

2006

## Analysis of hyperspectral change and target detection as affected by vegetation and illumination variations

Joseph Meola  
*University of Dayton*

Follow this and additional works at: [https://ecommons.udayton.edu/graduate\\_theses](https://ecommons.udayton.edu/graduate_theses)

---

### Recommended Citation

Meola, Joseph, "Analysis of hyperspectral change and target detection as affected by vegetation and illumination variations" (2006). *Graduate Theses and Dissertations*. 4381.  
[https://ecommons.udayton.edu/graduate\\_theses/4381](https://ecommons.udayton.edu/graduate_theses/4381)

This Thesis is brought to you for free and open access by the Theses and Dissertations at eCommons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of eCommons. For more information, please contact [mschlangen1@udayton.edu](mailto:mschlangen1@udayton.edu), [ecommons@udayton.edu](mailto:ecommons@udayton.edu).

**Analysis of Hyperspectral Change and Target Detection as  
Affected by Vegetation and Illumination Variations**

Thesis

Submitted To

School of Engineering

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree

Master of Science in Electrical Engineering

by

Joseph Meola

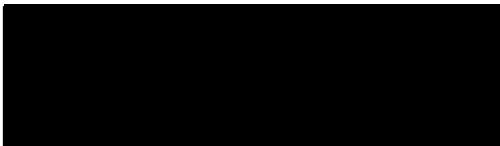
UNIVERSITY OF DAYTON

Dayton, Ohio

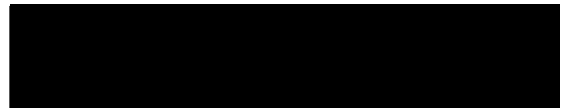
August 2006

**Analysis of Hyperspectral Change and Target Detection as  
Affected by Vegetation and Illumination Variations**

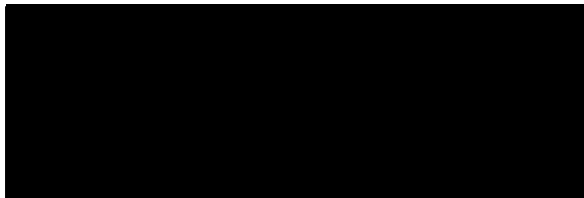
APPROVED BY:



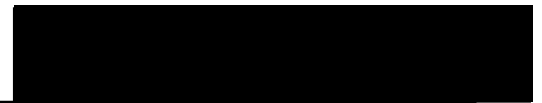
Russell Hardie, Ph.D.  
Advisory Committee Chairman  
Professor Electrical Engineering  
University of Dayton



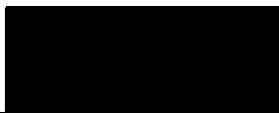
Michael Eismann, Ph.D.  
Committee Member  
Research Engineer  
Air Force Research Lab



Raúl Ordóñez, Ph.D.  
Committee Member  
Professor Electrical Engineering  
University of Dayton



Kenneth J. Barnard, Ph.D.  
Committee Member  
Research Engineer  
Air Force Research Lab



Donald L. Moon, Ph.D.  
Associate Dean  
Graduate Engineering Programs & Research  
School of Engineering



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

## **ABSTRACT**

### **Analysis of Hyperspectral Change and Target Detection as Affected by Vegetation and Illumination Variations**

Joseph Meola  
University of Dayton, 2006

Advisor: Dr. Russell C. Hardie

This paper covers the use of detection algorithms on hyperspectral data of a scene collected over several seasons. The effectiveness of specific target detection algorithms on scenes with different illumination conditions such as shadows, low sun angles, and seasonal vegetation changes is examined. Various algorithms are currently employed to detect anomalies within a scene. Specifically, a simple Mahalanobis distance measure and spectral matched filter algorithm are often employed on hyperspectral data to detect specific objects within a scene. These algorithms are limited to use on a single scene acquired at a specific time. When hyperspectral data from the same scene at different times is available, more sophisticated algorithms utilizing linear predictors such as chronochrome and covariance equalization are used for detection.

Using a push-broom style imaging spectrometer mounted on a pan and tilt, visible to near infrared data of a scene containing specific objects is gathered. Hyperspectral system characterization and calibration is performed to ensure viable data is produced. Data collection occurs from late August 2005 until May 2006 to obtain a wide range of illumination and vegetation conditions. After collecting and calibrating the

scene data, the change detection algorithms are employed to assess background suppression for various conditions. These scene conditions can include daily illumination change, seasonal illumination change, and seasonal vegetation change. Often, the chronochrome and covariance equalization predictors work well for certain portions of the scene but struggle in others. This result demonstrates how areas of a scene may undergo different temporal transformations. In addition, comparing the change detection algorithms to single instance methods allows for the assessment of any advantages offered by a specific method. Using a signal to clutter ratio, the various algorithms are evaluated and conclusions are drawn.

## ACKNOWLEDGMENTS

I would like to take this time to thank the numerous people who have made the completion of this thesis possible. First, I would like to thank my advisor, Russell Hardie for providing guidance to my research and offering suggestions and assistance for the numerous problems I encountered. I would also like to thank Ken Barnard and Mike Eismann at AFRL for serving on my committee and assisting my research. I cannot count the number of times I went to them with questions concerning a wide range of topics. Each time, they stopped whatever they were working on to fully answer my question. My thanks go out to Raúl Ordóñez as well for serving on my committee.

For all the help in answering questions and obtaining equipment, I also thank Phil Maciejewski. I am indebted to Jeremiah Flerchinger for the endless hours he spent writing code and helping with the setup and calibration of the instrument. In general, I would like to thank all the guys at AFRL SNJT. Not only do they all set aside time to answer my questions, they are also one of the greatest bunch of people I have encountered to work with. I can always count on them for a laugh, even when I am stressed.

From the University of Dayton, Loretta Christon and Marilyn Knisley have provided endless hours of assistance. Both seem to have the answer to every procedural question I ask. Thank you both for your kindness and willingness to help. Thank you to fellow graduate student Pat Hytla for helping with data collection. Thanks to all my friends both in Dayton and back home who have provided moral support throughout graduate school.

Last but certainly not least, I would like to thank my family. My sisters have provided encouragement and support at all times during my schooling. My parents have done so much for me in life. I cannot begin to thank them enough. For the endless sacrifices they both have made to put my sisters and I through school, I thank them. I do not know where I would be today if I did not have the parents I do. I thank God for both of them every day. I hope they are as proud of me as I am of them.

## TABLE OF CONTENTS

<b>APPROVAL PAGE</b> .....	ii
<b>ABSTRACT</b> .....	iii
<b>ACKNOWLEDGMENTS</b> .....	v
<b>TABLE OF CONTENTS</b> .....	vi
<b>LIST OF FIGURES</b> .....	viii
<b>LIST OF TABLES</b> .....	xvi
<b>CHAPTER 1: Introduction</b> .....	1
1.1 Hyperspectral Concept.....	1
1.2 Research Goals.....	4
<b>CHAPTER 2: Hyperspectral Data Collection</b> .....	5
2.1 Instrument Specifications.....	5
2.2 Spectral Calibration.....	11
2.3 Noise Characterization.....	16
2.4 Radiometric Calibration.....	26
2.5 Second Order Correction.....	30
2.6 Non-linearity Characterization.....	34
2.7 Data Collection.....	36
<b>CHAPTER 3: Hyperspectral Target and Change Detection Theory</b> .....	46
3.1 Hyperspectral Data Modeling.....	46
3.2 Target Detection Theory.....	48
3.3 Change Detection Theory.....	52
3.4 Atmospheric Compensation.....	55
3.5 Principal Component Transformation.....	65
<b>CHAPTER 4: Experimental Results</b> .....	70

4.1 Experimental Testing Methodology.....	70
4.2 Illumination Change Background Suppression.....	74
4.3 Seasonal Change Background Suppression.....	81
4.4 Illumination Change Signal to Clutter Ratio.....	88
4.5 Seasonal Change Signal to Clutter Ratio.....	92
<b>CHAPTER 5: Conclusion.....</b>	<b>96</b>
5.1 Summary.....	96
5.2 Future Considerations.....	101
<b>APPENDIX A: Scene Color Images.....</b>	<b>103</b>
A.1 Reference Images.....	103
A.2 Illumination Change Images.....	104
A.3 Seasonal Change Images.....	106
<b>APPENDIX B: MATLAB Code.....</b>	<b>112</b>
B.1 Calibration Code.....	112
B.2 Data Conversion Code.....	127
B.3 Data Extraction Code.....	133
B.4 Detection Code.....	138
B.5 Statistics Code.....	143
<b>REFERENCES.....</b>	<b>154</b>



## LIST OF FIGURES

1.1.1	Electromagnetic Spectrum.....	1
1.1.2	Basic HSI System Setup.....	2
1.1.3	Hyperspectral Data Concept.....	3
2.1.1	Imaging Spectrometer.....	5
2.1.2	Focal Plane of Camera.....	7
2.1.3	FPA Responsivity.....	8
2.1.4	Pan and Tilt.....	9
2.2.1	HeNe Laser Output.....	12
2.2.2	Output Spectrum for Ar Lamp.....	13
2.2.3	Output Spectrum for Kr Lamp.....	13
2.2.4	Output Spectrum for HgNe Lamp.....	14
2.2.5	Ar Lamp Camera Output.....	14
2.2.6	Kr Lamp Camera Output.....	14
2.2.7	HgNe Lamp Camera Output.....	15
2.3.1	Fixed Pattern Noise.....	17
2.3.2	FPN Removed.....	17
2.3.3	Row Noise.....	20
2.3.4	SNR Threshold for Focal Plane.....	22
2.3.5	SNR Across Row.....	22

2.3.6	SNR $\geq$ 18 for Limited Spectral and Row Range.....	23
2.3.7	Bin Frequency Response.....	24
2.3.8	Filter Frequency Responses.....	24
2.4.1	Calibrated Integrating Sphere Data.....	27
2.5.1	Shortpass Filter Response.....	31
2.5.2	Second Order Effects in Focal Plane.....	31
2.5.3	Second Order Effects Across Row.....	31
2.6.1	Non-linear Error at 564fL.....	35
2.6.2	Non-linear Error at 203fL.....	35
2.6.3	Non-linear Error at 15fL.....	36
2.7.1	Scene Setup.....	38
2.7.2	Overhead Camera and Panel Locations.....	38
2.7.3	Seasonal Solar Trajectory (June – December).....	40
2.7.4	Seasonal Solar Trajectory (December – June).....	41
2.7.5	SE 30°, SA 106° September 2, 2005.....	41
2.7.6	SE 58°, SA 180° September 2, 2005.....	41
2.7.7	SE 30°, SA 106° September 2, 2005.....	42
2.7.8	SE 30°, SA 120° September 30, 2005.....	42
2.7.9	SE 61°, SA 180° August 25, 2005.....	43
2.7.10	SE 35°, SA 180° November 3, 2005.....	43
2.7.11	SE 50°, SA 177° September 22, 2005.....	44
2.7.12	SE 45°, SA 180° March 7, 2006.....	44
2.7.13	Panel Tilt Change.....	45

2.7.14	Tarp Change.....	45
2.7.15	Panels Removed.....	45
3.2.1	Segmented Scene Image.....	50
3.2.2	Scatter Plot with Ellipsoid Decision Surface.....	50
3.3.1	Change Detection Process.....	53
3.4.1	NDVI Theory.....	58
3.4.2	Varying Threshold Levels.....	58
3.4.3	NDVI with IR Threshold.....	59
3.4.4	Vegetation Scene Pixels.....	59
3.4.5	Lab Reflectance Spectra.....	60
3.4.6	Scene Spectral Radiance Spectra.....	60
3.4.7	Well Illuminated Scene (1).....	61
3.4.8	Mean Vegetation and Shade Spectra for (1).....	61
3.4.9	AC for Silver Panel using Shade for (1).....	61
3.4.10	AC Error using Shade for (1).....	61
3.4.11	Scene with Illumination Change (2).....	62
3.4.12	Mean Vegetation and Shade for (2).....	62
3.4.13	AC for Silver Panel using Shade for (2).....	63
3.4.14	AC Error using Shade for (2).....	63
3.4.15	Scene with Seasonal Changes (3).....	64
3.4.16	Mean Vegetation and Shade for (3).....	64
3.4.17	AC for Silver Panel using Shade for (3).....	64
3.4.18	AC Error using Shade for (3).....	64

3.5.1	PC Transform Concept.....	66
3.5.2	Eigenvalues for August 25, 2005 SA 180° Covariance.....	67
3.5.3	PC Band 1.....	68
3.5.4	PC Band 2.....	68
3.5.5	PC Band 6.....	68
3.5.6	PC Band 10.....	68
4.1.1	Regions of Interest.....	72
4.2.1	Illumination Change Region Total Variance.....	75
4.2.2	Illumination Background Suppression for Whole Scene using Whole Scene to Compute Transform.....	76
4.2.3	Illumination Background Suppression for Grass using Whole Scene to Compute Transform.....	76
4.2.4	Illumination Background Suppression for Trees using Whole Scene to Compute Transform.....	77
4.2.5	Illumination Background Suppression for Panels using Whole Scene to Compute Transform.....	77
4.2.6	Illumination Background Suppression Summary using Whole Scene to Compute Transform.....	78
4.2.7	Illumination Background Suppression for Whole Scene using Whole Scene to Compute Transform.....	79
4.2.8	Illumination Background Suppression for Grass using Grass to Compute Transform.....	79
4.2.9	Illumination Background Suppression for Trees using Trees to Compute Transform.....	80
4.2.10	Illumination Background Suppression for Panels using Panels to Compute Transform.....	80
4.2.11	Illumination Background Suppression by Region Summary.....	80
4.3.1	Seasonal Change Region Total Variance.....	82

4.3.2	Seasonal Change Background Suppression for Whole Scene using Whole Scene to Compute Transform.....	83
4.3.3	Seasonal Change Background Suppression for Grass using Whole Scene to Compute Transform.....	83
4.3.4	Seasonal Change Background Suppression for Trees using Whole Scene to Compute Transform.....	84
4.3.5	Seasonal Change Background Suppression for Panels using Whole Scene to Compute Transform.....	84
4.3.6	Seasonal Change Background Suppression Summary using Whole Scene to Compute Transform.....	85
4.3.7	Seasonal Change Background Suppression for Whole Scene using Whole Scene to Compute Transform.....	85
4.3.8	Seasonal Change Background Suppression for Grass using Grass to Compute Transform.....	85
4.3.9	Seasonal Change Background Suppression for Trees using Trees to Compute Transform.....	86
4.3.10	Seasonal Change Background Suppression for Panels using Panels to Compute Transform.....	86
4.3.11	Seasonal Change Background Suppression by Region Summary.....	87
4.4.1	Illumination Change Background Suppression for Natural Scene.....	89
4.4.2	Illumination Change SCR for Beige Panel.....	90
4.4.3	Illumination Change SCR for Green Panel.....	90
4.4.5	Illumination Change SCR for SMF on Beige Panel using Different Methods.....	91
4.4.6	Illumination Change SCR for SMF on Green Panel using Different Methods.....	91
4.5.1	Seasonal Change Background Suppression for Natural Scene.....	92
4.5.2	Seasonal Change SCR for Beige Panel.....	93

4.5.3	Seasonal Change SCR for Green Panel.....	93
4.5.5	Seasonal Change SCR for SMF on Beige Panel using Different Methods.....	94
4.5.6	Seasonal Change SCR for SMF on Green Panel using Different Methods.....	94
A.1.1	May 5, 2006 SA 120° Illumination Background Suppression Reference.....	103
A.1.2	August 25, 2005 SA 180° Seasonal Background Suppression Reference.....	103
A.1.3	May 23, 2006 SA 180° SCR Reference.....	104
A.2.1	May 8, 2006 8:00.....	104
A.2.2	May 8, 2006 9:00.....	104
A.2.3	May 8, 2006 10:00.....	105
A.2.4	May 8, 2006 11:00.....	105
A.2.5	May 8, 2006 12:00.....	105
A.2.6	May 9, 2006 13:00.....	105
A.2.7	May 9, 2006 14:00.....	105
A.2.8	May 9, 2006 15:00.....	105
A.2.9	May 20, 2006 16:00.....	105
A.2.10	May 20, 2006 17:00.....	106
A.2.11	May 22, 2006 18:00.....	106
A.2.12	May 22, 2006 19:00.....	106
A.3.1	August 24, 2005 SA 180°.....	106
A.3.2	September 1, 2005 SA 180°.....	106
A.3.3	September 2, 2005 SA 180°.....	107
A.3.4	September 6, 2005 SA 180°.....	107
A.3.5	September 7, 2005 SA 180°.....	107

A.3.6	September 12, 2005 SA 180°	107
A.3.7	September 22, 2005 SA 180°	107
A.3.8	September 27, 2005 SA 180°	107
A.3.9	September 30, 2005 SA 180°	108
A.3.10	October 4, 2005 SA 180°	108
A.3.11	October 6, 2005 SA 180°	108
A.3.12	October 14, 2005 SA 180°	108
A.3.13	October 15, 2005 SA 180°	108
A.3.14	October 17, 2005 SA 180°	108
A.3.15	October 18, 2005 SA 180°	109
A.3.16	October 26, 2005 SA 180°	109
A.3.17	November 2, 2005 SA 180°	109
A.3.18	November 3, 2005 SA 180°	109
A.3.19	November 10, 2005 SA 180°	109
A.3.20	January 12, 2006 SA 180°	109
A.3.21	January 23, 2006 SA 180°	110
A.3.22	February 23, 2006 SA 180°	110
A.3.23	February 24, 2006 SA 180°	110
A.3.24	February 28, 2006 SA 180°	110
A.3.25	March 2, 2006 SA 180°	110
A.3.26	March 7, 2006 SA 180°	110
A.3.27	March 27, 2006 SA 180°	111
A.3.28	April 10, 2006 SA 180°	111

A.3.29 April 11, 2006 SA 180° .....	111
A.3.30 April 18, 2006 SA 180° .....	111
A.3.31 May 9, 2006 SA 180° .....	111



**LIST OF TABLES**

2.1.1 Rotary Stage Specifications from Aerotech.....10

2.7.1 Data Specifications.....37

## CHAPTER 1

### Introduction

An overview of the research topic and technology used is given here. The goals and experimental procedure for the research are discussed as well.

#### 1.1 Hyperspectral Concept

Hyperspectral imaging (HSI) is part of the more general field of remote sensing. This field involves gathering information about an object or material within a scene without directly coming into contact with it.<sup>1</sup> A hyperspectral imager accomplishes this task by gathering reflected or emitted radiation from an object over a given wavelength range. For this hyperspectral system, the wavelength range resides in the visible to near infrared (VNIR) portion of the spectrum where reflected radiation dominates as demonstrated in Figure 1.1.1.

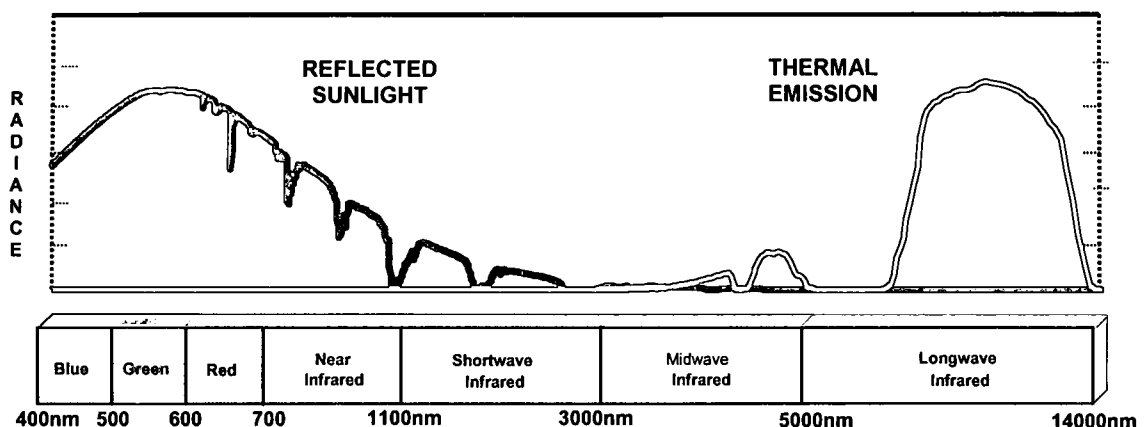
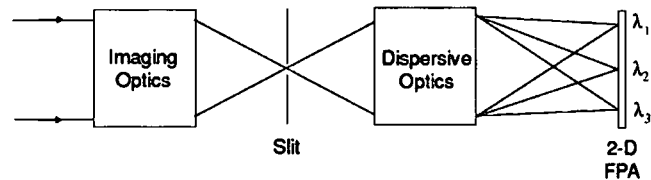


Figure 1.1.1: Electromagnetic Spectrum

Radiation reflected at a particular wavelength depends upon the molecular composition of the object being observed and theoretically can be used to identify the object. For example, grass appears green to the human eye because it reflects green wavelengths (~560nm) of light more than others in the visible region. If an imager can collect data at a number of different wavelengths over a broad range, one can more accurately identify objects using the acquired “spectral signature”.<sup>1</sup>

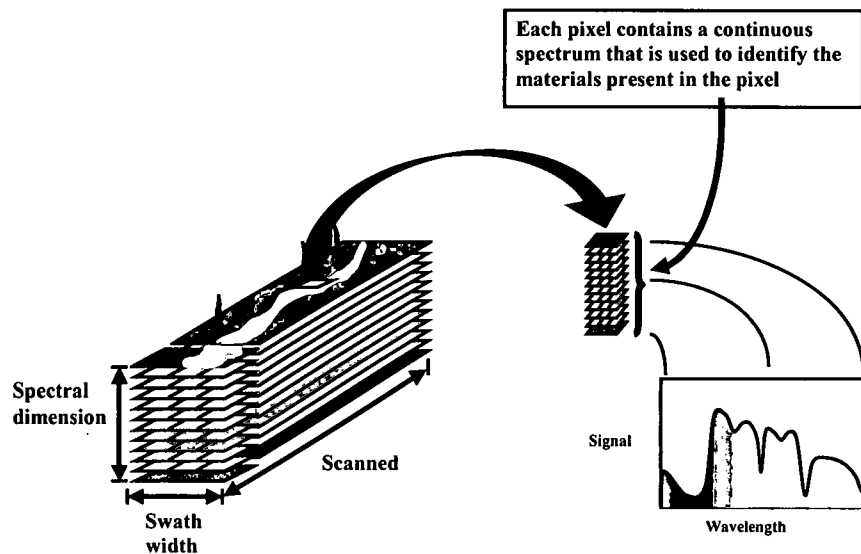
Data for this research is collected using a hyperspectral imaging spectrometer.

Figure 1.1.2 demonstrates a simplistic imaging spectrometer setup.



**Figure 1.1.2: Basic HSI System Setup**

The imaging optic, usually a lens, focuses light onto the slit where it is eventually dispersed using either a prism or diffraction grating. Dispersed light falls onto a focal plane where the information is saved. Consequently, each frame of data constitutes a vertical slice of a scene at many different wavelengths of light. Each column on the focal plane acquires the same spatial portion of the scene at a different wavelength. A hyperspectral data cube is constructed by collecting many frames while scanning the instrument across a scene as demonstrated in Figure 1.1.3.



**Figure 1.1.3: Hyperspectral Data Concept**

Typically, an imaging spectrometer resides upon a moving platform, such as an airplane, that provides the scanning motion for collecting multiple frames. In this case, the instrument rests on a pan and tilt powered by two rotary stage motors to perform scene scanning. Within the hyperspectral data cube, each pixel in the spectral dimension or “hyperpixel” provides the acquired spectral signature of an object. The quality and success of each collection of hyperspectral data depends strongly upon the type of equipment used. From actual data collection through calibration, numerous pieces of equipment are used. Chapter 2 discusses the calibration techniques and results for the hyperspectral data collected. In addition, the specifications of the various instruments used are given.

Hyperspectral imaging is typically used on aircraft or satellite to accurately identify specific objects over vast areas of land. The specific objects of interest, or “targets”, may represent enemy tanks for military applications or specific types of vegetation for agricultural applications. Whatever the application, the goal is to identify all the targets within a scene while limiting the number of false detections or “false

alarms”. Chapter 3 discusses the theory behind hyperspectral target detection and specific techniques such as spectral matched filtering, Mahalanobis distance, and change detection using chronochrome and covariance equalization. The latter two methods utilize hyperspectral data from an earlier time to try and identify targets or changes in the same scene at a later time.

## **1.2 Research Goals**

This research attempts to assess the effects of illumination and seasonal vegetation changes to a specific scene on the ability to detect targets. The hyperspectral imager used for this study is mounted on a pan and tilt and placed in a tower several stories above ground level. Data collection software is created to ensure the imager scans the same expanse of land time after time with near perfect registration. Using this method, the same scene is observed numerous times over a period between August 2005 and May 2006. The observed scene possesses grass, trees, and aluminum panels. Chapter 2 discusses specific data collection methodology for testing scene illumination changes and seasonal changes as well. Change detection algorithms attempt to suppress background to focus on anomalous changes to the scene. Consequently, background suppression using chronochrome and covariance equalization prediction methods is investigated for the illumination and seasonal change cases. A signal to clutter ratio is used to assess the effectiveness of all the algorithms for target detection in different cases. Chapter 4 discusses the metrics for background suppression and signal to clutter ratio as well as the experimental results for the data gathered. Chapter 5 summarizes the research process and results and provides considerations for future work.

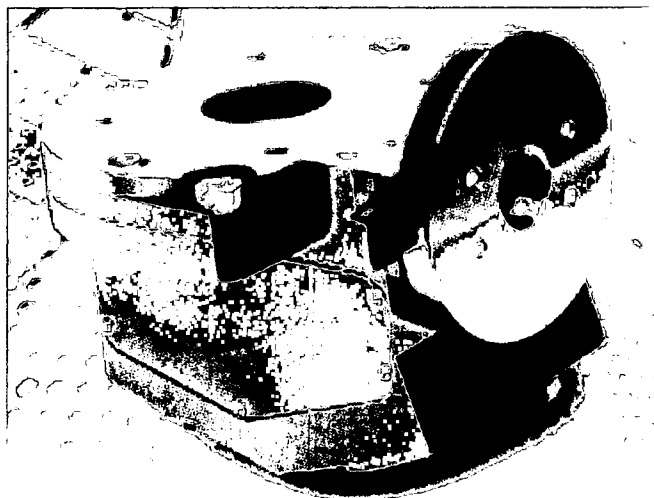
## **CHAPTER 2**

### **Hyperspectral Data Collection**

The process of collecting hyperspectral data involves a number of steps. For this research, an imager mounted on a pan and tilt gathers the data. In order to produce accurate and reliable data, a spectral calibration, noise characterization, and radiometric calibration must be performed. In addition, a specific plan is developed to produce data capable of use for evaluating various illumination and vegetation changes. Each step in the collection process and the instruments involved are discussed here.

#### **2.1 Instrument Specifications**

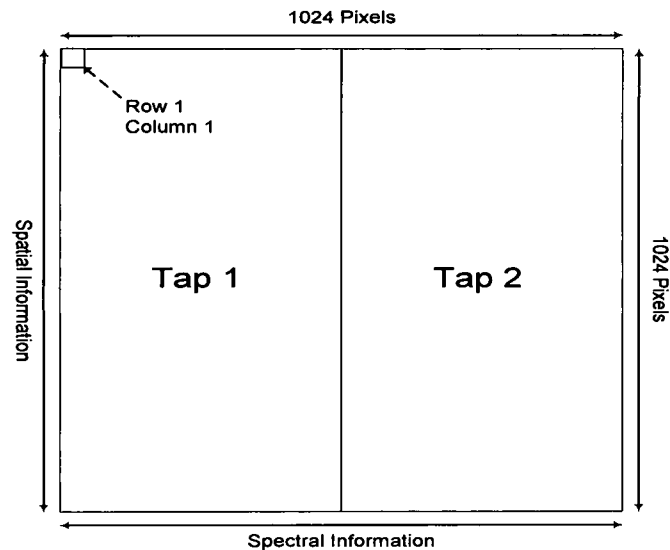
The dispersion instrument used for data collection is a Hyperspec VS-25 Imaging Spectrograph produced by Headwall Photonics displayed in Figure 2.1.1.<sup>2</sup>



**Figure 2.1.1: Imaging Spectrometer**

The VS-25 has a spectral range of 400nm – 1000nm over 6mm dispersion using a holographic diffraction grating. Using a 12 $\mu$ m slit width, the approximate full-width-half-maximum (FWHM) spectral resolution of the system is 2nm according to the spec sheet. The slit lies vertically when scanning resulting in horizontal spectral information and vertical spatial information. Keystone and smile distortion are less than 0.1% with the system designed at f/2. A Navitar 50mm lens (DO-5018) is attached to the C-mount just in front of the slit. This lens is set at f/4 for data collection purposes the help limit vignetting. With focal plane elements the same size as the slit width, the resulting instantaneous field of view (IFOV) for the system is 0.24mrad.

The camera used in conjunction with the spectrograph is a Dalsa Pantera TF 1M60 monochrome area scan CCD.<sup>3</sup> Light dispersed by the spectrograph falls onto the camera focal plane for capturing. The square focal plane array (FPA) contains 1024 x 1024 square pixels, each having 12 $\mu$ m side lengths with a 12 $\mu$ m pitch. This pixel size results in a square FPA with a side length of 12.288mm. The FPA is divided into two taps, a right and left half, transferring data at a rate of 40MHz. With two taps, the camera is capable of obtaining and transferring data at a maximum rate of 60 frames per second (fps) with a minimum exposure time of 15.685 $\mu$ s and up to 12-bit data precision. Figure 2.1.2 displays the basic orientation of the FPA.

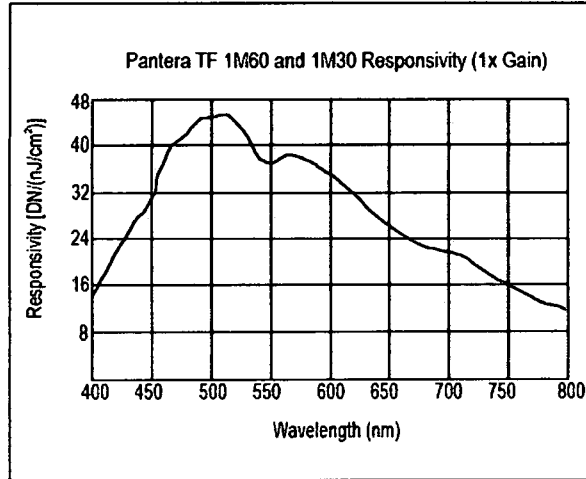


**Figure 2.1.2: Focal Plane of Camera**

The spectrograph disperses light across the focal plane. Therefore, each pixel on the focal plane corresponds to a specific wavelength of light. A spectral calibration of the system maps a specific center wavelength to each pixel. More information on the spectral calibration is discussed later.

The camera allows for a number of data capturing modes, including both internal and external triggering. For data collection, an external source triggers the camera. The pulse width of the trigger determines the exposure time for each frame captured with exposure beginning at the up slope of the trigger and ceasing at the down slope. Currently, the system operates with an exposure time of 70ms. Along with the exposure time, the responsivity of the silicon FPA plays an important role in the signal obtained during each capture. Figure 2.1.3 displays the responsivity of the camera.





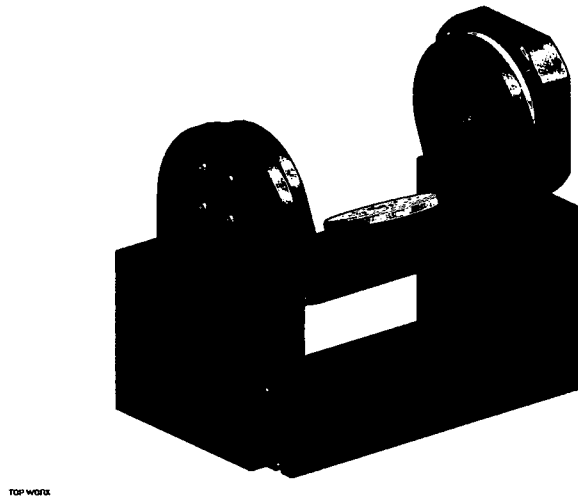
**Figure 2.1.3: FPA Responsivity**

The data provided only extends to 800nm, whereas the dispersion of the spectrograph extends to 1000nm. One can assume the response continues to fall off as the wavelength increases. Ideally, the focal plane response would continue to rise linearly as wavelength increases to compensate for decreasing photon energy of higher wavelength light. However, this system is far from ideal and is designed for use primarily in the visible range of light. The responsivity of the focal plane and length of exposure time determine the magnitude of the resulting digital numbers as a frame is captured. The exact value is determined using,

$$D = I(\lambda)R(\lambda) \quad (2.1.1)$$

where  $D$  is the raw digital number from the camera,  $I(\lambda)$  is the energy of light at a particular wavelength in  $\text{nJ}/\text{cm}^2$ , and  $R(\lambda)$  is the responsivity of the focal plane at that particular wavelength in  $\text{DN}/(\text{nJ}/\text{cm}^2)$ . Due to the low throughput of the system, the analog gain of the camera is set to 4x to increase signal levels. The analog gain amplifies noise as well but the overall signal to noise ratio is improved with a higher gain.

The spectrograph with lens and camera are mounted on a pan and tilt driven by two rotary motor stages. These stages provide the necessary scanning motion for collecting hyperspectral scene data. Figure 2.1.4 displays the design of the pan and tilt with the stages.



**Figure 2.1.4: Pan and Tilt**

The larger rotary stage is used to control the inclination of the spectrograph by adjusting the cradle pitch. Due to the heavy weight of both the cradle and spectrograph, the larger stage is preferred for this function. The smaller rotary stage controls the horizontal scanning of the spectrograph. Both stages are manufactured by Aerotech and belong to the ADRS Series with the smaller being the 150mm version and the larger being the 200mm version.<sup>4</sup> These brushless servo motors are used for their high torque and position accuracy and repeatability. Table 2.1.1 describes the specifications for the motors.

Table 2.1.1: Rotary Stage Specifications from Aerotech<sup>4</sup>

ADRS Series		ADRS-100	ADRS-150	ADRS-200
Width		100 mm	150 mm	200 mm
Tabletop Diameter		95 mm	140 mm	190 mm
Height		55 mm	60 mm	60 mm
Aperture		6 mm	15 mm	25 mm
Motor (-A/-B)		S-76-35-A/S-76-35-B	S-130-39-A/S-130-39-B	S-180-44-A/S-180-44-B
Torque Output	Peak	1.8 N-m (15.9 lb-in)	11.7 N-m (103.6 lb-in)	30.0 N-m (265.5 lb-in)
	Continuous	0.4 N-m (3.5 lb-in)	2.9 N-m (25.7 lb-in)	7.5 N-m (66.4 lb-in)
Resolution		0.87-87.3 $\mu$ rad (0.18-18 arc sec)	0.315-31.5 $\mu$ rad (0.065-6.5 arc sec)	
Max Speed <sup>(1)</sup>		1500 rpm	600 rpm	400 rpm
Accuracy	Uncalibrated	388 $\mu$ rad (80 arc sec)		
	Calibrated <sup>(2)</sup>	29.1 $\mu$ rad (6 arc sec)	48.5 $\mu$ rad (10 arc sec)	48.5 $\mu$ rad (10 arc sec)
Repeatability		14.6 $\mu$ rad (3 arc sec)	19.4 $\mu$ rad (4 arc sec)	19.4 $\mu$ rad (4 arc sec)
Max Load <sup>(1)</sup>	Axial	70 N (15.7 lb)	200 N (45 lb)	400 N (89.9 lb)
	Radial	30 N (6.7 lb)	100 N (22.5 lb)	200 N (45 lb)
Axial Error <sup>(4)</sup>		2 $\mu$ m	5 $\mu$ m	5 $\mu$ m
Radial Error <sup>(4)</sup>		3 $\mu$ m	5 $\mu$ m	5 $\mu$ m
Tilt Error		48.5 $\mu$ rad (10 arc sec)	97 $\mu$ rad (20 arc sec)	97 $\mu$ rad (20 arc sec)
Inertia	Unloaded	0.00038 kg-m <sup>2</sup>	0.00242 kg-m <sup>2</sup>	0.00843 kg-m <sup>2</sup>
Total Mass		2.0 kg	4.3 kg	7.6 kg
Finish	Tabletop	Hardcoat		
	Stage	Black Anodize		

In order to maintain good image registration for the data scenes collected, the position accuracy of the pan and tilt is essential. Each of these motors is positioned using an Aerotech NDrive motion controller which provides the external triggering for the camera as well.

In order to obtain highly accurate spectra of the targets present within the scene, a handheld field spectrometer is used. This instrument allows the user to retrieve a spectral signature directly from an object virtually eliminating atmospheric effects. Highly accurate spectral data without the presence of atmospheric interference is often termed "lab spectra". Lab spectra are useful when trying to truth a scene and when trying to compensate for atmospheric effects in remotely sensed data. Atmospheric compensation is important when attempting to identify specific materials within a scene. More on this topic is addressed later.

The handheld spectrometer used is a FieldSpec®Pro FR produced by Analytical Spectral Devices, Inc. This unit operates in the visible to near infrared (VNIR) and the shortwave infrared (SWIR) with a spectral range of 350nm – 2500nm and a resolution of about 3nm in the VNIR and 10nm in the SWIR.<sup>5</sup> The sampling period is about 1.4nm in the VNIR and 2nm in the SWIR. Spectral data is obtained from objects using a contact probe with a built-in halogen light providing illumination similar to that of the sun. This probe helps eliminate atmospheric effects since it provides illumination and is pressed directly against the object of interest. A fiber optic cable transfers light from the foreoptic into the actual spectrometer system. Utilizing a laptop, the data is processed and displayed using supplied software. This instrument is used to obtain spectra from all four aluminum panels in the scene and from several tarps as well.

## **2.2 Spectral Calibration**

Light entering the spectrometer is dispersed by the diffraction grating in the direction perpendicular to slit orientation. This light falls onto the focal plane directly behind the grating. With the slit oriented vertically on the system, the vertical dimension on the focal plane contains spatial information whereas the horizontal dimension contains spectral information. When a snapshot is taken, each pixel on the focal plane corresponds to a portion of the scene at a specific wavelength. When the hyperspectral system is first constructed, the spectral information on the focal plane is unknown. Consequently, a procedure known as a spectral calibration is performed. Spectral calibration is the process of mapping a center wavelength value to each pixel on the FPA. Ideally, the center wavelength should vary only from one column to the next on the FPA as this is the direction of light dispersion. However, the center wavelength can change

from row to row as well, constituting errors within the system such as smile distortion, quadratic dispersion, or sensor misalignment. The general equation for wavelength mapping is

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

where  $m$  is the focal plane row,  $n$  is the focal plane column and  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ , and  $f$  are the coefficients describing the wavelength mapping.<sup>6</sup> In order to determine the coefficient values, light sources with specific known wavelengths must be used. For example, a specific Helium-Neon (HeNe) laser produces a red beam of light with a center wavelength of about 632.8nm and a very narrow bandwidth. If this laser is shone onto the slit of the hyperspectral system and a snapshot is taken, a bright vertical line should appear in the image. Ideally, this line would appear in a single column on the focal plane. That column position  $n$  corresponds to a wavelength mapping of 632.8nm. Figure 2.2.1 displays the output of the camera when a HeNe laser is used to flood the input slit. The specific R-30025 HeNe laser produced by Newport possesses a minimum output power of 1.5mW.<sup>7</sup>



**Figure 2.2.1: HeNe Laser Output**

The spectral line corresponding to 632.8nm is plainly seen as is the second order which will be discussed later. In an imperfect system, the line may sprawl across other columns when changing row position.

In order to solve for the mapping coefficients, more sources in addition to the HeNe must be used. Gas lamps are often employed to perform spectral calibration as they produce light possessing well known peaks at specific spectral locations.<sup>8</sup> Oriel pencil style calibration lamps are used for this purpose as these produce narrow, intense spectral lines by exciting various gases.<sup>9</sup> The lamps come equipped with an 115V AC power supply capable of delivering an adjustable current between 0mA and 20mA. For this spectral calibration, the lamps used are 6030 Argon operated at 10mA, 6031 Krypton operated at 10mA, and 6034 Mercury-Neon operated at 18mA. Observable spectral peaks for these gas lamps are shown in Figures 2.2.2 – 2.2.4.

