correlation, they are averaged as a single cluster of spectra. All bands insufficiently correlated with at least one other spectral band are discarded as too noisy. The number of clusters created from this "correlation induced" clustering algorithm is taken to be the number of endmembers present in the image. In this way the algorithm simultaneously reduces dimensionality as it estimates the number of endmember spectra present in the image.

3.2.1. Nonnegative Matrix Factorization

Initial attempts into NMF demonstrated that while the technique was capable of separating hyperspectral data into a reduced set of latent components, the time required and error involved in the process made the approach unsuitable in situations where large sets of images are to be processed. Figures 3-2 and 3-3 show the image reconstruction error and the 15 separated images following 10 iterations of NMF, a process which required 30.634 seconds to complete.

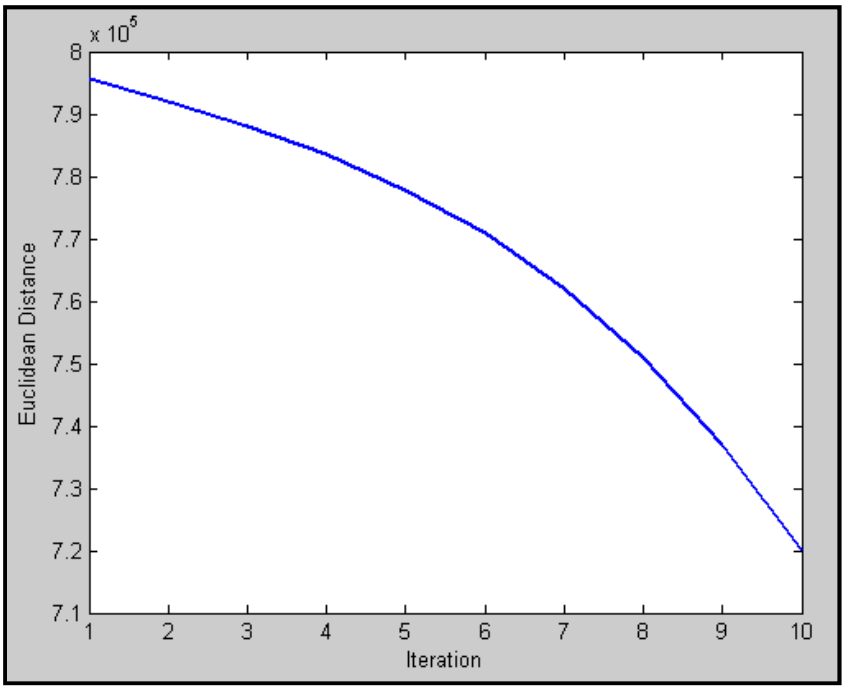


Figure 3-2. Reconstruction Error for ARES 1F in 10 Iterations

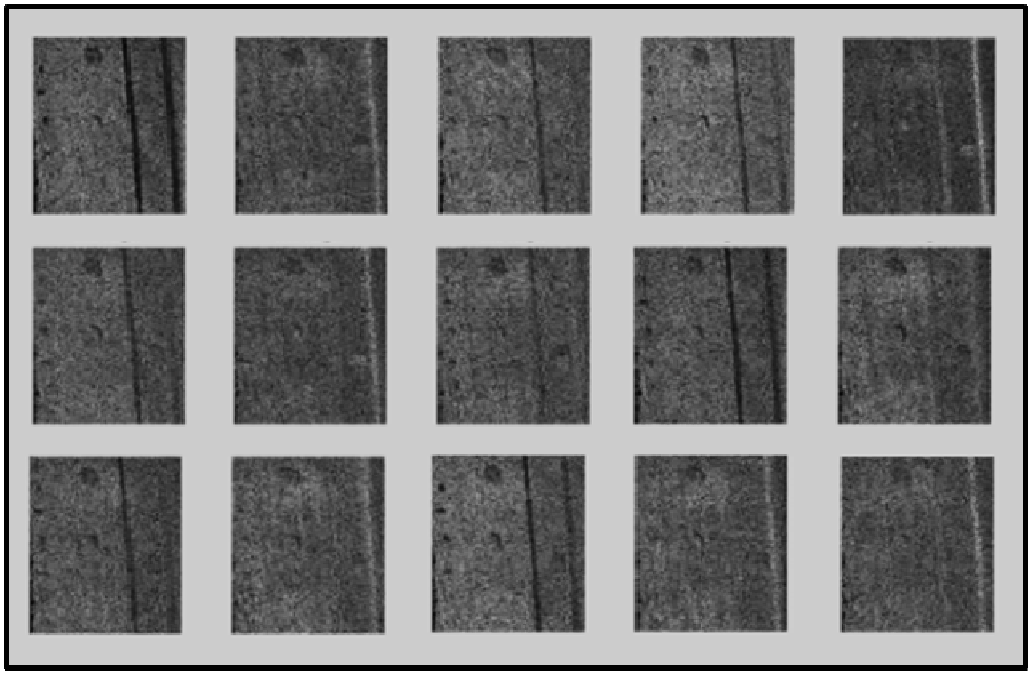


Figure 3-3. Latent Components following 10 iterations of NMF

Given that figure 3-2 indicates the NMF algorithm had yet to converge to a minimized reconstruction error, a second attempt at NMF target identification was made, this time with 50 iterations of the NMF algorithm. When the number of iterations is increased to 50, the NMF algorithm shows signs of converging at a reconstruction error on in the range of 1.75×10^5 (Figure 3-4), and the resulting separated images are somewhat more well separated (Figure 3-5), yet actual targets could not be autonomously identified. Furthermore, the algorithm required 135.1 seconds for completion, clearly too long for any real time application. Further increasing the number of iterations to 100, as depicted in figures 3.6 and 3.7, provided only marginal improvements in signal separation, yet required 256.7 seconds (~4 minutes 17 seconds).

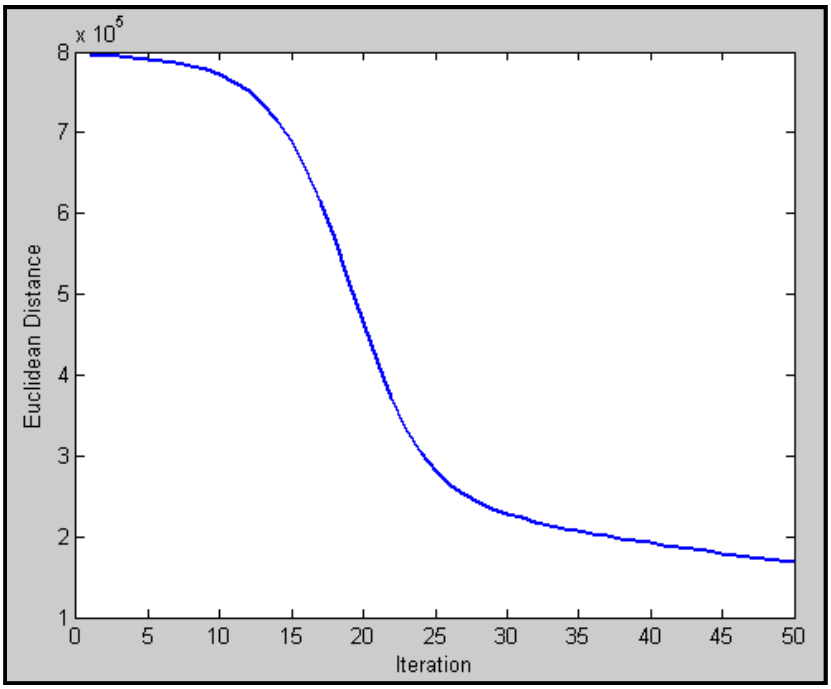


Figure 3-4. Reconstruction Error for ARES 1F in 50 Iterations

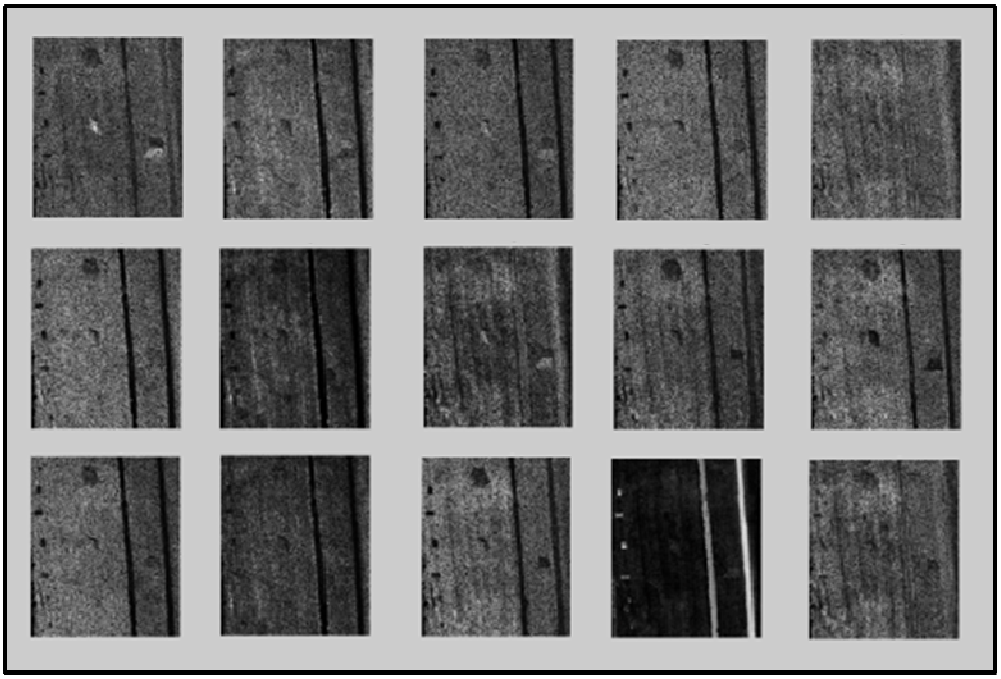


Figure 3-5. Latent Components following 50 iterations of NMF

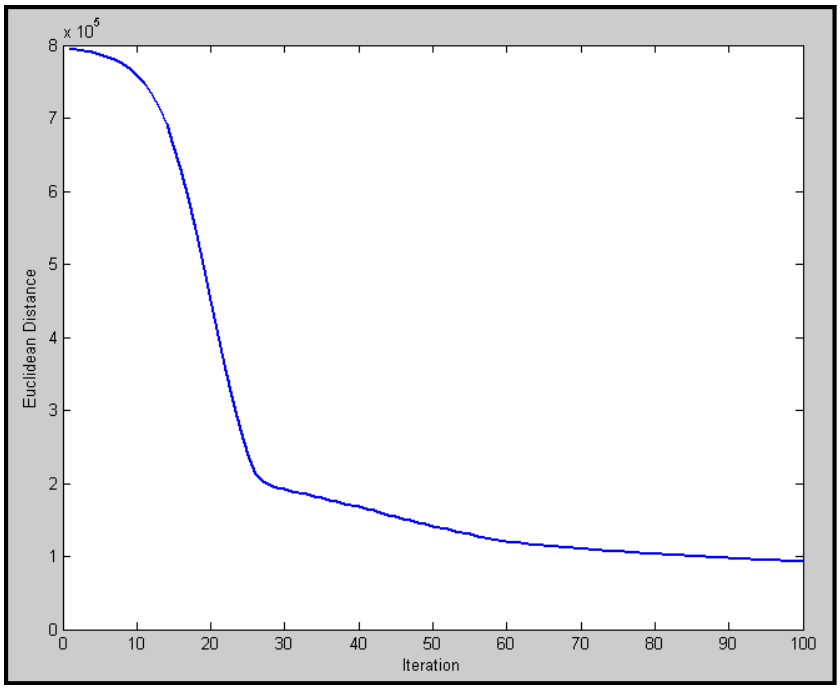


Figure 3-6. Reconstruction Error for ARES 1F in 100 Iterations

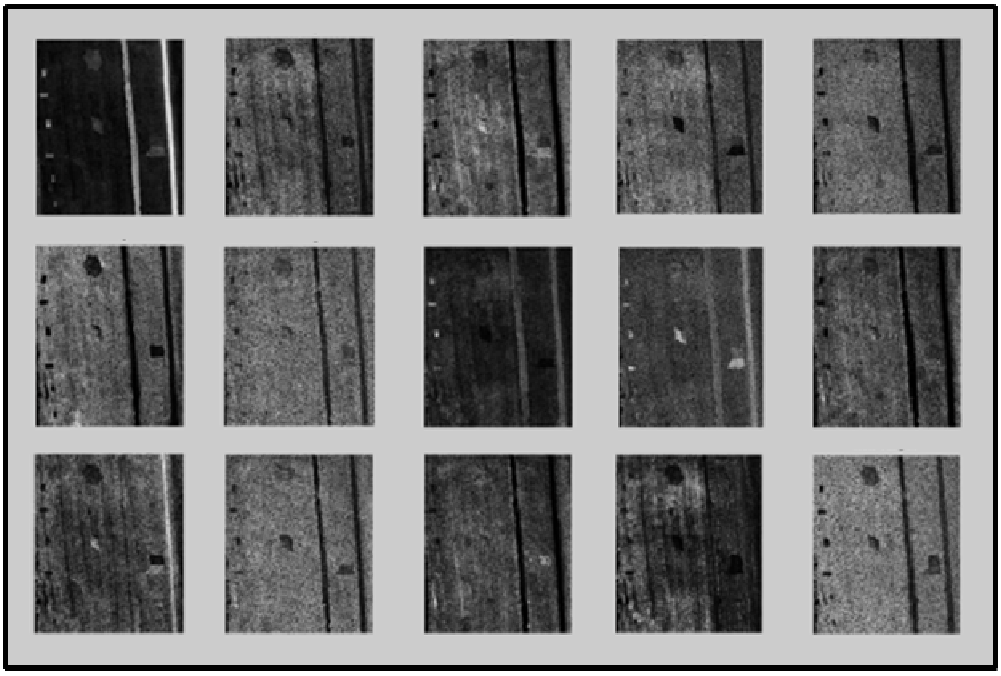


Figure 3-7. Latent Components following 100 iterations of NMF

A second attempt at identifying targets was made by including ICA following NMF as a dimensionality reduction step. Of course, this dictates violation of the assumption of strict

nonnegativity of components and or abundance fractions at the ICA portion of the algorithm, but given that NMF alone as unable to sufficiently isolate signal from noise the conditition of nonnegativity was abandoned. This approach was capable of isolating six of the nine targets present in the image (figure 3-7), but even after accomplishing 150 iterations of NMC, followed by ICA (a process requiring excess of 370 seconds) only about 52% of the true target pixels could be identified. Furthermore the combined NMF/ICA procedure could not consistently isolate targets, likely due to the stochastic nature of both NMF and ICA.

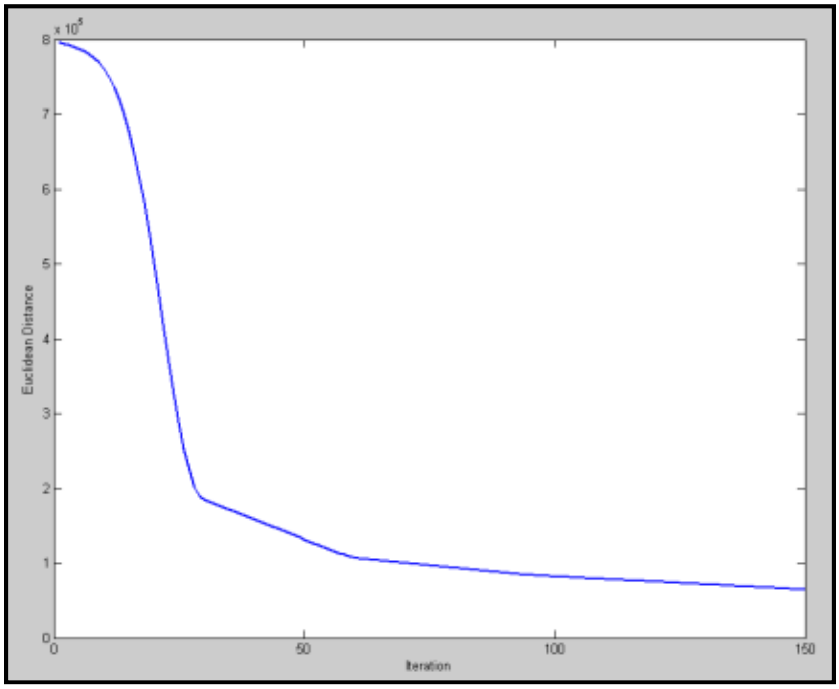


Figure 3-8. Reconstruction Error for ARES 1F in 150 Iterations

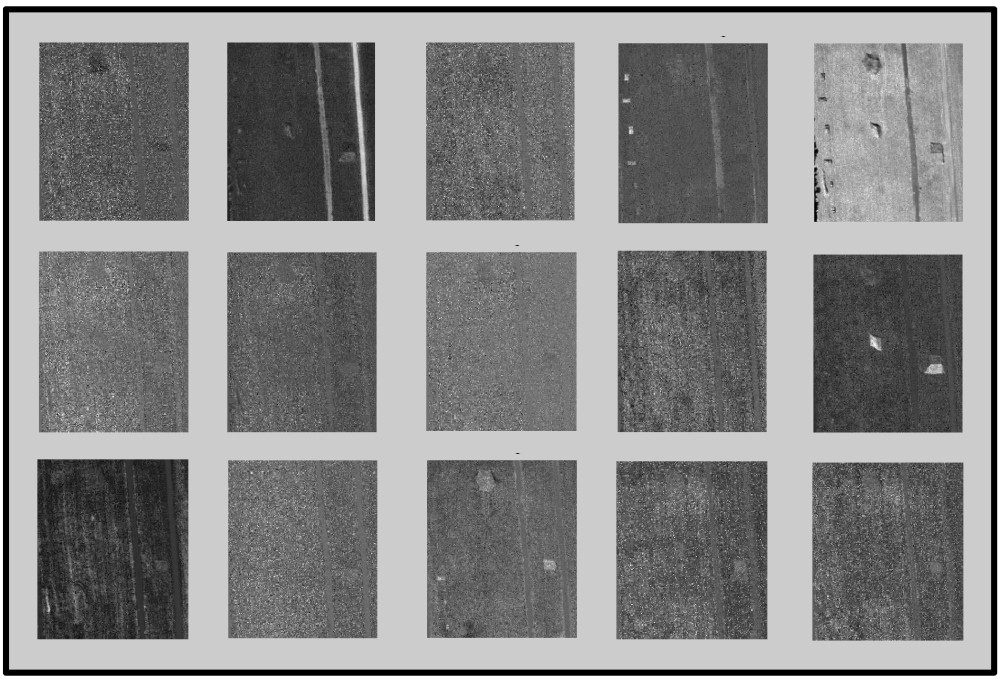


Figure 3-9. Independent Components after 150 iterations of NMF followed by ICA

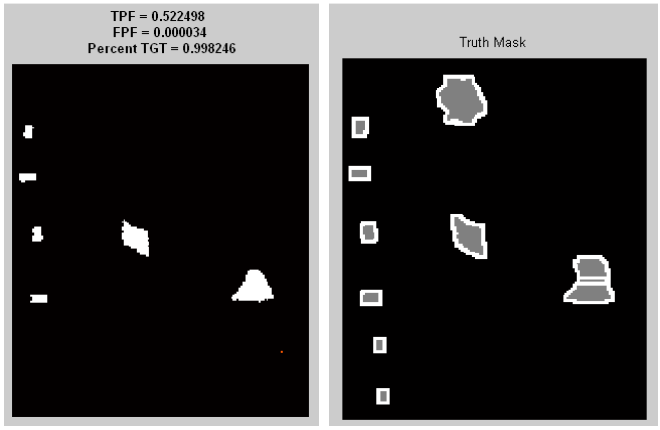


Figure 3-10. Target Pixels found versus Truth Mask after 150 Iterations of NMF followed by ICA

Interestingly nine of the fifteen independent components depicted in figure 3-7 appear to be comprised primarily of noise, leaving what appear to be six significant latent components. In a final attempt to improve the performance of the NMF portion of the algorithm a set of tests was completed in which the number of endmembers was fixed between 6 and 14. Figure 3-9 shows

the results from the tests including 6 through 11 latent components. As observed in figure 3-7 there are consistently six components which appear to contain separated signals, while all additional components appear to be comprised of primarily noise. These additional "noise" components are highlighted by red boxes in image 3-8. The tests with 6 through 9 components did not isolate any target pixels, while the 10 component test isolated four of the nine targets and a mere 14.5% of all target pixels and the 11 component test isolated six of the nine targets and about 51.2% of all target pixels. Tests of 12 through 14 components were conducted, but are not depicted since the general trend of each component beyond the sixth being comprised primarily of noise continued. When the image was factored into 14 components, 67.6% of the target pixels were located, but as stated previously these results were inconsistent based on the stochastic nature of both NMF and ICA. In each test run time was recorded at values ranging between 375.8 and 386.5 seconds.

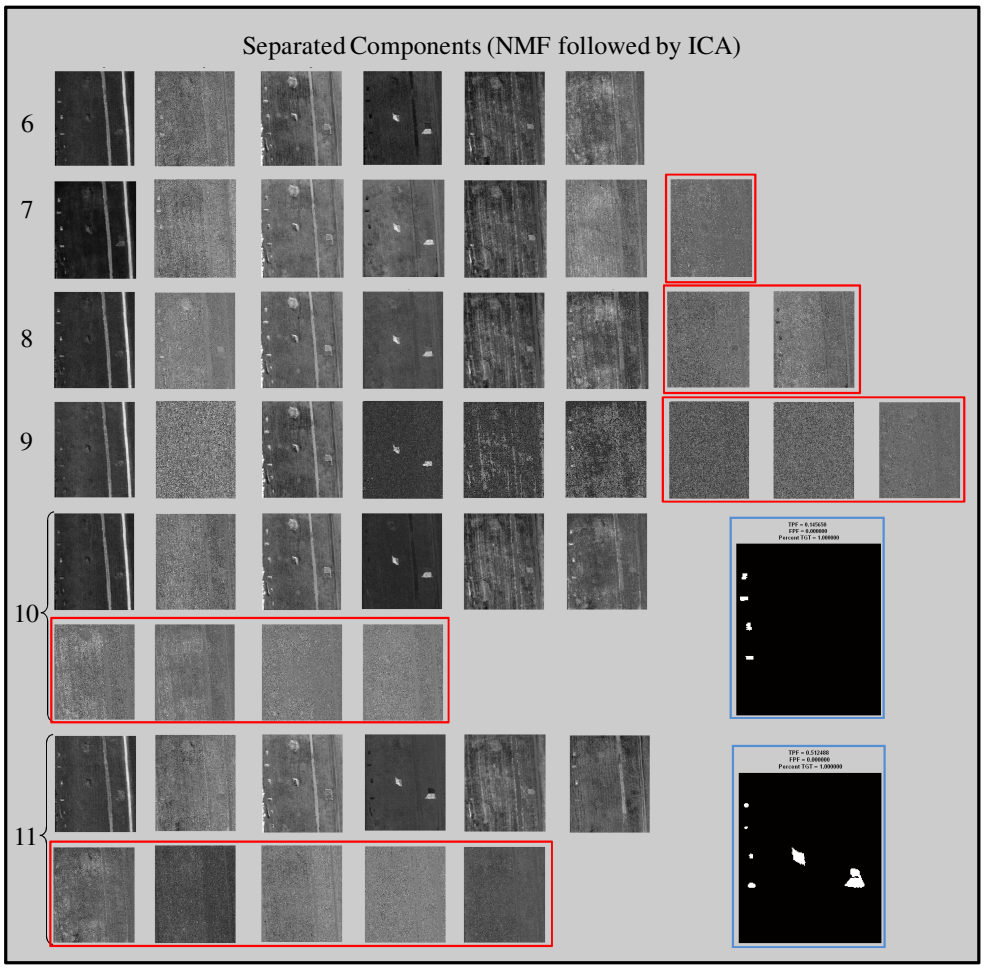


Figure 3-11. ARES 1F NMF-ICA algorithm test results (Components fixed at 6 through 11)

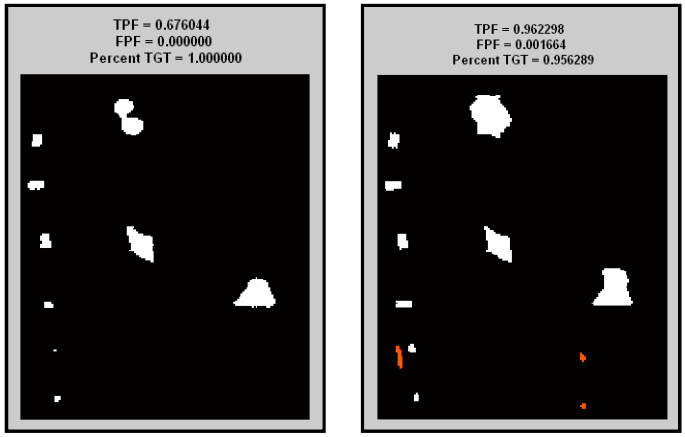


Figure 3-12. ARES 1F NMF-ICA algorithm test results (14 Components) vs. AutoGAD

It should be noted that the NMF-ICA algorithm required in excess of 420 seconds to produce a True Positive Fraction (TPF) of target pixels of only 0.676 as depicted in figure 3.10, while the original version of AutoGAD required only 6.6 seconds to produce a TPF of 0.962. These relatively poor results from the NMF-ICA algorithm indicate that although NMF closely matches the theory underlying the linear mixture model that it does not produce superior results to PCA followed by ICA, nor is it suitable in situations where algorithm run time is critical.

3.3. A Spectral Clustering Approach

As an alternative to PCA with the secant line method for estimating the dimensionality of a hyperspectral data set, a clustering algorithm could be applied to simultaneously estimate the appropriate number of dimensions and aggregate raw hyperspectral data into a more manageable set of spectral data for ICA. Williams [2007] addresses the idea of clustering pixels together by applying several clustering algorithms including the K-means, X-means, and ISODATA algorithms for clustering pixels in the spatial dimensions.

This type approach presents a twofold problem. First because the K-means algorithm searches the spatial dimension for groups of pixels with similar spectra, an assumption of no *a priori* knowledge of how pixels ought to be clustered is required. In other words, we cannot expect pixels in close physical proximity to be associated into a single cluster, nor can we expect physically separated pixels to be disassociated. Unfortunately this requires the K-means algorithm to perform multiple permutations of the possible clusters in search for one grouping which minimizes the measure of separation between clustered pixels. The price of such a methodology is the computational expense of searching through the possible arrangements of clustered pixels in an attempt to find one which reduces difference between clustered pixels. The

second challenge is that this approach to clustering requires some initial estimate of the number of clusters. Although these algorithms can increase or decrease the number of clusters, the convergence to a solution can be very dependent on the initial number and location of the clusters. Furthermore, the correct number of clusters can be heavily dataset dependent, so in any truly automated anomaly detection algorithm exploiting clustering, the requirement to provide an initial estimate of the clusters present is undesirable.

Williams also addressed exploiting intra-band correlation to rapidly reduce dimensionality, but exploited the results of dimensionality reduction to then speed the process of spatial clustering, thereby exposing the algorithm to the same limitations discussed in the previous paragraph. Additionally Williams applied a methodology requiring a covariance stationary dataset; despite the fact he recognized that this was not necessarily the case. This meant he was forced to experiment to determine the appropriate number of bands to discard between retained spectral signals. His results demonstrated that if every fifth band was retained for follow-on spatial clustering, acceptable results could be achieved. Unfortunately, by assuming a covariance stationary system, Williams was forced to retain too many bands (every fifth) in sections of the data cube where intra-band correlation was highest, and too few in sections where correlation was lowest.

3.3.1. Simultaneous Dimensionality Estimation and Reduction by Spectral Clustering

Rather than seeking spectrally similar pixels in the spatial dimension to accelerate target detection, this research proposes that intra-spectral band clustering be applied in an attempt to define the spectral dimensionality of the data, and to provide a reduced dimension dataset to ICA as used in Johnson's AutoGAD algorithm. The first advantage to this approach is the fact that a

certain amount of a priori knowledge can be assumed and heavily exploited. In the spatial dimension the correlation between any two pixels cannot be assumed based on physical proximity, but in the spectral dimension this is not necessarily the case. The correlation colorgraph below shows the typical relationship between spectral bands within hyperspectral data. Bright green regions indicate high positive correlation between bands, bright red indicates strong negative correlation, and yellow indicates near zero correlation between bands.

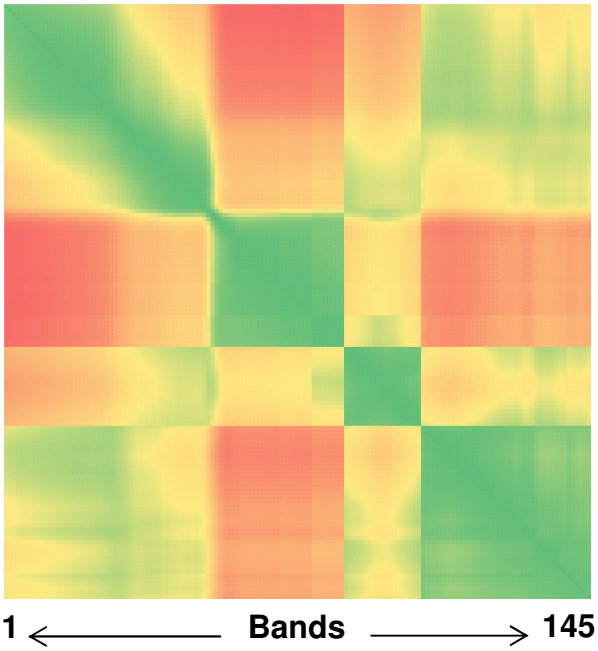


Figure 3-13. Spectral Band Correlation

Clearly some structure between bands is present in the data. Most obvious is the fact that the green regions, of highly correlated bands, are aligned primarily along the diagonal. This result is to be expected from simple observation of a set of single band images as depicted in Figure 3-14. This arrangement of images plainly shows the strong correlation between images in neighboring spectral bands. At the same time it indicates the repetitive nature of the information contained in the hyperspectral dataset. As will be seen in the next section the repetitive nature of

the hyperspectral data allows for a rapid yet dynamic approach to grouping closely related portions of the dataset.

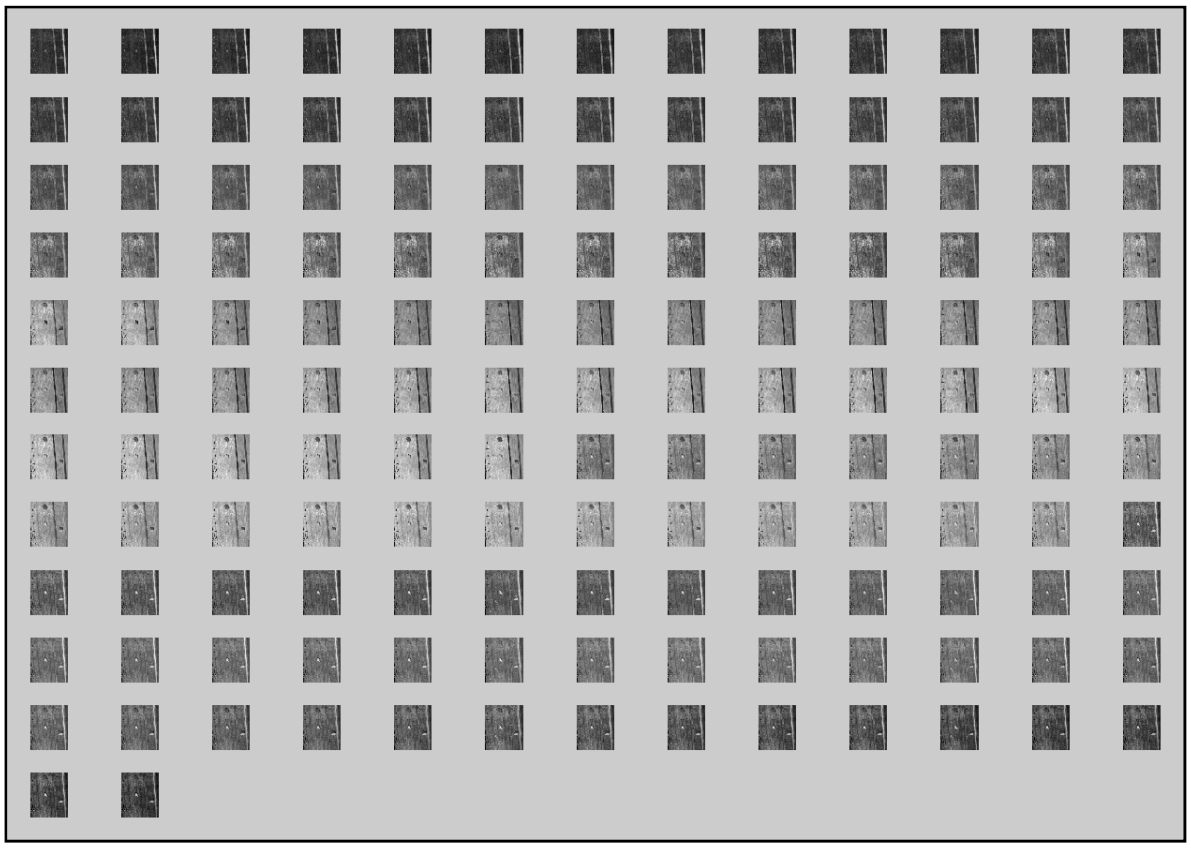


Figure 3-14. 145 Single Spectral Band Images

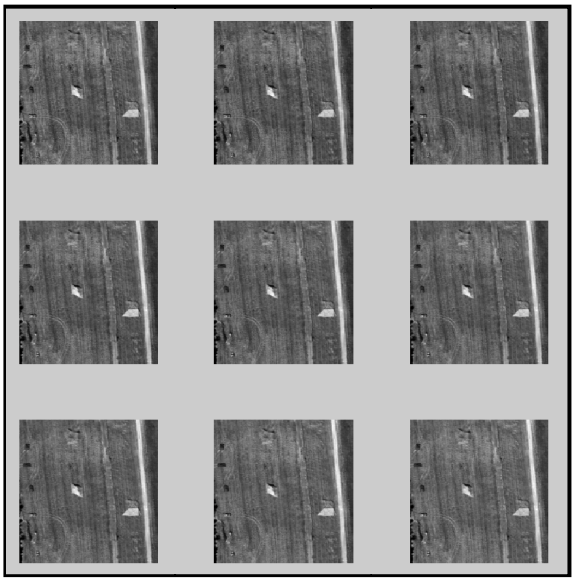


Figure 3-15. 9 Highly Correlated Neighboring Bands

When the correlation colorgraph is reoriented so that the matrix diagonal is aligned vertically, observation of the arrangement of the highly correlated sections of the graph indicates the likely number of clusters an algorithm ought to form. In Figure 3-16, thirteen clusters have been identified by simple visual inspection. As Williams indicated retention of every band forces subsequent processes to unnecessarily manipulate duplicate data. Rather than arbitrarily discarding four of every five spectral bands, or attempting to produce a complete correlation matrix for each processed image, the arrangement of highly correlated bands along the diagonal will be exploited to perform comparisons only between bands in relatively close spectral proximity.

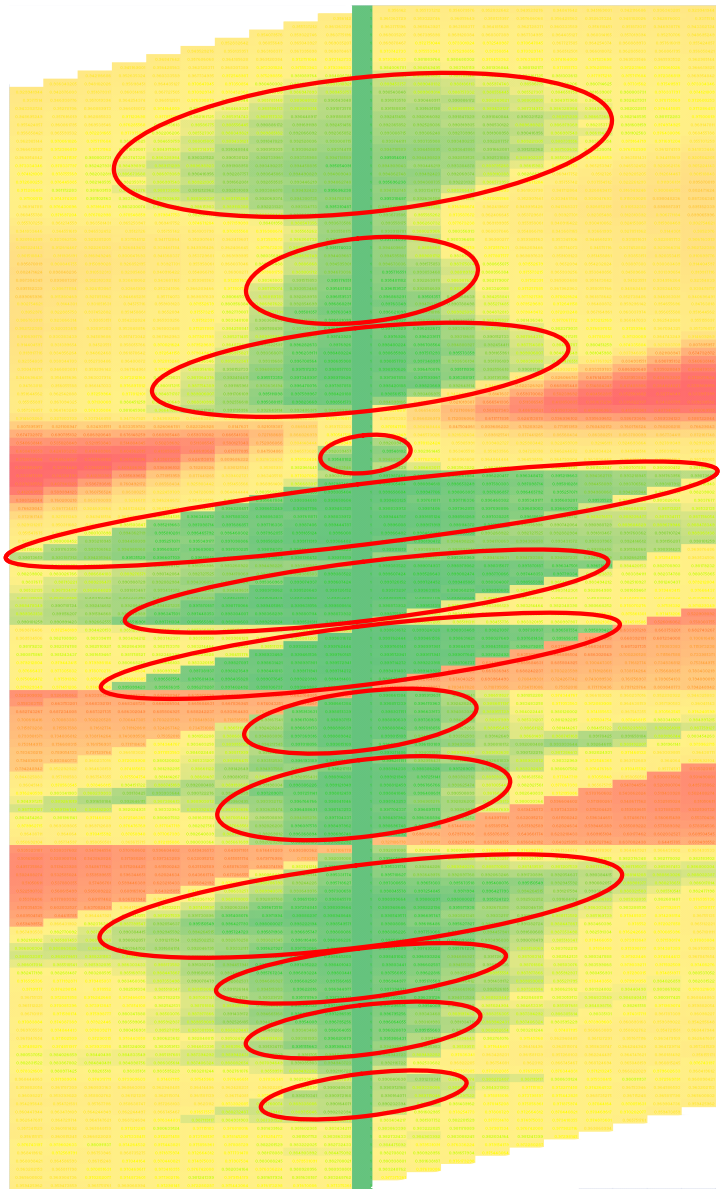


Figure 3-16. Correlation Plot with Diagonal Oriented Vertically

This is accomplished by beginning with band 1 and calculating its correlation with band 2. If the correlation between the two bands exceeds the desired threshold, the two bands are retained as a cluster, and the correlation is calculated between band 1 and 3. Bands are clustered together until the intra-band correlation drops below the user defined Correlation Threshold. Then if two or more bands have been clustered, the last band which exceeded the target threshold

is assumed to be the center of the cluster, and the process begins a second time by finding the correlation between the center spectral band and the adjacent band. Clustering continues as before until the correlation between the center band and the n^{th} band in sequence again falls below threshold. At this point all spectral bands in the cluster are averaged together, to be returned as one of k clustered spectra. Bands which do not meet the intra-band correlation requirement with at least one nearest neighbor are discarded.

3.3.2. Comparison of Spatial Clustering to Spectral Clustering

Figures 3-17 and 3-18 provide a comparison of K-means clustering in the spatial dimension to the spectral clustering algorithm developed as part of this thesis. K-means clustering (Figure 3-17) operates by locating individual pixels with similar spectral signatures and associating them into one of k clusters.

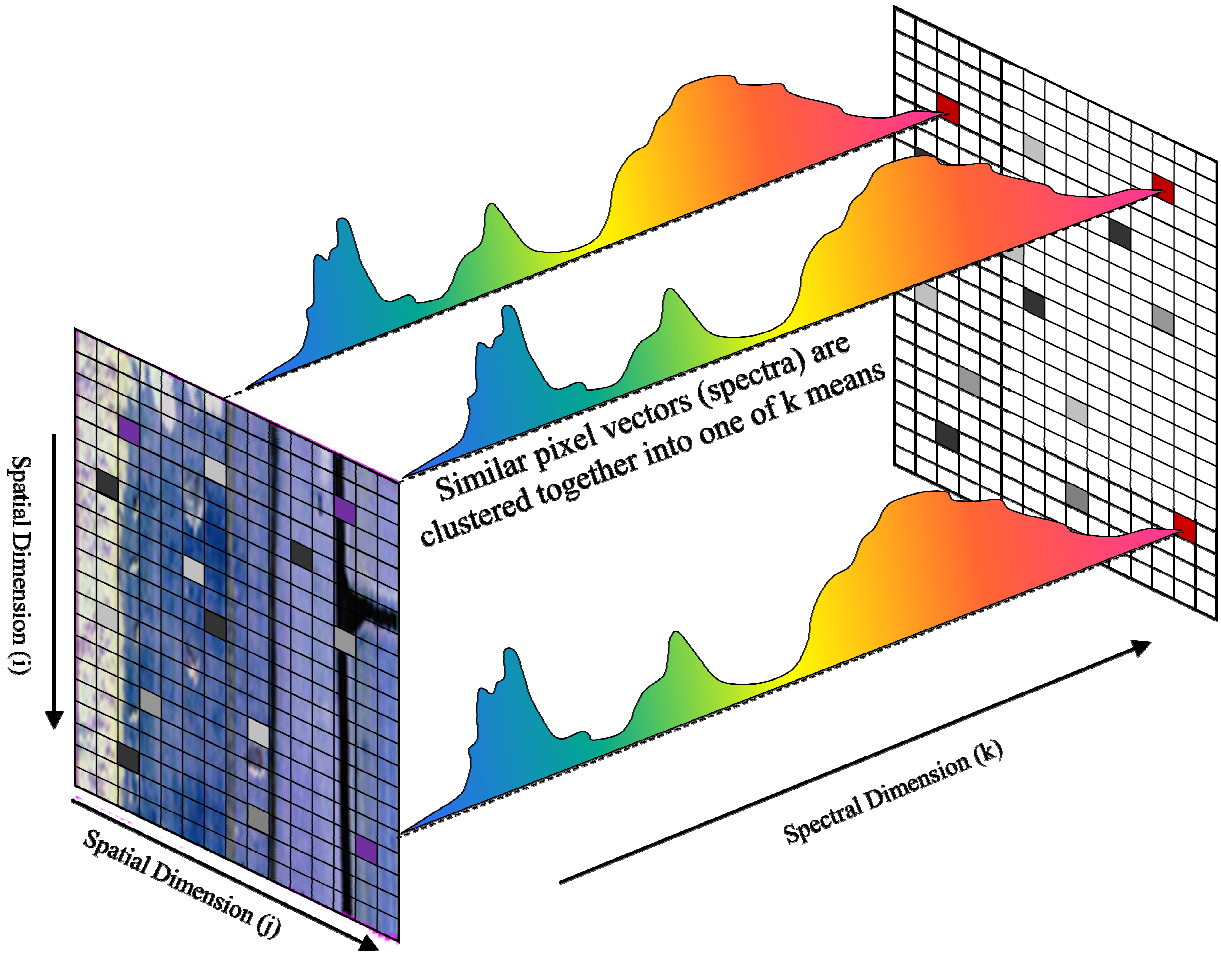


Figure 3-17. K-means Clustering of Similar Pixels

Figure 3-18 makes clear the primary advantage provided by clustering in the spectral dimension. Rather than seeking pixels with similar spectral signatures, this approach seeks adjacent spectral bands where corresponding pixels produce similar reflectances. This eliminates the need to make an assumption of no a priori knowledge about the likelihood of two bands being similar. Instead an assumption is made that adjacent bands are more likely to be similar than widely separated bands.

