EARLY FOREST FIRE DETECTION VIA PRINCIPAL COMPONENT ANALYSIS OF SPECTRAL AND TEMPORAL SMOKE SIGNATURE

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

David C. Garges

June 2015

COMMITTEE MEMBERSHIP

TITLE:	Early Forest Fire Detection via Principal Component Analysis of Spectral and Temporal Smoke Signature
AUTHOR:	David C. Garges
DATE SUBMITTED:	June 2015
COMMITTEE CHAIR:	John Saghri, PhD Professor of Electrical Engineering Cal Poly
COMMITTEE MEMBER:	John Jacobs, PhD Raytheon Professor of Practice Cal Poly
COMMITTEE MEMBER:	Jane Zhang, PhD Associate Professor Cal Poly

ABSTRACT

Early Forest Fire Detection via Principal Component Analysis of
Spectral and Temporal Smoke Signature

David Garges

The goal of this study is to develop a smoke detecting algorithm using digital image processing techniques on multi-spectral (visible & infrared) video. By utilizing principal component analysis (PCA) followed by spatial filtering of principal component images the location of smoke can be accurately identified over a period of exposure time with a given frame capture rate. This result can be further analyzed with consideration of wind factor and fire detection range to determine if a fire is present within a scene.

Infrared spectral data is shown to contribute little information concerning the smoke signature. Moreover, finalized processing techniques are focused on the blue spectral band as it is furthest away from the infrared spectral bands and because it experimentally yields the largest footprint in the processed principal component images in comparison to other spectral bands. A frame rate of .5 images/sec (1 image every 2 seconds) is determined to be the maximum such that temporal variance of smoke can be captured. The study also shows eigenvectors corresponding to the principal components that best represent smoke and are valuable indications of smoke temporal signature.

Raw video data is taken through rigorous pre-processing schemes to align frames from respective spectral band both spatially and temporally. A multi-paradigm numerical computing program, MATLAB, is used to match the field of view across five spectral bands: Red, Green, Blue, Long-Wave Infrared, and Mid-Wave Infrared. Extracted frames are aligned temporally from key frames throughout the data capture. This alignment allows for more accurate digital processing for smoke signature.

Clustering analysis on RGB and HSV value systems reveal that color alone is not helpful to segment smoke. The feature values of trees and other false positives are shown to be too closely related to features of smoke for in solely one instance in time.

A temporal principal component transform on the blue spectral band eliminates static false positives and emphasizes the temporal variance of moving smoke in images with higher order. A threshold adjustment is applied to a blurred blue principal component of non-unity principal component order and smoke results can be finalized using median filtering. These same processing techniques are applied to difference images as a more simple and traditional technique for identifying temporal variance and results are compared.

Keywords: Image Processing, Multi-spectral Video, Principal Component Analysis (PCA), Temporal Variance, Feature Space, Median Filtering, Principal Component Images

ACKNOWLEDGMENTS

I would like to thank Professor Saghri, Dr. Jacobs, Professor Zhang and Tim Davenport for their advice, support and encouragement through the entirety of the project. I learned many skills and techniques towards approaching a big data project and this is something that I will carry with me for the remainder of my engineering career. I would also like to think Dr. Gary Hughes for his Matlab technical support, without him, I would probably still be importing a .raw image. Finally, I would like thank my family and friends as this project would not have been completed without their unwavering support and encouragement.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

1.1 Motivation

Forest fires and wildfires are a great risk to human lives and they often times cause great ecological and economical damage. Historically, humans visually scan forest environments from elevated towers to prepare a timely emergency response to any signs of a potential forest fire. This method susceptible to human error and requires constant human attendance to remote locations. An autonomous forest fire detection system on the other hand gives the ability to eliminate many labor difficulties and hazards resulting from human operated towers [**1**].

To combat these natural disasters cost effectively, it is vital to detect forest fires in the early stages of combustion [**2**]. On Average, the United States spends $900 million/year to combat forest fires [**3**] and the fires themselves cause $733 million/year in property and crop damage [**4**]. In light of these factors, there is much motivation to develop a system that can accurately identify the early stages of a forest fire, and trigger an immediate and cost-effective response.

1.2 Background

Conventional in-home smoke detectors use technology based on ionization and photometry to sense smoke particles in the air [**5**]. Successful operation of these detection systems relies on the device and sensors to be in the immediate vicinity of the smoke and fire. For this reason, the technological detection methods used by in-home smoke detectors are ineffective at detecting the presence of a forest fire at a far but observable distance.

Remote Sensing is an important field of engineering that deals with acquiring information about an object from afar without making physical contact with the object [**6**]. Currently, as an alternative to using human operated towers to catch wildfires in its early stages, satellites use remote sensing for larger scale fire detection [**7**], [**8**], and [**9**]. Unfortunately, the spatial resolution of satellite imagery is limited by temporal resolution and ground sampling distance. Other recent

developments in ground based Early Forest Fire Detection have been accomplished by the Croation iForestFire. In this system, a Web Information System relays a multiple vision systems to an operator who's task is to make a final decision on suspicious regions autonomously filtered by camera systems from and array of monitoring sites [**10**]. False alarm rates are reduced through the FAR system where information in the visible spectrum is compared to information in the infrared spectrum [**11**]. Although autonomous ground-based fire detection methods exist, there is much motivation and room for improvement in this important field of engineering.

1.3 Spectral and Temporal Smoke Signature

Using fire scene images from data collected at Raytheon in Goleta California, the spectral smoke signature is investigated. The location of "smoke" pixels will be examined in an image's feature space to see if the identification of an explicit smoke signature is possible. Furthermore, the work in [**12**] and [**13**] has shown that the use of principal component analysis (PCA) can accurately classify atmospheric aerosols from multispectral satellite data using MODIS (Moderate Resolution Imaging Spectroradiometer) radiance information and GOCART aerosol speciation. A delta detection method can also be accomplished with principal component analysis by using temporal data to classify a "change" in a scene between two images at different times. Moreover, the work in [**14**] and [**15**] has shown that the dispersion and flickering of smoke above a fire is sensitive to PCA. Since smoke detection using multi-temporal PCA requires a set of image as an input, there is motivation to investigate optimal frame rates and which subsequent order of PCA best emphasizes the movement of smoke through time. Selective threshold adjustment and other image processing techniques can ultimately separate the moving smoke from a stagnant background.