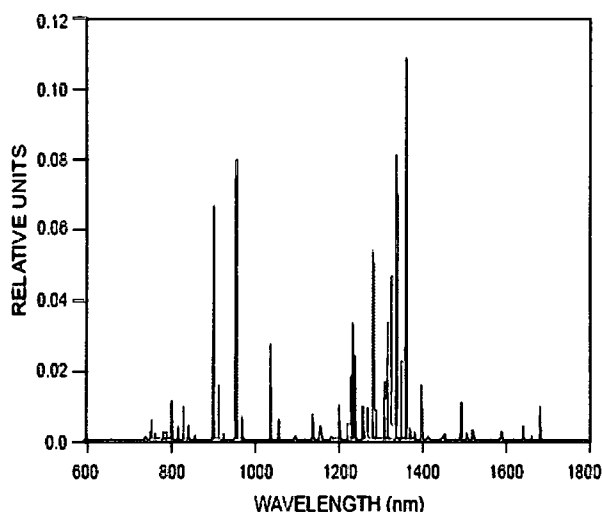


Figure 2.2.2: Output Spectrum for Ar Lamp<sup>9</sup>

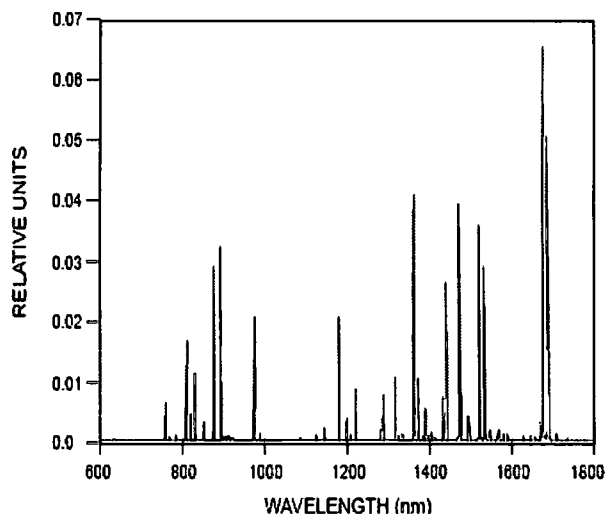
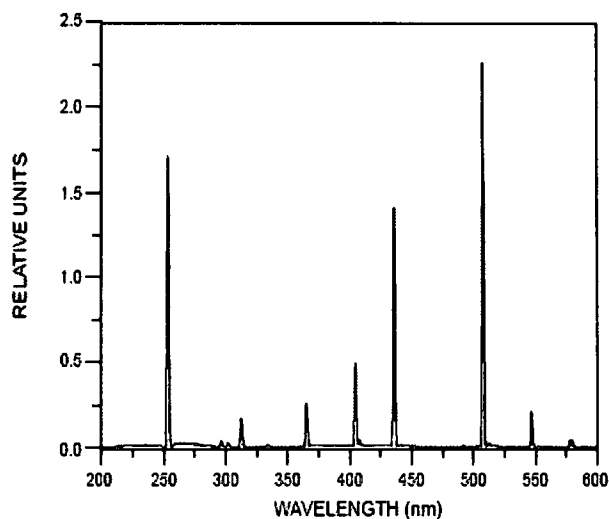
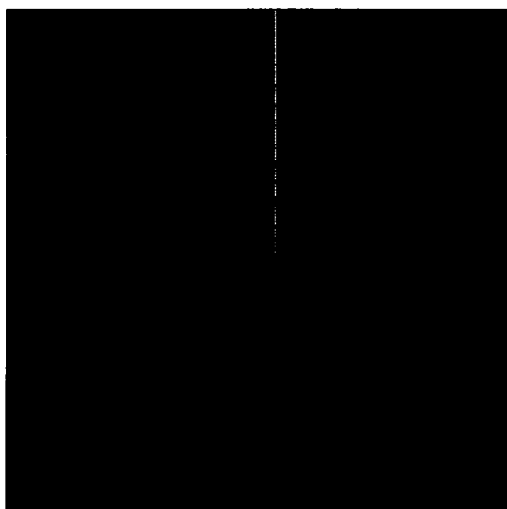


Figure 2.2.3: Output Spectrum for Kr Lamp<sup>9</sup>

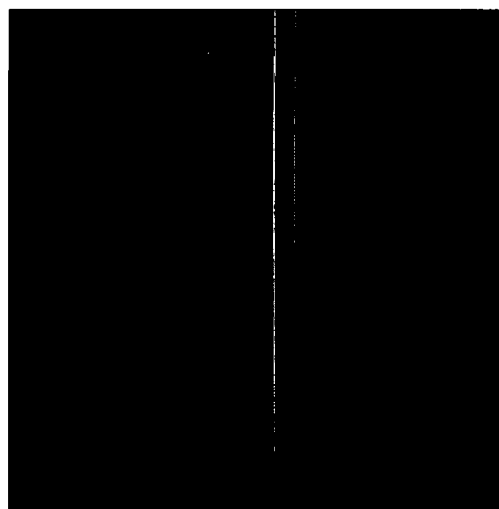


**Figure 2.2.4: Output Spectrum for HgNe Lamp<sup>9</sup>**

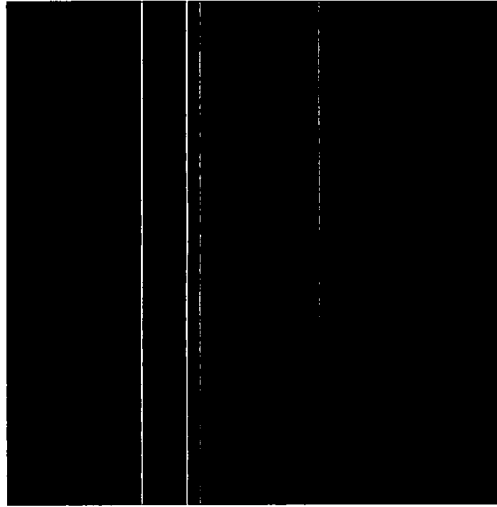
Figure 2.2.5 displays the camera output when an Argon gas lamp is used to illuminate the slit. Dominant spectral lines appear on the focal plane which can be identified using the known HeNe wavelength and documentation provided with the lamps. Figures 2.2.6 and 2.2.7 display the camera outputs for Krypton and Mercury-Neon lamps.



**Figure 2.2.5: Ar Lamp Camera Output**



**Figure 2.2.6: Kr Lamp Camera Output**



**Figure 2.2.7: HgNe Lamp Camera Output**

Using these calibration sources, an over-determined system of linear equations is produced by sampling  $M$  rows on  $N$  different spectral lines. An estimate of the mapping coefficients is found using the least-squares solution for the system of equations. After performing the necessary calculations, the coefficients are estimated as,

$$a = b = c = 0$$

$$d = 0.0016$$

$$e = 1.1933$$

$$f = 105.7009$$

where  $a$ ,  $b$ , and  $c$  are set to zero since their values are less than  $10^{-6}$ . The resulting wavelength mapping is then

$$\lambda(m,n) = 1.1933n + 0.0016m + 105.7009 \quad [\text{nm}] \quad (2.2.2)$$

with the units in nanometers. Ideally, the wavelength would depend solely upon column position. However, due to some rotational error, a row dependency exists as well. Using a rotational error correction, one can eliminate the row dependency and utilize an equation depending only on column position. From the spectral characterization performed, a rotational error of  $0.0768^\circ$  is determined. Utilizing this knowledge, the data



acquired is rotated 0.0768° counter-clockwise using a bilinear interpolation method for correction. The rotation occurs about the center of the focal plane. Consequently, the wavelength mapping equation utilizes the positions designated by the center row. The simplified relationship is described by,

$$\lambda(n) = 1.1933n + 106.5198 \quad [\text{nm}] \quad (2.2.3)$$

where the row dependency no longer exists. This equation depends solely upon the column position due to the rotational correction utilized. For all further purposes, this wavelength mapping is used. For this system, the wavelength mapping resulting from the spectral calibration is used throughout the data collection period.

### 2.3 Noise Characterization

Noise presents quite an obstacle when trying to acquire data and calibrate a system. A high signal to noise ratio (SNR) is essential to performing an accurate absolute radiometric calibration. In addition, the calibration process aids in dealing with certain types of noise within the system. When dealing with a camera such as the one within this system, different types of noise exist and must be characterized. Some noise processes within the system dominate more than others. Consequently, identifying the dominant process and reducing it in any way possible is vital for system performance. Two general types to consider are temporal noise and pattern noise. Temporal constitutes a frame-to-frame change in a pixel when viewing a uniform source.<sup>10</sup> Pattern refers to a spatial pattern that does not change from frame to frame.<sup>11</sup> Within this camera system, dark current is one form of pattern noise considered since it varies from pixel to pixel. Dark current, however, does have a temporal variation to it as well. The variation from pixel to pixel of dark current is termed fixed pattern noise (FPN). The spatial variation in dark

current results from differences in pixel size, doping density, and impurities trapped during fabrication.<sup>11</sup> Dark current itself results mainly from the thermal output of the camera. Therefore, dark current can be significantly reduced by cooling the camera. While the temporal variation of dark current cannot be removed, the spatial variation can be accounted for and extracted. An example of the FPN is seen by averaging a large number of frames of the same integrating sphere scene. An integrating sphere provides spatially uniform illumination for the system which is invaluable when trying to characterize noise. Figure 2.3.1 represents a row across the focal plane obtained by averaging 1500 frames of the light source.

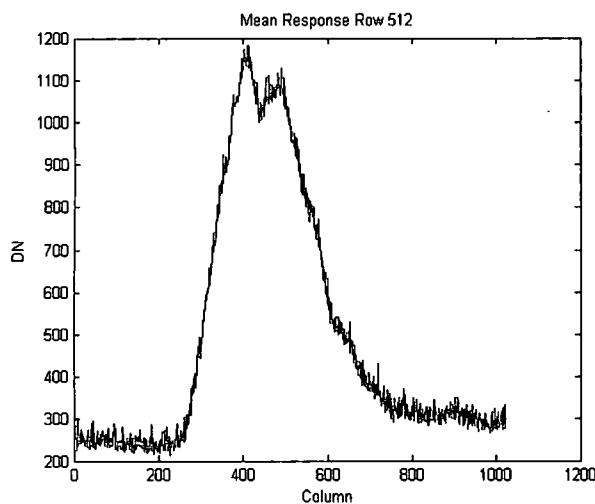


Figure 2.3.1: Fixed Pattern Noise

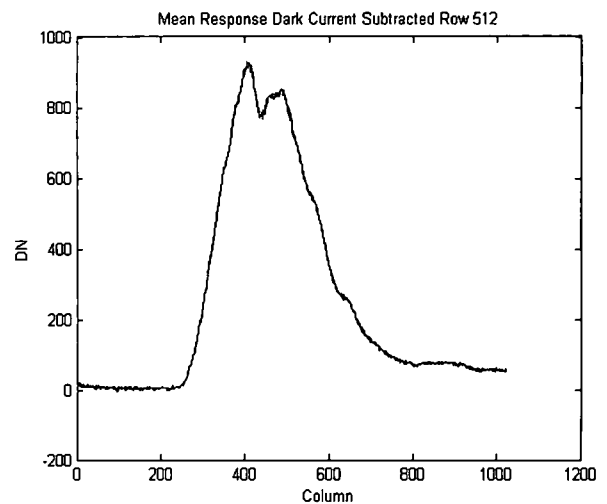


Figure 2.3.2: FPN Removed

At first glance, the row response appears very noisy. However, when subtracting the mean dark current, the signal appears much cleaner. Figure 2.3.2 displays the same average row with the dark current subtracted.

The slight variations that still remain in the signal are due to temporal noise. Temporal noise can never be completely eliminated but only attenuated through

averaging. By using a simplified noise model, a better characterization of the system can be formulated. A noise model for each pixel of the system appears as

$$\eta_{SYS} = \sqrt{\eta_{SHOT}^2 + \eta_{FLOOR}^2 + \eta_{PATTERN}^2} \quad (2.3.1)$$

with the power of the independent noise processes adding to the total noise.<sup>11</sup> The row and column dependency is inferred for convenience. Each of these general noise categories can be broken down further to represent specific contributors. While numerous noise contributors exist, the main ones specific to this system are examined. The shot noise is decomposed to,

$$\eta_{SHOT}^2 = \eta_{PE}^2 + \eta_{DARK}^2 \quad (2.3.2)$$

where PE represents photoelectron noise and DARK represents the temporal variation in the dark current. Photoelectron noise varies both temporally and spatially simply due to the discrete nature of electrons.<sup>11</sup> For this system, the temporal noise characteristics are of concern. Subdividing the pattern noise results in,

$$\eta_{PATTERN}^2 = \eta_{FPN}^2 + \eta_{PRNU}^2 \quad (2.3.3)$$

where FPN is the fixed pattern noise associated with dark current mentioned earlier. PRNU represents the Photoresponse Non-uniformity of the pixels in the focal plane. Theoretically, each pixel should have the same response to a photon of a given wavelength. Any variation in this response is termed PRNU. For this system, the PRNU is accounted for in a radiometric calibration technique that is discussed later. The floor noise within the model represents the amplifier noise of the system. This amplifier noise exists both temporally and spatially. The total noise model can be reduced further considering the pattern noise can be accounted for and removed. With this term removed, the model simplifies to

$$\eta_{SYS} = \sqrt{\eta_{SHOT}^2 + \eta_{FLOOR}^2} \quad (2.3.4)$$

where only the shot and floor noise are considered.

The temporal variations associated with these terms remain a major concern. In order to assess the statistics of the noise, one must analyze the system in a probabilistic fashion. When evaluating the focal plane, one can view each pixel as a random variable and a number of temporal samples of each pixel represent a random sequence. As such, each random sequence possesses a sample mean defined by,

$$\hat{\mu}(m,n) = \frac{1}{I} \sum_{i=1}^I d(m,n,i) \quad (2.3.5)$$

where  $I$  is the number of temporal samples of pixel  $d(m,n)$  with  $m$  and  $n$  designating the row and column position of the pixel and  $i$  designates the temporal sample.<sup>12</sup> Similarly, the random sequence possesses a sample variance defined by

$$\hat{\sigma}^2(m,n) = \frac{1}{I-1} \sum_{i=1}^I [d(m,n,i) - \hat{\mu}(m,n)]^2 \quad (2.3.6)$$

These two statistics are useful in determining signal and noise power. If the system is sampling a constant signal such as an integrating sphere output, any differences between samples result from noise. The differences or noise present are characterized by the variance of the signal. With enough samples, a good estimate of the signal can be determined through averaging. The number of samples needed to produce a good estimate depends upon the noise present. For this system, 1500 samples are taken to obtain an estimate of the mean and variance. While the values calculated are still sample values, the exorbitant number of frames should produce a fairly accurate measure of the desired quantities. Since scene data is calibrated using only 100 frames, a comparison of

the sample mean and variance for this estimate as opposed to a 1500 sample estimate provides an idea of any error resulting from the use of fewer samples.

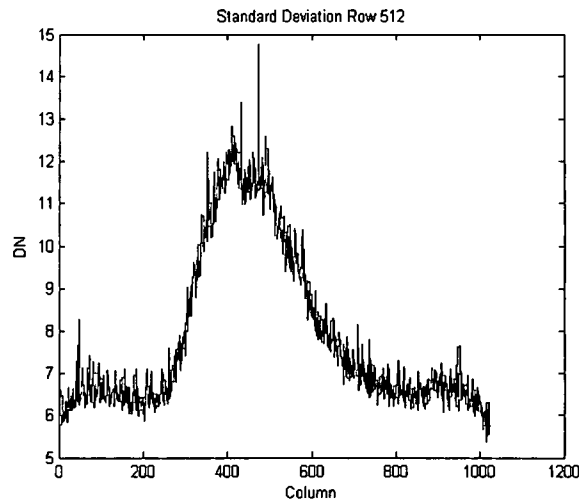
Before comparing the sample values, one must determine what region of the focal plane offers usable data. This differentiation is based upon the signal present for a specific pixel versus the noise present. The variance of a constant signal offers a good estimate of the noise power. Using the noise model from before, the noise power for a given pixel can be defined as

$$\eta_{SYS}^2 = \hat{\sigma}^2(m, n) \quad (2.3.7)$$

where  $\hat{\sigma}^2(m, n)$  is the sample variance of the random sequence determined using 1500 samples. From here, the noise for a given pixel is,

$$\eta_{SYS} = \hat{\sigma}(m, n) \quad (2.3.8)$$

where  $\hat{\sigma}(m, n)$  is termed the sample standard deviation of a pixel. Figure 2.3.3 displays the standard deviation of a row calculated using 1500 frames.



**Figure 2.3.3: Row Noise**

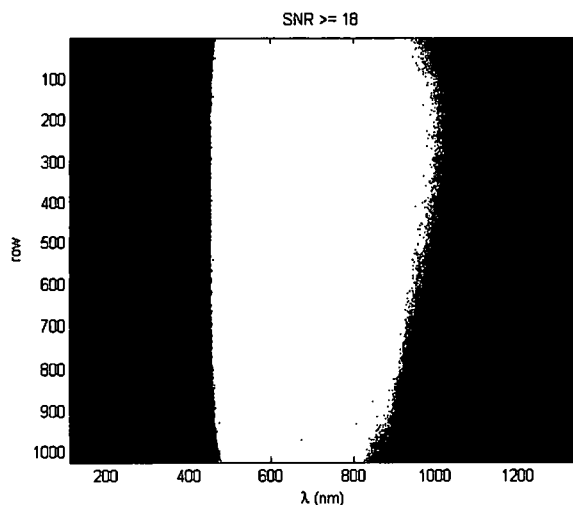
The noise across the focal plane rises and falls in a similar fashion as the signal seen in Figure 2.3.1. As mentioned earlier, the signal is determined by averaging the samples of a pixel. Consequently, the signal for a pixel is defined as

$$s = \hat{\mu}(m, n) \quad (2.3.9)$$

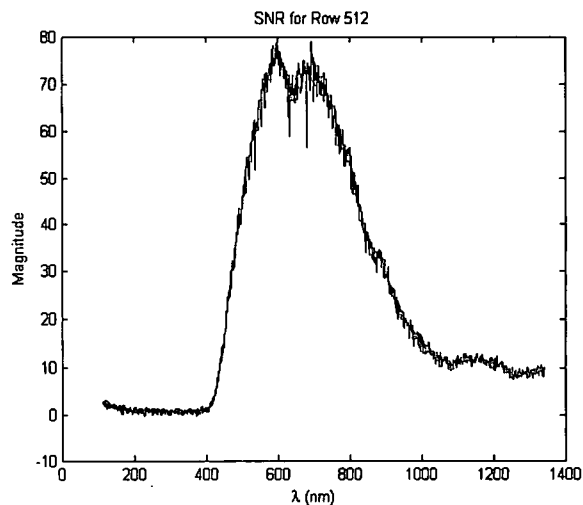
where  $\hat{\mu}(m, n)$  is the sample mean of the signal produced from 1500 samples. The row and column dependency of the signal again is inferred. After calculating the signal and noise for each pixel, one can use the SNR to determine pixels offering reliable data. SNR is defined as,

$$SNR = \frac{s}{\eta_{sys}} = \frac{\hat{\mu}(m, n)}{\hat{\sigma}(m, n)} \quad (2.3.10)$$

where the result is unitless. After finding the SNR for every pixel, a threshold is set to differentiate between good and bad data. The initial threshold depends upon the desired SNR for the final data. With spectral binning or filtering applied later, the initial SNR will rise in proportion to the square root of the bin length. Binning constitutes an averaging of data coupled with down sampling whereas filtering does not automatically imply down sampling. In addition to the filtering gain, the scene itself is brighter than the integrating sphere resulting in a higher SNR. The minimum SNR chosen for the integrating sphere data after binning is 50, assuming the actual signal from the scene is higher. Using a bin length of 8, the initial SNR must be at least 18. Figure 2.3.4 shows portions of the focal plane possessing at least this minimum.

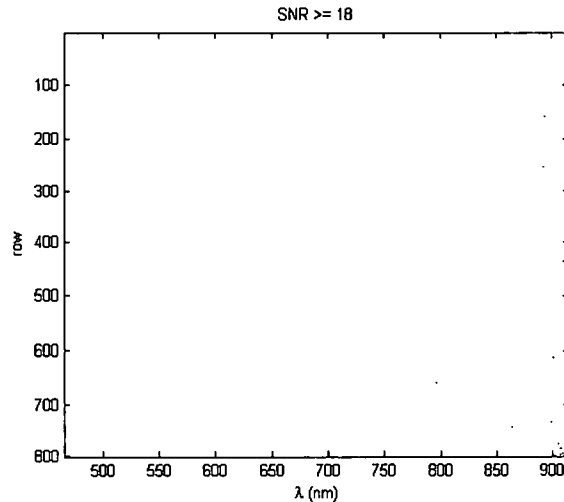


**Figure 2.3.4: SNR Threshold for Focal Plane**



**Figure 2.3.5: SNR Across Row**

The image shows how a large portion of the FPA is unusable. A non-uniform signal down a column causes a falloff in SNR at the top and bottom as well. Viewing the row response for SNR offers better insight to the data. Figure 2.3.5 displays the SNR across a row for the focal plane. From the figure, one observes the signal rise and falloff sharply when approaching 400nm and 1000nm respectively. Again, this trait stems back to the integrating sphere output, focal plane silicon response, and diffraction grating dispersion. As a result of the falloff across both row and column, a spectral range and row range is chosen meeting the minimum SNR requirement. Setting a spectral range around 450nm to 900nm coupled with a row range of 1 to 800 meets this requirement. Figure 2.3.6 shows the threshold results in this range.

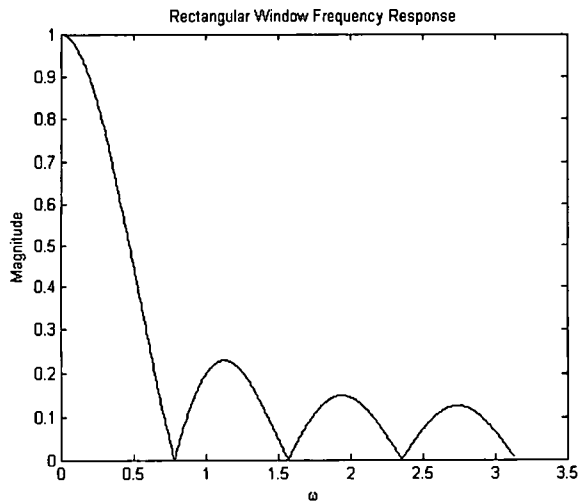


**Figure 2.3.6: SNR  $\geq 18$  for Limited Spectral and Row range**

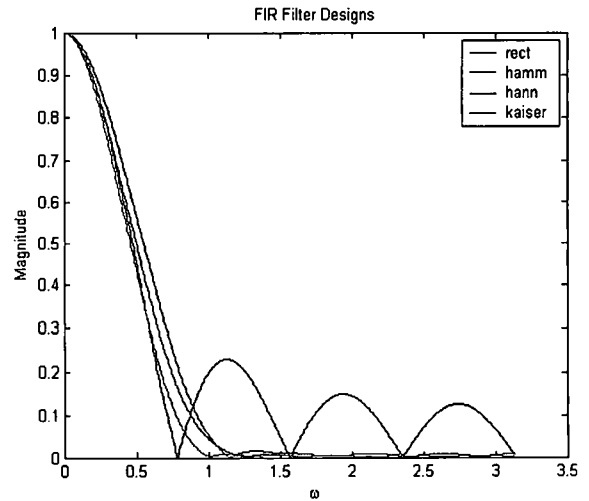
Only a few pixels in the lower right corner do not meet the minimum within this range. After binning or filtering, the spectral resolution of the system is reduced to about 8nm. Depending on the spectral sampling chosen after the filtering process, the number of bands for the stated range will vary.

Once the appropriate range of data is selected, a method of spectral averaging or filtering is chosen. Low pass filtering of data produces the same averaging effect with some benefits over binning. Spectral binning utilizes a rectangular finite impulse response (FIR) filter. The frequency response of the rectangular window of length 8 is seen in Figure 2.3.7.





**Figure 2.3.7: Bin Frequency Response**



**Figure 2.3.8: Filter Frequency Responses**

Binning performed on data results in unwanted side lobes in the frequency domain. These side lobes can produce undesirable effects on the data. If an FIR filter is designed to match the main lobe response of the rectangular window, the same gain in SNR can be achieved with the benefit of reduction in side lobe magnitude. Figure 2.3.8 displays several FIR filters designed using different window types.<sup>13</sup> All of the filters designed follow the main lobe response of the rectangular window fairly well and attenuate the side lobes. The disadvantage of using the other designs is the added filter length. Where the rectangular window uses only 8 coefficients, the other designs utilize 13 coefficients. Due to the large range of the data being filtered, the added length does not pose a problem. For data gathered in this research, the Kaiser design is used as it models the main lobe best while suppressing side lobes.

After determining the filter design and area of the focal plane meeting the SNR requirements, the significance of using fewer frames to develop an average radiometric signal is determined. Since the average radiometric frame is used for calibration, one relies on its accuracy. By comparing the confidence intervals associated with 100 and

1500 samples, an idea of the estimation error for the sample mean is established. A confidence interval is thought of as a range of values which very likely contains the true value of the parameter being estimated.<sup>14</sup> Usually, a certain confidence percentage is associated with the specified interval. The width of the interval is directly related to the confidence level. In this case, the probability of the true mean value lying within a specific interval is found via,

$$P\left[\hat{\mu}(m,n) - \frac{z\hat{\sigma}(m,n)}{\sqrt{I}} \leq \mu(m,n) \leq \hat{\mu}(m,n) + \frac{z\hat{\sigma}(m,n)}{\sqrt{I}}\right] = 1 - 2F_{t-1}(-z) \quad (2.3.11)$$

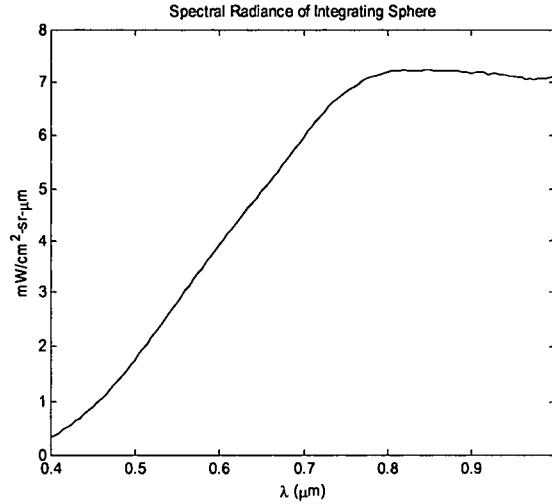
where  $\mu(m,n)$  is the actual mean,  $F_{t-1}$  is the sample-dependent Student's t-distribution, and  $z$  is the value determining the confidence of the interval. Using the probability, one can define the confidence interval as

$$\left(\hat{\mu}(m,n) - \frac{z\hat{\sigma}(m,n)}{\sqrt{I}}, \hat{\mu}(m,n) + \frac{z\hat{\sigma}(m,n)}{\sqrt{I}}\right) \quad (2.3.12)$$

which describes the range the actual mean resides for a given certainty. The width of the interval is dependent upon the number of samples, the sample standard deviation, and the value of  $z$ . Typically, a table is used to determine a value for  $z$  corresponding to a certain confidence level and a specific number of samples. For a 95% confidence interval, the  $z$ -value for 100 samples is 1.984 whereas the  $z$ -value for 1500 samples is closer to the infinite sample value of 1.960.<sup>14</sup> Therefore, the difference between the two values is relatively minute. Coupling this small difference with the knowledge that the standard deviations for 100 and 1500 samples do not vary greatly, one can assume the confidence intervals for both are very similar. Consequently, not much confidence is lost in the sample mean estimate when using 100 frames as opposed to 1500.

## 2.4 Radiometric Calibration

With a better understanding of the system noise, one can proceed with the calibration procedure. Ideally, every pixel within a given FPA column should possess exactly the same value when observing a uniform source. In this case, the source is an integrating sphere which acts as an accurately calibrated uniform radiator producing a specific spectral radiance for a given lamp luminance.<sup>15</sup> Radiance refers to the power per unit projected area per unit solid angle of the optical radiation whereas spectral radiance has an added wavelength dependency. Luminance is a photometric quantity weighted by an eye response function describing the brightness of the source. By placing the spectrograph system with lens and camera at the exit port of the integrating sphere, the resulting digital numbers from the camera at any given pixel can be accurately traced back to a specific calibrated spectral radiance value for the corresponding wavelength. The process of mapping digital numbers to specific radiometric values and correcting pixel response non-uniformity is termed an absolute radiometric calibration. Due to the variability of pixel response and noise processes, a separate radiometric calibration must be performed on every data set collected. The integrating sphere used for calibration purposes is a Labsphere USS-600 with a 6-inch sphere diameter and a 2-inch exit port diameter. Figure 2.4.1 displays the calibrated spectral radiance data for the integrating sphere with a measured luminance value of 634.8 foot-lamberts or 2174.8 cd/m<sup>2</sup>.



**Figure 2.4.1: Calibrated Integrating Sphere Data**

The calibrated integrating sphere allows for absolute radiometric calibration and noise characterization of the system. While photometric quantities are used to describe the lamp brightness, these quantities are weighted by an eye spectral response function. The final calibrated data is in radiometric spectral radiance units, which offers a better measure of the actual number of photons reaching the focal plane.

An adjustable iris near the lamp source allows the user to decrease the luminance level and corresponding spectral radiance. A silicon detector within the sphere measures the photometric quantity of luminance at any given time. The decrease in the spectral radiance exiting the sphere corresponds to the fractional decrease in this photometric luminance value. The values are found using,

$$L_{\lambda A}(\lambda) = \frac{L_{VM}}{634.8} L_{\lambda C}(\lambda) \quad (2.4.1)$$

where  $L_{\lambda A}$  is the adjusted spectral radiance value,  $L_{\lambda C}$  is the calibrated spectral radiance value, and  $L_{VM}$  is the measured luminance in foot-lamberts. Consequently, one can use the luminance measurement from the silicon detector to determine the exact spectral

radiance of the sphere. An integrating sphere such as this one proves to be an invaluable tool for characterizing and calibrating a hyperspectral system.

Ideally, when the system observes the output of the integrating sphere, every pixel down a given column on the FPA should possess the same digital value. However, due to system effects, temporal noise, and PRNU, each pixel corresponding to the same spectral radiance may not possess the same digital number. The radiometric calibration accounts for this inequality by establishing a gain and offset to convert each digital number to the correct spectral radiance value associated with that pixel. When correcting collection data, a two-point calibration is used. Typically, the two-point calibration utilizes a light frame and a dark frame to produce a gain and offset term for mapping digital numbers to spectral radiance units. One describes the mapping using,

$$L_{\lambda}(m,n) = a(m,n)D(m,n) + b(m,n) \quad (2.4.2)$$

where  $D$  is the raw digital number of a pixel,  $L_{\lambda}$  is the spectral radiance,  $a$  is the gain term, and  $b$  is the offset term. One obtains a light frame by averaging a number of frames collected at a high radiance. The dark frame is produced by averaging a number of frames at a lower radiance value. Adjusting the iris on the integrating sphere allows for the acquisition of different light levels. While performing the calculation for the linear terms, one must keep in mind that the wavelength corresponding to a specific pixel depends on both column and row position, as described by (2.2.2). By performing a rotational correction on the data, the row dependency of wavelength is eliminated. Assuming wavelength may vary with row position as well, the gain and offset can be determined using,

$$a(m,n) = \frac{L_{\lambda L}(m,n) - L_{\lambda D}(m,n)}{D_L(m,n) - D_D(m,n)} \quad (2.4.3)$$

and

$$b(m,n) = \frac{L_{\lambda D}(m,n)D_L(m,n) - L_{\lambda L}(m,n)D_D(m,n)}{D_L(m,n) - D_D(m,n)} \quad (2.4.4)$$

where  $L_{\lambda D}$  is the spectral radiance of the dark frame,  $L_{\lambda L}$  is the spectral radiance of the light frame,  $D_D$  is the raw digital number of the dark frame and  $D_L$  is the raw digital number of the light frame. If a rotational correction is applied, the spectral radiance values used should not depend upon row position. The actual spectral radiance value corresponding to a given pixel may need to be interpolated if the pixel center wavelength falls in between the wavelengths used for the calibration data provided with the integrating sphere.

The equations above represent the general approach to a two-point calibration scheme. A more specific case for the two point calibration uses a near saturation frame and a completely dark frame to calculate the gain and offset. By collecting with the iris on the sphere completely open, one obtains the brightest frame possible for that particular integrating sphere and light source. For this sphere and system, the brightest frame does not approach saturation. Conversely, the darkest frame is obtained by placing a lens cap on the system and acquiring data. The spectral radiance with the lens cap on is zero. In this case, the dark frame itself can be viewed as the dark current offset term. With enough samples taken, the noise is reduced and an accurate measure of the dark current exists. From earlier, a 100-frame average provides a good estimate. After subtracting the dark current from the light frame, the gain term can be found simply by dividing the

spectral radiance data by the dark-current-subtracted light frame data. The gain is a simplified version of (2.4.3) described by,

$$a(m, n) = \frac{L_{\lambda}(m, n)}{D_{L-D}(m, n)} \quad (2.4.5)$$

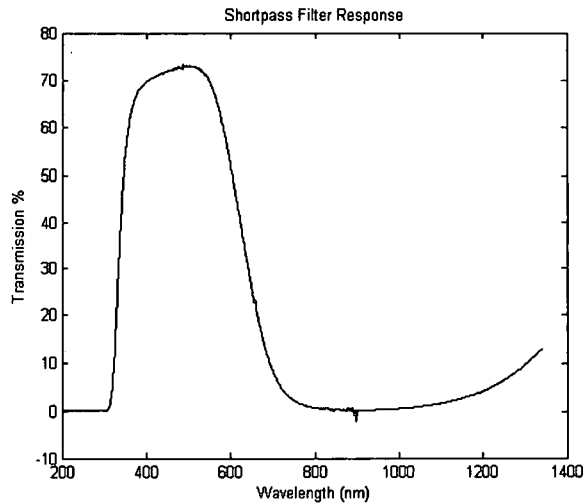
where  $D_{L-D}$  represents the dark-current-subtracted light frame.

## 2.5 Second Order Correction

This spectrograph system requires the use of the specific calibration case above due to a second order dispersion effect present from the dispersion grating. A blazed dispersion grating is designed to shift the diffraction envelope maximum into a specific order.<sup>16</sup> The typical diffraction grating equation is described by

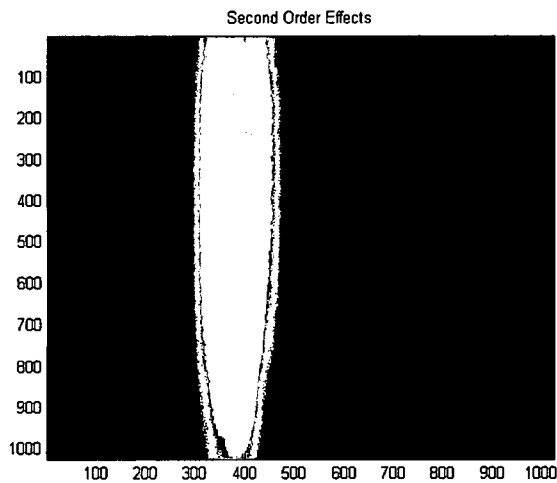
$$a(\sin \theta_i + \sin \theta_m) = m\lambda, \quad m = 0, \pm 1, \pm 2, \dots \quad (2.5.1)$$

where  $\theta_i$  is the incident angle,  $\theta_m$  is the diffracted angle for a specific order  $m$ ,  $a$  is a constant dealing with the grating period.<sup>16</sup> The equation describes how the diffraction angle depends upon both wavelength and order. In this system, the design of the grating places maximum power into the first diffraction order. However, other orders exist as well, appearing with less intensity at integer multiples of the first diffraction. For example, the second diffraction order for 400nm light appears at 800nm on the focal plane and the third order appears at 1200nm. Consequently, at 800nm on the focal plane, the incident light is actually a blend of first order 800nm light and second order 400nm light. By placing a shortpass filter on the system lens, one can observe the second order effects. Figure 2.5.1 displays the filter response.

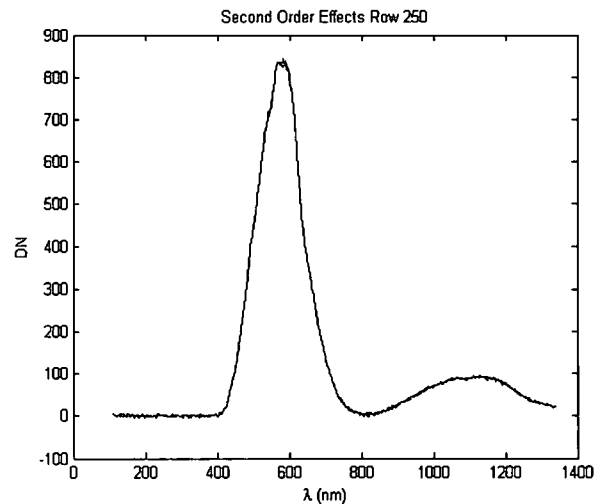


**Figure 2.5.1: Shortpass Filter Response**

The filter essentially stops any light higher than 700nm from entering the system. Figure 2.5.2 displays an average focal plane image when the system is illuminated using the integrating sphere with the shortpass filter in place and dark current subtracted.



**Figure 2.5.2: Second Order Effects in Focal Plane**



**Figure 2.5.3: Second Order Effects Across Row**

The second order effects may be difficult to see in this image. If a row response is observed, the effect is a little more visible. Figure 2.5.3 displays a row response. In other spectrograph systems, an Order Sorting Filter is placed on the focal plane to remove



the other diffraction orders. Since this system does not possess one, the removal of the second order effects involves processing of the data. The free spectral range of a spectrograph system is the range in which there is no overlapping of diffraction orders.<sup>16</sup> In this case, the system is designed for the range of 400nm – 1000nm and any light outside this range is considered insignificant. Consequently, second order effects do not appear until 800nm resulting in a free spectral range of 400nm – 800nm. This range does not need to be corrected. However, the 800nm – 1000nm range requires second order removal.

Essentially, the second order removal process requires the determination of the relative diffraction efficiency  $\eta_{12}$  of the grating. The relationship of the diffraction efficiency to the first and second order terms is,

$$I_{2\lambda} = \eta_{12} I_{\lambda} \quad (2.5.2)$$

where  $I_{\lambda}$  is the intensity of the first diffraction order and  $I_{2\lambda}$  is the intensity of the second diffraction order. In a perfect system, this relative diffraction efficiency term can be utilized to subtract off the second order light. Unfortunately, due to system irregularities such as PRNU, a separate gain term must be calculated for each pixel needing correction. This gain calculation requires the use of the short-pass-filtered integrating sphere data. The second order removal process begins by describing a general pixel response,

$$D(m,n) = \begin{cases} A_{\lambda}(m,n)L_{\lambda}(\lambda) + B(m,n) & \lambda < 800nm \\ A_{\lambda}(m,n)L_{\lambda}(\lambda) + B(m,n) + A_{\lambda/2}(m,n)L_{\lambda}(\lambda/2) & \lambda \geq 800nm \end{cases} \quad (2.5.3)$$

where  $D(m,n)$  is the raw digital number for a pixel in the short-pass filtered data,  $A_{\lambda}(m,n)$  is the gain term associated with a pixel for light at the wavelength designated by

the column position,  $A_{\lambda/2}(m, n)$  is the gain term associated with a pixel for light at half the wavelength designated by the column position,  $L_{\lambda}(\lambda)$  is the spectral radiance coming into the system at the wavelength designated by the column position,  $L_{\lambda}(\lambda/2)$  is the spectral radiance coming into the system at half the wavelength designated by the column position, and  $B(m, n)$  is the dark current offset. Above 800nm, a second order term is incorporated. With the short-pass filter in place, the response can be re-written as,

$$D(m, n) = \begin{cases} A_{\lambda}(m, n)L_{\lambda}(\lambda) + B(m, n) & 400nm \leq \lambda \leq 500nm \\ B(m, n) + A_{\lambda/2}(m, n)L_{\lambda}(\lambda/2) & 800nm \leq \lambda \leq 1000nm \end{cases} \quad (2.5.4)$$

for the relevant spectral ranges. Utilizing a dark measurement to remove the offset term, the equation simplifies even further.

$$D(m, n) = \begin{cases} A_{\lambda}(m, n)L_{\lambda}(\lambda) & 400nm \leq \lambda \leq 500nm \\ A_{\lambda/2}(m, n)L_{\lambda}(\lambda/2) & 800nm \leq \lambda \leq 1000nm \end{cases} \quad (2.5.5)$$

By mapping the second order pixels back to their corresponding first order pixels, a second order gain term  $\alpha$  can be found by

$$\alpha(m', n') = \frac{D(m, n)}{D(m', n')} = \frac{A_{\lambda/2}(m, n)L(\lambda)}{A_{\lambda}(m, n)L(\lambda)} = \frac{A_{\lambda/2}(m, n)}{A_{\lambda}(m, n)} \quad (2.5.6)$$

where the prime notation is used to signify the location of the pixel/s producing the second order effect. Utilizing (2.2.3), the exact pixel to pixel mapping can be found. Note that first order light producing the second order effect may be a fraction of one or two columns. After determining the gains, the correction is performed by simple multiplication and subtraction described by,

$$D_C(m, n) = \begin{cases} D(m, n) & \lambda < 800nm \\ D(m, n) - \alpha(m', n')D(m', n') & \lambda \geq 800nm \end{cases} \quad (2.5.7)$$

where  $D_c$  is the corrected pixel value. After performing the correction, the radiometric gain term for calibration is found using (2.4.5).