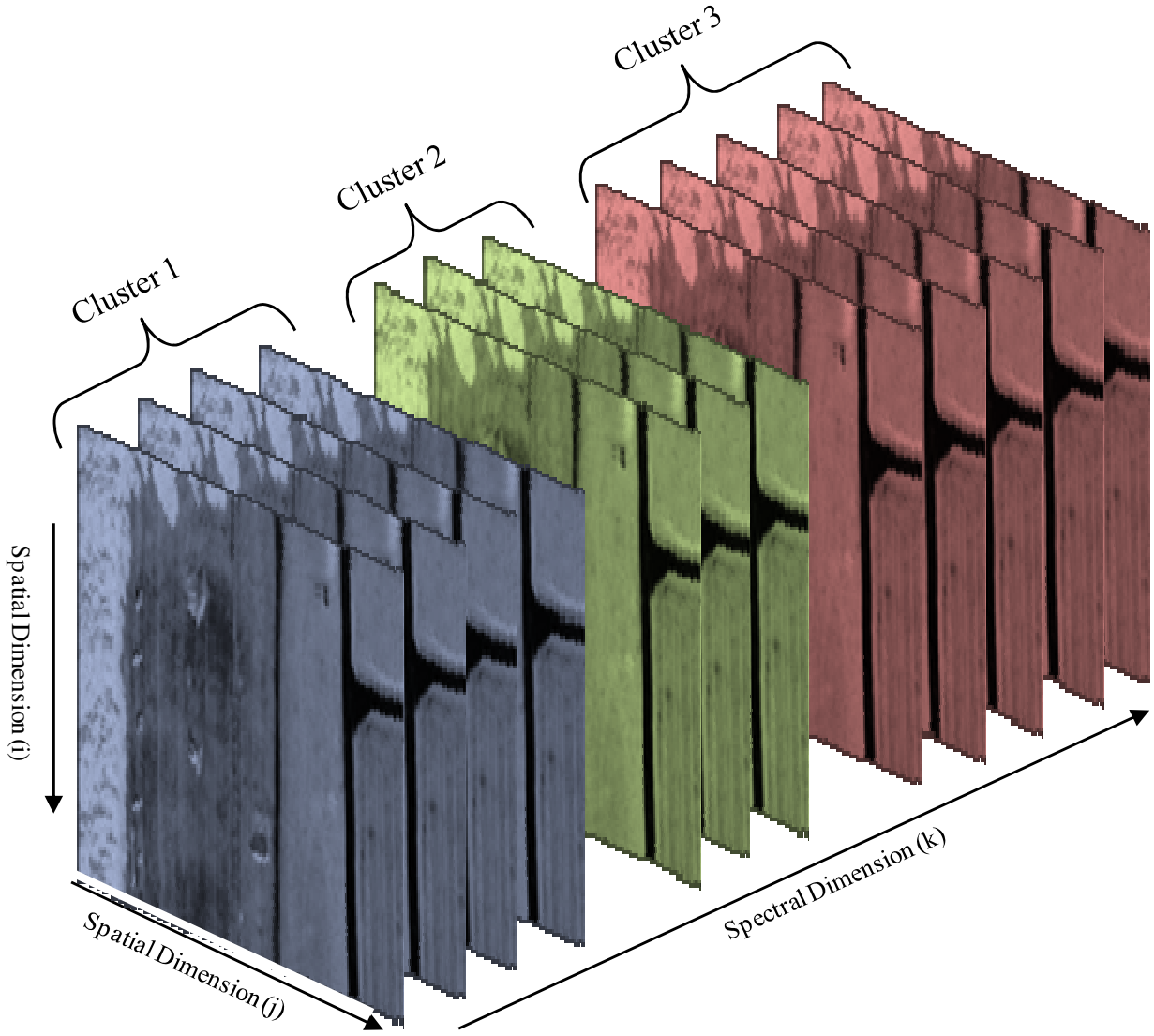


Figure 3-18. Spectral Clustering of Similar Spectral Bands

This approach provides several advantages over previous clustering methodologies. First, the algorithm is entirely deterministic. Spatial clustering begins with some random set of cluster center points and iteratively rearranges the clusters to approach a minimized difference between clustered elements. As a result the random selection of center points adds a stochastic element to the outcome. This particular spectral clustering algorithm begins at the first spectral band and only clusters bands which exceed the assigned correlation threshold. There are no stochastic

inputs and the algorithm will repeatedly return the same set of clustered spectral bands given a specific correlation threshold. This fact contributes to the second advantage, speed. Because the algorithm is deterministic, all clusters are formed in a single iteration of calculating correlations along the diagonal. In fact, whereas a complete correlation matrix of 145 spectral bands requires 21,045 intra-band correlations to be calculated, each of which might require calculation of the correlation between two 1x30,000 vectors, by searching for clusters only among neighboring spectra only 145 intra-band correlations are calculated. It does this by calculating correlation values down the first superdiagonal of the correlation matrix. Once a correlation value is found which exceeds the threshold allowing the band to be retained and averaged with adjacent bands, the algorithm continues along the same row within the correlation matrix, calculating values until a band is insufficiently correlated with the starting band to considered "alike". The algorithm establishes this band as the "center point" for averaging and continues by calculating that bands correlation with adjacent bands, until the intra-band correlation again falls below the established threshold. Figure 3-19 provides a visual depiction of how this works. Bands contained in the rectangles exceed an intra-band correlation threshold of 0.985 relative to their center points and were clustered into 15 dimensions. Bands at the center of each cluster are identified by red triangles. Several bands at either end of the correlation colorgraph are not members of any cluster. These bands are adjacent to noise containing absorption bands and failed to exceed the correlation threshold with any immediately adjacent bands. Because less correlated bands tended to be near noise containing portions of the hyperspectral image, any band not exceeding the correlation threshold with at least one of its immediate neighbors was discarded.

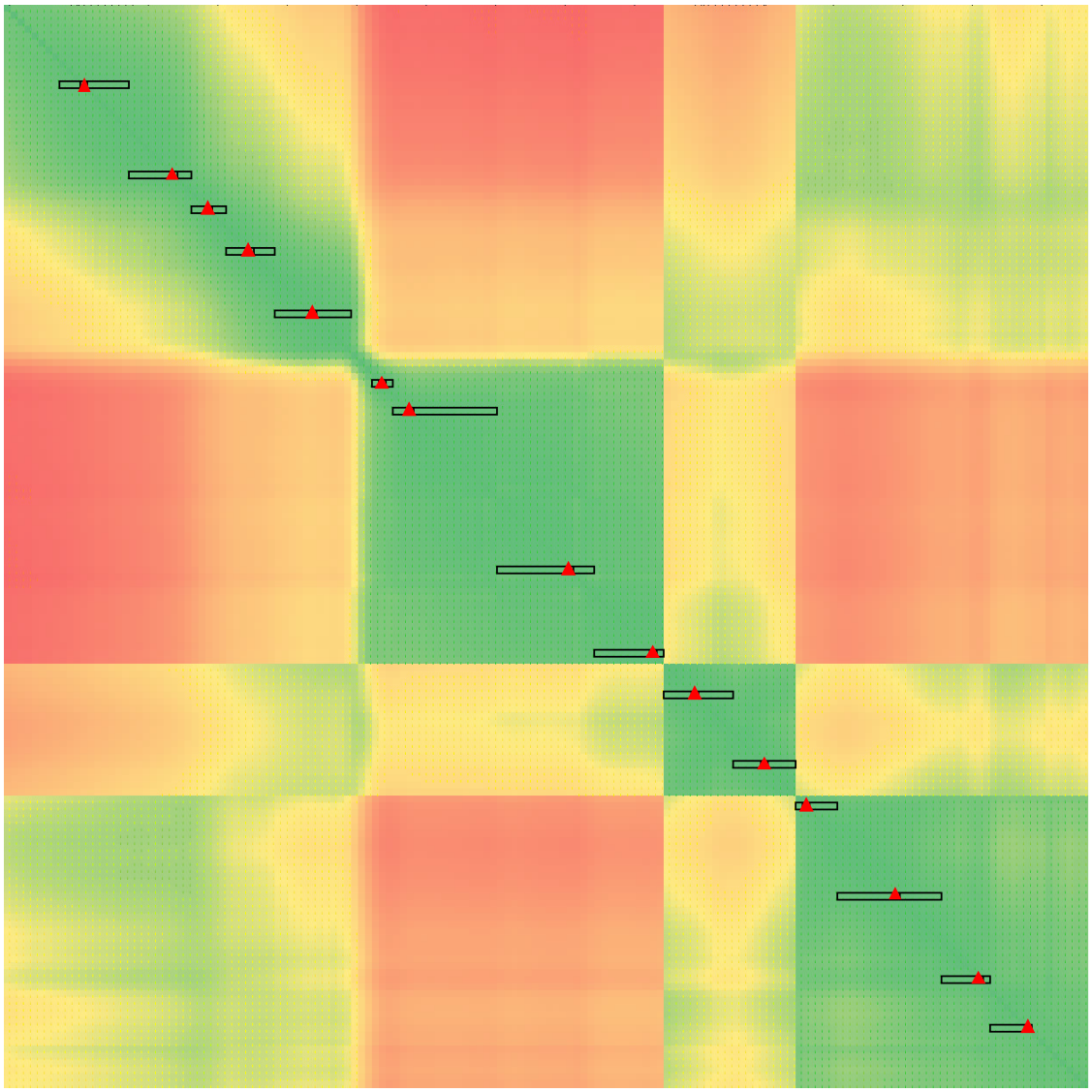


Figure 3-19. Correlation Colorgraph Depicting Clustered Bands

The third advantage is the elimination of any requirement to estimate the number of expected clusters beforehand. Rather the predefined required correlation drives how many or how few clusters will be generated. A low required correlation produces fewer large clusters, while a high intra-band correlation requirement produces many small clusters. Finally, the algorithm overcomes non-static covariance by associating spectral bands into dynamically sized clusters. As few as two spectra may be clustered together if no other spectra are sufficiently correlated. On the other hand, if a single series of bands are highly correlated, clustering

continues until the similarity between bands drops sufficiently. This reduces processing time by preventing duplicate information from being passed into ICA for further processing.

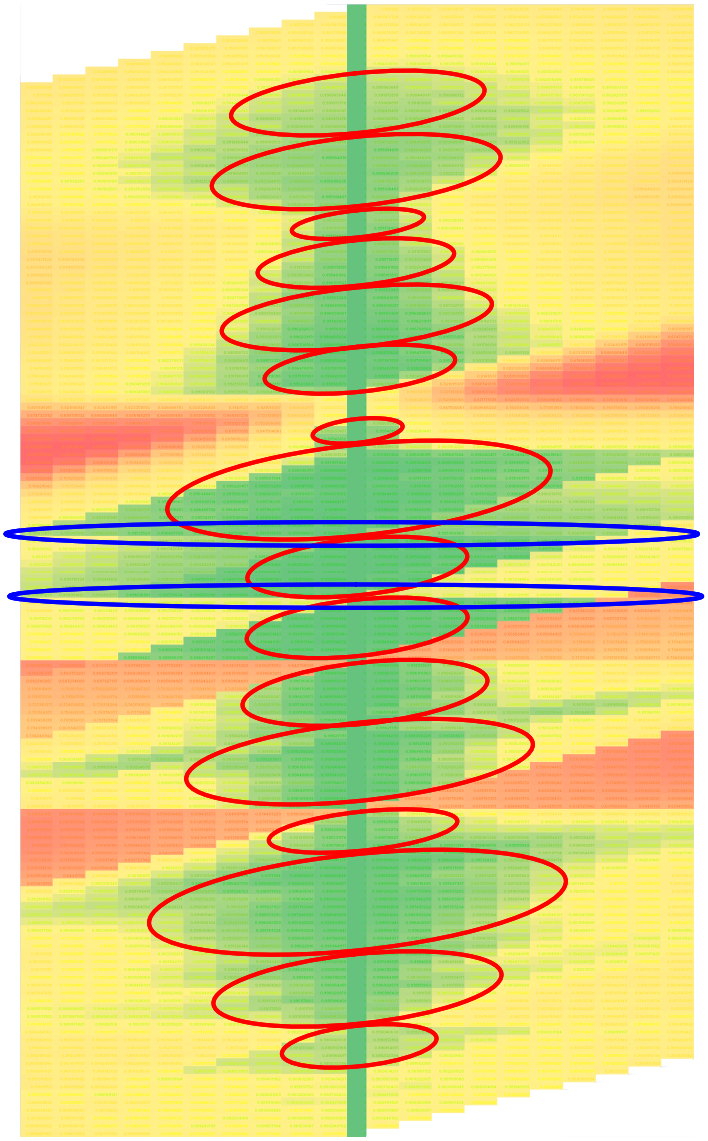


Figure 3-20. Correlation Plot Clusters as produced by Spectral Clustering Algorithm

Figure 3-20 displays the 16 clusters identified by the spectral clustering algorithm from the 145 spectral bands remaining after removal of the absorption bands defined by Smetek [2007:23]. Interestingly two of the returned clusters "straddled" absorption bands, by including spectral bands from either side of the removed sections. The location of the two absorption

bands is denoted by a blue oval around the two spectral bands on either side of the absorption band. This observation led to a closer look at the removed absorption bands. Figures 3-21(a) and (b) show the so called absorption bands with one non-absorption band on either end. Both of the removed sections provide no obvious indication of noise, or lack of information present in the data in these regions. This same inspection was conducted for all eight available data sets and in each case, no cause for removal of the bands could be found. Additionally, it was recognized that if in the future noise were present in any band, as would be expected along absorption bands, the algorithm would find insufficient correlation between bands to result in clustering two or more bands. This would then result in the removal of any problem spectral bands. With this in mind these two absorption bands were included in all eight hyperspectral datasets as usable.

Figure 3-21(a). Absorption Bands 73 – 77 with Bands 72 & 78

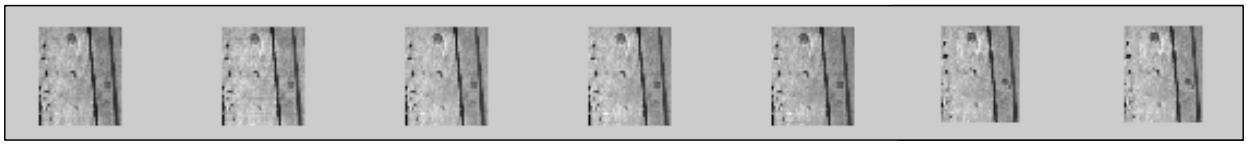


Figure 3-21(b). Absorption Bands 87 – 91 with Bands 86 & 92

Following reincorporation of bands 73-77 and 87-91 the clustering algorithm was reapplied to the 156 spectral bands. Figure 3-22 displays the 15 clusters identified by the algorithm. While this is slightly more dimensions than might have been expected by visual inspection of the colorgraph, it was found to be acceptable and no further attempts were made to increase the size of the clusters produced by the algorithm. The individual spectral bands contained in these clusters were averaged together and provided to ICA as a 15 dimension dataset for processing to

produce the most independent rotation of the remaining data. By taking this approach, PCA was eliminated completely, as was the requirement to assess the dimensionality of the data from an eigenvalues plot.

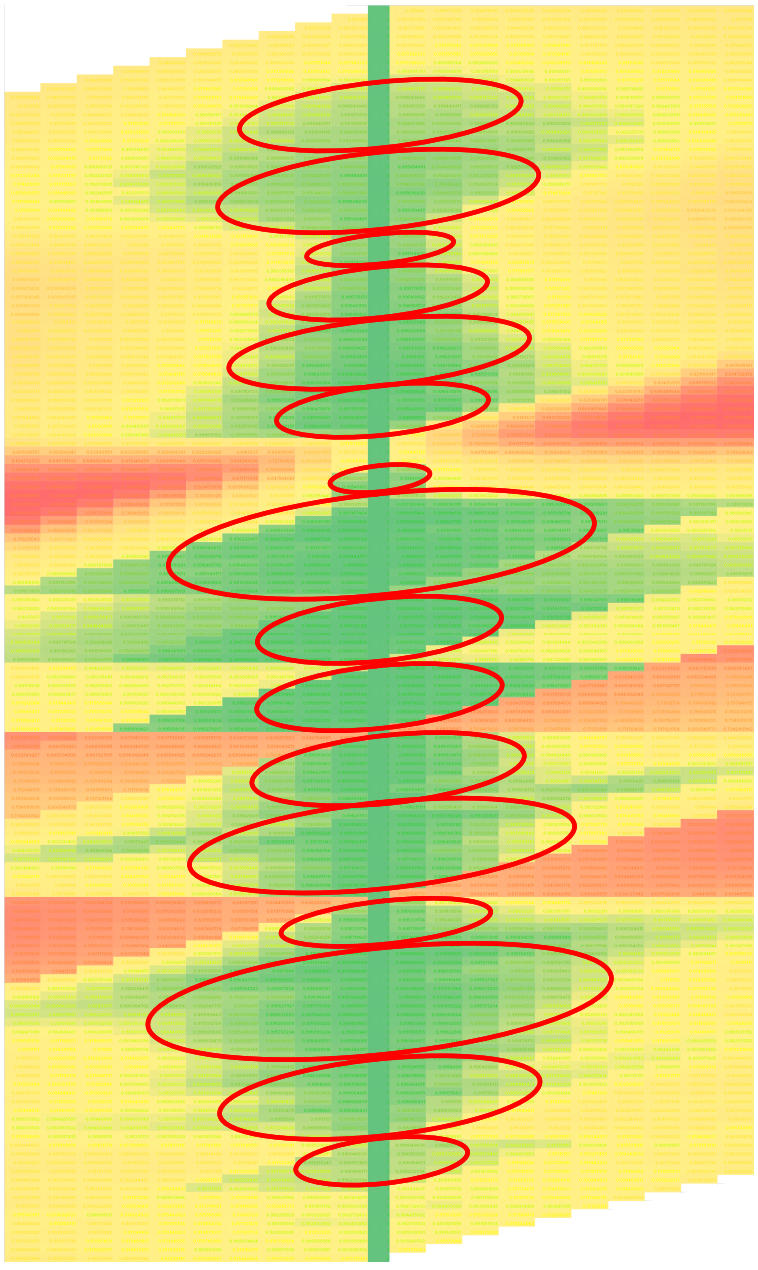


Figure 3-22. Correlation Plot Clusters produced by Spectral Clustering Algorithm (156 bands)

Figures 3-23 through 3-25 depict the first three steps of the modified AutoGAD algorithm, Preprocessing, Feature Extraction I, and Feature Extraction II. The only change made to the preprocessing portion of the algorithm is the increase in the number of non-absorption bands retained from 145 to 156. The first portion of feature extraction is changed substantially from the original, shown in figure 2-10, by replacing PCA and MDSL with the clustering algorithm described above. Feature Extraction II can be viewed as the engine of both original AutoGAD and the updated AutoGAD by correlation. In this step the FastICA algorithm is called on to solve for the abundance matrix and then used to unmix independent components of the clustered dimensions found in the previous step.

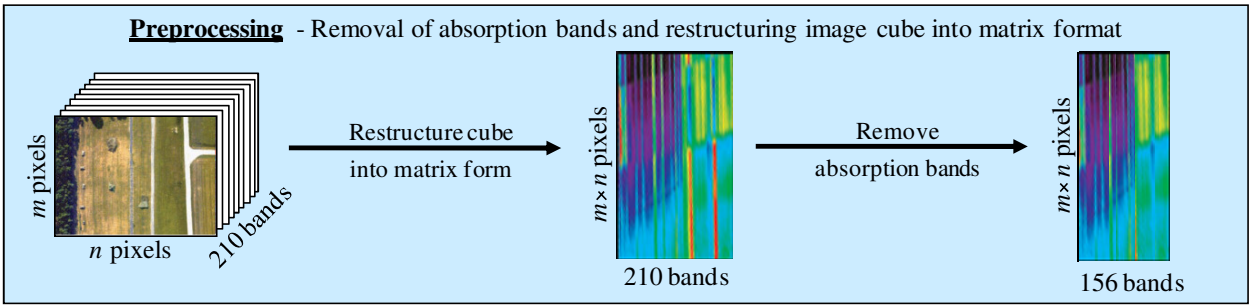


Figure 3-23 AutoGAD-SC Preprocessing of Hyperspectral Data Cube

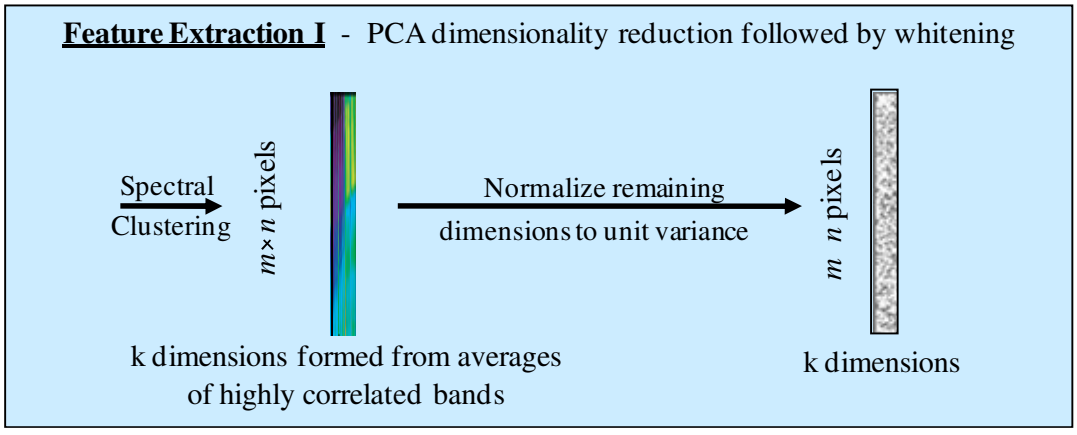


Figure 3-24 AutoGAD-SC Feature Extraction I (Clustering, and Whitening)

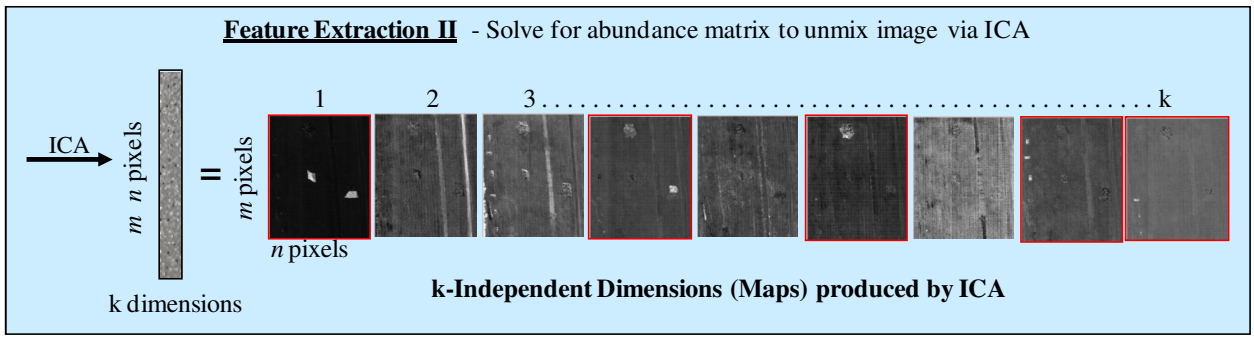


Figure 3-25 AutoGAD-SC Feature Extraction II
(ICA with sample of resulting independent components in red)

3.4 Separation of Target Pixels from Background

Once ICA completes the process of extracting independent features from the original spectral signals, the characteristics of these independent features themselves is used to identify which features are likely to contain targets and which features are not. Four characteristics of each signal are measured and compared against user defined thresholds in an attempt to retain only target containing maps. These four characteristics include maximum component score, potential target signal to noise ratio (PT SNR), kurtosis, and potential target fraction (PTF). Maximum component score is simply a direct measurement of the maximum single pixel score returned by ICA. Likewise, kurtosis is a direct measure of the fourth moment of inertia for each signal. Both kurtosis and PTF require the formation of a signal histogram relating component scores for each pixel to the frequency of each scores occurrence in each component. In order to construct the set of signal histograms a suitable component score bin width must be selected. AutoGAD used a user defined histogram bin width with a recommended setting of 0.05 [Johnson: 2008,114]. This parameter was replaced in AutoGAD-SC by a simple means for estimating a distribution's optimal bin width, Scott's Rule.

3.4.1. Scott's Rule for Histogram Bin Width

As discussed in section 2.4.1, Scott's rule for determination of the optimal histogram bin width [1979] was employed in place of an arbitrary user defined bin width setting. Derivation of the formula for optimal bin width

$$h_n = 3.49sn^{-1/3} \tag{0.30}$$

requires knowledge of the true underlying distribution, however Scott refers to work by Tukey [1977:623] which suggests assuming a Gaussian distribution as the reference standard for the underlying distribution. Scott recognized that the assumption of an underlying Gaussian distribution as the source of the data may not be accurate in many circumstances, and points out that use of the equation on non-Gaussian data will not produce a "Gaussian" looking histogram. He then demonstrated the use of equation 3.1 on several non-Gaussian data sets and provided a graphical method to adjust for non-Gaussian skewness, kurtosis, or bimodality [Scott:1979, 608]. An approximation of these correction factors was added to the algorithm, so as to produce the best possible bin width estimate based on the data contained in each component.

3.4.2. Estimating the Threshold between Background and Signal

One of the primary reasons for creation of these histograms is the estimation of breakpoint between those pixels representing background and outlying target returns. Recall from section 2.4 that this breakpoint was identified by locating the first zero bin on the signal histogram. All pixels with a component score greater than this threshold were identified as potential target pixels, while all others were assumed to be likely background pixels. This approach presents two related problems associated with estimating this threshold. Essentially the separation between background and outliers is found at the point on the signal histogram where

the transition between peak and tail occurs. Figure 3-26 shows a signal histogram from ARES 1D with each pixel color coded to identify its status as target or background. Figure 3-27 shows an enlarged view of the region of interest. If the first zero bin method is used to identify the breakpoint between peak and tail, the resulting threshold between target and background pixels is at a component score of 7.0034.

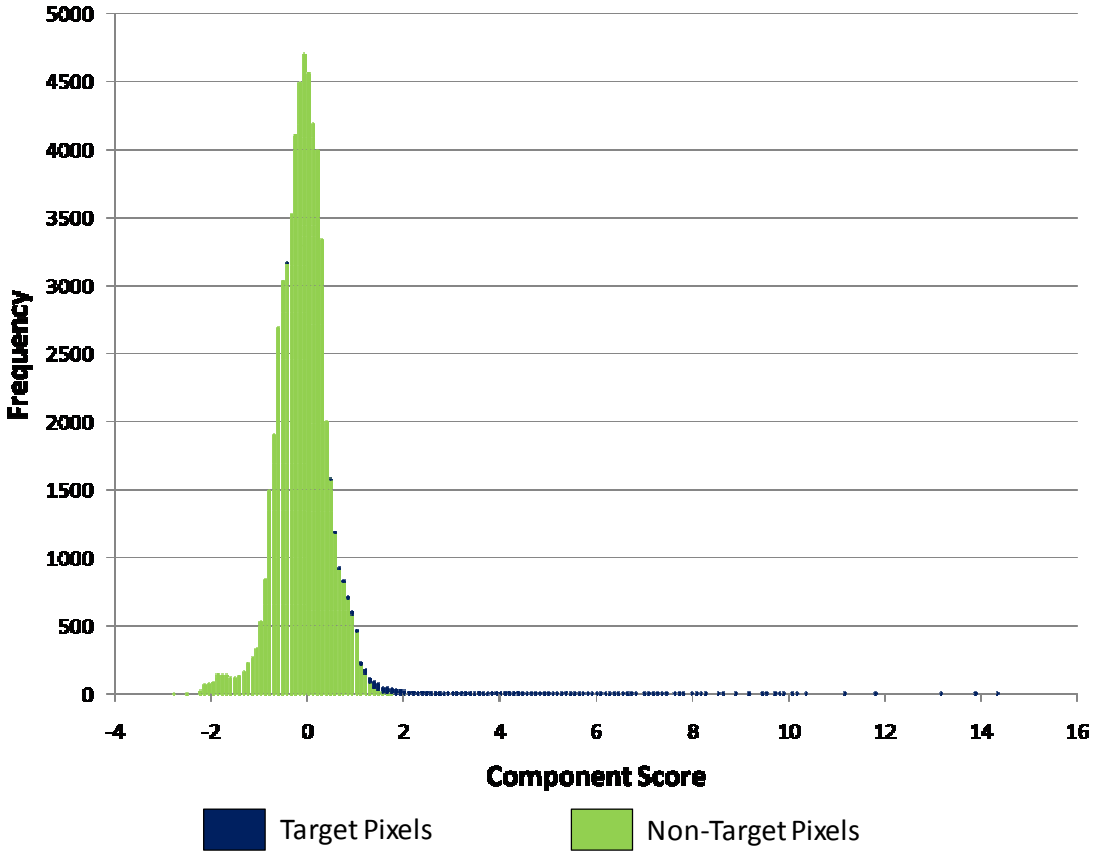


Figure 3-26. ARES 1D Signal Histogram

Clearly when the first zero bin method is applied to this particular data set, it fails to include a significant number of actual target pixels. In fact on this particular map with the threshold set at 7.0034, only 43 of 672 target pixels are correctly identified. The problem in this situation is twofold, the location of the first zero bin is heavily dependent on the selected bin

width, and second the first zero bin frequently does not occur until well into the "tail" of the histogram.

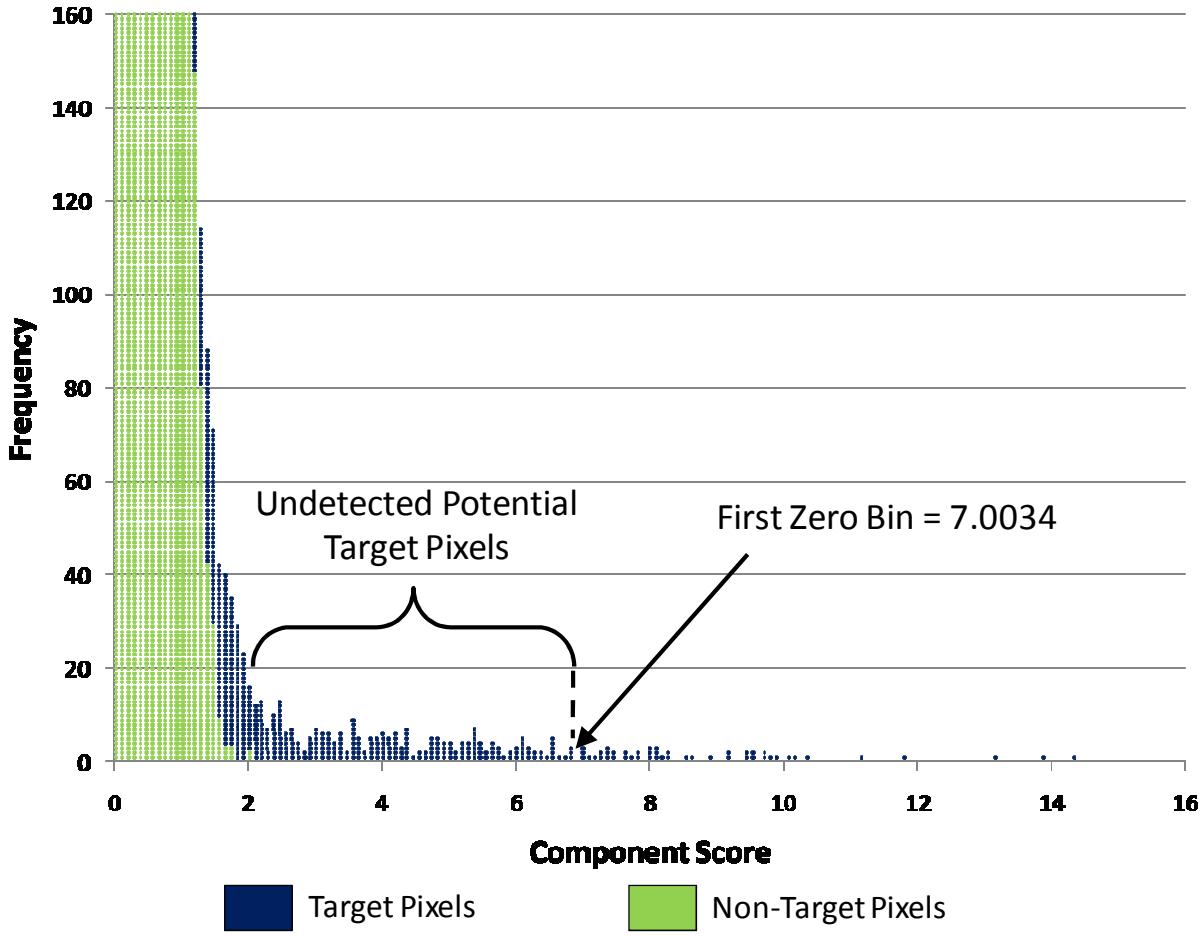


Figure 3-27. ARES 1D Signal Histogram with First Zero Bin Identified

Instead of the identifying the first zero bin as the transition between background and outlier type pixels, this thesis attempted to locate the "knee in the curve" between the non-Gaussian mound and tail portions of the signal histogram. To accomplish this Johnson's MDSL technique was adapted to measure the maximum distance between the peak of each histogram and the maximum signal score location. Figure 3-28 shows a graphical depiction of how this algorithm selects the threshold between background and potential targets. Figure 3-29 narrows in on the knee in the curve and shows that this method estimates the threshold to be a component

score of 1.5753. Using this threshold 435 of the 672 target pixels are correctly identified as potential targets with 6 non-target pixels incorrectly included as part of the potential target signal. This represents a 912% increase in potential target signal identification which has a dramatic impact on the PT SNR value as will be demonstrated in the next section.

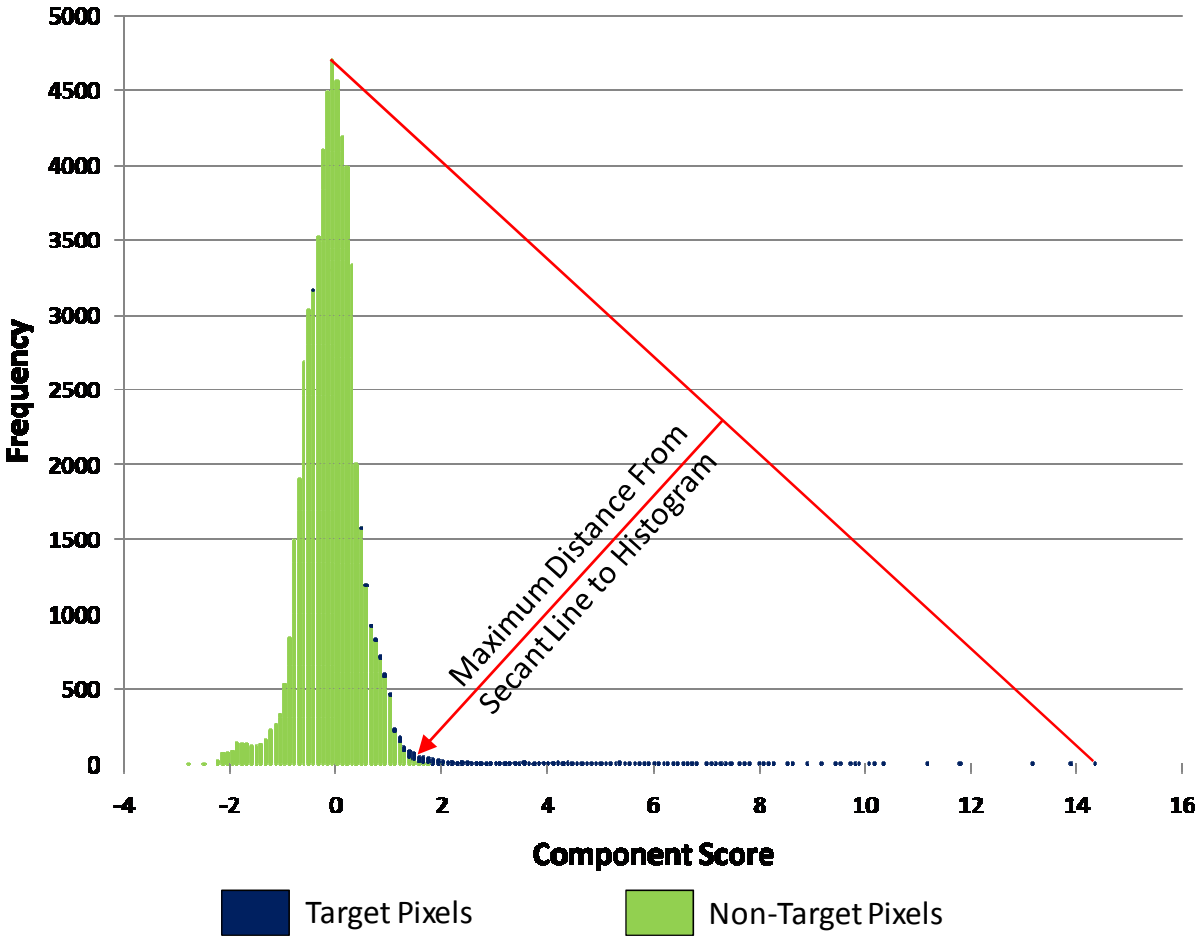


Figure 3-28. ARES 1D Signal Histogram and MDSL technique for signal-background threshold estimation

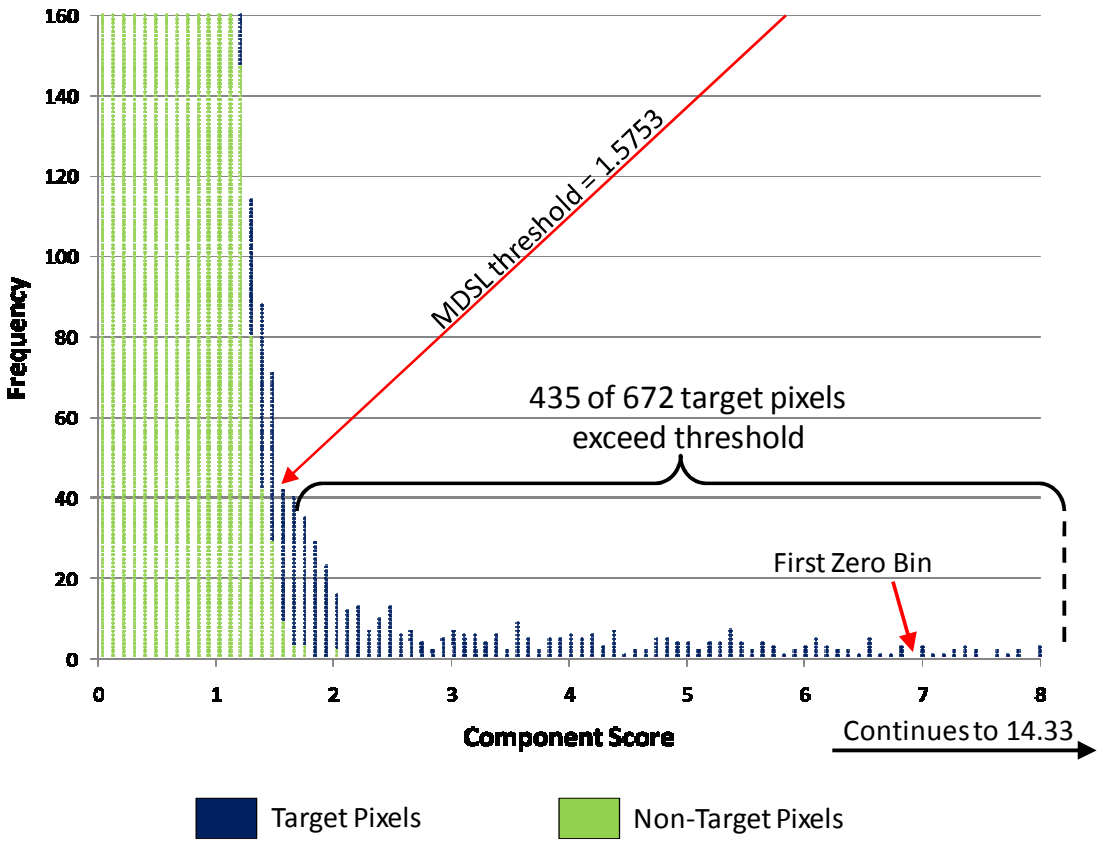


Figure 3-29. ARES 1D Signal Histogram with MDSL threshold identified

By adopting the MDSL method, the need to correct for the fact that Scott's rule assumes an underlying Gaussian distribution is somewhat alleviated. Correcting for non-Gaussianity primarily serves to reduce bin width so as to make the resulting histogram a somewhat rougher appearance, but the general shape remains the same. As a result the threshold defined by MDSL changes only very slightly when the correction factor is applied. Figure 3-30 shows the effect of applying Scott's correction factor on location of the threshold between signal and background pixels. Of the five target containing maps, the largest affect the correction produces is a shift from a threshold at a component score of 1.75 with the correction to 1.862 without the correction. In fact, inclusion of the correction factor actually generated a slight reduction in the number of target pixels found. Figure 3-31 shows that without the correction factor the

algorithm detected 97.38% of all target pixels, but that with the addition of the correction factor the number of target pixels detected reduced slightly to 97.22%. Because this correction factor requires calculation of skew for each signal distribution and would then reduce bin width to correct for non-Gaussian skew, it tended to increase algorithm run time. Since the process produced only a relatively small impact on the location of the threshold between signal and background, provided no improvement in target detection, and came with an associated time penalty it was removed as unnecessary.