1.4 Proposed Early Forest Fire Detection

A land-based multi-spectral and multi-temporal video processing scheme is proposed to determine whether smoke, and thus fire, is present within a scene. The method will utilize

temporal, spatial, and spectral information from multiple camera views to segment and identify the smoke plume. Spectral features of the smoke are compared to the spectral features of other objects in the fire scene to investigate the uniqueness of the smoke's spectral signature. Ideally, if the feature location of a pixel is within the vicinity of the smoke signature, it can be identified as smoke with a certain amount of confidence depending on the uniqueness of its spectral features. Using sets of images through time, the temporal variance of the smoke is identified through simple difference images and principal component transformation; a more complicated technique that provides better results. Other digital image processing techniques are utilized throughout to enhance the outcomes of each technique.

1.5 Thesis Outline

This thesis is structured in the following way: Chapter 2 discusses the theory and algorithms behind the image processing techniques that are investigated throughout this study. Chapter 3 describes the methods of data capture, and provides a thorough documentation of the pre-processing techniques used in the alignment of data. Chapter 4 provides a thorough investigation of a variety of feature spaces and where the smoke signature resides within each one. The various levels of contribution from different channels will also be explored. Chapter 5 explores the identification of temporal variance through difference imaging on different periods and frequencies of exposure. Chapter 6 shows successful results from selective threshold adjustment to principal component images on different periods and frequencies of exposure. A discussion of the final results, weaknesses, and future work concludes the study in chapter 7.

Chapter 2: Theory and Algorithms

This chapter provides background to the theories and mathematical processes used in this study to create a successful smoke detection algorithm. The majority of processing and analysis is accomplished with MATLAB, a multi-paradigm numerical computing environment developed by MathWorks that allows for convenient matrix manipulations and data processing. Built-in MATLAB functions used will be identified when relevant and documented scripts are included in Appendix A.

2.1 Feature Space

A feature vector is a numerical n-dimensional vector that numerically represents some object. In this study, a feature vector can be constructed using values that correspond to the pixel location of an image or set of images. The feature space is the associated vector space that encompasses the image vectors [**16**]. As an example, a set of three images can be equivalently displayed in their three-dimensional feature space. Each pixel location in the image space has 3 coordinate values picked up from the respective red, green, and blue image. Throughout this study, fire scenes will be analyzed in both the feature space and the image space in order to successfully identify smoke. Figures 1 and 2 offer a comparison of the red, green, and blue channels plotted as a visible image (image space) and plotted in the feature space.