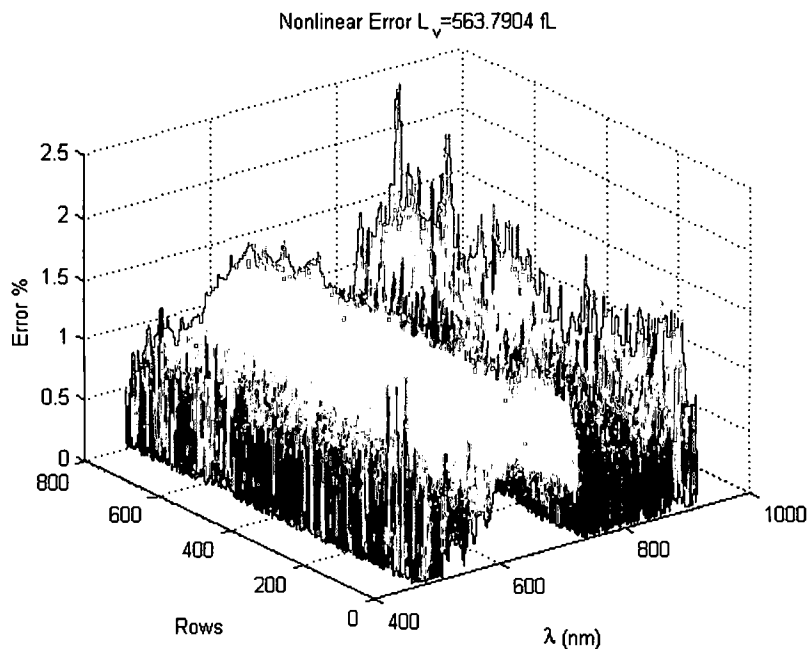
## 2.6 Non-linearity Characterization

A linear radiometric calibration method is performed on the data. Using this method assumes the response of each pixel to light of a given wavelength is linear. Under this assumption, an increase in radiance creates an increase in the digital number described by a gain and offset term. If the response is non-linear, higher order terms are necessary to perform an accurate radiometric calibration. Typically, non-linearity is tested using a point near the noise level and a point at 90% full well capacity for a pixel to compute the best fit straight line.<sup>17</sup> Unfortunately, the integrating sphere does not emit enough light to approach 90% saturation. Consequently, only a limited range of values are tested for non-linearity. Several frames in between the light and dark frame are taken for testing. Using the gain and offset term found from the two extreme points, the spectral radiance of the other light levels is calculated using (2.4.2) and compared to the ideal spectral radiance values found using (2.4.1). The non-linear error is measured as a percentage of the full scale response or in this case, the spectral radiance values for the light frame used for the linear fit. The non-linear error is,

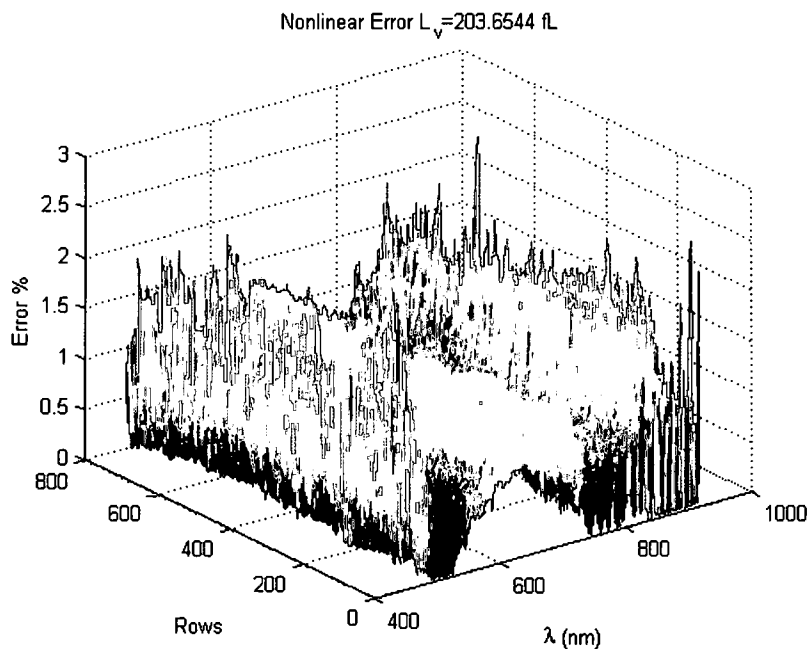
$$\varepsilon_{NL} = \frac{|L_{\lambda\lambda}(m,n) - L_{\lambda}(m,n)|}{L_{\lambda\lambda}(m,n)} * 100 \quad (2.6.1)$$

where the respective radiance values are described in (2.4.1), (2.4.2), and (2.4.3). Measurements are taken from the integrating sphere within the luminance range of 0 to

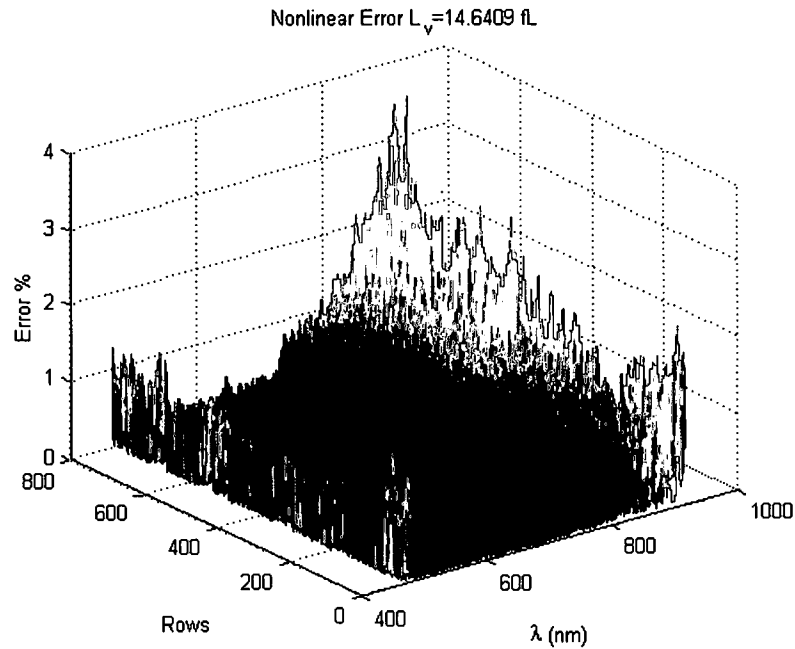
634fL and the non-linear error is computed after spectral filtering. Figures 2.6.1 to 2.6.3 display the error for different light levels.



**Figure 2.6.1: Non-linear Error at 564fL**



**Figure 2.6.2: Non-linear Error at 203fL**



**Figure 2.6.3: Non-linear Error at 15fL**

For all cases, the error remains less than 4%. This degree of non-linear error is tolerated for the applications of this data.

## 2.7 Data Collection

With the system set up and the calibration process in place, the discussion now turns to actual data collection of a particular scene. The goal of this research is to test the effectiveness of various target and change detection algorithms in various scene conditions. Specifically, the effects of illumination changes and vegetation changes upon these algorithms are of interest. Vegetation changes can include dying grass, leaves senescing, or possibly the cut of the grass within the scene. An illumination change for the purposes of this research refers to varying solar positions producing non-uniform illumination and varying shadow regions within the scene. Diminished direct solar radiation due to cloud cover is another interesting change to consider. However, due to a

couple of factors, this instrument cannot produce viable data for those conditions. One factor deals with the length of time required for each data collect. Due to a somewhat long exposure time and slow processing speed of the computer, each data collect requires about 2 minutes and 40 seconds. With variable cloud cover, a constant scene illumination is difficult to achieve for this length of time. Secondly, the hyperspectral system possesses a low throughput resulting in a low SNR for reduced scene light levels. Consequently, data collects are limited to days in which no cloud cover exists. Table 2.7.1 summarizes the specifications associated with collected and calibrated data.

**Table 2.7.1: Data Specifications**

Spectral Range	460 nm – 900 nm
Spectral Resolution	~8 nm
GSD	~4 cm
Spectral Sampling	~3.6 nm
f/#	f/4
IFOV	0.24 mrad
Exposure Time	70 ms

The ground sample distance (GSD) is estimated using the known length of aluminum panels within the scene. The instantaneous field of view (IFOV) is determined using the focal length of the lens (50mm) and the pixel size (12 $\mu$ m). The rest of the specifications result from the data calibration process.

In order to perform the algorithm testing, a specific scene with natural and man-made objects present is set up to be monitored over an observation period stretching from late August 2005 to late May 2006. Figure 2.7.1 displays the scene used in this research as collected by the hyperspectral sensor.



**Figure 2.7.1: Scene Setup**



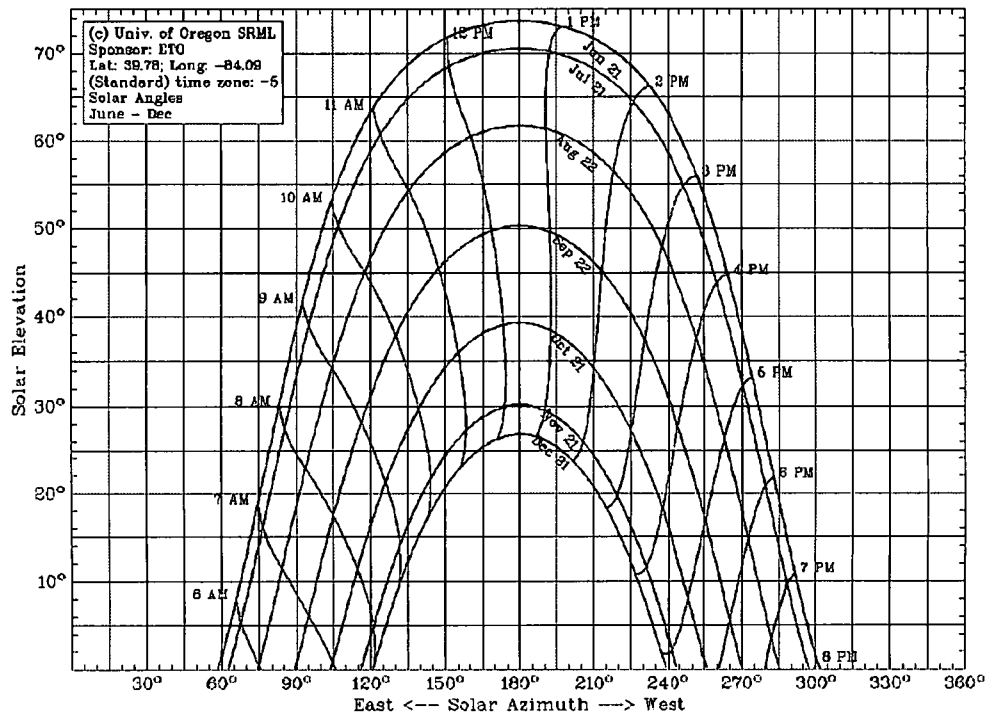
**Figure 2.7.2: Overhead Panel and Camera Locations**

A color image of the scene was produced using spectral bands from the hyperspectral data cube corresponding to red, green, and blue wavelengths. The scene consists of four aluminum panels each painted a distinct color residing within a predominantly vegetative environment. Off-the-shelf krylon spray paint was used for the black panel whereas the green and beige were painted using Chemical Agent Resistant Coating (CARC). The silver panel consists of flame sprayed aluminum, providing a highly reflective, diffuse surface. Each panel rests on an aluminum frame which allows for tilt adjustment. Both the orientation and tilt of each target remain relatively constant over the observation period in an attempt to eliminate undesired scene changes. These panels face a northeastern direction towards the hyperspectral camera as demonstrated in Figure 2.7.2. The camera looks down towards the scene from a tower about five stories from ground level. Due to the geographic orientation of the scene, shadows are often present within the data. As demonstrated later, shadows play a dominant role in scene changes due to lower solar elevation angles during fall and winter months. Solar elevation and azimuth

angles are recorded for each data collection using a solar position calculator found at the National Oceanic and Atmospheric Administration (NOAA) website.<sup>18</sup> This website allows the user to enter location coordinates in degrees, minutes, and seconds along with day and time information to produce the exact solar position for the provided data. The solar elevation is measured as degrees up from the horizon whereas the solar azimuth is measured as degrees clockwise from north.

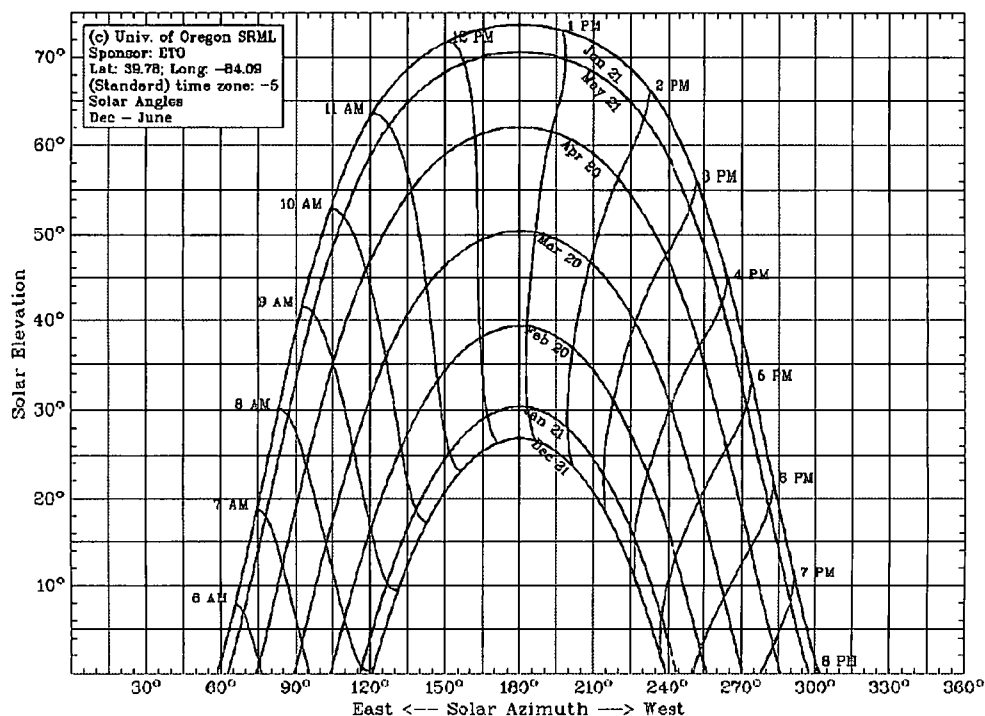
In an attempt to separate test variables, a data collection approach is devised. Ideally, one would like to separately test illumination and seasonal vegetation changes. By selecting specific solar elevation and azimuth angles of interest, one less variable changes between data collects. Unfortunately, one cannot separate illumination and seasonal changes completely. For instance, if data is collected at a time when the sun has an azimuth angle of  $180^\circ$  on two separate days, the solar elevation angles at those times will differ due to seasonal trajectory changes of the sun. However, one can investigate elevation illumination changes versus azimuth illumination changes by maintaining one constant. For this research, solar positions of interest are chosen as Solar Azimuth  $180^\circ$  (SA  $180^\circ$ ), Solar Azimuth  $120^\circ$  (SA  $120^\circ$ ), and Solar Elevation  $30^\circ$  (SE  $30^\circ$ ). The SA  $180^\circ$  position is chosen as it represents the point at which the sun reaches its maximum elevation for that given day. This peak angle varies from day to day. Figure 2.7.3 demonstrates seasonal changes in sun trajectory from June to December as produced by the University of Oregon Solar Radiation Monitoring Laboratory (SRML).<sup>19</sup>





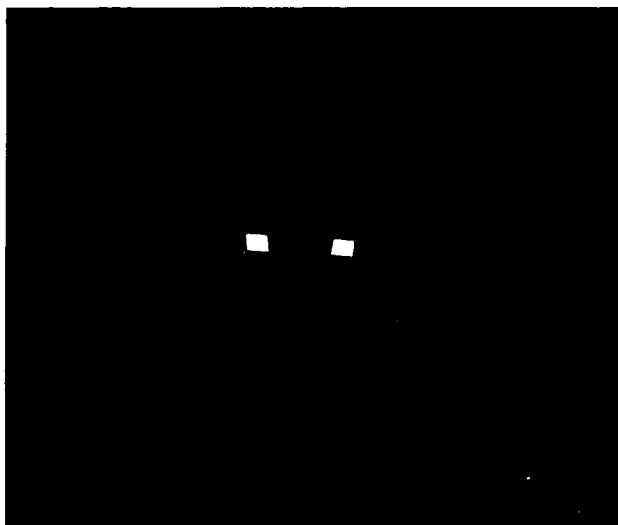
**Figure 2.7.3: Seasonal Solar Trajectory (June – December)**

The SA 120° position is chosen to produce illumination of the panels from the east since they face a northeastern direction. SE 30° is chosen as an elevation angle capable of being achieved throughout most of the year. The only month of the year in which the sun does not reach a peak at or above 30° is December. This low elevation angle already produces shadowing problems due to the location of trees behind the panels. Any lower angle would not produce sufficient illumination conditions. As expected, the solar trajectory follows a similar pattern from December through June as demonstrated in Figure 2.7.4.



**Figure 2.7.4: Seasonal Solar Trajectory (December – June)**

Both general and specific illumination changes can be examined using the various data collect times. For example, the Figures 2.7.5 and 2.7.6 demonstrate the illumination differences between SE 30° and SA 180° within the same day.



**Figure 2.7.5: SE 30°, SA 106°  
September 2, 2005**



**Figure 2.7.6: SE 58°, SA 180°  
September 2, 2005**

In this example, both the solar azimuth and elevation angle are changing from one scene to the next. In Figure 2.7.5, the sun radiates at a lower elevation angle and further east resulting in greater reflection from the targets but darker vegetation. Conversely, Figure 2.7.6 demonstrates a more uniform scene illumination. Figures 2.7.7 and 2.7.8 demonstrate a case in which only SA changes.

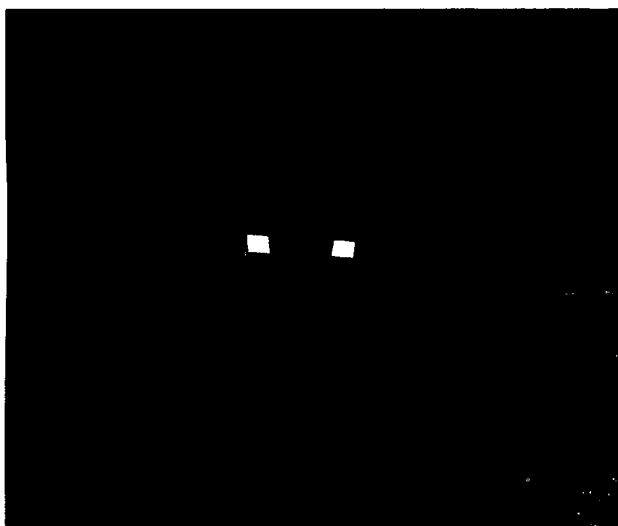


Figure 2.7.7: SE 30°, SA 106°  
September 2, 2005

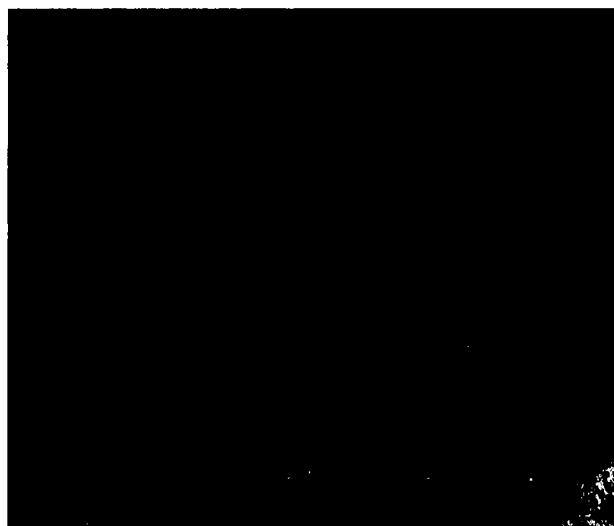


Figure 2.7.8: SE 30°, SA 120°  
September 30, 2005

Both scenes have the same solar elevation. However, in the second scene the sun has moved further south, increasing the presence of tree shadows within the scene.

Determining the effects of seasonal changes upon the algorithms remains a bit more difficult. As evident in the solar path charts, solar position varies over the seasons. In fact, the senescing of leaves is directly related to the decrease in temperature and sunlight in the later months of the year. Consequently, looking at seasonal vegetation changes often requires dealing with illumination changes as well. For example, Figures 2.7.9 and 2.7.10 possess the same SA but different SE.



**Figure 2.7.9: SE 61°, SA 180°  
August 25, 2005**

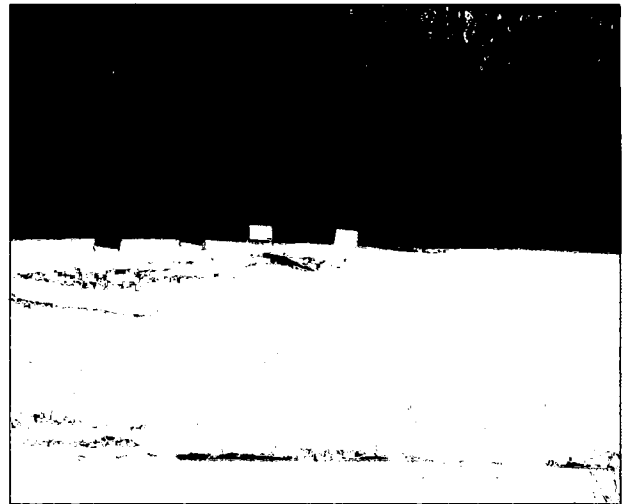


**Figure 2.7.10: SE 35°, SA 180°  
November 3, 2005**

Due to a lower SE in Figure 2.7.10, shadows from the trees exist in addition to the leaves senescing. The grass is not as healthy in Figure 2.7.9 as well. In some rare cases, the comparison of only vegetation changes is possible. Looking at the solar path charts of Figures 2.7.3 and 2.7.4, the months of August and September have similar solar paths as March and April. Portions of these months provide opportunities to observe seasonal vegetation changes with similar solar illumination present. Unfortunately, due to inclement weather not many suitable data sets exist for March and early April. Some good examples do exist as demonstrated in Figures 2.7.11 and 2.7.12.



**Figure 2.7.11: SE 50°, SA 177°  
September 22, 2005**



**Figure 2.7.12: SE 45°, SA 180°  
March 7, 2006**

The two images display similar illumination conditions. However, leaves do not exist and the grass is dead in the March scene. In this case, the changes are mainly vegetative.

Numerous other data sets exist in addition to the ones already displayed here. In addition to the regular solar position data collects at SA 180°, SA 120°, and SE 30°, a collection set was created to extensively compare illumination changes on the scene without any vegetative changes. A hypercube was collected every half hour from 8am through 7pm to capture various illumination conditions with 12 sets for the normal scene setup and 10 sets with the panels tilted back to simulate change. This extensive set was compiled using May 8, 9, 20, and 22 since the entire collection could not be performed in one day. In all, over 120 different data collection sets exist from late August 2005 to late May 2006. In a number of data sets, artificial changes have been added to acts as targets or changes for testing specific algorithms. These artificial changes include tilting the panels back to simulate their disappearance, covering selected panels with tarps to simulate color change or an attempt at concealing them, or completely removing the targets. Figures 2.7.13 through 2.7.15 demonstrate these cases.



**Figure 2.7.13: Panel Tilt Change**



**Figure 2.7.14: Tarp Change**



**Figure 2.7.15: Panels Removed**

Overall, the data collection may not be exhaustive for this research topic but it provides a wide range of scenarios to examine. Only a portion of the data collected is used for testing purposes. The rest may be valuable for future work or research.

## CHAPTER 3

### Hyperspectral Target and Change Detection Theory

After collecting and calibrating numerous data sets, a vast collection of hyperspectral data cubes or “hypercubes” exist. Generally, these hypercubes are used to spectrally identify materials within a scene. Applying statistical models to the data allows for the development of specific target detection algorithms. These algorithms may or may not require prior information about the targets. For algorithms requiring *a priori* information, atmospheric compensation may be necessary in order to accurately model target scene spectra. In addition, a Principal Component transformation is often employed to reduce overall data size and the number of computations necessary to perform detection. The modeling and theory behind target detection and data transforms is discussed here.

#### 3.1 Hyperspectral Data Modeling

Traditional target and change detection techniques involve statistical hypothesis testing. In order to analyze the hypercubes mathematically to detect targets, one must model the data in a statistical sense. Each hyperpixel  $\mathbf{x}_i$  within a data cube as described by Figure 1.1.3 can be thought of as a realization of a random vector. These column vectors are of length  $K$  where  $K$  is the number of spectral bands. The hypercube in its original form possesses two spatial dimensions and one spectral dimension. For processing purposes, the random vectors are often viewed in the data matrix form of

$$\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_N] \quad (3.1.1)$$

where  $N$  is the number of random vector samples (hyperpixels) and the dimensionality is now  $K \times N$ . Being random vectors, the probability density function (PDF)  $f(\mathbf{x}_i)$  describes the variability of the data. Within a hyperspectral scene, a differentiation is often made between objects of interest and everything else. The objects of interest, or targets, typically do not comprise a large portion of the scene. The rest of the scene, or background, is often modeled using a multivariate Gaussian distribution.<sup>20</sup> As such, the PDF for the random vector  $\mathbf{x}_i$  is defined as

$$f(\mathbf{x}_i) = \frac{1}{(2\pi)^{K/2} |\mathbf{C}|^{1/2}} \exp \left\{ -\frac{1}{2} [\mathbf{x}_i - \mathbf{m}]' \mathbf{C}^{-1} [\mathbf{x}_i - \mathbf{m}] \right\} \quad (3.1.2)$$

where  $\mathbf{C}$  is the covariance matrix,  $\mathbf{m}$  is the mean vector, and  $'$  is used to denote the transpose operation. The mean vector and covariance matrix are determined from the data using

$$\mathbf{m} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i = \frac{1}{N} \mathbf{X} \mathbf{u} \quad (3.1.3)$$

and

$$\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})' = \frac{1}{N-1} \mathbf{X} \mathbf{X}' - \mathbf{m} \mathbf{m}' \quad (3.1.4)$$

where  $\mathbf{u}$  is a unit row vector of length  $N$  and  $i$  is the pixel index. Both  $\mathbf{m}$  and  $\mathbf{C}$  represent sample statistics since they are derived through averaging of the data. However, assuming enough background pixels exist and targets do not comprise a large portion of the scene, the sample statistics do provide an unbiased estimate of the true mean and covariance of the data. Both the mean vector and covariance matrix are extremely useful in data transformations and detection algorithms. In particular, the



covariance matrix describes the spread of data and any correlation existing between bands.

### 3.2 Target Detection Theory

The goal of many hyperspectral image processing algorithms is to assign a label to every hyperpixel within the scene. For target detection algorithms, each pixel is labeled as either target or background. Single-instance target detection algorithms rely upon the statistics of only the scene being processed. These algorithms fall into two categories: (1) those in which no knowledge of the target exists or (2) those in which some prior knowledge about the target statistics is available. Simple anomaly detection using a Mahalanobis distance (M-distance) falls into the first category whereas spectral matched filtering (SMF) falls in the second. Multiple-instance target detection, or change detection, uses scene data from an earlier time to attempt to increase the probability of successful detection. These algorithms typically involve a linear prediction of time-2 data using the statistics of both data sets. Chronochrome (CC) and Covariance Equalization (CE) represent two linear prediction methods used for change detection.

Target detection algorithms are derived using statistical decision procedures which rely upon hypothesis theory, data modeling, and conditional probability density functions (CPDF).<sup>20</sup> In order to define a decision statistic, a likelihood ratio of the CPDFs is developed

$$l(\mathbf{x}_i) = \frac{f_1(\mathbf{x}_i | H_1)}{f_0(\mathbf{x}_i | H_0)} > \eta \quad (3.2.1)$$

where  $f_1(\mathbf{x}_i | H_1)$  is the CPDF of  $\mathbf{x}_i$  under the hypothesis  $H_1$  that it corresponds to a target and  $f_0(\mathbf{x}_i | H_0)$  is the CPDF of  $\mathbf{x}_i$  under the hypothesis  $H_0$  that it corresponds to

background. This condition tests the likelihood of a given pixel being a target. If the likelihood is greater than some threshold  $\eta$ , the pixel is declared a target. Since many of the data models utilize a multivariate Gaussian, a log likelihood ratio is often used to define a detection statistic

$$d(\mathbf{x}_i) = \log[f_1(\mathbf{x}_i | H_1)] - \log[f_0(\mathbf{x}_i | H_0)] > \eta \quad (3.2.2)$$

where again a threshold is applied as the decision boundary.<sup>21</sup> The specific form which the detection statistic takes depends upon the models used for the CPDFs. Typically, one uses the multivariate Gaussian model for background. Target models vary from case to case.

Simple anomaly detection generally attempts to select spectrally abnormal pixels within a given scene. Using the hypothesis model, the detection problem is modeled as

$$\begin{aligned} H_0 : \quad \mathbf{x}_i &= \mathbf{b} \\ H_1 : \quad \mathbf{x}_i &= \mathbf{s} \end{aligned} \quad (3.2.3)$$

where  $H_0$  corresponds to the hypothesis of the pixel being background  $\mathbf{b}$  and  $H_1$  corresponds to the hypothesis of the pixel being a target  $\mathbf{s}$ . For this specific algorithm, the CPDF under  $H_0$  is modeled using the multivariate Gaussian

$$f_0(\mathbf{x}_i | H_0) = \frac{1}{(2\pi)^{K/2} |\mathbf{C}|^{1/2}} \exp \left\{ -\frac{1}{2} [\mathbf{x}_i - \mathbf{m}]^T \mathbf{C}^{-1} [\mathbf{x}_i - \mathbf{m}] \right\} \quad (3.2.4)$$

with mean vector and covariance matrix described by (3.1.3) and (3.1.4). No *a priori* knowledge of the target exists resulting in a uniform CPDF of

$$f_1(\mathbf{x}_i | H_1) = \text{constant} \quad (3.2.5)$$

for the operating range of the sensor. Using the log likelihood ratio, the resulting detection statistic is

$$d(\mathbf{x}_i) = [\mathbf{x}_i - \mathbf{m}]^T \mathbf{C}^{-1} [\mathbf{x}_i - \mathbf{m}] \quad (3.2.6)$$

where constants are ignored. This statistic represents a Mahalanobis distance measure for vectors. A Mahalanobis distance differs from a typical Euclidean measure in that the distance is measured relative to the standard deviation of the background. Consequently, a threshold applied to this statistic represents an ellipsoidal surface with the principal axis oriented in the direction of maximum data spread.<sup>22</sup> Figure 3.2.2 displays a 2-D scatter plot of the colored regions in image in Figure 3.2.1.

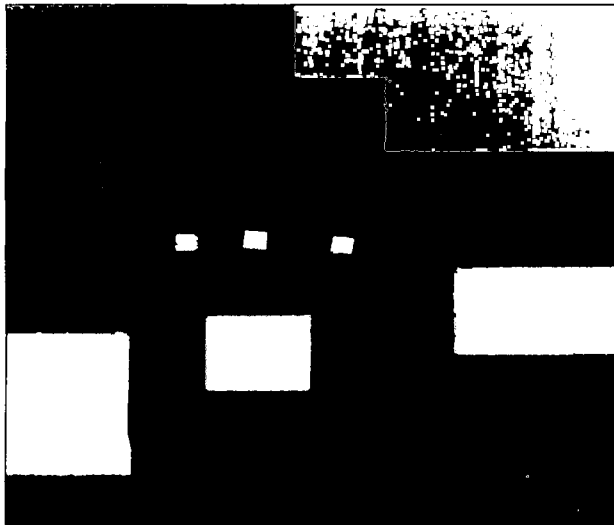


Figure 3.2.1: Segmented Scene Image

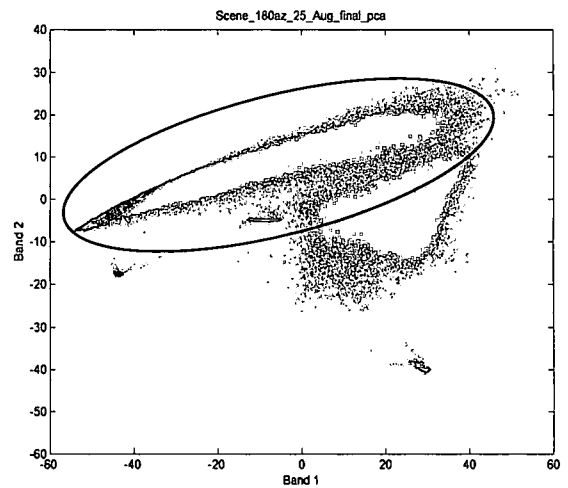


Figure 3.2.2: Scatter Plot with Ellipsoid Decision Surface

Using the trees as background, an ellipsoidal decision surface has been inserted into the scatter plot to demonstrate how the M-distance measure works. Anything outside of the decision surface would be considered a target. Adjusting the threshold varies the size of the ellipsoid and consequently the number of pixels declared a target. The two aluminum panels on the right lie completely outside the standard spread of data. As a result, the M-distance measure easily identifies them as targets. The actual algorithm is extended to many more bands allowing for better distinction between background and targets. One

significant weakness associated with this algorithm is the detection of anything abnormal within the scene, regardless of whether or not it should be considered a target. In this case, the majority of the scene is vegetation. M-distance measure identifies any man-made object within the scene as a target. Consequently, a more selective algorithm is needed to avoid false detections.

In some cases, a certain degree of information concerning the target's spectral signature is available. With this knowledge in place, one still uses the same hypothesis method

$$\begin{aligned} H_0 : \mathbf{x}_i &= \mathbf{b} \\ H_1 : \mathbf{x}_i &= \mathbf{s} \end{aligned} \quad (3.2.7)$$

where in this case both the background and target are modeled as random variables.

Using the multivariate Gaussian distribution, the CPDFs of both are described by

$$f_0(\mathbf{x}_i | H_0) = \frac{1}{(2\pi)^{K/2} |\mathbf{C}_b|^{1/2}} \exp \left\{ -\frac{1}{2} [\mathbf{x}_i - \mathbf{m}_b]^t \mathbf{C}_b^{-1} [\mathbf{x}_i - \mathbf{m}_b] \right\} \quad (3.2.8)$$

and

$$f_1(\mathbf{x}_i | H_1) = \frac{1}{(2\pi)^{K/2} |\mathbf{C}_s|^{1/2}} \exp \left\{ -\frac{1}{2} [\mathbf{x}_i - \mathbf{m}_s]^t \mathbf{C}_s^{-1} [\mathbf{x}_i - \mathbf{m}_s] \right\} \quad (3.2.9)$$

where background and target statistics are denoted by  $(\mathbf{m}_b, \mathbf{C}_b)$  and  $(\mathbf{m}_s, \mathbf{C}_s)$  respectively. Background statistics are found through estimation as before. However, a large enough sample set for the target is typically unavailable. Consequently, the background and target covariance are often assumed equal.<sup>20</sup> A reference target signature is used as the target mean. The log likelihood ratio results in a detection statistic of

$$d(\mathbf{x}_i) = [\mathbf{x}_i - \mathbf{m}]^t \mathbf{C}^{-1} [\mathbf{x}_i - \mathbf{m}] - [\mathbf{x}_i - \mathbf{s}]^t \mathbf{C}^{-1} [\mathbf{x}_i - \mathbf{s}] \quad (3.2.10)$$

where  $\mathbf{C}$  is the background covariance and  $\mathbf{s}$  is the reference target spectrum. Reducing some terms results in

$$d(\mathbf{x}_i) = \mathbf{m}'\mathbf{C}^{-1}\mathbf{m} + \mathbf{s}'\mathbf{C}^{-1}\mathbf{s} + 2[\mathbf{s} - \mathbf{m}]'\mathbf{C}^{-1}\mathbf{x}_i \quad (3.2.11)$$

Ignoring terms not dependent on  $\mathbf{x}_i$  and subtracting a constant results in

$$d(\mathbf{x}_i) = 2[\mathbf{s} - \mathbf{m}]'\mathbf{C}^{-1}\mathbf{x}_i - 2[\mathbf{s} - \mathbf{m}]'\mathbf{C}^{-1}\mathbf{m} \quad (3.2.12)$$

which finally simplifies to the commonly used form

$$d(\mathbf{x}_i) = [\mathbf{s} - \mathbf{m}]'\mathbf{C}^{-1}[\mathbf{x}_i - \mathbf{m}] \quad (3.2.13)$$

by ignoring multiplicative constants. The SMF essentially projects each data vector  $\mathbf{x}_i$  in the direction of the reference target spectrum  $\mathbf{s}$ . Any vector similar to a target produces a greater detection statistic value. The threshold applied to this statistic represents a hyperplanar decision surface.<sup>23</sup> SMF offers the advantage of detecting specific types of targets resulting in fewer false alarms if covariance and Gaussian distribution assumptions hold true. However, the algorithm does require some prior knowledge of the target spectrum.

### 3.3 Change Detection Theory

Change detection acts as an extension to single-instance target detection. When multiple instances of scene data exist, the use of this class of algorithms may provide advantages over single-instance methods. General change detection algorithms attempt to identify abnormal changes from one instance of the scene to the next. Ideally, the algorithms can differentiate between natural illumination and vegetation changes versus abnormal ones such as objects moving or drastic spectral transformations. This research attempts to evaluate and identify weaknesses associated with these methods. The change detection process involves linear prediction of time-2 from time-1 data described by

$$\hat{\mathbf{y}}_i = \mathbf{T}\mathbf{x}_i + \mathbf{d} \quad (3.3.1)$$

where  $\mathbf{T}$  is a linear gain matrix,  $\mathbf{d}$  is the offset vector,  $\mathbf{x}_i$  is time-1 data, and  $\hat{\mathbf{y}}_i$  is the predicted time-2 data.<sup>23</sup> For mean-subtracted data, the offset term vanishes. Essentially, the methods discussed here predict time-2 data using statistics from time-1 and time-2 data. The prediction is then compared to the actual time-2 data to determine an error statistic described by

$$\boldsymbol{\varepsilon}_i = \hat{\mathbf{y}}_i - \mathbf{y}_i \quad (3.3.2)$$

where  $\mathbf{y}_i$  is the actual time-2 data. A subsequent algorithm or decision process is applied to the error data to determine targets. Figure 3.3.1 summarizes the process.

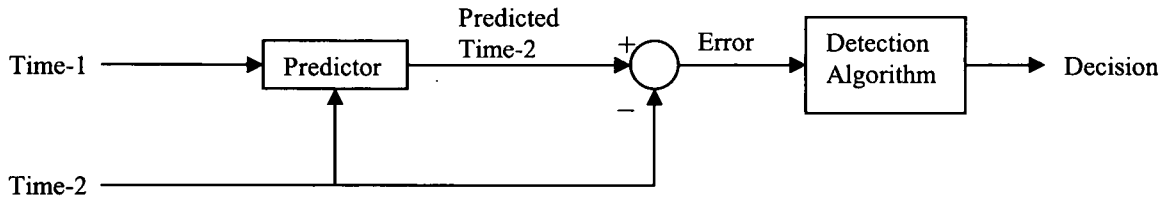


Figure 3.3.1: Change Detection Process

An effective predictor models the background very well and suppresses it in the error image. Consequently, abnormal changes stand out in the error statistic. Typically, the error statistic data is evaluated using a single instance method such as the M-distance discussed earlier to determine targets. This simple change detection technique takes the form

$$d(\boldsymbol{\varepsilon}_i) = (\boldsymbol{\varepsilon}_i - \mathbf{m}_\varepsilon)^t \mathbf{E}^{-1} (\boldsymbol{\varepsilon}_i - \mathbf{m}_\varepsilon) \quad (3.3.3)$$

where  $\mathbf{E}$  is the error covariance matrix and  $\mathbf{m}_\varepsilon$  is the mean error vector. Since this method requires no prior target information, it is often termed “blind” change detection.

Two methods discussed here, chronochrome and covariance equalization, vary in the linear prediction method applied. As such, each has advantages and limitations.

