EARLY FOREST FIRE DETECTION USING TEXTURE ANALYSIS OF PRINCIPAL COMPONENTS FROM MULTISPECTRAL VIDEO

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical Engineering

by
Timothy M. Davenport
June 2012
COMMITTEE MEMBERSHIP

TITLE: Early Forest Fire Detection using Texture Analysis of Principal Components from Multispectral Video

AUTHOR: Timothy M. Davenport

DATE SUBMITTED: June 2012

COMMITTEE CHAIR: Dr. John Saghri, Associate Professor

COMMITTEE MEMBER: Dr. John Jacobs, Raytheon Professor of Practice

COMMITTEE MEMBER: Dr. Jane Zhang, Associate Professor
ABSTRACT

EARLY FOREST FIRE DETECTION USING TEXTURE ANALYSIS OF PRINCIPAL COMPONENTS FROM MULTISPECTRAL VIDEO

Timothy Davenport

The aim of this study is to incorporate the spectral, temporal and spatial attributes of a smoke plume for Early Forest Fire Detection. Image processing techniques are used on multispectral (red, green, blue, mid-wave infrared, and long-wave infrared) video to segment and identify the presence of a smoke plume within a scene.

The temporal and spectral variance of a smoke plume is captured through Principal Component Analysis (PCA) where the Multispectral-Multitemporal PCA is performed on a sequence of video frames simultaneously. The presence of a plume existing in one of the higher order principal components is determined by the texture of its spatial content. The texture is characterized by statistical descriptors derived from the principal component’s joint probability density distribution of intensities occurring within a spatial relationship, known as a Gray Level Co-Occurrence Matrix (GLCM). Initial analysis is performed on selected frames where only a subset of time is considered. Once the parameters are chosen from the static analysis, the algorithms are executed on video through time to validate the method.

The results show that a smoke plume is readily segmented via PCA. Based on the five spectral bands over 3 seconds sampled at 1 second, the plume exists in the 7th principal component. Within these principal components, the smoke’s presence is best identified by the correlation texture descriptor. The smoke is very spatially correlated compared to the scene at large. Therefore a spike in the spatial correlation of the principal components is all that is needed to identify the start of the smoke plume.

Keywords: Image Processing, Video, Principal Component Analysis, PCA, Texture Analysis, Gray Level Co-Occurrence Matrix, GLCM, Multispectral, Visible, Infrared
ACKNOWLEDGMENTS

I would like to thank Dr. Saghri & Dr. Jacobs for their advice, support and encouragement throughout the whole project. They helped to shape the way I think about analyzing data, this is something that I will carry with me through the rest of my engineering career. I would also like to thank Raytheon Employees Steve Botts, Marc Bauer, Chris Tracy, Dana Schneider for their support during the June 2011 Fire Test and Gary Hughes from FLIR for his help with IR cameras. This work would not have been completed without the major support and encouragement from my family and friends.
# TABLE OF CONTENTS

LIST OF TABLES ......................................................................................................................... viii
LIST OF FIGURES ........................................................................................................................ ix

CHAPTER 1 Introduction ................................................................................................................. 1
Early Forest Fires Detection and Current Systems ................................................................. 1
Principal Component Analysis (PCA) ..................................................................................... 1
Texture Analysis .................................................................................................................... 2
Proposed Early Forest Fire Detection ................................................................................... 2

CHAPTER 2 Theory and Algorithms ............................................................................................. 3
Principal Component Analysis (PCA) ..................................................................................... 3
PCA Approach for Image Processing ................................................................................... 3
Principal Component Transformation (PCT) ........................................................................ 6
PCT Example .......................................................................................................................... 7
Texture Analysis .................................................................................................................... 8
Gray Level Co-Occurrence Matrix (GLCM) ......................................................................... 8
Texture Descriptors ............................................................................................................... 9
GLCM and Descriptor Example ............................................................................................ 10

CHAPTER 3 Data Acquisition ....................................................................................................... 11
Fire Test ...................................................................................................................................... 11
Data Format .......................................................................................................................... 12

CHAPTER 4 Pre-Preprocessing .................................................................................................... 13
Image Registration ................................................................................................................ 13
Keyframes ............................................................................................................................. 17

CHAPTER 5 Multispectral Principal Component Analysis .......................................................... 18
Original Raw Data ............................................................................................................... 18
Z-score of the Raw Data ......................................................................................................... 22
Nonlinearity of Infrared .......................................................................................................... 25

CHAPTER 6 Multi-Spectral-Temporal Principal Component Analysis .......................................... 32
PCA at two time instances ...................................................................................................... 33
Effect of noise on plume segmentation ............................................................................... 37
Effect of sample time ........................................................................................................... 38

CHAPTER 7 Texture Analysis in Principal Component Space .................................................... 40
Gray Level Co-Occurrence Matrix..............................................................................................................40
GLCM Texture Descriptors evaluated through time. ..................................................................................44
CHAPTER 8 Conclusion ...............................................................................................................................46
Bibliography ..............................................................................................................................................48
Appendix A: MATLAB Scripts and Functions ..........................................................................................51
LIST OF TABLES

Table 1: Quantized Pixel Angles ................................................................. 8
Table 2: Example GLCM with texture descriptors ........................................ 10
Table 3: Camera Specifications .................................................................... 11
Table 4: Raw Frame Statistics ..................................................................... 18
Table 5: Principal Component of Raw Frame Statistics ................................. 21
Table 6: Z-score Frame Statistics .................................................................. 23
Table 7: Principal Component Z-score Statistics ........................................... 24
Table 8: Z-score statistic after a nonlinear mapping with $y = 0.51$ ............. 29
Table 9: PC statistics for nonlinear Z-scores ................................................. 31
Table 10: Texture Descriptors for the GLCMs in Figure 33 .......................... 41
LIST OF FIGURES

Figure 1: (a) RGB feature space, (b) Principal Components of RGB spectra ........................................ 7
Figure 2: Data Acquisition Process ....................................................................................................... 12
Figure 3: Unregistered FOV .................................................................................................................. 14
Figure 4: Registered FOV ...................................................................................................................... 16
Figure 5: Spectral Frames (a) Red, (b) Green, (c) Blue, (d) MWIR, (e) LWIR .................................... 18
Figure 6: Histograms of original raw frames .......................................................................................... 18
Figure 7: 2-dimensional spectral feature scatter plots of raw frames ..................................................... 19
Figure 8: Contribution of each spectral band to the Principal Component loadings .............................. 21
Figure 9: Spectral Principal Components of Raw Frames (a) PC1, (b) PC2, (c) PC3, (d) PC4, (e) PC5 .... 21
Figure 10: Principal Component Histograms of raw images ................................................................ 22
Figure 11: Histogram of Raw Z-scores ................................................................................................... 22
Figure 12: Contribution of each spectral Z-score to the Principal Component loadings ...................... 24
Figure 13: Principal Components of the Z-scores (a) PC1, (b) PC2, (c) PC3, (d) PC4, (e) PC5 .............. 24
Figure 14: Principal Components of Z-score .......................................................................................... 24
Figure 15: Nonlinear Mapping from 14-bit to floating point ..................................................................... 25
Figure 16: Nonlinear Mapping for various $\gamma$ and their associated histograms ................................ 26
Figure 17: Magnitude of the PCT loadings ............................................................................................. 27
Figure 18: Mean and standard deviation of the Loadings for each Principal Component for various $\gamma$ ...................................................................................................................... 28
Figure 19: Z-scores after nonlinear mapping with $\gamma = 0.51$ ............................................................... 29
Figure 20: Principal Component Loadings from Z-scores nonlinearly mapped with $\gamma = 0.51$ ........ 30
Figure 21: Principal Components of the Z-scores (a) PC1, (b) PC2, (c) PC3, (d) PC4, (e) PC5 ............ 31
Figure 22: Principal Components of Z-scores nonlinearly mapped with $\gamma = 0.51$ ......................... 31
Figure 23: PCA Process ......................................................................................................................... 32
Figure 24: Spectral bands for the same two instance of time in Figure 25 ............................................ 33
Figure 25: Visible Image of scene for two instances in time. (a) Minimal smoke plume (b) Expanding smoke plume .................................................................................................................. 33
Figure 26: Principal Components of the frames of Figure 24, from left to right, top to bottom: PC 1 to 10 ................................................................................................................................................ 35
Figure 27: The 7th Principal Component of a 2 time instant temporal-spectral PCA segmenting the smoke plume (A) : Smoke Plume, (B) : Car, (C) : Truck ................................................................. 35
Figure 28: Magnitude of PCT loadings from each dimension of the spectral/temporal image set .......... 36
Figure 29: Visible Image of scene for two instances in time. (a) Weak smoke plume (b) Bus moves in frame ....................................................................................................................................... 37
Figure 30: Temporal-Spectral PCs for time instance shown in Figure 29 .............................................. 38
Figure 31: Appearance of Plume depending on sample time ................................................................. 38
Figure 32: PCs of 3 spectral frames sampled at 60 seconds ................................................................. 39
Figure 33: Construction of GLCM ........................................................................................................... 40
Figure 34: Selected Gray Level Co-Occurrence Matrixes, (a) PC, (b) [-20, -20], (c) [-2, 0], (d) [0,-2], (e) [6, 16] ............................................................................................................. 41
Figure 35: Texture Descriptors for principal components ................................................................. 43
Figure 36: Difference between Texture Descriptors of principal component with and without smoke plume ............................................................................................................................. 43
Figure 37: Four Texture Descriptors for the summation of 7th PC GLCMs (0, 45, 90, and 135) through time .......................................................................................................................... 44
Figure 38: GLCM Correlation for each spatial distance 0, 45, 90, and 135 ................................. 45
Figure 39: Early Forest Fire Detection .................................................................................................. 46
Figure 40: GLCM Correlation for 7th PC showing start, continuation and end of smoke plume. The images shown above and below the correlation plot are 7th PCs at the times indicated by the red lines. ............................................................................................................... 47
CHAPTER 1 Introduction

Early Forest Fires Detection and Current Systems

Human operated towers, visually scanning forest environments to detect forest fires, have been a historical, but costly solution. The need for an operator to continually observe a region from a tower poses many labor difficulties and hazards that can be eliminated by autonomous systems [17]. An alternative to the human operated tower is satellite based detection described in [2], [23] and [24]. While satellite imagery may be ideal for large scale damage assessment of fire aftermath, the spatial resolutions limited by their ground sampling distance and the temporal resolutions limited by their passing coverage over a given area are not sufficient solutions to early detection [4]. Recently many ground based Early Forest Fire Detection solutions have been proposed. The Croatian iForestFire provides a Web Information System that relays multiple vision systems to an operator that makes final decisions on suspicious regions autonomously generated by the camera systems at the monitoring site. The Visible cameras view 16 preset positions where detection takes 15 seconds to compute at each position. This provides a 4 minute minimum time between detections at any specific location [3]. A ground based multispectral approach is implemented by the FAR system in [1] that compares information in the visible spectrum to information in the Infrared to reduce false alarm rates.

Principal Component Analysis (PCA)

The work in [8] and [9] has shown the use of PCA to classify aerosols from multispectral satellite data. Using MODIS radiance information from seven bands
together with GOCART aerosol speciation, the Principal Components were found to be sensitive to specific types of aerosols. Change detection methods may also be accomplished via PCA as shown in [11], where temporal data is used to classify the presence of a change within a scene between two time instances. Temporal Segmentation of video is also performed in [18] through PCA which utilized 100 frames to classify scene transitions.

**Texture Analysis**

In [16] the Gray-Level Co-occurrence Matrix (GLCM) is used to detect the spatial textures of smoke from visible data after a background subtraction method is performed. Experiments presented in [21] show that wavelets may be used to decompose smoke texture from a scene and where GLCM is used to find the statistical characteristics of the smoke from the wavelet.

**Proposed Early Forest Fire Detection**

Based on information gathered from existing Early Forest Fire Detection methods, a ground based multispectral solution is proposed which will segment the smoke and heat plume through Temporal-Spectral PCA and classify the presence of a smoke plume through GLCM texture analysis. The smoke plume will refer to the smoke generated from the fire that has moved far enough away so the heat and thus the IR energy have dissipated. The heat plume will refer to the air or smoke that is still near the fire and still contains IR energy.
CHAPTER 2 Theory and Algorithms

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a special type of eigendecomposition that linearly combines sample data weighted by the eigenvectors of their covariance matrix. The covariance of course being a symmetric matrix with a unity diagonal is very useful because eigendecomposition of symmetric real matrixes will always be projected onto an orthogonal basis or coordinate systems. Therefore, this decomposition provides a method for transforming the original dimensions of the data to dimensions of orthogonal variance such that each dimension is completely uncorrelated with any other dimension. In the realm of image processing this transformation provides a way to both aggregate similarities to reduce redundant correlated information within a set of images or accentuate differences that do not correlate with the majority of the image set.