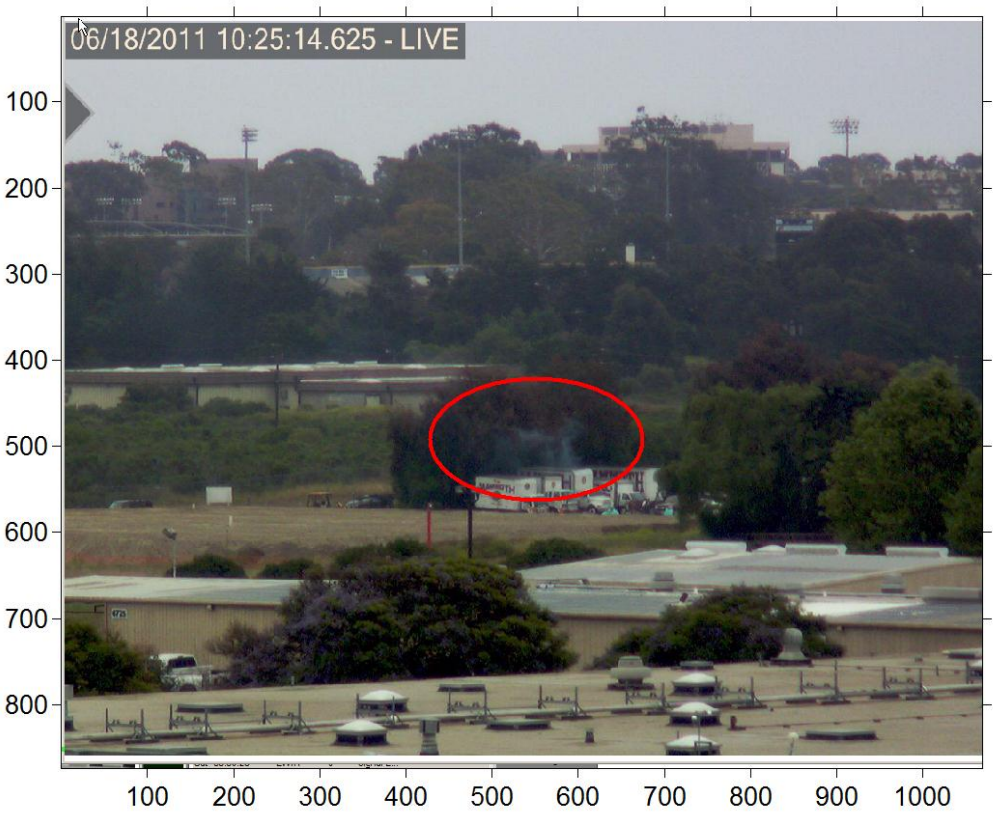

Figure 1: Visible Image of Active Fire Scene with Identified Smoke

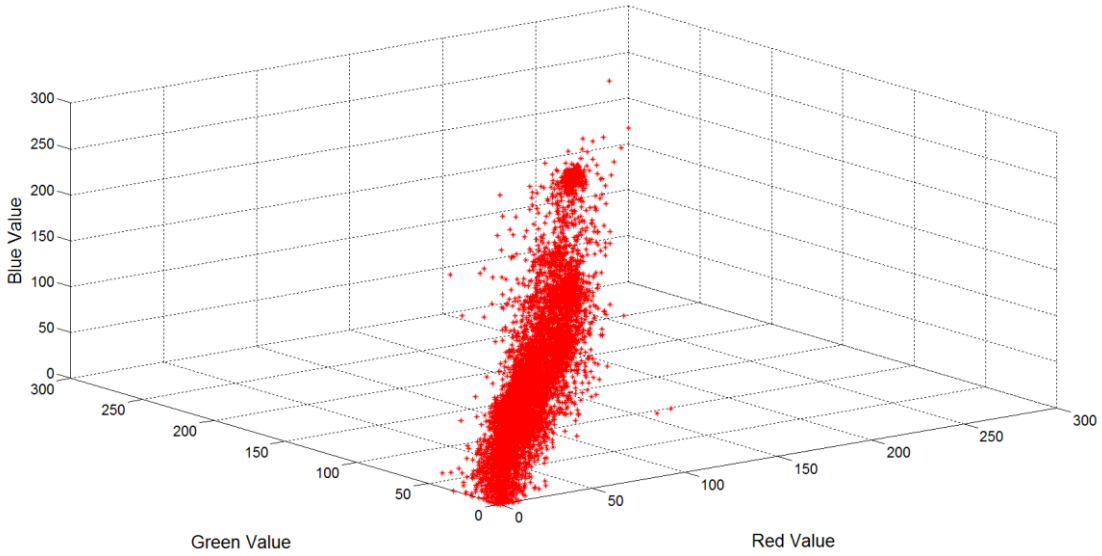

Figure 2: RGB Feature Space of Fire Scene

The normalized rgb feature space is also useful for analysis. The conversion equations can be seen below where lowercase letters represent the new normalized values.

$$r = \frac{R}{R+G+B} \; g = \frac{G}{R+G+B} \; b = \frac{B}{R+G+B} \tag{1}$$

2.2 HSL and HSV

HSL (hue-saturation-lightness) and HSV (hue-saturation-value) are two of the most common cylindrical coordinate representations of the RGB color model. These two representations, inspired by the color wheel, rearrange the geometry of the RGB color space to give a more intuitive representation of color. The angle around the vertical axis represents "hue" and the distance from the central axis is represented by "saturation". The height corresponds to the system's representation of luminance in relation to the saturation. In this study, the spectral signature of smoke will be investigated in both the RGB and HSV representations of an image's feature space.

2.3 Principal Component Transform

Principal component analysis (PCA) is a statistical process that results in as eigenvalue decomposition of sample data. Since eigenvalue decomposition is a linear transformation PCA maintains the signal integrity while revealing the internal structure of the data in a way that best explains the variance in the data. These new dimensions of orthogonal variance become linearly uncorrelated variables called principal components. Each principal component accounts for as much of the variability in the data as possible under the constraint that orthogonal and uncorrelated with preceding components. The number of principal components is always equal to or less than the number of dimensions that the data represents and is sensitive to the relative scaling of original variables [**17**]. Figure 3 shows the effect of a principal component transformation on the feature space of an image.

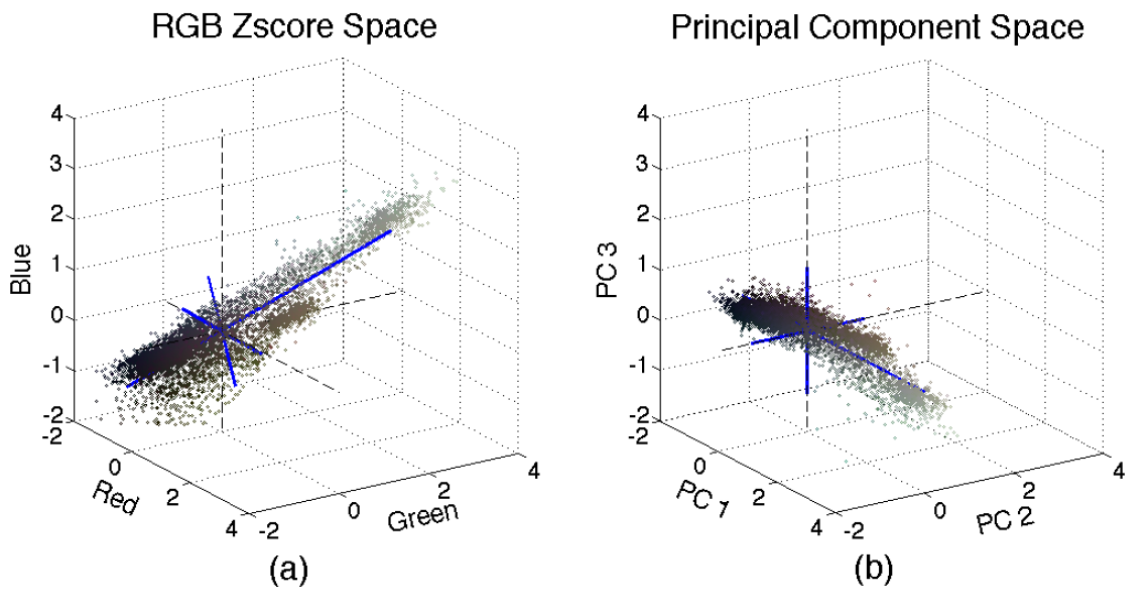

Figure 3: Principal Component Transformation of the Feature Space of a Visible Image

2.4 Gaussian Mapping

In probability theory, the Gaussian distribution is a commonly used continuous probabilistic distribution. This distribution is an important part of data analysis as it is able to represent real-valued random variables whose distributions are not known. This type of statistical analysis is commonly used in natural and social sciences to analyze and make conclusions about different sets of data. The equation for a Gaussian distribution can be seen below.