The chronochrome linear prediction method relies upon joint statistics of time-1 and time-2 data. From processing theory, the estimate represents the least-mean-square-error prediction.<sup>23</sup> The CC prediction matrix and offset are found using

$$\mathbf{T}_{CC} = \mathbf{C}_{xy} \mathbf{C}_x^{-1} \quad (3.3.4)$$

and

$$\mathbf{d}_{CC} = \mathbf{m}_y - \mathbf{C}_{xy} \mathbf{C}_x^{-1} \mathbf{m}_x \quad (3.3.5)$$

where  $\mathbf{C}_x$  is the covariance matrix of time-1 data and  $\mathbf{C}_{xy}$  is the cross-covariance matrix.

This cross-covariance matrix is determined using

$$\mathbf{C}_{xy} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \mathbf{m}_x)(\mathbf{y}_i - \mathbf{m}_y)^T \quad (3.3.6)$$

where  $\mathbf{m}_x$  and  $\mathbf{m}_y$  are the respective mean vectors of time-1 and time-2 data. In order to determine the gain and offset prediction terms, good image registration is necessary. As registration decreases, the cross-covariance estimate suffers. Since change detection requires image registration, this issue should not present a problem.

Covariance equalization represents a method in which excellent image registration is not required to produce valid estimation parameters. This method actually represents the entire class of linear predictors that produce an estimate possessing the same mean and covariance as the actual time-2 data.<sup>23</sup> The respective gain matrix and offset vector are found using

$$\mathbf{T}_{CE} = \mathbf{C}_y^{\frac{1}{2}} \mathbf{C}_x^{-\frac{1}{2}} \quad (3.3.7)$$

and

$$\mathbf{d}_{CE} = \mathbf{m}_y - \mathbf{C}_y^{\frac{1}{2}} \mathbf{C}_x^{-\frac{1}{2}} \mathbf{m}_x \quad (3.3.8)$$

where  $\mathbf{C}_y$  is the covariance matrix of time-2 data.  $\mathbf{C}_y^{\frac{1}{2}}$  and  $\mathbf{C}_x^{-\frac{1}{2}}$  are used as symbolic notation with the actual calculation of these parameters involving the use of eigenvectors and eigenvalues. Under the constraints applied for this solution, CE does not rely upon cross-statistics to perform estimation. As a result, this method handles poor registration better than CC. Ultimately, one must register data to perform change detection since a pixel by pixel subtraction occurs. In this sense, CE may not offer any advantage over CC. However, CE can present a good solution for spectrum evolution purposes. If a spectrum of a target exists in one scene, CE can produce an estimate of the same target spectrum in a completely unrelated scene. This spectrum evolution attempts to predict how the spectrum will appear in the scene. With CC relying on cross-covariance to perform estimation, spectrum evolution does not work for spatially unrelated scenes. Ultimately, if one performs blind change detection CC most likely offers a better solution. However, CE still provides benefits in change detection and spectral estimation.

### 3.4 Atmospheric Compensation

Some detection algorithms, such as SMF, require prior target spectral information. In most cases, target information on file must be converted to represent the scene spectrum. From the calibration discussion, each calibrated data vector  $\mathbf{x}_i$  represents a spectrum for a material in spectral radiance units. Typically for the VNIR portion of the electromagnetic spectrum, reflectance of an object is the specific parameter of interest. The pupil plane spectral radiance measured by the hyperspectral sensor is



affected not only by the material reflectance but by atmospheric path radiance and transmission parameters as well. Consequently, one must compensate for these atmospheric characteristics in order to relate the measured spectral radiance values of  $\mathbf{x}_i$  to actual material reflectance values. Each material has a lab reflectance signature which is measured in a lab to eliminate atmospheric effects. These lab spectra are often used as the *a priori* information for target detection algorithms. An accurate atmospheric compensation (AC) method converts lab reflectance spectra to scene radiance and vice-versa. Methods used for compensation typically fall into four categories: (1) scene-derived corrections (2) radiative transfer models (RTM) (3) ground-calibration methods and (4) hybrid radiative-transfer-ground methods.<sup>26</sup> Ground-calibration methods use knowledge of materials within the scene to perform the compensation. If a material with a known lab reflectance exists within the scene, one can determine the atmospheric parameters by comparing the measured spectral radiance to the expected lab spectrum. Ground-calibration methods are discussed here.

The empirical line method (ELM) attempts to model atmospheric parameters using materials within a scene with known reflectance spectra.<sup>24</sup> With many known materials in the scene, a linear regression is performed to estimate atmospheric path radiance and transmission using gain and offset terms. In this manner, reflectance data is estimated from scene data and vice versa. The transformation is performed using

$$\mathbf{x}_i = \mathbf{A}\mathbf{p}_i + \mathbf{b} \quad (3.4.1)$$

where  $\mathbf{p}_i$  is the  $K \times 1$  lab reflectance vector corresponding to the material in pixel  $i$ ,  $\mathbf{A}$  is the  $K \times K$  diagonal gain matrix, and  $\mathbf{b}$  is the  $K \times 1$  offset vector. For the case when two

known materials appear, the gain and offset are calculated using slope and intercept equations for a line. These appear in the form of

$$\mathbf{A}_{kk} = \frac{(x_n)_k - (x_m)_k}{(\rho_n)_k - (\rho_m)_k} \quad (3.4.2)$$

and

$$\mathbf{b}_k = \frac{(x_m)_k(\rho_n)_k - (x_n)_k(\rho_m)_k}{(\rho_n)_k - (\rho_m)_k} \quad (3.4.3)$$

where  $k$  is the spectral index and  $m$  and  $n$  represent the two pixels chosen as the materials used for ELM with their corresponding reflectance vectors. Limitations of this method include the requirement of *a priori* knowledge of in-scene materials and the assumption of uniform atmosphere and illumination across the image.<sup>24</sup>

More specific applications of the ELM look for common scene constituents that regularly appear in most scenes. The compensation process becomes more automated if one can identify specific materials without using ground truth. Vegetation represents one easily identifiable material which appears in many hyperspectral scenes and is often used as one of the ELM materials. Healthy vegetation demonstrates a sharp rise in reflectance in the 700nm range. The low reflectance prior to this spectral point is attributed to absorption by chlorophyll. The cutoff for this chlorophyll absorption exists around 700nm leaving only the high reflectance resulting from the internal structure of leaf. This sharp rise in reflectance is often called the “red edge” of vegetation. Many algorithms take advantage of this red edge to identify vegetation automatically within a scene. In particular, the Normalized Difference Vegetation Index (NDVI) is often used as a vegetation indicator.<sup>25</sup> NDVI for a pixel is described by

$$NDVI_i = \frac{(x_i)_{ir} - (x_i)_{red}}{(x_i)_{ir} + (x_i)_{red}} \quad (3.4.4)$$

where *ir* and *red* are the spectral indices corresponding to 860nm and 660nm respectively. The NDVI is applied for each hyperpixel in the scene and a threshold is set to create a mask indicative of healthy vegetation. To better understand NDVI, a scatter plot is useful. Figure 3.4.1 displays a scatter plot for a typical scene using the two bands of interest.

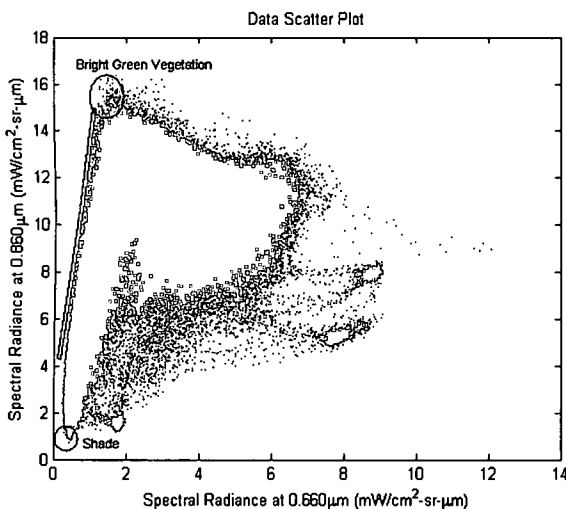


Figure 3.4.1: NDVI Theory

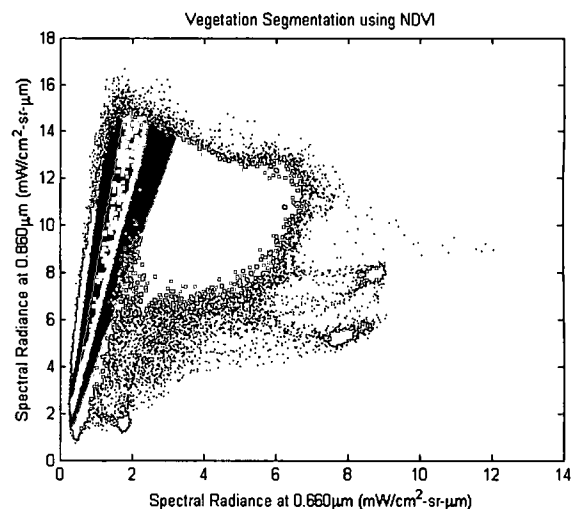
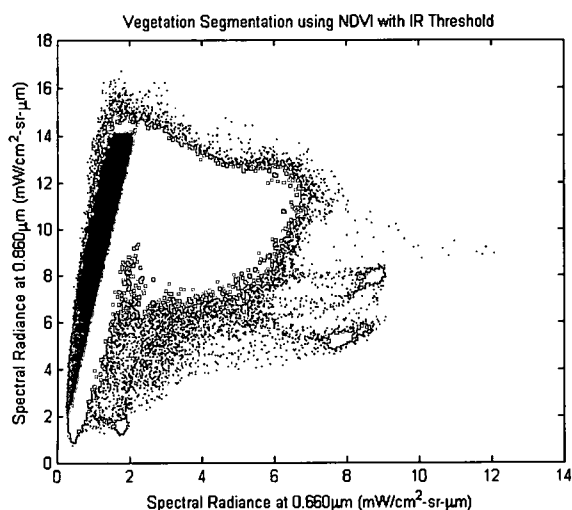


Figure 3.4.2: Varying Threshold Levels

The NDVI method attempts to isolate the pixels located along the vegetation edge displayed in green in Figure 3.4.1. Essentially, the threshold applied for NDVI corresponds to the slope of a line passing through the origin of the data scatter plot. Anything above the line corresponds to vegetation. Figure 3.4.2 demonstrates this application by applying different threshold levels to the data. Within the scatter plot, the regions overlap with blue representing the lowest threshold. Generally speaking, as the threshold is increased, fewer pixels are selected as representative of vegetation. The normalizing factor in NDVI removes the brightness constituent from the equation. Consequently, there is no differentiation made between vegetation in the shade and bright

vegetation. Bright vegetation is more desirable as a true representation. By applying a threshold to the infrared band (860nm), one can select bright vegetation. Figure 3.4.3 demonstrates this method.



**Figure 3.4.3: NDVI with IR threshold**



**Figure 3.4.4: Vegetation Scene Pixels**

The majority of the bright vegetation pixels are found in the top right portion of the scene in Figure 3.4.4. The infrared threshold version of NDVI works best in selecting vegetation pixels. Once the pixels are identified, a mean vegetation spectrum is estimated through averaging.

After finding vegetation within the scene, another reference material is needed for the ELM. A refined ELM attempts to model a zero-reflectance object within the scene using RTM for use as the second material.<sup>26</sup> For the purposes of this experiment, a shade point is used to represent the zero-reflectance object. A couple of different techniques are employed to obtain a shade spectrum from a scene. One method involves selecting the lowest spectral radiance value for every band within the data to compose a shade spectrum. Another method determines which hyperpixel possesses the lowest spectral mean and chooses that pixel as representative of shade. The fact that shade exists within

the scene violates a general assumption of ELM but in most cases shade is unavoidable. Since vegetation and shade typically exist within a scene, these two materials offer an opportunity for automatic atmospheric compensation for scenes in which no *a priori* spatial knowledge exists.

The ELM is used on a number of different data sets to test the accuracy of the AC for varying scene conditions using a number of different materials for gain and offset determination. Figure 3.4.5 displays the lab reflectance spectra for materials of interest within the scene. The panels' lab spectra were determined using the handheld field spectrometer while the grass and deciduous tree spectra were taken from the ASTER spectral library made available by NASA's Jet Propulsion Laboratory.<sup>27</sup>

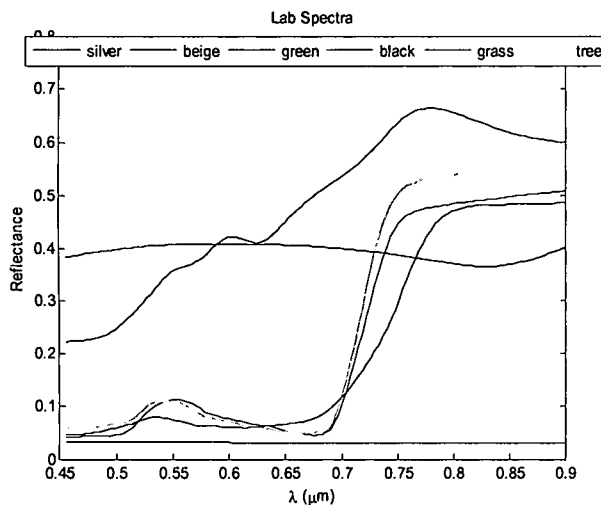


Figure 3.4.5: Lab Reflectance Spectra

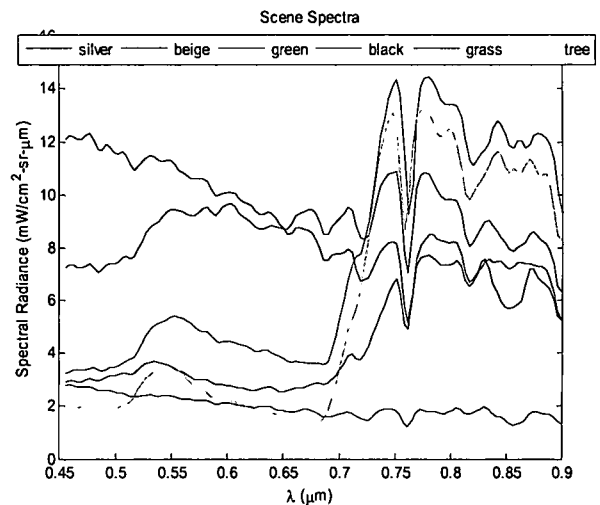


Figure 3.4.6: Scene Spectral Radiance Spectra

Figure 3.4.6 displays the corresponding scene spectral radiance vectors obtained by manually selecting pixels from the scene in Figure 3.4.7 thought to best represent the respective material. The relationship between the lab and scene spectra is readily seen, as are the atmospheric effects.



Figure 3.4.7: Well Illuminated Scene (1)

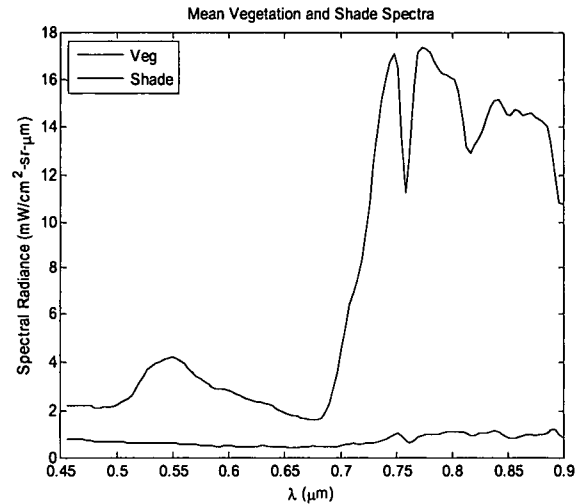


Figure 3.4.8: Mean Vegetation and Shade Spectra for (1)

In Figure 3.4.7, the panels receive direct solar illumination as do the tree tops. A mean vegetation spectrum is acquired using NDVI with the IR threshold. A shade spectrum is obtained using the hyperpixel with the lowest spectral mean. Figure 3.4.8 displays these spectra for the well illuminated scene. Using the ELM, atmospheric compensation is performed on the lab reflectance spectrum for the silver panel. For these results, the estimated shade spectrum is used in combination with other scene constituents as the second material.

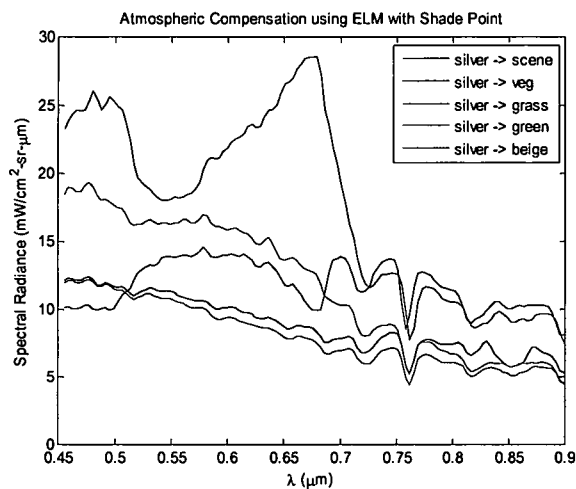


Figure 3.4.9: AC for Silver Panel using Shade for (1)

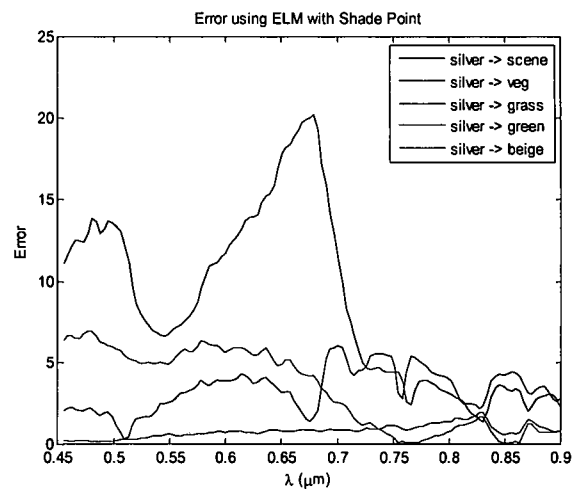


Figure 3.4.10: AC Error using Shade for (1)

Figure 3.4.9 demonstrates how well the compensation works for the silver panel using different materials as the second point for ELM. The beige panel in conjunction with the shade point accurately transforms the silver lab reflectance spectrum producing an accurate portrayal of the actual scene silver radiance spectrum. In this case, the grass and mean vegetation spectra do not work well. This poor compensation may result from inaccurate lab reflectance spectra for the grass and tree vegetation for this particular environment. The grass and deciduous reflectance spectra obtained from the ASTER database may not truly represent the grass and trees within the scene. Grass often displays characteristics of soil in scene data as well. Using a handheld field spectrometer may produce a more accurate representation of the grass and tree reflectance spectra which in turn would improve the compensation results. Non-uniform illumination conditions within the scene may play a role in unsuccessful compensation as well.

A robust AC method should be able to handle a variety of scenes. The next case tested uses a scene similar to the first but with an illumination change as seen in Figure 3.4.11.

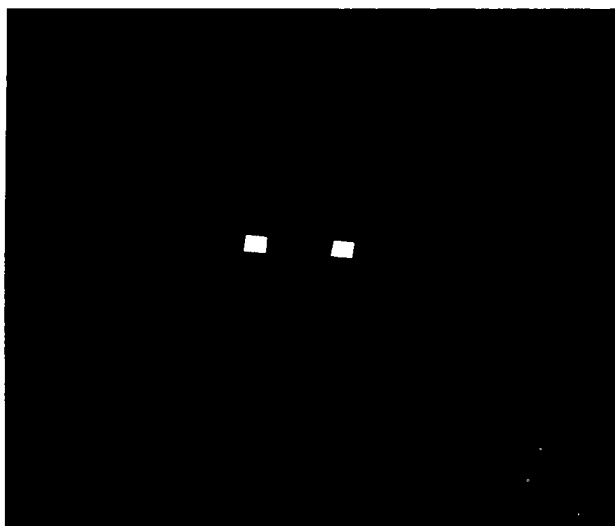


Figure 3.4.11: Scene with Illumination Change (2)

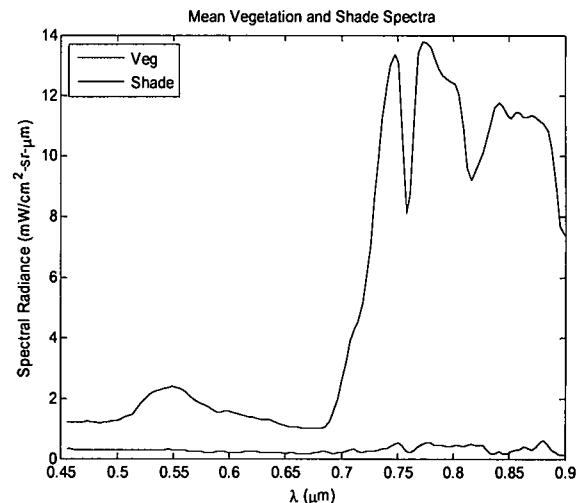


Figure 3.4.12: Mean Vegetation and Shade for (2)

For this scene, a lower solar elevation angle causes the panels to appear bright due to their orientation. However, the trees and grass are darker which is manifested in the mean vegetation spectrum of Figure 3.4.12. Using shade as the first material, the ELM is used as before and the results are shown in Figures 3.4.13 and 3.4.14.

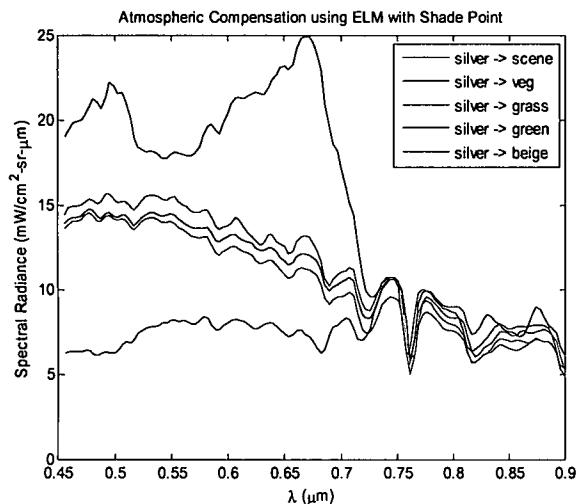


Figure 3.4.13: AC for Silver Panel using Shade for (2)

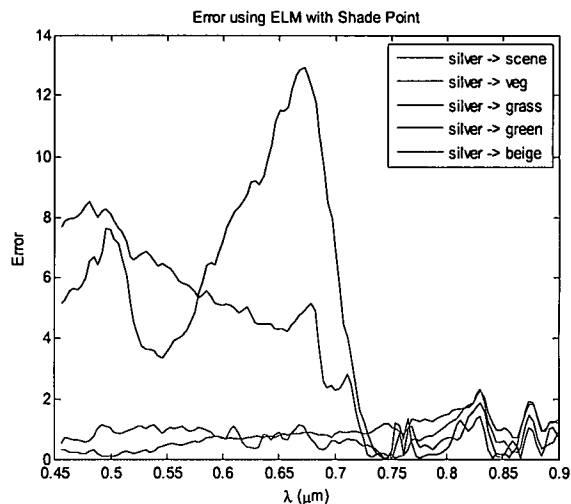


Figure 3.4.14: AC Error using Shade for (2)

In this example, grass and vegetation still do not perform well. Again, this error most likely results from inaccurate lab reflectance data for the two materials. The beige panel performs well as the second reference material as does the green panel.

The final example offers both illumination and seasonal scene changes. The scene in Figure 3.4.15 was taken in early November with the tree leaves senescing and a relatively low peak solar elevation. Consequently, the panels lie in the shadows and the tree leaves no longer possess a dominant red edge. As a result, the NDVI method identifies the grass within the scene as vegetation.



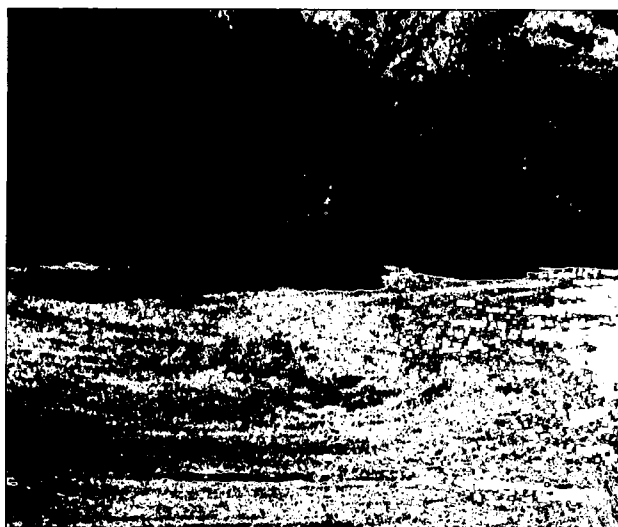


Figure 3.4.15: Scene with Seasonal Changes (3)

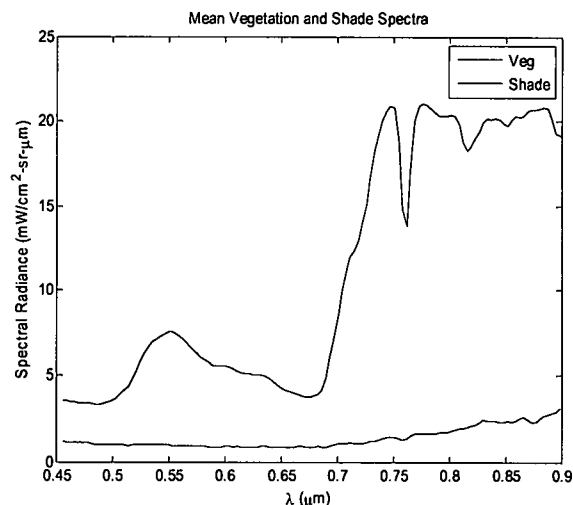


Figure 3.4.16: Mean Vegetation and Shade for (3)

Figure 3.4.16 verifies this theory as the mean vegetation spectrum resembles grass rather than the deciduous leaf spectra seen earlier. With this in mind, the same AC cases as before are analyzed. Figures 3.4.17 and 3.4.18 describe the results obtained using the shade spectrum with other materials.

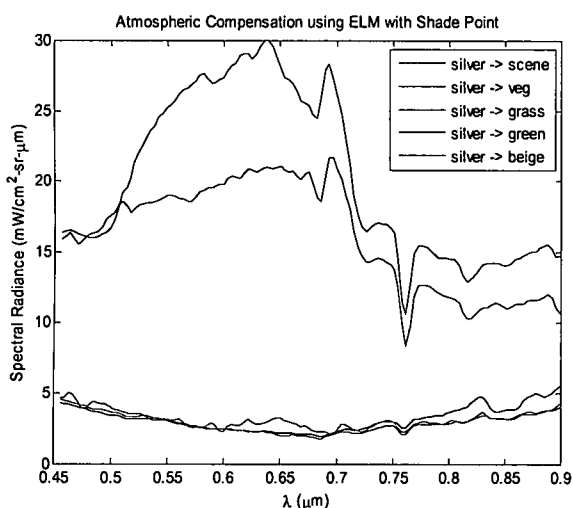


Figure 3.4.17: AC for Silver Panel using Shade for (3)

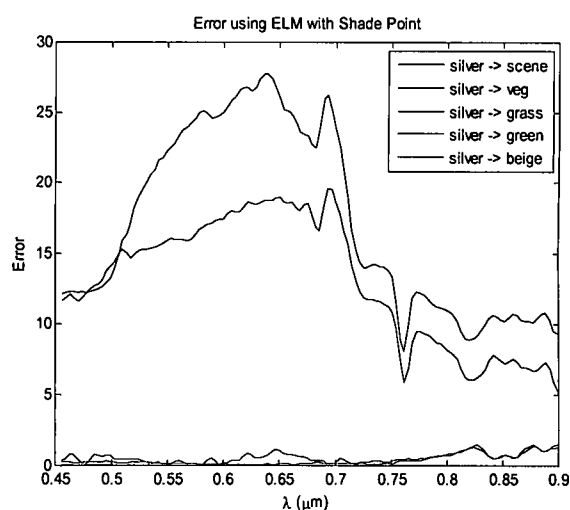


Figure 3.4.18: AC Error using Shade for (3)

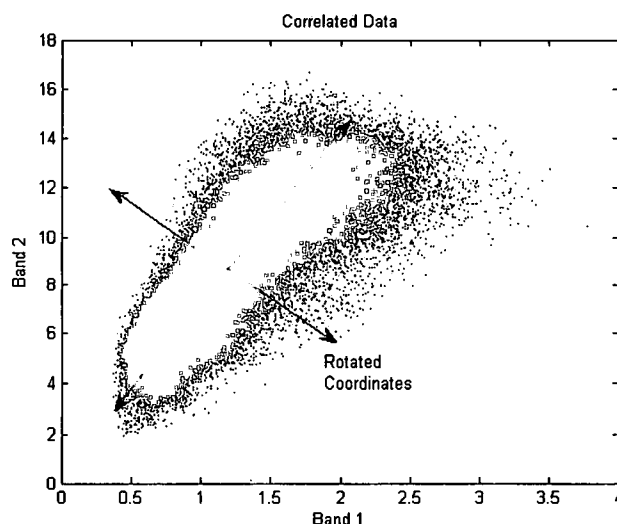
Since the mean vegetation spectrum no longer represents deciduous leaves, the error is amplified for that case. In actuality, the knowledge of leaf senescence would play a role

in the choice of lab spectrum used to represent the vegetation. The beige and green panels both produce accurate results even with shadows present.

The accuracy of in-scene atmospheric compensation is essential when trying to identify objects within a hyperspectral scene. For these examples, the ELM utilizing NDVI does not work very well due to inaccurate lab reflectance spectra for vegetation. Essentially, some knowledge of the scene constituents is necessary when trying to perform AC. Mismatching scene spectra with incorrect lab spectra produces significant error. When accurate lab spectra are used such as those gathered from the panels with a field spectrometer, the compensation produces excellent results. The ELM using an aluminum panel with a shade spectrum offers robustness as well. This combination works well for scenes demonstrating both illumination and seasonal changes. Consequently, AC for a panel within the scene is performed using another panel coupled with a shade spectrum. Since this method relies upon spatial truth information for the scene, it does not represent the best approach for automated compensation.

### **3.5 Principal Component Transformation**

Due to the size of each hyperspectral data file, employing detection algorithms creates a computational burden for processors. Consequently, methods are applied in an attempt to reduce data size without losing valuable information. Hyperspectral data is often over sampled in the spectral dimension to ensure small spectral features are not omitted. As a consequence, strong correlations exist between different spectral bands. Figure 3.5.1 displays an example of two spectral bands that display some degree of correlation.



**Figure 3.5.1: PC Transform Concept**

The two bands in the figure demonstrate some degree of linear dependence. As one band increases, the other generally does as well. Consequently, one dimension does not offer a great deal of information. A principal component (PC) transform attempts to take advantage of the correlations by producing a coordinate system in which specific dimensions offer the most information. Afterwards, dimensions not providing much info can be eliminated, reducing the overall size of the data set. For the data displayed above, a rotation would produce a coordinate system in which one band offers more info than the other. While in this case, both dimensions are still necessary, more extreme correlations allow for disposal of redundant bands.

Correlations within data manifest themselves as off-diagonal elements in the covariance matrix. Purely uncorrelated data results in a diagonal covariance matrix. In order to eliminate the redundant information offered by correlated spectral bands, the PC transformation is used. This transform essentially performs a coordinate rotation on the data in order to provide a representation in which no correlation exists.<sup>28</sup> The linear PC transform is modeled using

$$\tilde{\mathbf{x}}_i = \mathbf{G}\mathbf{x}_i \quad (3.5.1)$$

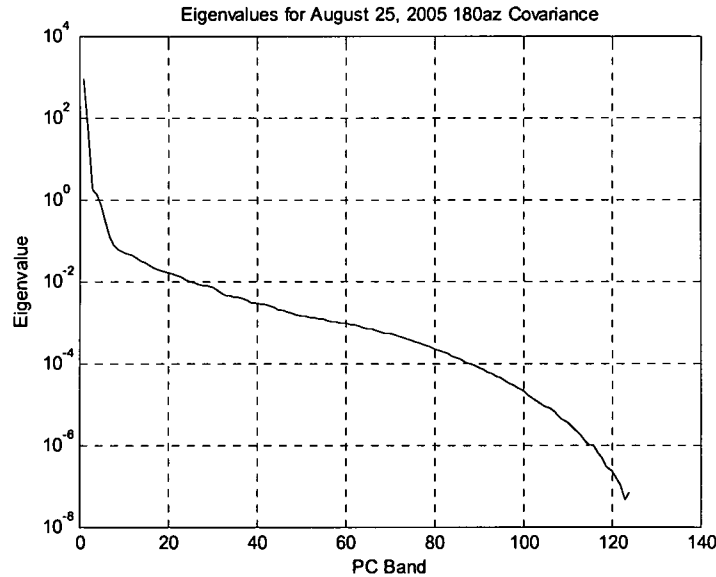
where  $\tilde{\mathbf{x}}_i$  is the random vector in PC space and  $\mathbf{G}$  is a  $K \times K$  matrix performing the coordinate rotation. The transform is subject to the constraint of a diagonal covariance matrix. From here, the statistics of the rotated data are determined to be

$$\tilde{\mathbf{m}} = \mathbf{G}\mathbf{m} \quad (3.5.2)$$

and

$$\tilde{\mathbf{C}} = \mathbf{G}\mathbf{C}\mathbf{G}' \quad (3.5.3)$$

where  $\tilde{\mathbf{m}}$  and  $\tilde{\mathbf{C}}$  represent the mean vector and covariance matrix of the transformed data. At this point, the transformation matrix  $\mathbf{G}$  is recognized as the transposed matrix of eigenvectors for the original covariance matrix  $\mathbf{C}$ . The corresponding eigenvalues of the covariance matrix describe the variance of the data in PC space. Figure 3.5.2 displays the eigenvalues for the August 25, 2005 SA 180° data set.

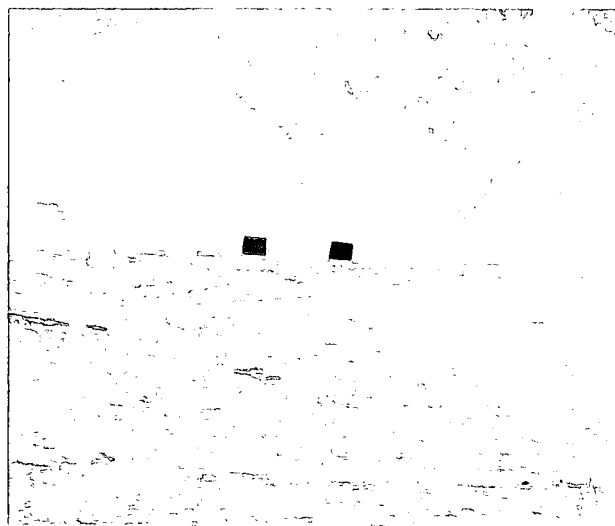


**Figure 3.5.2: Eigenvalues for August 25, 2005 SA 180° Covariance**

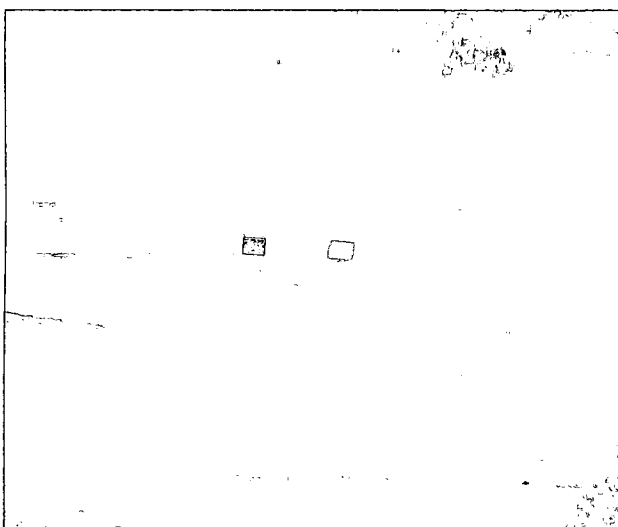
One can use these values to determine a sufficient number of PC bands to retain which still provide the majority of information present. Using the first ten PC bands retains about 99.94% of the data variance. By looking at various PC bands, one observes how past a certain point the image simply resembles noise. Figures 3.5.3 to 3.5.6 display different PC bands for the August data set.



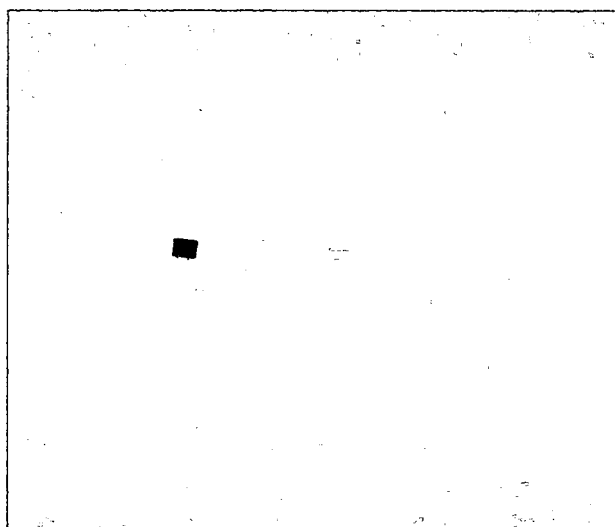
**Figure 3.5.3: PC Band 1**



**Figure 3.5.4: PC Band 2**



**Figure 3.5.5: PC Band 6**



**Figure 3.5.6: PC Band 10**

Beyond PC band 9, the image resembles noise. The sum of the eigenvalues for the remaining bands beyond 9 describes the total noise variance. For this data set, the

computed total noise variance is about 0.56. This knowledge is useful when discussing background suppression values. Conversion into the PC domain is often used to reduce data file size and reduce computational complexity. Some detection algorithms take advantage of PC properties to achieve better performance. For the purposes of this research, all the data is transformed to the PC domain and reduced to ten bands. In order to compare data sets, the same eigenvector transform is used on all the data sets. Since the scene is similar in all cases, one can assume the majority of information is retained for all scenes.

## **CHAPTER 4**

### **Experimental Results**

With knowledge of detection theory in place, specific experiments are run on the data sets collected. First, background suppression for illumination and seasonal changes using chronochrome and covariance equalization is analyzed. Afterwards, the effects of this background suppression at improving target detection are viewed. The advantage of change detection over single instance target detection is determined. SMF results for different cases are compared as well.

#### **4.1 Experimental Testing Methodology**

After discussing the theory behind target and change detection, a method for comparing these algorithms and assessing their effectiveness in dealing with illumination and vegetation changes is devised. The majority of target detection algorithms often refer to “background” and “target” in a general sense. In order to effectively test these algorithms, one must define these terms as they relate to the data sets present. The original scene is set up with both natural (trees and grass) and man-made (panels) objects. For suppression analysis in change detection, both the natural and man-made objects are considered background. This designation allows for comparison of suppression for both classes. For target detection using the single instance methods and change detection methods, the panels are considered targets. In the change detection case, the panels are removed in the time-1 scene. Consequently, their appearance in time-2 makes them

targets. In summary, the panels may represent either background or targets depending on the experiment performed.

From the discussion on change detection, the algorithms attempt to predict time-2 data using statistics from time-1 and time-2 data. This prediction essentially attempts to eliminate background within the scene to focus on any anomalous changes that may have occurred. Consequently, background suppression represents a good metric for measuring the effectiveness of the chronochrome and covariance equalization algorithms. Using this metric assumes that if the background is suppressed successfully, any changes in the scene are more easily detected. Sample variance of the background supplies a good measure for comparing suppression. Sample statistics are used in all cases since the true values cannot be determined.

For testing purposes, the first ten principal component bands are used as these constitute about 99.94% of the total variance of the data. For change detection, the error between the predicted and actual time-2 data given by (3.3.2) is used to detect targets. A comparison between time-2 variance and error variance provides a measure for background suppression. Since a single eigenvector transformation is used for all the data sets, the principal component data for each may not be completely uncorrelated. In order to produce an accurate measure of background suppression, each covariance matrix is first diagonalized using its eigenvectors and the corresponding eigenvalues are used as the variance measure.<sup>29</sup> The trace of the diagonalized covariance matrix now represents the total variance. Using this measure, background suppression is defined as

$$\Gamma = 10 \log \left( \frac{\hat{\sigma}_y^2}{\hat{\sigma}_\epsilon^2} \right) \quad (4.1.1)$$



where  $\hat{\sigma}_y^2$  is the total variance of time-2 data and  $\hat{\sigma}_e^2$  is the total variance of the error data. From here, background suppression for the entire image or just portions of the image can be determined. Since the scene contains three distinctly observable classes in the form of panels, trees, and grass, the suppression of each class is worthwhile to determine. Figure 4.1.1 describes each region of interest.