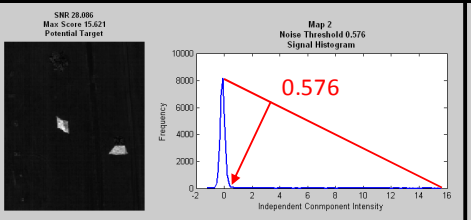
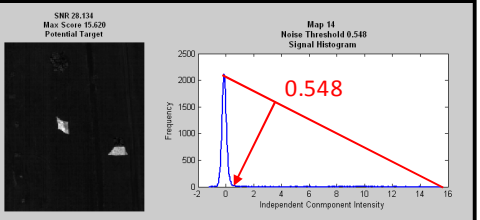
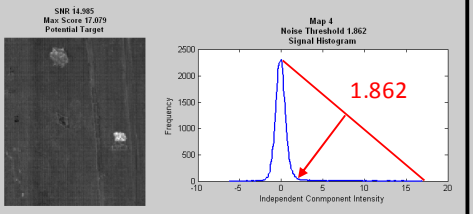
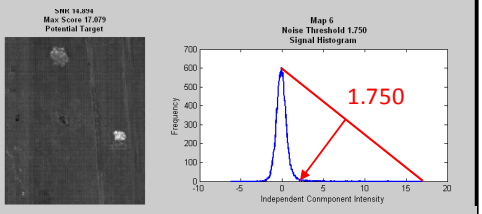
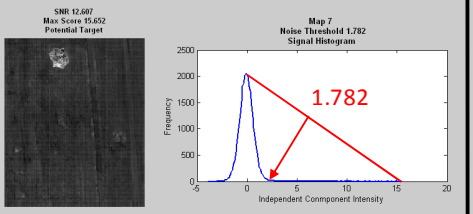
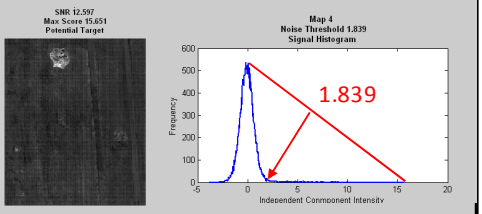
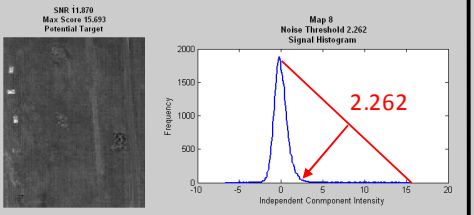
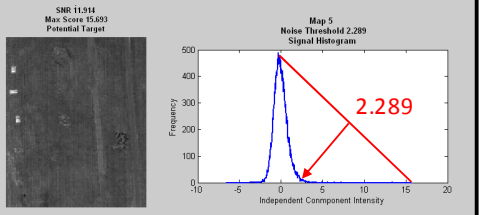
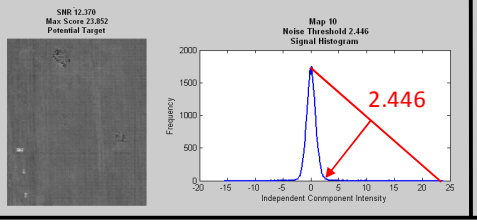
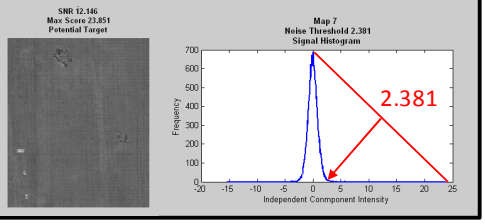
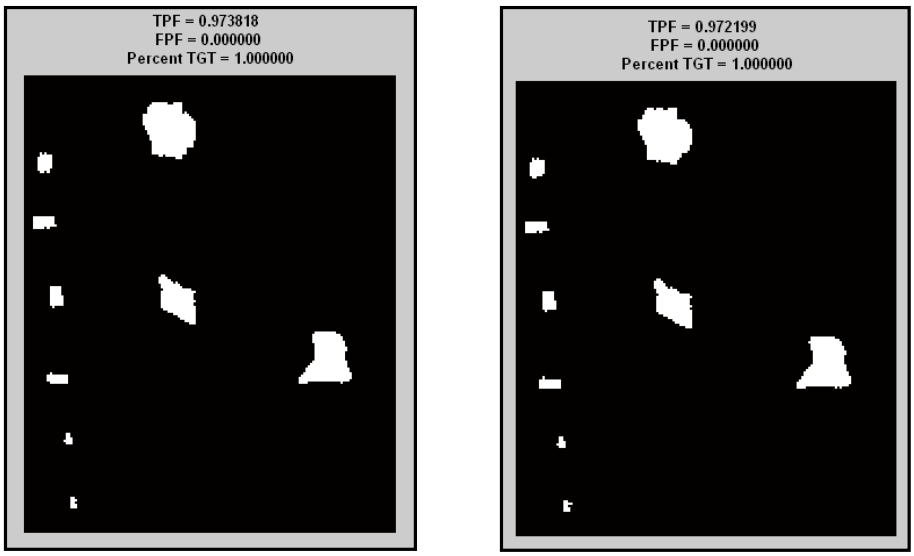
Target Component	Without Non-Gaussian Correction	With Non-Gaussian Correction
1		
2		
3		
4		
5		

Figure 3-30. Comparison of Signal/Background Thresholds developed with and without adjusting for non-Gaussian behavior while estimating bin width by Scott's Rule



(a) Without non-Gaussian Adjust (b) With non-Gaussian Adjust

Figure 3-31. Comparison of Target Pixels found with and without Scott's Rule Bin Width Adjust for Underlying non-Gaussian Distribution

3.4.3. Calculation of Signal to Noise Ratio

Once the line separating potential target pixels and background pixels has been defined the signal to noise ratio is measured. As with Johnson's AutoGAD algorithm PT SNR is calculated by the following formula

$$\begin{aligned} PT\ SNR_{dB} &= 10 \bullet \log_{10}\left(\frac{\text{power(potential target signal)}}{\text{power(background)}}\right) \\ &= 10 \bullet \log_{10}\left(\frac{\text{var(potential target signal)}}{\text{var(background)}}\right) \\ &= 10 \bullet \log_{10}\left(\frac{\sigma_t^2}{\sigma_b^2}\right) \end{aligned} \tag{0.31}$$

but use of the Maximum Distance Secant Line rather than first zero bin to define the threshold generates a substantial difference in signal to noise values for each map. Figure 3-32 shows the component scores for all pixels from ARES 1F, in each of its 15 dimensions. The horizontal axis

corresponds to the pixels left-right position in the image, while the vertical axis corresponds to each pixel's component score. The solid line represents the location of the breakpoint between potential target signal and background as found by the MDSL technique, while the dotted line indicates the location of this breakpoint as found by the first zero bin method. Note that in each map the first zero bin method identifies a higher IC score (further into the tail) as the threshold. For reference the 15 associated image maps are provided in figure 3-33.

The shift upwards in this threshold tends to produce lower PT SNR values based on the fact that many fewer target pixels, which are more variant than background pixels, are included in the PT SNR measurement. Two components do not follow this trend, however. Both Map 2 and Map 10 produce larger PT SNR values when the first zero bin is applied, than by the MDSL method, yet neither of these maps contain true target pixels which lie far enough from background to be correctly identified. In addition target pixel containing Map 7 produces a PT SNR value of only 6.533 when measured by the first zero bin, while Map 2 produces a PT SNR value of 6.789. Thus any threshold for discriminating between target and non-target maps based on PT SNR that is set low enough to capture Map 7 must also retain Map 2. However when the MDSL technique is applied, any PT SNR threshold between 8.992 (the highest PT SNR for a non-target map) and 11.870 (the lowest target map PT SNR value) correctly captures all target pixel containing maps (6, 7, 13, 14, and 15) while discarding all others.

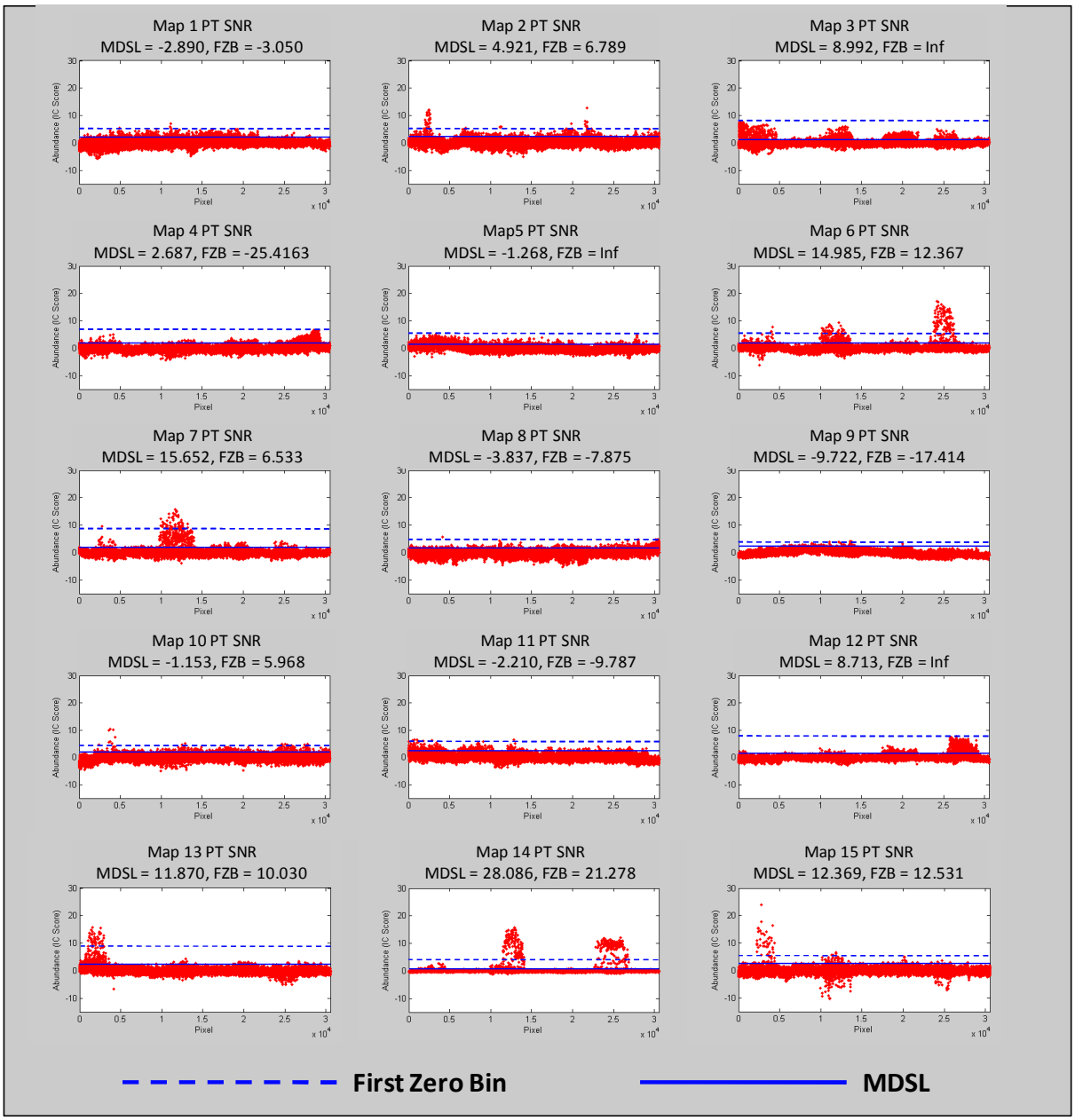


Figure 3-32. Potential Target Signal-Background Thresholds for ARES 1F by MDSL and First Zero Bin Methods and their associated PT SNR values

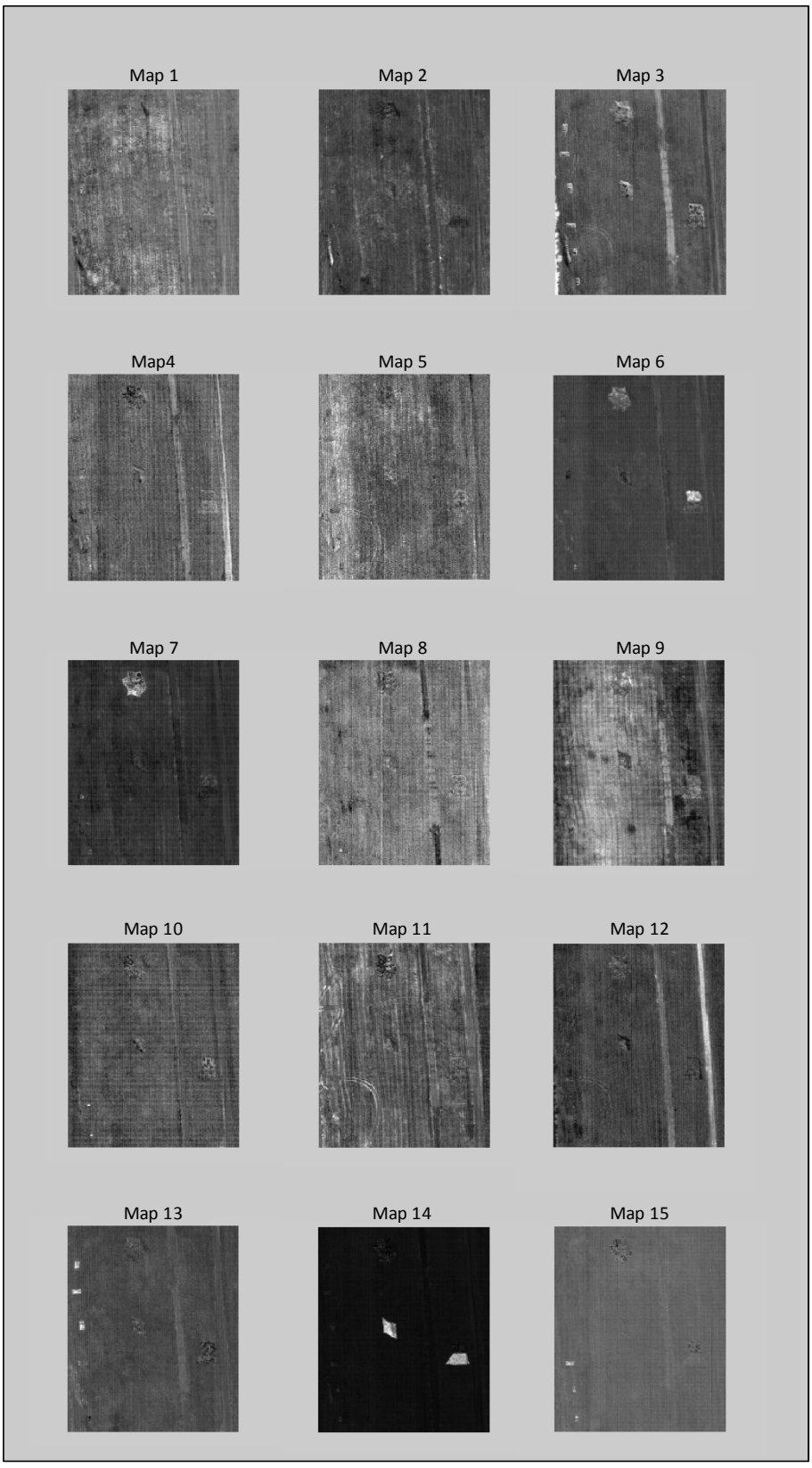


Figure 3-33. Maps from ARES 1F associated with PT SNR Tables in Figure 3-32

3.4.4. Potential Target Fraction (PTF)

The final measure used to discriminate between signals containing likely target pixels and signals that are not likely to contain targets is the Potential Target Fraction (PTF). PTF is defined for each map as the number of pixels with a component score greater than the signal threshold divided by the total number of pixels. This parameter provides the user with some ability to define how densely targets of a single type material are expected to be placed within the scene. More importantly by correctly setting a maximum PTF, it prevents common naturally occurring anomalies, such as sagebrush, from being defined as target pixels because they occur too frequently.

Figure 3-34 shows this process for ARES 1D. Note that any map with its PT SNR value or maximum IC score displayed in red is recognized as a non target map based on the value in red. Map 1 contains primarily actual target pixels and little else. The algorithm calculates the Potential Target Fraction (PTF) at 1.124% of the total image. This target volume is in concert with what might be expected for man-made targets within the image. Maps 4 and 8 both represent cases where the PT SNR values and Maximum IC Score exceed those of Map 1, and would likely be retained for further processing if PTF was not considered. However both of these maps have a substantially higher concentration of potential target pixels than Map 1. Their PTF values are found to be 4.024% and 3.766% respectively. By setting a PTF threshold of 3.5%, these two maps will be excluded from further processing, eliminating any false positive pixels they might identify as targets.

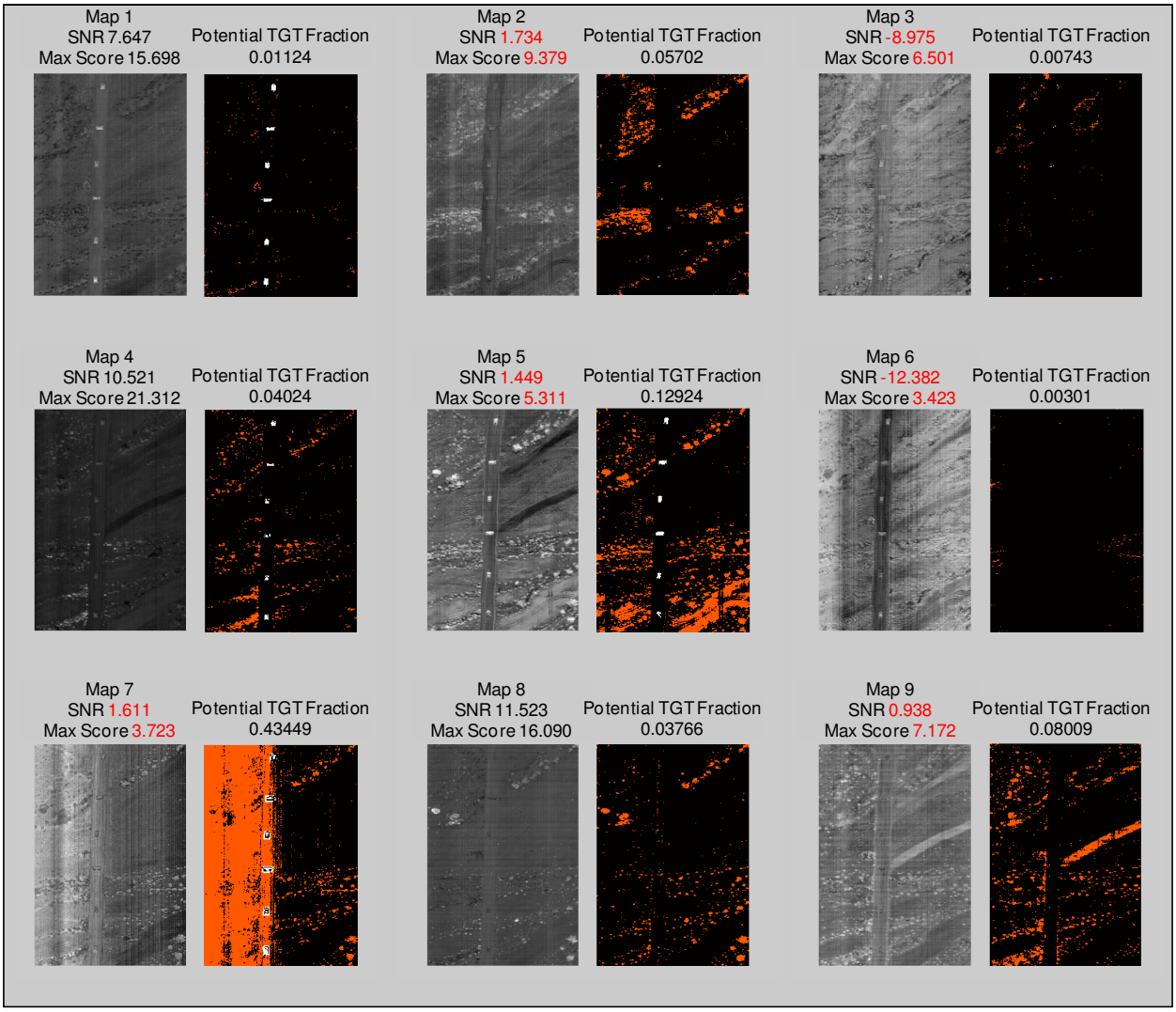


Figure 3-34. ARES 1D Component Images, Potential Target Images, and Potential Target Fractions

3.5. Discrimination between Target and Non-Target Components

AutoGAD discriminated between target maps and non-target maps based on the maximum IC score and the PT SNR value. Johnson demonstrated in Figure 3-35 that by setting a PT SNR threshold of 2 dB and a max IC score of 10 to ARES 1D, 1F, 1D, and 2D, all target maps are retained, while only two non-target maps are incorrectly retained. This same analysis

was accomplished, with the addition of kurtosis and PTF as threshold options. Results are shown in Figures 3-36 and 3-37.

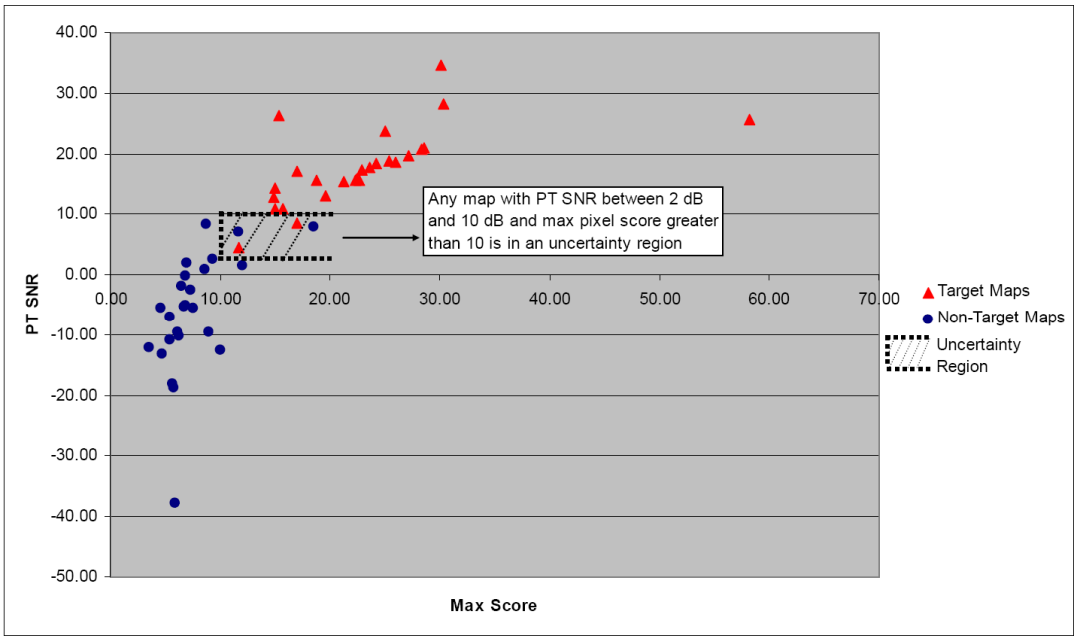


Figure 3-35. AutoGAD Uncertainty Region in Max Score and PT SNR Feature Space (ARES 1D, 1F, 2D, and 2F) [Johnson: 2008:160]

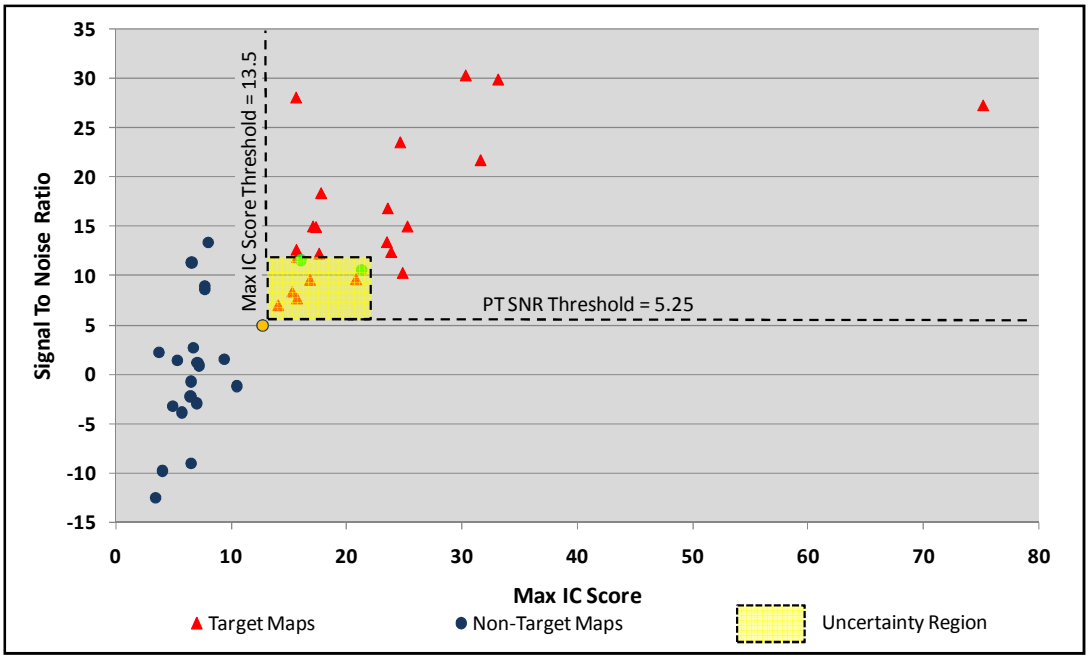


Figure 3-36. AutoGAD-SC Uncertainty Region in Max Score and PT SNR Feature Space (ARES 1D, 1F, 2D, and 2F)

Notice that just as with AutoGAD, the uncertainty region produced for AutoGAD-SC includes two non-target maps, shown in green. These two maps are both products of ARES 1D and as seen in Figure 3-37, both maps produce relatively high kurtosis values (41.26 and 39.42). However both maps produce somewhat larger PTF values (0.0402 and 0.0377) than might be expected for manmade objects in the field of view. Given that the largest PTF value produced by any target map from the four tested images is 0.0196, the author selected 0.035 as a PTF threshold. One non-target map, identified by the orange marker in Figures 3-36 and 3-37, falls inside this region. This particular map would be eliminated based on both its maximum IC score and its PT SNR values. So by retaining any map with kurtosis greater than 10, a PTF value less than 0.035, a PT SNR value greater than 5.25, and a maximum IC score greater than 13.5, all target maps are isolated from all non-target maps.

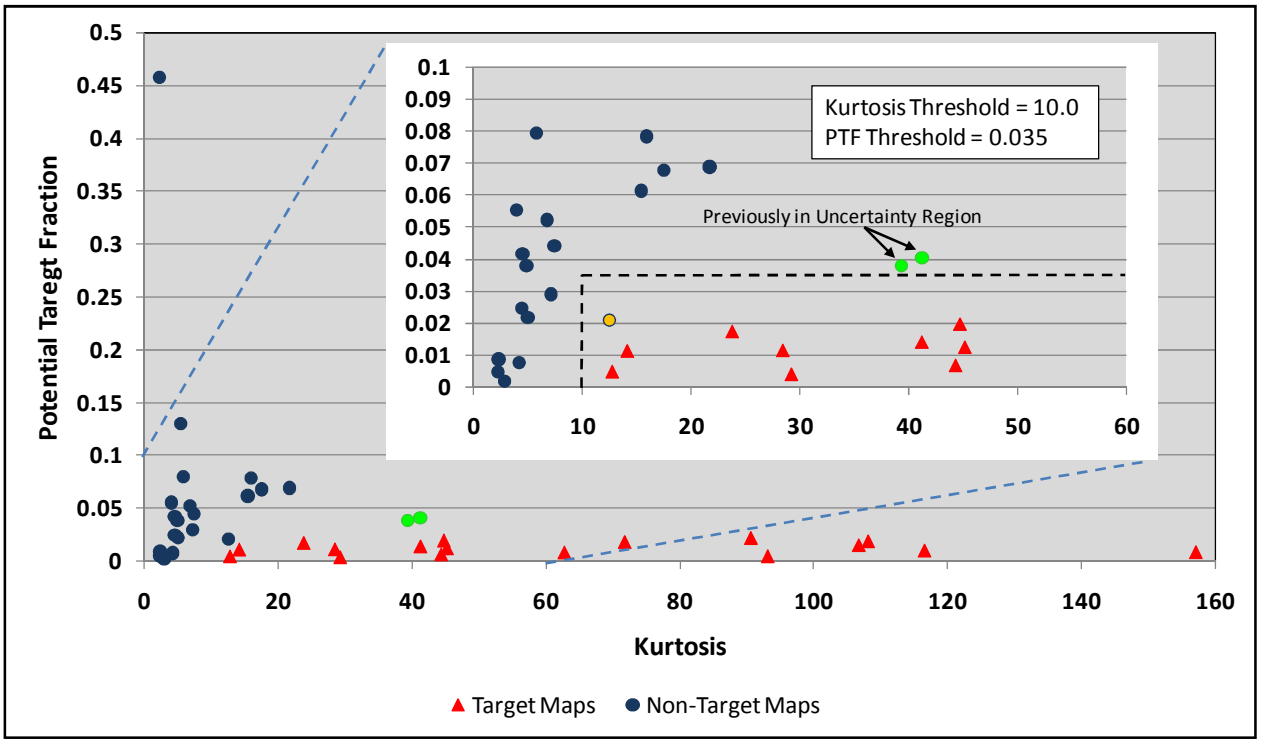


Figure 3-37. AutoGAD-SC Uncertainty Region in PTF and Kurtosis Feature Space (ARES 1D, 1F, 2D, and 2F)

3.6. Left Partial Kurtosis (LPK)

The independent components produced by Fast ICA are asymmetrically distributed about a mean of zero. By convention each distribution is oriented so that the right or positive direction tail is longer than the left. This tends to place the majority of the target pixels in the right tail of signal histogram, however target pixels may also fall in the left tail of a signal histogram and appear as dark returns in the image maps. Figure 3-38 displays the maps produced by ARES 2D. In the adjacent recolored images notice how bright returns appear distinctly different from the surrounding background, but that there are also apparent targets which do not correspond to bright returns in the grey scale map. These targets correspond to darker than background pixels on the grey scale map, and fall in the left hand tail of the signal histogram (Figure 3-39).

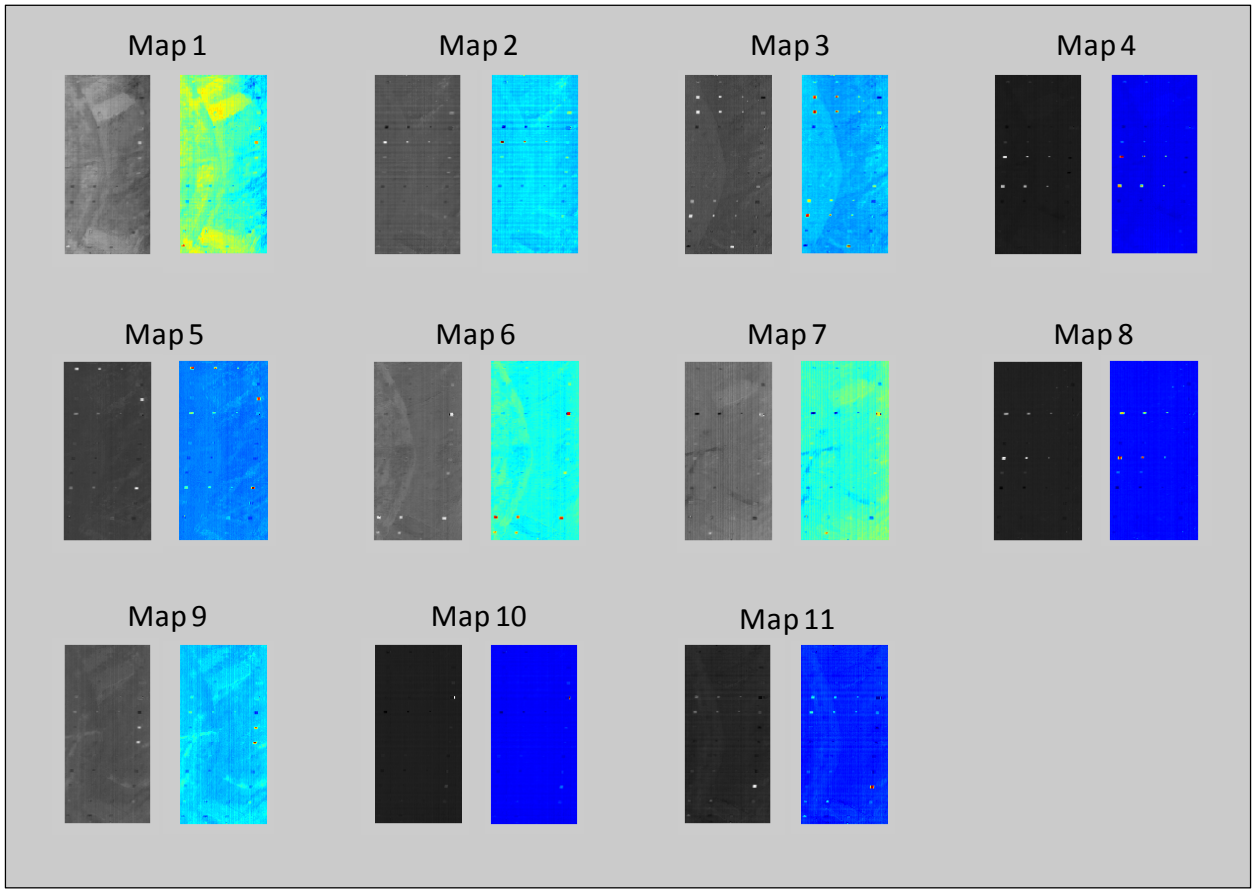


Figure 3-38. ARES 2D Component Maps and False Color Images

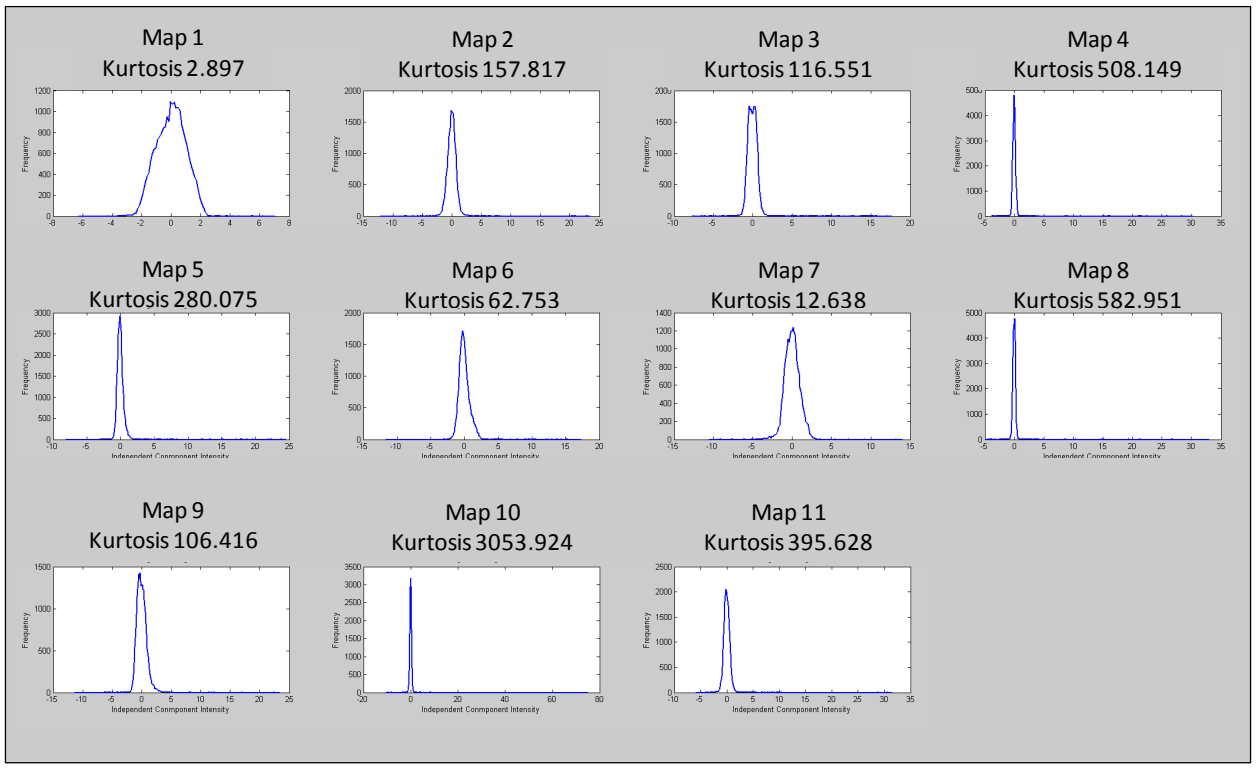


Figure 3-39. ARES 2D Signal Histograms showing outliers in Left and Right Tails

Kurtosis measures the peakedness and tail weight of a distribution. Since this detection algorithm operates by locating outliers in the tails of independent signals, high kurtosis values are a strong indicator that target pixels are likely to be present. For ease of code implementation AutoGAD-SC utilizes MatLAB's built in definition of kurtosis (equation 2.16):

$$\text{kurt}\left(x\right) = \frac{E\left\{x - \mu^4\right\}}{\sigma^4} \tag{0.32}$$

As an indicator however, kurtosis does not provide information relating to the weight of the left tail versus the right tail. Skew provides information regarding the direction in which the heavier tail might be found, but does not provide adequate detail to determine whether or not one tail contributes more to kurtosis than the other. Figure 3-39 provides the kurtosis values for the independent components of ARES 2D. Note that although Map 8 has a kurtosis value more than

five times that of Map 9, the left hand tail of Map 9 is roughly twice the length of the left tail in Map 8. Upon further inspection of these two maps with their associated signal histograms, the significance of this observation becomes apparent. Figure 3-40 shows that map 9 contains two targets which lie in the extreme right tail of the distribution, but contains as many as 15 distinguishable targets found in the left tail.

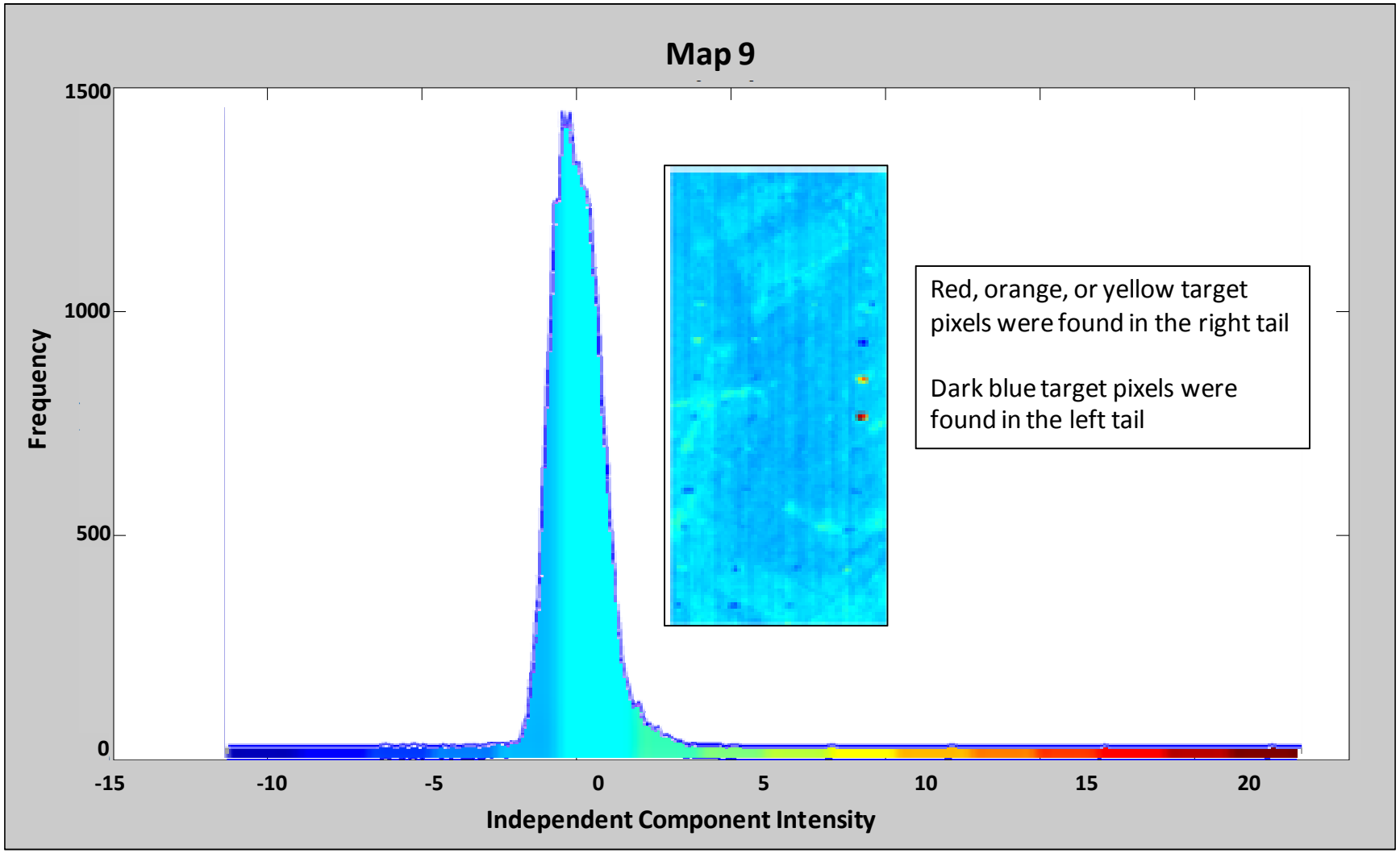


Figure 3-40. ARES 2D Map 9: False Color Map and Signal Histogram

Figure 3-41 shows the same depiction for map 8. Notice that although this map contains target pixels found in the left tail of the distribution, there are fewer of them (7 targets versus 15 in Map 9), and they are less distinguishable from background pixels. By measuring the kurtosis of only the left tail perhaps a determination can be made on whether or not to seek target pixels in both tails of the distribution or only in the right hand side

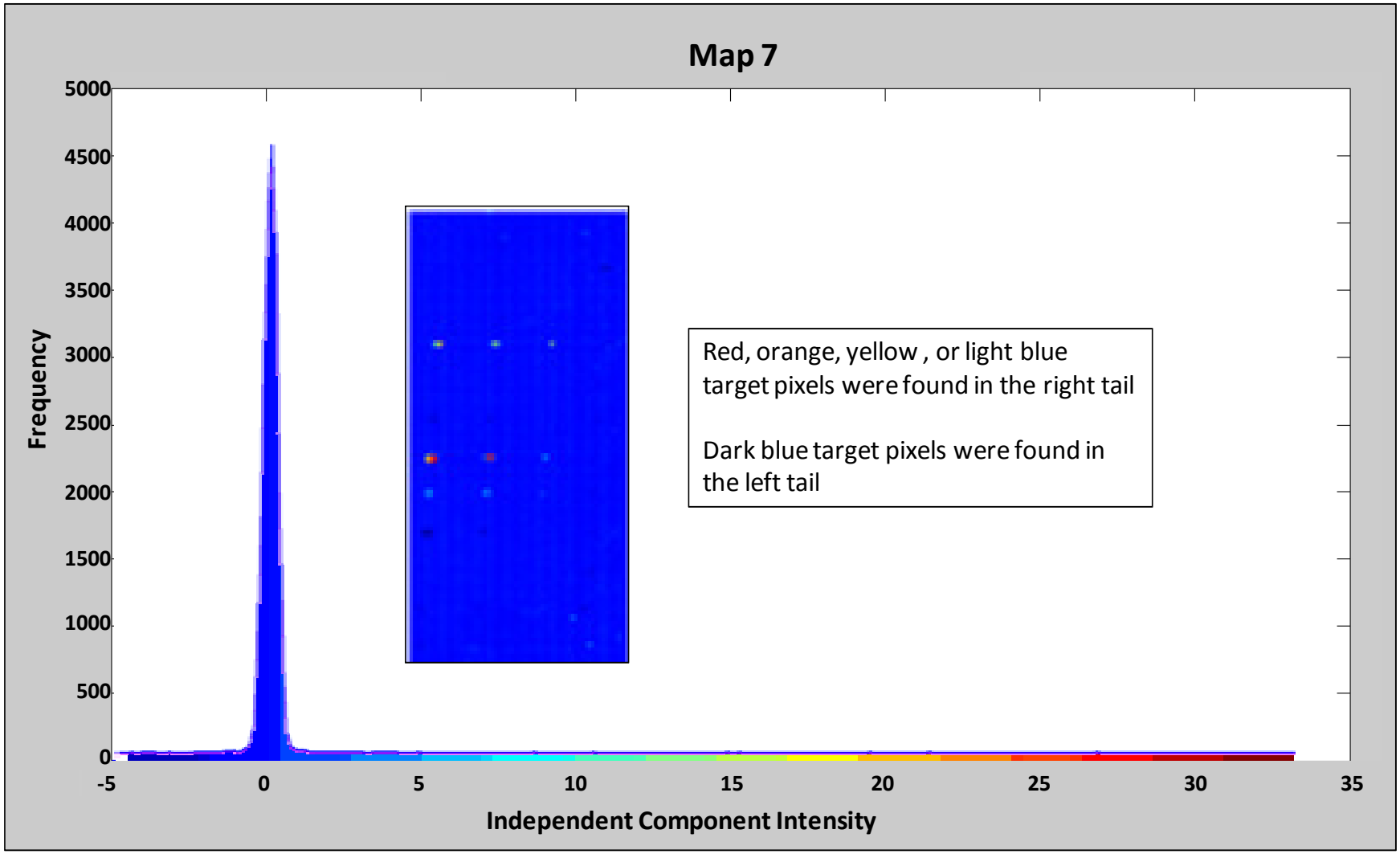


Figure 3-41. ARES 2D Map 8: False Color Map and Signal Histogram

Another way of defining LPK is derived from the fourth moment used to develop the equation for kurtosis.

$$E\{x-\mu\}^4 = \int_{-\infty}^{\infty}(x-\mu)^4 f(x)dx \tag{0.33}$$

For any distribution, total kurtosis can be separated at its mean into left and right partial kurtosis values as shown below.

$$\int_{-\infty}^{\infty}(x-\mu)^4 f(x)dx = \int_{-\infty}^{\mu}(x-\mu)^4 f(x)dx + \int_{\mu}^{\infty}(x-\mu)^4 f(x)dx \tag{0.34}$$

And since each IC has been centered as a part of ICA, the mean μ, is known to be 0, thus each tail produces a separate contribution to kurtosis by:

Left Tail: $$\int_{-\infty}^{\mu}(x-\mu)^4 f(x)dx = \int_{-\infty}^{0}(x-0)^4 f(x)dx = \int_{-\infty}^{0}x^4 f(x)dx \tag{0.35}$$

Right Tail: $$\int_{\mu}^{\infty}(x-\mu)^4 f(x)dx = \int_{\mu}^{\infty}(x-0)^4 f(x)dx = \int_{0}^{\infty}x^4 f(x)dx \tag{0.36}$$

Equations 3.6 and 3.7 indicate that the left partial kurtosis can be found simply by basing the calculation only on those observations which are less than the mean value (or greater than the mean for right partial kurtosis). This measure of relative contribution to overall kurtosis provides some indication of the tendency for a given tail to contain outliers.