PCA Approach for Image Processing

Given a set of image frames, $I$, represented as an $R \times C \times F$ three-dimensional matrix with $R$ rows, $C$ columns and $F$ frames (be they temporal or spectral), each frame is vectorized into $F$ column vectors of $M \times 1$ dimensions, where $M = RC$ is the total amount of pixels within the frame.

$$I = \left[ \begin{array}{c} i_{11} \cdots i_{1C} \\ \vdots \ \vdots \\ i_{R1} \cdots i_{RC} \end{array} \right], \ldots, \left[ \begin{array}{c} i_{11F} \cdots i_{1CF} \\ \vdots \ \vdots \\ i_{R1F} \cdots i_{RCF} \end{array} \right]$$

$$vec(I) = \left[ \begin{array}{c} i_{11} \cdots i_{1F} \\ \vdots \ \vdots \\ i_{M1} \cdots i_{MF} \end{array} \right]$$
The matrix $X$ is created as a subset of $I$ with $M \times N$ dimensions, where $N < F$. Especially in the temporal case the $N$ subsets of $I$ need not be contiguous frames and preferably is selected from some $j^{th}$ frame instead. $X$ represents the data to be analyzed through PCA and is composed of $M$ observations of pixels by $N$ number of spectral or temporal dimensions.

$$X = \begin{bmatrix}
  i_{1n} & \cdots & i_{1N} \\
  \vdots & \ddots & \vdots \\
  i_{Mn} & \cdots & i_{MN}
\end{bmatrix}, \quad \text{where } n < n + 2j < n + Nj < F \quad (3)$$

The Principal Coefficient loadings (or coefficients) are sensitive to scale, so to ensure the analysis is pulled towards an overly influential dimension, $X$ is normalized to its standard scores or Z scores which provides zero mean and unity variance. $Z$ is calculated for each column by subtracting of the column means and dividing by the column standard deviations:

$$Z_n = Zscores(x_n) = \frac{x_n - \mu_n}{\sigma_n} = \frac{x_n - \frac{1}{M} \sum_{m=1}^{M} x_{mn}}{\sqrt{\frac{1}{M} \sum_{m=1}^{M} \left( x_{mn} - \frac{1}{M} \sum_{m=1}^{M} x_{mn} \right)^2}}, \quad \forall \ n \in [1, N] \quad (4)$$

$$Z = \begin{bmatrix}
  x_{11} - \mu_1 & \cdots & x_{1N} - \mu_N \\
  \sigma_1 & \cdots & \sigma_N \\
  \vdots & \ddots & \vdots \\
  x_{M1} - \mu_1 & \cdots & x_{MN} - \mu_N \\
  \sigma_1 & \cdots & \sigma_N
\end{bmatrix}$$

The covariance of the $Z$-scores is then computed by:

$$C = cov(Z) = \frac{1}{M} ZZ^T \quad (5)$$

$$= \frac{1}{M} \begin{bmatrix} Z_{11} & \cdots & Z_{1N} \\
  \vdots & \ddots & \vdots \\
  Z_{M1} & \cdots & Z_{MN} \end{bmatrix} \begin{bmatrix} Z_{11} & \cdots & Z_{1N} \\
  \vdots & \ddots & \vdots \\
  Z_{M1} & \cdots & Z_{MN} \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1N} \\
  \vdots & \ddots & \vdots \\
  c_{N1} & \cdots & c_{NN} \end{bmatrix}$$
This covariance matrix, $C$, is then $N \times N$ dimensions and represents the variance of each frame with respect to itself and the other frames. The eigenvector, $v_i$, and eigenvalue, $\lambda_i$, pairs of the $N \times N$ covariance matrix obey the following relationships:

$$Cv_i = \lambda_iv_i, \quad \text{where} \ v_i \perp v_j, \quad \forall \ i, j \in [1, N], \quad i \neq j$$

(6)

However, since all eigenvectors are desired they may also be computed to satisfy:

$$V^{-1}CV = D$$

(7)

Where $V$ is a modal matrix containing $N$ column vectors with eigenvectors of length $N$ and $D$ is an $N \times N$ diagonal matrix containing each eigenvalue along the diagonal, both arranged in ascending order. In the case of PCA, the eigenvalues of the covariance matrix are the variances of the data in the eigenvector direction.

$$V = [v_1 \ \cdots \ v_N] \quad \text{and} \quad D = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \vdots \\ 0 & & \lambda_N \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \vdots \\ 0 & & \sigma_N^2 \end{bmatrix}$$

(8)

The loadings that provide a linear combination to the principal components are then formed by the matrix $W$ where the columns of $V$ are arranged in descending order by their corresponding eigenvalue along the diagonal of $D$.

$$w_i = v_{N-i+1}, \quad \forall \ i \in [1, N]$$

where $\lambda_i \geq \lambda_{i+1}, \quad \forall \ i \in [1, N - 1]$

(9)

$$W = [v_N \ \cdots \ v_1]$$
Since the eigenvalues are from the covariance matrix, this simply orders the principal component loadings in descending order of variance. Implying that once transformed, the 1\textsuperscript{st} dimension of the principal components will contain the majority of variance from all the frames in \( X \) and the 2\textsuperscript{nd} principal component will contain the 2\textsuperscript{nd} most variance (in an uncorrelated or orthogonal direction) from all the frames in \( X \), and likewise for the remaining components.

**Principal Component Transformation (PCT)**

The coefficients of \( W \) are the principal component loadings that are used to transform pixels from the original dimensions of the data, \( X \), (temporal or spectral) into dimensions of principal components, \( Y \),

\[
Y = (W^T \text{Zscore}(I)^T)^T = \left( \begin{bmatrix} \nu_N^T \\ \vdots \\ \nu_1^T \end{bmatrix} \begin{bmatrix} \frac{x_{11} - \mu_1}{\sigma_1} & \cdots & \frac{x_{1M} - \mu_1}{\sigma_1} \\ \frac{x_{1N} - \mu_N}{\sigma_N} & \cdots & \frac{x_{MN} - \mu_N}{\sigma_N} \end{bmatrix} \right)^T
\]

The principal components, \( Y \), of size \( M \times N \) are composed of \( M \) pixels and \( N \) principal components. Unvectorizing \( Y \) back into an \( R \times C \times N \) three-dimensional matrix allows the visualization of the principal components as multiple two-dimensional images.

\[
Y = \begin{bmatrix} y_{11} \cdots y_{1C} \\ \vdots \ \vdots \ \vdots \\ y_{R1} \cdots y_{RC} \end{bmatrix}, \ldots, \begin{bmatrix} y_{11N} \cdots y_{1CN} \\ \vdots \ \vdots \ \vdots \\ y_{R1N} \cdots y_{RCN} \end{bmatrix}
\]
PCT Example

The principal component analysis may be easily understood graphically as well. Figure 1 shows an example of multi-spectral PCA. Figure 1.(a) shows a Red, Green, and Blue frame in their spectral feature space, where each spectral frame is an axis and each pixel’s spectral intensity z score is plotted along this coordinate system. The blue lines indicate the eigenvectors of RGB covariance. Strictly speaking, the eigenvectors should be scaled by the square root of their corresponding eigenvalue giving the standard deviation of the spectra in that direction; however, the eigenvectors are artificially lengthened for display purposes. In RGB space, the largest eigenvector, or the largest direction of variance, is seen along the eigenvector. \( \mathbf{v}_N = [0.5784, 0.5831, 0.5705]^T \). Figure 1.(b) shows the Principal Components of the RGB spectra once transformed. The largest covariance eigenvector now exists purely along the x-axis direction, \( \mathbf{v}_N = [1.0, 0.0, 0.0]^T \). In its simplest terms, PCA reorients data to a new axis define by the directions of largest variances.

![Figure 1: (a) RGB feature space, (b) Principal Components of RGB spectra](image)
Texture Analysis

Texture Analysis consists of defining measures for various patterns based on both their intensities and their spatial relationships.

Gray Level Co-Occurrence Matrix (GLCM)

A popular method for quantifying the spatial-intensity relationship is through a Gray Level Co-Occurrence Matrix (GLCM). This Matrix specifies the number of times two pixel intensity pairs co-occur, or reside in spatial relationship to one another based on predefined positions. The GLCM may be considered the joint probability of intensity pairs in an image. The GLCM, $G$, is generated from an $R \times C$ image, $I$, with intensity levels $i, j \in [1, L]$, by:

$$g_{ij} = \sum_{r=1}^{R} \sum_{c=1}^{C} \begin{cases} 1, & I(r, c) = i \land I(r \pm d\phi_1, c \pm d\phi_2) = j \\ 0, & \text{else} \end{cases}$$

(12)

where $d$ is the distance between pixels pairs and $\phi_{1,2}$ is a quantized angle from the pixel, $I(r,c)$, of interest. Typical quantized angles are shown in Table 1.

$G$, will be a $L \times L$ matrix that is calculated for one spatial distance (i.e. one $d$ and $\phi_{1,2}$). However, multiple sets of $G$ may also be generated for multiple distances such as the relationship of a pixel to the pixel at $90^\circ$ and the pixel at $45^\circ$. With multiple GLCMs, descriptors will be created for each direction and may be averaged or summed to capture the texture isotropically.

<table>
<thead>
<tr>
<th>Angle</th>
<th>$[\phi_1 \phi_2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^\circ$</td>
<td>$[0 \ 1]$</td>
</tr>
<tr>
<td>$45^\circ$</td>
<td>$[-1 \ 1]$</td>
</tr>
<tr>
<td>$90^\circ$</td>
<td>$[-1 \ 0]$</td>
</tr>
<tr>
<td>$135^\circ$</td>
<td>$[-1 \ -1]$</td>
</tr>
</tbody>
</table>
Texture Descriptors

Texture descriptors provide a quantifiable value that is unique to various types of texture within an image. These measures are computed from the GLCM, $G$, and are used as scalar values to describe the content of the Co-Occurrence Matrix.

The Contrast of the GLCM given by:

$$\text{Contrast} = \sum_{i=1}^{L} \sum_{j=1}^{L} (i - j)^2 g_{ij}$$

(13)

describes the range of intensity values between the two pixel pairs over the whole image. If $G$ is constant the contrast would be 0.

Correlation is a measure of how correlated each pixel is to its pair over the entire image and may be in the range of the typical negative and positive correlation, $[-1,1]$, respectively. The correlation is defined by:

$$\text{Correlation} = \frac{\sum_{i=1}^{L} \sum_{j=1}^{L} (i - \mu_i)(j - \mu_j)g_{ij}}{\sigma_i \sigma_j}$$

(14)

where the scalars $\mu_i = \sum_{j=1}^{L} ig_{ij}$ and $\mu_j = \sum_{i=1}^{L} jg_{ij}$ are the means computed from the rows and columns of $G$, respectively; The scalar standard deviations $\sigma_i = \sum_{j=1}^{L} (i - \mu_i)^2 g_{ij}$ and $\sigma_j = \sum_{i=1}^{L} (j - \mu_j)^2 g_{ij}$ are also computed from the row and columns of $G$, respectively. An image with large regions of similar textures would have a high correlation, where random images will contain no pixel correlation.

The Energy, sometimes referred to as the Angular Second Moment (ASM), of $G$ accounts for the uniformity of the image. If $G$ is constant, the energy is equal to one. Higher values of energy suggest that the image is less random. This is calculated by the sum of the squared pixel pairs:
Homogeneity sometime referred to as the Inverse Difference Moment (IDM), of an image measure how close the distribution of $G$ is to a completely diagonal GLCM and is computed by:

$$\text{Homogeneity} = \sum_{i=1}^{L} \sum_{j=1}^{L} \frac{g_{ij}}{1 + (i - j)^2}$$

A homogeneous image would be completely constant, and contain unvarying or slow varying gray values.

**GLCM and Descriptor Example**

An Example of a GLCM and its descriptors is shown for an eight intensity level image in Table 2 where $d = 1$ and $\phi = 0^\circ$ or $[0, 1]$ specifying a distance one pixel to the right.

<table>
<thead>
<tr>
<th>Image</th>
<th>Co-Occurrence Matrix</th>
<th>Texture Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I = \begin{bmatrix} 6 &amp; 5 &amp; 7 &amp; 6 &amp; 4 \ 6 &amp; 8 &amp; 4 &amp; 1 &amp; 2 \ 8 &amp; 5 &amp; 7 &amp; 5 &amp; 1 \ 8 &amp; 1 &amp; 2 &amp; 3 &amp; 2 \ 7 &amp; 1 &amp; 5 &amp; 1 &amp; 2 \end{bmatrix}$</td>
<td>$G = \begin{bmatrix} 1 &amp; 2 &amp; 3 &amp; 4 &amp; 5 &amp; 6 &amp; 7 &amp; 8 \ 1 &amp; 0 &amp; 3 &amp; 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 \ 2 &amp; 0 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 3 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 4 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 \ 5 &amp; 2 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 0 &amp; 2 \ 6 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 1 &amp; 0 &amp; 0 \ 7 &amp; 1 &amp; 0 &amp; 0 &amp; 0 &amp; 1 &amp; 1 &amp; 0 \ 8 &amp; 1 &amp; 0 &amp; 0 &amp; 1 &amp; 1 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>Contrast 9.7000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Correlation 0.2500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Energy 0.0750</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Homogeneity 0.3367</td>
</tr>
</tbody>
</table>
CHAPTER 3 Data Acquisition

Fire Test

On June 18th, 2011 a fire test was conducted at Raytheon Vision Systems (RVS) in Goleta, California. The day was windy and clear. The large amounts of data obtained on that day provided the means to analyze the temporal, spectral and spatial, characteristics of fire, heat plumes, and smoke plumes; the results of which are presented in this study.

Four types of cameras were mounted on a tower to record the combustion of various natural fuels in a large barbeque. The barbeque was positioned 875 yards away from the tower. The cameras included an 8-bit visible camera, and three 14-bit infrared (IR) cameras: cooled mid-wave infrared, cooled long-wave infrared, and uncooled long-wave infrared. Camera specifications are shown in Table 3. The fuels burned included wild oats, pine needles, pine cones, palm leaves, thistle, wet and dry leaves, wild rosemary, and apple wood. Attempts were made during the acquisition to incrementally add fuel to simulate a growing fire characterized by an initial small smoke output that ramps up to a larger smoke plume.

<table>
<thead>
<tr>
<th>Table 3: Camera Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible</td>
</tr>
<tr>
<td>Bit Depth</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>FPS</td>
</tr>
</tbody>
</table>
Data Format

The test generated over 300 GB of video. The IR video was stored as 16-bit RAW bit streams with zero padding on the least significant bit (LSB) to represent the 14-bit data, and the visible was stored as an 24-bit (8-bit per channel) AVI file. The Cooled MWIR and Cooled LWIR were stored as one dual band bit stream and the Uncooled LWIR was stored as a separate bit stream. Figure 2 explains the data acquisition process. From the Raw data obtained from the fire test, individual frames were created to utilize image processing techniques explained in the following chapters.

Figure 2: Data Acquisition Process
CHAPTER 4 Pre-Preprocessing

Unfortunately, the data collected from each camera contain multiple inconsistencies with respect to each other that required preprocessing before the intended analysis could be performed. Each of the four cameras had different fields of view (FOV) and three out of four were asynchronously recorded while only the visible video was time stamped. Both these issues posed a problem for any multispectral analysis which would require pixels to be spatially and temporally aligned across each spectral frame. To spatially align the FOVs, affine image registration was used. To temporally align the frames, key frames were manually selected in the beginning, middle and end of each video where noticeable events occurred within the scene such as a man walking out of a building or car passing behind a tree. The camera frame rates were all consistent so there was a linear relationship between the key frames.