Figure 4.1.1: Regions of Interest

Green refers to the grass region, red, the panel region, and blue, the tree region. Typically, the statistics of the entire image are used to determine the linear transformation used for CC and CE described by (3.3.4), (3.3.5), (3.3.7), and (3.3.8). Consequently, a single linear transform is applied upon grass, trees, and aluminum panels to form the time-2 prediction. Looking at the suppression of an individual portion of the image provides insight into how well the prediction works for that respective region. In addition to this method, a separate linear prediction can be applied to each class. The suppression for this case is examined as well.

Specific data sets are chosen to determine the effectiveness for both CC and CE at suppressing background for illumination changes and seasonal changes. For daily illumination changes, the May 5, 2006 SA 120° data set is used as time-1 data. To test illumination change cases, the extensive data collection acquired on May 8, 9, 20, and 22 from 8:00 to 19:00 is used in an attempt to limit seasonal changes. The solar angles from this set vary from 82° to 280° for SA and 18° to 67° for SE providing a wide range of illumination conditions. To test seasonal changes, which include illumination and vegetation variability, the SA 180° data sets from late August 2005 through late May 2006 are used in conjunction with the August 25, 2005 SA 180° time-1 data set. Please refer to Appendix A for color images of all the data sets used for these experiments.

In addition to testing background suppression, the change detection algorithms are compared to the Mahalanobis distance measure used for single instance target detection. Comparisons are performed for the spectral matched filter using a respective target spectrum obtained in different ways as well. The comparison is performed for a spectrum obtained directly from the scene, from evolution using CC and CE on a spectrum from the reference scene, and from AC on a lab spectrum. For these comparisons, the beige and green aluminum panels are considered targets in separate cases, the other panels are ignored, and the rest is considered background. In order to use the advantages of the change detection algorithms, the reference image must possess some sort of drastic difference from the time-2 scene. The May 23, 2006 SA 180° data set has the panels removed with a fairly uniform scene illumination and acts as a good reference set. The signal to clutter ratio (SCR) for the detection statistic of each

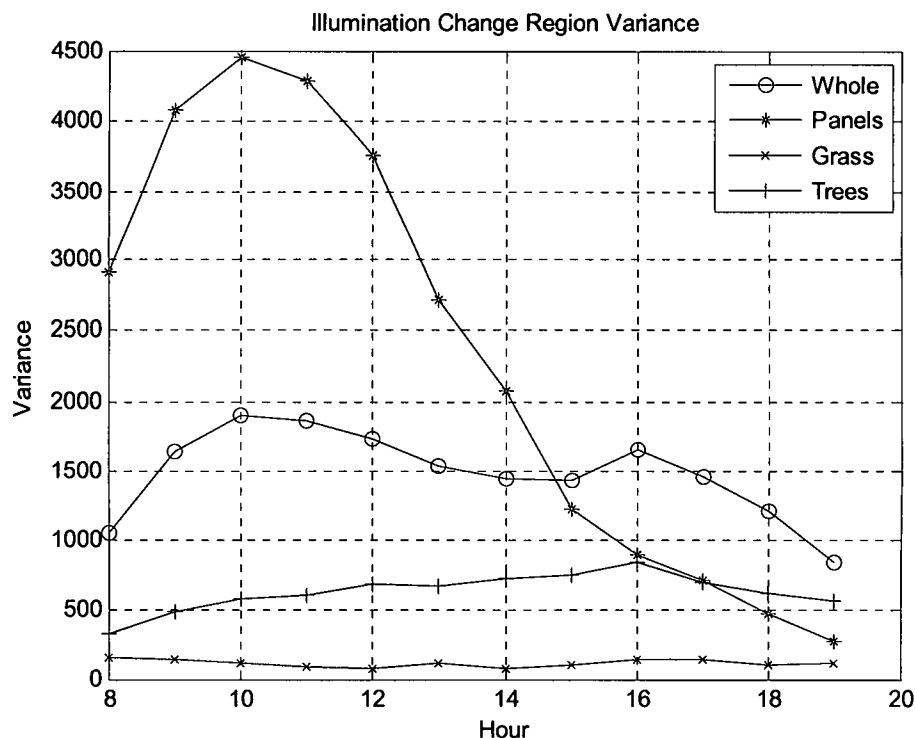
algorithm is used as the metric for comparing the different detection algorithms. For this study, SCR is defined as

$$SCR = \frac{\hat{\mu}_s - \hat{\mu}_b}{\hat{\sigma}_b} \quad (4.1.2)$$

where  $\hat{\mu}_s$  is the mean of the target pixels in the detection statistic,  $\hat{\mu}_b$  is the mean of the background pixels in the detection statistic, and  $\hat{\sigma}_b$  is the standard deviation of the background pixels in the detection statistic. Again, both illumination changes and seasonal changes are tested using the same data sets as the background suppression experiments.

#### 4.2 Illumination Change Background Suppression

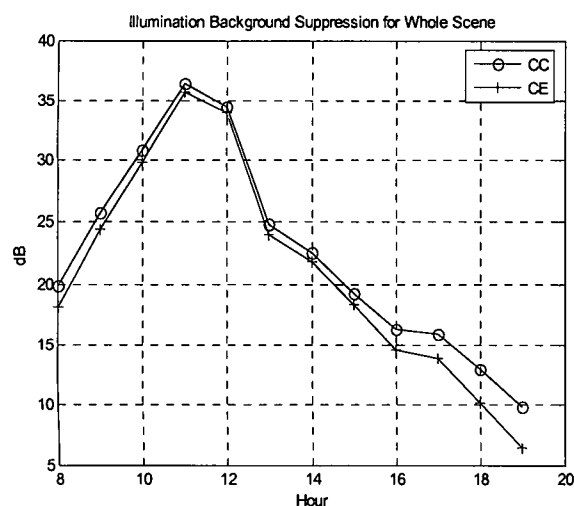
Illumination variability acts as the first test case. For a complete collection of color images for the data sets used in this case, please refer to Appendix A. For all of the cases, a differentiation is made between the region of the scene used to create the linear transforms  $\mathbf{T}$  and  $\mathbf{d}$  for both CC and CE and the region of the scene observed for suppression analysis. For the following cases, the entire scene is used to compute the gain matrix  $\mathbf{T}$  and the offset vector  $\mathbf{d}$  for both CC and CE methods. Using the suppression definition of (4.1.1), the variance of the error statistic is compared to the variance of the actual time-2 data. Figure 4.2.1 displays the total variance of specific regions of the time-2 data as described in Figure 4.1.1.



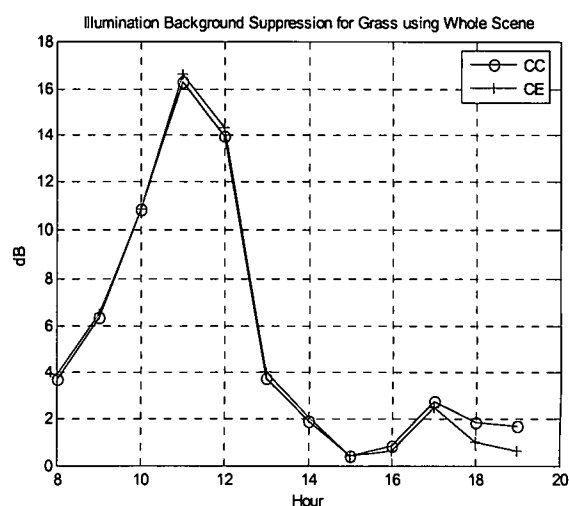
**Figure 4.2.1: Illumination Change Region Total Variance**

For all the regions, total variance is well above the noise floor of around 0.56 determined earlier from the PC discussion. The panel variance peaks when they reflect the most direct solar radiation. Due to their orientation, the panels are brightest around 10:00. The grass region variance remains relatively constant throughout the day. Due to the orientation of the trees, their variance increases as the sun approaches the west and illuminates the entire region more uniformly. For change detection, a good prediction produces an error statistic with very low variance for background resulting in higher suppression. One anticipates good background suppression for the varying illumination cases since the entire scene undergoes a general illumination shift. A single linear transform should adequately simulate this change. Figure 4.2.2 demonstrates the

background suppression for the whole scene for the data sets collected from 8:00 to 19:00.



**Figure 4.2.2: Illumination Background Suppression for Whole Scene Using Whole Scene to Compute Transform**

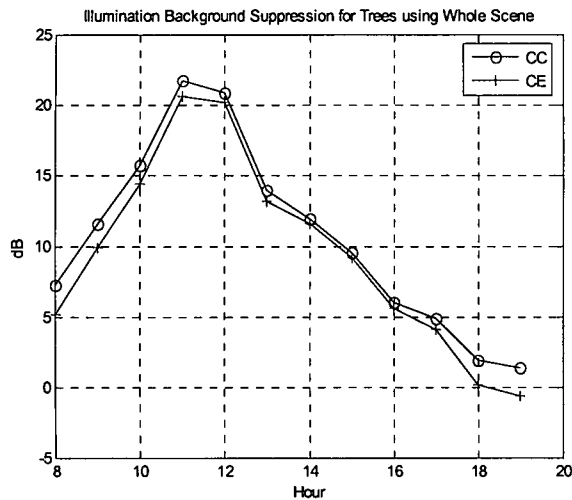


**Figure 4.2.3: Illumination Background Suppression for Grass Using Whole Scene to Compute Transform**

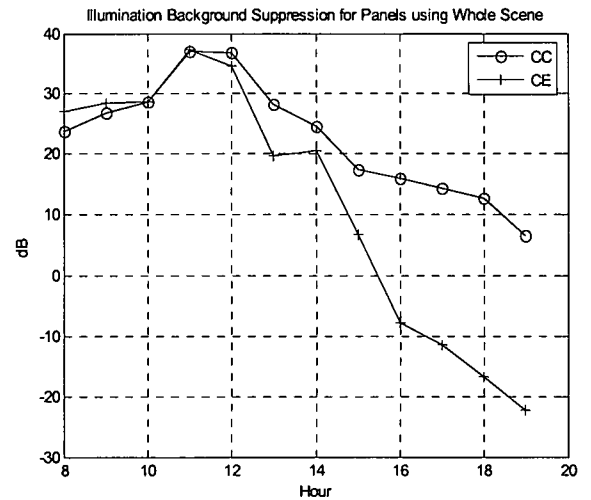
As expected, the CC and CE prediction methods perform relatively well at background suppression. Suppression is greatest when the solar position is most similar to that of the May 5 SA 120° reference image. The solar illumination at 11:00 is most similar to the reference image resulting in the best background suppression. Suppression continues to decrease as the solar position varies from hour to hour due to non-uniform illumination conditions. More prominent shadows manifest themselves early and late in the day adding to the complexity of linear prediction and reduced clutter suppression.

Using the same linear transforms developed from the entire scene statistics, the variance of specific regions of the scene are observed to determine how well suppression is performed. Figure 4.2.3 describes the suppression of the grass region. The CC and CE methods still provide some suppression for grass but not as great as the entire scene suppression levels. Since the total variance of the grass region is fairly low to begin with, suppression values for this region are not very great. In addition, grass in general is fairly

difficult to suppress from day to day due to spatial variability from growth, cutting patterns from lawnmowers, or even variations within the soil beneath. Viewing Figure A.1.1, one observes distinct mowing patterns within the reference image. Similar patterns exist in the early illumination change images for Appendix A.2. However, these patterns vary in the later images and, coupled with other scene changes, result in poor suppression. Shadows and illumination conditions of the grass region play an important role as well. Figures 4.2.4 and 4.2.5 describe the suppression of the tree and panel regions respectively.



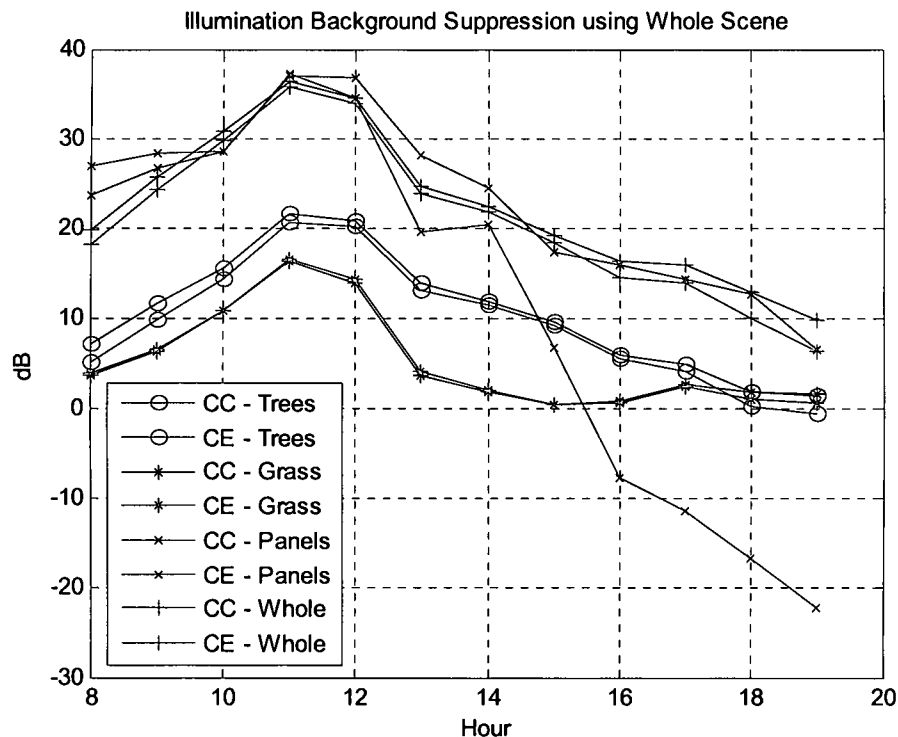
**Figure 4.2.4: Illumination Background Suppression for Trees Using Whole Scene to Compute Transform**



**Figure 4.2.5: Illumination Background Suppression for Panels using Whole Scene to Compute Transform**

A similar falloff before and after 11:00 is observed for both cases. In the time-1 tree region, uniform illumination does not exist. By observing image A.1.1 in Appendix A, one notices how certain areas of the tree region are shaded whereas others are illuminated. Consequently, a single transformation may not adequately describe the changes occurring to the entire tree region. The panels present a different problem due to their specular nature. Positioning of the sun coupled with the tilt and orientation of the

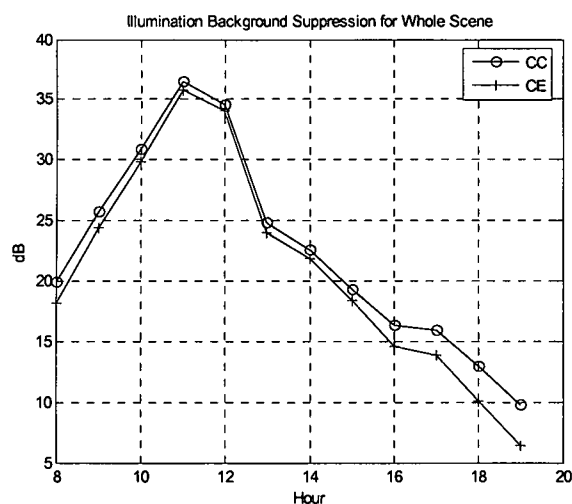
panels provides different reflection characteristics than the remainder of the scene at times. For example, the grass and trees in Figure A.2.12 are very well illuminated. However, the panels in the image are darker than the reference image of Figure A.1.1. Consequently, the linear transform applied to the whole image does not work as well for the panels resulting in less suppression or no suppression at all for the CE case. In addition, the panel region variance decreases later in the day causing suppression values to decline as well. Figure 4.2.6 summarizes the results discussed for this case.



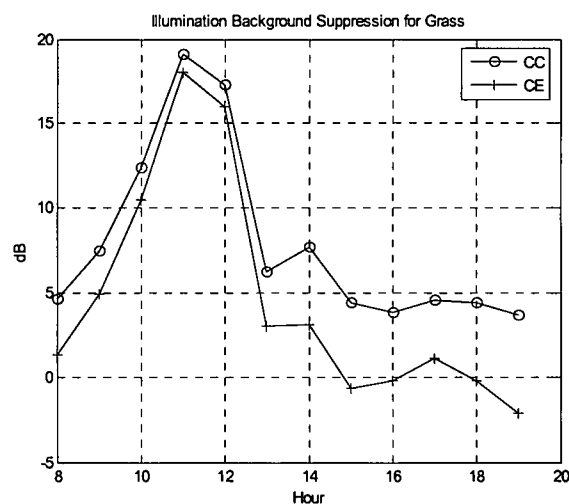
**Figure 4.2.6: Illumination Background Suppression Summary using Whole Scene to Compute Transform**

Instead of using the entire scene to calculate a single gain matrix  $T$  and offset vector  $d$ , the image is manually segmented into tree, grass, and panel regions. Using the statistics of each respective region, a separate linear transform is developed and applied for prediction of that region. The suppression of each of these regions is determined.

Figure 4.2.7 again displays the suppression of the entire scene using one linear transform for the whole scene. Figure 4.2.8 displays the suppression of the grass region using the statistics of this region to develop the linear transform. Again, the grass region remains difficult to suppress even further. However, this method produces better results than simply using the single transform developed using the entire scene.



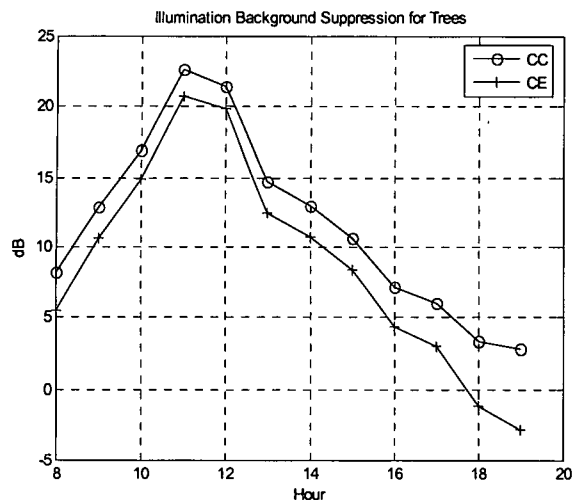
**Figure 4.2.7: Illumination Background Suppression for Whole Scene Using Whole Scene to Compute Transform**



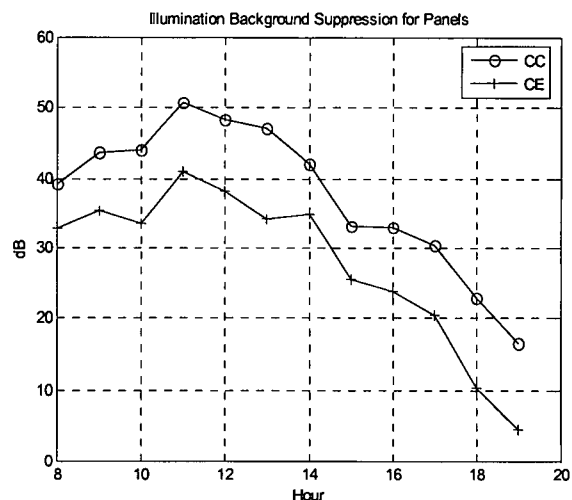
**Figure 4.2.8: Illumination Background Suppression for Grass Using Grass to Compute Transform**

Figures 4.2.9 and 4.2.10 describe the suppression of the tree and panel regions using transforms developed for each of their respective regions. Again, the separate transforms result in better suppression than the single transform case.



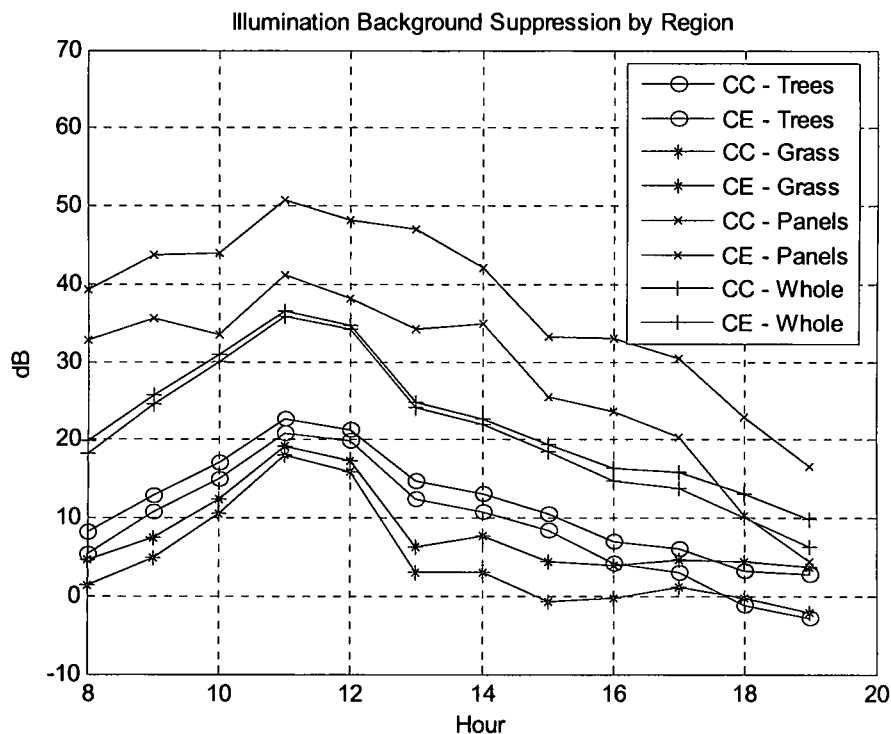


**Figure 4.2.9: Illumination Background Suppression for Trees using Trees to Compute Transform**



**Figure 4.2.10: Illumination Background Suppression for Panels using Panels to Compute Transform**

Figure 4.2.11 summarizes the results just discussed. The same falloff in suppression early and late in the day exists.

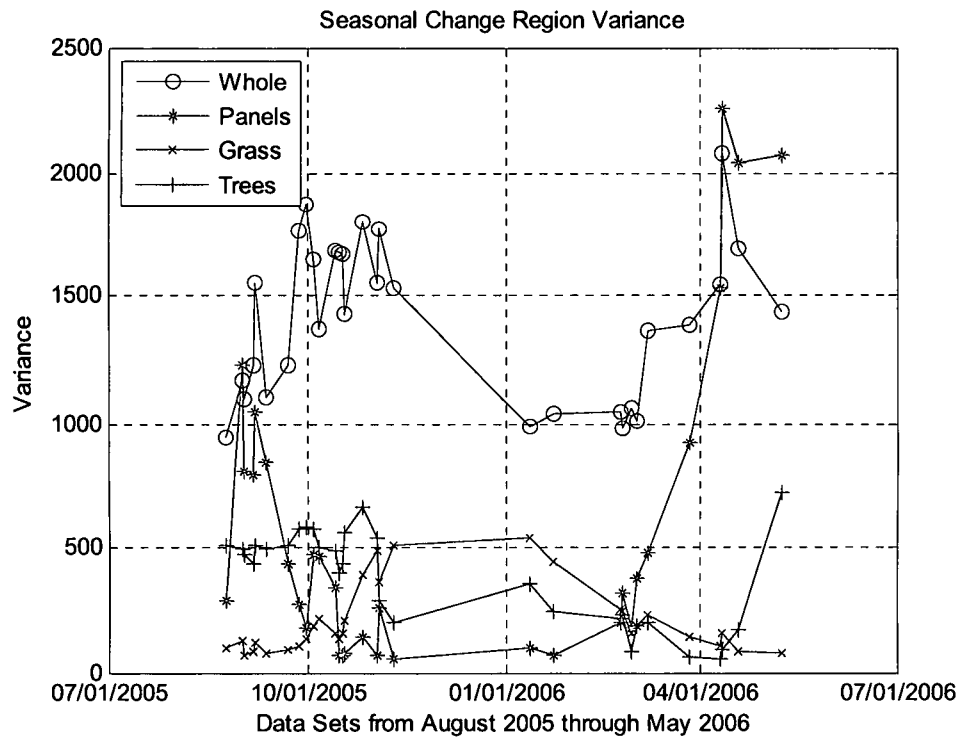


**Figure 4.2.11: Illumination Background Suppression by Region Summary**

For the most part, illumination changes are handled fairly well by the change detection algorithms. In some cases, a single transformation developed using the entire scene does not work well on all the regions resulting in poor suppression. Likewise, a separate transform developed for each region typically produces better results. For the tree and grass regions, the segmented suppression results are only marginally better than the single transform case. One can expect this result since the majority of the scene is comprised of grass and trees. Due to their similar spectral nature in the illumination change cases, the transforms for the two regions are fairly similar to the entire scene transform.

#### **4.3 Seasonal Change Background Suppression**

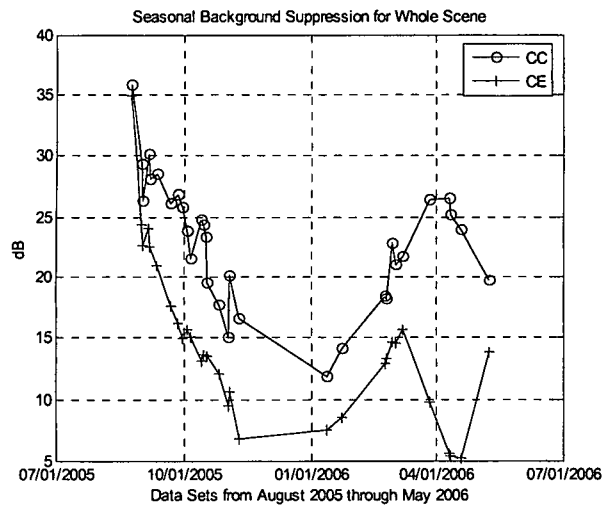
After observing suppression for illumination changes within a scene, the next case possesses more complexity by adding vegetation variability to drastic seasonal illumination change. Over the course of several seasons, the solar elevation angle varies greatly. Likewise, tree leaves senesce and grass conditions alter significantly. The same experimental approach as the illumination change case is used. For a complete list of color images for the scenes used for seasonal background suppression, see Appendix A.3. Figure 4.3.1 displays the total variance of specific regions for the scenes used in this study.



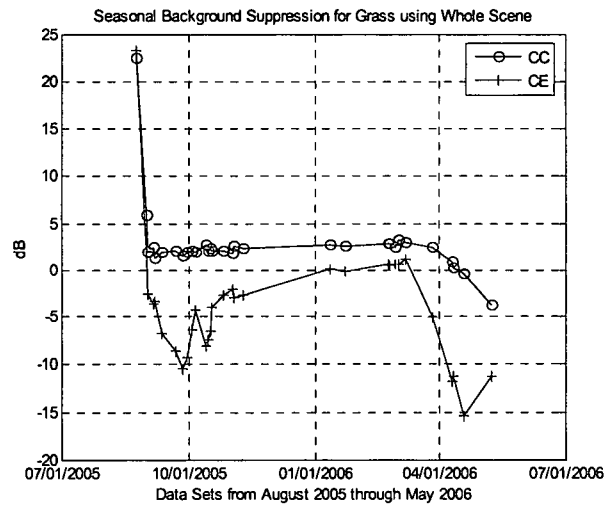
**Figure 4.3.1: Seasonal Change Region Total Variance**

Tree region variance declines as the trees become bare and illumination decreases. As the leaves return in April and SE rises, variance increases. Grass region variance increases as the grass dies and shadows appear more prominently in the area. Panel region variance decreases and later increases according to illumination conditions.

In the first case, a single transform is developed and applied to the whole scene. Figure A.1.2 displays the August 25, 2005 SA 180° time-1 scene used for seasonal study. Again, the background suppression of the entire scene and specific regions is determined. Figures 4.3.2 through 4.3.5 display the results of this case.

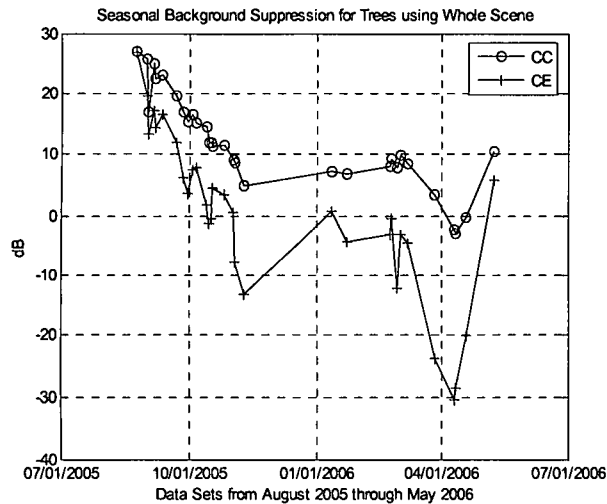


**Figure 4.3.2: : Seasonal Background Suppression for Whole Scene Using Whole Scene to Compute Transform**

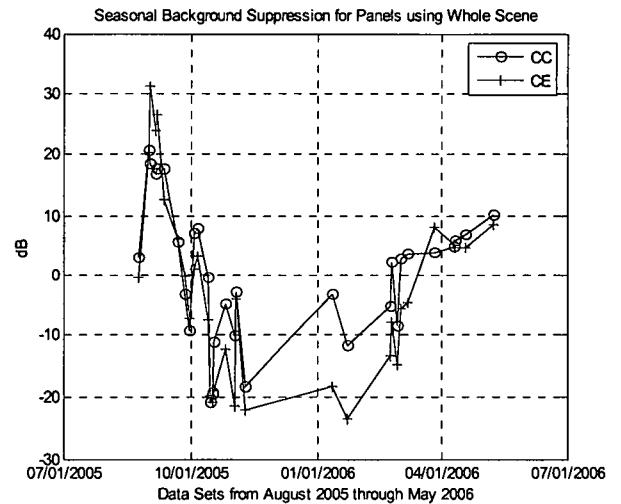


**Figure 4.3.3: Seasonal Background Suppression for Grass Using Whole Scene to Compute Transform**

As expected, the transform works well for total scene suppression early on and generally depreciates as the seasons change reaching a minimum in late December and early January. After this point, suppression generally improves again. The suppression results for the whole scene are attributed to the seasonal illumination conditions to a certain extent. Referring back to the solar trajectory plots of Figures 2.7.3 and 2.7.4, a partial explanation for the suppression results manifests itself. In December, solar elevation and daylight time are at a minimum. All of the data sets used for the seasonal study are collected near maximum SE for that particular day. The maximum SE for the time-1 scene is near  $60^\circ$  whereas that in December is below  $30^\circ$ . Lower solar angles result in less scene radiance and lower scene variance in some cases. The low elevation results in drastic shadows present within the scene as well. Shadows are difficult to predict and suppress. As the solar elevation increases, the suppression improves. Again, the grass region is not suppressed very well due to low initial variance and spatial differences from scene to scene.

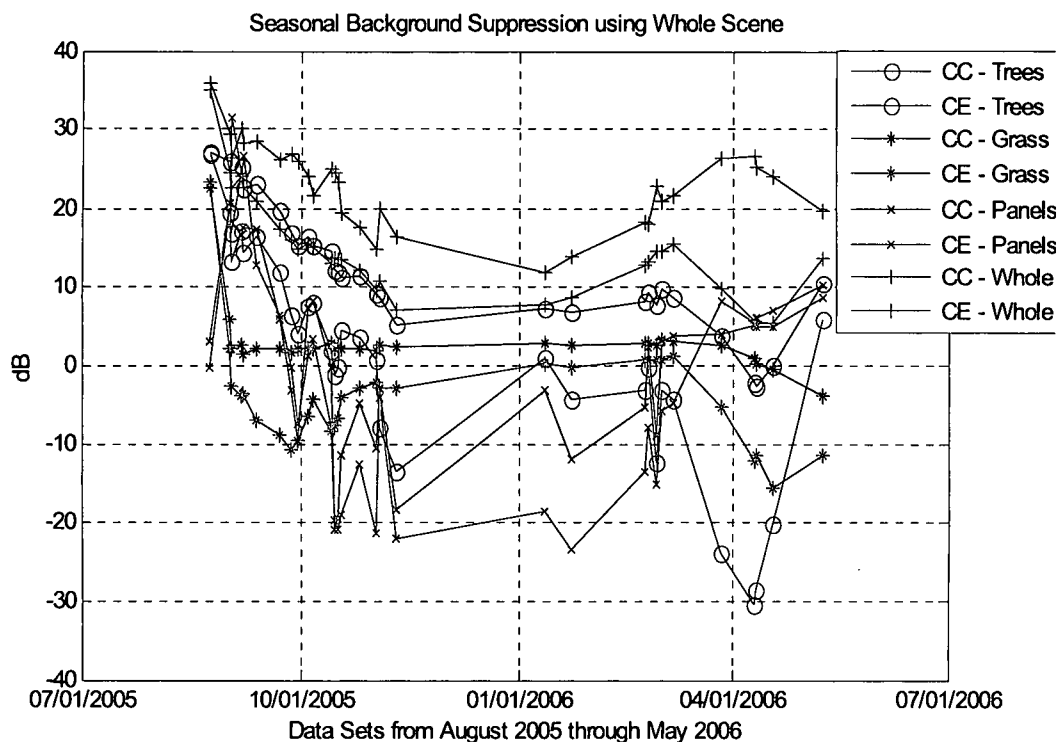


**Figure 4.3.4: Seasonal Background Suppression for Trees Using Whole Scene to Compute Transform**



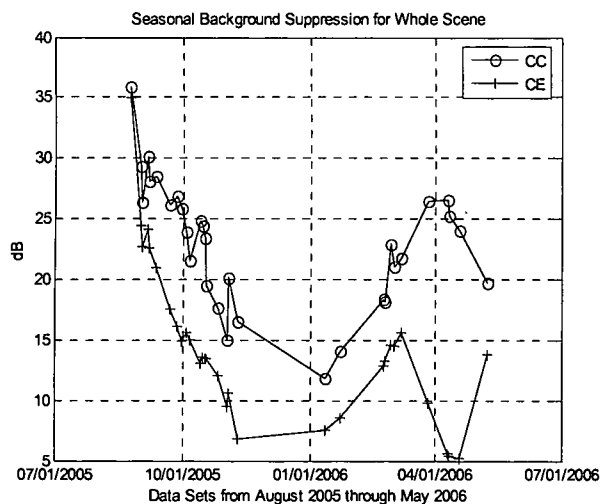
**Figure 4.3.5: Seasonal Background Suppression for Panels Using Whole Scene to Compute Transform**

Suppression for the tree region decreases as the leaves senesce and fall off. In addition, the illumination of the tree region is not uniform or constant through the seasons. In April and early May, suppression begins to improve as the leaves come back and illumination begins to resemble that in the time-1 image. Suppression of the panel region drops extremely past early September. This poor suppression is attributed to lower solar elevation angles. These lower angles result in poor illumination of the panels and later produce prominent shadows across the panels as well. The variance of the panel region decreases making further suppression difficult. In addition, the illumination change is difficult to characterize and predict using a single transform for the whole scene. The entire scene illumination does not change as drastically as the panel region. Figure 4.3.6 summarizes the results.

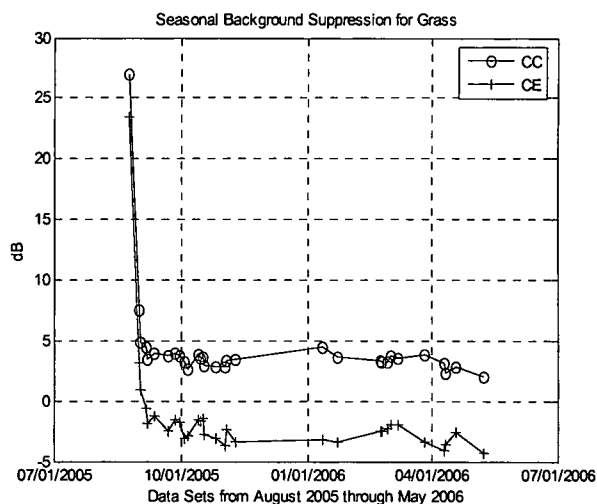


**Figure 4.3.6: Seasonal Background Suppression Summary using Whole Scene to Compute Transform**

Next, the scene is segmented and a separate transform is used for each region to assess suppression. Figure 4.3.7 reiterates the entire scene suppression results.

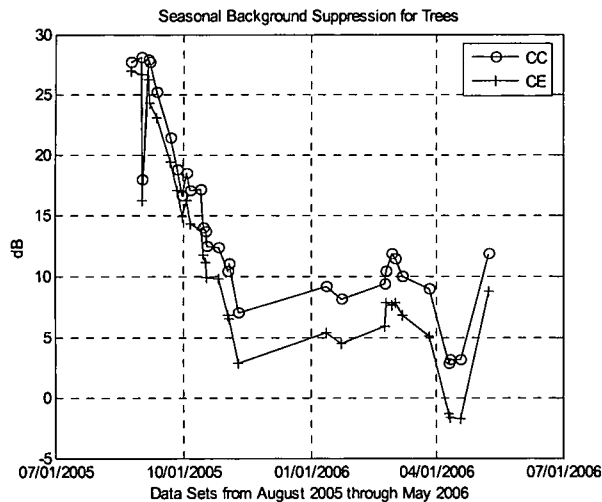


**Figure 4.3.7: Seasonal Background Suppression for Whole Scene Using Whole Scene to Compute Transform**

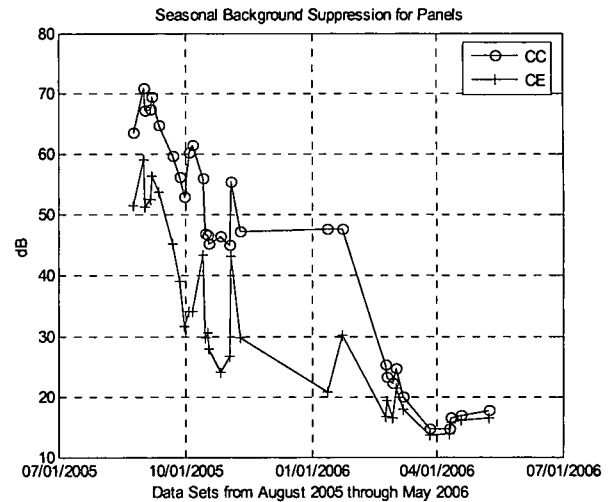


**Figure 4.3.8: Seasonal Background Suppression for Grass Using Grass to Compute Transform**

A separate transform used for the grass region does not produce much better results than the single transform. Covariance equalization prediction actually hurts suppression. Even when the variance of the grass region increases from November through January, suppression remains low. Grass is difficult to characterize and predict from day to day, let alone month to month. The differences in growth, cut, health, and illumination make suppression extremely difficult. Drastic shadows appear in the upper grass region during the winter months making the region even more difficult to characterize. Figure 4.3.9 displays the results for the tree region.



**Figure 4.3.9: Seasonal Background Suppression for Trees Using Trees to Compute Transform**



**Figure 4.3.10: Seasonal Background Suppression for Panels Using Panels to Compute Transform**

Suppression for this region is better than the single transform case. However, suppression still decreases as seasonal leaf changes occur. One transform for this region may still be insufficient due to varying illumination conditions, differing leaf types, and portions of the region that are actually stems and tree trunks. The leaves do not senesce at the same rate or in the same fashion across the scene. Consequently, the tree region needs to be segmented even further, creating a transform for each segmented class. With the trees completely bare and a low SE in January and February producing uniform tree

illumination, the suppression results begin to improve. However, as the SE gradually increases creating varying shadows within the tree region and as the leaves begin to grow back sporadically, the single transform does not work well. By late April and early May, the trees are full and suppression rises as region variance increases and the time-2 scene begins to resemble the reference. Figure 4.3.10 displays the regional suppression results for the panels. In this case, region suppression works dramatically better than the single whole scene transform result. A transform for this region accurately captures the varying illumination and shadow conditions. The drastic decrease in suppression from February 15 to the 23 results from an inadvertent change in panel tilt between the two collections. Consequently, the suppression results are lower than expected after February 15.