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

The parameter μ is the mean or expectation of the distribution and represents the most common output of the probabilistic distribution. The parameter σ is the standard deviation and this quantity represents how vast the spread of variation of the data is. MATLAB provides the ability to receive a distribution of data and map that distribution to a Gaussian function while computing mean and standard deviation that corresponds to that data. These parameters can then be used to further process and manipulate a data set.

2.5 Z-Score

Data taken from different sources can often times have different ranges and weights. A normalization process known as Z-scoring scales tabulated data such that it has a mean of 0 and a standard deviation of unity while still preserving data trends and relationships. Z-scoring different sets of data provides the ability for equal weight comparison without bias. In this study, data from a visible camera is properly compared to data from an infrared camera after Z-scoring has taken place.

2.6 Difference and Addition of Images

Images can be represented as data arrays that contain the respective values from a grid of pixels. Values from different arrays can be added or subtracted from one another to represent the addition or subtraction of two images. The difference image looks at the resulting absolute values of a subtraction of two images in order to maintain a commutative property.

2.7 2-D Image Blurring

A bilateral filter can be used to smooth or blur and image while preserving edges and reducing noise. The intensity value in each pixel is changed in accordance to a weighted average of intensity values from nearby pixels. Adobe Photoshop provides a variety of different blurring functions similar to a bilateral filter with much flexibility in different parameters. Specifically, this study uses the "surface blur" function embedded in the Photoshop environment.

2.8 Gamma Correction and AGC

Gamma Nonlinearity and Gamma Correction references a non-linear operation used to optimize the usage of bits when encoding an image. Images that are not gamma-encoded may allocate too much bandwidth to high outlier data values and not enough bandwidth to lower more relevant data. Gamma correction was first developed to counter the non-linearity of cathode ray tube (CRT) displays such that images could be displayed in full range [**18**]. By passing the input

signal through the Gamma function, the data can be spread in such a way that relevant data differences can be observed at all spectra.

Below we can see the Gamma function and figure 4 shows how the function counters the CRT response to produce a linear relationship between the data's true value and its displayed value. The effects of different Gamma values can also be observed.

$$Gamma\ Function: V_{out} = AV_{in}^{\gamma} \tag{3}$$

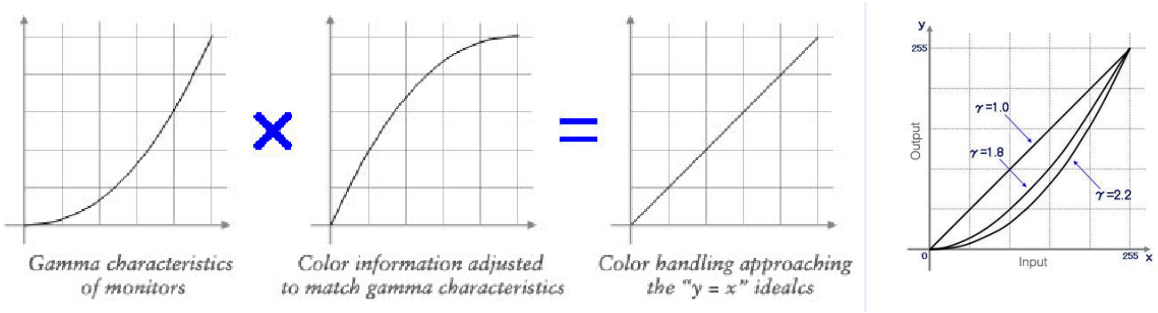

Figure 4: Gamma Correction for Cathode Ray Tubes

An automatic gain control function, *agc.m*, has been created using Gamma correction principles to better display features in the IR images. The effects of the automatic gain control can be observed below in the following IR image of a fire scene in figures 5 and 6 below. The *agc* function can be found in appendix A.