Image Registration

With different FOVs for each camera there was no direct spatial correspondence between pixels in each frame. To solve this, an affine transformation was employed to register one FOV to another. In this case, the Visible Cameras had the narrowest FOV, and the uncooled long-wave IR had the largest FOV, with the cooled mid-wave IR and long-wave IR in the middle. The result is that the visible frame has a footprint in the IR frames, shown in Figure 3. To align the FOVs, the IR frames with resolutions all of 640 x 480 had to be registered to the visible frame with resolution 1086 x 876. Twenty correlation coordinates were hand selected in the original FOVs to define control points for the transformation between each frame based on recognizable features such as light poles, or building corners Figure 3 shows the control points.
The coordinate correlating was done at a pixel-level view of the scene to try to maintain as much accuracy as possible. These twenty pixel coordinates \( \{x_{IR}, y_{IR}\} \) and \( \{x_{Vis}, y_{Vis}\} \) are used to find a transformation between frames that will map the location of a visible pixel to a sub-pixel location within the IR frame since the visible resolution is larger than the IR resolution; that is,
The coefficients of the Transformation matrix are established by solving for a best-fit solution to the two systems:

$$\begin{bmatrix} x_{IR}^1 \\ \vdots \\ x_{IR}^{20} \end{bmatrix} = \begin{bmatrix} x_{Vis}^1 & y_{Vis}^1 & 1 \\ \vdots & \vdots & \vdots \\ x_{Vis}^{20} & y_{Vis}^{20} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$  \hspace{1cm} (18)

$$\begin{bmatrix} y_{IR}^1 \\ \vdots \\ y_{IR}^{20} \end{bmatrix} = \begin{bmatrix} x_{Vis}^1 & y_{Vis}^1 & 1 \\ \vdots & \vdots & \vdots \\ x_{Vis}^{20} & y_{Vis}^{20} & 1 \end{bmatrix} \begin{bmatrix} d \\ e \\ f \end{bmatrix}$$  \hspace{1cm} (19)

Once the coefficients are found, the mesh grid (a two dimensional array of Cartesian coordinates) of the visible frame is then transformed with the help of equation (17) to find its corresponding IR frame location, resulting in a sub-pixel mesh grid for the IR the same size as the visible frame. The IR frame is then resampled onto this new mesh grid through bicubic interpolation.

$$IR_{registered}(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x_{IR}^i y_{IR}^j$$  \hspace{1cm} (20)

where \(a_{ij}\) are the weighting coefficients derived from sixteen closest pixels in the unregistered IR frame. The registered fields of view are shown in Figure 4.
Each spectral frame is now registered and the transformed FOVs all coincide with the visible FOV. Both the cooled MWIR, Figure 4.(b), and the cooled LWIR, Figure 4.(c), registrations produced satisfactory results and the up sampled images retained enough detail to continue the analysis. The Uncooled LWIR, Figure 4.(d), registration, on the other hand did not up sample as well. The original FOV of the uncooled LWIR was the largest out of all the FOVs resulting in relatively low spatial resolution. The interpolation required to register the uncooled LWIR FOV with the visible FOV from the small amount of information is too great and the registered frame is corrupted by
sampling noise and loss of detail. Therefore, the Uncooled LWIR is not discussed in the remainder of the study, and the Mid-wave IR and Long-wave IR will be referred to as simply MWIR and LWIR, respectively.

**Keyframes**

The MWIR and LWIR cameras were recorded through synchronous bit streams, however they were not recorded synchronous to the visible video. Therefore any temporal analysis that was to be performed across the spectrum would not be correlated. The start times of each recording was placed in the filenames and the visible frames were time stamped which provided general start and stop times to align the frames. From this key frames were hand selected where noticeable events took place within the scene. This was done at the beginning, middle and end of the videos. By extrapolating from the key frames based on the frames per second of each video the MWIR/LWIR was temporally aligned with the visible video.
CHAPTER 5 Multispectral Principal Component Analysis

Original Raw Data
Analysis of the video began with considering the scene within the spectral domain.

Figure 5 shows one instance in time with a frame for each spectrum along with their pixel probability density functions or histograms in Figure 6.

The Range of values within each frame, the mean intensity, the standard deviation, the skewness, and the kurtosis is shown in Table 4. Standard deviation describes the dispersion of intensities from the mean. Skewness describes the general shape of the distribution, where positive values indicate that right side of the mean is longer and that most of the intensities lie to the left of the mean. Kurtosis is a measure of how much the distribution peaks with how long the tails are, also describing how prone the distribution is to outliers.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>MWIR</th>
<th>LWIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>242</td>
<td>238</td>
<td>222</td>
<td>8383</td>
<td>15170</td>
</tr>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5717</td>
<td>4089</td>
</tr>
<tr>
<td>Mean</td>
<td>79</td>
<td>80</td>
<td>79</td>
<td>6083</td>
<td>4265</td>
</tr>
<tr>
<td>Std</td>
<td>53.756</td>
<td>54.57</td>
<td>54.05</td>
<td>256.67</td>
<td>246.2</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.8685</td>
<td>0.9485</td>
<td>1.0845</td>
<td>1.3391</td>
<td>14.513</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>2.6517</td>
<td>2.8278</td>
<td>3.302</td>
<td>4.4854</td>
<td>524.47</td>
</tr>
</tbody>
</table>
While the skewness and the kurtosis of the first four bands, Red, Green, Blue and MWIR, are near the values for a normal Gaussian distribution, both the skewness and the kurtosis of the LWIR describe a different situation. Due to the large counts of intensity values near the mean of the LWIR the small number of values near the max are not seen in the histogram. However, both the large skew and kurtosis suggest that there is a grouping of high intensity outliers. These high intensities come from the heat of the fire.

Another view of the data as a two-dimensional feature plot, shown in Figure 7, provide further insight into the nature of the LWIR skew. Each scatter is comprised of two spectral features where the pixel intensities of one band is plotted against the intensities within another band. The diagonal plots are shown for completeness but do not represent anything since the bands are plotted against themselves. The LWIR does in fact contain a group of high intensity value outliers. This remains constant for most of the frame throughout the video. Theoretically these large values should offset the results of any PCA performed spectrally. Nonetheless, an initial PCA is performed on these spectral frames.
Each frame is 768 x 768 pixels and is transformed into 589824 x 1 column vector to build a 589824 x 5 matrix containing all the pixels for each of the 5 bands ordered by Red, Green, Blue, MWIR, and LWIR. The 5 x 5 correlation matrix, $C$, is computed:

$$
C = \begin{bmatrix}
2889.7 & 2901.3 & 2798.0 & 1016.8 & 2131.9 \\
2901.3 & 2977.9 & 2863.8 & 536.74 & 1824.3 \\
2798.0 & 2863.8 & 2921.4 & -1150.7 & 549.39 \\
1016.8 & 5326.74 & -1150.7 & 65881 & 51893 \\
2131.9 & 1824.3 & 549.39 & 51983 & 60616
\end{bmatrix}
$$

The diagonal of the covariance matrix are the variances of each band, and the overly large variances of MWIR and LWIR at $c_{44}$ and $c_{55}$, respectively will alter the basis resulting from PCA. The eigenvectors calculated from this covariance show the basis in the direction of the large variances of the MWIR and LWIR where the first principal component consists almost entirely of these two bands. The Loadings for the first principal component are the coefficients along the first column vector in $W$,

$$
W = \begin{bmatrix}
0.0199 & -0.2402 & -0.5218 & -0.4674 & 0.6717 \\
0.0151 & -0.2518 & -0.5272 & -0.3402 & -0.7367 \\
-0.0032 & -0.2677 & -0.5073 & 0.8155 & 0.0778 \\
0.7244 & 0.6248 & -0.2900 & 0.0279 & -0.0040 \\
0.6889 & -0.6457 & 0.3293 & -0.0046 & 0.0013
\end{bmatrix}
$$

$$
D = \begin{bmatrix}
115278 & 0 & 0 & 0 & 0 \\
0 & 12136.6 & 0 & 0 & 0 \\
0 & 0 & 7761.04 & 0 & 0 \\
0 & 0 & 0 & 80.2815 & 0 \\
0 & 0 & 0 & 0 & 29.8770
\end{bmatrix}
$$
The large IR proportion of the first PC is shown in Figure 8 where the PCT loadings are plotted over each band to show the relative contributions for each Principal Component.

The PCT of the original 589824 x 5 matrix are unvectorized back into five 768 x 768 frames shown in Figure 9. The images shown in Figure 9 represent the floating point values linearly mapped back to 8-bits. The original floating point values are shown in the histograms of Figure 10 and their statistics are tabulated in Table 5.
Z-score of the Raw Data
As expected, the large scale of the MWIR and the LWIR dominated the principal components of the spectrum. The majority of the coefficients are loaded towards the IR, and to not contain much of the visible spectrum. To account for the differences in scale of the 14-bit and 8-bit data the statistical Standard score or Z-score is employed. This Z-score will normalize the data such that the effect of the IR has on the PC loadings is reduced. By using mean subtracted and standard deviation divided data to create the Z-scores, the normalized data has zero mean and unity variance. The histograms of the Z-scores are shown in Figure 11.
The statistics for the Z-scores in Table 6 show the zero mean and unity variance while the skewness and kurtosis that describe the overall shape of the distribution is left unchanged. When PCA is carried out on the normalized data the IR bands do not dominate the Principal Components as evident by the matrices $C$, $W$, and $D$ as well as the plot of spectral contributions for each component shown in Figure 12.

$$C = \begin{bmatrix} 1 & 0.9890 & 0.9630 & 0.0737 & 0.1611 \\ 0.9890 & 1 & 0.9710 & 0.0383 & 0.1358 \\ 0.9630 & 0.9710 & 1 & -0.0829 & 0.0413 \\ 0.0737 & 0.0383 & -0.0829 & 1 & 0.8212 \\ 0.1611 & 0.1358 & 0.0413 & 0.8212 & 1 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.5759 & -0.0308 & -0.0977 & -0.4652 & 0.6644 \\ 0.5756 & -0.0550 & -0.0552 & -0.3331 & -0.7427 \\ 0.5641 & -0.1392 & 0.0742 & 0.8065 & 0.0802 \\ 0.0603 & 0.7065 & -0.6898 & 0.1451 & -0.0194 \\ 0.1235 & 0.6910 & 0.7114 & -0.0332 & 0.0065 \end{bmatrix}$$

$$D = \begin{bmatrix} 2.9741 & 0 & 0 & 0 & 0 \\ 0 & 1.8133 & 0 & 0 & 0 \\ 0 & 0 & 0.1755 & 0 & 0 \\ 0 & 0 & 0 & 0.0270 & 0 \\ 0 & 0 & 0 & 0 & 0.0102 \end{bmatrix}$$

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>MWIR</th>
<th>LWIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>3.0413</td>
<td>2.8882</td>
<td>2.6474</td>
<td>8.9609</td>
<td>44.1683</td>
</tr>
<tr>
<td>Min</td>
<td>-1.4605</td>
<td>-1.4732</td>
<td>-1.4600</td>
<td>-1.4259</td>
<td>-0.8395</td>
</tr>
<tr>
<td>Std</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.8685</td>
<td>0.9485</td>
<td>1.0845</td>
<td>1.3391</td>
<td>14.513</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>2.6517</td>
<td>2.8278</td>
<td>3.3020</td>
<td>4.4854</td>
<td>524.47</td>
</tr>
</tbody>
</table>

Table 6: Z-score Frame Statistics
Using the Z-score before PCA, produces components which include more of each spectrum. The large high intensity values of the fire from the MWIR and LWIR do not dominate the first principal component, but occur in higher order components. Figure 13 shows the PCs linearly mapped to 8-bits. Figure 14 and Table 7 show the histograms and statistic of the PCs.

![Figure 12: Contribution of each spectral Z-score to the Principal Component loadings](image)

**Figure 12: Contribution of each spectral Z-score to the Principal Component loadings**

![Figure 13: Principal Components of the Z-scores](image)

(a) PC1, (b) PC2, (c) PC3, (d) PC4, (e) PC5

**Figure 13: Principal Components of the Z-scores**

![Figure 14: Principal Components of Z-score](image)

**Figure 14: Principal Components of Z-score**

<table>
<thead>
<tr>
<th></th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
<th>PC4</th>
<th>PC5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>7.3265</td>
<td>34.0936</td>
<td>29.0876</td>
<td>0.9399</td>
<td>1.0008</td>
</tr>
<tr>
<td>Min</td>
<td>-2.6317</td>
<td>-1.9651</td>
<td>-0.9419</td>
<td>-1.0087</td>
<td>-1.2036</td>
</tr>
<tr>
<td>Mean</td>
<td>-6.4146e-16</td>
<td>1.2814e-15</td>
<td>-3.5034e-15</td>
<td>-2.9298e-17</td>
<td>5.5318e-17</td>
</tr>
<tr>
<td>Std</td>
<td>1.7246</td>
<td>1.3466</td>
<td>0.41889</td>
<td>0.16422</td>
<td>0.10083</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.86067</td>
<td>3.5396</td>
<td>44.686</td>
<td>-0.47175</td>
<td>0.18784</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>2.6297</td>
<td>61.845</td>
<td>2574.3</td>
<td>4.196</td>
<td>8.5442</td>
</tr>
</tbody>
</table>