Previous work allowed the user to determine whether or not a lower threshold for background pixel was to be estimated, allowing pixels lower than the threshold to be identified as target pixels in the left tail of the distribution. However addition of the MDSL technique for estimating the threshold between target and background pixels generated a tighter threshold than the zero bin method, as described in section 3.4.2. This tighter threshold between left tail outliers and background pixels produced a tendency for false positive pixels to be detected in signals with short left tails. To mitigate this tendency the switch dictating whether or not to

generate a lower threshold was replaced by a switch based on the contribution to kurtosis produced by the left tail of each target signal histogram.

Recall from sections 2.3.3 and 2.3.4 that the process of whitening and ICA leaves each component with a mean of zero and a variance of one. The enables a rather simple approach for quantifying the contribution of the left tail to the overall distribution's kurtosis. Left partial kurtosis was calculated by splitting each distribution into two halves at the mean value of zero, and then directly measuring the kurtosis of those points with IC scores less than zero. This measurement cannot be considered a true kurtosis value but does provide some information regarding the contribution towards overall kurtosis produced by the values less than the mean. Left partial kurtosis was then compared to a user defined parameter, left partial kurtosis threshold (LPKT), to determine whether or not the presence of target pixels in the left tail of the distribution was expected. If the measured left partial kurtosis exceeded the LPKT, a threshold would be established to identify outlier pixels to the left of the distribution.

Table 3-1 shows the overall kurtosis values for all target maps produced by ARES 1D, 1F, 2D, and 2F along with their associated left partial kurtosis values. Lines shaded in dark grey represent target maps in which target pixels in the left tail were clearly identifiable from background. Lines shaded light grey represent maps in which a lower threshold produce neither additional target pixels nor significant false positive pixels. Unshaded lines represent maps which contained primarily background pixels in the left tail of the signal distribution making false positive detection likely. Given that the kurtosis threshold used to discriminate between target and non-target map was set at 10, and that all target maps with clearly identifiable target pixels in the left tail have LPK values greater than 10, the LPK threshold was also set to 10.

Image	Abundance Map	Overall Kurtosis Value	Left Partial Kurtosis (LPK)
ARES 2D	Map 10	3053.8891	187.5735
ARES 2D	Map 8	582.9253	109.1019
ARES 2D	Map 6	508.1359	80.8883
ARES 2D	Map 1	158.6498	75.4261
ARES 2D	Map 5	280.0673	64.1115
ARES 2D	Map 3	116.5282	56.9521
ARES 1F	Map 4	44.336	51.139
ARES 2D	Map 4	105.9545	48.0245
ARES 2D	Map 9	62.7229	46.6658
ARES 2D	Map 7	12.8814	36.5863
ARES 2D	Map 2	395.1442	32.8436
ARES 2F	Map 6	29.2578	19.703
ARES 2F	Map 2	45.2127	17.95
ARES 1F	Map 5	41.235	10.345
ARES 1F	Map 12	71.745	9.271
ARES 2F	Map 7	90.5685	8.3446
ARES 2F	Map 3	93.1217	7.0766
ARES 2F	Map 5	23.8729	6.7918
ARES 1D	Map 8	14.167	5.1712
ARES 1F	Map 8	44.741	5.023
ARES 1F	Map 1	108.121	4.895
ARES 2F	Map 9	28.3315	4.212

Table 3-1. Overall Kurtosis and Left Partial Kurtosis for ARES 1D, ARES 1F, ARES 2D, and ARES 2F

Figures 3-42 and 3-43 show the impact of making a dynamic decision as to whether or not thresholding for outlier target pixels ought to be conducted both above and below the background. Each of the target maps is shown along with its associated signal histogram. Notice that those distributions with short left tails tend to have few if any pixels below the main band of background pixels, and that no lower threshold is produced.

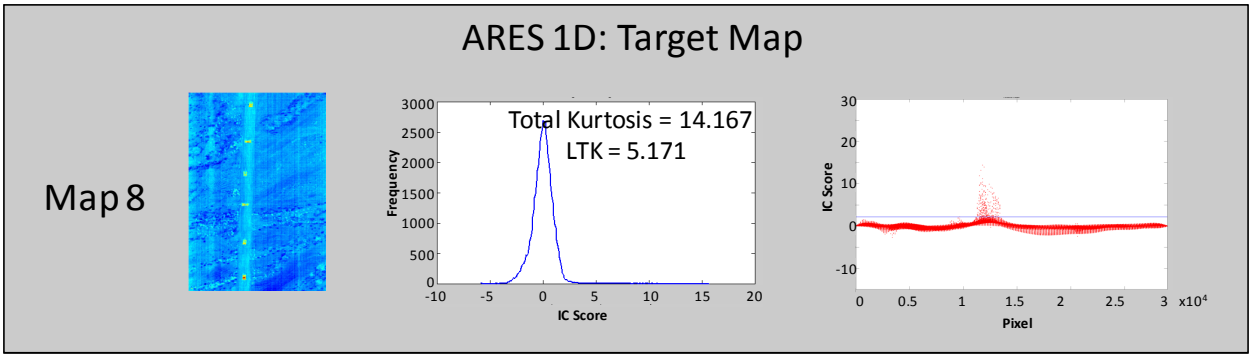


Figure 3-42. ARES 1D Map 8: False Color Map, Signal Histogram, and Single Side Target Pixel Threshold

Notice how the left tail of the signal histogram has a larger shoulder than the right tail. This portion of the left tail corresponds to the darker returns produced by road, rock, and sagebrush features. The sloped transition between peak and tail on the left side of the distribution is likely to contain a large number of non-target pixels recognized as outliers should the MDSL technique be applied to identify the threshold between signal and background.

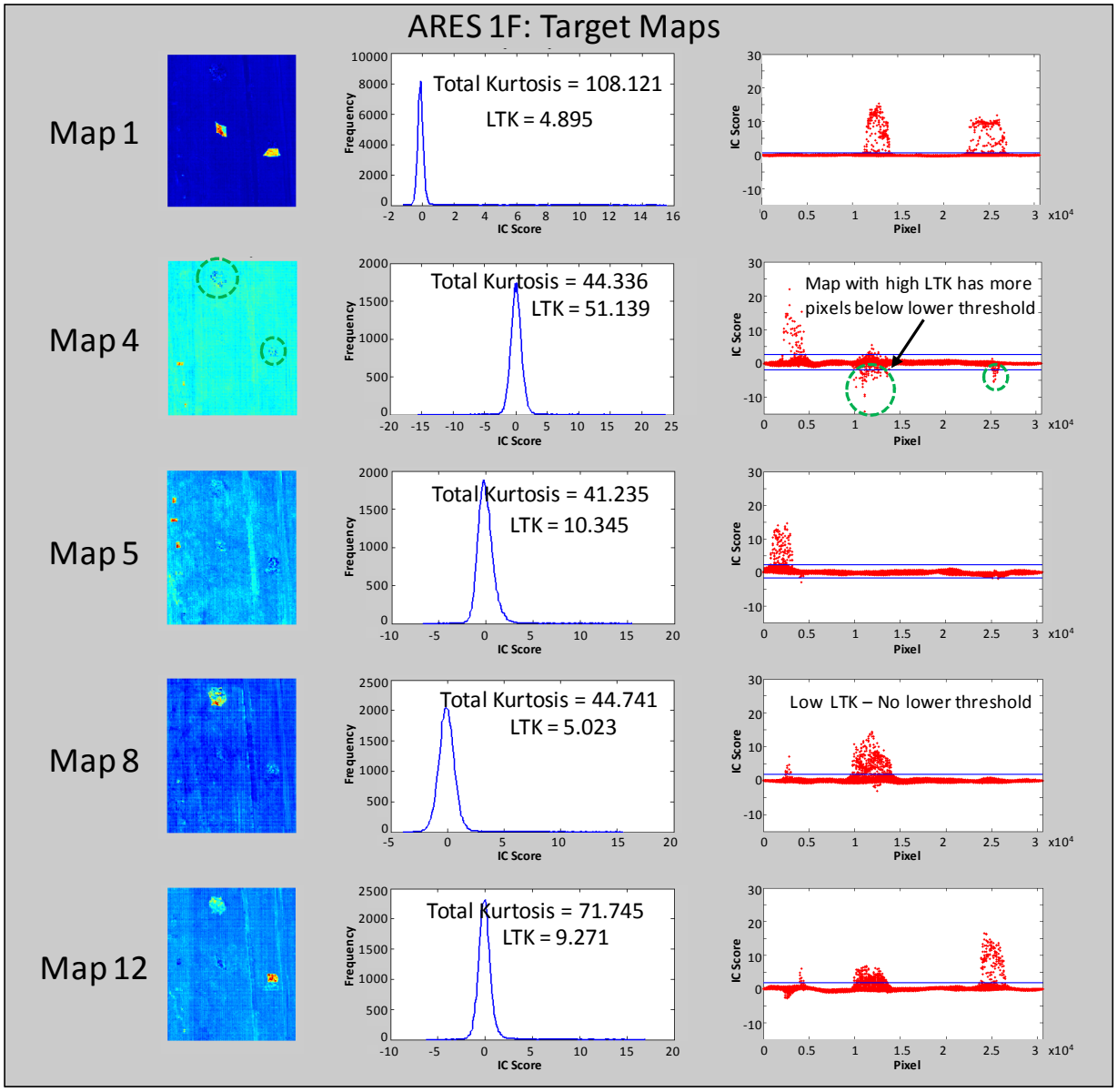


Figure 3-43. ARES 1F: False Color Maps, Signal Histograms, and Single or Two Sided Target Pixel Thresholding Based on LPK

Figure 3-43 provides examples of both one and two sided thresholding based on the left partial kurtosis. Notice how in Map 4 the LPK of 51.139 actually exceeds the overall kurtosis value of 44.336. When a lower threshold is applied to this particular map we see a large number of target pixels detected in the left tail (circled in green). Map 8 provides the alternative case, where although the full signal has a of 44.741, the left tail provides little contribution, and

produces a LPK value of only 5.023. If a lower threshold had been applied, few if any target pixels would have been detected.

3.7. Adaptive Iterative Noise Filtering

Previous work [Johnson, 2008:163] adopted an Iterative Adaptive Noise (IAN) Filter or Wiener filter to reduce false positive detections as a result of noise occurring within background regions of the image. IAN filtration accomplishes this by more heavily smoothing regions of the image where the local variation is similar to the full image variation and applying less smoothing to regions in which the variance is significantly higher than that of the full image. Johnson adopts a canned MATLAB algorithm known as 'wiener2' to accomplish this portion of the AutoGAD algorithm. This algorithm smoothes each target map by observing each pixel's component score in relation to the component scores of those pixels within a smoothing window. For each pixel the mean score and variance within the window are calculated by

$$\begin{aligned} \mu &= \frac{1}{NM} \sum_{n_1, n_2 \in \eta} a^2(n_1, n_2) \\ \sigma^2 &= \left[\frac{1}{NM} \sum_{n_1, n_2 \in \eta} a^2(n_1, n_2) \right] - \mu^2 \end{aligned} \tag{0.37}$$

where

$N =$ length of the smoothing window
$M =$ width of smoothing window
$a(n_1, n_2) \equiv$ IC score of the pixel at location (n_1, n_2) in the neighborhood η
$\mu \equiv$ mean pixel score in the neighborhood η
$\sigma^2 \equiv$ variance of pixel scores in the neighborhood η

Each pixels score is replaced with a filtered score based upon its current value, $a(n_1,n_2)$, the local variance, σ^2, and the total component variance, v^2, by equation 3.5 below.

$$b(n_1,n_2) = \mu + \frac{\sigma^2 - v^2}{\sigma^2}\left[a(n_1,n_2) - \mu\right] \tag{0.38}$$

where

$b(n_1,n_2) \equiv$ new pixel score
$v^2 \equiv$ overall system (component) variance

Two decisions are critical in producing adequate signal smoothing of background noise, and in turn reducing the likelihood of false positive pixel detection. The first of these is window size. Large windows will tend to produce a variance estimate closer to that of the overall system, which then applies more smoothing to the indexed pixel, $a(n_1,n_2)$. If the window is too large, excessive smoothing can occur to target pixels, making them indistinct from background pixels. As Johnson points out, a smaller window can prevent this occurrence, but then repeated iterations of the smoothing algorithm are required to effectively smooth background pixels given the small window size [2008:166]. In this work as with Johnsons a 3x3 pixel window was chosen for all testing. This leads to the second decision, the number of iterations of the smoothing algorithm to perform on each target component. Given that reduction in overall run time is a goal of this thesis, the number of smoothing iterations applied to each target map is critical since each iteration comes with an associated expense in terms of time.

AutoGAD applied a two level approach to solving this problem. Three user defined parameters were established to exercise control over the number of smoothing iterations. Two parameters define the number of iterations of smoothing to be applied, a more smoothing is applied to maps with low signal to noise ratios, and less smoothing repetitions to maps with high SNR values. The third parameter establishes the threshold low SNR and high SNR values. In

Johnson's work a 3x3 pixel window was utilized with a the high number of smoothing iterations set to 100, the low number of smoothing iterations set to 20, and the SNR threshold set to 10 dB. Thus any map with a PT SNR value of less than 10 dB would be cycled through 100 iterations of the IAN filter, while any map with a PT SNR value greater than 10 dB would receive only 20 iterations of IAN filtering.

A test of AutoGAD was conducted to determine how much time was spent performing the IAN filtration portion of the algorithm. The four images, ARES 1D, ARES 1F, ARES 2D, and ARES 2F, were tested over 100 iterations and PT SNR values along with time required for IAN filtering were recorded. Following the test the number of iterations performed in each case was determined. Table 3-2 shows the average number of smoothing iterations and the associated time requirement to each of the target maps found by AutoGAD in the four images. Notice that for some maps the number of iterations applied is exactly 20 or 100. This occurred in cases where the map in question consistently produced the same PT SNR value and was consistently identified as a target map. In those cases where the average number of filtrations was something other than 20 or 100, the stochastic nature of ICA resulted in variations in the PT SNR value of the map, or the map was inconsistently identified as a target map based on its Maximum IC score.

Image	Map	Number of repetitions identified as target containing	Average PT SNR when retained	Averge Number of IAN Filtering Iterations when retained	Average Time for IAN Filtering when retained (sec)	Variance of Time for IAN Filtering
ARES 1D	1	100	8.0284	100.0	1.3521	0.0178
ARES 1D	3	100	4.5569	100.0	1.3559	0.0213
ARES 1F	1	100	26.2662	20.0	0.1522	0.0006
ARES 1F	2	100	14.4335	20.0	0.1493	0.0004
ARES 1F	3	100	10.9542	20.0	0.1537	0.0005
ARES 1F	6	100	7.1157	100.0	0.7090	0.0044
ARES 2D	1	100	25.7392	25.4	0.1100	0.0001
ARES 2D	2	100	34.8119	24.8	0.1092	0.0001
ARES 2D	3	100	28.2084	29.4	0.1106	0.0002
ARES 2D	4	100	20.9599	23.8	0.1111	0.0002
ARES 2D	5	100	22.7185	25.0	0.1118	0.0002
ARES 2D	6	100	18.3578	21.8	0.1115	0.0002
ARES 2D	7	100	17.1696	22.6	0.1130	0.0002
ARES 2D	8	100	15.5405	26.4	0.1117	0.0002
ARES 2D	9	100	12.9649	22.6	0.1112	0.0002
ARES 2D	11	100	10.8019	22.0	0.1463	0.0138
ARES 2F	1	100	21.9327	20.0	0.2350	0.0003
ARES 2F	2	100	21.7296	20.0	0.2376	0.0005
ARES 2F	3	100	20.9701	20.0	0.2368	0.0003
ARES 2F	4	100	18.8668	20.0	0.2386	0.0004
ARES 2F	5	100	18.3510	20.0	0.2385	0.0005
ARES 2F	6	100	15.9259	28.8	0.3386	0.0837
ARES 2F	7	89	18.5978	20.0	0.2369	0.0003
ARES 2F	8	100	17.1906	28.8	0.3376	0.0833
ARES 2F	9	89	18.6327	20.0	0.2385	0.0004
ARES 2F	10	89	17.1234	20.0	0.2352	0.0002
ARES 2F	11	89	16.9719	20.0	0.2357	0.0003
ARES 2F	12	88	12.7377	42.7	0.4953	0.1708
ARES 2F	13	63	9.4635	92.4	1.0706	0.0757
ARES 2F	14	31	9.3301	100.0	1.1522	0.0006
ARES 2F	15	2	7.5029	100.0	1.1419	0.0000
ARES 2F	16	27	5.0579	100.0	1.1507	0.0006
ARES 2F	17	3	5.7525	100.0	1.1546	0.0002

Table 3-2. Iterations and Time Required to perform IAN Filtration on ARES 1D, ARES 1F, ARES 2D, and ARES 2F Target Maps

Based on the results it can be seen that for ARES 1D the two identified target maps require 100 iterations of IAN filtration, based on their PT SNR values of 8.028 and 4.557 respectively. This results in a time penalty of 2.708 seconds. Given that over the same 100 test repetitions the average time required for the AutoGAD algorithm to process ARES 1D was 6.0789 seconds with a variance of 0.7156 seconds, the contribution due to IAN filtration is in excess of 44.55% of the entire algorithm run time. Table 3-3 show the same results for all four of the tested images

Image	Average Algorithm Run Time	Variance of Algorithm Run Time	Average Total Contibution from IAN Filtering (sec)	Variance	% Contribution due to IAN Filtration
ARES 1D	6.0789	0.7156	2.7080	0.0722	44.55%
ARES 1F	4.3668	0.5246	1.1642	0.0098	26.66%
ARES 2D	3.0842	0.0315	1.1465	0.0149	37.17%
ARES 2F	35.6002	72.1013	4.5408	0.8878	12.75%

Table 3-3. Contribution to AutoGAD run time due to IAN Filtration on ARES 1D, ARES 1F, ARES 2D, and ARES 2F Target Maps

By enabling only two levels of signal filtration, all target maps meeting the criteria for one of the two filtration options are treated identically, regardless of their actual signal to noise value. For example two maps from ARES 1D are recognized by AutoGAD as target containing, Map 1 and Map 3 in Table 3-2. Map 3 had an average SNR value of 4.5569 and although Map 1 had an average SNR value of 8.0284, nearly twice that of Map 3, both were treated with 100 iterations of IAN filtration. Likewise Maps 1 and 3 from ARES 1F are both treated with 20 iterations of IAN filtration despite the fact that Map 1 has a SNR value nearly three times that of Map 3. Figure 3-44 compares these two maps as produced by AutoGAD along with their associated abundance plots. Map 1 has well separated target pixels and very low background variation, both of which produce a higher SNR. Because of this fewer than 20 iterations of IAN filtration are required. Map 3 has a somewhat more noisy background, as indicated by more color variation in the false color map, and the wider background band in the abundance plot. Although the target pixels are fairly well separated from background, this image benefits from 20 iterations of filtration, and since only two levels of filtration are possible Map 1 must also be exposed to 20 iterations of filtration.

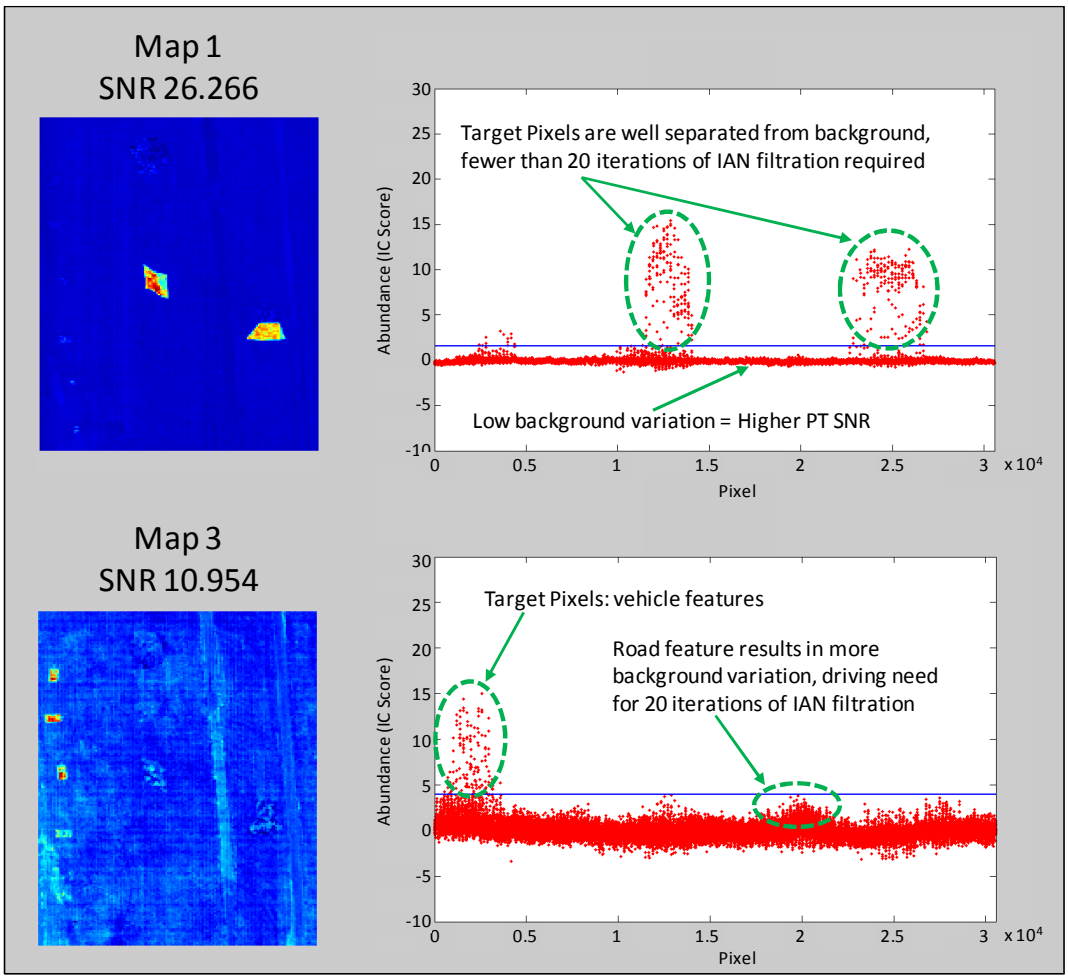


Figure 3-44. ARES 1F Maps 1 and 3, both treated with 20 iterations of IAN filtration

By comparing the impact of reducing the number of iterations to five on these two maps, it becomes apparent that 20 iterations of filtration on Map 1 is excessive, while on Map 3 it is appropriate. In the upper half of Figure 3-45 Maps 1 and 3 are shown with the resulting target pixel detection map from AutoGAD. Pixels circled in green were detected from Map 3 and pixels circled in orange were detected from Map 1. Notice how when the number of IAN filtration iterations is reduced from 20 to 5, neither the Map 1 image nor the target pixels produced by Map 1 change substantially. However when Map 3 is filtered for only 5 iterations, the size of the targets detected by it reduces slightly and more importantly the band of false

positive pixels at the lower left of the image increases, indicating that additional IAN filtration iterations are required.

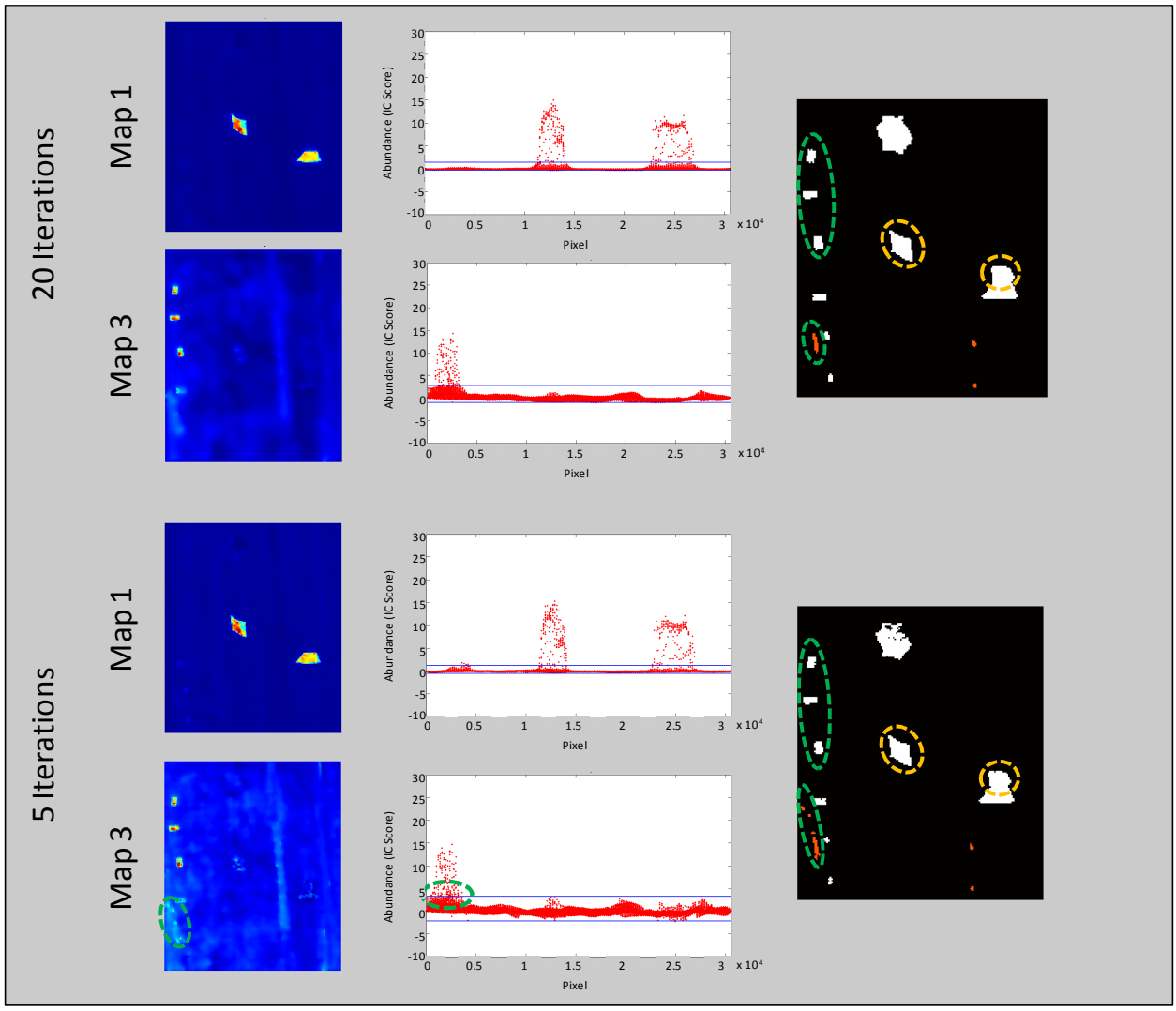


Figure 3-45. ARES 1F Maps 1 and 3,
Comparison of 20 versus 5 iterations of IAN filtration

This work replaces the two level filtration technique with a variable number of IAN filtration iterations based on a simple ratio between a target map's PT SNR value and the PT SNR threshold at which a map is accepted as likely to contain target pixels. The number of IAN filtration iterations performed on each map is then:

$$Iter_i = \left\lfloor C \frac{T_{PT\ SNR}}{SNR_i} \right\rceil \tag{0.39}$$

where

$C \equiv$ A user defined iteration coeffient
$T_{SNR} \equiv$ The PT SNR threshold for identification of target maps
$SNR_i \equiv$ The PT SNR value for map i
$\lfloor\ \rceil \equiv$ Indicates round to the nearest whole number

Equation 3.6 then acts as a rheostat controlling the number of iterations of IAN filtration based on the PT SNR value specific to each map. When the coefficient C is set to a value of 50, the number of iterations can be simultaneously reduced, and remain adequate for each map based on its PT SNR value. For example, both Map 1 and 3 from ARES 1F were treated with 20 iterations of filtration by AutoGAD. Under this new procedure, the PT SNR values for Map 1 and Map 3 would lead to 10 and 24 iterations respectively.

Map 1:

$$Iter_1 = \left\lfloor 50 \frac{5.25}{26.2662} \right\rceil = \lfloor 9.993 \rceil = 10 \text{ iterations} \tag{0.40}$$

Map 3:

$$Iter_3 = \left\lfloor 50 \frac{5.25}{10.9542} \right\rceil = \lfloor 23.9634 \rceil = 24 \text{ iterations} \tag{0.41}$$

A second test measuring the number of IAN filtration iterations performed and time required was performed, this time using AutoGAD-SC with the number of iterations performed calculated by equation 3.6. Once again 100 repetitions each of ARES 1D, 1F, 2D, and 2F were performed. Table 3-4 shows the results of this experiment.

Image	Map	Number of repetitions identified as target containing	Average PT SNR when retained	Averge Number of IAN Filtering Iterations when retained	Average Time for IAN Filtering when retained (sec)	Variance of Time for IAN Filtering
ARES 1D	3	100	7.6614	41.2	0.5879	0.0001
ARES 1F	1	100	28.0870	11.0	0.0795	0.0001
ARES 1F	2	100	14.9829	21.0	0.1508	0.0001
ARES 1F	3	100	12.6025	25.0	0.1793	0.0001
ARES 1F	4	100	12.3709	25.0	0.1780	0.0000
ARES 1F	5	100	11.8689	26.9	0.1918	0.0001
ARES 2D	1	100	27.2974	12.0	0.0627	0.0000
ARES 2D	2	100	29.9440	10.9	0.0561	0.0000
ARES 2D	3	100	30.3097	10.0	0.0524	0.0000
ARES 2D	4	100	21.6397	15.0	0.0789	0.0000
ARES 2D	5	100	23.5278	13.0	0.0678	0.0000
ARES 2D	6	100	16.7014	19.0	0.0984	0.0000
ARES 2D	7	100	18.3687	17.0	0.0886	0.0000
ARES 2D	8	100	13.5429	23.6	0.1224	0.0000
ARES 2D	9	100	14.9777	21.0	0.1086	0.0000
ARES 2D	10	98	7.3193	43.2	0.2220	0.0012
ARES 2F	1	100	13.3655	24.9	0.2894	0.0059
ARES 2F	2	100	10.9310	29.2	0.3423	0.0020
ARES 2F	3	100	10.9200	29.9	0.3456	0.0040
ARES 2F	4	100	11.1197	29.6	0.3457	0.0071
ARES 2F	5	100	10.7639	29.8	0.3467	0.0026
ARES 2F	6	99	9.3257	34.6	0.3975	0.0045
ARES 2F	7	1	9.2003	34.0	0.3845	0.0000
ARES 2F	8	25	9.3988	33.7	0.3999	0.0031
ARES 2F	9	1	6.9870	45.0	0.5093	0.0000

Table 3-4. Iterations and Time Required to perform IAN Filtration on ARES 1D, ARES 1F, ARES 2D, and ARES 2F Target Maps (Iterations determined as function of PT SNR value)

Image	Average Algorithm Run Time (sec)	Variance	Contibution from IAN Filtering (sec)	Variance	% Contribution due to IAN Filtration
ARES 1D	4.4347	0.3491	0.5879	0.0722	13.26%
ARES 1F	4.3765	0.2695	0.7795	0.0098	17.81%
ARES 2D	2.8725	0.0310	0.9534	0.0149	33.19%
ARES 2F	14.1303	3.1215	2.1722	0.8878	15.37%

Table 3-5. Contribution to AutoGAD-SC run time due to IAN Filtration on ARES 1D, ARES 1F, ARES 2D, and ARES 2F Target Maps

Comparison of Table 3-2 to 3-4 reveals two improvements in terms of algorithm run time, the number of retained target maps upon which filtration must be applied, and the number of iterations of filtration performed on each map. The tables indicate fewer maps were retained as target containing for ARES 1D, 2D, and 2F when the AutoGAD-SC algorithm was used, than

when original AutoGAD was employed. For example, during the 100 AutoGAD test repetitions, 11% of trials produced 7 target maps, 55% produced 13 target maps, 32% produced 14 target maps, and 2% produced 15 maps. Yet when AutoGAD-SC was tested using the same image 74% of the trials produced 6 target maps, while the other 26% produced 7 target maps. In other words an average of 12.7 target maps were filtered when AutoGAD was used versus only 6.3 maps when using AutoGAD-SC algorithm. Thus the overall time spent on noise filtration of target maps is reduced by simply reducing the number of target maps.

The second mechanism reducing the time required to filter noise from target maps was reduction of the number of iterations applied based on each maps SNR value. In the 100 repetitions applying AutoGAD to ARES 1D (table 3-2), exactly 400 target maps were detected and filtered. Of these maps 300 met the requirements to be treated with only 20 iterations of filtration, the remaining 100 were treated with 100 iterations. This amounts to an average of 40 filtration iterations for each identified target map. Each of the 300 target maps receiving 20 iterations of filtration required an average of 0.152 ± 0.016 seconds for filtration at the 95% confidence level, while the 100 maps receiving 100 iterations needed an average of 0.709 ±0.051 seconds of filtration time. As shown in Table 3-3 this amounted to a total of 1.1642 seconds of filtration time on average for ARES 2D.

When AutoGAD-SC was applied to the same test using the same image, 500 target maps were found, one more per iteration. It might be expected that by retaining 25% more maps the time required to perform IAN filtration would increase. However the addition of a mechanism which tailors the amount of filtration specific to each map's SNR value, the average number of iterations applied to a target map was reduced from 40 to 21.786, which drove a corresponding reduction in total time required to conduct IAN filtration from 1.1642 seconds to 0.7795 seconds.

In the case of ARES 1F, reducing the number of filtration iterations does not overcome the fact that one additional target map must be manipulated so the overall algorithm run time is not significantly changed. However the contribution to total run time due to IAN filtration was reduced from 26.66% when using AutoGAD to 17.81% when using AutoGAD-SC.

3.8. Target Pixel Detection

Following IAN filtration target pixels are identified using the previously established threshold between signal and background IC scores. Previous work required formation of a histogram from the filtered target map signals, followed by identification of a new signal-background threshold using the first zero bin technique. Recall that the MDSL technique described in section 3.4.2 identifies the actual "knee in the histogram" separating background pixels from outlier pixels. Because the MDSL technique more closely approximates the true threshold between signal and noise, it was possible eliminate the steps measuring a new threshold on the reduced noise signals.

3.9 AutoGAD-SC Process Overview

Figure 3-45 compares the original AutoGAD algorithm to AutoGAD-SC. Several significant changes in the original were made in a attempt to improve target pixel detection, reduce false positive pixel detection, and improve algorithm run time. These modifications include:

1. Rapid dimensionality estimation/reduction by averaging highly correlated adjacent bands
2. Application of Scott's Rule for optimum bin width estimation, eliminating user defined parameter

3. Application of MDSL technique to estimate the bound between outlier pixels (targets) and background pixels
4. Addition of Kurtosis and Potential Target Fraction (PTF) to discriminate between target and non-target maps
5. Using each target map PT SNR value to determine the number of iterations of IAN filtration to perform
6. Addition of Left Partial Kurtosis (LPK) as a measure of the tail independent contribution to kurtosis and application of LPK to determine which maps are likely to contain target pixels in both tails of their signal histogram

A complete listing of AutoGAD-SC, spectral correlation clustering, and FastICA code can be found in Appendix A.

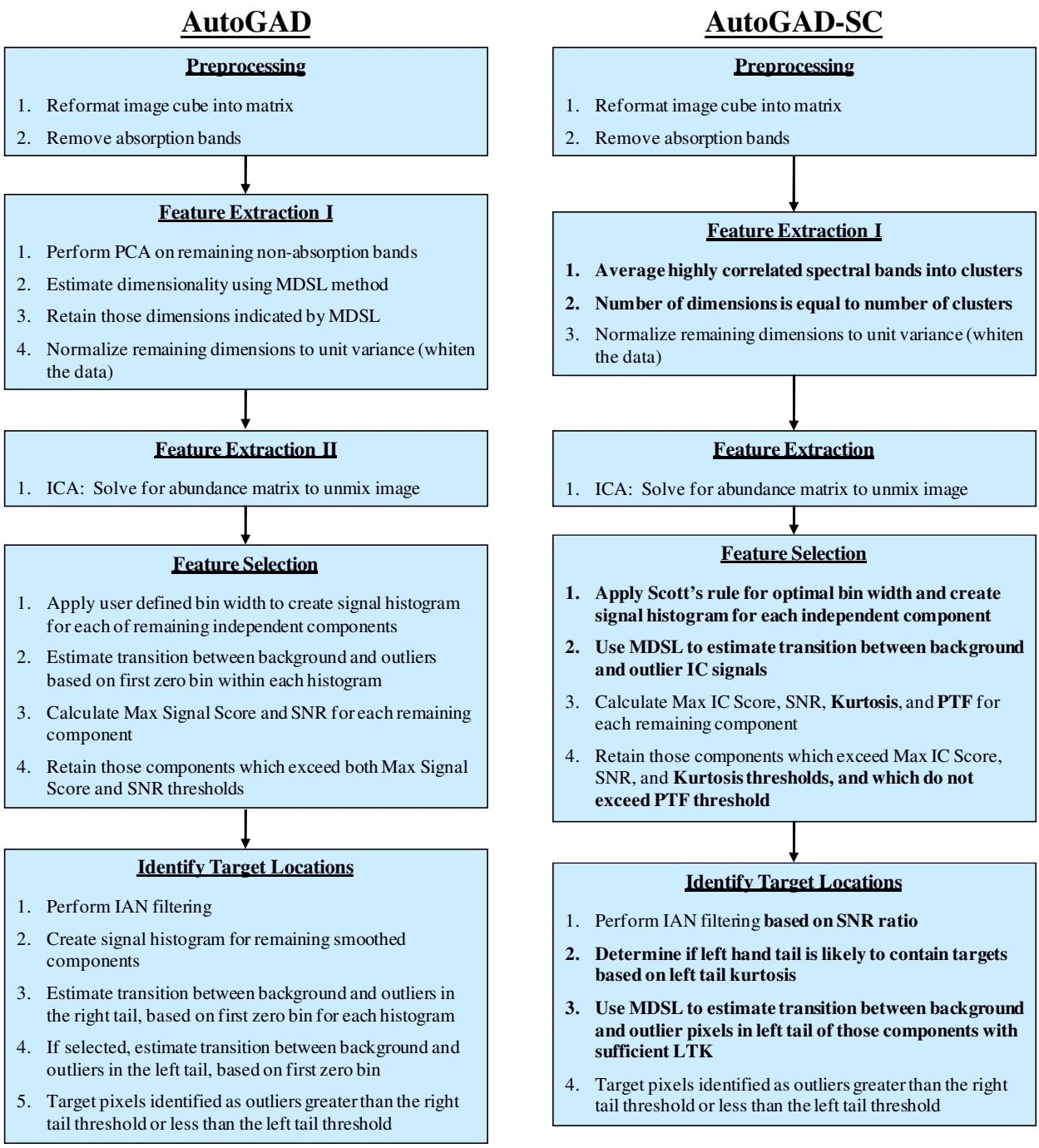


Figure 3-46. Process Comparison AutoGAD vs. AutoGAD-SC

3.10. Robust Parameter Design for New Algorithm

Thus far in development of AutoGAD-SC parameter settings were established partially based off settings recommended by Johnson for AutoGAD, partially from experimentation conducted in Sections 3.5 and 3.6, and partially by trial and error. These settings are listed in table 3-6 below.

Parameter Name	Setting
Required Correlation Threshold	0.985
Potential Target Fraction Threshold	3.50%
Maximum IC Score Threshold	13.5
Kurtosis Threshold	10
PT SNR Threshold	5.25
Left Partial Kurtosis Threshold	10
IAN Filtration Iterations Coefficient	50

Table 3-6. Original Parameter Settings Chosen for AutoGAD-SC

Four parameter settings available for adjustment in AutoGAD-SC are not included in the list above as they were held to a single setting throughout development and testing. The first two of these four parameters represent settings used internally by the FastICA algorithm, function and orthogonalization. The function switch specifies which of two measures of non-gaussianity are applied within the objective function by the algorithm. Tests conducted by Johnson [2008:139] indicated that of the two options, the pow3 setting produced the less variation in the results. The orthogonalization switch establishes whether ICs will be located in parallel (symmetric) or one by one (deflationary). The symmetric setting was held throughout development and testing of this algorithm based on tests conducted by Koo [2007:45]. The signal smoothing switch was set to "on", based on Johnson's demonstration that IAN filtration reduced the FPF. Finally, the window size was held constant to the same setting employed by Johnson, 3 pixels, partially to provide consistency when AutoGAD-SC results are compared to AutoGAD, and partially to simplify testing by utilizing only continuous control variables.

The main issue in establishing parameter settings for an algorithm such as AutoGAD-SC is that it must simultaneously maximize response, while minimizing variability when provided various images as inputs. For this reason Taguchi's Crossed Array Design using Signal-to-Noise Ratios as a measure of variance was utilized in an effort to isolate those parameters which maximize response, while simultaneously reducing variance.

3.10.1. Taguchi's Crossed Array Design

The Taguchi approach to the RPD problem includes the use of an orthogonal experimental design in which the control variables (from Table 3-6) occupy an inner array, while the noise variables (the images included in the test) occupy the outer array [Myers, 2002:539]. The experiment utilized consisted of a 3^7 full factorial inner array and an outer array consisting of the same four sample images, ARES 1D, 1F, 2D, and 2F. Table 3-7 lists the levels at which each control variable was tested.

Parameter Name	Low	Mid	High
Required Correlation Threshold	0.981	0.985	0.989
Potential Target Fraction Threshold	2.50%	3.50%	4.50%
Maximum IC Score Threshold	12.5	13.5	14.5
Kurtosis Threshold	9	10	11
PT SNR Threshold	4.5	5.25	6.5
Left Partial Kurtosis Threshold	9	10	11
IAN Filtration Iterations Coefficient	40	50	60

Table 3-7. Experimental Design Factor Levels for 3^7 Inner Array

Three responses variables were measured at each of the 2187 possible parameter setting combinations for each of the four images, true positive fraction, false positive fraction, and algorithm run time. Each of the recorded observations was then standardized to remove dimension issues when making comparisons between responses. Marginal mean values were then calculated for each factor at each level by averaging all responses at that factor level. For

example the marginal mean value of TPF when the correlation threshold is set to 0.981 is found by averaging all TPF values produced at a correlation threshold setting of 0.981. These response specific values were then plotted in an attempt to isolate the factors with the greatest influence over response. Figures 3-47(a-c) show the marginal means plots for the three responses, when correlation threshold is isolated. A complete set of marginal means curves can be found in Appendix B.