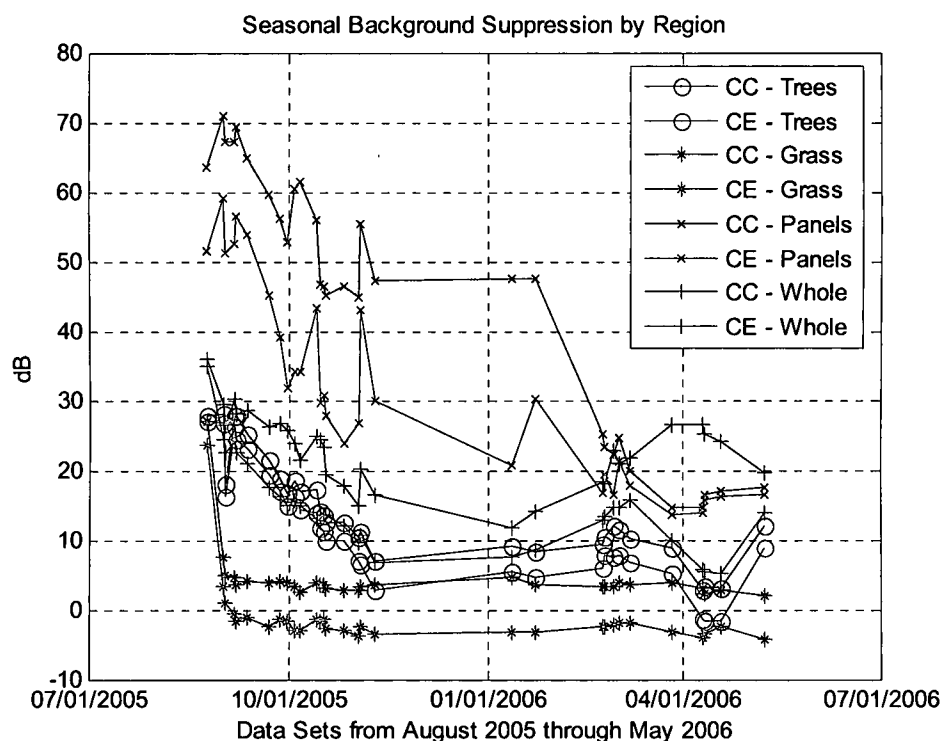


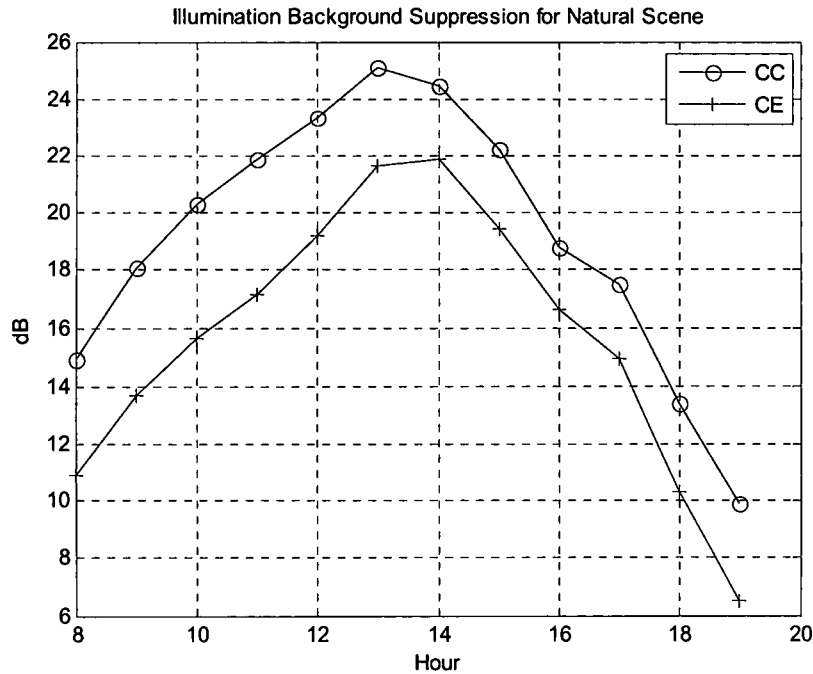
Figure 4.3.11: Seasonal Background Suppression by Region Summary



Figure 4.3.11 summarizes the regional suppression results. Generally speaking, regional suppression outperforms the single scene transform method. In some cases, further segmentation of the scene is necessary as portions of the regions change in different ways. A single transform cannot adequately describe illumination changes, shadows emerging, leaves senescing, and grass alterations.

#### **4.4 Illumination Change Signal to Clutter Ratio**

After determining the suppression results of the change detection prediction methods, the effects of suppression at improving detection performance must be analyzed. The change detection algorithms of CC and CE are compared in performance to the Mahalanobis distance algorithms using the signal to clutter ratio of (4.1.2). For the change detection algorithms, the time-1 data has the panels removed, as demonstrated in the scene of Figure A.1.3. Consequently, the appearance of panels in any subsequent scenes represents a target. The beige panel and green panel are labeled target in two separate cases to demonstrate how the results vary when using a more easily detectable panel versus one resembling background. Regardless of which panel is labeled target, the other panels are ignored and the remainder of the scene is considered background. Only the background region is used to determine statistics used for detection in an attempt to limit the impact of the panels on these statistics. Using the same data sets as the illumination change suppression analysis, the three algorithms are compared. Figure 4.4.1 displays the suppression results for the natural region using the reference scene without panels.



**Figure 4.4.1: Illumination Change Background Suppression for Natural Scene**

Suppression for the natural scene peaks around the 13<sup>th</sup> hour, suggesting this time produces illumination conditions resembling those of the reference scene. These suppression results may be useful when analyzing the SCR results for change detection.

In addition to testing M-distance and change detection, various forms of spectral matched filtering are compared using different methods to produce the beige and green panel signatures. In one case, the respective panel spectrum is extracted directly from the scene. This spectrum should produce an upper bound with which to judge other methods. In another case, atmospheric compensation is performed on the respective lab signature using a shade point and the silver panel to produce the reference spectrum. In the last two cases, CC and CE are used to evolve a respective spectrum extracted from the reference scene to the time-2 scene. These results are compared using the SCR metric. SCR for the change detection and M-distance cases should not be compared to that found for the SMF cases since the two detection statistic are produced via different methods.

For the change detection algorithms, only the natural regions, excluding the panels, are used to produce the linear transform values of  $\mathbf{T}$  and  $\mathbf{d}$ . This method limits any effect the panels may have on estimating the background statistics. In real world situations, targets should only comprise a fraction of the scene and would not affect the background statistics to a great degree. However, the panels in this scene do comprise a significant portion. To avoid any undesired effects, they are omitted from statistical calculation. Figure 4.4.2 displays the SCR results of the various detection algorithms for the beige panel.

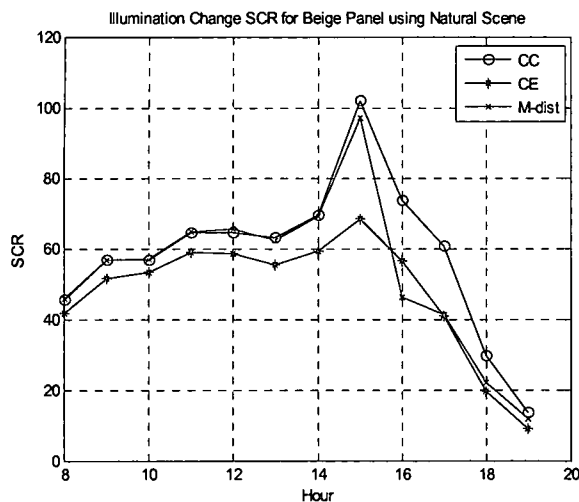


Figure 4.4.2: Illumination Change SCR for Beige Panel

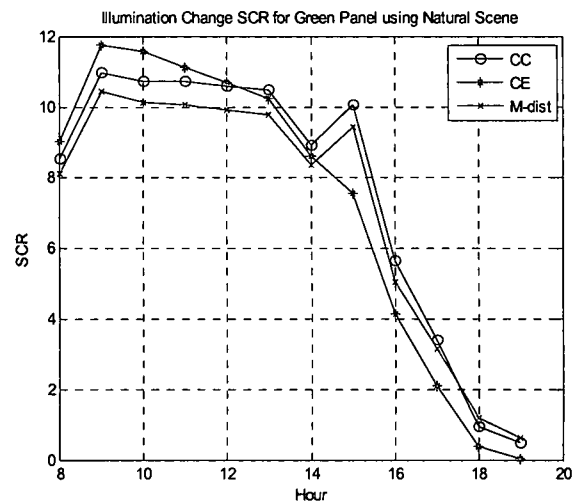
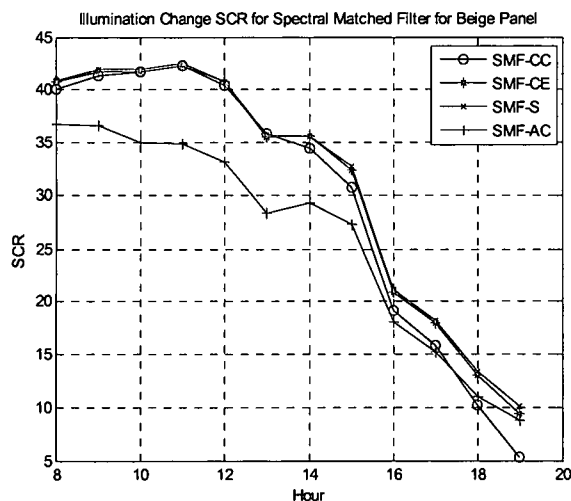


Figure 4.4.3: Illumination Change SCR for Green Panel

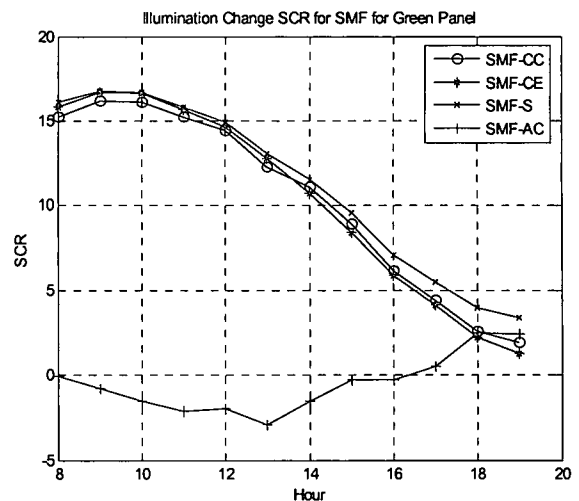
The change detection algorithms do not provide any advantage for the beige panel detection until about the 15<sup>th</sup> hour. This fact may result from how spectrally different the beige panel is than the background. The separation between background and target in this case is very large and change detection does little to improve. However, as the day progresses and the beige panel receives less illumination, CC begins to improve SCR over the simple M-distance. The target and background separation is not as great leaving some room for improvement for change detection. Green panel results of Figure 4.4.3

reiterate the late afternoon and early evening results found for the beige panel. The green panel spectrally resembles grass and trees making it more difficult to detect. Consequently, the target and background separation is not as great. Suppressing the variance of the background for this case helps increase the separation between background and target. The SCR results seem to depend more upon the target itself rather than the suppression levels produced by the change detection algorithms. Once the target becomes extremely difficult to detect later in the day, the performance of all the detectors decreases and change detection does not provide any improvement.

Next, the performance of SMF for illumination change using the various reference spectra is determined. Figure 4.4.4 displays the SCR results for SMF on the beige panel using an in-scene spectrum (SMF - S), an AC lab spectrum (SMF - AC), a spectrum evolved from the time-1 scene using CC (SMF - CC), and a spectrum evolved from time-1 using CE (SMF - CE).



**Figure 4.4.4: Illumination Change SCR for SMF on Beige Panel using Different Methods**



**Figure 4.4.5: Illumination Change SCR for SMF on Green Panel using Different Methods**

The results reinforce the background suppression results for the illumination change case. While the in-scene spectrum produces the best results, the evolved signatures from CC and CE result in nearly the same SCR. This result suggests the predictors are accurately modeling the illumination change. The AC signature leaves room for improvement as the SCR is not as large in this case. For all the signatures used, the SCR generally drops later in the day as the panel becomes more difficult to detect. The results for the green panel case in Figure 4.4.5 reiterate those of the beige panel. The evolved signatures work nearly as well as the in-scene spectral. For this panel, atmospheric compensation does not produce an accurate green spectrum resulting in poor matched filtering results.

#### 4.5 Seasonal Change Signal to Clutter Ratio

Using the same time-1 scene without panels as the previous case, seasonal changes and SCR are explored. Figure 4.5.1 displays the seasonal background suppression for the natural scene.

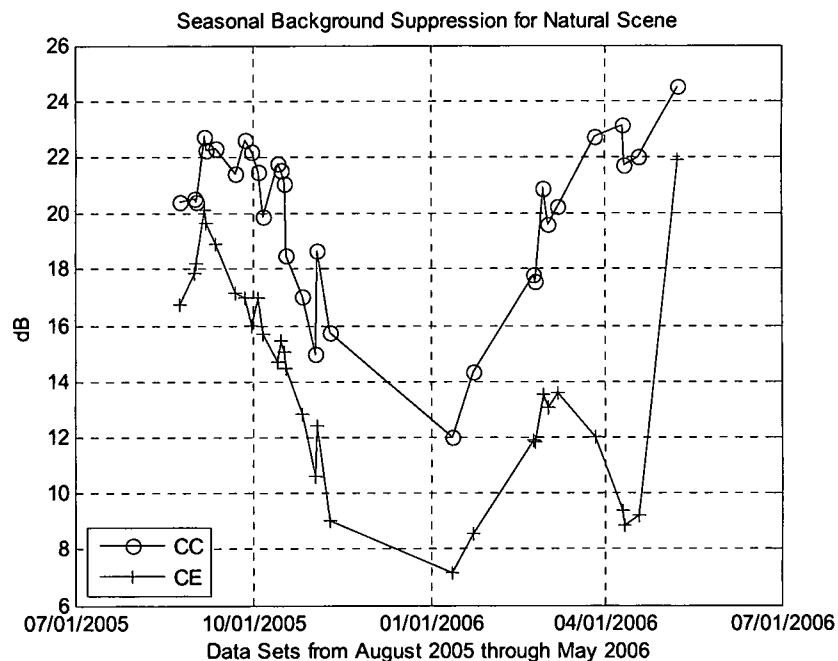


Figure 4.5.1: Seasonal Change Background Suppression for Natural Scene

Figures 4.5.2 and 4.5.3 display the SCR results for the beige and green panel respectively.

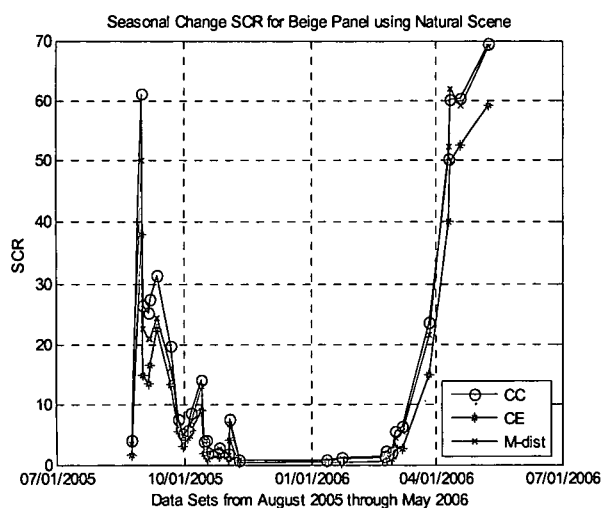


Figure 4.5.2: Seasonal Change SCR for Beige Panel

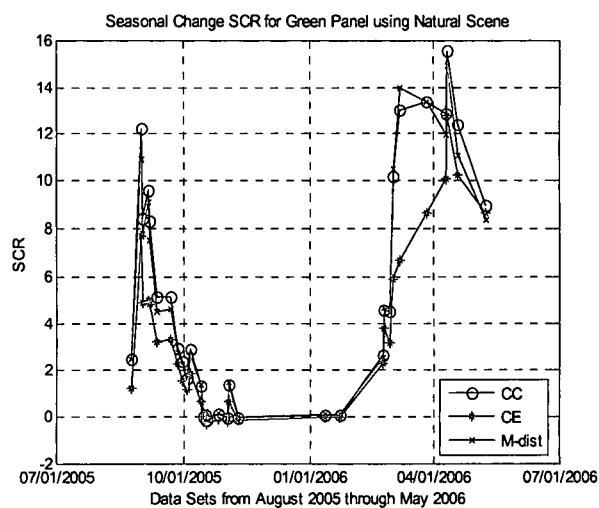
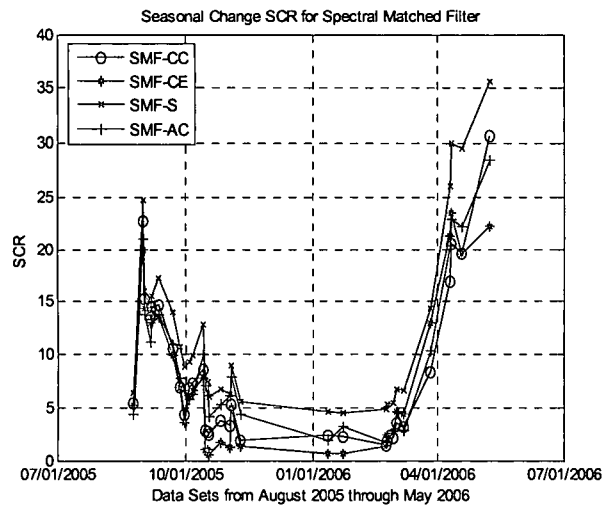


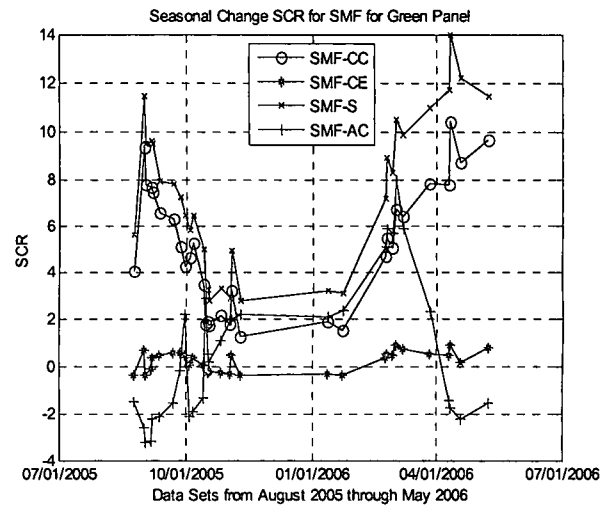
Figure 4.5.3: Seasonal Change SCR for Green Panel

In a general sense, a balance exists between target detectability and the improvement in performance provided by change detection. Early on, CC increases SCR for both the beige and green panel cases. As the panels become extremely difficult to detect in the winter months, performance for all the detectors deteriorates and change detection does not offer improvement. In late spring, the beige panel is well illuminated extremely easy to detect for M-distance. Consequently, change detection does not improve performance. For the green panel case, CC still improves performance in the spring period since the panel is still difficult to detect. From these results, conclusions about the impact of background suppression levels on detector performance are difficult to determine. For the most part, CE does not offer an advantage over a simple M-distance measure in the seasonal change cases.

Using the same methods as the illumination change case, SCR for spectral matched filtering is observed for both the beige and green panel. Figure 4.5.4 displays the SCR results using beige panel signatures acquired in different manners.



**Figure 4.5.4: Seasonal Change SCR for SMF on Beige Panel using Different Methods**



**Figure 4.5.5: Seasonal Change SCR for SMF on Green Panel using Different Methods**

Again, the in-scene beige panel signature provides an upper bound for performance. The prediction results for CC and CE deteriorate as the seasons change resulting in poor performance. In fact, using AC provides a better signature for SMF than CC or CE in many of the beige panel cases. As the seasonal changes become more drastic, CC and CE predictors no longer offer a viable signature. When the scenes begin to resemble that of the reference, the predictor results improve. For the green panel case, the results in Figure 4.5.5 resemble those of the beige panel. However, the CE evolved signature does not work well for the green panel. Due to the difficulty in detecting the green panel, any errors introduced from evolving the signature using CE are amplified resulting in poor detection. The AC signature does not perform well at the beginning and end of the study time but improves slightly in the late fall through late winter months. This may result

from less illumination on the panel reducing the degree of error produced in the AC green spectrum. General SCR results deteriorate during the winter months due to poor illumination conditions.



## **CHAPTER 5**

### **Conclusion**

A brief summary of this research and the results produced is given here. A general overview of the system calibration and data collection process as well as testing theory is discussed. Final conclusions are given with possible future study to perform as well.

#### **5.1 Summary**

This research assesses the impact illumination and vegetation changes upon a scene have on hyperspectral target and change detection. Researching this topic requires observations of the same scene over the course of several seasons using a hyperspectral imager. The entire hyperspectral data collection process is very involved requiring numerous steps. Beginning with the assembly of the hyperspectral imager, detailed specifications of all the equipment used is necessary. After constructing the imager, a spectral calibration is performed using lamps and a laser to map center wavelength locations on the focal plane array of the camera. The spectral calibration describes any distortion or sensor misalignment as well. Using an integrating sphere, a noise characterization is performed on the system to determine a suitable wavelength and spatial range for the focal plane. Noise characterization helps determine the degree of filtering necessary for SNR and the resulting spectral resolution of the system. After

each data collection, an absolute radiometric calibration is applied to eliminate pixel response non-uniformity and to trace digital numbers to radiometric values.

With the spectrometer system set up in a tower, scene data is collected from late August 2005 through May 2006. Both man-made and natural objects reside within the scene allowing for various algorithm tests. The repeatability and accuracy of the pan and tilt produces near perfect image registration from image to image. This achievement permits change detection analysis. Based on probability hypothesis testing, change detection attempts to suppress background and detect anomalous changes using a Mahalanobis distance measure on the error data. Essentially, a linear prediction of time-2 data using statistics from time-1 and time-2 data is compared to actual time-2 data to develop the error statistic. Ideally, a good predictor suppresses objects not changing within the scene. Therefore, background suppression is used as the metric for gauging how well the algorithms are performing. In order to determine the effectiveness of the background suppression using chronochrome and covariance equalization at assisting in target detection, a signal to clutter ratio is used for comparison between change detection and the single instance detector of Mahalanobis distance. A comparison of SMF using spectra obtained in various ways is performed as well. The signal to clutter ratio compares the mean of the target pixels in the detection statistic image versus the standard deviation of the background pixels in the detection statistic.

For testing of illumination changes, the data sets collected from 8:00 to 19:00 between May 8, 2006 and May 22, 2006 are used. These sets provide a vast range of solar positions for comparison. A comparison is made between using the whole scene to compute a single transform for change detection and segmenting the image into tree,

grass, and panel regions and producing a transform for each separate region. The single transform captures the illumination changes fairly well and helps in total and regional background suppression. Generally, when the illumination conditions at time-2 resemble those of the time-1 image, suppression is greatest. As solar position varies more from the reference image, suppression decreases. Using a separate transform for each region typically increases suppression, especially for the panel region. Variance and suppression for the panel region depend strongly upon the illumination conditions. Suppression levels for grass are typically low because of low region variance to begin with. In addition, mowing patterns, growth of the grass, and shadow regions can vary from day to day making prediction and suppression difficult.

For testing seasonal changes, the data sets collected at SA 180° from late August 2005 through May 2006 are used. Seasonal changes include both illumination variability resulting from maximum solar elevation changes and vegetation variability resulting from leaves senescing and returning and grass states changing. The same method for testing illumination changes is applied here. A single transformation determined using the whole scene does not produce significant suppression throughout the seasons as the illumination change case. Shadows appear in the scene, leaves senesce and fall off, grass conditions change, and general scene illumination is not uniform. In cases from fall through early spring, changes occur to the tree region in a different fashion than both the grass and panel regions. The single transform does not suppress each region very well. Suppression of the panel region is very poor using this method due to shadow variations upon them as well as low region variance. Using a separate transform for each region helps in seasonal suppression as well. The greatest suppression increase is seen for the

panel region. Room for improvement still exists for the grass and tree regions. Suppression in the tree region is least effective when separate changes are occurring. During the fall, leaves senesce and drop off at different rates. In addition, some areas of the trees lie in shadow while others are well illuminated. What is labeled as the tree region includes not only leaves, but stem and tree trunk as well. Consequently, further segmentation of the region may improve suppression results. Since variance for the grass region is low to begin with, improvements in suppression may be difficult. Changes in growth, health, shadows, and mowing patterns from month to month add to suppression difficulty. Further segmentation of the grass region may help as well.

A signal to clutter ratio is used to compare the performance of the change detection algorithms of CC and CE to the single instance method of Mahalanobis distance. The reference image for change detection has the panels removed. Therefore, their appearance in subsequent images allows them to act as targets. For this study, the beige and green panels are focused on as targets in separate cases. In addition to testing change detection and M-distance, the SCR for SMF is observed. For this case, reference spectra for the beige and green panel are obtained directly from the scene, from evolving a signature from an earlier scene using CC and CE, and from performing AC on the respective lab signature. Results for the beige and green panel are looked at separately to analyze differences between an easily detectable target and one which resembles background.

Using the same illumination change case files, the M-distance and change detection algorithms are compared. From the results, the impact of background suppression provided by the change detection algorithms is difficult to determine. When

the target is easily detectable, such as the beige panel in well illuminated conditions, change detection does not seem to provide any advantage. However, if the target and background separation decreases, as in the case of the green panel or the beige panel when not well illuminated, change detection does improve SCR results. If the detectability of the target remains very low, the performance of all the algorithms decreases and change detection does not provide improvement.

For the SMF comparison, the results reinforce those from background suppression. In the illumination change cases, the evolved signatures using CC and CE perform nearly as well as the signatures extracted directly from the scene suggesting the predictors accurately model the change. For the seasonal change cases, as the changes become more drastic, the predictors do not perform as well resulting in a much lower SCR for SMF than obtained using an in-scene spectrum. In general, detection of the panels becomes more difficult from late fall through early spring due to the presence of shadows and poor panel illumination. In some cases, atmospheric compensation does not produce accurate signatures resulting in poor detection performance. This poor compensation may result from non-uniform scene illumination conditions or inaccurate lab spectra.

Overall, the detection algorithms handle slight illumination changes very well. As changes become more drastic, performance typically decreases. The change detection algorithms can provide advantages over single instance detectors depending upon the detectability of the target. For these cases, segmenting the image in some fashion for suppression may increase performance even further. When multiple seasonal changes occur within a scene, background suppression and target detection become more difficult.

In this case, further segmentation of the image may provide some advantages for change detection.

## **5.2 Future Considerations**

Background suppression testing is performed in this research using manual segmentation of the image into grass, tree, and panel regions. For future study, using an automated segmentation method to determine a number of classes may provide a way to improve background suppression even further. For example, within the tree region a number of separate classes may exist. A differentiation can be made between well illuminated leaves and those in the shade. Stems and the base of trees can also form a separate class. During the fall and early spring, automated segmentation may provide benefits for separating regular leaves, dying leaves, and leaves beginning to grow back in. In the grass region, rocks, soil, shaded grass, healthy grass, and dying grass could all form separate classes. After spatially segmenting the scene using an automated method, a separate transform can be applied to each region for change detection. This segmented method of background suppression could significantly help during fall and spring time months when a number of changes are occurring simultaneously. One can also determine which scene to use for segmentation purposes. For this study, the time-1 images may not possess many different classes. However, using a scene from fall or early spring may produce a number of different classes with differences in vegetation health and shadow presence in the scene. A number of different segmentation techniques can be considered to improve background suppression and change detection results. In addition, further testing can be performed upon different types of targets to assess when change detection

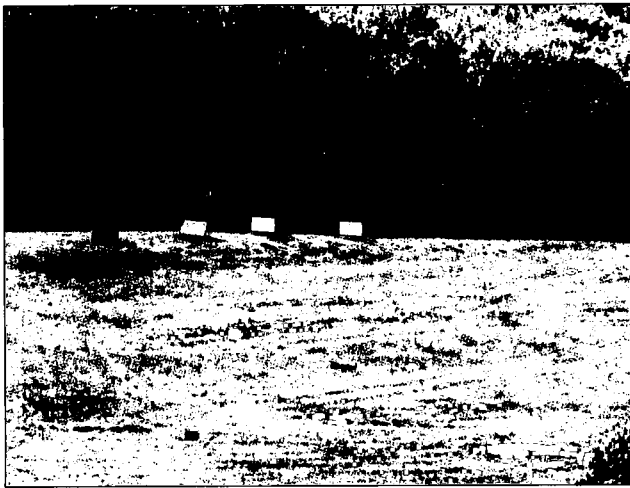
does provide advantages over simple single-instance methods. Considerations should be given to improving atmospheric compensation techniques as well.

## **APPENDIX A**

### **Scene Color Images**

This section of the appendix provides color images of all the scenes used for Background Suppression and SCR analysis. The section is subdivided into the reference images (time-1), the illumination change images (time-2), and the seasonal change images (time-2).

#### **A.1 Reference Images**

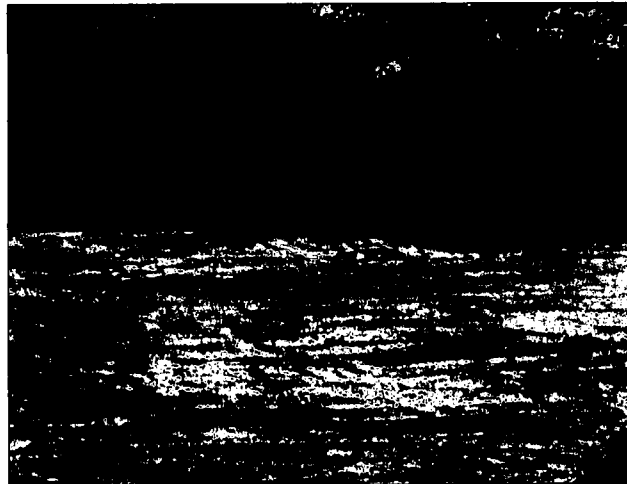


**Figure A.1.1: May 5, 2006 SA 120°  
Illumination Background Suppression Reference**



**Figure A.1.2: August 25, 2005 SA 180°  
Seasonal Background Suppression Reference**



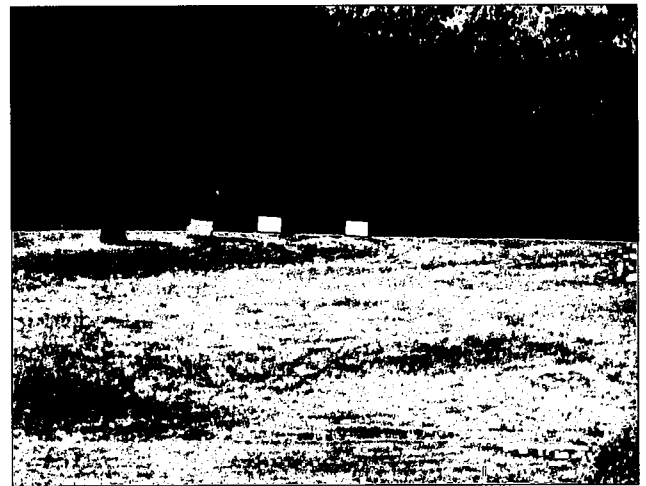


**Figure A.1.3: May 23, 2006 SA 180°  
SCR Reference**

## **A.2 Illumination Change Images**



**Figure A.2.1: May 8, 2006 8:00**



**Figure A.2.2: May 8, 2006 9:00**



Figure A.2.3: May 8, 2006 10:00

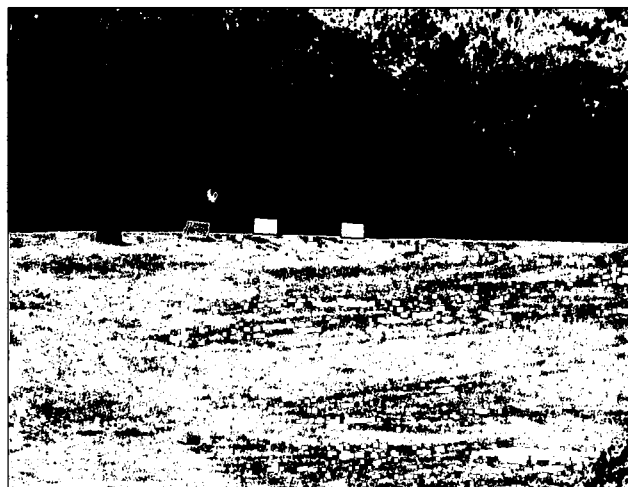


Figure A.2.4: May 8, 2006 11:00

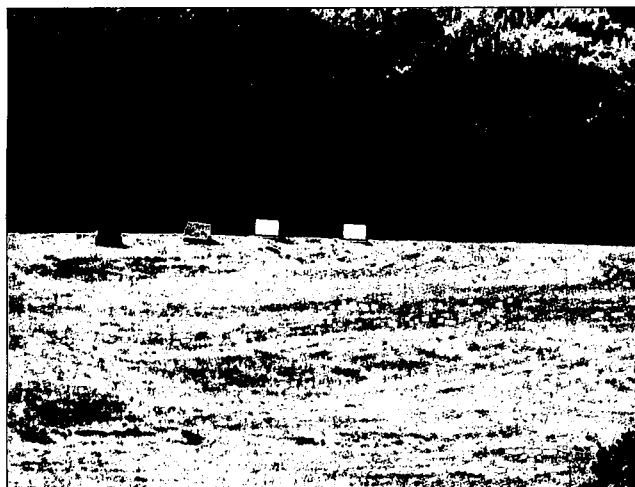


Figure A.2.5: May 8, 2006 12:00



Figure A.2.6: May 9, 2006 13:00



Figure A.2.7: May 9, 2006 14:00



Figure A.2.8: May 9, 2006 15:00



Figure A.2.9: May 20, 2006 16:00



Figure A.2.10: May 20, 2006 17:00



Figure A.2.11: May 22, 2006 18:00



Figure A.2.12: May 22, 2006 19:00

### A.3 Seasonal Change Scenes



Figure A.3.1: August 24, 2005 SA 180°



Figure A.3.2: September 1, 2005 SA 180°



Figure A.3.3: September 2, 2005 SA 180°

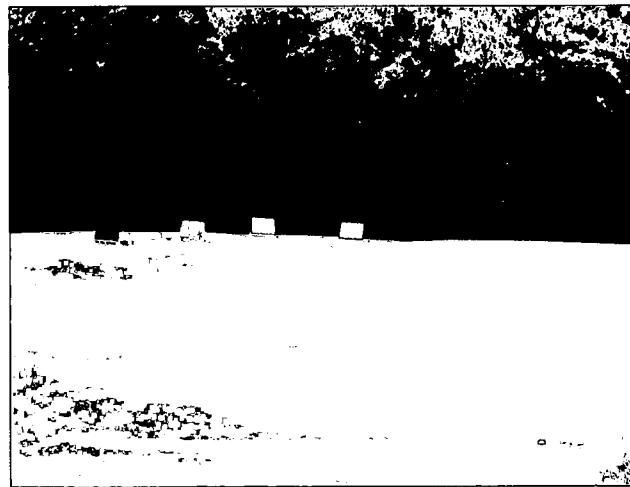


Figure A.3.4: September 6, 2005 SA 180°



Figure A.3.5: September 7, 2005 SA 180°



Figure A.3.6: September 12, 2005 SA 180°



Figure A.3.7: September 22, 2005 SA 180°

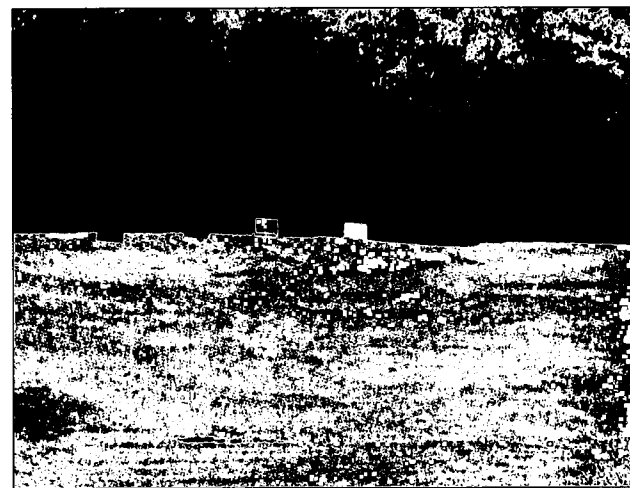


Figure A.3.8: September 27, 2005 SA 180°

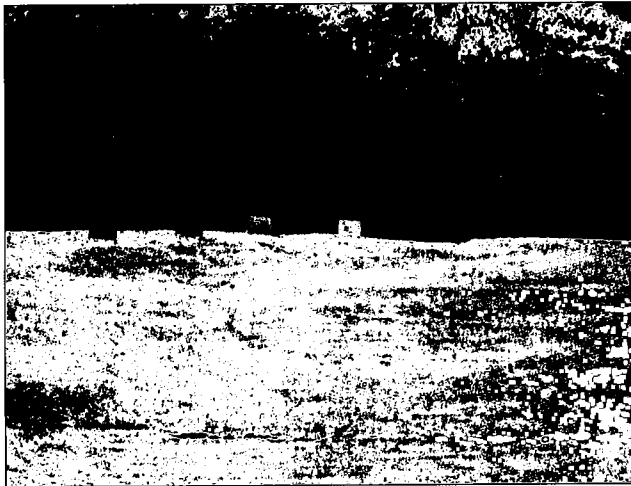


Figure A.3.9: September 30, 2005 SA 180°



Figure A.3.10: October 4, 2005 SA 180°

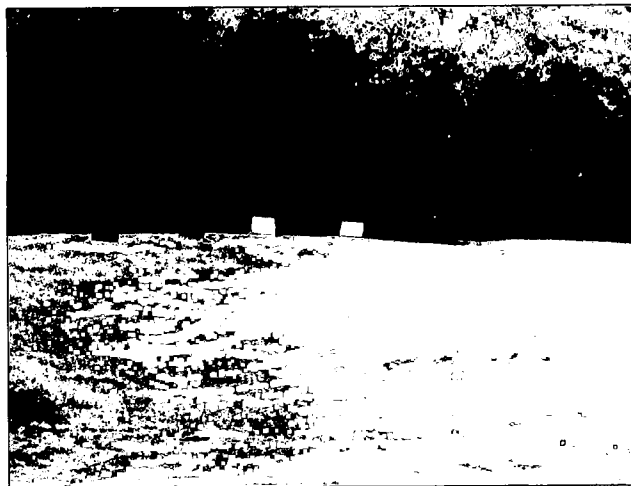


Figure A.3.11: October 6, 2005 SA 180°

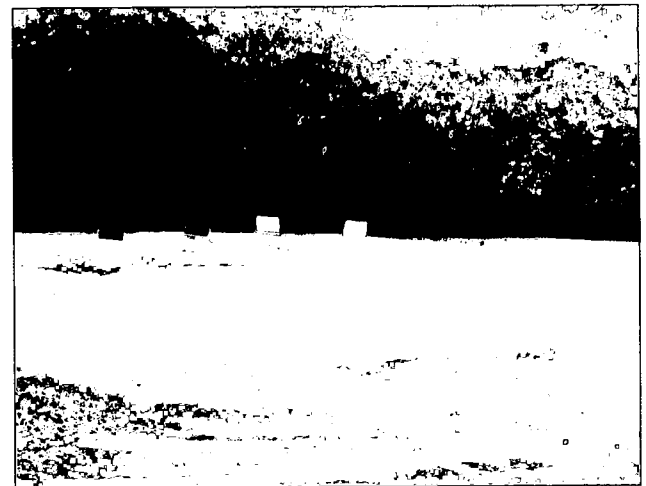


Figure A.3.12: October 14, 2005 SA 180°



Figure A.3.13: October 15, 2005 SA 180°



Figure A.3.14: October 17, 2005 SA 180°



Figure A.3.15: October 18, 2005 SA 180°



Figure A.3.16: October 26, 2005 SA 180°



Figure A.3.17: November 2, 2005 SA 180°



Figure A.3.18: November 3, 2005 SA 180°

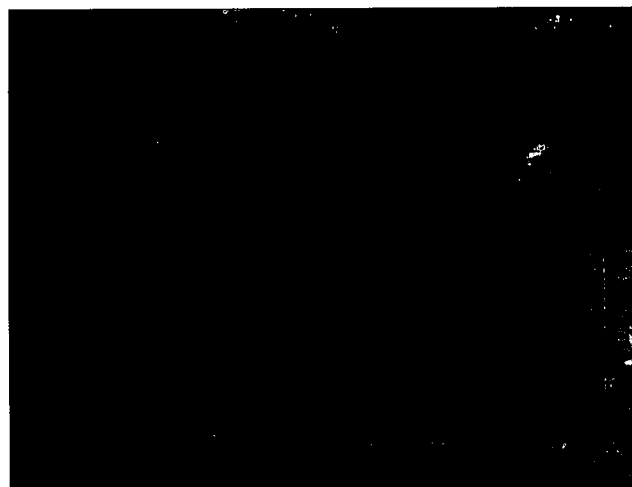


Figure A.3.19: November 10, 2005 SA 180°



Figure A.3.20: January 12, 2006 SA 180°

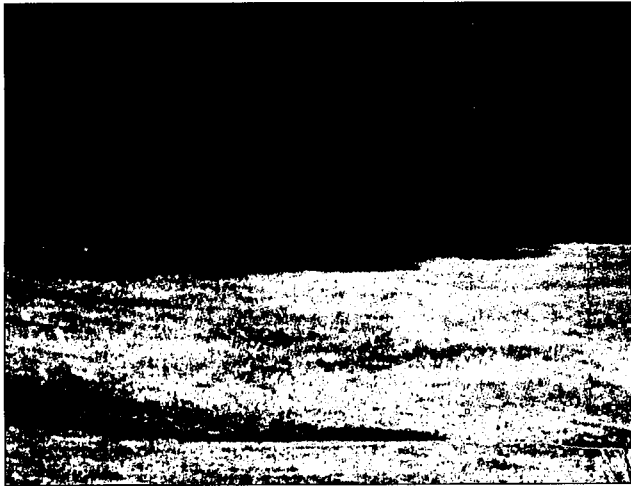


Figure A.3.21: January 23, 2006 SA 180°

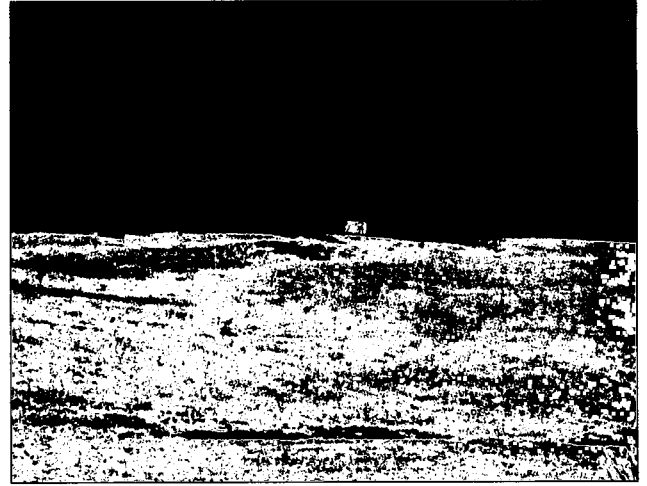


Figure A.3.22: February 23, 2006 SA 180°



Figure A.3.23: February 24, 2006 SA 180°



Figure A.3.24: February 28, 2006 SA 180°



Figure A.3.25: March 2, 2006 SA 180°

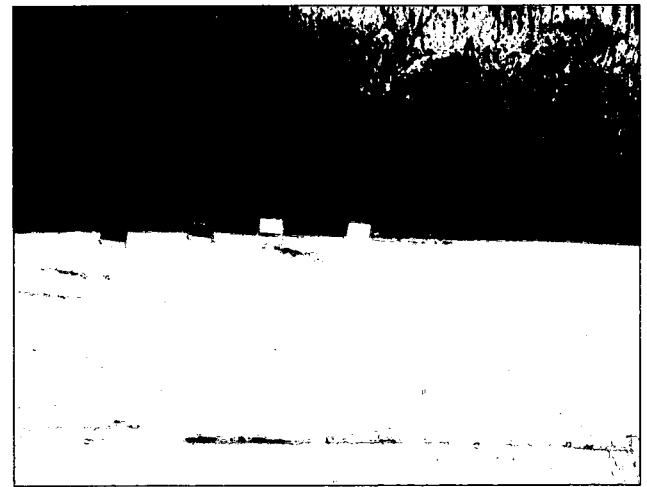


Figure A.3.26: March 7, 2006 SA 180°

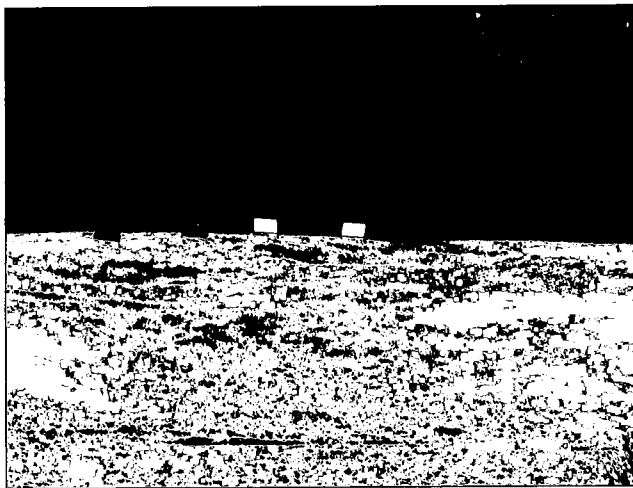


Figure A.3.27: March 27, 2006 SA 180°

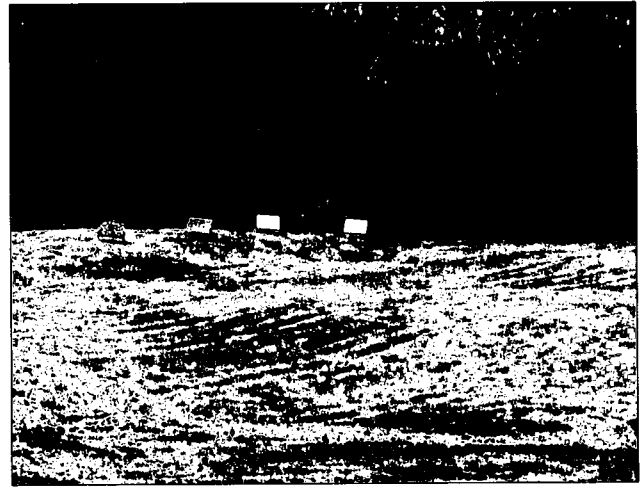


Figure A.3.28: April 10, 2006 SA 180°



Figure A.3.29: April 11, 2006 SA 180°



Figure A.3.30: April 18, 2006 SA 180°



Figure A.3.31: May 9, 2006 SA 180°



## APPENDIX B

### MATLAB Code

This section lists the important functions and files used in MATLAB for analyzing, correcting, creating, and viewing data. Below each function name is a brief description of the operations performed by that function.