**Table 7: Principal Component Z-score Statistics**
Nonlinearity of Infrared

While the Z-score of each band alleviated the disparity among the variances, the MWIR and LWIR distribution shapes are still skewed. To solve this issue a nonlinear mapping it performed on each pixel’s intensity through the relation,

\[ y = 256 \left( \frac{x^\gamma - x_{\min}^\gamma}{x_{\max}^\gamma - x_{\min}^\gamma} \right), \quad \forall \ x \in X, \quad 0 \leq \gamma \leq 2 \]

where \( \gamma \) is the degree of nonlinearity applied to the 14-bit image \( X \), and \( Y \), is the floating point output image containing all \( y \) pixels. The transfer function of this mapping is shown for multiple \( \gamma \) in Figure 15. Most of the information is contained in the low values of image, so a \( \gamma \) which extenuates this region is desired. The fire is present in the high intensity values of the image, while the heat plume is more in the midrange values. In Figure 16 the 14-bit MWIR input image and the floating point MWIR output image is plotted along with their histograms to show the transformation of the intensity values. Since the Z-scores of the spectrum are taken after the nonlinear mapping, the scale factor of floating point 256 does not matter. Only the degree to which the IR frames are mapped will alter the PCA.
To understand the effect of the nonlinear mapping to the contributions of the PCA, the magnitudes of the principal component loadings are evaluated for a range of nonlinear degrees. The nonlinearity of the IR alters the extent that each band correlates with each other, thus moving the direction of the covariance matrix eigenvectors from which the principal component basis is formed. These loading magnitudes are plotted over $\gamma$ values from 0 to 2 in Figure 17.
The nonlinearity of the IR unquestionably affects the contributions to each principal component as seen in Figure 17. Especially in the case of PC1, a zero IR contribution exists around $\gamma$ equal to 0.7. Likewise, PC2 contains zero visible information for this IR nonlinearity. This shows that for some non linear mappings the 1st and 2nd PCs will only contain information from one side of the spectrum. The 3rd, 4th, and 5th PCs have larger ranges of contribution, where an optimal point may be found. With the goal of aggregating the spectral content from all bands through PCA, a mapping that produces principal components with as much information from all bands is desired.
To find an optimal $\gamma$, the mean value of the loadings and their standard deviations for each PC is plotted for $\gamma$ values in the range of 0 to 2. When the contributions from each band are most equal, the standard deviation of the loadings will be small. When the mean is large and the standard deviation is small the value of $\gamma$ is produce the most similar and largest contribution from each of the spectral bands. This situation exists in the 3$^{rd}$ principal component as seen in Figure 18 at $\gamma = 0.51$. There is another minimum point in the 4$^{th}$ PC for a little higher value of $\gamma$, however since the higher order PC represent lower degrees of variance and less information using an optimal point in a lower PC is preferable.
This mapping mildly emphasizes the mean and shifts the low intensity values to higher. The raw Z-score histograms and Nonlinear Z-score histograms in Figure 11 and Figure 19, respectively, show this emphasis. A comparison of the Raw Z-score statistics and the Nonlinear Z-score statistics in Table 6 and Table 8, respectively, also show this emphasis as well as a reduction in both skewness and kurtosis, providing more of a Gaussian-like distribution.

Table 8: Z-score statistic after a nonlinear mapping with $\gamma = 0.51$

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>MWIR</th>
<th>LWIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>3.0413</td>
<td>2.8882</td>
<td>2.6474</td>
<td>5.352</td>
<td>15.843</td>
</tr>
<tr>
<td>Min</td>
<td>-1.4605</td>
<td>-1.4732</td>
<td>-1.4600</td>
<td>-2.7904</td>
<td>-2.1494</td>
</tr>
<tr>
<td>Mean</td>
<td>-1.4735e-14</td>
<td>1.4464e-15</td>
<td>-4.6259e-15</td>
<td>-7.3645e-14</td>
<td>-1.2299e-13</td>
</tr>
<tr>
<td>Std</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.8685</td>
<td>0.9485</td>
<td>1.0845</td>
<td>0.5673</td>
<td>1.9147</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>2.6517</td>
<td>2.8278</td>
<td>3.3019</td>
<td>3.0681</td>
<td>14.559</td>
</tr>
</tbody>
</table>

PCA is performed on these Z-scores in the same fashion as the raw frames, the results of which are shown in Figure 21. These principal components were generated from the new PCT matrix, $\mathbf{W}$, based on the eigenvectors of the new covariance matrix $\mathbf{C}$,

$$
\mathbf{C} = \begin{bmatrix}
1 & 0.9890 & 0.9630 & -0.0867 & 0.0840 \\
0.9890 & 1 & 0.9710 & -0.1225 & 0.0487 \\
0.9630 & 0.9710 & 1 & -0.2442 & -0.0703 \\
-0.0867 & -0.1225 & -0.2442 & 1 & 0.9587 \\
0.0840 & 0.0487 & -0.0703 & 0.9587 & 1
\end{bmatrix}
$$
The new loadings resulting from the nonlinear mapping are shown in Figure 20. The 1st principal component is still weighted towards the three visible bands, and the 2nd towards the two IR bands, however the 3rd has a similar magnitude of contribution for each of the five bands. The 4th is weighted towards blue and MWIR while the 5th is mostly Red and Green.

The new loadings resulting from the nonlinear mapping are shown in Figure 20. The 1st principal component is still weighted towards the three visible bands, and the 2nd towards the two IR bands, however the 3rd has a similar magnitude of contribution for each of the five bands. The 4th is weighted towards blue and MWIR while the 5th is mostly Red and Green.

\[
W = \begin{bmatrix}
-0.5650 & 0.1349 & 0.3830 & -0.2750 & 0.6635 \\
-0.5691 & 0.1094 & 0.2819 & -0.1784 & -0.7435 \\
-0.5730 & 0.0203 & -0.5946 & 0.5576 & 0.0823 \\
0.1584 & 0.6842 & 0.4136 & 0.5793 & -0.0028 \\
0.0589 & 0.7080 & -0.4993 & -0.4959 & -0.0111
\end{bmatrix}
\]

The new loadings resulting from the nonlinear mapping are shown in Figure 20. The 1st principal component is still weighted towards the three visible bands, and the 2nd towards the two IR bands, however the 3rd has a similar magnitude of contribution for each of the five bands. The 4th is weighted towards blue and MWIR while the 5th is mostly Red and Green.

Figure 20: Principal Component Loadings from Z-scores nonlinearly mapped with $\gamma = 0.51$
For comparison to the Raw and Z-score PCA, Figure 22 and Table 9 show the distributions of the PCs derived from the nonlinear Z-scores. From the $\gamma$ options available in Figure 18, this represents the best mixture of the spectrum. The remainder of this study will use nonlinear mapping for further analysis and smoke plume detection.

**Spectral Zscore ($\gamma_{IR} = 0.51$) Principal Component Histograms**

![Histograms of principal components](image)

**Figure 22: Principal Components of Z-scores nonlinearily mapped with $\gamma = 0.51$**

**Table 9: PC statistics for nonlinear Z-scores**

<table>
<thead>
<tr>
<th></th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
<th>PC4</th>
<th>PC5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>2.7438</td>
<td>14.5428</td>
<td>0.7702</td>
<td>0.6637</td>
<td>0.9990</td>
</tr>
<tr>
<td>Min</td>
<td>-4.2997</td>
<td>-2.2148</td>
<td>-6.9317</td>
<td>-6.3611</td>
<td>-1.1992</td>
</tr>
<tr>
<td>Mean</td>
<td>-2.0955e-14</td>
<td>-1.5737e-13</td>
<td>3.4699e-14</td>
<td>2.0833e-14</td>
<td>1.8436e-15</td>
</tr>
<tr>
<td>Std</td>
<td>1.7287</td>
<td>1.3957</td>
<td>0.1733</td>
<td>0.1521</td>
<td>0.1009</td>
</tr>
<tr>
<td>Skewness</td>
<td>-1.2139</td>
<td>1.4547</td>
<td>-9.4442</td>
<td>-10.239</td>
<td>0.1982</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>3.5051</td>
<td>5.3012</td>
<td>321.40</td>
<td>343.39</td>
<td>8.5357</td>
</tr>
</tbody>
</table>
CHAPTER 6 Multi-Spectral-Temporal Principal Component Analysis

With the results of the Spectral Principal Components establishing the degree of nonlinearity that should be applied to apply to the Infrared, PCA will be performed to the temporal dimension simultaneously with the spectral dimension. When performing principal component analysis on a set of multi-temporal images, information that is highly correlated among all the temporal frames appears in the lower principal component, while information that is unique with relatively low occurrence in the analyzed frames appears in the higher order principal components. Figure 23 shows the data flow of this temporal-spectral PCA. The Nonlinearity will be applied to the IR frames, all frames will be vectorized, normalized by their standard score, analyzed for their covariance eigenvectors and finally transformed into their Principal Components.

![Figure 23: PCA Process](image-url)
PCA at two time instances

Adding one instance of time to the spectral PCA will have the effect of aggregating the spectral content and producing temporal change detection. Figure 25 shows a temporal sequence of 2 visible frames. In this case (b) would be the current time sample at time $t$, and (a) is the previous sample at $t - t_s$, where $t_s$ is 1 second or 30 frames. The frames of each spectral band at these times are shown in Figure 24. This type of sequence would represent a previously static scene that an initial plume begins to appear in. After the PCT, some of the PCs should capture the variance of the temporal signature of the smoke and some should capture the spectral signature of the smoke.

Figure 25: Visible Image of scene for two instances in time.
(a) Minimal smoke plume
(b) Expanding smoke plume

Figure 24: Spectral bands for the same two instance of time in Figure 25
Each 768 x 768 frames is normalized and vectorized into a 589824 x 1 column vector and grouped into a 589824 x 10 matrix representing the 10 dimensions of 5 bands over 2 time instances. This sequence of frames then yields the 10 x 10 covariance matrix, C:

\[
C = \begin{bmatrix}
1 & 0.9889 & 0.9635 & -0.0879 & 0.0920 & 0.9939 & 0.9853 & 0.9588 & -0.0848 & 0.0806 \\
0.9889 & 1 & 0.9713 & -0.1234 & 0.0581 & 0.9861 & 0.9938 & 0.9673 & -0.1200 & 0.0478 \\
0.9635 & 0.9713 & 1 & -0.2446 & -0.0570 & 0.9605 & 0.9682 & 0.9931 & -0.2409 & -0.0652 \\
-0.0879 & -0.1234 & -0.2446 & 1 & 0.9335 & -0.0886 & -0.1230 & -0.2457 & 0.9984 & 0.9396 \\
0.0920 & 0.0581 & -0.0570 & 0.9335 & 1 & 0.0913 & 0.0582 & -0.0583 & 0.9393 & 0.9817 \\
0.9939 & 0.9861 & 0.9605 & -0.0886 & 0.0913 & 1 & 0.9889 & 0.9633 & -0.0854 & 0.0799 \\
0.9853 & 0.9938 & 0.9682 & -0.1230 & 0.0582 & 0.9889 & 1 & 0.9710 & -0.1194 & 0.0479 \\
0.9588 & 0.9673 & 0.9931 & -0.2457 & -0.0583 & 0.9633 & 0.9710 & 1 & -0.2420 & -0.0665 \\
-0.0848 & -0.1200 & -0.2409 & 0.9984 & 0.9393 & -0.0854 & -0.1194 & -0.2420 & 1 & 0.9277 \\
0.0806 & 0.0478 & -0.0652 & 0.9196 & 0.9817 & 0.0799 & 0.0479 & -0.0665 & 0.9277 & 1
\end{bmatrix}
\]

The eigenvectors of this covariance matrix correctly ordered into the PC Transformation matrix, W, along with the corresponding eigenvalue matrix, D, that represent the variances contained in the components, are:

\[
W = \begin{bmatrix}
-0.4002 & 0.0912 & 0.1092 & 0.3285 & -0.4448 & -0.2179 & -0.3551 & -0.1549 & 0.5623 & -0.0109 \\
-0.4030 & 0.0735 & 0.0790 & 0.2273 & 0.4473 & 0.2123 & -0.4193 & 0.5831 & -0.1151 & -0.0051 \\
-0.4054 & 0.0109 & -0.1039 & -0.5557 & -0.0806 & -0.0473 & -0.4454 & -0.3733 & -0.4117 & 0.0043 \\
0.1092 & 0.4848 & 0.5044 & -0.1602 & 0.0151 & -0.0362 & -0.0090 & 0.0103 & 0.0031 & 0.6864 \\
0.0355 & 0.5022 & -0.4018 & 0.0486 & -0.3266 & 0.6899 & 0.0024 & -0.0034 & 0.0013 & -0.0098 \\
-0.4004 & 0.0909 & 0.1092 & 0.3102 & -0.3900 & -0.1893 & 0.4409 & 0.1498 & -0.5643 & -0.0036 \\
-0.4030 & 0.0736 & 0.0815 & 0.2067 & 0.5066 & 0.2346 & 0.3503 & -0.5781 & 0.1186 & 0.0144 \\
-0.4053 & 0.0103 & -0.1041 & -0.5768 & -0.0268 & -0.0209 & 0.4272 & 0.3740 & 0.4105 & -0.0033 \\
0.1079 & 0.4869 & 0.4406 & -0.1501 & 0.0536 & -0.0774 & 0.0001 & -0.0080 & -0.0039 & -0.7250 \\
0.0371 & 0.4980 & -0.5759 & 0.0970 & 0.2763 & -0.5750 & 0.0020 & 0.0022 & -0.0006 & 0.0518
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
5.9612 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 3.8331 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.1022 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0485 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0180 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.0168 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.0132 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0032 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0027 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0013
\end{bmatrix}
\]

W is used compute the linear combination of the PCT. Once the 589824 x 10 matrix of Principal Components are obtained the matrix is unvectorized back into 10 frames of 768 x 768 pixels. The floating point images are mapped back to 8-bits with \( y = \)equal to 0.5. These 8-bit Principal Component images are shown in Figure 26.
By adding the temporal dimension to the analysis the smoke appears in the 7th Principal Component at point A in Figure 27. However, the 7th PC also shows passing car, B, and a truck, C.
Figure 28: Magnitude of PCT loadings from each dimension of the spectral/temporal image set

While the weights or contributions of each 10 spectral or temporal dimensions for the $n^{th}$ principal component may be obtained from the $n^{th}$ column of $W$, the large matrix is somewhat difficult to decipher. Figure 28 provides a more efficient way to determine which temporal or spectral dimensions had the most affect on a given Principal component. The absolute values of the PCT loadings are shown against the dimension from which they are comprised of. In the case of the $7^{th}$ PC containing the smoke plume the PC is mostly made up from the current and previous visible bands, with a little of the
previous MWIR. This indicates that the IR is not providing significant information for smoke plume detection.