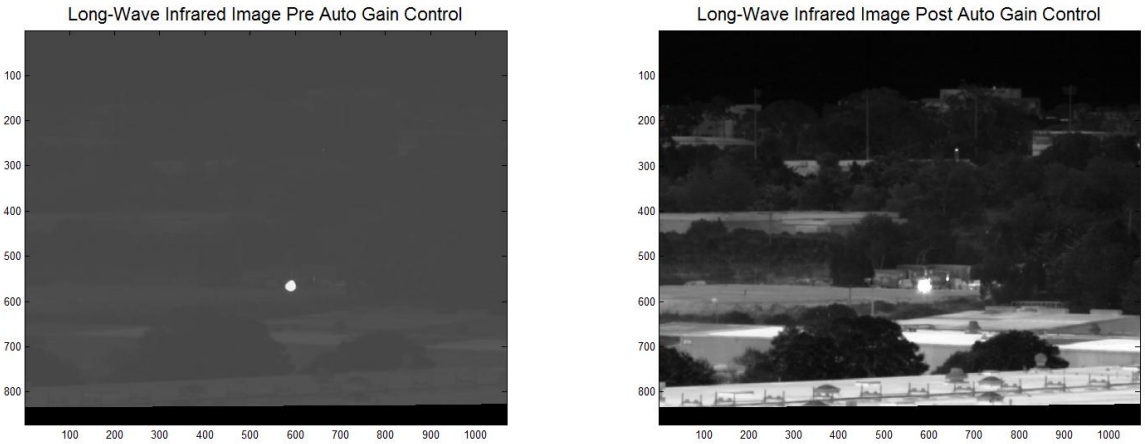

Figure 5: Automatic Gain Control on a Long-Wave Infrared Image

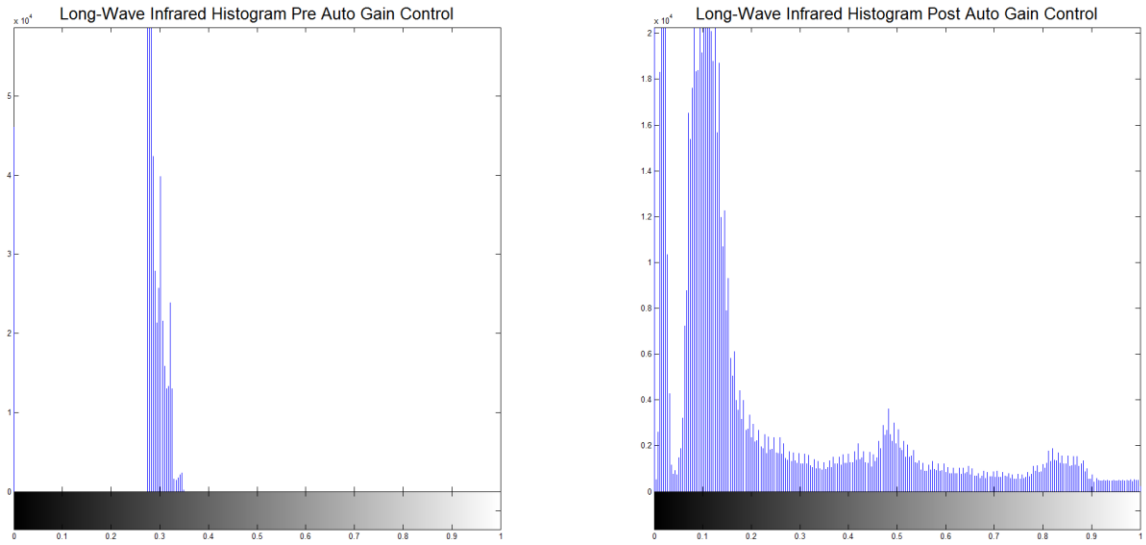

Figure 6: Automatic Gain Control on a Long-Wave Infrared Image Histogram

2.9 Histogram Equalization and Auto Contrast

Histogram equalization is used to increase the global contrast of an image by better distributing the intensities across the histogram. This method can be very useful in images where both the backgrounds and foregrounds are light or dark. Using histogram equalization can effectively spread out the most frequently used intensity values.

In matlab, *histeq(I)*, transforms an intensity image I and outputs and image with a flat and desirable histogram. Although histogram equalization produces unrealistic effects in photographs, the transform can be very useful in scientific images and data processing [**19**]. The effects of histogram equalization can be observed on the IR image below in figures 13 and 14.

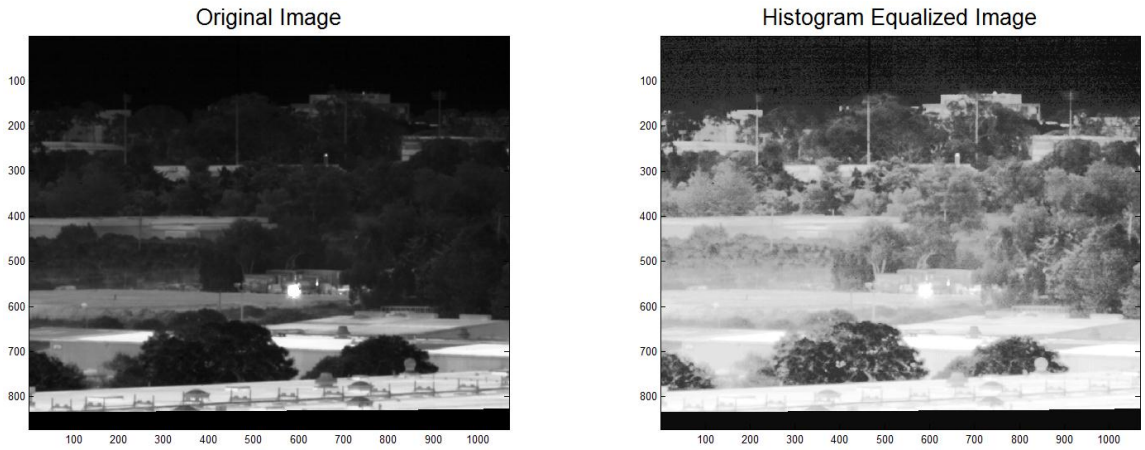

Figure 7: Histogram Equalization on a Long-Wave Infrared Image

2.10 Median Filtering

In digital image processing, it is often desirable to be able to reduce the amount of noise in an image. Median filtering is a non-linear transformational technique that works well at eliminating "Salt and Pepper" noise while preserving edges. This filter runs through the image entry by entry and replaces each entry with the median value of neighboring entries. The median of a list of numbers can be found by picking the middle index of all ordered numbers in that list. The size of the neighborhood of pixels is known as the window, and the size of this parameter can be chosen to accommodate the size of the noise.

Matlab provides *medfilt2*, a 2-dimensional median filtering function which by default outputs an image where each pixel contains the median value in a 3-by-3 neighborhood around the corresponding pixel in the input image. Different sizes of neighborhood classification are used in stages of median filtering in this study. Below we can observe the effects of median filtering on noise removal at one point in the smoke detection algorithm in figure 8.

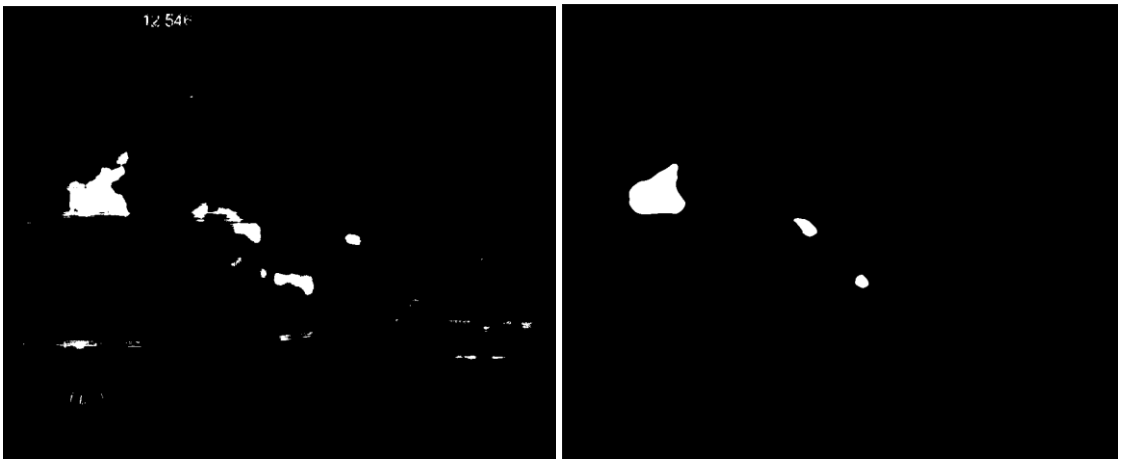

Figure 8: Median Filtering Example

Chapter 3: Data Acquisition and Preprocessing

3.1 Data Acquisition

A fire test was conducted at Raytheon Vision Systems (RVS) on June 18th, 2011 at Raytheon Vision Systems in Goleta, California. The data obtained on this clear and windy day accommodated the means to investigate the temporal, spectral and spatial characteristics of different components of a fire scene. This study is the result of copious analysis of this data.