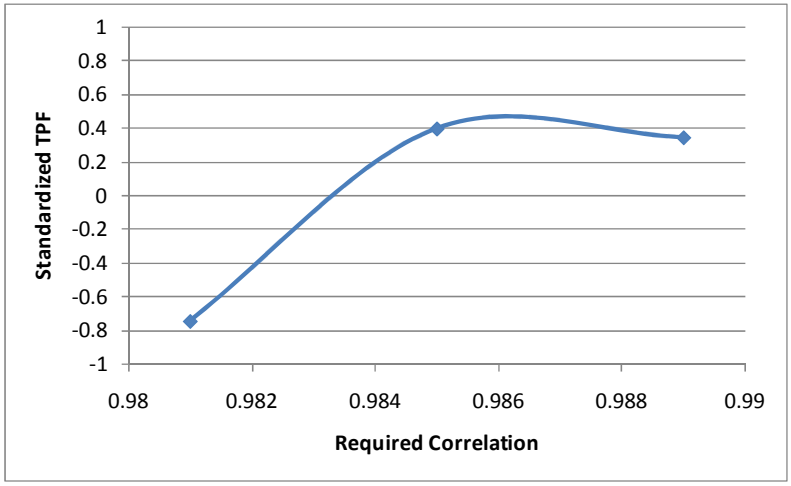


Figure 3-47(a). Marginal Mean Plot: Standardized TPF vs. Correlation Threshold

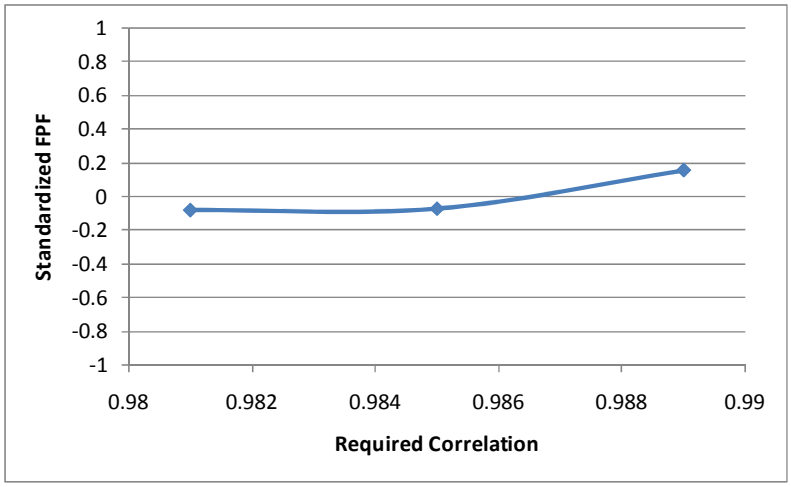


Figure 3-47(b). Marginal Mean Plot: Standardized FPF vs. Correlation Threshold

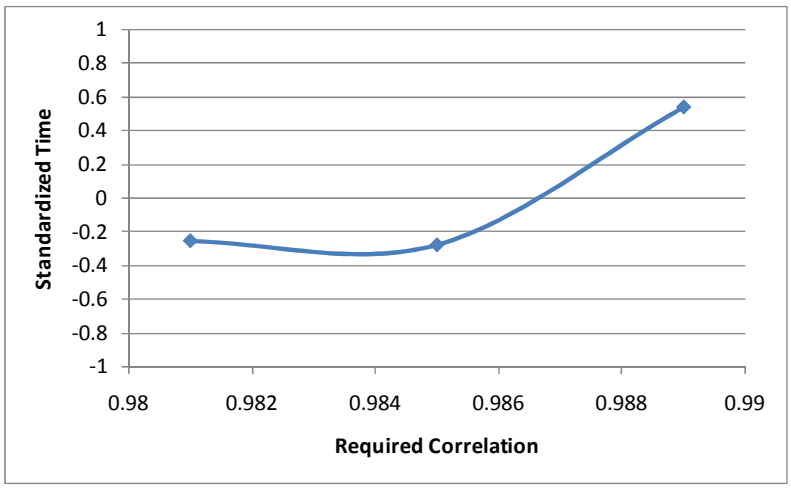


Figure 3-47(c). Marginal Mean Plot: Standardized Time vs. Correlation Threshold

In Figure 3-47(a) higher values of standardized TPF indicate correlation threshold settings which identified more true positive pixels on average. Likewise, in both Figures 3-37 (b) and (c) low values of standardized FPF and time indicate settings which on average resulted in fewer false positives or ran in less time, respectively. Correlation threshold then appears to bear some influence on all three responses and is a good candidate for control over response. In addition, choice of setting for correlation threshold is relatively simplified by the fact that to achieve a better than average TPF either 0.985 or 0.989 must be set (Figure 3-47(a)), while in order to achieve less than average run time either 0.981 or 0.985 must be set (Figure 3-47(c)). So the only setting which produces both a high TPF response and low time response is a correlation threshold of 0.985.

Taguchi suggests several summary statistics known as signal-to-noise ratios as mechanisms for accounting for both process mean and variance. The measure of SNR used in this thesis is given by the equation.

$$\text{SNR}=10\cdot\log_{10}\left(\frac{\bar{y}^2}{s^2}\right) \qquad (0.42)$$

where

$\overline{y} \equiv$ mean response for a specific parameter setting

$s^2 \equiv$ variance of response for a specific parameter setting

By equation 3.13 an SNR value is produced for each possible combination of control variables, based on the mean response and standard deviation across the tested noise variables (in this experiment noise included the four images). SNR provides a mechanism to gauge response against variation in the response. Large responses with low variation produce higher SNR values; small response values with high variation produce low SNR values, thus high SNR values are generally desirable as they represent settings which produce low variation. Marginal SNR values were then calculated by averaging the SNR response with a single control variable held constant at one of its tested levels, exactly as marginal mean values were calculated above. Plots of SNR versus the PTF factor level settings are shown in Figures 3-48(a-c).

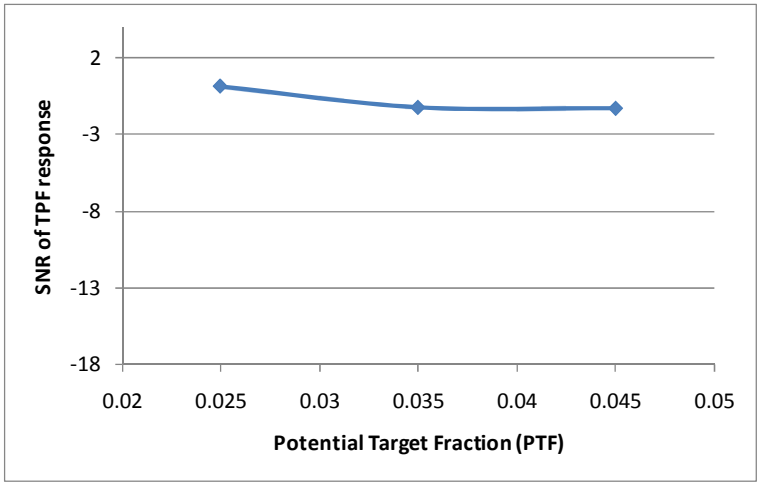


Figure 3-48(a). Marginal SNR Plot: SNR of TPF vs. Potential Target Fraction

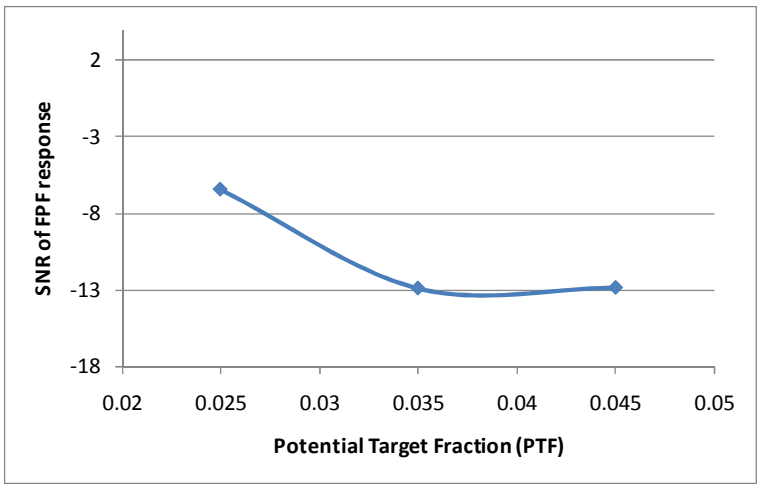


Figure 3-48(b). Marginal SNR Plot: SNR of FPF vs. Potential Target Fraction

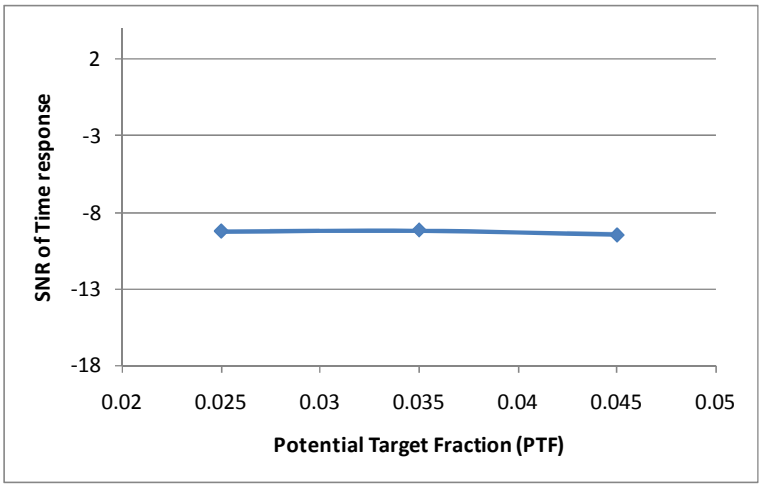


Figure 3-48(c). Marginal SNR Plot: SNR of Time vs. Potential Target Fraction

Figures 3-48(a) and (c) provide little indication that one PTF factor setting results in any less response variation than any other setting, but Figure 3-48(b) clearly demonstrates a higher SNR response when PTF is set to its low setting (0.025). Given that PTF produces little impact on either TPF or Time response and has a positive effect on the FPF response (see Appendix A), a PTF setting of 0.025 is appropriate.

With seven parameters and three response variables, a total of 21 marginal mean plots and 21 marginal SNR plots were produced. Simultaneously optimizing three responses while minimizing variance by inspecting 42 plots proved to be somewhat cumbersome, so in an

attempt to reduce the complexity, the three marginal mean responses and SNR values were combined into pair of values based on the formulas below:

$$\overline{y}_{\text{tot}} = \overline{y}_{\text{TPF}} - \overline{y}_{\text{FPF}} - \overline{y}_{\text{Time}} \tag{0.43}$$

$$SNR_{\text{tot}} = SNR_{\text{TPF}} + SNR_{\text{FPF}} + SNR_{\text{Time}} \tag{0.44}$$

where

$\overline{y} \equiv$ partial mean response taken at a single combination of control settings across noise variables

$SNR \equiv SNR$ values specific to a single combination of control settings across noise variables

This simple combination of responses was possible because all original response data was standardized as part of the analysis. By combining the three sets of partial mean responses and the three sets of partial SNR responses into two groups, determination of the Taguchi derived parameter settings was greatly simplified. Figures 3-49(a) through (g) show the marginal mean and marginal SNR responses. In both sets of graphs below, higher values represent desirable results. In the left column of graphs high marginal mean values indicate a combination of high TPF, with low FPF and low time required. In the right column high marginal SNR values indicate low high response values coupled with low variation of the responses.

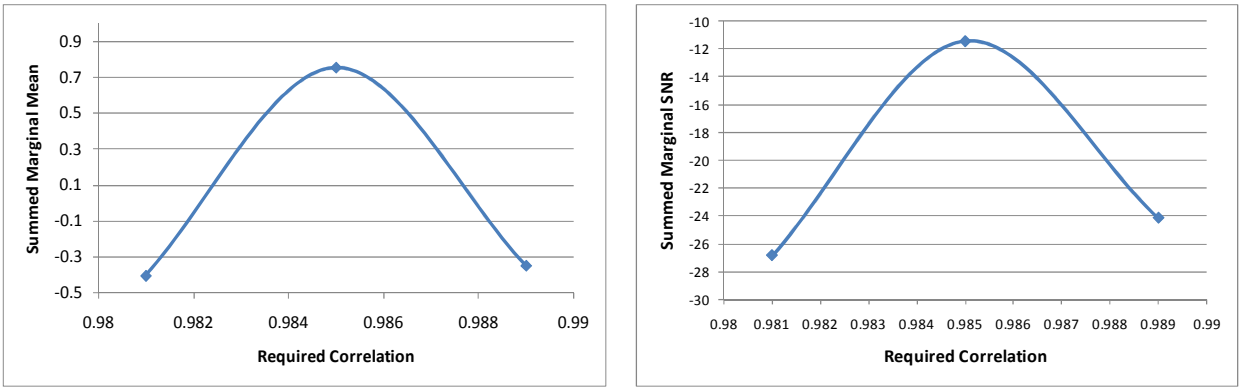


Figure 3-49(a). Summed Marginal Means (left) & Summed SNR (right) vs. Correlation Threshold

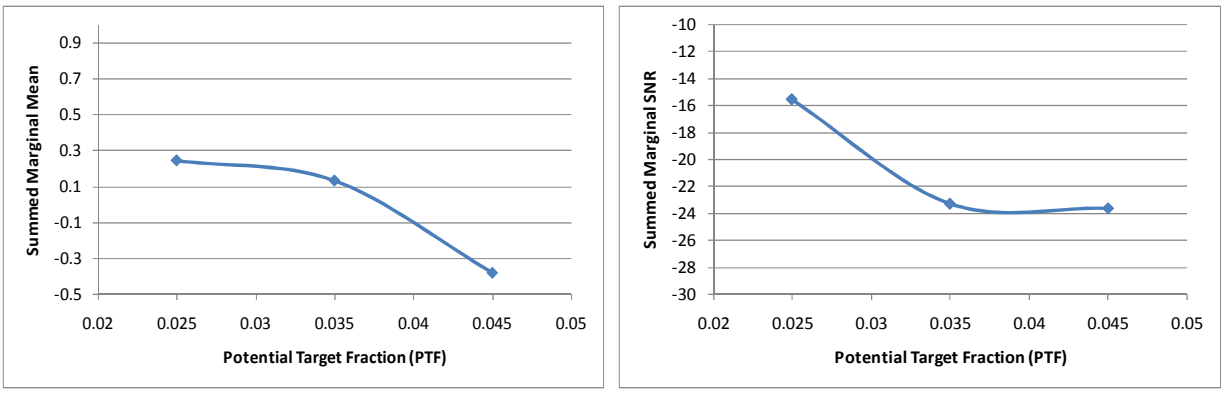


Figure 3-49(b). Summed Marginal Means (left) & Summed SNR (right) vs. Potential Target Fraction Threshold

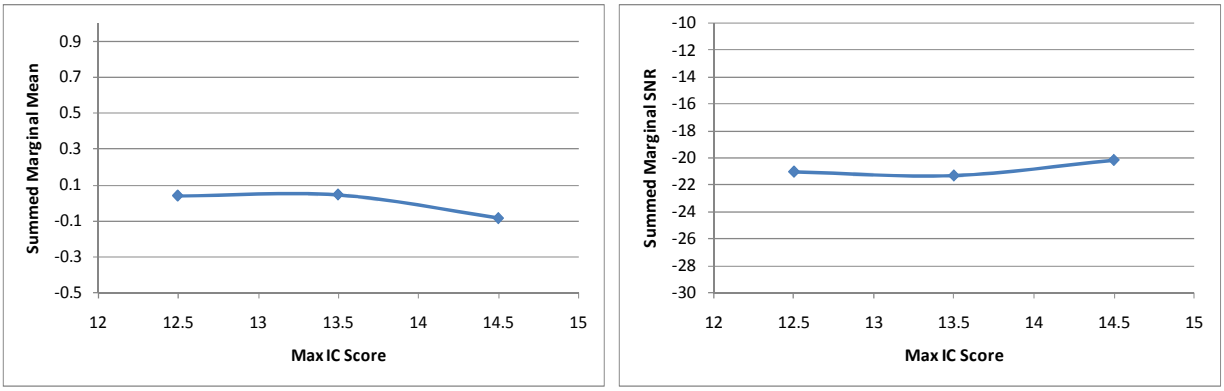


Figure 3-49(c). Summed Marginal Means (left) & Summed SNR (right) vs. Max IC Score Threshold

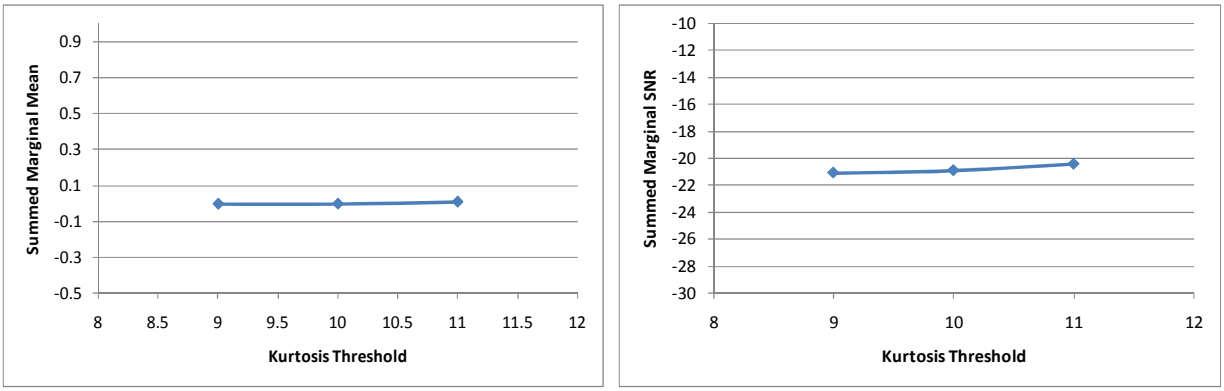


Figure 3-49(d). Summed Marginal Means (left) & Summed SNR (right) vs. Kurtosis Threshold

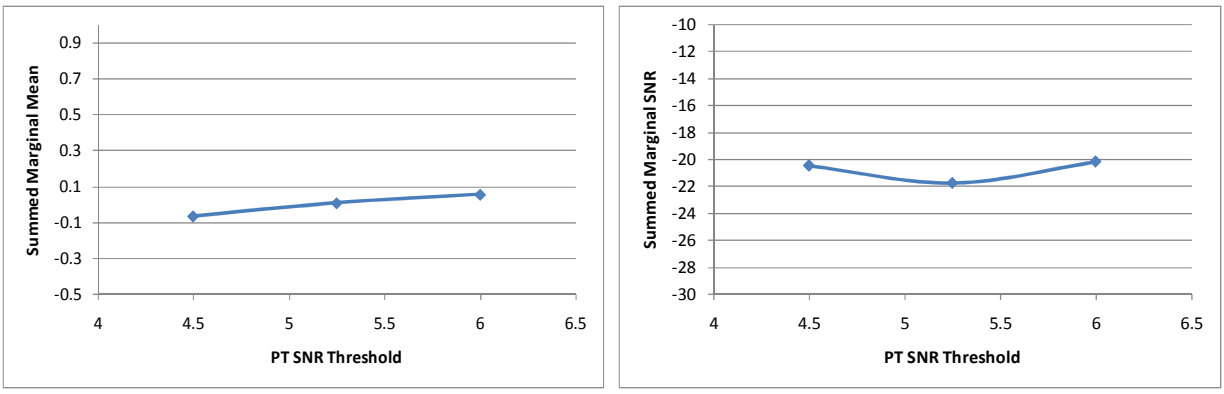


Figure 3-49(e). Summed Marginal Means (left) & Summed SNR (right) vs. PT SNR Threshold

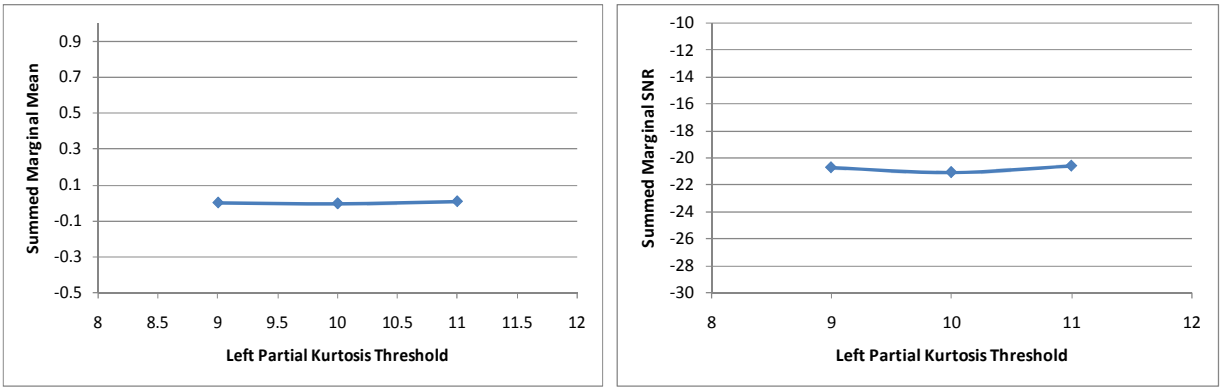


Figure 3-49(f). Summed Marginal Means (left) & Summed SNR (right) vs. Left Partial Kurtosis Threshold

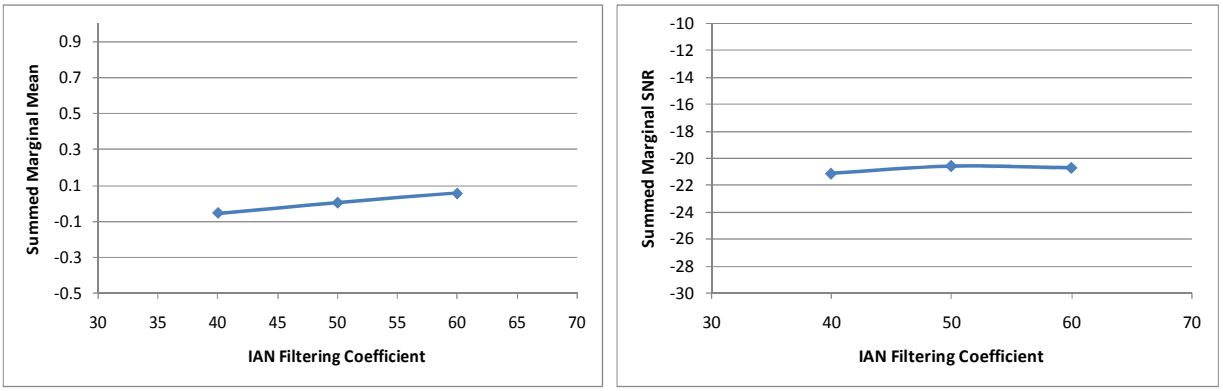


Figure 3-49(g). Summed Marginal Means (left) & Summed SNR (right) vs. IAN Filtering Coefficient

Inspection of Figure 3-37(a) through (g) yielded Taguchi style optimized parameter settings. Table 3-8 lists the selected settings used for validation in Chapter 4.

Parameter Name	Optimal for Response	Optimal for Variance	Selected Setting
Required Correlation Threshold	0.985	0.985	0.985
Potential Target Fraction Threshold	2.50%	2.50%	2.50%
Maximum IC Score Threshold	12.5 or 13.5	14.5	12.5
Kurtosis Threshold	Any	11	11
PT SNR Threshold	6.0	4.5 or 6.0	6.0
Left Partial Kurtosis Threshold	Any	9 or 11	11
IAN Filtration Iterations Coefficient	60	50 or 60	60

Table 3-8. Optimal AutoGAD-SC settings based on Taguchi Crossed Array Experiment

The Taguchi crossed array design is generally accepted as one of the more easily implemented robust parameter design methods, but two valid criticisms of the technique exist. First, the crossed array design fails to account for interaction between factors when identifying optimal factor level settings. While typically a first order model may suffice, the complexity of AutoGAD-SC with respect to its parameter-response interactions generated some concern as to whether or not a first order model would sufficiently describe the interaction between parameter settings and the actual responses. The second criticism is that application of the Taguchi methodology can only produce optimal responses at tested factor level settings. In other words, although correlation threshold was tested over the range 0.981 to 0.989, there is no provision for interpolation between the three tested factor levels (0.981, 0.985, and 0.989). This limits the utility of the approach on input parameters of a continuous nature.

3.10.2. Response/Variance Model Optimization

As an alternative to the Taguchi crossed array results, the experimental data produced by the 3^7 factorial design was used to generate predictive models for each of the three responses. The same data was also used to produce noise induced variation models of each of the three responses. Once again four tested images were treated as uncontrollable noise factors. As before the first step in analyzing the results of the experiment was standardization of the

response data. By standardizing upfront, comparisons between responses could be made, as well as comparisons between algorithm response and variation in the responses. The image averages and variances shown in Table 3-9 represent a selection of the 2187 response observations produced for each of the four tested images. These values were produced by calculating the mean and variance of the response from the four tested images given a specific set of factor level settings.

Parameter Settings							Image Averages			Image Variances		
Correlation	PTF	Max IC Score	Kurtosis	PT SNR	LTK	IAN Iteration Coeff	TPF	FPF	Time	TPF	FPF	Time
0.981	0.025	12.5	9	4.5	9	40	-0.784	-0.347	-0.273	2.522	0.588	0.764
0.985	0.025	12.5	9	4.5	9	40	0.428	-0.060	-0.360	0.078	1.179	0.291
0.989	0.025	12.5	9	4.5	9	40	0.373	0.456	0.769	0.075	1.231	2.325
0.981	0.035	12.5	9	4.5	9	40	-0.409	0.024	-0.169	2.723	0.386	1.122
0.985	0.035	12.5	9	4.5	9	40	0.421	-0.157	-0.077	0.086	0.801	0.972
0.989	0.035	12.5	9	4.5	9	40	0.379	0.444	0.726	0.079	1.138	2.398
0.981	0.045	12.5	9	4.5	9	40	-0.412	0.857	-0.227	2.663	2.464	0.846
0.985	0.045	12.5	9	4.5	9	40	0.423	0.842	-0.089	0.082	3.232	0.963
0.989	0.045	12.5	9	4.5	9	40	0.379	0.513	0.721	0.079	1.001	2.357

Table 3-9. Selection of Input Factor Levels with Average Responses and Variance of Responses (for ARES 1D, 1F, 2D, and 2F)

Analysis of Variance (ANOVA) was performed on the six sets of responses (three mean responses and three variance responses). All regression models were produced in JMP using stepwise regression in both directions with entry and exit α values set to 0.25. ANOVA tables for the six models are included in Appendix C. Table 3-10 lists the number of terms included in each of the six models and the adjusted R^2 values. All six models are provided in equations 3.16 through 3.22.

Model	Degrees of Freedom	Adjusted R-Square
TPF Mean Model	19	0.970415
TPF Variance Model	44	0.979905
FPF Mean Model	18	0.810429
FPF Variance Model	19	0.793691
Time Mean Model	15	0.776211
Time Variance Model	22	0.783308

Table 3-10. Model Degrees of Freedom and Adjusted R^2

True Positive Fraction Response Model:

$$
\begin{aligned}
TPF = & -133.0129 + 136.523\,T_C + 4.56\,T_{PTF} - 0.07184\,T_{Max} - 0.0191\,T_{PT\ SNR} \\
& - 0.00181\,T_{Iter} - 1704.989\,(T_C - 0.985)\,(T_{PTF} - 0.035) \\
& + 21.526\,(T_C - 0.985)\,(T_{Max} - 13.5) + 4.242\,(T_C - 0.985)\,(T_{PT\ SNR} - 5.25) \\
& + 0.4173\,(T_C - 0.985)\,(T_{Iter} - 50) - 3.098\,(T_{PTF} - 0.035)\,(T_{Max} - 13.5) \\
& - 0.00584\,(T_{Max} - 13.5)\,(T_{PT\ SNR} - 5.25) - 0.000344\,(T_{Max} - 13.5)\,(T_{Iter} - 50) \\
& + 0.000479\,(T_{PT\ SNR} - 5.25)\,(T_{Iter} - 50) \\
& + 1139.658\,(T_C - 0.985)\,(T_{PTF} - 0.035)\,(T_{Max} - 13.5) \\
& + 0.1258\,(T_C - 0.985)\,(T_{Max} - 13.5)\,(T_{Iter} - 50) \\
& + 37372.56\,(T_C - 0.985)^2 - 329.984\,(T_{PTF} - 0.035)^2 - 0.068\,(T_{Max} - 13.5)^2 \\
& + 0.0000625\,(T_{Iter} - 50)^2
\end{aligned}
\tag{0.45}
$$

Variance of True Positive Response Model:

$$
\begin{aligned}
\mathrm{Var}(\mathrm{TPF}) = {} & 313.996 - 317.891T_c + 5.516T_{PTF} - 0.056T_{Max} - 0.006T_K - 0.003T_{PT\ SNR} \\
& + 0.0036T_{LPK} - 0.00092T_{Iter} - 2102.452\,(T_c - 0.985)\,(T_{PTF} - 0.035) \\
& + 22.628\,(T_c - 0.985)\,(T_{Max} - 13.5) + 0.211\,(T_c - 0.985)\,(T_{PT\ SNR} - 5.25) \\
& + 0.1244\,(T_c - 0.985)\,(T_{Iter} - 50) - 3.785\,(T_{PTF} - 0.035)\,(T_{Max} - 13.5) \\
& + 0.681\,(T_{PTF} - 0.035)\,(T_K - 13.5) + 1.132(T_{PTF} - 0.035)\,(T_{PT\ SNR} - 5.25) \\
& + 0.168\,(T_{PTF} - 0.035)\,(T_{Iter} - 50) - 0.0118(T_{Max} - 13.5)\,(T_K - 10) \\
& + 0.00275\,(T_{Max} - 13.5)\,(T_{PT\ SNR} - 5.25) - 0.00125\,(T_{Max} - 13.5)\,(T_{LPK} - 10) \\
& - 0.00047\,(T_{Max} - 13.5)\,(T_{Iter} - 50) - 0.00356(T_K - 10)\,(T_{PT\ SNR} - 5.25) \\
& - 0.00304\,(T_K - 10)\,(T_{LPK} - 10) - 0.000644(T_K - 10)\,(T_{Iter} - 50) \\
& - 0.00064\,(T_K - 10)\,(T_{Iter} - 50) - 0.00458(T_{PT\ SNR} - 5.25)\,(T_{LPK} - 10) \\
& - 0.00391\,(T_{PT\ SNR} - 5.25)\,(T_{Iter} - 50) + 0.00016\,(T_{LPK} - 10)\,(T_{Iter} - 50) \\
& + 1421.439\,(T_c - 0.985)\,(T_{PTF} - 0.035)\,(T_{Max} - 13.5) \\
& - 651.492\,(T_c - 0.985)\,(T_{PTF} - 0.035)\,(T_{PT\ SNR} - 5.25) \\
& - 61.866\,(T_c - 0.985)\,(T_{PTF} - 0.035)\,(T_{Iter} - 50) \\
& + 0.1969\,(T_c - 0.985)\,(T_{Max} - 13.5)\,(T_{Iter} - 50) \\
& + 1.3667\,(T_c - 0.985)\,(T_{PT\ SNR} - 5.25)\,(T_{Iter} - 50) \\
& + 0.7870\,(T_{PTF} - 0.035)\,(T_{Max} - 13.5)\,(T_K - 10) \\
& - 0.2179\,(T_{PTF} - 0.035)\,(T_{PT\ SNR} - 5.25)\,(T_{Iter} - 50) \\
& - 0.0165\,(T_{Max} - 13.5)\,(T_K - 10)\,(T_{LPK} - 10) \\
& - 0.0017\,(T_{Max} - 13.5)\,(T_{PT\ SNR} - 5.25)\,(T_{Iter} - 50) \\
& + 0.00125\,(T_{Max} - 13.5)\,(T_{LPK} - 10)\,(T_{Iter} - 50) \\
& - 0.01718\,(T_K - 10)\,(T_{PT\ SNR} - 5.25)\,(T_{LPK} - 10) \\
& + 0.00128\,(T_K - 10)\,(T_{PT\ SNR} - 5.25)\,(T_{Iter} - 50) \\
& + 0.00137\,(T_{PT\ SNR} - 5.25)\,(T_{LPK} - 10)\,(T_{Iter} - 50) \\
& + 85.4536\,(T_c - 0.985)\,(T_{PTF} - 0.035)\,(T_{PT\ SNR} - 5.25)\,(T_{Iter} - 50) \\
& + 74495\,(T_c - 0.985)^2 - 576.55\,(T_{PTF} - 0.035)^2 - 0.0405\,(T_{Max} - 13.5)^2 \\
& - 0.0237\,(T_{PT\ SNR} - 5.25)^2 - 0.00021\,(T_{Iter} - 50)^2
\end{aligned}
\tag{0.46}
$$

False Positive Fraction Response Model:

$$
\begin{aligned}
FPF = & -29.784 + 30.175\,T_C + 34.998\,T_{PTF} - 0.0114\,T_{Max} - 0.115\,T_{PT\,SNR} \\
& - 0.0112\,T_{LPK} - 0.0999\,T_{Iter} - 4406.581\,(T_C - 0.985)\,(T_{PTF} - 0.035) \\
& - 1.634\,(T_C - 0.985)\,(T_{Max} - 13.5) - 3.19\,(T_C - 0.985)\,(T_{PT\,SNR} - 5.25) \\
& - 3.572\,(T_C - 0.985)\,(T_{LPK} - 10) - 1.158\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25) \\
& - 0.122\,(T_{PTF} - 0.035)\,(T_{Iter} - 50) + 0.0174\,(T_{Max} - 13.5)\,(T_{PT\,SNR} - 5.25) \\
& + 0.0022\,(T_{PT\,SNR} - 5.25)\,(T_{Iter} - 50) + 7008.688\,(T_C - 0.985)^2 \\
& + 1667.65\,(T_{PTF} - 0.035)^2 + 0.0272\,(T_{PT\,SNR} - 5.25)^2 \\
& + 0.000147\,(T_{Iter} - 50)^2
\end{aligned}
\tag{0.47}
$$

Variance of True Positive Response Model:

$$
\begin{aligned}
Var(FPF) = & \; 31.0116 - 32.965\,T_C + 94.502\,T_{PTF} + 0.00836\,T_{Max} - 0.1012\,T_{PT\,SNR} \\
& - 0.0105\,T_{Iter} - 12503.47\,(T_C - 0.985)\,(T_{PTF} - 0.035) \\
& - 4.585\,(T_C - 0.985)\,(T_{Max} - 13.5) - 13.27\,(T_C - 0.985)\,(T_{PT\,SNR} - 5.25) \\
& - 0.993\,(T_C - 0.985)\,(T_{Iter} - 50) + 2.547\,(T_{PTF} - 0.035)\,(T_{Max} - 13.5) \\
& + 7.305\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25) + 0.556\,(T_{PTF} - 0.035)\,(T_{Iter} - 50) \\
& - 0.0018\,(T_{Max} - 13.5)\,(T_{Iter} - 50) + 0.0077\,(T_{PT\,SNR} - 5.25)\,(T_{Iter} - 50) \\
& - 2091.687\,(T_C - 0.985)\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25) \\
& - 196.296\,(T_C - 0.985)\,(T_{PTF} - 0.035)\,(T_{Iter} - 50) \\
& - 27315.97\,(T_C - 0.985)^2 + 7234.12\,(T_{PTF} - 0.035)^2 \\
& + 0.000435\,(T_{Iter} - 50)^2
\end{aligned}
\tag{0.48}
$$

Time Response Model:

$$\begin{aligned}\text{Time} = & -98.422 + 99.329\,T_C + 0.695\,T_{PTF} + 0.0161\,T_{PT\,SNR} + 0.0062\,T_{LPK} \\ & + 0.0026\,T_{Iter} - 2.297\,(T_C - 0.985)\,(T_{PT\,SNR} - 5.25) \\ & + 2.37\,(T_C - 0.985)\,(T_{LPK} - 10) + 0.290\,(T_C - 0.985)\,(T_{Iter} - 50) \\ & + 0.964\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25) - 0.394\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25) \\ & + 0.0066\,(T_{PT\,SNR} - 5.25)\,(T_{LPK} - 10) - 0.00099\,(T_{PT\,SNR} - 5.25)\,(T_{Iter} - 50) \\ & - 0.60\,(T_C - 0.985)\,(T_{PT\,SNR} - 5.25)\,(T_{Iter} - 50) \\ & + 2.269\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25)\,(T_{LPK} - 10) \\ & + 26279.7\,(T_C - 0.985)^2\end{aligned} \tag{0.49}$$

Variance of Time Response Model:

$$\begin{aligned}\text{Var}(\text{Time}) = & -152.814 + 154.937\,T_C - 1.533\,T_{PTF} + 0.022\,T_{Max} + 0.139\,T_{PT\,SNR} \\ & + 0.0116\,T_{LPK} + 0.0036\,T_{Iter} + 1.846\,(T_C - 0.985)\,(T_{Max} - 13.5) \\ & - 8.005\,(T_C - 0.985)\,(T_{PT\,SNR} - 5.25) + 3.882\,(T_C - 0.985)\,(T_{LPK} - 10) \\ & + 0.444\,(T_C - 0.985)\,(T_{Iter} - 50) + 0.548\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25) \\ & - 0.725\,(T_{PTF} - 0.035)\,(T_{LPK} - 10) + 0.027\,(T_{Max} - 13.5)\,(T_{PT\,SNR} - 5.25) \\ & + 0.002\,(T_{Max} - 13.5)\,(T_{Iter} - 50) + 0.005\,(T_{PT\,SNR} - 5.25)\,(T_{LPK} - 10) \\ & - 0.0011\,(T_{PT\,SNR} - 5.25)\,(T_{Iter} - 50) + 0.002\,(T_{LPK} - 10)\,(T_{Iter} - 50) \\ & + 9.963\,(T_C - 0.985)\,(T_{Max} - 13.5)\,(T_{PT\,SNR} - 5.25) \\ & + 0.482\,(T_C - 0.985)\,(T_{Max} - 13.5)\,(T_{Iter} - 50) \\ & - 1.132\,(T_C - 0.985)\,(T_{PT\,SNR} - 5.25)\,(T_{Iter} - 50) \\ & + 4.237\,(T_{PTF} - 0.035)\,(T_{PT\,SNR} - 5.25)\,(T_{LPK} - 10) \\ & + 62684.7\,(T_C - 0.985)^2\end{aligned} \tag{0.50}$$

where

$T_C \equiv$ Correlation Threshold

$T_{PTF} \equiv$ Potential Target Fraction Threshold

$T_{Max} \equiv$ Max IC Score Threshold

$T_{PT\,SNR} \equiv$ PT SNR Threshold

$T_K \equiv$ Kurtosis Threshold

$T_{LPK} \equiv$ Left Partial Kurtosis Threshold

$T_{Iter} \equiv$ IAN Filtration Iterations Coefficient

A single objective function representing the combined standardized values of the three responses and their variances was produced by the following equation:

$$\max z = \mathrm{TPF} - \mathrm{Var(TPF)} - \mathrm{FPF} - \mathrm{Var(FPF)} - \mathrm{Time} - \mathrm{Var(Time)}$$

s.t.

$$\begin{aligned}
T_C &\geq 0.981 \quad \text{and} \quad T_C \leq 0.989 \\
T_{PTF} &\geq 0.025 \quad \text{and} \quad T_{PTF} \leq 0.045 \\
T_{Max} &\geq 12.5 \quad \text{and} \quad T_{Max} \leq 14.5 \\
T_{PT\ SNR} &\geq 4.5 \quad \text{and} \quad T_{PT\ SNR} \leq 6.0 \\
T_K &\geq 9 \quad \text{and} \quad T_K \leq 11 \\
T_{LPK} &\geq 9 \quad \text{and} \quad T_{LPK} \leq 11 \\
T_{Iter} &\geq 40 \quad \text{and} \quad T_{Iter} \leq 60
\end{aligned} \tag{0.51}$$

Because there is no objective way to determine the relative importance of the three responses and their variances, no coefficients were included in formation of the function. However, coefficients could be assigned based on a user's preference. Premium Solver in Microsoft Excel was then utilized to maximize the objective function. Table 3-11 provides the resulting optimized parameter settings along with the author's original settings and the settings derived from Taguchi's crossed array design. These three sets of parameters were then applied to AutoGAD-SC during performance tests of the new algorithm against AutoGAD using Johnson's recommended settings.

Parameter Name	Original Settings	Crossed Array Settings	Response/Variance Optimization Settings
Required Correlation Threshold	0.985	0.985	0.98514236
Potential Target Fraction Threshold	3.50%	2.50%	2.69%
Maximum IC Score Threshold	13.5	12.5	12.5
Kurtosis Threshold	10	11	9
PT SNR Threshold	5.25	6	6
Left Partial Kurtosis Threshold	10	11	11
IAN Filtration Iterations Coefficient	50	60	60

Table 3-11. AutoGAD-SC Parameter Settings for Validation Testing

Before reporting the results of the performance tests of AutoGAD-SC it is instructive to note the similarities and the differences between the parameter settings provided by the two RPD techniques. Both the crossed array and the Response/Variance optimization approaches arrived at a correlation threshold identical to the one identified by the author as a good original setting. Crossed array could only produce the centerpoint (0.985), while the Response/Variance optimization technique provided a setting of 0.98514. The two techniques chose parameter settings which differed from the author's initial parameter settings for all six remaining parameters. The two techniques indicated the same (or similar) parameter settings for five of the six remaining parameters and selected opposite extremes of tested parameter settings as their recommended settings for only kurtosis threshold. This appears partially due to the fact that kurtosis threshold appears t o have limited affect on the responses in light of its relatively flat marginal means slope. Recall that the Taguchi method accounts only for first order effects, but upon review kurtosis threshold appears several times as an interaction effect in the variance model for TPF, the only model in which T_K appears. For this reason the crossed array design likely mildly understated the impact of setting T_K to its low level.

IV. Results and Analysis

4.1 Comparison of AutoGAD to AutoGAD-SC Dimensionality Estimation

Dimensionality assessment is one of the key determiners of speed and accuracy for hyperspectral blind signal separation. With this in mind a series of four tests were performed comparing the number of dimensions for each of the eight ARES images as estimated by AutoGAD to the number of dimensions as estimated by dynamic spectral clustering. All AutoGAD parameters employed during testing were as recommended by Johnson during his work [2007] (table 4-1).

Parameter Name	Setting
Dimension Adjust	0
Maximum IC Score Threshold	10.0
Bin Width SNR	0.05
PT SNR Threshold	2.0
Bin Width Ident	0.05
Threshold Both Sides	0
Smooth Iterations High	100
Smooth Iterations Low	20
Low SNR	10

Table 4-1. AutoGAD Parameter Settings for Validation Testing

Testing was conducted on a Dell Precision 490 PC equipped with dual Xeon® 2.99 GHz processors and 3.00 GB of RAM and running Microsoft Windows XP Pro. Code execution was accomplished in MATLAB R2007a, with all applications other than the computers operating system and ordinary network activity discontinued for the duration of the test. Testing consisted of 100 timed repetitions of HSI processing and target detection for all eight test images. Four responses were recorded following each repetition; number of dimensions, TPF, FPF, and run time.

Table 4-2 compares the dimensionality assessments produced by AutoGAD-SC to those made by AutoGAD. Recall that AutoGAD employs PCA followed by Johnson's MDSL to make

a dimensionality estimate, while AutoGAD-SC relies on the number of spectral clusters found to be present in the original data cube. Although the difference in the number of dimensions estimated by the two techniques is small, AutoGAD-SC tends to identify fewer endmembers than the original AutoGAD algorithm. Recognizing that the number of dimensions drives the run time of most computationally expensive portion of the algorithm, ICA, it is expected that AutoGAD-SC will tend to run somewhat faster than AutoGAD.

Image	AutoGAD	AutoGAD-SC
ARES 1C	8	10
ARES 1D	8	9
ARES 1F	15	15
ARES 2C	11	10
ARES 2D	13	11
ARES 2F	18	9
ARES 3F	13	9
ARES 4F	14	8

Table 4-2. Number of Dimensions (Endmembers) Estimated by AutoGAD and AutoGAD-SC

4.2 Comparison of Detection Results

Prior to reporting test results, Figures 4-1through 4-8 display a comparisons showing AutoGAD-SC performance alongside the performance of AutoGAD. Recognize that each of these results was based on a single instantiation of the algorithm processing a sample image, and that no assessment of how variant either algorithm's performance was can be made from the results below. The comparisons below are made between AutoGAD (left) and AutoGAD-SC using the author's original parameter settings (right). Between each target pixel detection image, a true color image is depicted for reference.