**Effect of noise on plume segmentation**

To test the strength and capability of this PCT against non smoke plume noise rejection, the same PCT matrix, $W$, was applied to another two instances in time. During this sequence there is an extremely weak smoke signal and large ‘noise’ signal generated by a bus traveling through the frame. This sequence is shown in Figure 29 where the bus enters in the second frame. The principal components of the sequence are shown in Figure 30. The $7^{th}$ PC which normally contains the smoke plume signal does not show smoke at this instance and is polluted by the bus. This time instance will be used to test the final approach for its ability to reject non smoke plume signals.
Figure 30: Temporal-Spectral PCs for time instance shown in Figure 29

Effect of sample time

Figure 31 shows which Principal Component the smoke plume and heat plume appear in for different sample times. It is interesting to note that at $t_s = 3.333$ s and $t_s = 9$ s the 6th PC captures both the smoke plume and the heat plume.

The final parameters for the PCT were then chosen based on the results from the texture analysis discussed in Chapter 7. The texture analysis was performed for
different combinations of sample rates and sample numbers. The one which provided the highest spatial correlation was a PCA of 3 spectral frames sampled at 60 seconds used for the remainder of the Processing, an instance of this is shown in Figure 32.

Figure 32: PCs of 3 spectral frames sampled at 60 seconds
CHAPTER 7 Texture Analysis in Principal Component Space

Gray Level Co-Occurrence Matrix

With the smoke plume largely segmented from the background in the 7th Principal Components, a noticeable feature is that the smoke is consistently uniform compared to the background. The Principal Components that contain the plume has random noise everywhere but the plume. This is also the case during frames when there is no plume. This uniformity is unique to the smoke plume alone, and may be described by a spatial similarity of pixels within the plume. To identify the spatial similarity, texture descriptors computed from the PC’s Gray Level Co-occurrence Matrix (GLCM) are used to describe spatial features in the principal components.

Figure 33: Construction of GLCM
The Process shown in Figure 33 describes the construction of a Gray-Level Co-Occurrence Matrix for multiple directions from a grayscale image. In the example case of Figure 33, the $0^\circ$ GLCM is the sum of co-occurrences for each intensity that exist next to every other intensity within the spatial relationship of four pixels to the right. This is done for all 256 intensities values, thus the GLCM is $256 \times 256$. The $45^\circ$, $90^\circ$, and $135^\circ$ GLCMs are computed in the same way. There is a large density of spatial pairs around 128 and 128; this is due to the nonlinear mapping that places the 0 of the floating point Principal Component at 128 mid gray. Figure 34.(b)-(e) shows the GLCM of four different pixel spatial relations for the Principal Component shown in Figure 34.(a)

Table 10 tabulates the texture descriptors for these selected GLCMs.

![Figure 34: Selected Gray Level Co-Occurrence Matrixes, (a) PC, (b) [-20, -20], (c) [-2, 0], (d) [0, -2], (e) [6, 16]]

<table>
<thead>
<tr>
<th>GLCM</th>
<th>(a)[-20, -20]</th>
<th>(b) [-2, 0]</th>
<th>(c) [0, -2]</th>
<th>(d) [6, 16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contrast</td>
<td>67.098</td>
<td>57.753</td>
<td>56.459</td>
<td>65.454</td>
</tr>
<tr>
<td>Correlation</td>
<td>0.0040</td>
<td>0.1372</td>
<td>0.1564</td>
<td>0.0246</td>
</tr>
<tr>
<td>Energy</td>
<td>0.0030</td>
<td>0.0032</td>
<td>0.0033</td>
<td>0.0030</td>
</tr>
<tr>
<td>Homogeneity</td>
<td>0.2435</td>
<td>0.2753</td>
<td>0.2908</td>
<td>0.2483</td>
</tr>
</tbody>
</table>
For the Principal Component containing the smoke plume GLCMs are computed for distances up to 20 pixels and angles from 0 to 360° to establish an optimal distance for identifying the smoke plume texture. This creates GLCMs for all pixel pairs within a 20 x 20 region. Each one of these GLCM was evaluated for the contrast, correlation, energy, and homogeneity descriptors.

Figure 35 shows the descriptor values of GLCMs in all directions for a scene with smoke and a scene without smoke. As expected, the farther the spatial relationship becomes, the more contrast exists between all pixels in the image. This is measures the amount of local variation, and is explained by an increase in the likelihood that pixels farther away will have a larger range of intensity value variation. In addition, the correlation, energy and homogeneity decrease as the spatial distance increases. Intuitively, pixels close to each other should have a high correlation where as distant pixels should be uncorrelated. All of these describe levels of randomness within the image. For a completely random image, even nearby pixels will be uncorrelated, non uniform (energy) or non-homogeneous. Yet, for an image that has a large non random object or feature, the pixels that are part of this texture will have a higher correlation even when they are spatially distant. Likewise if the texture is uniform the texture should have higher energy and homogeneity.

Figure 36 is the difference of the descriptors between the scene with the smoke plume and the scene without the smoke plume. Clearly there is not much difference in energy or homogeneity to indicate the presence of a plume; however, contrast and correlation, in particular, provide a noticeable difference for some spatial directions. The correlation difference reaches a maximum at a distance of 10 pixels.
Figure 35: Texture Descriptors for principal components

GLCM Texture Descriptors during/before smoke difference for $d = [0, 20]$ & $\phi = [0, 360]$

Contrast

Correlation

Energy

Homogeneity

Figure 36: Difference between Texture Descriptors of principal component with and without smoke plume
GLCM Texture Descriptors evaluated through time.

The Statistics generated from the GLCM of the 7th PC will now be evaluated over time for the whole video. The Processes involves taking three instances of the 5 bands and transforming them by the PCT Matrix, W, built in Chapter 6. From these PCs the 7th is used to create four GLCMs, as described above, for the four 10 pixel distance angles. This spatial relationship represents the maximum difference between PCs with smoke and PCs without smoke. An overall texture correlation is determined based on the summation of the correlation at each pixel angle. This is done for an OR operation effect for the probabilities of a high correlation in any direction instead of an AND operation which would specifying that all directions must be correlated. This process is continued for each new sample over the course of the entire video, each time using the same PCT computed earlier for saving computational efficiency. Figure 37 shows the results of each texture descriptor evaluated through time for the video.

![Contrast](image1)
![Correlation](image2)
![Energy](image3)
![Homogeneity](image4)

Figure 37: Four Texture Descriptors for the summation of 7th PC GLCMs (0, 45, 90, and 135) through time
The video sequence analyzed began with a very low amount of smoke that grows to a billowing plume at $t = 1100 \text{ s}$ which lasts 600 seconds, then dies down. From the results of the texture descriptors, correlation provides the most interesting response. Contrast, Energy, and Homogeneity do not work as smoke texture descriptors because they are contaminated by objects that produce high contrast or uniform objects in the 7th PC such as passing vehicles and pedestrians. The large spikes in these descriptors are explained by these events. Correlation, on the other hand, tracks the spatial size of plume directly. The plot in Figure 38 shows the correlation for each direction used for the measurement. Pixels directly to the right do not discriminate the time instances with smoke verses time instances without smoke. Since smoke tends to rise, it is not surprising that the best directions for differentiating the presence of smoke are the three upper distances (directly over head and to the left and right). During the time when smoke is in the scene these spatial relationships are highly correlated.

![GLCM Correlation for each spatial distance](image)

**Figure 38:** GLCM Correlation for each spatial distance 0, 45, 90, and 135
CHAPTER 8 Conclusion

The Early Forest Fire Detection method performed in this study is explained from raw data acquisition to smoke plume detection in the block diagram of Figure 39. By utilizing the temporal and spectral variances of smoke, Principal Component Analysis segments the plume from the surrounding environment. From the principal component that contains the variances of the smoke plume, texture analysis classifies the presence of smoke within the scene for a given instance of time. The classification is accomplished by a spike in the correlation of the spatial texture of the principal component containing smoke. The final output of the image processing techniques is shown in Figure 40. The 7th PC is shown above and below the correlation plot for the times indicated by the red lines. The size of the plume is proportional to the spatial correlation of the smoke plume texture. As the smoke grows in size the correlation increases, and as the smoke size diminishes the correlation decreases. A correlation threshold at 0.5 would signal the detection of the early onset of a forest fire. Time instances before and after the outburst of the plume contain object movement which would cause a false alarm through other methods. The most notable object is in the bottom right image of Figure 40 which shows
the presence of a bus, however the correlation which captures the plume and signals the
detection of the smoke is not affected by this interference. This method is
computationally inexpensive by utilizing a multispectral-multitemporal dimension
reduction through PCA to segments the plume. As indicated by the eigenvectors of the
PCA the IR hardly contributes any information, and the same results are obtained when
only the Red, Green, and Blue spectrums are used. And the generated of the GLCM is
computationally acceptable within the sample rate needed. Based on the smoke PC, the
spatial texture is evaluated in such a way that smoke is easily discriminated from the
environment and moving objects such as vehicles and pedestrians.

Figure 40: GLCM Correlation for 7th PC showing start, continuation and end of smoke plume. The images
shown above and below the correlation plot are 7th PCs at the times indicated by the red lines.
Bibliography


Appendix A: MATLAB Scripts and Functions

%% Matlab Script  
% Script      : CLWMWIRvid2frame.m
% Author      : Tim Davenport
% Date        : 10/1/11
% Description : Extracts individual frames of the dual band Cooled Long
%               Wave InfraRed (CLWIR) & Cooled Mid Wave InfraRed (CMWIR)
%               .raw video streams from the June 18 fire test at Raytheon
%               Vision Systems (RVS) in Goleta, CA.
% Notes       : 1. Video files were originally provided in the folder:  
%                Fire Data 6-18-11 MW LW  
%                It has since been renamed to:  
%                Fire Data 6-18-11 UCLW CMW CLW  
% 2. The test video files are separated into multiple "burns"  
%    that used various types of fuel. Both the source files  
%    and the resulting frames are labeled as such, and the  
%    frames for each burn are stored in separate folders.  
% 3. Pixel data is 14-bit w/ LSBs zero-padded to 16  
%    so frame is rightshifted by dividing by 4.  
%    Eg: #### #### #### ##00 >> ## #### #### ####  
% 4. Dual band video was captured with MidWave of left and  
%    LongWave on right (MW has less 'blooming')  
%    Resolution is (2*640=1280)x480  
%    Frame Rate is 60 fps for both (30fps per band)  
%    Frame size is 1280x480x2 = 1228800 bytes  
% Revision(s) : 01/27/12 - cleaned code: updated description, comments, and  
%                     superfluous syntax to be more elegant/succinct.
%------------------------------------------------------------------------

close all; clear; clc; tic
time = tic;

%% User Input: Source & Destination directory locations
% Source location: raw video files
vidfldr = ['Volumes/FreeAgent GoFlex Drive/EFFD/Original_Data/'
           'Fire_Data_6-18-11_UCLW_CMW_CLW/';
% Target directory: raw frames will be stored here
frmfldr = '/Volumes/Macintosh HD 2/Thesis/';

%% Initialize & Allocate
MWcols = 640;
LWcols = 640;
nrows = 480;
frmnum = 0;
fps = 30;

% filename format: date_timestamp_fuel.raw
vidfiles = 
    {'2011-06-18 101158 LW-MW-14bit_setup.raw'
    '2011-06-18 102125 LW-MW-14bit_charcoal1.raw'
    '2011-06-18 102454 LW-MW-14bit_oats1.raw'
    '2011-06-18 102925 LW-MW-14bit_oats2.raw'
    '2011-06-18 102949 LW-MW-14bit_oats3.raw'
    '2011-06-18 103820 LW-MW-14bit_pine_needles.raw'
    '2011-06-18 104857 LW-MW-14bit_pine_cones.raw'
    '2011-06-18 105906 LW-MW-14bit_pine_cones.raw'
    '2011-06-18 110557 LW-MW-14bit_palm.raw'
    '2011-06-18 110912 LW-MW-14bit_thistle.raw'
    '2011-06-18 111644 LW-MW-14bit_wet_leaves.raw'
    '2011-06-18 112306 LW-MW-14bit_wet_dryl.raw'
    '2011-06-18 112757 LW-MW-14bit_wet_dry2.raw'
    '2011-06-18 113916 LW-MW-14bit_rosemary.raw'
    '2011-06-18 114605 LW-MW-14bit_applewood1.raw'}
vidlabels = {'setup1', 'charcoal1', 'oats1', 'oats2', 'oats2b', 'oats3', 'pine_needles', 'pine_cones', 'palm', 'thistle', 'wet_leaves', 'wet_dry1', 'wet_dry2', 'rosemary', 'applewood1', 'applewood2', 'extinguish'};

% mkdir(LWfrmfldr);
% mkdir(MWfrmfldr);
read_attempt = 0;

%% Video Processing
for i=1:1:numel(vidfiles)
    vidtic = tic;
    disp(['Opening Video : ' vidfiles{i}]);
    % Update file names & locations
    vidname = vidfiles{i};
    vidlabel = vidlabels{i};
    vidpath = [vidfldr vidname];
    LWprefix = ['CLWIR_' vidlabel '_'];
    MWprefix = ['CMWIR_' vidlabel '_'];
    LWsavefldr = [frmfldr LWprefix vidlabel '/'];
    MWsavefldr = [frmfldr MWprefix vidlabel '/'];

    mkdir(LWsavefldr);
    mkdir(MWsavefldr);
    fid = fopen(vidpath,'r');
    disp(['Extracting frames...'])
    % Extract successive frames from each raw video stream until eof
    while ~feof(fid)
        % Read in raw video frame of 1280x480 = 614400 16-bit pixels
        dualfrm = uint16(fread(fid,[(MWcols+LWcols), nrows],'uint16'));
        if ~isempty(dualfrm)
            frmnum = frmnum + 1;
            % Skip every other frame (ie. move file position 1228800 bytes)
            if t <= 1
                % this is because at 60 fps (each band at 30) the new frame is
                % captured on left side while the right still has the previous
                % frame, then when the new right side frame is capture the
                % 2 are in sync. Thus there is an 'inbetween' sample (@60fps)
                % where the 2 frames are missaligned time-wise. For Example:
                %
                % t  lw  mw
                % --------------
                % t0  1  1
                % t(1/60)  2  1 (skip)
                % t(2/60)  2  2
                % t(3/60)  3  2 (skip)
            end
        end
    end
end
fseek(fid,2*(MWcols+LWcols)*nrows,'cof');