Over 300 GB of video data was generated from four types of cameras mounted on a tower 875 yards away and pointed at an elevated barbeque pit. A natural wildfire was simulated by burning various different fuels including wild oats, pine needles, pine cones, palm leaves, thistle, wet and dry leaves, wild rosemary, and apple wood in the barbeque pit. Fuels were added continuously in order to simulate a growing fire. The video analysis and image processing in this study focusses on a period of data that captures the burning of wild oats as fuel.

3.2 Video Data Format

The fire scene was captured in an 8-bit visible camera, and three 14-bit infrared (IR) cameras which include cooled mid-wave IR, cooled long-wave IR, and uncooled long-wave IR spectral bands. The visible data is saved as and AVI file with 24-bit pixel values. Each 24-bit value consists of three 8-bit vales which represent the Red, Green, and Blue channels. The IR data channels (CLWIR,CMWIR,ULWIR) are saved as 16-bit RAW bit streams with zero padding on the least significant bits to represent the captured 14-bit data.

Each frame in the visible data is time stamped with the date and time (up to milliseconds) of capture. The IR video streams are not time stamped but aligned exactly with each other and pre-organized into smaller video files corresponding to each fuel stage. An excel document is included with the video files which documents when noticeable events occur in the fire scene such as the changing of fuel or extinguishment of the fire. Observations and details are included

from a Fire Site Team and a Command Center Team to aid in the alignment between video streams and the subsequent extracted images.

The data specifications for all four cameras can be seen in table 1 below. IR cameras capture 30 IR frames a second with a resolution of 640 x 480 where each pixel is stored as a 14-bit value. The visible camera captures 3 visible frames a second with a resolution of 1086 x 873 where each pixel is stored as a 24-bit value. Each pixel in a visible frame is made of three 8-bit values representing the Red, Green, and Blue components each with a resolution of 1068 x 873.

Sensor	Visible	Cooled MWIR	Cooled LWIR	Uncooled LWIR
Resolution	1068 x 873	640 x 480	640 x 480	640 x 480
Bit Depth	3 x 8-bit	14-bit	14-bit	14-bit
Fames Per Sec	3	30	30	30

Table 1: Video Capture Specifications

3.3 Pre-Processing

In order to produce a successful smoke detection algorithm using the Goleta fire data, a considerable amount of pre-processing must be done to extract image frames from their respective video files and align them spatially and temporally. The following sections will follow the pre-processing techniques described in Tim Davenports' master's thesis, *Early Forest Fire Detection using Texture Analysis of Principal Components* [15]. Due to data corruption, the pre-processing techniques are revised and documented to help future fire-fighters advance to a point where relevant processing can take place. Weekly exchanges with Tim Davenport, Dr. John Jacobs, and Dr. John Saghri not only provided the means to successfully pre-process the Goleta fire data but also their guidance was vital in guiding this study to a successful completion.

It is important to note that although data for the Uncooled LWIR band is included in the Goleta fire data; this channel makes little contribution to the signature of the smoke and thus adds no new information about a fire scene. Moreover, the relevant field of view (FOV) from the

Uncooled LWIR data channel provides a much lower resolution in comparison to the visible channels and the alternative IR channels.

The end goal of the following sections of this chapter is to be able to view images of consecutive fire scenes with the same field of view in 5 temporally aligned dimensions (Red, Green, Blue, LWIR, and MWIR). These different dimensions of data will also be referred to as channels and the next sections give detailed analysis of how to extract frames and align them both spatially and temporally.

3.3.1 Frame Extraction from Visible and IR Data Channels

Figure 9 helps illustrate part of the data acquisition process. Visible data stored as an audio video interleave (AVI) file is extracted to frames saved as bitmaps (bmp). The Cooled MWIR and Cooled LWIR are stored as one as one dual band raw (RAW) bit stream and are extracted appropriately into separate raw images.

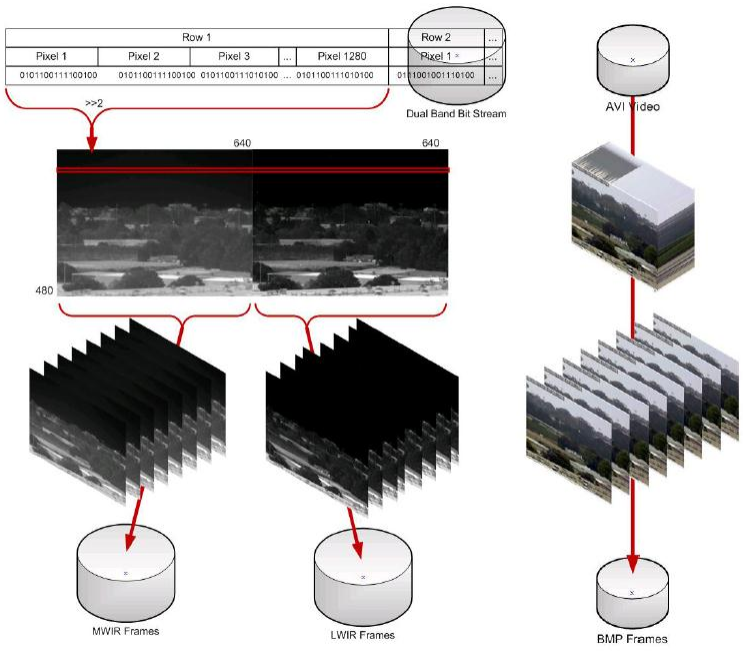

Figure 9: Multi-Spectral Frame Extraction Process

MATLAB script *visvid2frame* and takes an AVI video file as an input and extracts all frames as bmp images to a specified directory. Similarly, *CLWMWIRvid2frame* receives a dual

band RAW bit stream and extracts RAW images to as specified directory. These scripts were written by Tim Davenport and edited by David Garges for this study. Both of these scripts can be found in Appendix A.