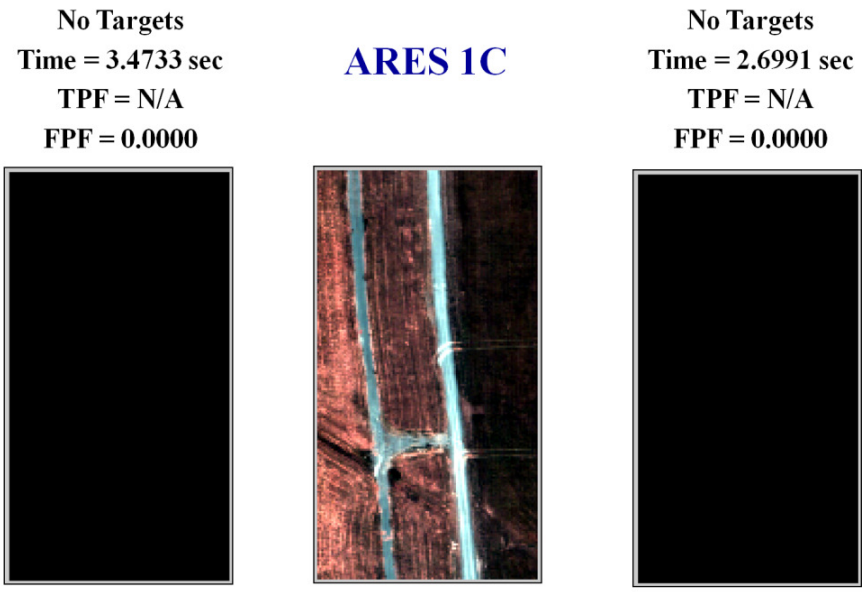


Figure 4-1(a). ARES 1C: AutoGAD (left) vs. AutoGAD-SC (right)

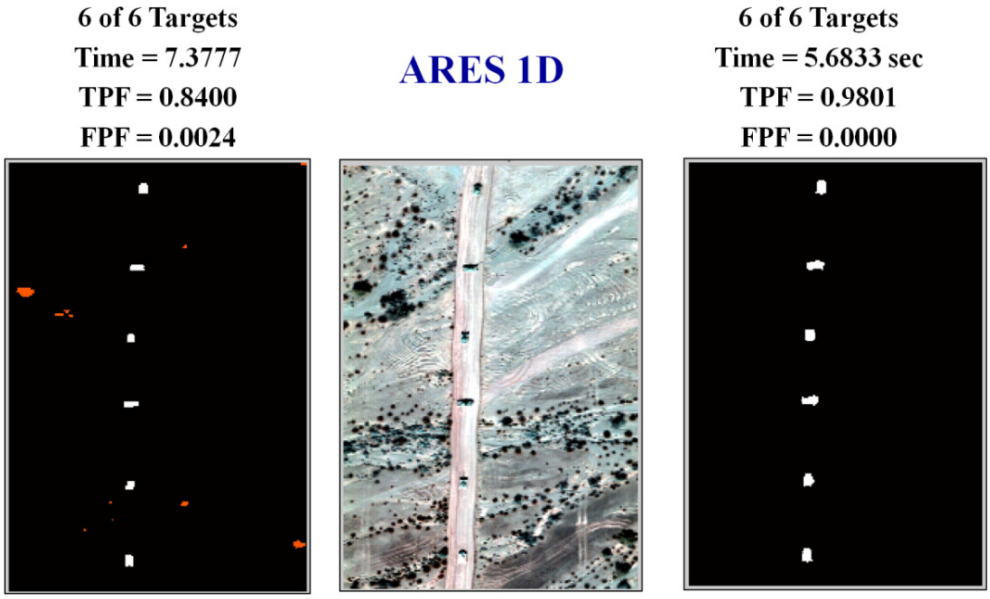


Figure 4-1(b). ARES 1D: AutoGAD (left) vs. AutoGAD-SC (right)

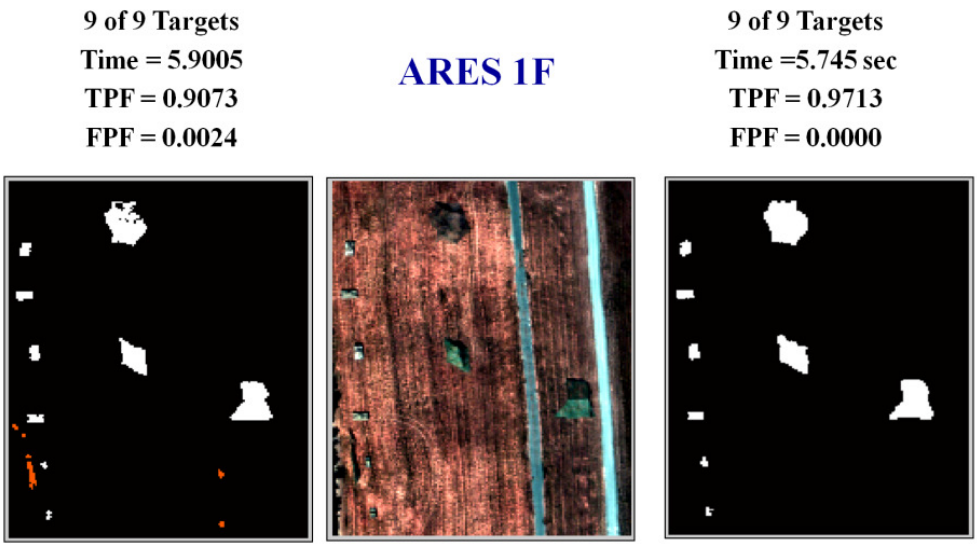


Figure 4-1(c). ARES 1F: AutoGAD (left) vs. AutoGAD-SC (right)

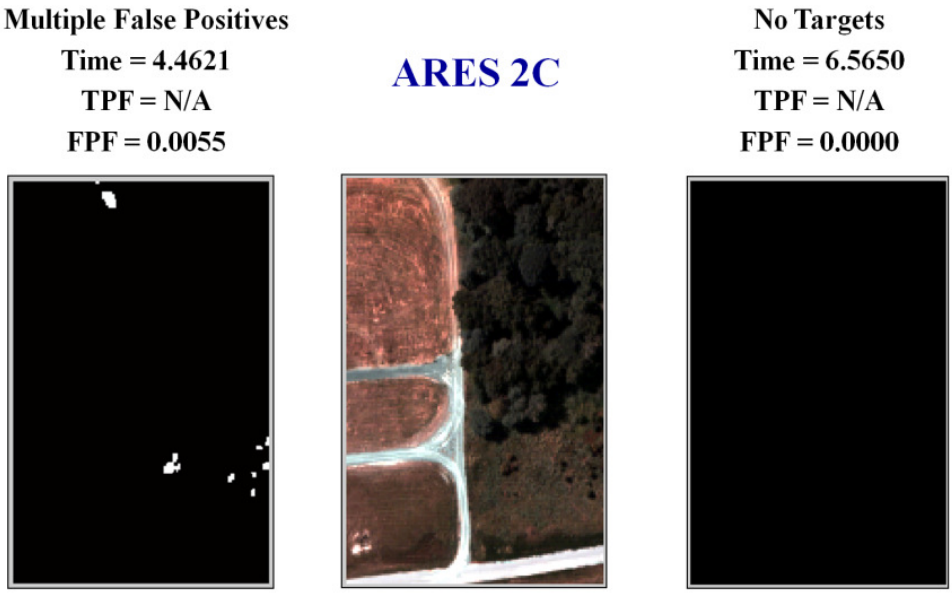


Figure 4-1(d). ARES 2C: AutoGAD (left) vs. AutoGAD-SC (right)

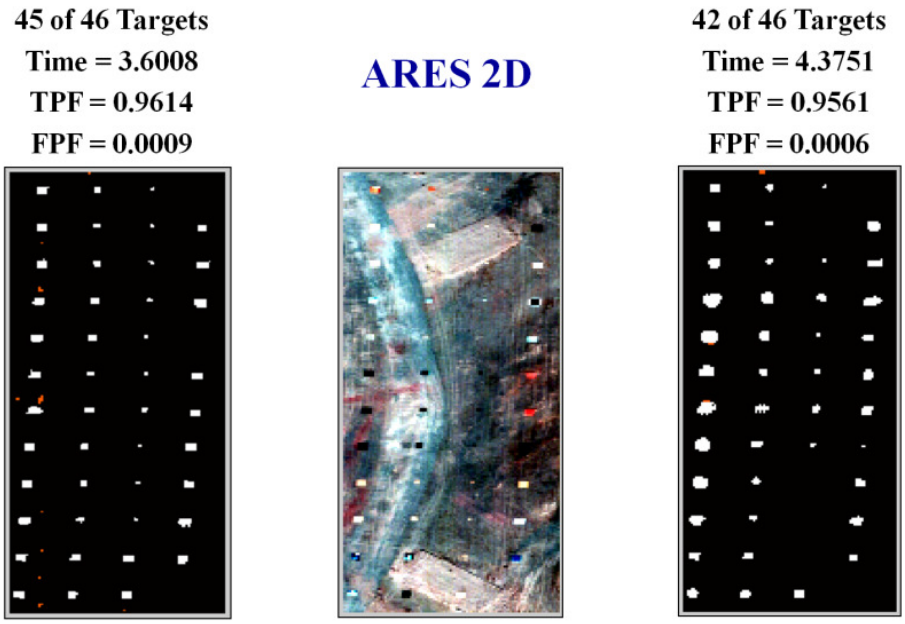


Figure 4-1(e). ARES 2D: AutoGAD (left) vs. AutoGAD-SC (right)

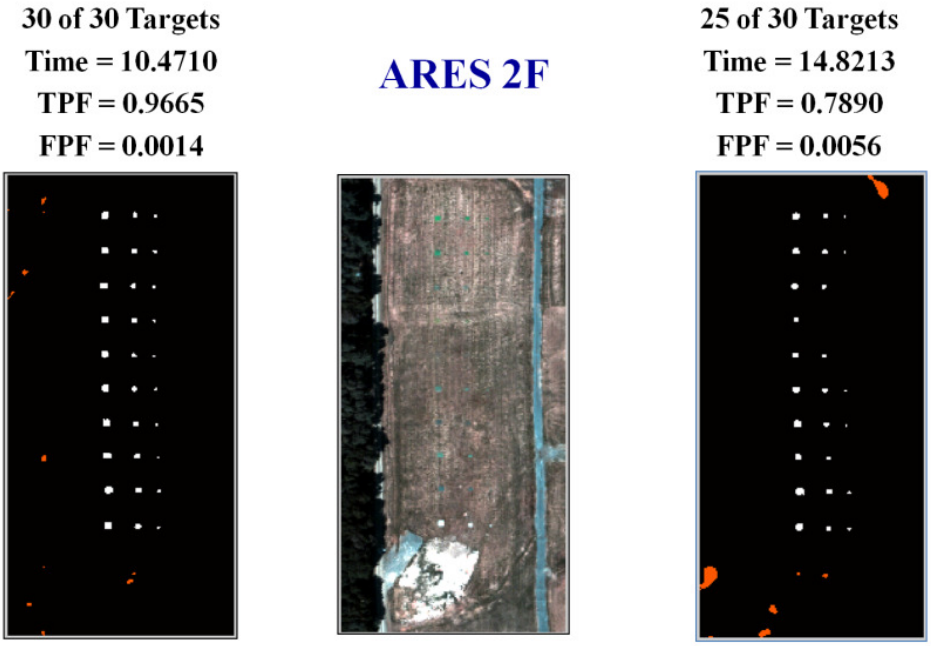


Figure 4-1(f). ARES 2F: AutoGAD (left) vs. AutoGAD-SC (right)

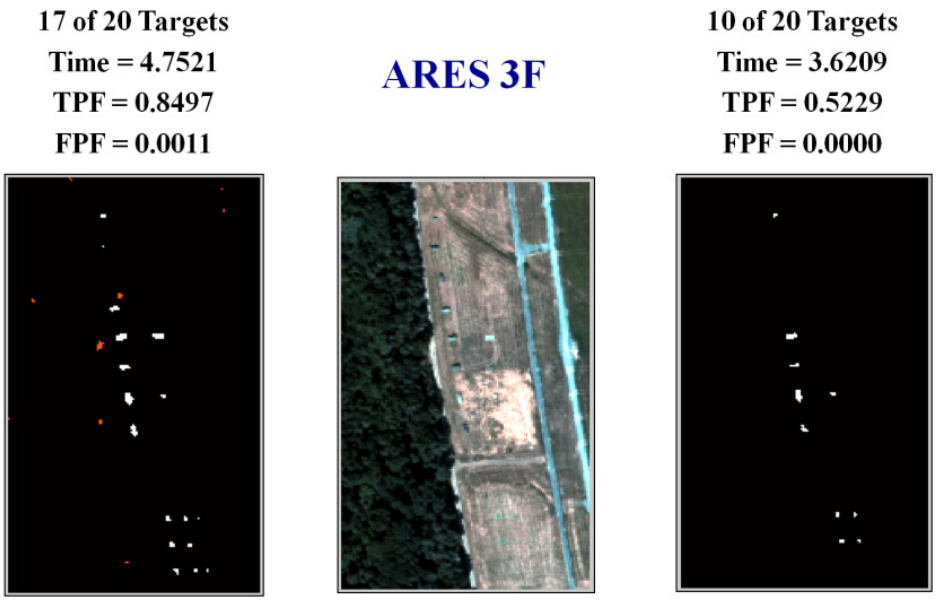


Figure 4-1(g). ARES 3F: AutoGAD (left) vs. AutoGAD-SC (right)

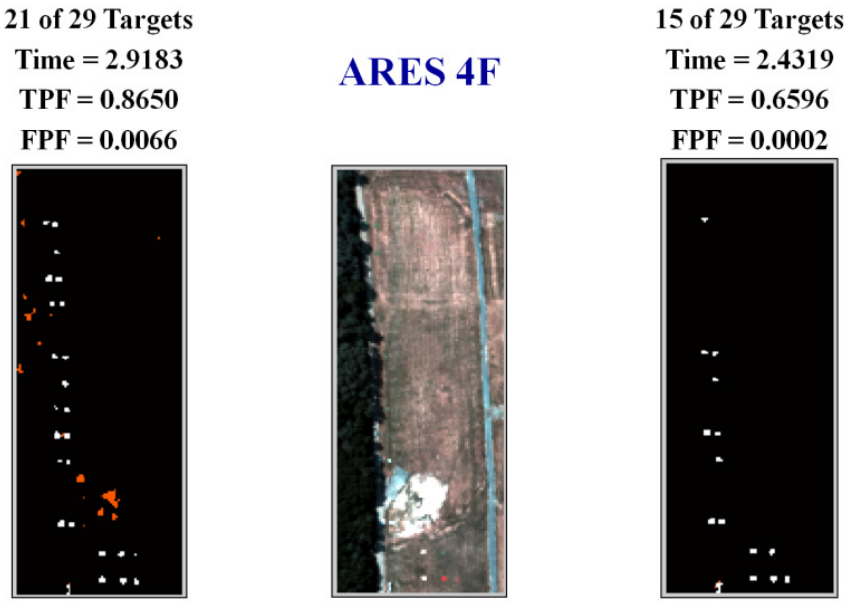


Figure 4-1(h). ARES 4F: AutoGAD (left) vs. AutoGAD-SC (right)

Although the above figures represent only one observation of a stochastic process, some trends can be noted from the results. First, AutoGAD tended to produce a somewhat higher FPF than did AutoGAD-SC. In fact, AutoGAD falsely identified one of the two non-target images (ARES 2C) as target containing, while AutoGAD-SC did not. The reduction in false positives

by AutoGAD-SC is accompanied by a reduction in TPF on several of the images. In particular AutoGAD-SC detected between 15 and 30% fewer target pixels than AutoGAD on ARES 2F, 3F, and 4F. Finally, run time for AutoGAD-SC is slightly faster on five of the eight images; however as we shall see in the next section the tested run time of 10.471 seconds produced by AutoGAD processing ARES 2F was substantially faster than typical.

Table 4-3(a) lists the overall average responses for all eight images from all 100 repetitions, while Table 4-3(b) provides the associated variance measurements for the eight images. Figures 4-2 through 4-4 divide each of these eight averages into a graphical comparison of performance by tested image. Note that on average AutoGAD produced a slightly higher TPF and FPF, required approximately 40% more time for algorithm completion, but tended to produce somewhat less variant TPF responses (over the eight images) than did AutoGAD-SC.

Algorithm	Applied Parameters	TPF	FPF	Time
AutoGAD-SC	Author Selected	0.88089	0.00149	4.556
AutoGAD-SC	Crossed Array	0.85769	0.00072	4.649
AutoGAD-SC	Response/Variance Optimization	0.85879	0.00090	4.752
AutoGAD	Recommended	0.92088	0.00203	7.879

Table 4-3(a). Average TPF, FPF, and Time from 8 tested images (100 iterations each)

Algorithm	Applied Parameters	Var(TPF)	Var(FPF)	Var(Time)
AutoGAD-SC	Author Selected	0.00030	9.425E-07	1.546
AutoGAD-SC	Crossed Array	0.00083	4.182E-07	1.358
AutoGAD-SC	Response/Variance Optimization	0.00019	1.118E-06	1.045
AutoGAD	Recommended	0.00001	4.401E-07	9.392

Table 4-3(b). Variance of TPF, FPF, and Time from 8 tested images (100 iterations each)

Figure 4-2 compares the results of testing AutoGAD for 100 iterations on each of the eight sample images to the three tests of AutoGAD-SC using the parameter settings derived in Section 3.10. All graphs include error bars representing the 95% confidence interval about the mean response. The graph indicates that AutoGAD-SC performed as well or better than AutoGAD for 5 of the eight tested images (ARES 1C, 1D, 1F, 2C, and 2D). However AutoGAD-SC performance in terms of TPF was between 15 and 20% lower for the three remaining images. Overall TPF performance was best for the original author developed parameter settings, based on a consistent 17% higher TPF when applied to ARES 3F.

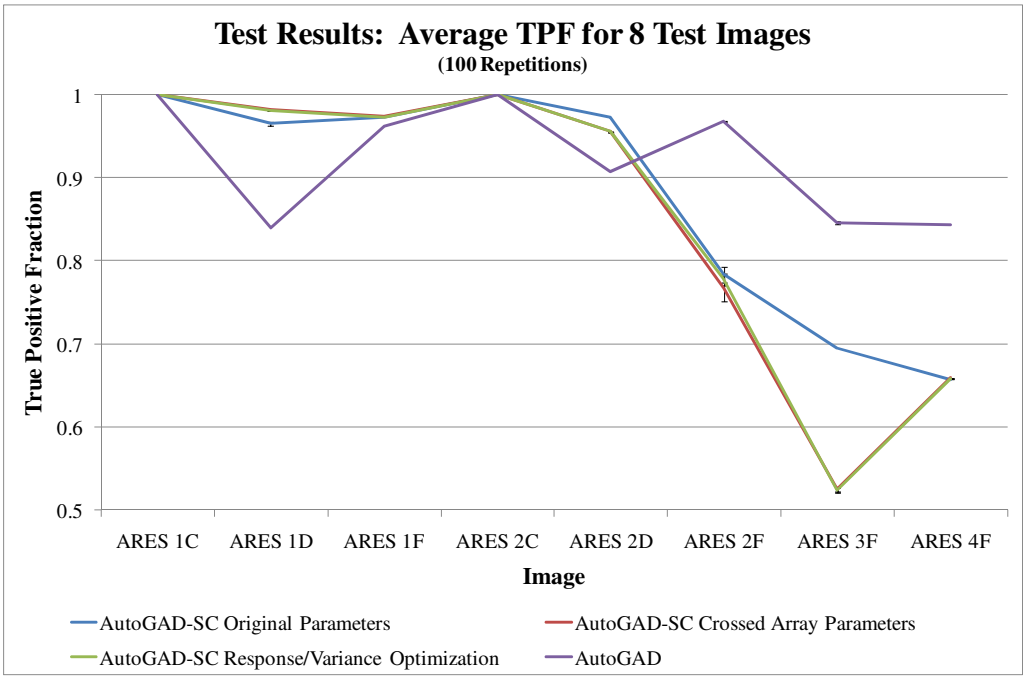


Figure 4-2. Mean TPF for AutoGAD-SC and AutoGAD with α = 0.05 Confidence Intervals

Figure 4-3 provides a similar comparison between AutoGAD and AutoGAD-SC FPF results. The chart indicates that AutoGAD typically detects more false positive pixels. On only one of the eight tested images does AutoGAD-SC incorrectly identify more pixels as targets than does AutoGAD, ARES 2F. In fact, this particular image is the only image in which greater than

0.098%, whereas the average FPF produced by AutoGAD for the same seven images was 0.203%. Of the three tested parameter settings the crossed array produced the lowest overall FPF values, however Response/Variance Optimization yielded only slightly higher average FPF values. The key result demonstrated in Figure 4-3 is the fact that AutoGAD consistently detects false positive pixels in ARES 2C, an image containing no targets, while AutoGAD-SC does not detect false positives regardless of which parameter settings were tested. Elimination of false positive detections is a key issue for any target detection so the fact that AutoGAD-SC correctly identifies both non-target images whereas AutoGAD consistently misidentified targets in one of the two images is significant.

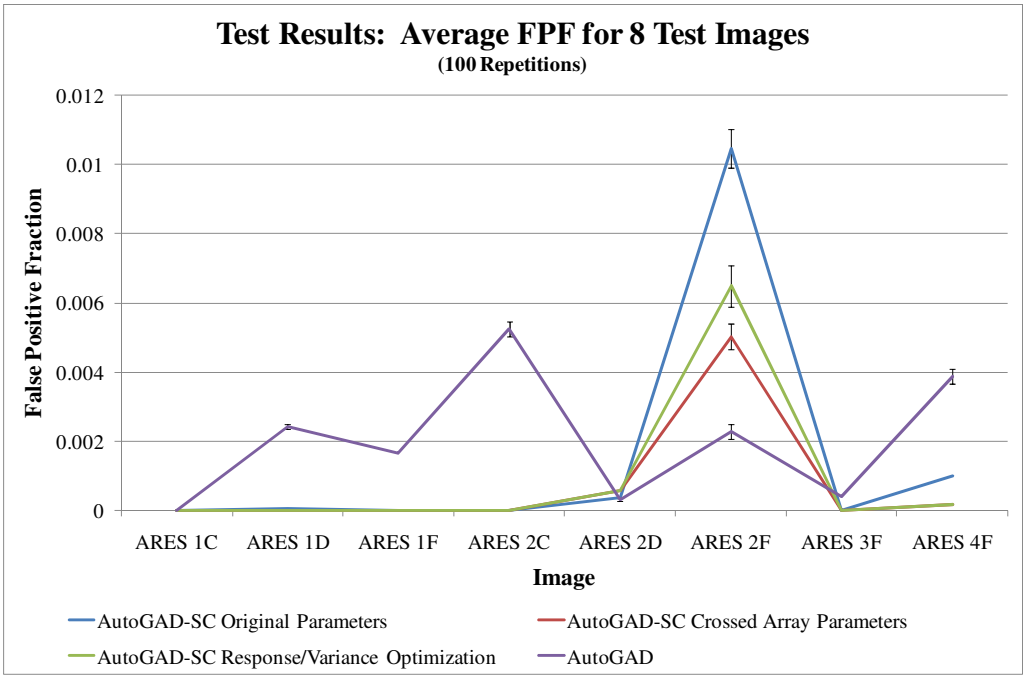


Figure 4-3. Mean FPF for AutoGAD-SC and AutoGAD with α = 0.05 Confidence Intervals

AutoGAD-SC demonstrates faster algorithm run times primarily because it processes ARES 2F in less than 15 seconds on average, regardless of parameter setting selection, whereas AutoGAD required an average of 34.78 seconds to process the same image (Figure 4-4). Recall

from Table 4-1 that AutoGAD estimated ARES 2F contained 18 dimensions, while AutoGAD-SC estimated the image contained only 9 dimensions. This dramatic reduction in dimensionality produced a 17% reduction in TPF, but generated a better than 50% decrease in algorithm run time. The question arises whether or not the higher TPF, lower FPF, and higher processing time for AutoGAD when processing ARES 2F are all factors related to the higher estimated dimensionality produced by AutoGAD. Figures 4-5(a) and (b) provide depictions of how AutoGAD separated the original ARES 2F signal into 18 components and how AutoGAD-SC separated the same image into only 9 components.

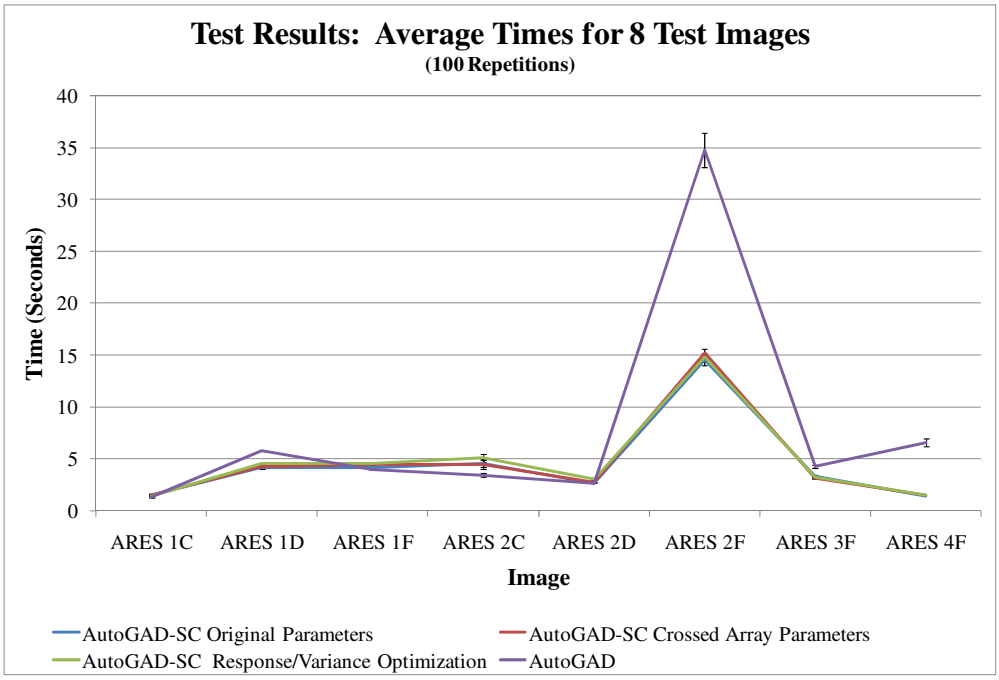


Figure 4-4. Mean Run Time for AutoGAD-SC and AutoGAD with α = 0.05 Confidence Intervals

Inspection of Figure 4-5 reveals that unlike ARES 1F (shown in Figure 2-15) AutoGAD does not overestimate the true dimensionality of ARES 2F. This is clear based on the presence of only one map consisting of primarily noise, map 9. All other bands isolate either some subset of the panels present in the image, the tree feature to the left of the image, the road feature to the

right, or the disturbed soil at the bottom. This indicates that the image actually contains at least 17 endmember spectra and that AutoGAD-SC underestimates the true dimensionality of the hyperspectral image somewhat (Figure 4-6).

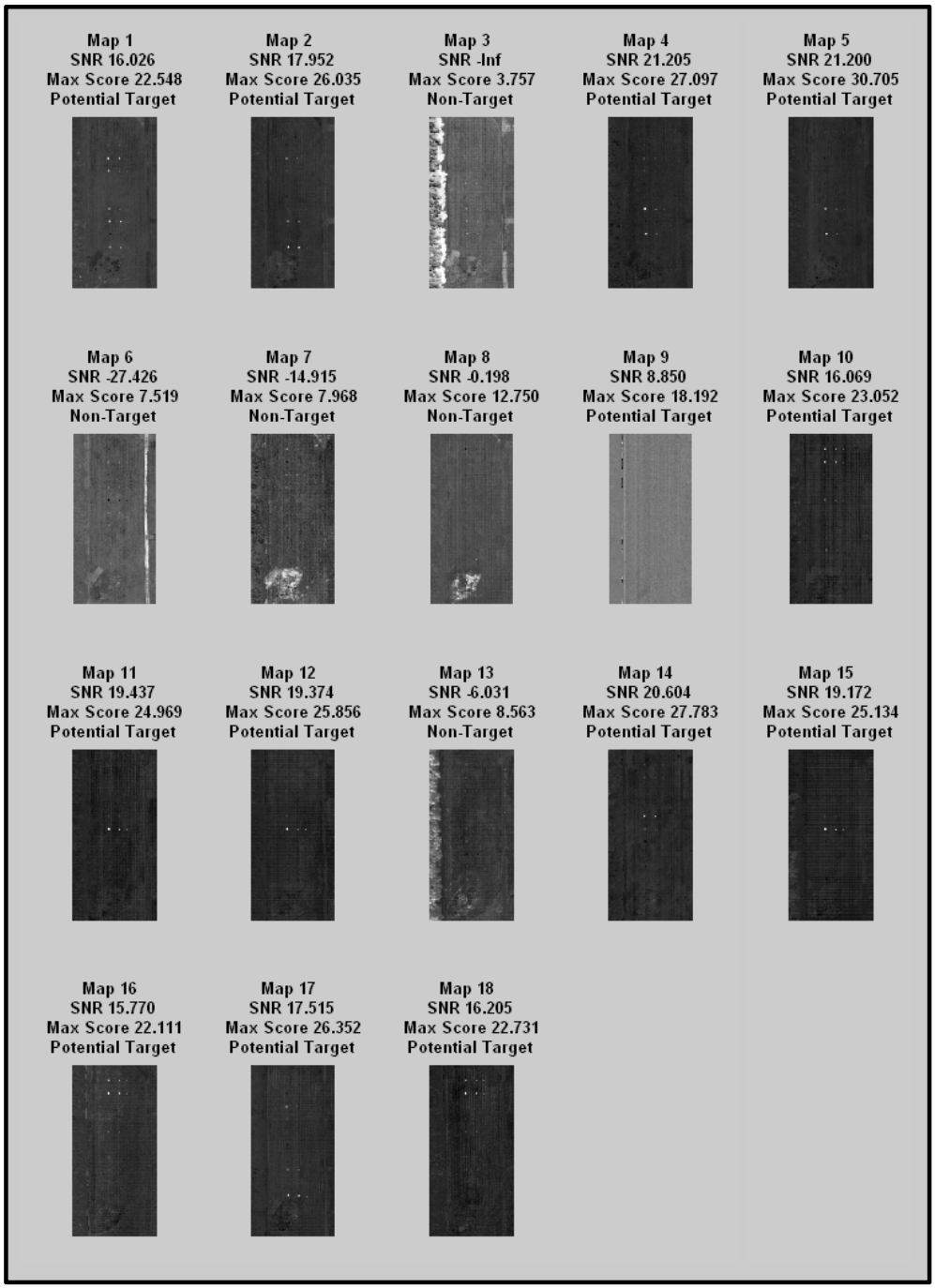


Figure 4-5. AutoGAD treatment of ARES 2F: 18 Independent Components

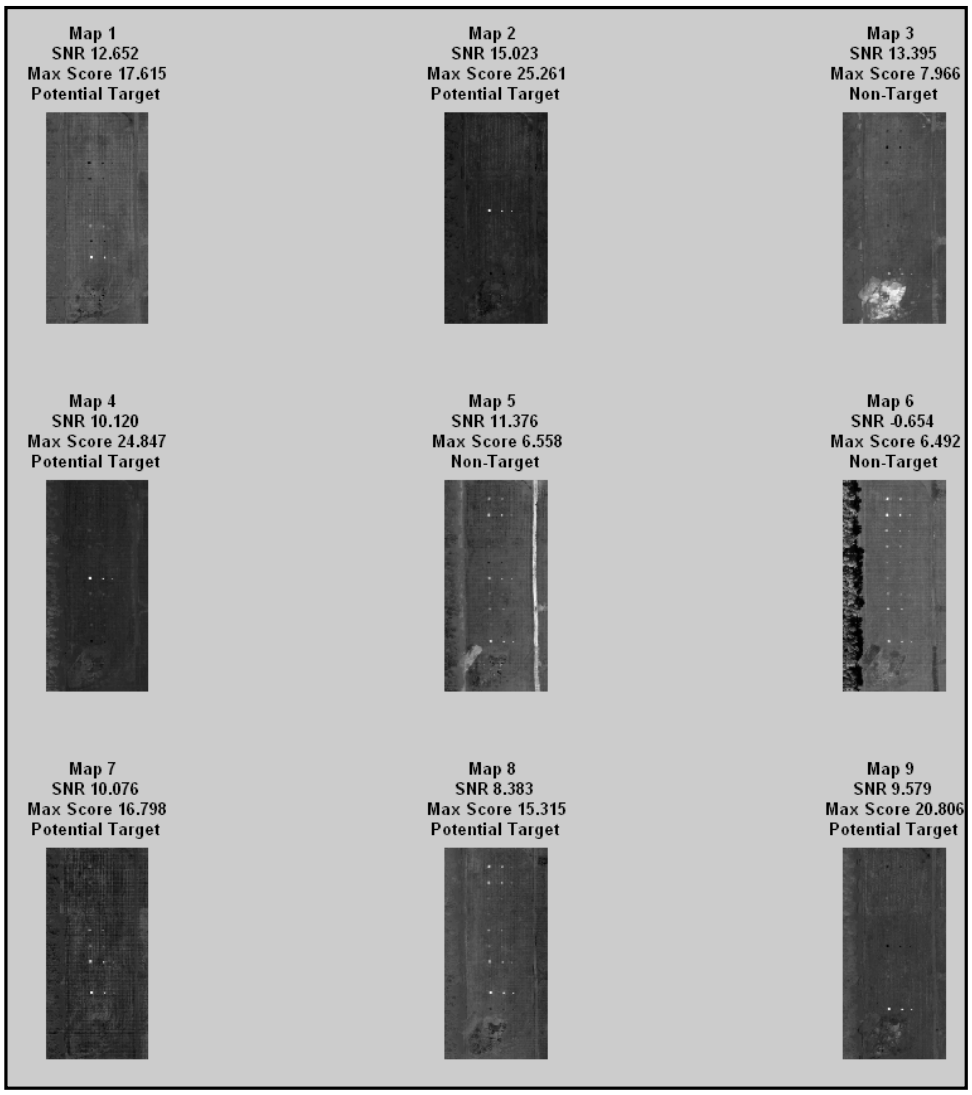


Figure 4-6. AutoGAD-SC treatment of ARES 2F: 9 Independent Components

Since the total variance in the responses was too small to effectively conduct comparisons while reviewing comparisons of the mean responses, Figures 4-7 through 4-9 are provided below. Figure 4-7 (a) and (b) display the variation in TPF responses for each of the eight images. In 4-7(a) AutoGAD clearly outperforms AutoGAD-SC in terms of variance when applied to ARES 2F. Of the three tested parameter settings the Response/Variance Optimization technique produced the lowest variance when applied to ARES 2F.

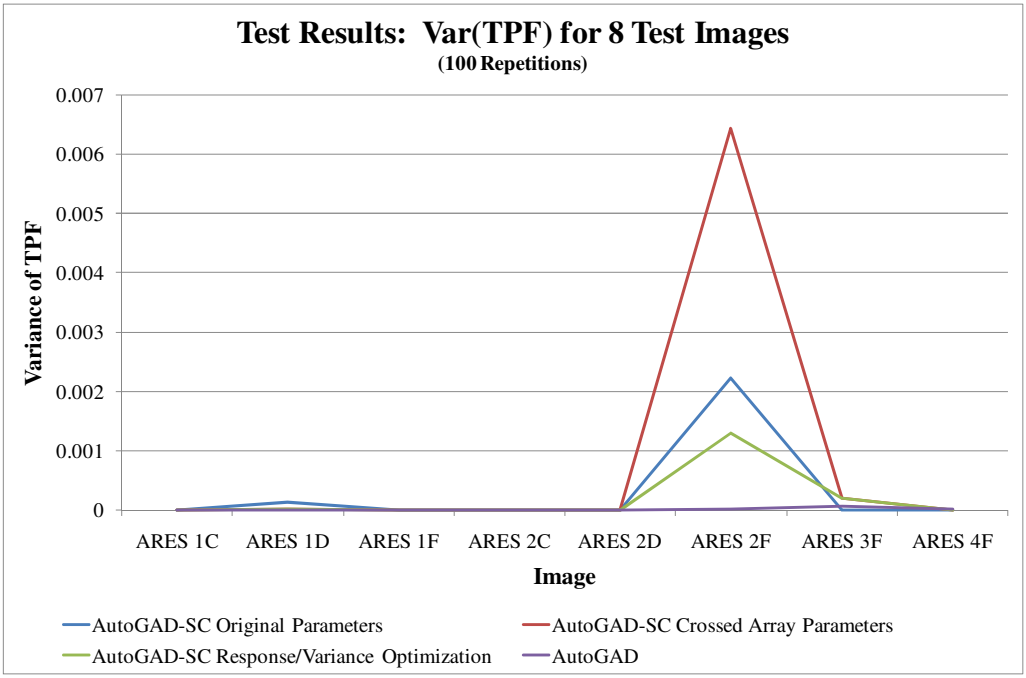


Figure 4-7(a). Variance of TPF for AutoGAD-SC and AutoGAD

Closer inspection of the TPF variance results, shown in Figure 4-7(b), reveals that for the remaining seven images AutoGAD continues demonstrate less variance than does AutoGAD-SC. In fact when the variance of TPF is averaged across all eight images as shown above in Table 4-3(b), AutoGAD demonstrates a full order of magnitude less TPF variation than AutoGAD-SC with any of the tested parameters. It is important to note however that this even the maximum variance recorded (0.0064 TPF^2 for ARES 2F processed by AutoGAD-SC with the crossed array parameters), represents a variance of only 0.84% of the TPF response using the same parameters. On average AutoGAD-SC variance using the author selected original parameters, crossed array parameters, and Response/Variance Optimization parameters represented only a 0.03%, a 0.09%, and a 0.02% variance from the average response respectively.

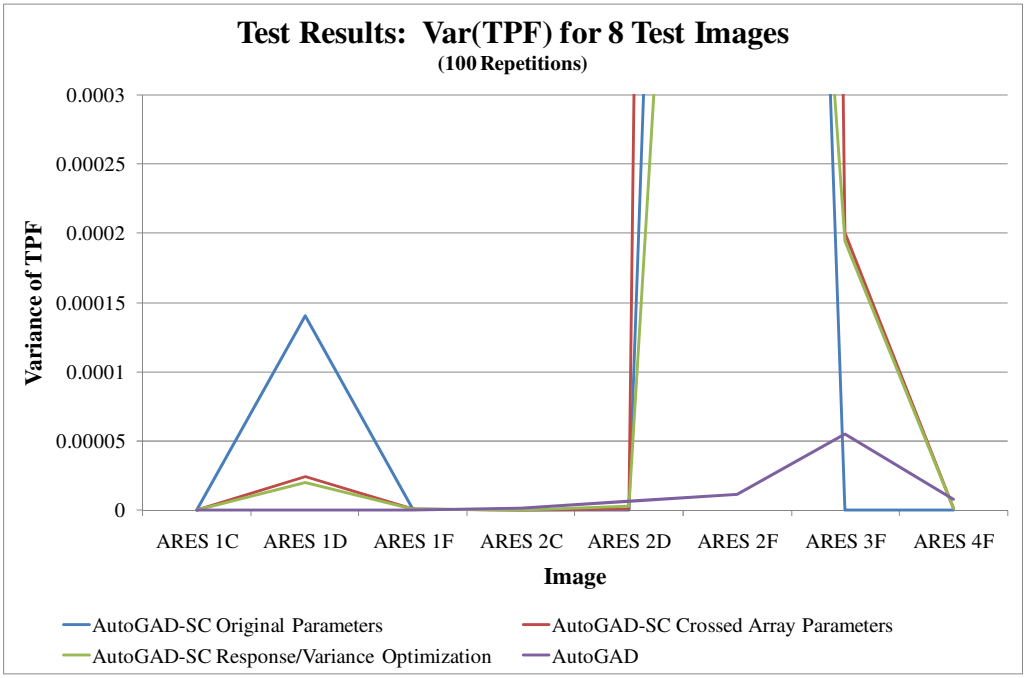


Figure 4-7(b). Variance of TPF for AutoGAD-SC and AutoGAD (scaled for detail)

Figure 4-8 presents the FPF variance resulting from tests on the eight images. Again ARES 2F tends to result in more variation when processed by AutoGAD-SC. This may be a byproduct of underestimating the number of dimensions contained in the image, however as with TPF, the variance in false positive pixels detected by AutoGAD-SC represents an exceptionally small fraction of the actual response values (0.0063%, 0.0058%, and 0.1% of the FPF for the author selected original parameters, crossed array parameters, and Response/Variance Optimization parameters respectively). When AutoGAD-SC was applied to any image other than ARES 2F, using any of the three test parameter settings, no variance was produced in FPF response. AutoGAD produced relatively low variance across all eight images, with a maximum variance of 1.19×10^{-6} for ARES 2C and an overall average of 4.4×10^{-7}.

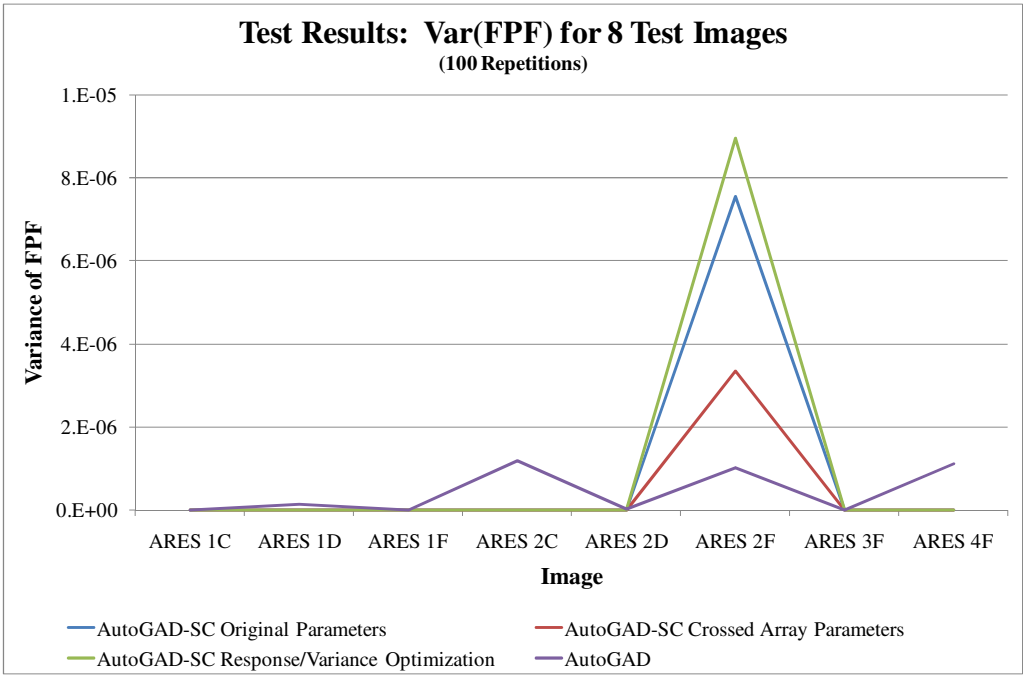


Figure 4-8. Variance of FPF for AutoGAD-SC and AutoGAD

Figure 4-9(a) shows that all four algorithm/parameter combinations performed with relatively low variation in run time. AutoGAD resulted in the highest variation when applied to ARES 1F, with roughly 9 of every 10 iterations requiring about 40 seconds to complete, while 1 in 10 required only 10 seconds for completion. This produced a time variance of 69.4 seconds^2, however Figure 4-9(b) indicates that for five of the seven remaining images AutoGAD produced the same or less run time variation than did AutoGAD-SC. Of the three sets of parameters applied to AutoGAD-SC those developed by Response/Variance model optimization consistently produced the least variation.