LWframe = dualfrm(1:640,:)./4;
MWframe = dualfrm(641:1280,:)./4;

LWsavepath = [LWsavefldr LWprefix int2str(frmnum) '.raw'];
MWsavepath = [MWsavefldr MWprefix int2str(frmnum) '.raw'];

LWfsav = fopen(LWsavepath,'w'); % open for write access
MWfsav = fopen(MWsavepath,'w'); % open for write access
fwrite(LWfsav,LWframe(:,:),'uint16'); % write frame 16-bit data
fwrite(MWfsav,MWframe(:,:),'uint16'); % write frame 16-bit data
fclose(LWfsav);
fclose(MWfsav);

disp(['Frames Saved : ' num2str(frmnum)]);
disp(['Video length : ' num2str(frmnum/fps) 's']);
disp(['Extract time : ' num2str(toc(vidtic)) 's']);
fclose(fid);
frmnum = 0;
end

end
function [framemat, framenum] = readframes(framefldr, bandname, testname, suffix, ...
    firstframe, frameinterval, lastframe, ...
    ncols, nrows)

%% Matlab Function -------------------------------------------%
% function [framemat, framenum, frames] = readframes(framefldr, bandname, testname, suffix ...
%     firstframe, frameinterval, lastframe ...
%     ncols, nrows)
% Function    : readframes.m
% Author      : Tim Davenport
% Date        : 10/3/11
% Description : Reads frames from folder and outputs 640x480xframenumber matrix of frames per frameinterval
% Inputs      : framefldr     = path of data folder ('string')
%               testname      = fire test name ('string')
%               bandname      = IR band ('string')
%               firstframe    = # of frame to start the read
%               frameinterval = # of frames between each read
%               lastframe     = # of frame to end the read
%               ncols         = # of pixel columns
%               nrows         = # of pixel rows
% Outputs     : framemat      = 3D matrix of frames by time (video)
%               framenum      = total # of frames read
% Example     : frmfldr  = '/Volumes/Macintosh HD 2/Thesis';
%               band     = 'CMWIR';
%               test     = 'rosemary';
%               [frames] = readframes(frmfldr, band, test, 1, 10, 30, 640, 480);
% Notes       : This function assumes the following folder hierarchy:
%                   /.../Frame folder/ (Eg. Extracted Frames)
%                       band1/ (Eg. UCLWIR)
%                           test1/ (Eg. palm)
%                                   band1_test1_1.raw (UCLWIR_palm_1.raw)
%                                   band1_test1_2.raw
%                                   band1_test1_3.raw
%                           ...
%                           test2/ (Eg. rosemary)
%                           test2/ (Eg. pine_cones)
%                           ...
%                       band2/ (Eg. CMWIR)
%                       band3/ ...
% Revision(s) : 11/08/11 - replaced strcat() with ['a ' 'b']
%                   - changed order of outputs
%                   - added support for band name input
%                   - added support for 'end' lastframe input
%                   11/11/11 - corrected isnumeric calculation
%                   - moved uint16 casting to per frame while reading
%                   - in instead of on whole matrix after (faster?)
%                   - added support for decimal frame interval
%                   01/31/12 - cleaned up syntax, variable names, and comments
%-------------------------------------------------------------------
% optional: tictime = tic;
%---------------------------------------------------------------

%% Calculate number of frames
if ~isnumeric(lastframe) % if 'end' grab till end of test
    if strcmp(bandname, 'VISIBLE')
lastframe = numel(dir([framefldr testname '/*.bmp'])); % use \ if WIN
   else
    lastframe = numel(dir([framefldr testname '/*.raw'])); % use \ if WIN
   end
end
framenum = floor((lastframe - firstframe + 1)/frameinterval);
if mod((lastframe - firstframe + 1),frameinterval) ~= 0;
   framenum = framenum + 1;
end

%% Initialize & Allocate
currentframe = firstframe;
framecnt = 0;
if strcmp(bandname,'VISIBLE')
   framemat = zeros(nrows, ncols,3, framenum);
else
   framemat = zeros(nrows, ncols,1, framenum);
end
fldr = [framefldr '/' bandname '/' testname '/']; % use \ if WIN
prefix = [bandname '_' testname '_'];

%% Open Each frame raw file and store in frame matrix
while currentframe <= lastframe
   if strcmp(bandname,'VISIBLE')
      filepath = [fldr prefix num2str(round(currentframe)) sufix '.bmp'];
      framecnt = framecnt + 1;
      framemat(:,:,framecnt) = uint8(imread(filepath, 'bmp'));
      currentframe = currentframe + frameinterval;
   else
      filepath = [fldr prefix num2str(round(currentframe)) sufix '.raw'];
      fid = fopen(filepath, 'r');
      framecnt = framecnt + 1;
      framemat(:,:,framecnt) = uint16(fread(fid,[ncols,nrows],'uint16'));
      currentframe = currentframe + frameinterval;
      fclose(fid);
   end
end

%% Error Notification
if framenum ~= framecnt
   disp('warning! : framenum - framecnt mismatch');
   framenum = framecnt;
end

%% End of Function

%optional: toctime = toc(tictime);
%------------------------------------------------------------------------
%% Matlab Script -----------------------------------------------
% Script      : alignUCLWIRFOV.m
% Author      : Tim Davenport
% Date        : 01/29/2012
% Description : spatially aligns all camera fields of view
% Note        :
% Revision(s) :
%---------------------------------------------------------------

close all; clear; tic; tic = tic;

%% User Input: Source & Destination directory locations
% Source location: raw un registered frame files
unRegfrmfldr = './Volumes/Macintosh HD/Thesis/';
% Target location: raw frames will be stored here
Regfrmfldr = './Volumes/FreeAgent GoFlex Drive/EFFD/Spatially Alligned/';

%% Initialize
band = 'UCLWIR';
nIRcols = 640;
nIRrows = 480;
nVISrows = 873;
nVIScols = 1068;
frmnum = 0;

vidlabels = { 'setup1'
               'charcoal1'
               'oats1'
               'oats2'
               'oats3'
               'pine_needles' redo
               'pine_cones'
               'palm'
               'thistle' redo
               'wet_leaves' redo
               'wet_dry1'
               'wet_dry2'
               'rosemary'
               'applewood1'
               'applewood2'
               'extinguish'};

%% Generate Transformation coefficients
VIS_x = [464 542 573 608 662 728 130 215 914 849 63 418 700 253 14 1030 863 541 609 331 ];
VIS_y = [577 573 571 566 291 639 625 157 146 751 721 846 801 433 805 646 565 436 685 328 ];
n = length (VIS_x);
A = ones ([n 3]);
A(:,1) = VIS_x;
A(:,2) = VIS_y;
ucIR_x = [303 322 330 339 353 369 219 242 415 398 203 291 361 250 190 442 300 320 338 270 ];
ucIR_y = [331 329 328 327 258 347 342 225 222 376 367 401 392 294 390 349 324 294 358 267 ];
abc_IR = A\ucIR_x;
def_IR = A\ucIR_y;

%% Transform frames
% create a reference grid where integer intersections represent the
% locations of pixels (pixel centers?) in the visible frame
[visible x visible y] = meshgrid (1:nVIScols, 1:nVISrows);

% create IR grids to sample from
[ir_x ir_y] = meshgrid (1:nIRcols, 1:nIRrows);

for i = 1:numel(vidlabels)                  % step thru each test raw frames
    aligntic = tic;
    disp(['Aligning Frames from : ' vidlabels(i) ' test']);
    % Update file names & locations
    testname = vidlabels(i);
    unRegPath = [unRegfrmfldr band '/' testname '/'];
    prefix = [band '_' testname '_'];
    frmnum = numel(dir([unRegPath '*.raw']));
    savefldr = [Regfrmfldr '/' band '/' testname '/'];
    mkdir(savefldr); % make folder for each video (or test)

    % Align each frame within test video
    h = waitbar(0,['Aligning frame FOVs for uclwir ' testname ' test']);
    for k = 1:frmnum
        % Read in raw video frame of 640x480 = 307200 16-bit pixels
        fid = fopen([unRegPath prefix num2str(k) '.raw'], 'r');
        unRegfrm = uint16(fread(fid,[nIRcols, nIRrows],'uint16'))';
        fclose(fid);

        % create grids that will store the locations in the IR frames where
        % pixels in the visible image lie. Start with an empty array, that is the
        % size of the visible image, and then fill it up pixel-by-pixel, using
        % [a b c] and [d e f] transformations calculated above
        resmpl_IR_x = zeros (nVISrows, nVIScols);
        resmpl_IR_y = zeros (nVISrows, nVIScols);

        % each point (visible x, visible y) in the visible image
        % corresponds to
        % a location in an IR image, here called (resmpl_IR_x, resmpl_IR_y).
        % The re-sample locations in each of the IR frames can be found
        using the
        % [a b c] and [d e f] transformations
        for jCol = 1:nVIScols
            for iRow = 1:nVISrows
                resmpl_IR_x(iRow,jCol) = abc_IR(1)*visible_x(iRow, jCol)...
                + abc_IR(2)*visible_y(iRow, jCol) + abc_IR(3); % resmpl_IR_x
                resmpl_IR_y(iRow,jCol) = def_IR(1)*visible_x(iRow, jCol)...
                + def_IR(2)*visible_y(iRow, jCol) + def_IR(3); % resmpl_IR_y
            end
        end

        % resample the IR images on the new grids.
        Regfrm = uint16 (interp2 (ir_x, ir_y, double (unRegfrm), ...
            resmpl_IR_x, resmpl_IR_y, 'bicubic'));

        % save new registered frame
        fid = fopen([savefldr prefix num2str(k) ' fov.raw'], 'w');
        fwrite(fid,Regfrm(:,:),'uint16');
        fclose(fid);

        waitbar(k/frmnum)
    end
end
close(h)
frmnum = 0;
disp(['Frames Registered : ' num2str(k)]);
disp(['Extract time : ' num2str(toc(aligntic)) ' s']);
end
%
%% End of Function
-------------------------------------------------------------------------

% toctime = toc(tic)
------------------------------------------------------------------------
%% Matlab Script -----------------------------------------------
% Script     : processOats.m
% Author      : Tim Davenport
% Date        : 01/31/2012
% Description : this script takes all registered spectral frames,  
                subsamples them, and stores them in a large .mat file
% Notes       : alter this instead of making new
% Revision(s) : n/a
%---------------------------------------------------------------------
% close all, clear, clc, tictime = tic;
%---------------------------------------------------------------------

%% Initialize and Allocate
nRowsRead = 873;
ncolsRead = 1068;
grabBigRow = 51:51+767;
grabBigCol = 151:918;
nRows = numel(grabBigRow);
nCols = numel(grabBigCol);
grabRow = 405:660;
grabc = 435:690;
nSmRows = numel(grabRow);
nSmCols = numel(grabc);
nPixel = nRows*nCols;
nPixSm = nSmRows*nSmCols;
nBands = 5;
frmfldr = '/Users/timothydavenport/Documents/MATLAB/data/EFFDdata/';

specname = {'Blue', 'Green', 'Red', 'MWIR', 'LWIR'};
LinLog = {'linear', 'linear', 'linear', 'log', 'log'};
test = 'oats1';
visFldrPrfx = 'VISIBLE/';
visprefix = 'VISIBLE_capture-13_*';
cmwirFldrPrfx = ['CMWIR/' test 'fov/' 'CMWIR_' test '_'];
clwirFldrPrfx = ['CLWIR/' test 'fov/' 'CLWIR_' test '_'];
% uclwirFldrPrfx = ['UCLWIR/' test '/' 'UCLWIR_' test '_'];
visExt = '.bmp';
irExt = '.raw';

IRfstep = 10;
VISfnums = round(linspace(7178, 8222, 460));
IRfnums = 10:IRfstep:4601;
uIRfnums = 1:IRfstep:4592;
Fnum = numel(IRfnums);

% allocated space - use for reading in data
maxPix = zeros(Fnum, nBands);
minPix = zeros(Fnum, nBands);
meanPix = zeros(Fnum, nBands);
medianPix = zeros(Fnum, nBands);
stdPix = zeros(Fnum, nBands);
BandCov = zeros(nBands, nBands, Fnum);

% maxPixSm = maxPix;
% minPixSm = minPix;
% meanPixSm = meanPix;
% medianPixSm = medianPix;
% stdPixSm = stdPix;
% BandCovSm = BandCov;
% maxPix = maxPix;

59
%% minPixZ = minPix;
%% meanPixZ = meanPix;
%% medianPixZ = medianPix;
%% stdPixZ = stdPix;
%% BandCovZ = BandCov;

%% maxPixSmZ = maxPix;
%% minPixSmZ = minPix;
%% meanPixSmZ = meanPix;
%% medianPixSmZ = medianPix;
%% stdPixSmZ = stdPix;
%% BandCovSmZ = BandCov;

%% SpecTime = zeros(nRows,nCols,nBands);
%% SpecTimeVec = zeros(nPixel,nBands,Fnum);
%% SpecTimeSm = zeros(nSmRows,nSmCols,nBands);
%% SpecTimeVecSm = zeros(nPixSm,nBands,Fnum);

%% Loop thru video
h = waitbar(0);
ofst = 199;
for f = 200:250
  % Read in and preprocess data - use load([frmfldr 'Oats1.mat']);
  {% Fvis = VISfnums(f);
   Fir = IRfnums(f);
   % Fuir = uIRfnums(f-ofst);
   %# read in frames from all spectrum
   visible = imread([frmfldr visFldrPrfx visprefix num2str(Fvis) visExt]);
   fid = fopen([frmfldr cmwirFldrPrfx num2str(Fir) ' fov' irExt],'r');
   cmwir = uint16(fread(fid,[nColsRead nRowsRead],'uint16'))';
   fclose(fid);
   fid = fopen([frmfldr clwirFldrPrfx num2str(Fir) ' fov' irExt],'r');
   clwir = uint16(fread(fid,[nColsRead nRowsRead],'uint16'))';
   fclose(fid);
   fid = fopen([frmfldr uclwirFldrPrfx num2str(Fuir) '_fov' irExt],'r');
   uclwir = uint16(fread(fid,[nColsRead nRowsRead],'uint16'))';
   fclose(fid);
   % c = [imadjust(cmwir(1:826,:));cmwir(827:end,:)];
   c = double(c)./double(max(max(c)));
   v = double(visible)./255;
   figure(1)
   imshow(cat(3,c,0.5.*v(:,:,[1 3])))
   %# sample subimage
   visibleSm = double(visible(grabRow,grabCol,:));
   cmwirSm = double(cmwir(grabRow,grabCol));
   clwirSm = double(clwir(grabRow,grabCol));
   uclwirSm = double(uclwir(grabRow,grabCol));

   visible = double(visible(grabBigRow,grabBigCol,:));
   cmwir = double(cmwir(grabBigRow,grabBigCol));
   clwir = double(clwir(grabBigRow,grabBigCol));
   uclwir = double(uclwir(1:nRows,1:nCols));