#### B.1 Calibration Code

These functions are used for radiometric calibration and characterizing the data sets collected by the sensor.

##### **fcalibrate**

```
function [size, wavelengths]=fcalibrate(base_name, bad_frame, row_range, wav_range, b,
Ts)

% function [size, wavelengths]=fcalibrate(base_name, bad_frame, row_range, wav_range,
b, Ts)
%
% This function performs a complete calibration on a raw data cube. The
% data cube (.envi) and associated radiometric light and dark files must be
% in the path when using this function. The calibration occurs in these
% steps: Second Order Correction, Filtering, Radiometric Calibration,
% Resampling, Cropping, and Discontinuity Correction. The final data
% file is in BIL format with 'double' precision.
%
% size:                Size of the calibrated data set [bands rows columns]
%
% wavelengths:         Wavelengths of the bands in calibrated data set (nm)
%
% base_name:           Name of file to be calibrated w/o extension
%
% bad_frame:           Garbage frame produced during data capturing
%                       due to pause (default of 712)
%
% row_range:           Rows to keep from initial data set
%                       [row_start:row_stop] (default [1:800])
%
% wav_range:           Range of wavelengths to keep [wav_start wav_stop]
%                       (default [450 950])
%
% b:                   FIR filter coefficients for filtering data
%                       (default is Kaiser Window design)
%
% Ts:                  Sampling period for filtered data
%                       (default of 8)
```

```

%
%
%   Author:      Joe Meola
%   Last Update:  11 January 2006

%% CALIBRATION SPECS

%   constants
bands=1024;           % Bands for raw data
rows=1024;            % Rows for raw scene data
columns=1024;         % Columns for raw scene data
frame_start=1;        % Start frame for radiometric files to be avg
frame_stop=100;       % Stop frame for radiometric files to be avg

%   Default Sampling Period
if(nargin < 6)
    Ts=8;              % Sampling period after filtering
end

%   Default Filter
if(nargin < 5)
    b=[0.0505 0.1613 0.3371 0.5573 0.7768 0.9397 1.0000 0.9397 0.7768 0.5573 0.3371
0.1613 0.0505];
end

%   Default Wavelength Range
if(nargin < 4)
    wav_range=[450 950]; % Viable wavelength range on focal plane
end

%   Default Row Range
if(nargin < 3)
    row_range=[1:800]; % Viable row range on focal plane
end

%   Default Bad Frame
if(nargin < 2)
    bad_frame=712;      % Bad frame due to pause while collecting
end

% Initial wavelength mapping on focal plane
wavelength=linspace(112.6883+1.2017,112.6883+1.2017*1024,1024);

% Determine Light and Dark Integrating sphere levels
rad_file_name='Radiometric_light_Params.txt';
[light_level, dark_level]=fparams(rad_file_name);

%% RADIOMETRIC MEAN CALCULATION
base_name_light='Radiometric_light';
base_name_dark='Radiometric_dark';
Cmean=fmean(base_name_light, rows, columns, frame_start, frame_stop);
Cmean=fmean(base_name_dark, rows, columns, frame_start, frame_stop);

%% SECOND ORDER CORRECTION
file_name=strcat(base_name, '.envi');
file_name_dark=strcat(base_name_dark, 'mean.raw');
base_name_light=strcat(base_name_light, 'mean');
fso(base_name_light, file_name_dark, 'radiometric', bad_frame);
fso(base_name, file_name_dark, 'scene', bad_frame);

%% FILTERING
base_name_light=strcat(base_name_light, '_2correct');
base_name1=strcat(base_name, '1');
file_name=strcat(base_name1, '.envi');
ffilter(base_name_light, 'radiometric', b);
ffilter(base_name1, 'scene', b);
delete(strcat(base_name1, '.envi'));

%% RADIOMETRIC CALIBRATION
base_name_light=strcat(base_name_light, '_filtered');

```

```

base_name2=strcat(base_name, '2');
frcal(base_name2, base_name_light, light_level, dark_level);
delete(strcat(base_name2, '.envi'));

%% RESAMPLING
base_name3=strcat(base_name, '3');
wav_samp=fsample(base_name3, Ts);
delete(strcat(base_name3, '.envi'));

%% CROPPING
base_name4=strcat(base_name, '4');
wav_crop=fcrop(base_name4, wav_range, row_range, wav_samp);
size=length(wav_crop) length(row_range) 1024];
delete(strcat(base_name4, '.envi'));

%% DISCONTINUITY CORRECTION
base_name5=strcat(base_name, '5');
fdc(base_name5, size);
delete(strcat(base_name5, '.envi'));

%% FINAL VALUES
wavelengths=wav_crop;

```

## fcrop

```

function wav_crop=fcrop(base_name, wav_range, row_range, wavelength)

% function wav_crop=fcrop(base_name, wav_range, row_range, wavelength)
%
% This function takes a data file and crops off unwanted bands and rows.
% The resulting cropped file is written directly to disk. The function
% outputs the cropped wavelength values. Data must be in BIL format and
% possess standard number of rows (1024) and columns (1024)
%
% wav_crop: Resulting wavelengths of file after
% cropping ( 1 x N)
%
% base_name: Base name of data file w/o .envi extension
%
% wav_range: Desired range of wavelengths (nm) [start_band stop_band]
%
% row_range: Desired row range [start_row:stop_row]
%
% wavelength: vector of wavelengths of file to be cropped (nm)
%
%
% Author: Joe Meola
% Last Update: 13 October 2005

% Input and Output file names
file_name=strcat(base_name, '.envi');
file_name_out=strcat(base_name(1:length(base_name)-1), '5.envi');

% Data Size assuming standard row and column size
rows=1024;
columns=1024;
bands=length(wavelength);

% Determine write range (remember image is flipped vertically)
row_write=row_range;

% Data format (default of 'double');
precision='double';
bytes=8;

% Determine index of desired bands and resulting wavelengths of output
wav_ind=find( (wavelength > wav_range(1)) & (wavelength < wav_range(2)) );

```

```

wav_crop=wavelength(wav_indx);

% Determine number of bands after cropping
N=length(wav_indx);

% Open files
fid_read=fopen(file_name);
fid_write=fopen(file_name_out, 'w');

% Cycle through file grabbing one column at a time in desired bands
for n=1:columns

    if(n==1) % 1st time around

        % Go to start band
        fseek(fid_read, (wav_indx(n)-1)*rows*bytes, 'cof');

        %determine location of start band
        position=ftell(fid_read);

        % Read in all rows and N bands
        c=fread(fid_read, [rows N], precision);

    else

        % Starting location of columns after 1st column will be
        % multiples of 1st column locations
        % for n>1, position column n is equal to position
        % of first column + rows*bands*bytes
        fseek(fid_read, position+((n-1)*rows*bands*bytes), 'bof');

        % Read in all rows and N bands
        c=fread(fid_read, [rows N], precision);

    end

    % Write out desired row range and bands
    fwrite(fid_write, c(row_write,:), precision);

end

% Close files
fclose(fid_read);
fclose(fid_write);

```

## fdc

```

function fdc(base_name, size);

% function fdc(base_name, size);
%
% This function corrects the discontinuity present in the left beginning
% and end portions of the scene data. The exact cause of the
% discontinuity is unknown currently. The correction is performed by
% averaging a range of rows in each band on either side of the
% discontinuity and determining a gain to apply for correction.
%
% base_name:          Base name of file w/o extension
%
% size:              Size of data [bands rows columns]
%
%
% Author:            Joe Meola
% Last Update:       11 January 2006

if(nargin < 2)
    size=[52 800 1024];

```

```

end

bytes=8;

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% Discontinuity column ranges
A=[1:42];
B=[43:685];
C=[686:1024];

% Set up mean values for columns on disc borders
meanA=zeros(1, bands);
meanAB=zeros(1, bands);
meanBC=zeros(1, bands);
meanC=zeros(1, bands);

% Use only grass region as is more uniform
r=[1:350];

file_name=strcat(base_name, '.envi');
file_name_out=strcat(base_name(1:length(base_name)-1), '_final.envi');

fid_read=fopen(file_name);

% Determine mean values using column on either side of discontinuity
status=fseek(fid_read, rows*bands*(A(end)-1)*bytes, 'bof');
Im=fread(fid_read, [rows, bands], 'double');
meanA= mean(Im(r,:));

Im=fread(fid_read, [rows, bands], 'double');
meanAB= mean(Im(r,:));

status=fseek(fid_read, rows*bands*(B(end)-1)*bytes, 'bof');
Im=fread(fid_read, [rows, bands], 'double');
meanBC= mean(Im(r,:));

Im=fread(fid_read, [rows, bands], 'double');
meanC= mean(Im(r,:));

fclose(fid_read);

% Determine discontinuity gains
dA=meanAB./meanA;
dC=meanBC./meanC;

% Create matrix for multiplication
multA=repmat(dA, rows,1);
multC=repmat(dC, rows,1);

fid_read=fopen(file_name);
fid_write=fopen(file_name_out,'w');

for n=1:columns

    Im=fread(fid_read, [rows, bands], 'double');

    if (n <= A(end))

        % Correct Column
        Im_out= Im.*multA;

        % Flip image upright
        Im_out=flipud(Im_out);

    elseif ( (n >= B(1) ) & (n <= B(end) ) )

        % Flip image upright

```

```

        Im_out=flipud(Im);

elseif ( (n >= C(1) ) & (n <= C(end) ) )

    % Correct Column
    Im_out= Im.*multC;

    % Flip image upright
    Im_out=flipud(Im_out);

end

% Eliminate any negative radiance values
[r b]=find( Im_out < 0);
Im_out(r,b)=0.01;

% Write out to file
fwrite(fid_write, Im_out, 'double');

end

fclose(fid_read);
fclose(fid_write);

```

## ffilter

```

function ffilter(base_name, file_type, b);

%function ffilter(base_name, file_type, b);
%
% This function applies a linear phase FIR filter to radiometric or scene
% data. The scene and radiometric files being corrected must be in BIL
% format. The resulting filtered files are written to disk with
% '_filtered' appended on the original file name. The default filter is
% length a length 13 FIR filter designed using a Kaiser window.
%
% Note: Matlab fills via column when using fread
%
% base_name:          The base name (w/o extension) of the file being
%                     corrected.
%
% file_type:          String declaring the type of file being
%                     filtered. Either "radiometric" or "scene"
%
% b:                  Filter coefficients (must be even symmetric to ensure Linear
%                     Phase)
%
% Run Time ~19.0400 minutes for scene data
%
% Note: Matlab fills via column when using fread
%
% Author:             Joe Meola
% Last Update:        21 December 2005
% Update Reason:      Changed default filter to kaiser design

%Filtering of mean data
if(nargin < 3)
    % Default filter coefficients (Kaiser window design)
    b=[0.0505 0.1613 0.3371 0.5573 0.7768 0.9397 1.0000 0.9397 0.7768 0.5573 0.3371
    0.1613 0.0505];
end

rows=1024;
columns=1024;

if (strcmp(file_type, 'radiometric'))
    file_name=strcat(base_name, '.raw');

```

```

%Open file
fid_read=fopen(file_name);
A=fread(fid_read, [rows columns], 'double');
fclose(fid_read);

%filter file
Afilter=conv2(A, b, 'same');

%Write filtered file to disk
file_name_out=strcat(base_name, '_filtered.raw');
fid_write=fopen(file_name_out, 'w');
count=fwrite(fid_write, Afilter, 'double');
fclose(fid_write);

end

if(strcmp(file_type, 'scene'))

    %Open file
    file_name=strcat(base_name, '.envi');
    fid_read=fopen(file_name);

    %Open file to write to
    file_name_out=strcat(base_name(1:length(base_name)-1), '2.envi');
    %   file_name_out=strcat(base_name, '_filtered.envi');
    fid_write=fopen(file_name_out, 'w');

    for n=1:columns

        %Read in one frame at a time
        A=fread(fid_read, [rows columns], 'double');

        %Filter
        Afilter=conv2(A, b, 'same');

        %Write to disk sequentially
        fwrite(fid_write, Afilter, 'double');

    end
    fclose(fid_read);
    fclose(fid_write);
end

```

## fmean

```

function Cmean=fmean(base_name, rows, columns, frame_start, frame_stop)

%   function Cmean=fmean(base_name, rows, columns, frame_start, frame_stop)
%
%   This function averages a given number of frames together for
%   radiometric calibration purposes. The function returns the average
%   frame and also writes out the average frame to file with a default .raw extension.
%   The files are also corrected for rotation. The output file is written with double
%   precision.
%   "mean" is added to the base name as the write out name.
%   The file is transposed and written out in a BIL format.
%
%   Note: Matlab fills via column when using fread
%
%   Cmean:                Average frame (1024 x 1024)
%
%   base_name:            The base name of the frame files to be averaged
%
%   frame_start:          Frame to start averaging
%
%   frame_stop:           Frame to stop averaging
%
%   rows:                 Rows per frame

```

```

%
% columns:           Columns per frame
%
%
% Author:            Joe Meola
% Last Update:       28 November 2005
% Update Reason:     Added rotational error correction

```

```
out_base_name=strcat(base_name, 'mean');
```

```

A=zeros(rows,columns);
Cmean=zeros(rows,columns);

```

```

index=[1:columns];
frames=frame_stop-frame_start + 1;

```

```

% Average frames
for n=frame_start:frame_stop

    frame=sprintf('%06g',n);
    file=strcat(base_name,frame,'.raw');

    fid_read=fopen(file);
    [A,count] = fread(fid_read,[columns rows],'uint16');

    % Transpose to create BIL
    A=A';

    Cmean=A/frames + Cmean;
    fclose(fid_read);
end

```

```

% Correct Rotational Error
Cmean=imrotate(Cmean, 0.076823, 'bilinear', 'crop');

```

```

%Write out mean file to current directory
file_out=strcat(out_base_name, '.raw');

```

```

fid_write=fopen(file_out, 'w');
count=fwrite(fid_write, Cmean, 'double');
fclose(fid_write);

```

## **fmean\_var**

```
function [Cvar, Cmean]=fmean_var(base_name, rows, columns, frame_start, frame_stop)
```

```

% function [Cvar, Cmean]=fmean_var(base_name, rows, columns, frame_start, frame_stop)
%
% This function determines the mean and variance between a number of frames.
% The function returns the mean and variance frames and also writes out the frames
% to file with a default or user specified name and .raw extension. The output file
% is written with double precision. "mean" and "var" are added
% to the base name as the write out name.
%
% Cmean:           Mean frame (rows x columns)
%
% Cvar:            Variance frame (rows x columns)
%
% base_name:       The base name of the frame files (string)
%
% rows            Rows per frame
%
% columns          Columns per frame
%
% frame_start:     Frame number to start at
%
% frame_stop:      Frame number to stop at

```



```

% Default output file names
out_var_base_name=strcat(base_name, 'var');
out_mean_base_name=strcat(base_name, 'mean');

A=zeros(rows, columns);
Cvar=zeros(rows, columns);
Cmean=zeros(rows, columns);

index=[1:columns];
frames=frame_stop - frame_start + 1;

% Determine Variance
for n=frame_start:frame_stop

    frame=sprintf('%06g',n);

    file=strcat(base_name,frame,'.raw');

    fid_read=fopen(file);
    [A,count] = fread(fid_read,[columns rows],'uint16');

    % Transpose to create BIL format
    A=A';

    Cvar=(A.^2)/frames + Cvar;
    Cmean=A/frames + Cmean;

    fclose(fid_read);
end

%Compute variance
Cvar=Cvar - Cmean.^2;

%Create output file names
file_out_mean=strcat(out_mean_base_name, '.raw');
file_out_var=strcat(out_var_base_name, '.raw');

%Write out mean file to current directory
fid_write=fopen(file_out_mean, 'w');
count=fwrite(fid_write, Cmean, 'double');
fclose(fid_write);

%Write out variance file to current directory
fid_write=fopen(file_out_var, 'w');
count=fwrite(fid_write, Cvar, 'double');
fclose(fid_write);

```

## **fparams**

```

function [light_level, dark_level]=fparams(file_name);

% function [light_level, dark_level]=fparams(file_name);
%
% This function reads the Radiometric Parameters text file to determine
% the integrating sphere light level data.
%
% file_name:          Name of parameter file
%                    ('Radiometric_light_Params.txt')
%
% light_level:        Integrating sphere light level
%
% dark_level:         Dark current offset level
%
% Author:             Joe Meola
% Last Update:        12 May 2006

```

```

% Open parameter file
fid_read = fopen(file_name);

% Read text file one line at a time to determine desired info
while 1

    % Get one line of text
    line = fgetl(fid_read);

    % If finished, fgetl returns -1
    if (line==-1)
        break
    end

    % Look for colon
    [first,second]=strtok(line,':');

    % Check if desired parameter
    switch first

        case 'Light'
            [f,s]=strtok(second);
            light_level=str2double(s);

        case 'Dark'
            [f,s]=strtok(second);
            dark_level=str2double(s);

    end
end

% close header file
fclose(fid_read);

```

## **frcal**

```

function frcal(base_name, rad_file_name, light_level, dark_level);

%function frcal(base_name, rad_file_name, light_level, dark_level);
%
% This function determines and applies a radiometric gain to scene data.
% The scene data must already be filtered and corrected for 2nd order effects,
% which in turn removes the dark offset. The radiometric gain is determined
% using a mean "light" frame from the integrating sphere and the corresponding
% luminance value of that "light" frame. USS600_cal_dat.mat must exist
% in the path for successful operation. The scene must be in BIL format.
% The resulting file is written with "_calibrated" appended onto the
% base_name.
%
% base_name:          The base name of the scene file to be corrected
%                     (.envi extension)
%
% rad_file_name:      The base name of the radiometric data file to use
%                     for correction. (.raw extension)
%
% light_level:        The measured luminance light level in foot-lamberts
%                     from the integrating sphere at the time of capture.
%
% dark_level:         The measured luminance dark level in foot-lamberts
%                     from the integrating sphere at the time of capture.
%
% RUN TIME ~22 minutes for scene data
%
% Author:             Joe Meola
% Last Update:        28 November 2005
% Update Reason:      Removed row dependency because rotation error corrected

load USS600_cal_data;

```

```

warning off MATLAB:divideByZero

% Indexing wavelength using:  lambda=1.2017*col  + 112.6883 (nm)
rows=1024;
columns=1024;
bands=1024;

row=[1:rows];
col=[1:columns];
wav_x=1.1933*col;
wav_x=repmat(wav_x, [rows 1]);
offset=106.5198;

% Create matrix indexing wavelength at every pixel
wavelength_mat=wav_x + offset;

L_634=zeros(length(row), length(col));

% Create Spectral Radiance Matrix from calibrated specs
for n=1:length(row)
    L_634(n, :)=interp1(wavelength, spec_rad, wavelength_mat(n,:), 'linear', 'extrap');
end

%Light level at time of calibration (fL)
cal_measure=634.8;
gain=(light_level - dark_level)/cal_measure;

%Adjust known integrating sphere data to actual measured light level
L_known=L_634*gain;

%Open radiometric light file already corrected for Second Order and
%filtered
rad_file=strcat(rad_file_name, '.raw');
fid_read=fopen(rad_file);
L_measured=fread(fid_read, [1024 1024], 'double');
fclose(fid_read);

%Determine Radiometric Gain
Radiometric_Gain=L_known./L_measured;

%Open file to be corrected
file_name=strcat(base_name, '.envi');
fid_read=fopen(file_name);

%Open file to write calibrated data
file_name_out=strcat(base_name(1:length(base_name)-1), '3.envi');
% file_name_out=strcat(base_name, '_calibrated.envi');
fid_write=fopen(file_name_out, 'w');

%Apply Gain to Second Order Corrected and filtered Scene and save
for n=1:columns

    A=fread(fid_read, [rows bands], 'double');

    Arad=A.*Radiometric_Gain;

    fwrite(fid_write, Arad, 'double');

end

fclose(fid_read);
fclose(fid_write);

```

## fsample

```

function wav_samp=fsample(base_name, T)

% function sampled_bands=fsample(base_name, T)

```

```

%
% This function will sample bands in a given filtered data file and write
% out the sampled file to disk. The data must be in BIL format and must be
% in original form, i.e., 1024 rows, 1024 columns, and 1024 bands. The
% program will sample every T bands, starting at band T. The file name
% must possess a .envi extension. The resulting output file will have
% "_spT" appended on the base_name, where T will be the sampling period.
%
%   sampled_bands:      The locations of the bands sampled
%
%   base_name:          Name of file to be sampled (.envi)
%
%   T:                  Sampling Period
%
%
%   Author:      Joe Meola
%   Updated:     5 October 2005

%% Size of original data file
columns=1024;
rows=1024;
bands=1024;

% Determine precision of data
precision='double'
bytes=8;

% Input and Output file names
file_name=strcat(base_name, '.envi');
file_name_out=strcat(base_name(1:length(base_name)-1), '4.envi');
% file_name_out=strcat(base_name, '_sp',num2str(T),'.envi');

% Open files for reading and writing
fid_read=fopen(file_name);
fid_write=fopen(file_name_out, 'w');

% Determine position of bands sampled
sampled_bands=T:T:bands;
sampled_bands=sampled_bands(1:length(sampled_bands)-1);

% Determine bands sampled
wavelength=linspace(106.5198+1.1933,106.5198+1.1933*1024,1024);
wav_samp=wavelength(sampled_bands);
N=length(T:T:bands)-1;      %Number of bands sampled

%Sampling Process
for n=1:columns

    for m=1:N

        if(n==1) % 1st time around

            fseek(fid_read, (T-1)*rows*bytes, 'cof');    %Start at band T
            position(m)=ftell(fid_read);                  %determine locations of sampling

            % Read in column 1 in band T*m
            c=fread(fid_read, rows, precision);

        else

            % Starting location of samples after 1st column will be
            % multiples of 1st column locations
            % for n>1, position column n in band T*m is equal to position
            % of first column in band T*m + rows*bands*bytes
            fseek(fid_read, position(m)+((n-1)*rows*bands*bytes), 'bof');

            % Read in column n in band T*m
            c=fread(fid_read, rows, precision);
        end
    end
end

```

```

        % Write out column n in band T*m
        fwrite(fid_write, c, precision);

    end

end

% Close files
fclose(fid_read);
fclose(fid_write);

```

## fso

```

function fso(base_name, file_name_dark, file_type, bad_frame);

% function fso(base_name, file_name_dark, file_type, bad_frame)
%
% This function corrects second order effects within a scene or
% radiometric file. The second order gains corresponding to the
% diffraction grating have already been calculated and are stored in the
% file 'Second_Order_Gains.raw'. This file must exist in a path folder
% for proper operation of the function. The user specifies via string if
% a "scene" file or "radiometric" file is being corrected. This
% correction must be applied to the raw scene taken directly from the
% spectrometer as it corrects for second order, removes unwanted frames,
% and converts the data to a BIL format. The radiometric file must be a mean
% file already in BIL format. The resulting files are written to disk
% with '_2correct' appended to the original file name.
%
% Note: Matlab fills via column when using fread
%
% base_name:          The base name of the file to be corrected
%
% file_name_dark:     The file name of the mean dark file
%
% file_type:          'scene' or 'radiometric' for type of file being corrected
%
% bad_frame:          Garbage frame due to delay in capture (either
%                     512 or 712)
%
% Run Time ~8.1 minutes for scene data
%
% Author:             Joe Meola
% Last Update:        28 November 2005
% Update Reason:      Added rotational error correction for scene and changed
%                     Second order columns and gains

if(nargin < 4)
    bad_frame=512;
end

%Open Second Order Gain file
fid=fopen('Second_Order_Gains.raw');
gains=fread(fid,[1024 411],'double');
fclose(fid);

%Open Mean Dark file
fid=fopen(file_name_dark);
mean_dark=fread(fid,[1024 1024],'double');
fclose(fid);

%Number of frames taken during scan of scene
frames=1029;
rows=1024;
columns=1024;

% Standard columns to be corrected and corresponding 1st order range
column_start2=613;          %Wavelength=~849.3304nm

```

```

column_end2=1023;                                %Wavelength=~1342nm
total=(column_end2 - column_start2)+1;
column_start=259;                                %Wavelength=~423.9286nm
column_end=464;                                   %Wavelength=~670.2771nm

%Correction for radiometric files
if (strcmp(file_type, 'radiometric'))
    file_name=strcat(base_name, '.raw');

    %Open file to be corrected
    fid=fopen(file_name);
    A=fread(fid, [1024 1024], 'double');
    fclose(fid);

    %Subtract Dark Offset
    A=A-mean_dark;

    %Set up second order matrix
    second_order=zeros(rows, total);

    % Calculate second orders present
    second_order(:, 1:2:total)=0.5*(A(:, column_start:column_end));
    second_order(:, 2:2:total)=0.137035*(A(:, column_start:column_end-1))+0.362965*...
    (A(:, column_start+1:column_end));

    % Remove Second Order

    A_fix=A;
    A_fix(:, column_start2:column_end2)=A(:, column_start2:column_end2) -
    second_order.*gains;

    %Write corrected file to disk
    file_name_out=strcat(base_name, '_2correct.raw');
    fid=fopen(file_name_out, 'w');
    count=fwrite(fid, A_fix, 'double');
    fclose(fid);

end

%Correction for scene files
if(strcmp(file_type, 'scene'))

    %Open scene to be corrected
    file_name=strcat(base_name, '.envi');
    fid=fopen(file_name);

    %Open file to write to
    file_name_out=strcat(base_name, '1.envi');
    fid2=fopen(file_name_out, 'w');

    %Read through scene one frame at a time
    %Skip bad frames 1, 712, 1027-1029
    for n=1:frames-3

        A=fread(fid, [rows columns], 'uint16');

        %leave out these frames
        if (n~=1) & (n~=bad_frame) )

            %transpose converts to BIL
            A=A';

            % Correct Rotational Error
            A=imrotate(A, 0.076823, 'bilinear', 'crop');

            %Subtract dark offset
            A=A - mean_dark;

```

```

        %Set up second order matrix
        second_order=zeros(rows, total);

        % Calculate second orders present
        second_order(:,1:2:total)=0.5*(A(:,column_start:column_end));
        second_order(:,2:2:total)=0.137035*(A(:,column_start:column_end-
        1))+0.362965*...
        (A(:,column_start+1:column_end));

        % Remove Second Order
        A_fix=A;
        A_fix(:, column_start2:column_end2)=A(:, column_start2:column_end2) -
        second_order.*gains;

        %Write to file sequentially
        fwrite(fid2, A_fix, 'double');
    end
end
fclose(fid);
fclose(fid2);
end

```

## fvar

```

function Cvar=fvar(base_name, rows, columns, frame_start, frame_stop, Cmean)

% function Cmean=fvar(base_name, rows, columns, frame_start, frame_stop, Cmean)
%
% This function determines the variance between a number of unformatted frames. The
% function returns the variance frame and also writes out the variance frame to file
with a default or
% user specified name and .raw extension. The output file is written
% with double precision. "var" is added to the base name as the write out name.
%
% Cvar:                Variance frame (rows x columns)
%
% base_name:           The base name of the frame files (string)
%
% rows                Rows per frame
%
% columns             Columns per frame
%
% frame_start:         frame number to start at
%
% frame_stop:          frame number to stop at
%
% Cmean:               Average of frames (rows x columns)
%
%
% Author:              Joe Meola
% Last Update:         10 August 2005

out_base_name=strcat(base_name, 'var');

A=zeros(rows, columns);
Cvar=zeros(rows, columns);

index=[1:columns];
frames=frame_stop - frame_start +1;

% Determine Variance
for n=frame_start:frame_stop

    frame=sprintf('%06g',n);
    file=strcat(base_name,frame, '.raw');

```

```

    fid=fopen(file);
    [A,count] = fread(fid,[rows columns],'uint16');

    A=A';                                %Fix orientation of frame

    Cvar=(A.^2)/frames + Cvar;

    fclose(fid);
end
%Variance computation
Cvar=Cvar - Cmean.^2;

%Write out variance file to current directory
file_out=strcat(out_base_name, '.raw');
fid=fopen(file_out, 'w');
count=fwrite(fid, Cvar, 'double');

```

## B.2 Data Conversion Code

These functions are used for Principal Component Transforms and for making predictions associated with change detection.

### fpca

```

function fpca(base_name, C, mn, Nk, size, precision, format)
%
%   function fpca(base_name, C, mn, Nk, size, precision, format);
%
%   This function converts a hyperspectral data set into its principal
%   component space based upon the scene statistics
%
%   base_name:           Base data file name w/o extension
%
%   C:                   Covariance Matrix (bands x bands)
%
%   mn:                   Spectral Mean (1 x bands)
%
%   Nk:                   Number of PC to keep
%
%   size:                 Size of hypercube [bands rows columns]
%                         (default [124 800 1024])
%
%   precision:           Data precision: 'uint16' or 'double'
%                         (default 'double')
%
%   format:              Data format: 'BIL' or 'BIP'
%                         (default 'BIL')
%
%
%   Author:              Joe Meola
%   Last Update:         23 February 2006
%
%   Default Data size
if(nargin < 5)
    size=[124 800 1024];
end

%   Default Data precision
if(nargin < 6)
    precision='double';
end

```



```

% Default Data format
if(nargin < 7)
    format='BIL';
end

% Default number of prin components to keep
if(nargin < 4)
    Nk=10;
end

% Calculate covariance matrix and mean if not provided
if(nargin < 3)
    if(nargin < 2)
        [C, mn]=fcov(base_name, size, format, precision);
    else
        mn=fspec_mean(base_name, size, format, precision);
    end
end

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% Input and Output files
file_name=strcat(base_name, '.envi');
file_name_out=strcat(base_name, '_pca.envi');

% Determine eigenvectors and eigenvalues
[V,E] = eig(C);

% Flip eigenvectors so in order from greatest to least
if(E(1,1) < E(end, end) )
    V=fliplr(V);
end

% BIL Format
if(strcmp(format, 'BIL'))

    fid_read=fopen(file_name);
    fid_write=fopen(file_name_out, 'w');

    % Cycle through file reading in one column of hyperpixels at a time
    for n=1:columns

        A=fread(fid_read, [rows bands], 'double');

        % cycle through each hyperpixel
        for m=1:rows
            % Apply PCA transform
            D(m,:)=(A(m,:)-mn)*V;
        end
        % write out prin components to keep
        fwrite(fid_write, D(:,1:Nk), 'double');

    end

    fclose(fid_read);
    fclose(fid_write);

end

```

## **fcc**

```

function fcc(base_name1, base_name2, mnx, Cx, mny, Cxy, size, format, precision);
%
% function fcc(base_name1, base_name2, mnx, Cx, mny, Cy, size, precision, format);
%
% This function performs a Chronochrome linear estimation on hyperspectral

```

```

% data from one time using the statistics of the data and data taken at a
% later time. The estimated data file and an estimation error file are both
% written to disk with "_CE" and "_CEerror" added to the base name.
%
% base_name1:          Base name of time-1 data
%
% base_name2:          Base name of time-2 data
%
% Cx:                  Covariance Matrix of time-1 data (bands x bands)
%
% Cxy:                 cross-covariance Matrix of time-1/time-2 data (bands x bands)
%
% mnx:                 Spectral Mean of time-1 (1 x bands)
%
% mny:                 Spectral Mean of time-2 (1 x bands)
%
% size:                Size of hypercube [bands rows columns]
%                     (default [124 800 1024])
%
% precision:           Data precision: 'uint16' or 'double'
%                     (default 'double')
%
% format:              Data format: 'BIL' or 'BIP'
%                     (default 'BIL')
%
%
% Author:              Joe Meola
% Last Update:         18 May 2006
%
% Default Data size
if(nargin < 7)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 9)
    precision='double';
end

% Default Data format
if(nargin < 8)
    format='BIL';
end

% File names
file_name1=strcat(base_name1, '.envi');
file_name2=strcat(base_name2, '.envi');

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% Linear Transform
L=Cxy*inv(Cx);

% Output files

file_name_out_error=strcat(base_name2, '_CCerror.envi');

% Open files
fid_read1=fopen(file_name1);
fid_read2=fopen(file_name2);
fid_error=fopen(file_name_out_error, 'w');

% BIL format
if(strcmp('BIL', format))

% Cycle through file reading in (rows) pixels at a time
for n=1:columns

```

```

    % Read time-1 and time-2 data
    x=fread(fid_read1, [rows bands], precision);
    y=fread(fid_read2, [rows bands], precision);

    % Cycle through every hyperpixel
    for m=1:rows
        % Perform transform and determine error
        y_hat(:,m)=L*(x(m,:)-mnx)';
        e(:,m)=y_hat(:,m) - (y(m,:)-mny)';
    end
    fwrite(fid_error, e', precision);

end

% BIP Format
elseif(strcmp(format, 'BIP'))

    for n=1:columns

        % Read in one column at a time in all bands
        x=fread(fid_read1, [bands rows], 'double');
        y=fread(fid_read2, [bands rows], 'double');

        % Cycle through every hyperpixel
        for m=1:rows
            y_hat(:,m)=L*(x(:,m)-mnx)';
            e(:,m)=y_hat(:,m) - (y(:,m) - mny)';
        end
        fwrite(fid_error, e, precision);
    end
end
else
    error(strcat(format, ' Invalid Data Format'));
end

% Close files
fclose(fid_read1);
fclose(fid_read2);
fclose(fid_error);

```

## fce

```

function fce(base_name1, base_name2, mnx, Cx, mny, Cy, size, format, precision);
%
% function fce(base_name1, base_name2, mnx, Cx, mny, Cy, size, precision, format);
%
% This function performs a Covariance Equalization linear estimation on hyperspectral
% data from one time using the statistics of the data and data taken at a
% later time. The estimated data file and an estimation error file are both
% written to disk with "_CE" and "_CEerror" added to the base name.
%
% base_name1:          Base name of time-1 data
%
% base_name2:          Base name of time-2 data
%
% Cx:                  Covariance Matrix of time-1 data (bands x bands)
%
% Cy:                  Covariance Matrix of time-2 data (bands x bands)
%
% mnx:                  Spectral Mean of time-1 (1 x bands)
%
% mny:                  Spectral Mean of time-2 (1 x bands)
%
% size:                Size of hypercube [bands rows columns]
%                      (default [124 800 1024])
%
% precision:           Data precision: 'uint16' or 'double'
%                      (default 'double')
%
% format:              Data format: 'BIL' or 'BIP'

```

```

%                               (default 'BIL')
%
%
%
%   Author:      Joe Meola
%   Last Update:  22 March 2006

%   Default Data size
if(nargin < 7)
    size=[124 800 1024];
end

%   Default Data precision
if(nargin < 9)
    precision='double';
end

%   Default Data format
if(nargin < 8)
    format='BIL';
end

%   Data size
bands=size(1);
rows=size(2);
columns=size(3);

%   Input and Output files
file_name=strcat(base_name1, '.envi');
file_name2=strcat(base_name2, '.envi');
file_name_out_error=strcat(base_name2, '_CEError.envi');

Ky=Cy^(1/2);
Kx=Cx^(-1/2);

% Affine transform  y_hat=Lx
L=Ky*Kx;

% Open files
fid_read=fopen(file_name);
fid_read2=fopen(file_name2);
% fid_write=fopen(file_name_out, 'w');
fid_error=fopen(file_name_out_error, 'w');

%   BIL format
if(strcmp('BIL', format))

%   Cycle through file reading in (rows) pixels at a time
for n=1:columns

    % Read time-1 and time-2 data
    x=fread(fid_read, [rows bands], precision);
    y=fread(fid_read2, [rows bands], precision);

    % Cycle through every hyperpixel
    for m=1:rows
        % Perform transform and determine error
        y_hat(:,m)=L*(x(m,:)-mnx)';
        e(:,m)=y_hat(:,m) - (y(m,:)-mny)';
    end
    fwrite(fid_error, e', precision);

end

%   BIP Format
elseif(strcmp(format, 'BIP'))

%   Cycle through file reading in (rows) pixels at a time
for n=1:columns

```

```

    % Read time-1 and time-2 data
    x=fread(fid_read, [bands rows], precision);
    y=fread(fid_read2, [bands rows], precision);

    % Cycle through every hyperpixel
    for m=1:rows
        % Perform transform and determine error
        y_hat(:,m)=L*(x(:,m)-mnx');
        e(:,m)=y_hat(:,m) - (y(:,m)-mny');
    end
    fwrite(fid_error, e, precision);

end

else
    error(strcat(format, ' Invalid Data Format'));
end

% Close files
fclose(fid_read);
fclose(fid_read2);
fclose(fid_error);

```

## fTchange

```

function Ty=fTchange(base_name1, base_name2, Tx, method, size, format, precision);
%
% function Ty=fTchange(base_name1, base_name2, Tx, method, size, format, precision);
%
% This function performs linear transform on a target spectrum. The linear
% transform methods used are Covariance Equalization 'CE', or
% Chronochrome, 'CC'.
%
% Ty: Estimated time-2 target spectrum [1 x bands]
%
% base_name1: Base name of time-1 data
%
% base_name2: Base name of time-2 data
%
% Tx: Time-1 target spectrum [1 x bands]
%
% method: Either 'CE' or 'CC'
%
% size: Size of hypercube [bands rows columns]
%       (default [124 800 1024])
%
% precision: Data precision: 'uint16' or 'double'
%            (default 'double')
%
% format: Data format: 'BIL' or 'BIP'
%         (default 'BIL')
%
% Author: Joe Meola
% Last Update: 23 March 2006
%
% Default Data size
if(nargin < 5)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 7)
    precision='double';
end

% Default Data format
if(nargin < 6)
    format='BIL';
end

```

```

end

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% Determine mean and covariance matrices for data sets
[Cx, mnx]=fcov(base_name1, size, format, precision);
[Cy, mny]=fcov(base_name2, size, format, precision);

% Covariance Equalization Method
if( strcmp(method, 'CE'))

    % Perform CE estimation
    L=(Cy^(1/2))*(Cx^-(1/2));
    d=mny - (L*mnx)';
    Ty=(L*Tx)' + d;

% Chronochrome method
elseif( strcmp(method, 'CC'))

    % Determine cross-covariance matrix
    Cxy=fcross_cov(base_name1, base_name2, mnx, mny, size, format, precision);

    % Perform CC estimation
    L=Cxy*inv(Cx);
    d=mny - (L*mnx)';
    Ty=(L*Tx)';

else
    error(strcat(method, ' Invalid Method'));
end

```

### B.3 Data Extraction Code

This code is used for extracting data from specific data sets such as a spectrum for a pixel or the entire scene in a band.