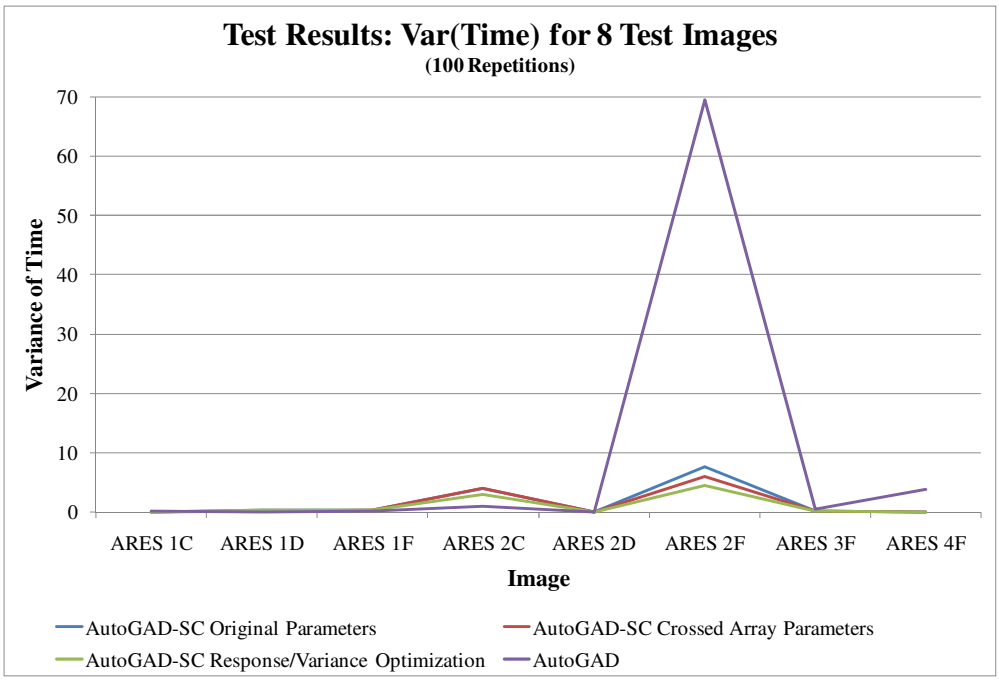


Figure 4-9(a). Variance of Run Time for AutoGAD-SC and AutoGAD

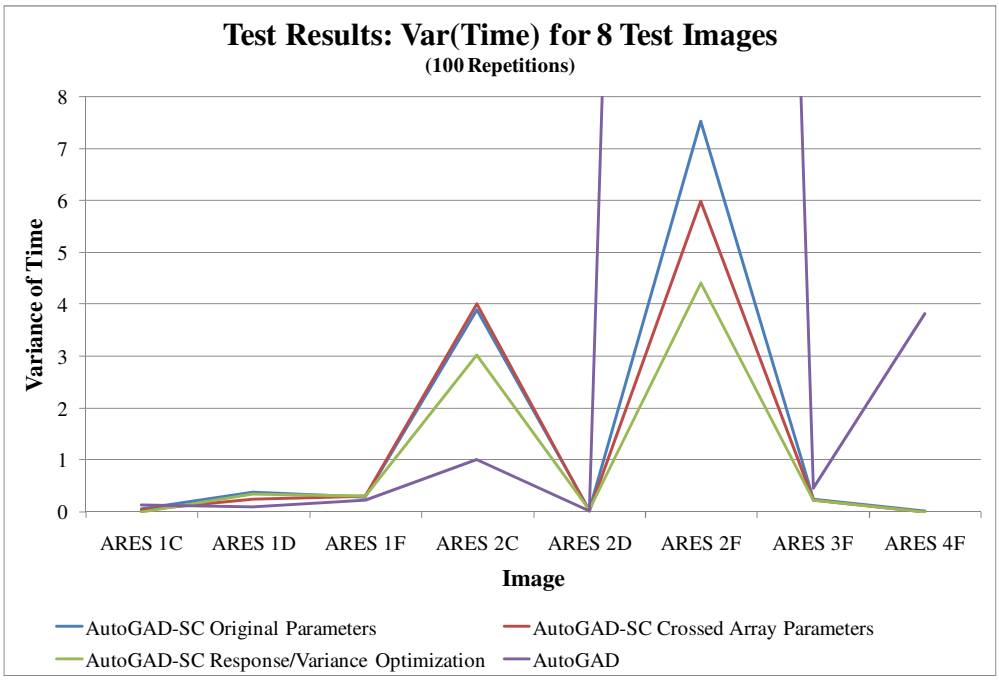


Figure 4-9(b). Variance of Run Time for AutoGAD-SC and AutoGAD (scaled for detail)

V. Conclusions and Recommendations

5.1. Assessment of AutoGAD-SC Performance

When taken as a whole AutoGAD-SC demonstrated relatively impressive performance improvements, even when compared to Johnson's original AutoGAD algorithm. Table 5-1 presents the percentage change in performance from AutoGAD based on the results of the testing performed in Section IV. Of the three tested parameter settings the original settings chosen by the author resulted in the least decrease in TPF, while providing substantial decreases in both false pixel detection and algorithm run time.

AutoGAD demonstrated an ability to autonomously detect targets in hyperspectral imagery using only the information present in the image itself and the characteristics of hyperspectral imagery. This thesis confirmed AutoGAD was capable of detecting roughly 90% of all target pixels, with less than 0.3% false positive pixels detected in an average of 7.88 seconds on a collection of eight real world hyperspectral images. AutoGAD-SC provides a means to trade a slight decrease in true positive pixel detection for a dramatic reduction in false positives while reducing processing time by as much as 42%.

Applied Parameters	TPF	FPF	Time
Author Selected	-4.34%	-26.80%	-42.18%
Crossed Array	-6.86%	-64.44%	-41.00%
Response/Variance Optimization	-6.74%	-55.44%	-39.69%

Table 5-1. AutoGAD-SC change in performance from baseline of AutoGAD

5.2. Limitations

Just as in development of AutoGAD, this thesis was produced based on a data set including only eight hyperspectral images, all taken in rural areas, where target classes consisted primarily of vehicles or panels laid out in open areas, and non-target classes consisted of grass,

brush, roads, dirt, rocks, and roads. All images were products of the HYDICE sensor, included 210 bands, of which 156 were retained for processing. Parameters such as Max IC score, PT SNR, kurtosis, and potential target fraction are all tuned for this particular sensor environment combination and would likely require recalibration given a different sensor or environment. Until AutoGAD-SC is tested for robust behavior in a variety of environments, its ability to consistently detect target pixels in any environment is uncertain.

A certain amount of caution is advisable in using target detection algorithms designed to identify statistical outliers as targets. AutoGAD-SC, like other anomaly detectors seeks pixels which stand out as different from all other pixels in the image scene. In the six target containing images examined in this research, all contained very specific man-made targets which stood out relatively clearly from the background. However, if a scene contained primarily clay buildings a few of which were covered with red tin roofs, AutoGAD-SC would be likely to identify the red tin as targets, regardless of the operators desired target set. The key here is that anomaly detectors require some end user interface or spectral matching algorithm to confirm that the anomalies detected are in fact true targets.

5.3. Contributions to the Field of HSI Target Detection

This research made the following contributions that were not found during a review of the current literature in the field:

1. A new heuristic for dynamically clustering spectral bands of a hyperspectral data cube by exploiting intra-band correlation was introduced. The author demonstrated through testing that this algorithm could simultaneously produce averaged spectra of clustered bands and assesses image dimensionality more rapidly than PCA.

2. A new technique for locating the breakpoint between background and likely target pixels was introduced. An adaptation of Johnson's MDSL technique for finding the "knee

in the curve" replaced the zero bin method of identifying the threshold between background and outlier in signal histograms.

3. Potential Target Fraction was introduced as a new mechanism for discrimination between target and non-target maps following ICA. This value enables the user to have an input regarding how densely targets are expected to be contained within the scene.

4. A new statistical parameter, Left and Right Partial Kurtosis, was introduced. This parameter estimates the overall contribution to kurtosis provided by each tail of a sample distribution provides by splitting the distribution into two halves at the mean and then measuring the kurtosis of the two half distributions independently.

5. True positives were increased and false positives reduced through the addition of a dynamic algorithm switch using the author defined Left Partial Kurtosis. This switch allows two sided thresholding for targets to be accomplished only on signals where target pixels are likely to be found in both tails of the signal histogram.

6. A variable control over how many iterations of Adaptive Iterative Noise Filtering was introduced which maintained true positives and reduced false positives, while decreasing processing time. A single user defined parameter replaced three parameters. This coefficient controls the number of iterations applied to each target map based on a function of the map's SNR value. In this way maps with high SNR values receive progressively less filtration than those with low SNR values.

5.4. Future Research

It is likely that anomaly detection algorithms could be improved by examining the spatial and spectral relationships between identified target pixels. For example pixels detected as vehicle sized and shaped areas in the image which are comprised entirely of similar spectral signatures are likely to be vehicle type targets. Manmade targets are frequently aligned in lines or regular groupings, indicating alignment with a road or assembly into some type of formation. Irregularly shaped, or excessively large/small groupings of target pixels may indicate false positive detections resulting from anomalies other than manmade objects. AutoGAD and AutoGAD-SC would both benefit from an algorithm which could examine and identify groupings of pixels identified as targets and then further classify each group into likely or

unlikely categories based on characteristics of the size and shape of the group of pixels or on variation of the spectral signatures contained within.

This research initially made an attempt to adhere to the non-negativity and sum-to-one constraints found in the LMM used to describe pixel reflectances within an image. Unfortunately NMF proved to be unsuitable as a process for unmixing hyperspectral data when algorithm run time is considered an important response. There is at least one other recently produced statistical process which appears to abide by both constraints. A technique known as Dependent Components Analysis (DCA) recently developed by José Nascimento [2006] has been applied to hyperspectral imagery for purposes other than anomaly detection and adheres to both constraints present in the LMM. Adaptation of this algorithm for the purposes of anomaly detection would provide a challenging an interesting extension to this work.

5.4. Conclusion

The addition of hyperspectral imagery to the Air Force's arsenal of ISR capabilities provides a means of resolving one of the major problems facing the Department of Defense today, that of how to efficiently manage the information produced by rapidly increasing sensor capabilities. Secretary of Defense Robert M. Gates pointed out the urgency of responding to the demands for improved ISR processes in a statement made on 23 October 2008, [Miles:2008].

> *The fusion of intelligence and operations has created "an insatiable appetite" for the information these systems provide and proof of the need to institutionalize intelligence operations.*
>
> Secretary Robert M. Gates

The merger of hyperspectral imagery with automated target detection techniques like AutoGAD and AutoGAD-SC provides one means, not only to "institutionalize intelligence operations" but to automate several steps in the process of target detection and identification.

The vast majority of images captured by ISR sensors searching for targets, contain nothing of interest, yet transmission of the non-target image for analysis requires valuable bandwidth and recognition of the image as non-target requires intervention by a human analyst. AutoGAD-SC relieves the burden of screening non-target images from target containing images from the analyst, simultaneously accelerating the target detection and identification processes. Furthermore AutoGAD-SC as an algorithm is fast and portable enough for the screening of hyperspectral imagery to be accomplished onboard the sensor platform, thereby eliminating the need to transmit each and every captured image over across bandwidth better utilized for actual target images.

The addition of a spectral matching algorithm would further accelerate the target detection process by providing a mechanism for reliably identifying targets based on the material properties of target pixels. Such an approach might not replace the expertise provided by a trained imagery analyst, but could certainly aid in the process of identification as well as increase the confidence in that identification. Ultimately the fusion of hyperspectral imagery with automated target detection algorithms such as AutoGAD-SC presents an opportunity to provide more timely, reliable intelligence to a force with an insatiable appetite for information.

Appendix A – MATLAB Code

A.1. AutoGAD-SC Code

```
%*********************************************************************%
%Band Correlation Clustered AutoGAD-SC                                 %
%                                                                     %
%                                                                     %
%Hyperspectral Autonomous Global Anomaly Detector (AutoGAD)           %
%Using FastICA                                                        %
%                                                                     %
%Author: Maj. Michael Miller                                          %
%Modified from AutoGAD v1.0 By Capt Robert Joseph Johnson             %
%Feb 2009                                                            %
%*********************************************************************%

%*********************************************************************%
% This program takes advantage of intra-band correlation to rapidly    %
% reduce the dimensionality of the image.  By doing so PCA is avoided  %
% and time is saved.  A side result of this is somewhat greater loss   %
% of information contained in the principal components, which can lead %
% to a reduction in the number of targets detected.  In particular     %
% small targets tend to be more difficult to detect with this method.  %
%*********************************************************************%

%*********************************************************************%
% Modifications made to AutoGAD:                                       %
% 1) Dimensionality reduction by clustering highly correlated bands    %
% 2) Elimination of PCA                                                %
% 3) Histogram bin width estimation by Scott's Rule                    %
% 4) Inclusion of Kurtosis and target fraction as criteria for         %
%       Target map selection.                                          %
% 5) Noise floor Maximum Distance Secant Line "knee in the curve"      %
%     (replaced first zero bin)                                        %
%*********************************************************************%

clear all;
close all;
clc;

%Tactical Decisions By User----------------------------------------------
functn=2;%objective function in ICA to use.  Options [1=tanh, 2=pow3]
orthogonalization=1;%find ICs in parallel (symm) or one by one (delf).
%Options [symm=1, defl=2]
req_corr=0.985;%Threshold correlation required for bands to be clustered together
target_fraction_thresh = 0.035; %The maximum fraction of the image expected
%to contain target pixels.
max_score_thresh=13.5;%threshold above which decision is made to declare target
Kurtosis_thresh=10;%threshold above which decision is made to declare target
PT_SNR_thresh=5.25;%threshold above which decision is made to declare target
threshold_both_sides=1;%1=identify outliers on both sides of IC signal,
%0=identify ouliers on side with highest magnitude scores only
Left_Kurt_Thresh=10;%If left side kurtosis is less than threshold program will
%not perform thresholding on both sides for that map
clean_sig=1;%0 = no signal smoothing, 1 = signal smoothing prior to target
%identification
iteration_coeff = 50;%Coefficient for the number of smoothing iterations based
% on PT_SNR
window_size=3;%image window size for smoothing
show_plots=1;%1=yes, 2=no
show_histogram=2;%1=yes, 2=no
```

```
show_spectra=2;%1=yes, 2=no (shows spectra of target pixels vs non-tgt pixels)
%--------------------------------------------------------------------------

switch num2str(functn)
    case '1'
        funct='tanh';
    case '2'
        funct='pow3';
end

switch num2str(orthogonalization)
    case '1'
        orthog='symm';
    case '2'
        orthog='defl';
end

%--------------Solicit User Input to Load HSI Image File--------------------
display('This program requires the Image Processing Toolbox for MATLAB.');
display('Make sure your version of MATLAB has this toolbox.');
display('  ');
display('Make sure you have in your working directory the all the files for');
display('FastICA and the Center_and_PCA.m file');
display('   ');
display('The first several lines in the AutoGAD algorithm detail default');
display('settings for AutoGAD.  If you would like to experiment');
display('changing these settings, hit ctrl c to interrupt this run.  Open');
display('up AutoGAD in the the editor and make changes.');
display('   ');
display('Please hit enter');
display('  ');
answer=input('');
display('Enter you image cube file name to be processed.');
display('File should be in .mat format ');
display('  ');
display('!Make sure to put it in single quotes!')
display('!Make sure the image cube is in the same directory as this code!');
display('  ');
temp1=input('');
temp2=struct2cell(load(temp1));
im_cube=temp2{1};
display('   ');
display ('Enter truth mask');
display('  ');
display('If you do not have a truth mask and this is a real target search');
display('with no truth knowledge, enter 0');
display('  ');
temp3=input('');
if temp3~=0;
    temp4=struct2cell(load(temp3));
    truth=temp4{1};
end
clear temp1
clear temp2
clear temp4
clc;
display('   ');
display('Please enter the good bands for this HSI sensor');
display('These are the bands that are NOT the atmospheric absorption bands');
display('  ');
display('If this the the 210 band HYDICE sensor, LtCol Tim Smetek concluded');
display('that the good_bands = [5:72, 78:85, 92:99, 116:134,158:199]');
display(' ');
```

```
display('If this is HYDICE data and you would like to keep these bands, type');
display('1 and hit enter');
display('   ');
display('If this is not HYDICE data or you do not want to keep those bands');
display('just hit enter and then enter the bands you wish to keep');
display('  ');
answer=input('');
if answer==1
    good_bands=[5:99,116:134,158:199];%Smetek's
bands[5:72,78:85,92:99,116:134,158:199];
else
    good_bands=input('good_bands = ');
end
%-------------------------------------------------------------------------

%-----------Ask User if they want to see color image-----------------------
display('  ');
display('Do you want to see a RGB image of your HSI file?');
display('  ');
display('If so, enter 1. If not just hit enter.          ');
display('   ');
answer=input('');
if answer==1
    Red=input('Please enter the band number for red, HYDICE is 50       ');
    display(' ');
    Green=input('Please enter the band number for green, HYDICE is 29     ');
    display('  ');
    Blue=input('Please enter the band number for blue, HYDICE is 22      ');

    R=im_cube(:,:,Red);
    G=im_cube(:,:,Green);
    B=im_cube(:,:,Blue);

    %Borrowed from Lt Col Tim Smetek, lines 142 - 163, offer a way to make
    %an RGB image look better.  The following lines are used in conjunction
    %with the mat2gray function to perform a 2% linear stretch on the image
    %data

    m1=size(R,1);
    n=size(R,2);
    low_id=floor(0.02*m1*n);
    hi_id=floor(0.98*m1*n);

    r_vec=reshape(R,m1*n,1);
    r_vec=sort(r_vec);
    r_vec=double(r_vec);
    min_R=r_vec(low_id);
    max_R=r_vec(hi_id);

    g_vec=reshape(G,m1*n,1);
    g_vec=sort(g_vec);
    g_vec=double(g_vec);
    min_G=g_vec(low_id);
    max_G=g_vec(hi_id);

    b_vec=reshape(B,m1*n,1);
    b_vec=sort(b_vec);
    b_vec=double(b_vec);
    min_B=b_vec(low_id);
    max_B=b_vec(hi_id);

    %The IPT function mat2gray to scales the values in each matrix between 0
    %and 1.  This is necessary because the matrices are of type double and
```

```
    %imshow requires double value matrices to be scaled between 0 and 1

    R=mat2gray(double(R),[min_R max_R]);
    G=mat2gray(double(G),[min_G max_G]);
    B=mat2gray(double(B),[min_B max_B]);

    %**Now stack the three matrices into a 3D array and display the image
    RGB=cat(3,R,G,B);
    figure (1)
    imshow(RGB,[]);
    title('True Color Image');
    impixelinfo;
    %**Turn-on the interactive pixel value utility
    clear R G B
    clear RGB
    clear r_vec g_vec b_vec
end
%--------------------------------------------------------------------------
tic;

%----Resize Image Cube into matrix where each row is a pixels-------------
%-----------------signature in the spectral bands-------------------------
dims=size(im_cube,3);
num_pixels=size(im_cube,1)*size(im_cube,2);
one=ones(num_pixels,1);
num_lines=size(im_cube,1);
num_col=size(im_cube,2);

%**Place all the pixel vectors into a single matrix where each row
%corresponds to a pixel vector
data_matrix=zeros(num_pixels,dims);
data_matrix_truth=zeros(num_pixels, 1);
for x=1:dims
    data_matrix(:,x)=reshape(im_cube(:,:,x),num_pixels,1);
end
if show_spectra == 2
    clear im_cube;
end
%If HSI cube is too large for MATLAB since MATLAB converts variables to
%double precision, this will make file smaller so that MATLAB can
%operate on it.
if num_pixels*dims > 25*10^6
    data_matrix=single(data_matrix);
end
%%%%%%%%%%%%%%%%%


if temp3~=0;
    data_matrix_truth=reshape(truth,num_pixels,1);
end
%--------------------------------------------------------------------------

%----------Keep bands that are not atmospheric absorption bands------------
data_matrix_new=data_matrix(:,good_bands);
dims=size(data_matrix_new,2);
clear data_matrix;
%--------------------------------------------------------------------------

%----------Set negative pixel values = 0 (remove bad pixels) --------------
[m,n] = size(data_matrix_new);
for i =1:m
    for j= 1:n
        if data_matrix_new(i,j) < 0
```

```
            data_matrix_new(i,j) = 0;
        end
    end
end
%-------------------------------------------------------------------------

%--------Group Correlated Bands together to reduce dimensionality----------
[Y,dims,k]=Dim_Redux_By_Corr3(data_matrix_new,dims,good_bands,req_corr);
clear data_matrix_new;
%-------------------------------------------------------------------------

%----------------------Perform ICA on reduced PCA space---------------------
[icasig, A, W]=fastica(Y','approach',orthog, 'g', funct, 'epsilon',...
    .00001, 'stabilization','on', 'verbose','off');
icasig=icasig';
%If an IC score has a high signals, make them always positive
icasig = icasig-one*mean(icasig);                   % Centers icasig if not centered
for j=1:k
    if abs(min(icasig(:,j)))>max(icasig(:,j))
        icasig(:,j)=-icasig(:,j);                   % By convention put longer tail in
positive direction
    end
end
clear Y
%-------------------------------------------------------------------------

%-------------------Find the Kurtosis of Each Signal----------------------
kurt=abs(kurtosis(icasig))';
%this statistic is used as part of determination of which maps are likely
%to contain identifiable target pixels
%-------------------------------------------------------------------------


%--------------------Find the Max Score of Each Signal--------------------
maxim=max(icasig)';
%this statistic is used as part of determination of which maps are likely
%to contain identifiable target pixels
%-------------------------------------------------------------------------

%--------------------Find the Skewness of Each Signal---------------------
skew=skewness(icasig)';

% This section calculates a correction factor for the fact that a
% non-gaussian distribution underlies the data, when during histogram
% creation Scott's rule (which assumes a gaussian distribution) is used
% The correction factor is used in the next section, but is currently
% commented out.
%for j=1:k
%    if abs(skew(j))>3
%        skewadj(j)=0.25;
%    else
%        skewadj(j) = 0.02*abs(skew(j))^5-0.1792*abs(skew(j))^4+...
%          0.6017*abs(skew(j))^3-0.8467*abs(skew(j))^2+0.0987*abs(skew(j))+1;
%    end
%end
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%--------------------Find the PT SNR of each signal-----------------------
%-------------------------------------------------------------------------

% ---------------Step 1: Build ICA Signal Histogram----------------------
```

```
for j=1:k
    bins{j}=[];
    freq{j}=[];

% Scott's Rule (1979) for class width (Correction factor commented out)
    bin_width_SNR(j) = 3.5*std(icasig(:,j))*(num_pixels)^(-1/3);   %*skewadj(j);
(Adjustment for non-Gaussian Skew)
    bins{j}=min(icasig(:,j)):bin_width_SNR(j):max(icasig(:,j));
    freq{j}=hist(icasig(:,j),bins{j});

%slope of line connecting peak of the histogram to the maximum value
    [maxfreq, index_freq] = max(freq{j});
    m_slope = (freq{j}(index_freq)- freq{j}(end))/(bins{j}(index_freq)-bins{j}(end));
    y_int= freq{j}(index_freq)-m_slope*bins{j}(index_freq);

% Locate Secondary Spikes or points above secant line to the right of histogram peak
    i = size(bins{j},2);
    while i > index_freq
        if (freq{j}(i) > freq{j}(i-1)+0.05*maxfreq)  | freq{j}(i) >
m_slope*bins{j}(i)+y_int+1;
            if index_freq <= i-1
                index_freq = i;
                m_slope = (freq{j}(index_freq)- freq{j}(1))/(bins{j}(index_freq)-
bins{j}(1));
                y_int= freq{j}(index_freq)-m_slope*bins{j}(index_freq);
            end
        end
        i = i-1;
    end
    length = sqrt((bins{j}(end)-bins{j}(index_freq))^2+(freq{j}(end)-
freq{j}(index_freq))^2);

%-----------Step 2: Calculate Euclidean distance from histogram------------
%--------------------to secant line between peak and tail
    Eqdist=[];
    x_int = -y_int/m_slope;
    for i=index_freq:size(bins{j},2)
        numer = abs((bins{j}(end)-bins{j}(index_freq))*(freq{j}(index_freq)...
            -freq{j}(i))-(bins{j}(index_freq)-bins{j}(i))*(freq{j}(end)-
freq{j}(index_freq)));
        Eqdist(i)=numer/length;
    end

%find the point on the histogram with the largest distance from the line
% connecting the endpoints
    [max_Eqdist, index_dim]=max(Eqdist);
    thresh_pt(j)=bins{j}(index_dim);

%------------------Plot unfiltered signal histogram------------------------
    d=ceil(sqrt(k));
    if show_histogram==1
        figure(10)
        subplot(d,d,j);
        plot(bins{j},freq{j},'LineWidth',2);
        title({sprintf('Map %i \n Noise Threshold %4.3f',j,thresh_pt(j))...
            ,'Signal Histogram'},'fontweight','b');
        xlabel('Independent Conmponent Intensity');
        ylabel('Frequency');
    end
end

%-------------Step 3: Calculate PTSNR from variance of--------------------
%-----------------signal vs variance of background-------------------------
```

```
PT_SNR=zeros(k,1);
for j=1:k
    potent_target=[];
    potent_bkrd=[];
    %find the indices of those pixels greater than threshold
    ind = icasig(:,j)>thresh_pt(j);
    %store those pixels greater than threshold in vector
    potent_target=icasig(ind,j);
    if size(potent_target,1)==0
        potent_target=0;
    end
    %find the indices of those pixels less than threshold
    ind2 = icasig(:,j)<=thresh_pt(j);

    %tgt_pct is used to determine if a map contains too many target pixels
    %for them to be likely to be actual targets rather than some natural anomaly
    tgt_pct(j) = sum(ind)/(sum(ind2)+sum(ind));

    %store those pixels less than threshold in vector
    potent_bkrd=icasig(ind2,j);
    power_target(j)=var(potent_target);
    power_bkrd(j)=var(potent_bkrd);
    PT_SNR(j)=10*log10(power_target(j)/power_bkrd(j));
end
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%quadres= horzcat(kurt,maxim,PT_SNR,tgt_pct');


%---Plot Abundance Maps from ICs Frames with PT SNR and Max Pixel Score----
if show_plots==1
    figure (4)
    for j=1:k
        subplot(d,d,j)
        ICsig(:,:,j)=reshape(icasig(:,j),num_lines,num_col);
        ICsig_grey(:,:,j)=mat2gray(double(ICsig(:,:,j)));
        imshow(ICsig_grey(:,:,j));
        if maxim(j)>=max_score_thresh && PT_SNR(j)>=PT_SNR_thresh...
                && kurt(j)>=Kurtosis_thresh && tgt_pct(j) <= target_fraction_thresh
            title({sprintf('Map %i \n SNR %4.3f \n Max Score %4.3f',j,PT_SNR(j)...
                ,maxim(j)),'Potential Target'},'fontweight','b');
        else
            title({sprintf('Map %i \n SNR %4.3f \n Max Score %4.3f',j,PT_SNR(j)...
                ,maxim(j)),'Non-Target'},'fontweight','b');
        end
    end
    clear ICsig;
    clear ICsig_grey;
%--------------------------------------------------------------------------

%----------------------------Plot IC signals-------------------------------
    figure(5)
    PT_SNR_line=one*thresh_pt;
    for j=1:k
        subplot(d,d,j)
        plot(icasig(:,j),'.', 'MarkerEdgeColor','r');
        hold on
        plot(PT_SNR_line(:,j),'LineWidth',2);
        xlabel('Pixel');
        ylabel('Abundance (IC Score)');
        if maxim(j)>=max_score_thresh && PT_SNR(j)>=PT_SNR_thresh
```

```
            title({sprintf('Map %i \n SNR %4.3f \n Max Score %4.3f',j,PT_SNR(j)...
                ,maxim(j)),'Potential Target'},'fontweight','b');
        else
            title({sprintf('Map %i \n SNR %4.3f \n Max Score %4.3f',j,PT_SNR(j)...
                ,maxim(j)),'Non-Target'},'fontweight','b');
        end
        axis([0,num_pixels,-15,30]);
    end
    clear PT_SNR_line
%-------------------------------------------------------------------------
end

%----------------Keep only Those Signals Above Both Thresholds-------------
ind_max=[];
ind_SNR=[];
ind_kurt=[];
ind_tgt_frac=[];
ind_both=[];
ind_max = maxim>=max_score_thresh;
ind_SNR = PT_SNR>=PT_SNR_thresh;
ind_kurt = kurt>=Kurtosis_thresh;
ind_tgt_frac = tgt_pct'<=target_fraction_thresh;
ind_both=ind_max+ind_SNR+ind_kurt+ind_tgt_frac;
[rind,cind]= find(ind_both==4);
if size(rind,1)==0
    display('NO TARGETS')
    target_sig=zeros(num_pixels,1);
    target_vec=zeros(num_pixels,1);
    num_tgt_maps = 0;                   % Added M. Miller to manage zero-target images
else
    target_sig=icasig(:,rind);
    thresh_pt_tgt=thresh_pt(:,rind);
    thresh_pt_tgtL=thresh_pt(:,rind);
    num_tgt_maps=size(target_sig,2);    % Moved from line 440 to line 437 by M. Miller
    index = 1;
    for j = rind'
        bins_tgt{index}=bins{j};
        freq_tgt{index}=freq{j};
        index=index+1;
    end
end
clear icasig;
clear bins;
clear freq;

for j=1:num_tgt_maps
    tgt_sig_map(:,:,j)=reshape(target_sig(:,j),num_lines,num_col);
end

%-------------------------------------------------------------------------
threshold_both_sides=0;
if size(rind,1)~=0
%-------------Show Abundance Maps of Retained Target Signals---------------
    if show_plots==1
        d=ceil(sqrt(num_tgt_maps));
        tgt_gray=[];
        figure(6)
        for j=1:num_tgt_maps
            subplot(d,d,j);
            tgt_sig_map_gray(:,:,j)=mat2gray(tgt_sig_map(:,:,j));
            imshow(tgt_sig_map_gray(:,:,j));
            title({sprintf('Map %i \n SNR %4.3f \n Max Score %4.3f',rind(j),...
                PT_SNR(rind(j)),maxim(rind(j))),'Potential Target'},...
```

```
                'fontweight','b');
        end
        clear tgt_sig_map_gray
    end
%-------------------------------------------------------------------------

%-----Split ICA Signals of any target map into two halves at mean = 0------
%-----------------and calculate left tail kurtosis value------------------
target_sig_sort=sort(target_sig);
clear target_sig
for j=1:num_tgt_maps
    neg_pixels=target_sig_sort(:,j)<0;
    neg_count=sum(neg_pixels);
    ica_left{j}=target_sig_sort(1:neg_count,j);
    left_kurt(j)=abs(kurtosis(ica_left{j}));
end

%----Set threshold for negative signal values on selected target maps------
%-----------if user specified this option (threshold both sides)-----------
if num_tgt_maps > 0  % && added by M. Miller
    for j=1:num_tgt_maps
        if left_kurt(j) > Left_Kurt_Thresh
        threshold_both_sides=1;

%Slope of line connecting peak of the histogram to the minimum value
        [maxfreq, index_freq] = max(freq_tgt{j});
        m_slope = (freq_tgt{j}(index_freq)- freq_tgt{j}(1))/(bins_tgt{j}(index_freq)-
bins_tgt{j}(1));
        y_int= freq_tgt{j}(index_freq)-m_slope*bins_tgt{j}(index_freq);

% Determine if any points on the histogram fall above the secant line
        i = 1;
        while i < index_freq
            if (freq_tgt{j}(i) > freq_tgt{j}(i+1)+0.05*maxfreq)  | freq_tgt{j}(i) >
m_slope*bins_tgt{j}(i)+y_int+1;
                if index_freq >= i+1
                    index_freq = i;
                    m_slope = (freq_tgt{j}(index_freq)-
freq_tgt{j}(1))/(bins_tgt{j}(index_freq)-bins_tgt{j}(1));
                    y_int= freq_tgt{j}(index_freq)-m_slope*bins_tgt{j}(index_freq);
                end
            end
            i = i+1;
        end
        length = sqrt((bins_tgt{j}(end)-bins_tgt{j}(index_freq))^2+(freq_tgt{j}(end)-
freq_tgt{j}(index_freq))^2);

% Calculate Euclidean distance from histogram to line connecting peak to end
        Eqdist=[];
        x_int = -y_int/m_slope;
        for i=1:index_freq
            numer = abs((bins_tgt{j}(1)-
bins_tgt{j}(index_freq))*(freq_tgt{j}(index_freq)-...
                freq_tgt{j}(i))-(bins_tgt{j}(index_freq)-
bins_tgt{j}(i))*(freq_tgt{j}(1)-freq_tgt{j}(index_freq)));
            Eqdist(i)=numer/length;
        end
%find the point on the histogram with the largest distance from the line
% connecting the endpoints
        [max_Eqdist, index_dim]=max(Eqdist);
        thresh_pt_ident_left(j)=bins_tgt{j}(index_dim);
        end
    end
```

```
end
%-------------------------------------------------------------------------


%----Clean (IAN Filtering)Target Signals prior to Identification-----------
    if clean_sig==1
        for j=1:num_tgt_maps
            iterations = round(iteration_coeff*PT_SNR_thresh/PT_SNR(rind(j)));
            for c=1:iterations
                    [tgt_sig_map(:,:,j)]=wiener2(tgt_sig_map(:,:,j), ...
                        [window_size,window_size]);
            end
        end
%-------------------------------------------------------------------------


%-------------------Plot IAN Filtered Target Maps-------------------------
        if show_plots==1
            for j=1:num_tgt_maps
                clean_map_gray(:,:,j)=mat2gray(tgt_sig_map(:,:,j));
            end
            figure(7)
            for j=1:num_tgt_maps
                subplot(d,d,j);
                imshow(clean_map_gray(:,:,j));
                colormap(jet)
                title(sprintf('Filtered Map %i',rind(j)),'fontweight','b');
            end
        end
    end
%-------------------------------------------------------------------------

%----------Identify Target Pixels from Selected Target Maps---------------
%-------------------------------------------------------------------------
    target_sig_clean=[];
    for j=1:num_tgt_maps
        target_sig_clean(:,j)=reshape(tgt_sig_map(:,:,j), num_pixels, 1);
    end

    target=zeros(num_pixels, num_tgt_maps);
    for j=1:num_tgt_maps
        target(:,j) = target_sig_clean(:,j)>thresh_pt_tgt(j);
    end

    target_vec=sum(target,2);
end
%-------------------------------------------------------------------------
%Checks both sides of the selected target signals for target pixels if user
%specified this option
if threshold_both_sides==1 && num_tgt_maps > 0  % && added by M. Miller
    target_left=zeros(num_pixels, num_tgt_maps);
    for j=1:num_tgt_maps
        if left_kurt(j) > Left_Kurt_Thresh
            target_left(:,j)= target_sig_clean(:,j)<thresh_pt_ident_left(j);
        end
    end

    target_vec_left=sum(target_left,2);
    target_vec=target_vec+target_vec_left;
end
%-------------------------------------------------------------------------
target_pic = reshape(target_vec,num_lines,num_col);
%-------------------------------------------------------------------------
```

```
%----------Plot Target Signals with Calculated Thresholds-----------------
if show_plots ==1
    if size(rind,1)~=0
        d=ceil(sqrt(num_tgt_maps));
        linetrh_ident=one*thresh_pt_tgt;
        if threshold_both_sides==1
            for j=1:num_tgt_maps
                if left_kurt(j) > Left_Kurt_Thresh
                    linetrh_ident_left(:,j)=thresh_pt_ident_left(j)*one;
                end
            end
        end
        figure(8)
        for j=1:num_tgt_maps
            subplot(d,d,j)
            plot(target_sig_clean(:,j),'.', 'MarkerEdgeColor','r');
            hold on
            plot(linetrh_ident(:,j),'LineWidth',2);
            if threshold_both_sides==1
                if left_kurt(j) > Left_Kurt_Thresh
                    plot(linetrh_ident_left(:,j),'LineWidth',2);
                end
            end
            xlabel('Pixel');
            ylabel('Abundance (IC Score)');
            title({sprintf('Map %i \n SNR %4.3f \n Max Score %4.3f',rind(j),...
                PT_SNR(rind(j)),maxim(rind(j))),'Potential Target'},...
                'fontweight','b');
            axis([0,num_pixels,-15,30]);
        end
        clear linetrh_ident
    end
    clear one
end
%-------------------------------------------------------------------------

%-------Grade Performance of AutoGAD if Truth Mask was Provided-----------
if temp3~=0;
%----------------------Confusion Matrix Calculation-----------------------
    ConfusMat=[];
    ConfusMat(1,1)=0; %(TP)
    ConfusMat(1,2)=0; %(FP)
    ConfusMat(2,1)=0; %(FN)
    ConfusMat(2,2)=0; %(TN)

    for i=1:num_pixels
        if target_vec(i,1)>= 1 && data_matrix_truth(i,1) >= 1
            ConfusMat(1,1)=ConfusMat(1,1)+1;
        else
            if target_vec(i,1)>= 1 && data_matrix_truth(i,1) == 0
                ConfusMat(1,2)=ConfusMat(1,2)+1;
            else
                if target_vec(i,1)== 0 && data_matrix_truth(i,1) == 1
                    ConfusMat(2,1)=ConfusMat(2,1)+1;
                else
                    if target_vec(i,1)== 0 && data_matrix_truth(i,1) == 0 || 2
                        ConfusMat(2,2)=ConfusMat(2,2)+1;
                    end
                end
            end
```

```
        end
    end

    APER = (ConfusMat(1,2)+ConfusMat(2,1))/(num_pixels);
    TPF = ConfusMat(1,1)/(ConfusMat(1,1)+ConfusMat(2,1));
    FPF = ConfusMat(1,2)/(ConfusMat(1,2)+ConfusMat(2,2));
    Perc_tgt = ConfusMat(1,1)/(ConfusMat(1,1)+ConfusMat(1,2));
%-------------------------------------------------------------------------

%-------------------Show Target Locations to the User---------------------
    target_vec_color=zeros(num_pixels,1);
    for i=1:num_pixels
        if target_vec(i,1)>=1 && data_matrix_truth(i,1)>=1
            target_vec_color(i,1)=4;
        elseif target_vec(i,1)>=1 && data_matrix_truth(i,1)==0
            target_vec_color(i,1)=2;
        end
    end
    target_pic_color = uint8(reshape(target_vec_color,num_lines,num_col));
    if size(rind,1)~=0
        figure(9)
        imshow(mat2gray(target_pic_color));
        colormap('Hot')
        title(sprintf('TPF = %4.6f \n FPF = %4.6f \n Percent TGT = %4.6f',...
            TPF, FPF,Perc_tgt),'fontweight','b');
        impixelinfo;
    elseif size(rind,1)==0
        figure(9)
        imshow(target_pic);
        title('No Targets Detected')
    end
    figure (2)
    imshow(truth,[]);
    title('Truth Mask');
    impixelinfo;
else
    if size(rind,1)~=0
        figure(9)
        imshow(target_pic)
        title({'Suspected Target Pixels'});
        impixelinfo;
    elseif size(rind,1)==0
        figure(9)
        imshow(target_pic)
        title({'No Targets Detected'});
        impixelinfo;
    end
end

%-----------Plot Target Pixel Spectra for each Target Map-----------------
if show_spectra ==1
    if size(rind,1)~=0
        for i=1:num_tgt_maps
            figure (i+10)
            tgt_pixel = 1;
            false_tgt_pixel=1;
            non_tgt_pixel=1;
            for j=1:num_lines
                for k=1:num_col
                    if target_pic_color(j,k)==4
                        tgt_spect(tgt_pixel,:)=im_cube(j,k,:);
                        subplot(3,1,1)
                        plot(tgt_spect(tgt_pixel,:),'b');
```

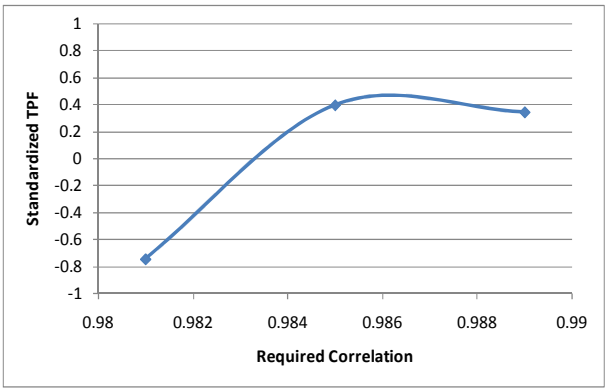
```
                        hold on
                        tgt_pixel=tgt_pixel+1;
                        title(sprintf('Target Pixels Raw Spectra'),'fontweight','b');
                    elseif target_pic_color(j,k)==2
                        false_tgt_spect(false_tgt_pixel,:)=im_cube(j,k,:);
                        subplot(3,1,2)
                        plot(false_tgt_spect(false_tgt_pixel,:),'r');
                        hold on
                        false_tgt_pixel=false_tgt_pixel+1;
                        title(sprintf('False Target Pixels Raw
Spectra'),'fontweight','b');
                    elseif rand<.001 && non_tgt_pixel<=20
                        non_tgt_spect(non_tgt_pixel,:)=im_cube(j,k,:);
                        subplot(3,1,3)
                        plot(non_tgt_spect(non_tgt_pixel,:),'y');
                        hold on
                        non_tgt_pixel=non_tgt_pixel+1;
                    end
                end
                axis ([0 210 0 15000]);
            end

        end
    end
end

time=toc
```

A.2. Spectral Clustering Code

```
%*********************************************************************%
%Spectral Band Clustering Subroutine                                  %
%                                                                     %
%                                                                     %
%Function called by AutoGAD to reduce dimensionality of               %
%Hyperspectral Data                                                   %
%                                                                     %
%Author: Maj. Michael Miller                                          %
%                                                                     %
%Feb 2009                                                             %
%*********************************************************************%

%*********************************************************************%
% This program seeks highly correlated adjacent bands within a        %
% hyperspectral image.  If two or more adjacent bands exceed a user   %
% specified level of correlation, they are retained as a single average%
% spectral signature of the "clustered" bands.  All clusters are       %
% returned to the main program (AutoGAD-SC) with the assessed number   %
% of dimensions based on the number of clusters formed.                %
%*********************************************************************%

function[data_matrix_final,new_dims,clusters] =
Dim_Redux_By_Corr(data_matrix_new,dims,good_bands,threshold)

first = 1;
data_matrix_final = [];
clusters = 0;
clustered_bands=[];
while first <= dims-1
   last = first;
   j = first+1;
   correl(first,j) = corr(data_matrix_new(:,first),data_matrix_new(:,j));
   while good_bands(j)==good_bands(j-1)+1 && correl(first,j) >= threshold && j < dims
       j=j+1;
       correl(first,j) = corr(data_matrix_new(:,first),data_matrix_new(:,j));
       last = last+1;
   end
   if last > first && j<dims
        center = last;
        j=last+1;
        correl(center,j) =corr(data_matrix_new(:,center),data_matrix_new(:,j));
        while correl(center,j) >= threshold && j < dims &&
good_bands(j)==good_bands(j-1)+1
            j=j+1;
            correl(center,j) = corr(data_matrix_new(:,center),data_matrix_new(:,j));
            last=last+1;
        end
   end
   if last-first>=2
   data_matrix_final =
horzcat(data_matrix_final,mean(data_matrix_new(:,first:last),2));
   clustered_bands = vertcat(clustered_bands,[first,last]);
   clusters=clusters+1;
   end
   first = last+1;
end
new_dims=size(data_matrix_final,2);
```

Appendix B – Taguchi Marginal Mean and Marginal SNR Plots

B.1. True Positive Fraction (TPF) Response and SNR Plots



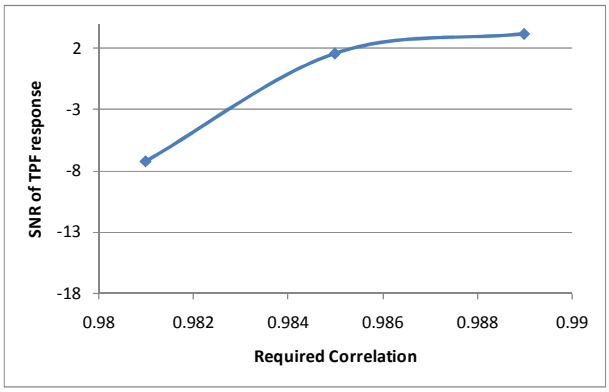
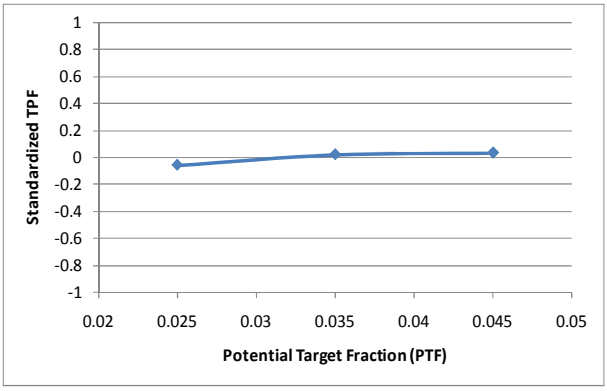