3.3.2 Spatial Alignment

The visible camera and the IR camera have different field of views such that initially there is no direct spatial correspondence between images from each camera. Although the visible camera has a larger resolution than that of the IR camera, it has a much narrower field of view. The IR images will be cropped and mapped to the FOV and resolution of the visible images using an affine transformation. Twenty hand selected correlation coordinates used for the alignment are displayed in Figure 10.

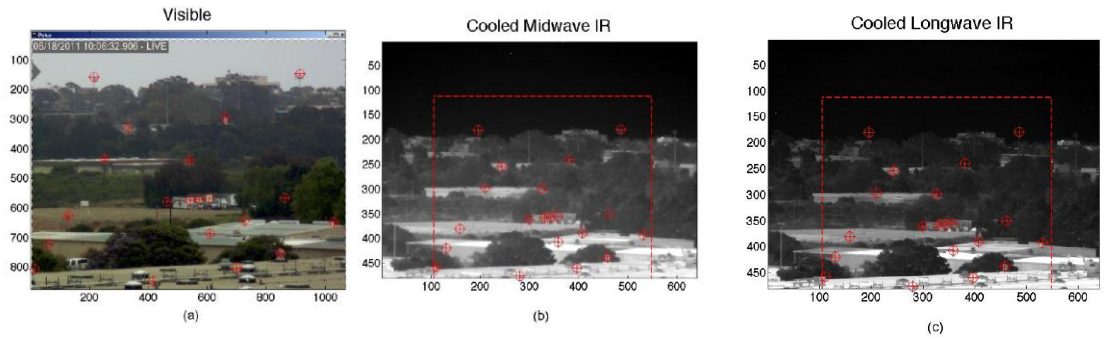

Figure 10: Unregistered FOV

The twenty pixel coordinates in the above images can be represented as $\{\boldsymbol{x_{IR}}, \boldsymbol{y_{IR}}\}$ and $\{\boldsymbol{x_{Vis}}, \boldsymbol{y_{Vis}}\}$ and are used to find the image transformation that will map FOV and resolution of the IR images to that of the Visible image.

$$\begin{bmatrix} \boldsymbol{x_{IR}} \\ \boldsymbol{y_{IR}} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} \boldsymbol{x_{Vis}} \\ \boldsymbol{y_{Vis}} \\ 1 \end{bmatrix} \tag{4}$$

The coefficients for the transform can be determined by solving for the best fit solution of the two systems.

$$\begin{bmatrix} x_{IR}^{1} \\ \vdots \\ y_{IR}^{20} \end{bmatrix} = \begin{bmatrix} x_{vis}^{1} & y_{vis}^{1} & 1 \\ \vdots & \vdots & \vdots \\ x_{vis}^{20} & y_{vis}^{20} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} x_{IR}^{1} \\ \vdots \\ y_{IR}^{20} \end{bmatrix} = \begin{bmatrix} x_{vis}^{1} & y_{vis}^{1} & 1 \\ \vdots & \vdots & \vdots \\ x_{vis}^{20} & y_{vis}^{20} & 1 \end{bmatrix} \begin{bmatrix} d \\ e \\ f \end{bmatrix} \tag{6}$$

After solving for the coefficients, equation 6 can be used to transform all IR images to have the same FOV as the visible images. The resolution can then be matched using equation 7 below.

$$IR_{registered}(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x_{IR}^{i} y_{IR}^{j} \tag{7}$$

Where a_{ij} are the weighting coefficients derived from sixteen closest pixels in the unregistered IR frame. The new registered images can be observed in figure 19 below. The matlab script *alignCLWIRFOV* is used to accomplish this specific transformation and can be found in Appendix A. This script takes IR frames and transformation coefficients as inputs and outputs spatially aligned IR images to a specified directory.

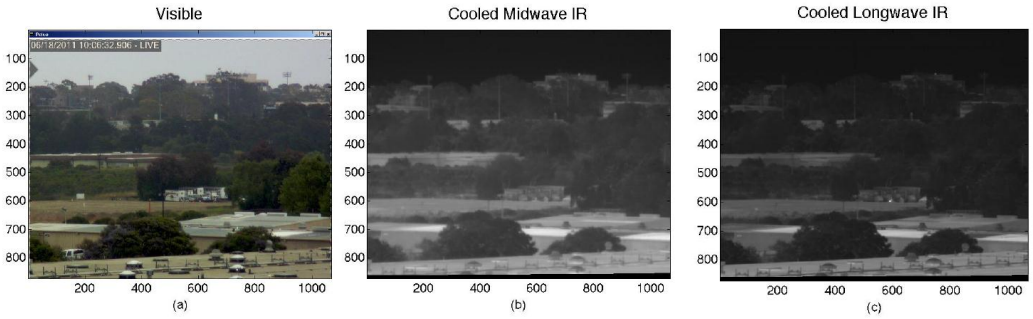

Figure 11: Registered FOV

3.3.3 Temporal Alignment

The IR camera and the visible camera captured frames are not synchronous to each other. Because of this, temporal analysis is not possible without temporal alignment. Time stamps on

visible images, and timing notes included with the Goleta data are used for general alignment. Key frames are hand selected from the beginning, middle, and end of the videos when noticeable events took place within the scene. Using key frames and the known frame rates of the video, frames can be extrapolated such that IR images are temporally aligned with the visible images. Figure 12 provides an illustration of temporally alignment between visible and infrared video frames.