#### fband

```

function A=fband(band, base_name, size, format, precision);

% function A=fband(band, base_name, size, format, precision);
%
% This function returns the image in a specific band of a hyperspectral cube.
% Note: Data file must have .envi extension
%
% A: Image in desired band (rows x columns)
%
% band: Band number (1:bands)
%
% base_name Base data file name w/o extension
%
% size: Size of hypercube [bands rows columns]
% (default [124 800 1024])
%
% precision: Data precision: 'uint16' or 'double'
% (default 'double')
%
% format: Data format: 'BIL' or 'BIP'
% (default 'BIL')
%
%

```

```

%
% Author:      Joe Meola
% Last Update: 5 January 2006

% Default Data size
if(nargin < 3)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 4)
    precision='double';
end

% Default Data format
if(nargin < 5)
    format='BIL';
end

% Data length
if (strcmp(precision, 'uint16'))
    bytes=2;
else
    bytes=8;
end

% Data size
rows=size(2);
columns=size(3);
bands=size(1);

file_name=strcat(base_name, '.envi');

%Reading image in BIL format
if(strcmp(format, 'BIL'))

    %Address offset used with fseek to advance to next column in desired band
    offset=(bands-1)*(rows)*bytes;

    fid=fopen(file_name);

    %Advance from beginning of file to first column in desired band
    status=fseek(fid,(band-1)*bytes*rows,'bof');

    %Read in first column of image
    A(:,1)=fread(fid,rows,precision);

    for n=2:columns

        %Advance to next column in same band
        status = fseek(fid,offset,'cof');
        %Read column
        A(:,n)=fread(fid,rows,precision);
    end

    fclose(fid);

%Reading image in BIP format
elseif(strcmp('BIP',format))

    fid=fopen(file_name);
    skip=bytes*(bands - 1);

    status = fseek(fid,(band-1)*bytes, 'bof');
    A=fread(fid,[rows columns],precision,skip);
    fclose(fid);

else
    error(strcat(format, ' Invalid Data Format'));

```

end

## **fcolor**

```
function Acolor=fcolor(base_name, wavelengths, size, format, precision);

% function Acolor=fcolor(base_name, wavelengths, size, format, precision);
%
% This function creates and displays a color image for a hyperspectral
% data cube. The values used for red, green, and blue are interpolated
% to approximately the middle of the respective color ranges. If no
% wavelengths are provided for the data, default values are loaded.
%
% Acolor:           Color image
%
% base_name:        Base data file name w/o extension
%
% wavelengths:      Wavelengths (nm) associated with data
%
% size:             Size of hypercube [bands rows columns]
%                  (default [124 800 1024])
%
% precision:        Data precision: 'uint16' or 'double'
%                  (default 'double')
%
% format:           Data format: 'BIL' or 'BIP'
%                  (default 'BIL')
%
%
% Author:           Joe Meola
% Last Update:      3 March 2006

% Default Wavelengths
if(nargin < 2)
    load wavelengths;
end

% Default Data size
if(nargin < 3)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 5)
    precision='double';
end

% Default Data format
if(nargin < 4)
    format='BIL';
end

% Find color boundaries and associated indices
blue_idx=find( (wavelengths < 520) & (wavelengths > 450) );
green_idx=find( (wavelengths < 600) & (wavelengths > 520) );
red_idx=find( (wavelengths < 670) & (wavelengths > 600) );

% Interpolate color values
Ared1=fband(red_idx(1), base_name, size, format, precision);
Ared2=fband(red_idx(end), base_name, size, format, precision);
Ared=(Ared1 + Ared2)/2;

Agreen1=fband(green_idx(1), base_name, size, format, precision);
Agreen2=fband(green_idx(end), base_name, size, format, precision);
Agreen=(Agreen1 + Agreen2)/2;

Ablue1=fband(blue_idx(1), base_name, size, format, precision);
Ablue2=fband(blue_idx(end), base_name, size, format, precision);
```



```

Ablue=(Ablue1 + Ablue2)/2;

% Construct color image
Acolor(:,:,1)=Ared;
Acolor(:,:,2)=Agreen;
Acolor(:,:,3)=Ablue;

mx=max(max(max(Acolor)));
mn=min(min(min(Acolor)));

Acolor=(Acolor - mn)/(mx - mn);

% Display Image
figure;
imagesc(Acolor);

% Write Image to current directory
im_name=strcat(base_name, '_color.jpg');
imwrite(Acolor, im_name);

```

## fspectrum

```

function spectrum=fspectrum(base_name, pixel, size, format, precision);

% function spectrum=fspectrum(base_name, pixel, size, format, precision);
%
% This function returns the spectrum of a given pixel.
%
% Note: Data file must have .envi extension
%
% spectrum:          pixel spectrum (1 x bands)
%
% pixel:             pixel location [row col]
%
% base_name:         Base data file name w/o extension
%
% size:              Size of hypercube [bands rows columns]
%                    (default [52 800 1024])
%
% precision:         Data precision: 'uint16' or 'double'
%                    (default 'double')
%
% format:            Data format: 'BIL' or 'BIP'
%                    (default 'BIL')
%
%
% Author:            Joe Meola
% Last Update:       5 January 2006
%
% Default data size
if (nargin < 3)
    size=[124 800 1024];
end

% Data size
rows=size(2);
columns=size(3);
bands=size(1);

% Default of 'double' precision
if (nargin < 5)
    precision= 'double';
end

if (strcmp(precision, 'uint16'))
    bytes=2;
else

```

```

        bytes=8;
    end

    % Default of 'BIL' format
    if (nargin < 4)
        format='BIL';
    end

    % Position of desired spectrum within image
    row=pixel(1);
    column=pixel(2);

    file_name=strcat(base_name, '.envi');

    % BIL Format
    if (strcmp('BIL', format))

        % Offset to desired column
        offset=(column-1)*(rows)*bytes*bands;
        fid=fopen(file_name);
        status=fseek(fid,offset,'bof');

        % Read in all rows and bands for desired column
        A=fread(fid,[rows bands],precision);
        fclose(fid);
        spectrum=squeeze(A(row,:));

    elseif (strcmp('BIP', format))

        col_offset=(column-1)*(rows)*bytes*bands;
        row_offset=(row-1)*bytes*bands;
        offset=col_offset + row_offset;
        fid=fopen(file_name);
        status=fseek(fid,offset,'bof');

        % Read in spectrum
        spectrum=fread(fid,[1 bands],precision);
        fclose(fid);

    else
        error(strcat(format, ' Invalid Data Format'));
    end
end

```

## scatter\_mask

```

function scatter_mask(mask, B1, B2);

% function scatter_mask(mask, B1, B2);
%
% This function allows a user to input a mask cube to select varioius
% regions of interest. These regions are scatter plotted using different
% colors.

% mask:                Cube denoting region masks (row x col x N)
%
% B1:                  Band 1
%
% B2:                  Band 2
%
%
% Author:              Joe Meola
% Last Update:         28 February 2006

% Get size and number of regions
[rows columns N]=size(mask);

% Check if number of regions allowed
if (N > 7)

```

```

        error('N cannot be greater than 7');
    end

    b1=reshape(B1, 1, rows*columns);
    b2=reshape(B2, 1, rows*columns);

    % Markers for display purposes
    marker=['.r'; '.b'; '.g'; '.k'; '.c'; '.m'; '.y'];

    % Collect N ROIs
    for n=1:N
        bw(n,:)=reshape(mask(:,:,n), 1, rows*columns);
    end

    % Display Scatter Plots
    figure;
    for n=1:N

        idx=find( bw(n,:) == 1);
        X=b1(idx);
        Y=b2(idx);
        h=plot(X, Y, marker(n,:), 'MarkerSize', 2);
        hold on;

    end
    hold off;

    xlabel('Band 1');
    ylabel('Band 2');

```

## B.4 Detection Code

This code employs various detection algorithms such as SMF, anomaly detection, and change detection.

### fanomaly

```

function Ad=fanomaly(base_name, mn, C, size, precision, format)
%
% function Ad=fanomaly(base_name, mn, C, size, precision, format);
%
% This function performs a simple anomaly detection on a data set using
% the M-distance measure and background statistics (C, mn).
% Note: Data file must have .envi extension
%
% Ad: Detection Statistic Image (rows x columns)
%
% base_name: Base data file name w/o extension
%
% C: Covariance Matrix (bands x bands)
%
% mn: Spectral Mean (1 x bands)
%
% size: Size of hypercube [bands rows columns]
%       (default [124 800 1024])
%
% precision: Data precision: 'uint16' or 'double'
%             (default 'double')
%
% format: Data format: 'BIL' or 'BIP'
%           (default 'BIL')
%
%

```

```

% Author:      Joe Meola
% Last Update: 19 January 2006

% Default Data size
if(nargin < 4)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 5)
    precision='double';
end

% Default Data format
if(nargin < 6)
    format='BIL';
end

if(nargin < 3)
    [C, mn]=fcov(base_name, size, format, precision);
end

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% Calculate inverse covariance matrix
Cinv=inv(C);

Ad=zeros(rows, columns);
file_name=strcat(base_name, '.envi');
fid_read=fopen(file_name);

% BIL Format
if (strcmp(format, 'BIL'))
    for n=1:columns

        % Read in one column at a time in all bands
        A=fread(fid_read, [rows bands], precision);

        % Cycle through every hyperpixel
        for m=1:rows
            Ad(m,n)=( A(m,:) - mn )*Cinv*( A(m,:) - mn');
        end

    end

% BIP Format
elseif(strcmp(format, 'BIP'))

    for n=1:columns

        % Read in one column at a time in all bands
        A=fread(fid_read, [bands rows], precision);

        % Cycle through every hyperpixel
        for m=1:rows

            Ad(m,n)=( A(:,m) - mn)*Cinv*( A(:,m) - mn' );

        end

    end

else
    error(strcat(format, ' Invalid Data Format'));
end

fclose(fid_read);

```

## fsmf

```
function Ad=fsmf(base_name, T, mn, C, size, precision, format)
%
%   function Ad=fsmf(base_name, T, mn, C, size, precision, format);
%
%   This function performs a spectral matched filter on a data set using
%   a target spectrum (T) and background statistics (C, mn).
%   Note: Data file must have .envi extension
%
%   Ad:                               Detection Statistic Image (rows x columns)
%
%   base_name:                         Base data file name w/o extension
%
%   T:                                Target spectrum (1 x bands)
%
%   C:                                Covariance Matrix (bands x bands)
%
%   mn:                               Spectral Mean (1 x bands)
%
%   size:                             Size of hypercube [bands rows columns]
%                                   (default [124 800 1024])
%
%   precision:                        Data precision: 'uint16' or 'double'
%                                   (default 'double')
%
%   format:                           Data format: 'BIL' or 'BIP'
%                                   (default 'BIL')
%
%
%   Author:                           Joe Meola
%   Last Update:                       5 January 2006

%   Default Data size
if(nargin < 5)
    size=[124 800 1024];
end

%   Default Data precision
if(nargin < 6)
    precision='double';
end

%   Default Data format
if(nargin < 7)
    format='BIL';
end

%   Data size
bands=size(1);
rows=size(2);
columns=size(3);

%   Calculate inverse covariance matrix
Cinv=inv(C);

%   Create Mean matrix to subtract from data
mn_matrix= repmat(mn, rows,1);

%   Open File for reading
file_name=strcat(base_name, '.envi');
fid_read=fopen(file_name);

%   BIL format
if(strcmp(format, 'BIL'));

    %   Matched Filter Algorithm
```

```

for n=1:columns

    % Read in one column at a time in all bands
    xm=fread(fid_read, [rows, bands], precision);

    % Take tranpose for correct orientation of image
    Ad(:,n)=(T*Cinv*(xm-mn_matrix)')';

end

% BIP format
elseif(strcmp(format, 'BIP'))

    mn_matrix=mn_matrix';

    % Matched Filter Algorithm
    for n=1:columns

        % Read in one column at a time in all bands
        xm=fread(fid_read, [bands rows], precision);

        % Take tranpose for correct orientation of image
        Ad(:,n)=(T*Cinv*(xm-mn_matrix)')';

    end
else
    error(strcat(format, ' Invalid Data Format'));
end

fclose(fid_read);

```

## fchange

```

function [Ad, C_error]=fchange(base_name1, base_name2, method, size, format, precision,
MASK);
%
% function [Ad, C_error]=fchange(base_name1, base_name2, method, size, format,
precision, MASK);
%
% This function performs change detection on two data sets. The linear
% estimation methods used are Covariance Equalization 'CE', or
% Chronochrome, 'CC'. A detection statistic image is returned.
%
% Ad: Detection Statistic [rows columns]
%
% C_error: Error Covariance Matrix [bands bands]
%
% base_name1: Base name of time-1 data
%
% base_name2: Base name of time-2 data
%
% method: Either 'CE' or 'CC'
%
% size: Size of hypercube [bands rows columns]
% (default [124 800 1024])
%
% precision: Data precision: 'uint16' or 'double'
% (default 'double')
%
% format: Data format: 'BIL' or 'BIP'
% (default 'BIL')
%
% MASK: Morphological mask denoting area to use for
% transform calculation
%
% Author: Joe Meola
% Last Update: 18 May 2006

```

```

% Default Data size
if(nargin < 4)
    size=[124 800 1024];
end

% Default Data format
if(nargin < 5)
    format='BIL';
end

% Default Data precision
if(nargin < 6)
    precision='double';
end

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% Default MASK
if(nargin < 7)
    MASK='whole';
end

% Determine mean and covariance matrices for data sets
[Cx, mnx]=fcov(base_name1, size, format, precision, MASK);
[Cy, mny]=fcov(base_name2, size, format, precision, MASK);

% Set up Detection Statistic
Ad=zeros(rows, columns);

% Covariance Equalization Method
if( strcmp(method, 'CE'))

    % Perform CE estimation
    fce(base_name1, base_name2, mnx, Cx, mny, Cy, size, format, precision);

    % Determine Error Covariance matrix
    base_name_error=strcat(base_name2, '_CEerror');
    [C_error, mn_error]=fcov(base_name_error, size, format, precision, MASK);

% Chronochrome method
elseif( strcmp(method, 'CC'))

    % Determine cross-covariance matrix
    Cxy=fcross_cov(base_name1, base_name2, mnx, mny, size, format, precision, MASK);

    % Perform CC estimation
    fcc(base_name1, base_name2, mnx, Cx, mny, Cxy, size, format, precision);

    % Determine Error Covariance matrix
    base_name_error=strcat(base_name2, '_CCerror');
    [C_error, mn_error]=fcov(base_name_error, size, format, precision, MASK);

else
    error(strcat(method, ' Invalid Method'));
end

% Perform Change Detection
Cinv=inv(C_error);
file_name_error=strcat(base_name_error, '.envi');

fid_error=fopen(file_name_error);

% BIL Format
if (strcmp(format, 'BIL'))
    for n=1:columns

        % Read in one column at a time in all bands

```

```

        e=fread(fid_error, [rows bands], precision);

        % Cycle through every hyperpixel
        for m=1:rows

            Ad(m,n)= (e(m,:)- mn_error)*Cinv*(e(m,:) - mn_error)';

        end

    end

    % BIP Format
    elseif(strcmp(format, 'BIP'))

        for n=1:columns

            % Read in one column at a time in all bands
            e=fread(fid_error, [bands rows], precision);

            % Cycle through every hyperpixel
            for m=1:rows

                Ad(m,n)=(e(:,m)' - mn_error)*Cinv*(e(:,m) - mn_error)';

            end

        end

    else
        error(strcat(format, ' Invalid Data Format'));
    end

    fclose(fid_error);

```

## B.5 Statistics Code

This code is used to calculate specific statistical values from data sets. These statistical values include spectral means, covariance matrices, ndvi data, and shade spectra.

### fcov

```

function [C, mn]=fcov(base_name, size, format, precision, MASK)
%
% function [C, mn]=fcov(base_name, MASK, size, precision, format);
%
% This function calculates the Covariance Matrix for a hyperspectral
% cube. The spectral mean is first calculated using fspec_mean or fsmean.
% Note: Data file must have .envi extension
%
% C:                               Covariance Matrix (bands x bands)
%
% mn:                               Spectral Mean (1 x bands)
%
% base_name:                        Base data file name w/o extension
%
% MASK:                             Morphological mask indicating region of interest
%                                   (default is entire image)
%
% size:                             Size of hypercube [bands rows columns]
%                                   (default [124 800 1024])
%
% precision:                        Data precision: 'uint16' or 'double'
%                                   (default 'double')

```



```

%
% format:          Data format: 'BIL' or 'BIP'
%                  (default 'BIL')
%
%
% Author:          Joe Meola
% Last Update:     11 May 2006

% Default Data size
if(nargin < 2)
    size=[124 800 1024];
end

% Default Data format
if(nargin < 3)
    format='BIL';
end

% Default Data precision
if(nargin < 4)
    precision='double';
end

% Default MASK
if(nargin < 5)
    MASK='whole';
end

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% If MASK not supplied, use whole image
if(ischar(MASK))

    if(strcmp(MASK, 'whole'))

        % Open file for reading
        file_name=strcat(base_name, '.envi');

        % Calculate mean vector
        mn=fspec_mean(base_name, size, format, precision);
        mn_mat=repmat(mn,rows,1);

        % Set up covariance matrix
        C=zeros(bands, bands);

        fid_read=fopen(file_name);

        % BIL format
        if(strcmp(format, 'BIL'))

            % Cycle through file reading in column of hyperpixels at a time
            for n=1:columns

                % Read in [rows bands]
                A=fread(fid_read, [rows bands], precision);

                % Running covariance calc
                C=C + ((A-mn_mat)'*(A-mn_mat))/(rows*columns);

            end

            % BIP format
        elseif(strcmp(format, 'BIP'))

            mn_mat=mn_mat';
        end
    end
end

```

```

% Cycle through file reading in column of hyperpixels at a time
for n=1:columns

    % Read in [bands rows]
    A=fread(fid_read, [bands rows], precision);

    % Running covariance calc
    C=C + ((A-mn_mat)*(A-mn_mat)')/(rows*columns);

end

% Bad data format provided otherwise
else
    error(strcat(format, ' Invalid Data Format'));
end
fclose(fid_read);

else
    error('"whole" is only valid string input');
end

% If MASK supplied, use only pixels of interest
else

    % Open file for reading
    file_name=strcat(base_name, '.envi');

    % Determine number of pixels of interest
    [R cm]=find(MASK == 1);
    N=length(R);

    % Calculate mean vector
    mn=fsmean(base_name, MASK, size, precision, format);

    % Set up covariance matrix
    C=zeros(bands, bands);
    fid_read=fopen(file_name);

    % BIL format
    if(strcmp(format, 'BIL'));

        % Cycle through file reading in column of hyperpixels at a time
        for n=1:columns

            % Read in [rows bands]
            A=fread(fid_read, [rows bands], precision);

            % Cycle through all hyperpixels
            for m=1:rows

                % Only use pixels of interest
                if(MASK(m,n)==1)

                    % Running cov calc
                    C=C + ((A(m,:)-mn)*(A(m,:)-mn)')/N;

                end

            end

        end

    end

    % BIP format
elseif(strcmp(format, 'BIP'))

    % Cycle through file reading in column of hyperpixels at a time
    for n=1:columns

        % Read in [bands rows]
        A=fread(fid_read, [bands rows], precision);

```

```

        % Cycle through all hyperpixels
        for m=1:rows

            % Only use pixels of interest
            if(MASK(m,n)~=1)
                % Running cov calc
                C=C + ((A(:,m)-mn')*(A(:,m)-mn'))/N;
            end
        end

        % Bad data format provided otherwise
        else
            error(strcat(format, ' Invalid Data Format'));
        end
        fclose(fid_read);
    end
end

```

### **fcross\_cov**

```

function Cxy=fcross_cov(base_name1, base_name2, mnx, mny, size, format, precision, MASK)
%
% function function Cxy=fcov(base_name1, base_name2, mnx, mny, size, format,
precision);
%
% This function calculates the cross-covariance matrix for two hyperspectral
% data cubes.
% Note: Data file must have .envi extension
%
% Cxy: Covariance Matrix (bands x bands)
%
% base_name_x: Base data file name for 1
%
% base_name_y: Base data file name for 2
%
% mnx: Spectral Mean of 1 (1 x bands)
%
% mny: Spectral Mean of 2 (1 x bands)
%
% size: Size of hypercube [bands rows columns]
%       (default [124 800 1024])
%
% precision: Data precision: 'uint16' or 'double'
%             (default 'double')
%
% format: Data format: 'BIL' or 'BIP'
%          (default 'BIL')
%
% Author: Joe Meola
% Last Update: 5 January 2006

% Default Data size
if(nargin < 5)
    size=[124 800 1024];
end

% Default Data format
if(nargin < 6)
    format='BIL';
end

% Default Data precision
if(nargin < 7)
    precision='double';
end

```

```

% Default MASK
if(nargin < 8)
    MASK='whole';
end

file_name1=strcat(base_name1, '.envi');
file_name2=strcat(base_name2, '.envi');

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% If MASK not supplied, use whole image
if(strcmp(MASK, 'whole'))

    % Create mean matrices
    mnx_mat= repmat(mnx, rows, 1);
    mny_mat= repmat(mny, rows, 1);

    % Setup cross covariance matrix
    Cxy=zeros(bands, bands);

    % Open files
    fid_read1=fopen(file_name1);
    fid_read2=fopen(file_name2);

    % BIL format
    if(strcmp(format, 'BIL'))

        % Cycle through file reading in column of hyperpixels at a time
        for n=1:columns
            x=fread(fid_read1, [rows bands], precision);
            y=fread(fid_read2, [rows bands], precision);

            % Running cross-cov calc
            Cxy=Cxy + ((y-mny_mat)'*(x-mnx_mat))/(rows*columns);
        end

        % BIP format
    elseif(strcmp(format, 'BIP'))

        % Transpose mean matrices for BIP
        mnx_mat=mnx_mat';
        mny_mat=mny_mat';

        % Cycle through file reading in column of hyperpixels at a time
        for n=1:columns
            x=fread(fid_read1, [bands rows], precision);
            y=fread(fid_read2, [bands rows], precision);

            % Running cross-cov calc
            Cxy=Cxy + ((x-mnx_mat)*(y-mny_mat'))/(rows*columns);
        end

        % Otherwise bad data format
    else
        error(strcat(format, ' Invalid Data Format'));
    end

    % Close files
    fclose(fid_read1);
    fclose(fid_read2);

% If MASK supplied, use only pixels of interest
else

    % Determine number of pixels of interest
    [R cm]=find(MASK == 1);
    N=length(R);

```

```

% Set up cross covariance matrix
Cxy=zeros(bands, bands);

% Open files
fid_read1=fopen(file_name1);
fid_read2=fopen(file_name2);

% BIL format
if(strcmp(format, 'BIL'));

    % Cycle through file reading in column of hyperpixels at a time
    for n=1:columns
        x=fread(fid_read1, [rows bands], precision);
        y=fread(fid_read2, [rows bands], precision);

        % Cycle through hyperpixels
        for m=1:rows
            % Use only pixels of interest
            if(MASK(m,n)==1)

                % Running cross-cov calc
                Cxy=Cxy + ((y(m,:)-mny)'*(x(m,:)-mnx))/N;
            end
        end
    end

    % BIP format
elseif(strcmp(format, 'BIP'))

    % Cycle through file reading in column of hyperpixels at a time
    for n=1:columns
        x=fread(fid_read1, [bands rows], precision);
        y=fread(fid_read2, [bands rows], precision);
        % Cycle through hyperpixels
        for m=1:rows
            % Use only pixels of interest
            if(MASK(m,n)==1)
                % Running mean
                Cxy=Cxy + ((x(:,m)-mnx)*(y(:,m)-mny'))/N;
            end
        end
    end

    % Otherwise bad data format
else
    error(strcat(format, ' Invalid Data Format'));
end

% Close files
fclose(fid_read1);
fclose(fid_read2);
end

```

## fsmean

```

function mn=fsmean(base_name, MASK, size, precision, format)

%
% function mn=fsmean(base_name, MASK, size, precision, format);
%
% This function will calculate the spectral mean for a region of interest
% for a hyperspectral data cube. The region of interest is either
% specified via a binary "MASK" matrix or the user inputs 'roi' for "MASK"
% to select the region interactively. The program determines the
% spectral mean via two different methods, depending on the total number
% of pixels of interest.
%
% mn: Spectral Mean (1 x bands)
%

```

```

% base_name:          Base data file name w/o extension
%
% MASK:              Region of Interest (rows x columns) or ('roi')
%
% size:              Size of hypercube [bands rows columns]
%                   (default [124 800 1024])
%
% precision:         Data precision: 'uint16' or 'double'
%                   (default 'double')
%
% format:            Data format: 'BIL' or 'BIP'
%                   (default 'BIL')
%
%
% Author:            Joe Meola
% Last Update:       02 March 2006

% Default Data size
if(nargin < 3)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 4)
    precision='double';
end

% Default Data format
if(nargin < 5)
    format='BIL';
end

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% If MASK input is string, allow interactive ROI
if(ischar(MASK);
    band=ceil(bands/2);
    figure;
    fimage(band, base_name, size, format, precision);
    MASK = roipoly;
end

% Find pixels of interest
[r c]=find(MASK == 1);
N=length(r);

% If number of pixels of interest small enough, faster to determine
% mean by looking at individual pixels
if(N < 15500)
    % Set up mean vector
    mn=zeros(1, bands);

    % Calculate mean vector
    for n=1:length(r);
        spectrum=fspectrum(base_name, [r(n) c(n)], size, format, precision);
        mn= mn + spectrum/N;
    end
else
    file_name=strcat(base_name, '.envi');
    fid_read=fopen(file_name);

    % BIL format
    if(strcmp(format, 'BIL'));
        % Set up mean vector
        mn=zeros(1, bands);
    end
end

```

```

    % Cycle through file reading in (rows) pixels at a time
    for n=1:columns
        % Read in [rows bands] and sum rows for averaging
        A=fread(fid_read, [rows bands], precision);

        % Running mean
        mn=(MASK(:,n)*A)/N + mn;
    end

    % BIP format
    elseif(strcmp(format, 'BIP'))
        % Set up mean vector
        mn=zeros(bands, 1);

        % Cycle through file reading in (rows) pixels at a time
        for n=1:columns
            % Read in [rows bands] and sum rows for averaging
            A=fread(fid_read, [bands rows], precision);

            % Running mean
            mn=(A*MASK(:,n))/N + mn;
        end

        mn=mn';
    else
        error(strcat(format, ' Invalid Data Format'));
    end
    fclose(fid_read);
end

```

## **fndvi**

```

function [ndvi, MASK]=fndvi(base_name, wavelengths, P, size, format, precision);
%
% function ndvi=fndvi(base_name, wavelengths, size, format, precision);
%
% This function creates a normalized difference vegetation index image
% using wavelengths of 660nm and 860nm as the red and IR bands. If
% wavelengths are not supplied, default values are loaded.
%
% ndvi: NDVI image (rows x columns)
%
% MASK: Morphological mask denoting brightest
% vegetation pixels based on parameters
%
% base_name: Base data file name w/o extension
%
% wavelengths: Wavelengths (nm) associated with data (1 x bands)
%
% P: Threshold for MASK (percent of pixels)
%
% size: Size of hypercube [bands rows columns]
% (default [124 800 1024])
%
% precision: Data precision: 'uint16' or 'double'
% (default 'double')
%
% format: Data format: 'BIL' or 'BIP'
% (default 'BIL')
%
% Author: Joe Meola
% Last Update: 4 April 2006
%
% Default Wavelengths
if(nargin < 2)
    load wavelengths;
end

```

```

end

% Default Threshold
if(nargin < 3)
    P=0.005;
end

% Default Data size
if(nargin < 4)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 6)
    precision='double';
end

% Default Data format
if(nargin < 5)
    format='BIL';
end

% Data size
bands=size(1);
rows=size(2);
columns=size(3);

% Number of pixels to use for veg
N=ceil(P*rows*columns);

% Bands to use for red and IR
Bred=660;
Bir=860;

% Determine indices for pertinent bands
red_idx=find(wavelengths < Bred);
ir_idx=find(wavelengths < Bir);

% Interpolate to determine actual wavelength images
Ared1=fband(red_idx(end), base_name, size, format, precision);
Ared2=fband(red_idx(end)+1, base_name, size, format, precision);

Air1=fband(ir_idx(end), base_name, size, format, precision);
Air2=fband(ir_idx(end)+1, base_name, size, format, precision);

r=permute(cat(3, Ared1, Ared2), [3 1 2]);
ir=permute(cat(3, Air1, Air2), [3 1 2]);

Ared=squeeze(interp1(wavelengths(red_idx(end):red_idx(end)+1), r, Bred));
Air=squeeze(interp1(wavelengths(ir_idx(end):ir_idx(end)+1), ir, Bir));

% Compute NDVI image
ndvi= (Air - Ared)./(Air + Ared);
mx=max(max(ndvi));
mn=min(min(ndvi));
ndvi=(ndvi - mn)/(mx - mn);

% Determine mask based on bright veg in IR
LIR=reshape(Air, 1, rows*columns);
NDVI=reshape(ndvi, 1, rows*columns);
Bmask=zeros(1, rows*columns);

IDX=find(NDVI > 0.92);

NIR=LIR(IDX);
[S IX]=sort(NIR, 'descend');

idx_t=IDX(IX(1:N));
Bmask(idx_t)=1;

```



```
MASK=reshape(Bmask, rows, columns);
```

## fshade

```
function shade=fshade(base_name, method, size, format, precision);
%
% function shade=fshade(base_name, method, size, format, precision);
%
% This function determines a shade point for a hyperspectral data cube.
% The function searches for the minimum pixel value in each spectral
% band if the 'min' method is used or determines the lowest average spectral
% band if the 'mean' method is used.
%
% shade:                Shade spectrum (1 x bands)
%
% base_name:            Base data file name w/o extension
%
% method:               'min' for minimum pixel or 'mean' for minimum
%                       average band
%
% size:                 Size of hypercube [bands rows columns]
%                       (default [124 800 1024])
%
% precision:            Data precision: 'uint16' or 'double'
%                       (default 'double')
%
% format:               Data format: 'BIL' or 'BIP'
%                       (default 'BIL')
%
%
% Author:               Joe Meola
% Last Update:          7 April 2006

% Default Data size
if(nargin < 3)
    size=[124 800 1024];
end

% Default Data precision
if(nargin < 5)
    precision='double';
end

% Default Data format
if(nargin < 4)
    format='BIL';
end

% Data size
rows=size(2);
columns=size(3);
bands=size(1);

% Set up files
file_name=strcat(base_name, '.envi');
minimum=zeros(columns, bands);

%Reading image in BIL format
if(strcmp(format, 'BIL'))

    % Find minimum pixel in each band
    if(strcmp(method, 'min'))
        fid_read=fopen(file_name);
        % Cycle through file reading in (rows) pixels at a time
        for n=1:columns
            % Read in [rows bands] and determine minimum values in column
            A=fread(fid_read, [rows bands], precision);
            minimum(n, :)=min(A);
        end
    end
end
```

```

        end
        fclose(fid_read);

        % Determine minimum for all columns
        shade=min(minimum);

    % Find lowest average band
    elseif(strcmp(method, 'mean'))
        frame=zeros(rows, columns);
        fid_read=fopen(file_name);
        % Cycle through file reading in (rows) pixels at a time
        for n=1:columns
            % Read in [rows bands] and determine minimum values in column
            A=fread(fid_read, [rows bands], precision);
            frame(:, n)=mean(A,2);
        end
        fclose(fid_read);

        % Determine lowest mean pixel
        [r_idx c_idx]=find( frame==(min(min(frame))));
        shade=fspectrum(base_name, [r_idx c_idx], size, format, precision);
    end

%Reading image in BIP format
elseif(strcmp('BIP',format))
    if(strcmp(method, 'min'))
        fid_read=fopen(file_name);

        % Cycle through file reading in (rows) pixels at a time
        for n=1:columns
            % Read in [rows bands] and determine minimum values in column
            A=fread(fid_read, [bands rows], precision);
            minimum(n, :)=min(A, [], 2);
        end

        fclose(fid_read);
        % Determine minimum for all columns
        shade=min(minimum, [], 2);

        % Find lowest average band
    elseif(strcmp(method, 'mean'))
        frame=zeros(rows, columns);
        fid_read=fopen(file_name);

        % Cycle through file reading in (rows) pixels at a time
        for n=1:columns
            % Read in [rows bands] and determine minimum values in column
            A=fread(fid_read, [bands rows], precision);
            frame(:, n)=mean(A,1)';
        end

        fclose(fid_read);
        % Determine lowest mean pixel
        [r_idx c_idx]=find( frame==(min(min(frame))));
        shade=fspectrum(base_name, [r_idx c_idx], size, format, precision);
    end
else
    error(strcat(format, ' Invalid Data Format'));
end

```

## REFERENCES

- 1) Shaw, Gary, and Dimitris Manolakis. "Signal Processing for Hyperspectral Image Exploitation." *IEEE Signal Processing Magazine* January 2002: 12-16.
- 2) Headwall Photonics. *Hyperspec VS Family of Imaging Spectrographs*. [www.headwallphotonics.com](http://www.headwallphotonics.com) .
- 3) DALSA Digital Imaging. *Pantera TF 1M60 Area Scan Cameras*. [www.dalsa.com](http://www.dalsa.com) .
- 4) Aerotech. *ADRS Series Mechanical Bearing Rotary Stages*. [www.aerotech.com](http://www.aerotech.com) .
- 5) Analytical Spectral Devices, Inc. *FieldSpec Pro Product Specifications*. [www.asdi.com](http://www.asdi.com) .
- 6) Simi, Christopher, et al. "Night Vision Imaging Spectrometer (NVIS) Calibration and Configuration: Recent Developments." *SPIE* Vol. 4381 (2001): 109:115.
- 7) Newport. *HeNe Lasers*. [www.newport.com](http://www.newport.com) .
- 8) Davis, Curtis, et al. "Calibration, Characterization and first Results with the Ocean PHILLS Hyperspectral Imager." *SPIE* Vol. 3753 (1999): 160:168.
- 9) Newport. *Oriel Pencil Style Calibration Lamps*. [www.newport.com](http://www.newport.com) .
- 10) D'Agostino, John A., and Curtis Webb. "3-D Analysis and Measurement Methodology for Imaging System Noise." *SPIE*. Vol. 1488 (1991): 110-117.
- 11) Holst, Gerald. *CCD Arrays Cameras and Displays*. 2<sup>nd</sup> ed. Winter Park, FL: JCD Publishing, 1998.
- 12) Ziemer, Rodger E. *Elements of Engineering Probability and Statistics*. Upper Saddle River, NJ: Prentice-Hall, 1997.
- 13) Mitra, Sanjit K. *Digital Signal Processing: A Computer Based Approach*. 3<sup>rd</sup> ed. New York, NY: McGraw-Hill, 2006.

- 14) Leon-Garcia, Alberto. *Probability and Random Processes for Electrical Engineering*. 2<sup>nd</sup> ed. Reading, MA: Addison Wesley, 1994.
- 15) Labsphere. "A Guide to Integrating Sphere Radiometry and Photometry." *Techguide*. [www.labsphere.com](http://www.labsphere.com).
- 16) Pedrotti, Frank L., and Leno S. Pedrotti. *Introduction to Optics*. 2<sup>nd</sup> ed. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- 17) Zadnik, Jerry, et al. "Calibration Procedures and Measurements for the COMPASS Hyperspectral Imager." *SPIE* Vol. 5425 (2004): 182:188.
- 18) Solar Position Calculator. *National Oceanic and Atmospheric Administration Surface Radiation Research Branch*.  
<http://www.srrb.noaa.gov/highlights/sunrise/azel.html>.
- 19) Sun Path Chart Program. *The University of Oregon Solar Radiation Monitoring Laboratory*. <http://solardat.uoregon.edu/SoftwareTools.html>.
- 20) Manolakis, Dimitris and Gary Shaw. "Detection Algorithms for Hyperspectral Imaging Applications." *IEEE Signal Processing Magazine* January 2002: 29-43.
- 21) Stein, David W., et al. "Anomaly Detection from Hyperspectral Imagery." *IEEE Signal Processing Magazine* January 2002: 58-69.
- 22) Gonzalez, Rafael C., et al. *Digital Image Processing using MATLAB*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.
- 23) Schaum, Al, and Alan Stocker. "Hyperspectral Change Detection and Supervised Matched Filtering Base on Covariance Equalization." *SPIE* Vol. 5425 (2004): 77-90.
- 24) Karpouzli, E., and T. Malthus. "The empirical line method for the atmospheric correction of IKONOS imagery." *International Journal of Remote Sensing*, Vol. 24, No. 5 (2003): 1143-1150.
- 25) Schowengerdt, Robert. *Remote Sensing: Models and Methods for Image Processing*. 2<sup>nd</sup> ed. San Diego, CA: Academic Press, 1997.
- 26) Xu, J., and J. Huang. "Refined empirical line method to calibrate IKONOS imagery." *Journal of Zhejiang University SCIENCE A*, Vol. 7, No.4 (2006): 641-646.
- 27) *ASTER Spectral Library*. NASA Jet Propulsion Laboratory. April 18, 2006  
<<http://speclib.jpl.nasa.gov/>>.

ROO2S92715

- 28) Richards, John A. *Remote Sensing Digital Image Analysis: An Introduction*. 2<sup>nd</sup> ed. New York, NY: Springer-Verlag, 1993.
- 29) Stark, Henry, and John W. Woods. *Probability and Random Processes with Applications to Signal Processing*. 3<sup>rd</sup> ed. Upper Saddle River, NJ: Prentice-Hall, 2002.