   SpecTime = cat(3,visible,cmwir,clwir);
   SpecTimeVec(:,:,f-ofst) = reshape(SpecTime,nPixel,nBands);
   SpecTimeSm = cat(3,visibleSm,cmwirSm,clwirSm);
   SpecTimeVecSm(:,:,f-ofst-250) = reshape(SpecTimeSm,nPixSm,nBands);
   SpecTimeVecZ = zscore(SpecTimeVec);
SpecTimeVecSmZ = zscore(SpecTimeVecSm);

% gather statistics - use load Oats1SpectrumStats
maxPix(f-ofst,:) = max(SpecTimeVec(:,:,f-ofst));
minPix(f-ofst,:) = min(SpecTimeVec(:,:,f-ofst));
meanPix(f-ofst,:) = mean(SpecTimeVec(:,:,f-ofst));
medianPix(f-ofst,:) = median(SpecTimeVec(:,:,f-ofst));
stdPix(f-ofst,:) = std(SpecTimeVec(:,:,f-ofst));
BandCov(:,:,f-ofst) = cov(SpecTimeVec(:,:,f-ofst));

maxPixSm(f-ofst,:) = max(SpecTimeVecSm(:,:,f-ofst));
minPixSm(f-ofst,:) = min(SpecTimeVecSm(:,:,f-ofst));
meanPixSm(f-ofst,:) = mean(SpecTimeVecSm(:,:,f-ofst));
medianPixSm(f-ofst,:) = median(SpecTimeVecSm(:,:,f-ofst));
stdPixSm(f-ofst,:) = std(SpecTimeVecSm(:,:,f-ofst));
BandCovSm(:,:,f-ofst) = cov(SpecTimeVecSm(:,:,f-ofst));

maxPixZ(f-ofst,:) = max(SpecTimeVecZ(:,:,f-ofst));
minPixZ(f-ofst,:) = min(SpecTimeVecZ(:,:,f-ofst));
meanPixZ(f-ofst,:) = mean(SpecTimeVecZ(:,:,f-ofst));
medianPixZ(f-ofst,:) = median(SpecTimeVecZ(:,:,f-ofst));
stdPixZ(f-ofst,:) = std(SpecTimeVecZ(:,:,f-ofst));
BandCovZ(:,:,f-ofst) = cov(SpecTimeVecZ(:,:,f-ofst));

maxPixSmZ(f-ofst,:) = max(SpecTimeVecSmZ(:,:,f-ofst));
minPixSmZ(f-ofst,:) = min(SpecTimeVecSmZ(:,:,f-ofst));
meanPixSmZ(f-ofst,:) = mean(SpecTimeVecSmZ(:,:,f-ofst));
medianPixSmZ(f-ofst,:) = median(SpecTimeVecSmZ(:,:,f-ofst));
stdPixSmZ(f-ofst,:) = std(SpecTimeVecSmZ(:,:,f-ofst));
BandCovSmZ(:,:,f-ofst) = cov(SpecTimeVecSmZ(:,:,f-ofst));

waitbar((f-ofst)/Fnum,h,[num2str(f-ofst)' of ' num2str(Fnum)])
end

Oatmax = max(maxPix)
Oatmin = min(minPix)
Oatmean = mean(meanPix)
Oatmedian = median(medianPix);
Oatstd = std(stdPix);

OatmaxSm = max(maxPixSm);
OatminSm = min(minPixSm);
OatmeanSm = mean(meanPixSm);
OatmedianSm = median(medianPixSm);
OatstdSm = std(stdPixSm);

OatmaxZ = max(maxPixZ);
OatminZ = min(minPixZ);
OatmeanZ = mean(meanPixZ);
OatmedianZ = median(medianPixZ);
OatstdZ = std(stdPixZ);

OatmaxSmZ = max(maxPixSmZ);
OatminSmZ = min(minPixSmZ);
OatmeanSmZ = mean(meanPixSmZ);
OatmedianSmZ = median(medianPixSmZ);
OatstdSmZ = std(stdPixSmZ);

close(h)
clear SpecTime SpecTimeVecZ
save(['Oats1_full_200.mat'],'SpecTime*','**Pix','-v7.3')

end
function [fprincompT fPCT ev C] = fPCA(I)

%% Matlab Script
%---------------------------------------------------------
% Script : fPCA.m
% Author : Tim Davenport
% Date : 01/31/2012

%% Description : this calculates the PCA transformation matrix
% Notes : alter this instead of making new
% Revision(s) : n/a
%---------------------------------------------------------

for i = 1:size(I,3)
    fprincompT(:,:,i) = princomp(I(:,:,i));
    C = cov(I(:,:,i));
    [fPCT(:,:,i) eigval] = eig(cov(I(:,:,i)));
    [ev ix] = sort(diag(eigval),'descend');
    fPCT(:,:,i) = fPCT(:,ix,i);
end

%% End of Function
---------------------------------------------------------
end

end

-------------------------------------------------------------------------
function PCvec = PCT(I,T,type)
%% Matlab Script ---------------------------------------------%%
% Script      : PCT.m
% Author      : Tim Davenport
% Date        : 01/31/2012
% Description : this function transforms data with a PCA transformation
% matrix.
% Notes       : alter this instead of making new
% Revision(s) : n/a
%E------%  E-----%  
if numel(size(T)) > 2
  if strcmp(type,'f') % spectrum PCT (for each time)
    PCvec = zeros(size(I));
    for i = 1:size(I,3)
      PCvec(:,:,i) = (T(:,:,i)'*squeeze(I(:,:,i))')';
    end
  else% time PCT (for each spectrum)
    PCvec = zeros(size(I,1),size(I,3),size(I,2));
    for i = 1:size(I,2)
      PCvec(:,:,i) = (T(:,:,i)'*squeeze(I(:,i,:))')';
    end
  end
else % handles spectral (1 sample in time) & f/t
  I = reshape(I,size(I,1),size(I,2)*size(I,3));
  PCvec = (T'*I')';
end
%% End of Function ---------------------------------------------%
function Imapped = map8(In,nl,type)

    Imapped = uint8(zeros(size(In))); % Imapped = uint8(zeros(size(In)));
    for i = 1:size(In,3)
        I = In(:,:,i);
        if strcmp(type,'negpos')
            Imapped(:,:,i) = uint8( 127 + 128.*(sign(I).*abs(I).^nl)./repmat(max(abs(I).^nl),[size(I,1) 1]));
            %                      ./ (max(max(abs(prctile(I,[1 99]).^nl))));
        elseif strcmp(type,'mag')
            Imapped(:,:,i) = uint8( 256.*(abs(I).^nl ) ./ (max(max(abs(prctile(I,[1 99])).^nl))));
        end
    end

%% End of Function

end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
function Imapped = mapNL(In,nl)
%% Matlab Script
%---------------------------------------------------------
% Script      : mapNL.m
% Author      : Tim Davenport
% Date        : 01/31/2012
% Description : this function nonlinearly maps data.
% Notes       : alter this instead of making new
% Revision(s) : n/a
%---------------------------------------------------------
% Imapped = double(zeros(size(In)));

for i = 1:size(In,3)
  I = In(:,:,i);
  for j = 1:size(I,2)
    if max(I(:,j)) > 256
      if mod(j,4) == 0
        Imapped(:,j,i) = 255.*((I(:,j) - min(min(I(:,j))))
                                  ./max(max(I(:,j)) - min(min(I(:,j))))).^nl(1);
        d = 1;
      else
        Imapped(:,j,i) = 255.*((I(:,j) - min(min(I(:,j))))
                                  ./max(max(I(:,j)) - min(min(I(:,j))))).^nl(2);
        d = 1;
      end
    else
      Imapped(:,j,i) = In(:,j,i);
    end
  end
end

%% End of Function
%---------------------------------------------------------
end
%---------------------------------------------------------
function img = makeImg(I,nRows,nCols,type,outd)
% Matlab Script
% Script      : makeImg.m
% Author      : Tim Davenport
% Date        : 01/31/2012
% Description : this function makes images from data for a certain format
% Notes       : alter this instead of making new
% Revision(s) : n/a
%-------------------------------------------------------------------------
if strcmp(type,'square')
    img = reshape(I,size(I,1),size(I,2)*size(I,4));
elseif strcmp(type,'tpca') & numel(size(I)) > 2
    img = reshape(I(:,1,:),nRows,nCols*size(I,3));
    for i = 2:size(I,2)
        img = [img ; reshape(I(:,i,:),nRows,nCols*size(I,3))];
    end
elseif strcmp(type,'tpcaSq-')
    img = [];
    iter = 1;
    l = ones(nRows,5);
    for j = 1:outd(1)
        imgR = [l];
        for i = 1:outd(2)
            if I(:,iter) > 256
                imgR = [ imgR l imadjust(reshape(I(:,iter)/2^24,nRows,nCols)) ];
            else
                imgR = [ imgR l (reshape(I(:,iter)/2^8,nRows,nCols)) ];
            end
        end
        img = [img ; q ; imgR ];
    end
else
    img = [img ; q ];
end
elseif strcmp(type,'tpcaSqV-')
    img = [];
    iter = 1;
    l = ones(nRows,5,3);
    for j = 1:outd(1)
        imgR = [l ];
        for i = 1:outd(2)
            imgR = [ imgR l (reshape(I(:,iter:iter+2)/2^8,nRows,nCols,3)) ];
        end
    end
else
    img = [img ; q ];
end
elseif strcmp(type,'tpcaSq')
    img = [];
    iter = 1;
    l = 255*ones(nRows,2);
    for j = 1:outd(1)
        imgR = [l ];
        for i = 1:outd(2)
            imgR = [ imgR l (reshape(I(:,iter),nRows,nCols)) ];
        end
    end
else
    img = [img ; q ];
end
iter = iter + 1;
imgR = [imgR 1];
q = 255*ones(2, size(imgR, 2));
img = [img ; q; imgR];
end
img = [img ; q];
elseif strcmp(type, 'tpca-')
  img = reshape(I(:,1,:), nRows, nCols*size(I, 3));
  for i = 2:size(I, 2)
    img = [img ; reshape(I(:,i,:), nRows, nCols*size(I, 3))];
  end
elseif strcmp(type, 'tpca')
  img = reshape(I(:,1,:), nRows, nCols*size(I, 3));
  for i = 2:size(I, 2)
    img = [img ; reshape(I(:,i,:), nRows, nCols*size(I, 3))];
  end
end
elseif strcmp(type, 'vml')
  if max(I(:,:,1)) > 1
    I = [I(:,:,1,:)/2^8 I(:,:,4:5,:)/2^14];
  end
  img = [reshape(I(:,1:3,:), nRows, nCols), ...]
        repmat(imadjust(reshape(I(:,4,1), nRows, nCols)), [1 1 3]) ...]
        repmat(imadjust(reshape(I(:,5,1), nRows, nCols)), [1 1 3]);
  for i = 2:size(I, 3)
    img = [img ; reshape(I(:,1:3,i), nRows, nCols), ...]
        repmat(imadjust(reshape(I(:,4,i), nRows, nCols)), [1 1 3]) ...]
        repmat(imadjust(reshape(I(:,5,i), nRows, nCols)), [1 1 3]);
  end
elseif strcmp(type, 'gml')
  if max(I(:,:,1)) > 1
    I = [I(:,:,1,:)/2^8 I(:,:,2:3,:)/2^14];
  end
  img = [reshape(I(:,1,1), nRows, nCols) ...]
        imadjust(reshape(I(:,2,1), nRows, nCols)) ...]
        imadjust(reshape(I(:,3,1), nRows, nCols));
  for i = 2:size(I, 3)
    img = [img ; reshape(I(:,1,i), nRows, nCols) ...]
        imadjust(reshape(I(:,2,i), nRows, nCols)) ...]
        imadjust(reshape(I(:,3,i), nRows, nCols));
  end
elseif strcmp(type, 'rgml')
  if max(I(:,:,1)) > 1
    I = [I(:,:,1,:)/2^8 I(:,:,3:4,:)/2^14];
  end
  img = [reshape(I(:,1:2,:), nRows, nCols*2) ...]
        imadjust(reshape(I(:,3,1), nRows, nCols)) ...]
        imadjust(reshape(I(:,4,1), nRows, nCols));
  for i = 2:size(I, 3)
    img = [img ; reshape(I(:,1:2,i), nRows, nCols*2) ...]
        imadjust(reshape(I(:,3,i), nRows, nCols)) ...]
        imadjust(reshape(I(:,4,i), nRows, nCols));
  end
elseif strcmp(type, 'rgbml')
  if max(I(:,:,1)) > 1
    I = [I(:,:,1,:)/2^8 I(:,:,4:5,:)/2^14];
  end
  img = [reshape(I(:,1:3,:), nRows, nCols*3) ...]
        imadjust(reshape(I(:,4,1), nRows, nCols)) ...]
        imadjust(reshape(I(:,5,1), nRows, nCols));
  for i = 2:size(I, 3)
    img = [img ; reshape(I(:,1:3,i), nRows, nCols*3) ...]
        imadjust(reshape(I(:,4,i), nRows, nCols)) ...]
        imadjust(reshape(I(:,5,i), nRows, nCols));
  end
elseif strcmp(type, 'rgbml')
  if max(I(:,:,1)) > 1
    I = [I(:,:,1,:)/2^8 I(:,:,4:5,:)/2^14];
  end
  img = [reshape(I(:,1:3,:), nRows, nCols*3) ...]
        imadjust(reshape(I(:,4,1), nRows, nCols)) ...]
        imadjust(reshape(I(:,5,1), nRows, nCols));
  for i = 2:size(I, 3)
    img = [img ; reshape(I(:,1:3,i), nRows, nCols*3) ...]
        imadjust(reshape(I(:,4,i), nRows, nCols)) ...]
        imadjust(reshape(I(:,5,i), nRows, nCols));
  end
elseif strcmp(type, 'rgbml_')
  if max(I(:,:,1)) > 1
    I = [I(:,:,1,:)/2^8 I(:,:,4:5,:)/2^8];
  end
  img = [reshape(I(:,1:3,:), nRows, nCols*3) ...]
        imadjust(reshape(I(:,4,1), nRows, nCols)) ...]
        imadjust(reshape(I(:,5,1), nRows, nCols));
  for i = 2:size(I, 3)
    img = [img ; reshape(I(:,1:3,i), nRows, nCols*3) ...]
        imadjust(reshape(I(:,4,i), nRows, nCols)) ...]
        imadjust(reshape(I(:,5,i), nRows, nCols));
  end
(reshape(I(:,5,1),nRows,nCols));
for i = 2:size(I,3)
    img = [img ; reshape(I(:,1:3,i),nRows,nCols*3)
          reshape(I(:,4,i),nRows,nCols)
          reshape(I(:,5,i),nRows,nCols))];
end
elseif strcmp(type,'rgbm')
    if max(I(:,1,:)) > 1
        I = [I(:,1:3,:)./2^8 I(:,4,:)./2^14];
    end
    img = [reshape(I(:,1:3,1),nRows,nCols*3)
           imadjust(reshape(I(:,4,1),nRows,nCols))];
    for i = 2:size(I,3)
        img = [img ; reshape(I(:,1:3,i),nRows,nCols*3)
               imadjust(reshape(I(:,4,i),nRows,nCols))];
    end
elseif strcmp(type,'gm')
    if max(I(:,:,1)) > 1
        I = [I(:,1,:)./2^8 I(:,2,:)./2^14];
    end
    img = [reshape(I(:,1,1),nRows,nCols)
           imadjust(reshape(I(:,2,1),nRows,nCols))];
    for i = 2:size(I,3)
        img = [img ; reshape(I(:,1,i),nRows,nCols)
               imadjust(reshape(I(:,2,i),nRows,nCols))];
    end
elseif strcmp(type,'v')
    if size(I,2) > 3
        I = I(:,1:3,:);
    end
    if max(I(:,1,:)) > 1
        I = I(:,1,:)./2^8;
    end
    img = reshape(I(:,1:3,1),nRows,nCols,3);
    for i = 2:size(I,3)
        img = [img ; reshape(I(:,1:3,i),nRows,nCols,3)];
    end
elseif strcmp(type,'rgb')
    if size(I,2) > 3
        I = I(:,1:3,:);
    end
    if max(I(:,1,:)) > 1
        I = I(:,1,:)./2^8;
    end
    img = reshape(I(:,1,1),nRows,nCols*size(I,2));
    for i = 2:size(I,3)
        img = [img ; reshape(I(:,1,i),nRows,nCols*size(I,2))];
    end
elseif strcmp(type,'rgb-')
    if size(I,2) > 3
        I = I(:,1:3,:);
    end
    if max(I(:,1,:)) > 1
        I = I(:,1,:);
    end
    img = reshape(I(:,1,:),nRows,nCols*size(I,3));
elseif strcmp(type,'ml')
    if size(I,2) > 2
        I = I(:,4:6,:);
    end
    if max(I(:,1,:)) > 1
        I = I(:,1,:)/2^24;
    end
    img = [];
    for i = 1:size(I,2)
        img = [img imadjust(reshape(I(:,i,1),nRows,nCols))];
    end
end
end
for i = 2:size(I,3)
    r = [];
    for j = 1:size(I,2)
        r = [r imadjust(reshape(I(:,j,i),nRows,nCols))];
    end
    img = [img ; r];
end
elseif strcmp(type,'ml-')
    if size(I,2) > 2
        I = I(:,4:6,:);
    end
    if max(I(:,:,1)) > 1
        I = I(:,:,1)./2^14;
    end
    img = [];
    for i = 1:size(I,3)
        img = [img imadjust(reshape(I(:,1,i),nRows,nCols))];
    end
    for i = 2:size(I,2)
        r = [];
        for j = 1:size(I,3)
            r = [r imadjust(reshape(I(:,j,i),nRows,nCols))];
        end
        img = [img ; r];
end
end