Figure B-1. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized TPF vs. Required Correlation Threshold



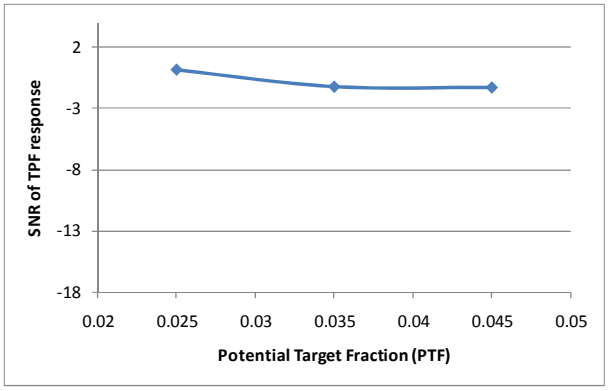
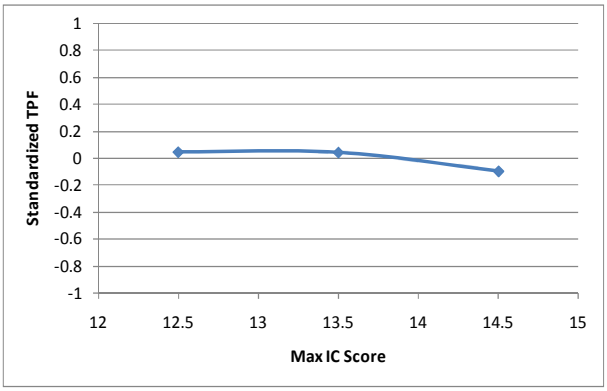


Figure B-2. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized TPF vs. Potential Target Fraction Threshold



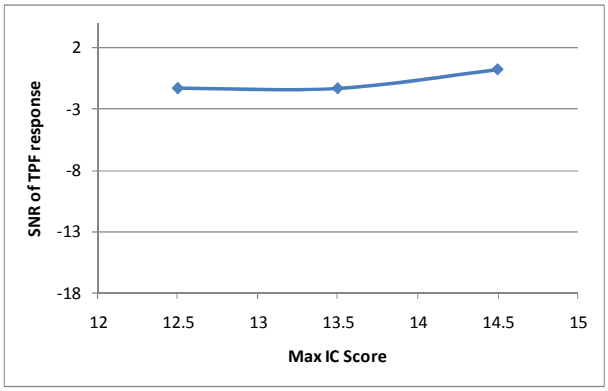


Figure B-3. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized TPF vs. Maximum IC Score Threshold

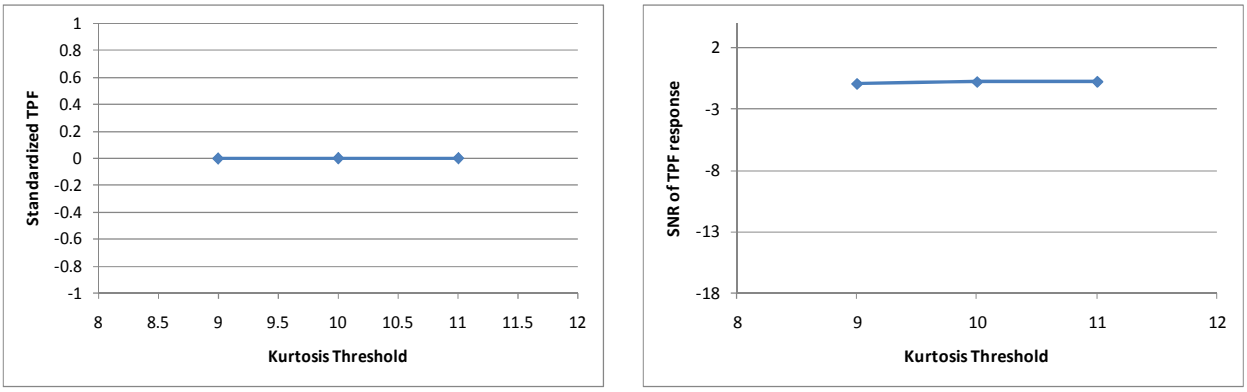


Figure B-4. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized TPF vs. Kurtosis Threshold

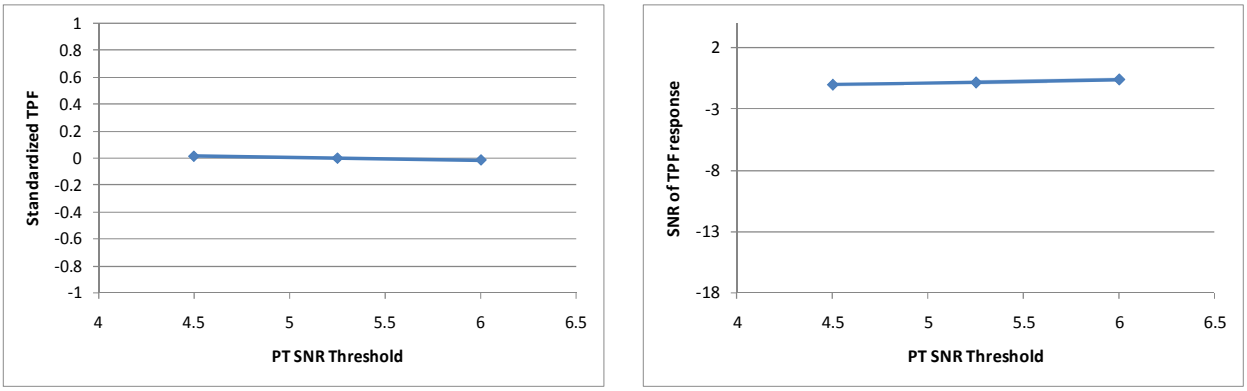


Figure B-5. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized TPF vs. PT SNR Threshold

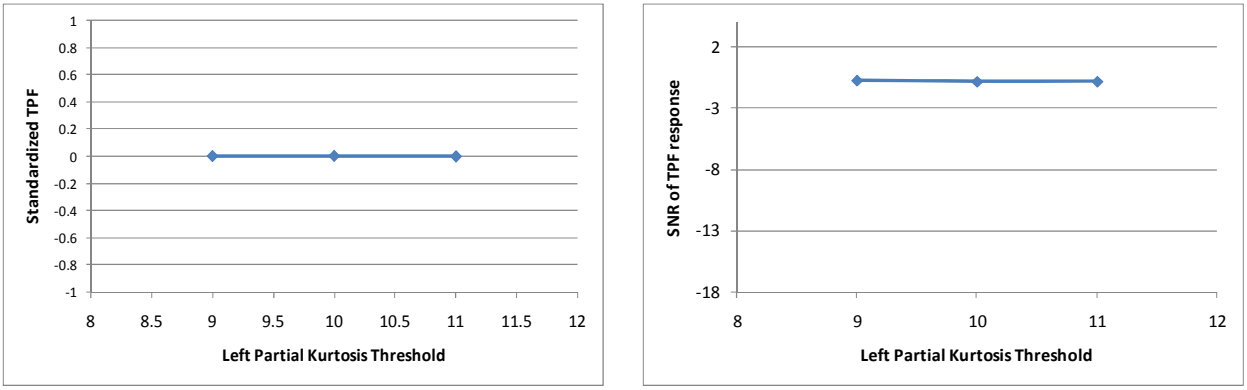


Figure B-6. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized TPF vs. Left Partial Kurtosis Threshold

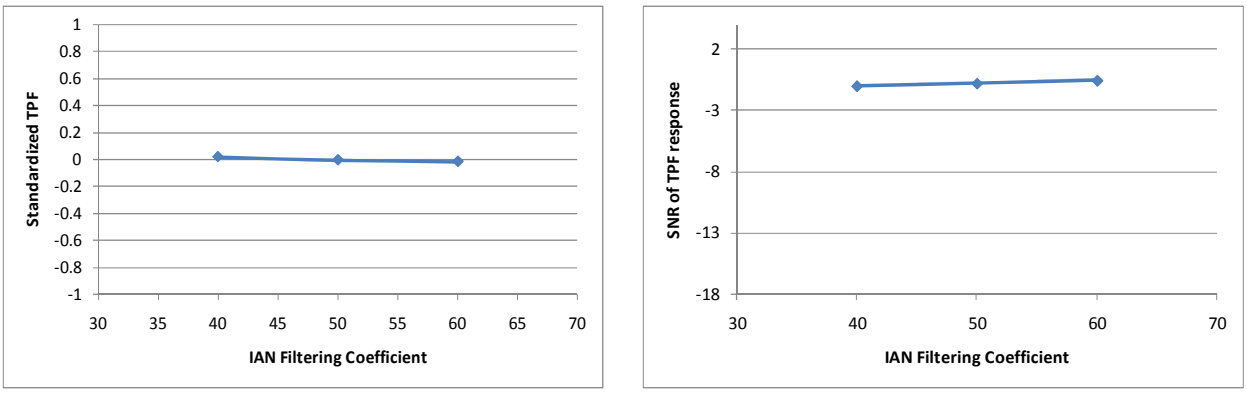


Figure B-7. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized TPF vs. Iterative Adaptive Noise Filtering Coefficient

B.2. False Positive Fraction (TPF) Response and SNR Plots

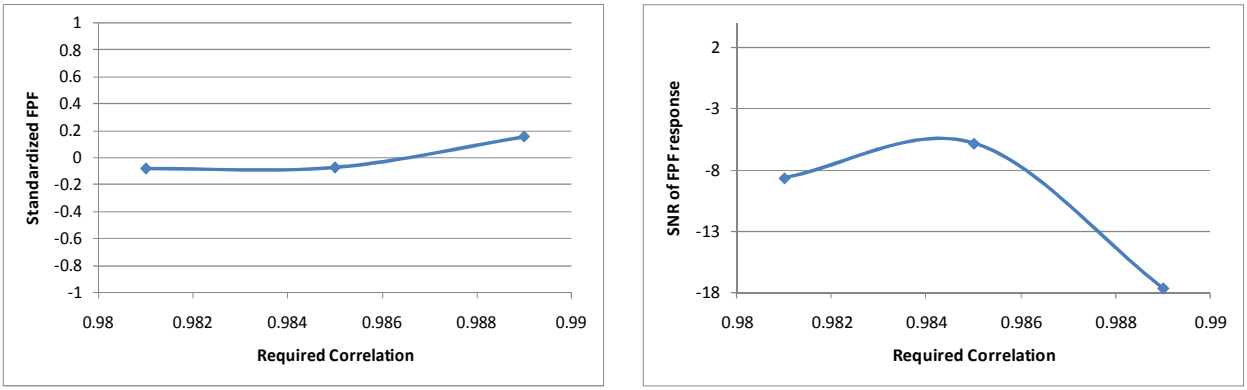


Figure B-8. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized FPF vs. Required Correlation Threshold

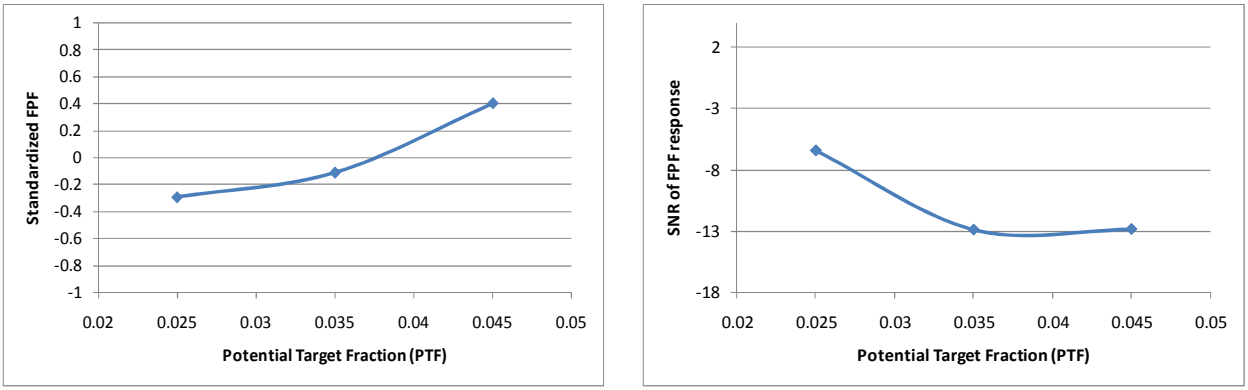


Figure B-9. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized FPF vs. Potential Target Fraction Threshold

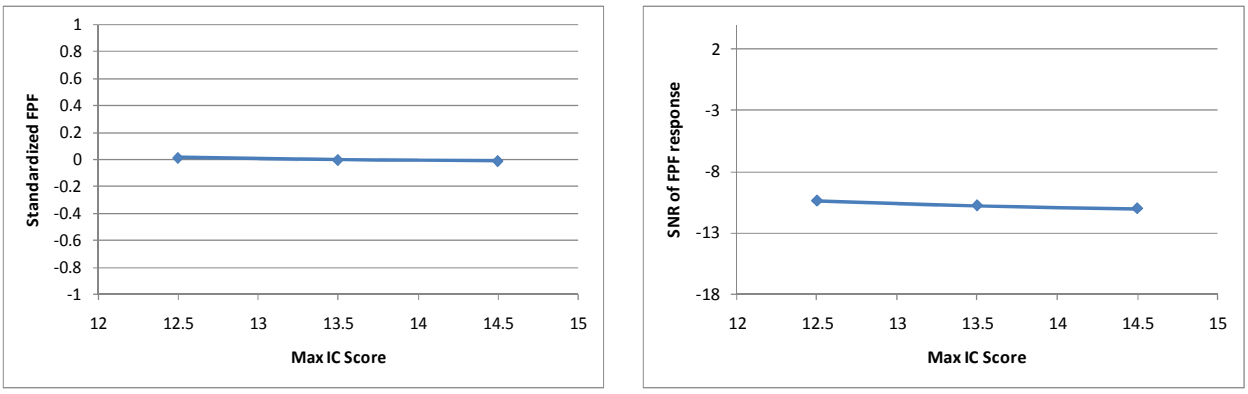


Figure B-10. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized FPF vs. Maximum IC Score Threshold

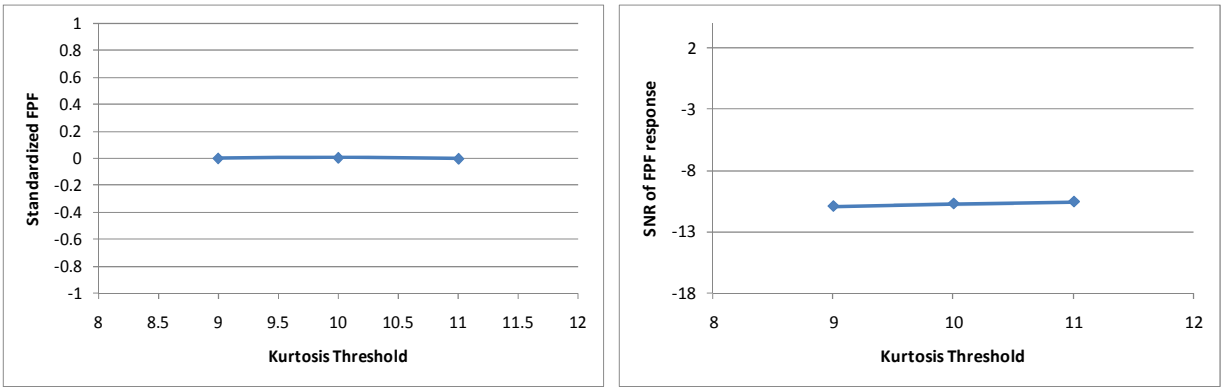


Figure B-11. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized FPF vs. Kurtosis Threshold

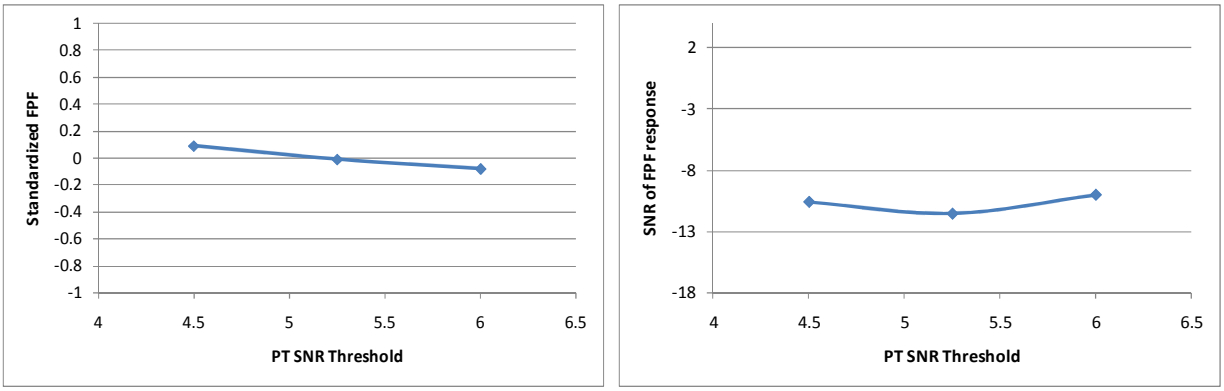
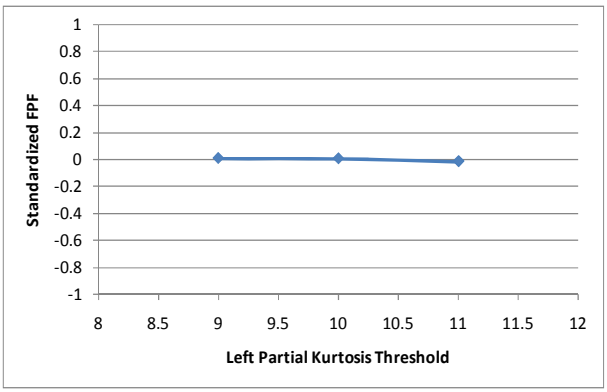


Figure B-12. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized FPF vs. PT SNR Threshold



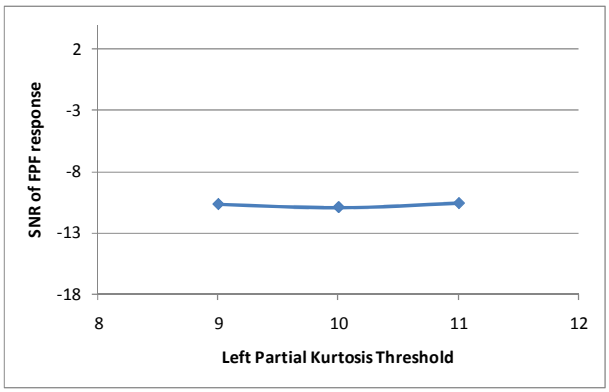
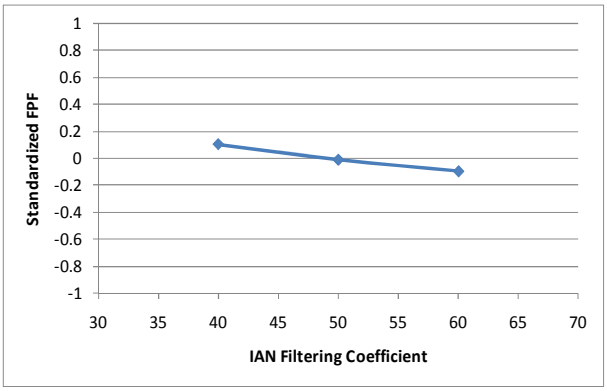


Figure B-13. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized FPF vs. Left Partial Kurtosis Threshold



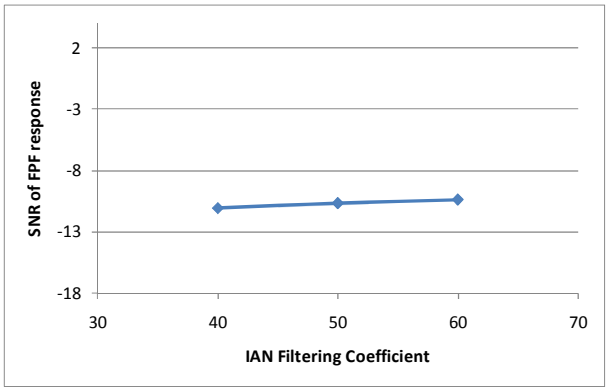
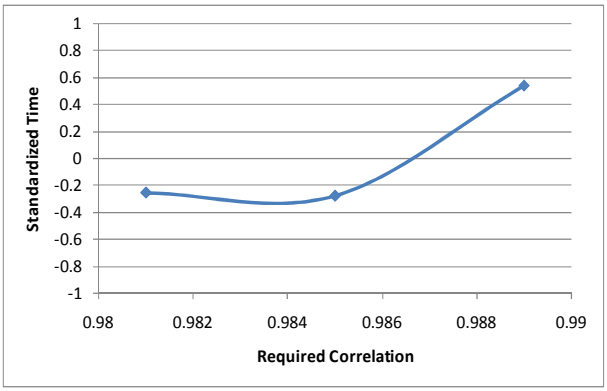


Figure B-14. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized FPF vs. Iterative Adaptive Noise Filtering Coefficient

B.3. Time Response and SNR Plots



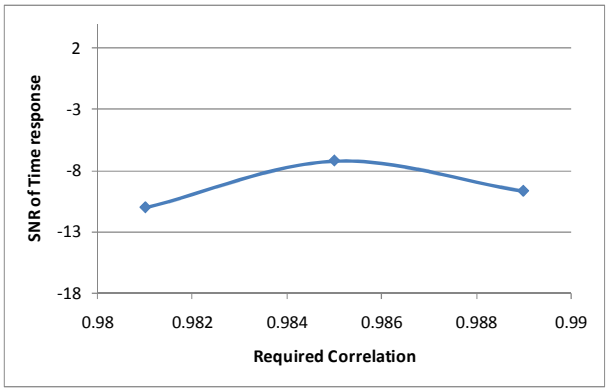


Figure B-15. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized Time vs. Required Correlation Threshold

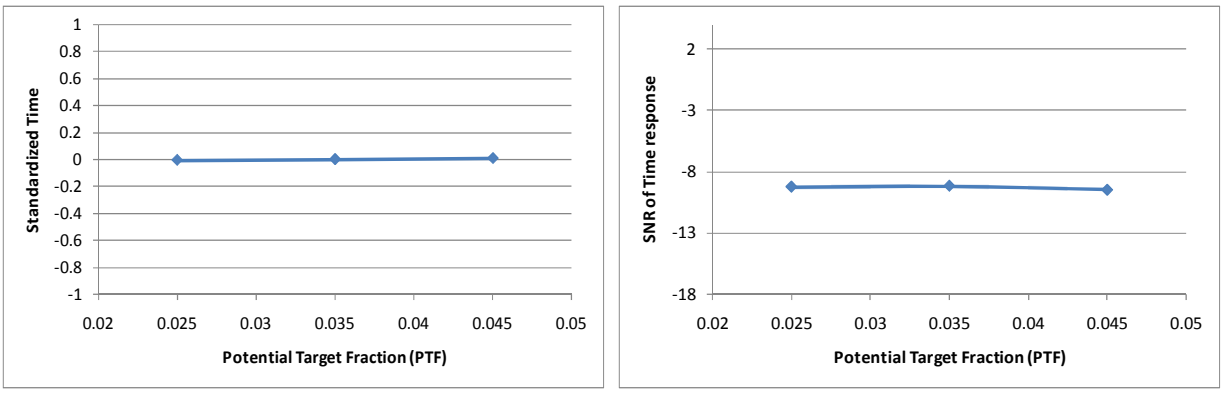


Figure B-16. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized Time vs. Potential Target Fraction Threshold

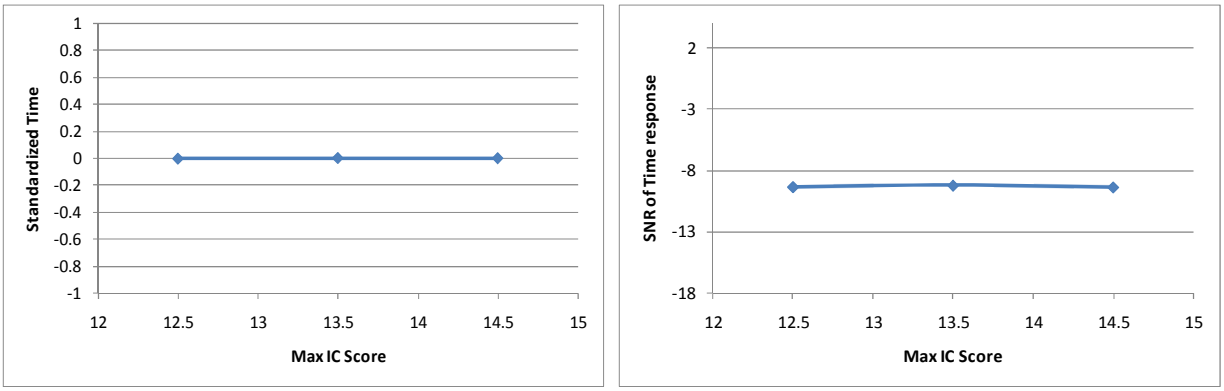


Figure B-17. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized Time vs. Maximum IC Score Threshold

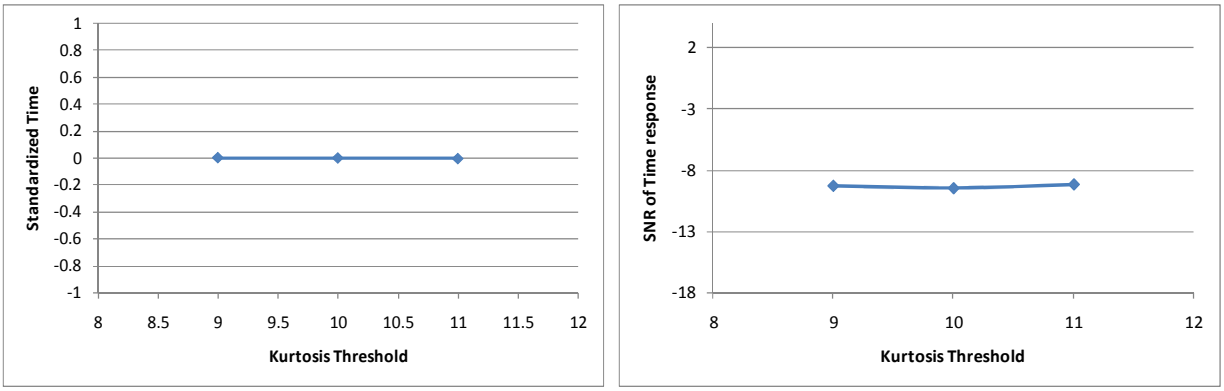


Figure B-17. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized Time vs. Kurtosis Threshold

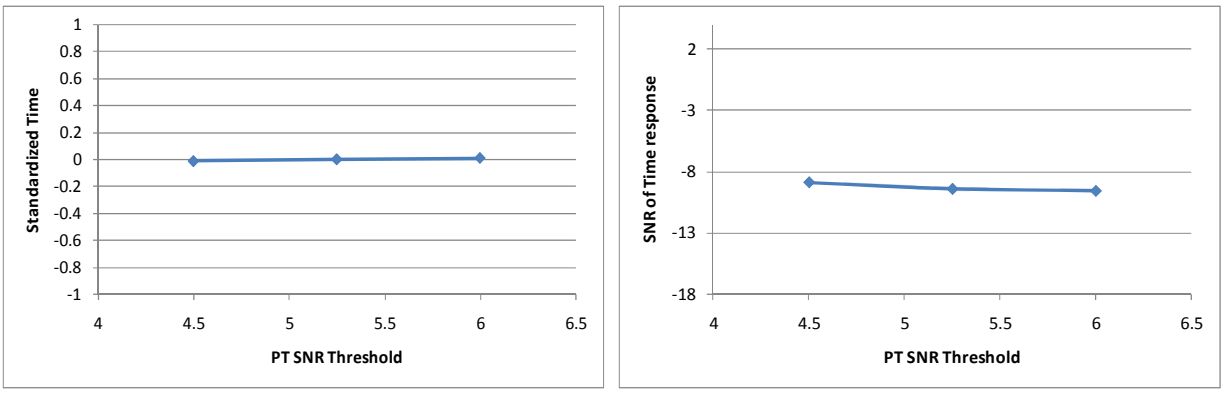


Figure B-18. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized Time vs. PT SNR Threshold

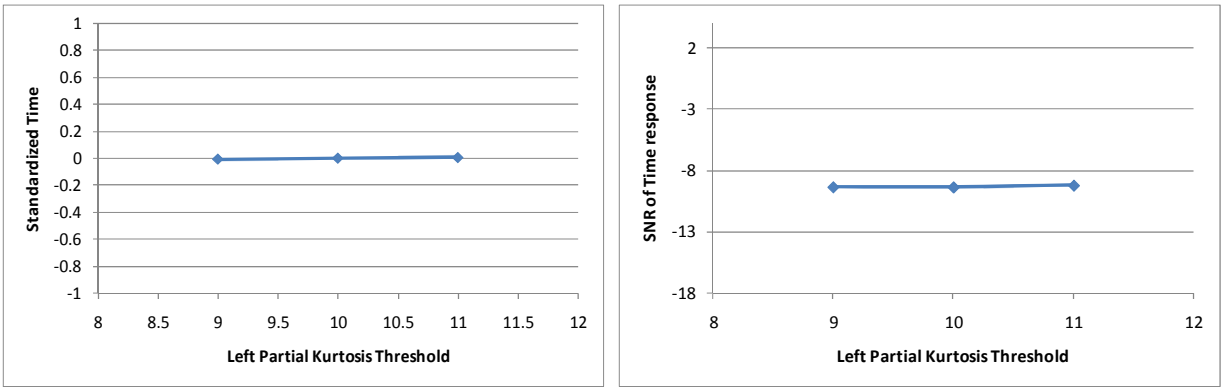


Figure B-19. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized Time vs. Left Partial Kurtosis Threshold

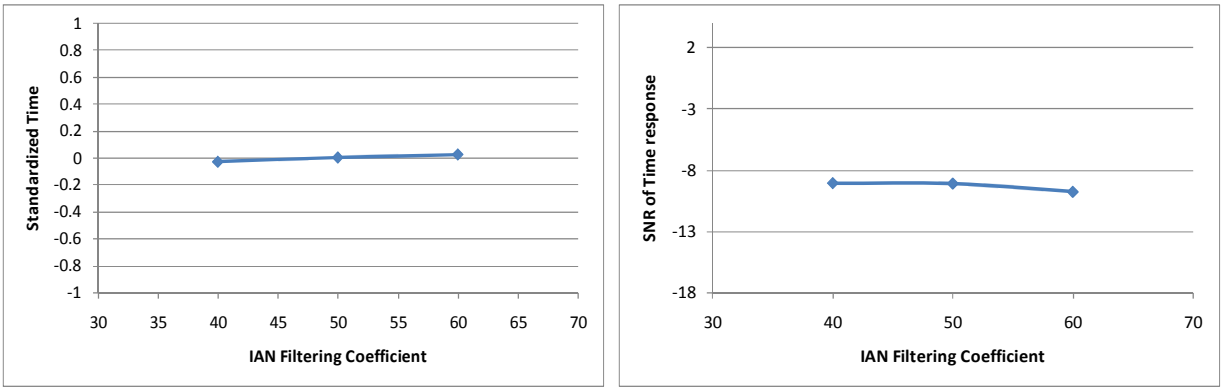


Figure B-16. Marginal Mean Plot (left) and Marginal SNR plot (right) Standardized Time vs. Iterative Adaptive Noise Filtering Coefficient

Blue Dart Submission Form

First Name: Michael Last Name: Miller

Rank (Military, AD, etc.): Major Designator # AFIT/BD/ENS/09-M

Student's Involved in Research for Blue Dart: Maj. Michael Miller, with Thesis Advisor: Dr. Kenneth Bauer

Position/Title: _____

Phone Number: DSN 785-3636 (X4591) E-mail: michael.miller@afit.edu and kenneth.bauer@afit.edu

School/Organization: Air Force Institute of Technology

Status: [X] Student [ ] Faculty [ ] Staff [ ] Other

Optimal Media Outlet (optional): _____

Optimal Time of Publication (optional): _____

General Category / Classification:

[ ] core values	[ ] command	[ ] strategy
[X] war on terror	[ ] culture & language	[ ] leadership & ethics
[X] warfighting	[ ] international security	[ ] doctrine

[X] other (specify): Intelligence Surveillance and Reconnaissance

Suggested Headline: Hyperspectral Imagery and Automatic Target Detection: 2020 Vision for Today's Air Force

Keywords: Hyperspectral Imagery, Unmanned Autonomous Arial Vehicle, Target Detection, Intelligence Surveillance and Reconnaissance

Field Level Autonomous Screening of Hyperspectral Imagery: Accelerating the ISR Process

In the midst of simultaneous wars in Iraq and Afghanistan the Department of Defense recognized a significant shortcoming in the Air Force's ability to provide sufficient Intelligence, Surveillance, and Reconnaissance (ISR) capabilities. Secretary of Defense Robert M. Gates spoke publicly on the subject of ISR at Maxwell AFB on 21 April 2008.

> *My concern is that our services are still not moving aggressively in wartime to provide resources needed now on the battlefield. . . While we have doubled this capability in recent months, it is still not good enough.*

While one aspect of the constraints relating to Air Force ISR support relates to the platforms and sensors providing the capability, the issue goes beyond the material aspects of ISR. By its very nature intelligence is a human endeavor, but the current ISR processes place too heavy a burden on the sensor operator and image analyst. Current processes rely on the analyst first to filter images containing nothing of interest from images which contain objects worthy of closer inspection. They must then attempt to identify these objects using in some cases a single photographic image, or when able by a comparison of sensors including electro-optical (EO), infrared (IR), and synthetic aperture radar (SAR). This dependence on the human to isolate likely target imagery from non-target imagery and then identify objects within images is unable to satisfy what Secretary Gates referred to as an *"insatiable appetite" for the information these systems provide*. There are however, emerging alternatives.

Hyperspectral Imagery (HSI) operates much as an ordinary digital camera in that it translates reflected light into pixel values representing some combination of the colors we recognize as red, green, and blue. When these pixels are correctly arranged into a grid they produce a single digital photograph. Hyperspectral imagery however operates beyond the human visual portion of the electromagnetic spectrum, by detecting reflected light spanning ultraviolet,

visual, and infrared bands. Furthermore, hyperspectral sensors divide light into as many as hundreds of "colors" as opposed to simply red, green, and blue. This effectively produces a stack of hundreds of images of the same target area, each based on how a narrow segment of sunlight is reflected off objects within the scene.

Hyperspectral imagery provides some advantages over traditional imaging techniques. Since hyperspectral imagery relies on reflected light, the sensor itself is passive, unlike SAR. Because the technique observes a target scene across a wider range of the electromagnetic spectrum than IR, it tends to be less degraded by atmospheric effects. Furthermore by dividing the spectrum into many narrow bands, hyperspectral techniques provide some inherent capability to defeat common camouflage, concealment, and deception (CCD) techniques which typically rely on masking objects in the visible portion of the spectrum. Finally, HSI provides the potential to identify targets based on the reflection properties or "spectral signatures" contained within the layers of the image.

The widespread application of HSI to UAV based remote sensing poses two challenges. First, by increasing spectral resolution of an image, larger file sizes demand greater bandwidth for transmission. Second, this increased spectral resolution means that each single image consists of hundreds of "layered" images, each providing an opportunity to detect a target within a narrow spectral band. This dramatically complicates the analysis burden placed on any system employing humans in the loop to detect and identify potential targets. Autonomous target detection algorithms employ statistical techniques to respond to both these challenges.

Operations Research specialists at the Air Force Institute of Technology have developed deployable algorithms which use the properties inherent in hyperspectral imagery to identify outliers likely to represent targets present in the scene. Most recently, algorithms consisting of

roughly 1000 lines of computer code have demonstrated an ability to identify in seconds upwards of 85% of all target pixels present in an image, with less than 1% of all not target pixels incorrectly identified as targets. Furthermore the latest effort, known as the Autonomous Global Anomaly Detection, Spectral Correlation (AutoGAD-SC) algorithm, correctly categorized all tested images into target or non-target classifications. Integration of this type of autonomous target detection algorithm along with hyperspectral imaging sensors precludes the requirement to relay each recorded image from airborne platform to imagery analysis facilities and eliminates the laborious process of manually filtering non-target images from target images. Instead initial screening reduces the set of images to include only those likely to contain actual targets.

Hyperspectral imaging sensors have reached a level of maturity, which enables their addition to the tools employed by ISR platforms. However, the sheer volume of information provided by HSI dictates some alternative to transmission of each image to a ground based unit for analysis by a human. AutoGAD-SC and other similar autonomous detection algorithms developed by AFIT provide a means to rapidly screen images, nominate likely targets, and provide a starting point for target identification.

(a) Bibliography

1. Brownlee, John. *The Warfighter's Edge: First Hyperspectral Images from Space.* Air Force Research Laboratory Public Affairs: Press Release 00-13**,** 2000.

2. Chang, Chein I. and Du, Qian. "Estimation of Number of Spectrally Distinct Signal Sources in Hyperspectral Imagery". *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 42, No. 3: 608-619, (March 2004).

3. Chang, Chein I. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification.* New York: Kluwer Academic/Plenum Publishers., 2003.

4. Chang, Chein I. *Hyperspectral Data Exploitation: Theory and Applications.* Hoboken, New Jersey: John Wiley and Sons., 2007.

5. Ding, C; He, X; Simon, HD. *On the equivalence of nonnegative matrix factorization and spectral clustering*. Proceedings of the 5th SIAM International Conference on Data Mining; April 2005; Newport Beach, Calif, USA. pp. 606–610.

6. Hyvärinen, Aapo and others. *Independent Component Analysis*. New York: John Wiley & Sons, Inc., 2001.

7. Hyvärinen, A. and Oja, E. *A Fast Fixed-Point Algorithm for Independent Component Analysis*. Neural Computation. 9:1483-1492, 1997.

8. Hoyer, Patrik O. *Non-negative Matrix Factorization with Sparseness Constraints*. Journal of Machine Learning Research. 5 (2004) 1457-1469.

9. Johnson, Robert. *Improved Feature Extraction, Feature Selection, and Identification Techniques that Create a Fast Unsupervised Hyperspectral Target Detection Algorithm.* MS thesis, AFIT/GOR/08-07. Department of Operational Sciences, Air Force Institute of Technology (AU), Wright Patterson AFB OH, March 2008.

10. Koo, Robert. *Feature Extraction Using Principal and Independent Component Analysis for Hyperspectral Imagery*. MS thesis, AFIT/GOR/07M-16. Department of Operational Sciences, Air Force Institute of Technology (AU), Wright Patterson AFB OH, May 2007.

11. Landgrebe, David A. *Signal Theory Methods in Multispectral Remote Sensing*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2003.

12. Miles, Donna. *Gates Reflects on Service as Wartime Defense Secretary*. American Forces Press Service. http://www.defenselink.mil/news/newsarticl.aspx?id=51642. 23 October 2008.

13. Myers, Raymond H. and Mongomery, Douglas C. *Response Surface Methodology.* Second Edition. New York, New York: John Wiley & Sons, Inc. 2002.

14. Smetek, Timothy E. *Hyperspectral Imagery Target Detection Using Improved Anomaly Detection and Signature Matching Methods.* Air Force Institute of Technology (AU), Wright Patterson AFB OH, May 2007.

15. Smith, Randall B. *Introduction to Hyperspectral Imaging with TNTmips ®.* MicroImages, Inc., 2006.

16. Stocker, Alan D., Eskandar Ensafi, and Clark Oliphint. "Applications of eigenvalue distribution theory to hyperspectral processing," *Proceedings of SPIE*, Vol. 5093:652-654, 657-664 (2003).

17. Stone, James V. *Independent Component Analysis-A Tutorial Edition.* Cambridge, Massechusetts. The MIT Press., 2004.

18. Wackerly, Dennis D. and others. *Mathematical Statistics with Applicatons*. 6th Edition. Pacific Grove, CA: Thomson Learning. 2002.

Vita

Major Michael K. Miller entered the Air Force in June of 1994 upon graduation from the United States Academy with a Bachelor of Science in Materials Science. He spent his first 18 months in the Air Force in a NASA internship program at Goddard Space Flight Center in Greenbelt, Maryland while earning a Master's of Science in Mechanical Engineering from George Washington University in 1996. Following completion of the Master's program at GWU he entered Undergraduate Pilot Training at Vance AFB, OK.

Upon graduation from pilot training he completed initial B-52H qualification and was assigned as a pilot to the 23d Bomb Squadron at Minot AFB, ND. In November 2000 he was assigned to the 49th Test and Evaluation Squadron at Barksdale AFB, LA. He spent the next five years conducting B-52 operational test, including integration of targeting pods on B-52s and demonstration and testing of datalinks enabling in-flight retargeting of Conventional Air Launched Cruise Missiles, and near real time non-traditional ISR from a B-52 equipped with targeting pods. In December 2003 he completed the B-52 Weapons Instructor Course where he was a Distinguished Graduate and was awarded the Best Paper Award for 2003B class.

In February 2006 he was assigned to Headquarters Air Force, Directorate of Programs, Combat Forces Division (AF/A8PC). There he was responsible for development of the Program Objective Memorandum for the B-1, B-2, and B-52. While assigned to the Pentagon he was deployed as an Air Planner to the Combined Joint Task Force, Horn of Africa in Djibuti. Upon graduation Mike has been assigned as the Operations Officer for the 340th Weapons Squadron at Barksdale AFB.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
03-26-2009	**Master's Thesis**	Sep 2008 - Mar 2009

4. TITLE AND SUBTITLE
EXPLOITATION OF INTRA-SPECTRAL BAND CORRELATION FOR RAPID FEACTURE SELECTION AND TARGET IDENTIFICATION IN HYPERSPECTRAL IMAGERY

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Miller, Michael, K., Major, USAF

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street, Building 642
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/09-10

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research extends the work produced by Capt. Robert Johnson for detecting target pixels within hyperspectral imagery (HSI). The methodology replaces Principle Components Analysis for dimensionality reduction with a clustering algorithm which seeks to associate spectral rather than spatial dimensions. By seeking similar spectral dimensions, the assumption of no a priori knowledge of the relationship between clustered members can be eliminated and clusters are formed by seeking high correlated adjacent spectral bands. Following dimensionality reduction Independent Components Analysis (ICA) is used to perform feature extraction. Kurtosis and Potential Target Fraction are added to Maximum Component Score and Potential Target Signal to Noise Ratio as mechanisms for discriminating between target and non-target maps. A new methodology exploiting Johnson's Maximum Distance Secant Line method replaces the first zero bin method for identifying the breakpoint between signal and noise. A parameter known as Left Partial Kurtosis is defined and applied to determine when target pixels are likely to be found in the left tail of each signal histogram. A variable control over the number of iterations of Adaptive Iterative Noise filtering is introduced. Results of this modified algorithm are compared to those of Johnson's AutoGAD [2007].

15. SUBJECT TERMS
Hyperspectral Imagery, spectral correlation, clustering, blind signal separation, unsupervised target detection, Independent Components Analysis, kurtosis, Left Partial Kurtosis, Right Partial Kurtosis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth W. Bauer (ENS)
U	U	U	UU	190	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4328; e-mail: kenneth.bauer@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18