%% End of Function
% Script      : stPCA GLCM.m
% Author      : Tim Davenport
% Date        : 01/31/2012
% Description : this script performs PCA and Texture Analysis on the variable SpecTimeVec
% Notes       : alter this instead of making new
% Revision(s) : n/a

% Initialize and Allocate
load cntrsKeep2
nRows = 768;
ncols = 768;
nPix = nRows*nCols;
nBand = 5;
ffire = 50+15; % was 15
ffire2 = 30;
gamma = 1;
IRgamma = [0.51 0.51];
Bname = {'red' 'green' 'blue' 'mwir' 'lwir'};
% bands = 1:3;
nPCs = numel(bands);
times = 3;
tnum = 2;
Havg = (tnum^-3).*ones(3,3,tnum);
tlength = 9;
pcaStep = 3;

it = 1;
jt = 1;
nband = numel(bands);
figure(1)
set(gcf,'position',[0 64 1280 1280]);
step = 6;
Num = 3;
ft = fliplr(50+63:-step:(50+63+1)-Num*step);
smokePC = [];
nft = numel(ft);
NUM = nband*nft;
testnl = mapNL(reshape(SpecTimeVec(:,bands,ft),nPix,NUM),IRgamma);
% testnlBUS =
mapNL(reshape(SpecTimeVec(:,bands,ft+112),nPix,NUM),IRgamma);
T = testnl; % SpecTimeVec(:,bands,ffire);
ir = randsample(NUM,NUM);
% TBUS = testnlBUS;
Z = (zscore(T));
% ZBUS = (zscore(TBUS));
% Z(isnan(Z)) = 0;
% Z(isinf(Z)) = 0;
\[ f_{Tz}, D \] = fPCA(Z);

% figure(1)
% for i = 1:NUM
% subplot(NUM,1,i), bar(abs(fTz(:,i)))
% end

% PC1(it,:) = fTz(:,1);
% PC2(it,:) = fTz(2,2);
% PC3(it,:) = fTz(3,3);
% PC4(it,:) = fTz(4,4);
% PC5(it,:) = fTz(5,5);

% figure(1)
% subplot 551, plot(0:0.01:2,abs(PC1(:,1))),ylabel('PC1')
% subplot 552, plot(0:0.01:2,abs(PC1(:,2)))
% subplot 553, plot(0:0.01:2,abs(PC1(:,3)))
% subplot 554, plot(0:0.01:2,abs(PC1(:,4)))
% subplot 555, plot(0:0.01:2,abs(PC1(:,5)))
% subplot(5,5,6), plot(0:0.01:2,abs(PC2(:,1))),ylabel('PC2')
% subplot(5,5,7), plot(0:0.01:2,abs(PC2(:,2)))
% subplot(5,5,8), plot(0:0.01:2,abs(PC2(:,3)))
% subplot(5,5,9), plot(0:0.01:2,abs(PC2(:,4)))
% subplot(5,5,10), plot(0:0.01:2,abs(PC2(:,5)))
% subplot(5,5,11), plot(0:0.01:2,abs(PC3(:,1))),ylabel('PC3')
% subplot(5,5,12), plot(0:0.01:2,abs(PC3(:,2)))
% subplot(5,5,13), plot(0:0.01:2,abs(PC3(:,3)))
% subplot(5,5,14), plot(0:0.01:2,abs(PC3(:,4)))
% subplot(5,5,15), plot(0:0.01:2,abs(PC3(:,5)))
% subplot(5,5,16), plot(0:0.01:2,abs(PC4(:,1))),ylabel('PC4')
% subplot(5,5,17), plot(0:0.01:2,abs(PC4(:,2)))
% subplot(5,5,18), plot(0:0.01:2,abs(PC4(:,3)))
% subplot(5,5,19), plot(0:0.01:2,abs(PC4(:,4)))
% subplot(5,5,20), plot(0:0.01:2,abs(PC4(:,5)))
% subplot(5,5,21), plot(0:0.01:2,abs(PC5(:,1))),xlabel('Red'),ylabel('PC5')
% subplot(5,5,22), plot(0:0.01:2,abs(PC5(:,2))),xlabel('Green')
% subplot(5,5,23), plot(0:0.01:2,abs(PC5(:,3))),xlabel('Blue')
% subplot(5,5,24), plot(0:0.01:2,abs(PC5(:,4))),xlabel('MWIR')
% subplot(5,5,25), plot(0:0.01:2,abs(PC5(:,5))),xlabel('LWIR')

fPCvecz = PCT(Z,fTz);
fPCvecNLz = map8(fPCvecz,gamma,'negpos');

PCimg = makeImg(fPCvecNLz,nRows,nCols,'tpcaSq',[nft nband]);

% imshow(PCimg)
% I = reshape(fPCvecNLz,nRows,nCols,NUM);
% GrayNeigh = [0 1; -1 1 ;-1 0; -1 -1].*10;
% GrayNeigh = [GrayNeigh; -2 -2 ;-2 2; -1 1 -1 1 -2 2 -2 2 -1 1 2 2; -2 0; 0 2];
% for i = 1:NUM
% GLCM = graycomatrix(I(:,:,smokePC),'NumLevels',256,'Offset',GrayNeigh);
% Gstats = graycoprops(GLCM);
% Con(it,jt) = sum(Gstats.Contrast);
% Cor(it,jt) = sum(Gstats.Corrlation);
% Enr(it,jt) = sum(Gstats.Energy);
% Hmo(it,jt) = sum(Gstats.Homogeneity);
% EntG(it,jt) = sum(entropy(GLCM));
% [~,Gp] = princomp(GLCM);
% G(:,i) = reshape(Ga,256*256,1);
% GLCM(:,i) = GLCM(:,);
% end
it = it + 1
jt = jt + 1
imshow(reshape(GLCM,256,256*size(GLCM,3)))

% GLCMimg = makeImg(GLCV,256,256,'tpcaSq',nfft,nband);
% GLCMimg = imresize(GLCMimg,size(PCimg));
% [h ax bigax p] = plotmatrix(fPCvecz,fPCvecz), title('PC Scatter Plot');
% fPCvecz = PCT(zscore(SpecTimeVec(:,bands,fFire)),fTz);
% delayimg = uint8( zeros(nRows,nPCs*nCols,tnum+1));
% sadf = 1234;
% generate Spectral Image Video
ImgVid = VideoWriter(['PC',datestr(now),'_avishi']);
ImgVid.FrameRate = 10;
open(ImgVid);

iter = 1;
Fnum = 250;
hs = waitbar(0);
ts =[];
clear Con Cor Enr Hmo CorIr
for f = 20:step:Fnum
    ftnow = fliplr( reshape(SpecTimeVec(:,bands,ftnow),nPix,NUM),IRgamma);
    visImg = makeImg(testnl(:,[1:3]),nRows,nCols,'v',[1 1]);
    % visImg = makeImg(testnl(:,1:5),nRows,nCols,'tpcaSq',[1 5]);
    fPCvecz = PCT(zscore(testnl),fTz);
    fPCvecNLz = map8(fPCvecz,gamma,'negpos');
    PCimg = makeImg(fPCvecNLz,nRows,nCols,'tpcaSq',[nft nband]);
    I = reshape(fPCvecNLz(:,4),nRows,nCols);
    % Iir = reshape(fPCvecNLz(:,34),nRows,nCols);
    % imshow([visImg; PCimg])
    figure(1)
    % imshow(PCimg)
    subplot 221, imshow(visImg); title('Visible','FontSize',24)
    subplot 222, imshow(I); title('PC5','FontSize',24)
    % GrayNeigh = [0 1; -1 1 ;-1 0; -1 -1].*10;
    % G = graycoprops(graycomatrix(I,'NumLevels',256,'Offset',GrayNeigh,'symmetric',true));
    % Gir = graycoprops(graycomatrix(Iir,'NumLevels',256,'Offset',GrayNeigh,'symmetric',true));
% Con(iter,:) = sum(Gir.Contrast);
Cor(iter,:) = (G.Correlation);
% CorIr(iter,:) = (Gir.Correlation);
% Enr(iter,:) = sum(Gir.Energy);
% Hmo(iter,:) = sum(Gir.Homogeneity);

% Con(iter,:) = sum(Gir.Contrast);
Cor(iter,:) = (G.Correlation);
% CorIr(iter,:) = (Gir.Correlation);
% Enr(iter,:) = sum(Gir.Energy);
% Hmo(iter,:) = sum(Gir.Homogeneity);

if iter == 25
    stop = 1234;
% end

% hold on
% subplot 221,plot(ts,Con),title('Contrast')
% subplot 222,plot(ts,Cor),title('Correlation')
% subplot 223,plot(ts,Enr),title('Energy')
% subplot 224,plot(ts,Hmo),title('Homogeneity')
% hold off

subplot(2,2,[3 4]), plot(ts,sum(Cor,2)), title('Correlation', 'FontSize',24)
set(gca,'FontSize',20);
xlabel('t_s (s)', 'FontSize',20), grid on
axis([0 2400 0 1])

if sum(Cor(iter,:),2) > 0.5
    subplot 221, RRRR = rectangle('position',[0,0,768,768], 'edgecolor','r', 'LineWidth',20);
    subplot(2,2,[3 4]), plot(ts,sum(Cor,2), 'r'),
    title('Correlation', 'FontSize',24)
    set(gca,'FontSize',20);
    xlabel('t_s (s)', 'FontSize',20), grid on, axis([0 2400 0 1])
    writeVideo(ImgVid, getframe(gcf));
    writeVideo(ImgVid, getframe(gcf));
    writeVideo(ImgVid, getframe(gcf));
    delete(RRRR)
end

iter = iter + 1;
writeVideo(ImgVid, getframe(gcf));
writeVideo(ImgVid, getframe(gcf));
writeVideo(ImgVid, getframe(gcf));
% waitbar(f/Fnum,h,[num2str(f) ' of ' num2str(Fnum)])
% end

close(ImgVid)
toc(tictime)