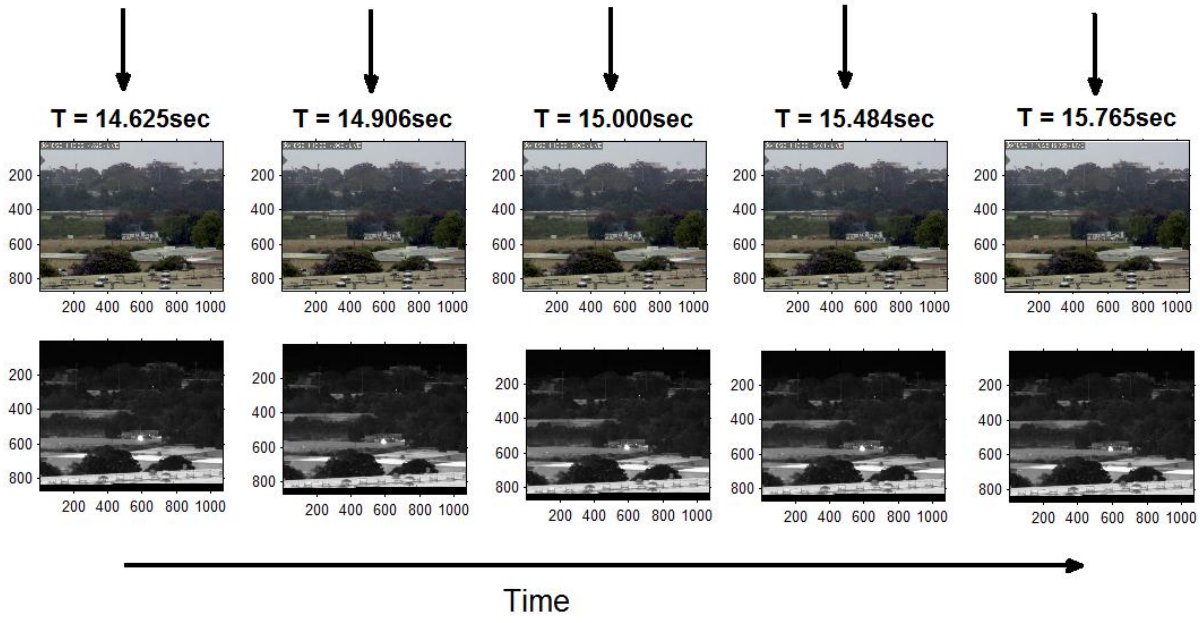

Figure 12: Temporal Alignment

3.3.4 Gamma Correction and Histogram Equalization of Fire Images

A combination of histogram equalization and gamma correction through the automatic gain control function is implemented on IR images to enhance contrast and spatial properties. A more detailed analysis of this process is explained in section 2.8 and 2.9.

3.4 Working Data

The end result of the pre-processing of the data gives access to 100 consecutive instances in time each with 5 dimensions corresponding to the 5 channels. The visible camera contributes the red, blue, and green channels while the IR camera contributes the long-wave infrared and

mid-wave infrared channel. All image frames have the same resolution and field of view which allows for consistent and robust digital image processing techniques to successfully identify smoke in a fire scene.

Chapter 4: Feature Space Analysis

4.1 Smoke Contribution From Infrared Images

The infrared channels are investigated to see their contribution to smoke information. Below in figures 13 and 14, a visible image is compared to two respective infrared images:

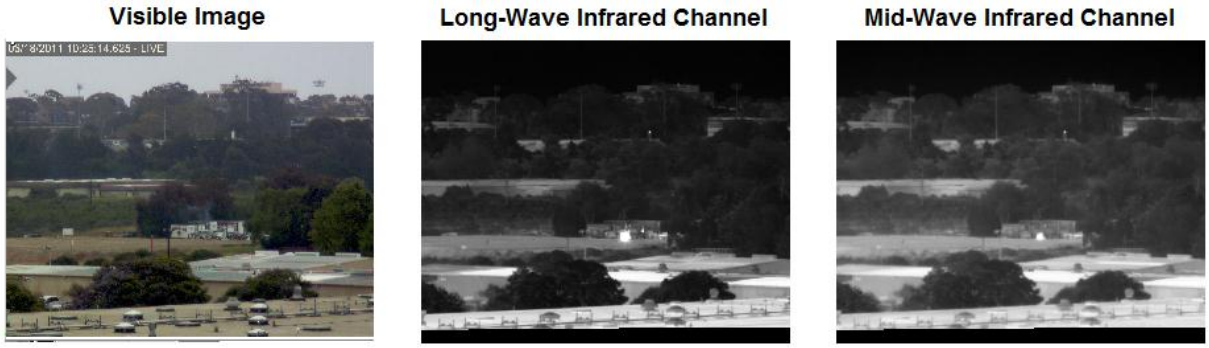

Figure 13: Visible and Infrared Comparison

Figure 14: Visible and Infrared Comparison (Zoomed)

From these comparisons, it is concluded that smoke emits a weak and undetectable infrared signature. Smoke can be clearly observed in the visible image and there is no sign of smoke in the temporally aligned infrared images. To add continuity to this conclusion, fire fighters use infrared cameras to locate humans in a burning building when smoke is too thick to see through with the naked eye [**20**]. From this point forward, the infrared channels will not be used in data analysis as they provide little to no information about the smoke signature.

4.2 RGB and normalized rgb Feature Space

An image from the data set with a time stamp of 14.625 seconds will be analyzed in various feature spaces. Below in figure 15 the smoke has been manually identified using the spray paint tool in Microsoft paint.

Figure 15: Fire Scene (Left) and Manually Identified Smoke (Right)

The RGB and the normalized rgb feature space from these images are displayed below in figures 16 and 17. Black values represent the manually selected smoke pixels while the red values represent all other pixels in the image.

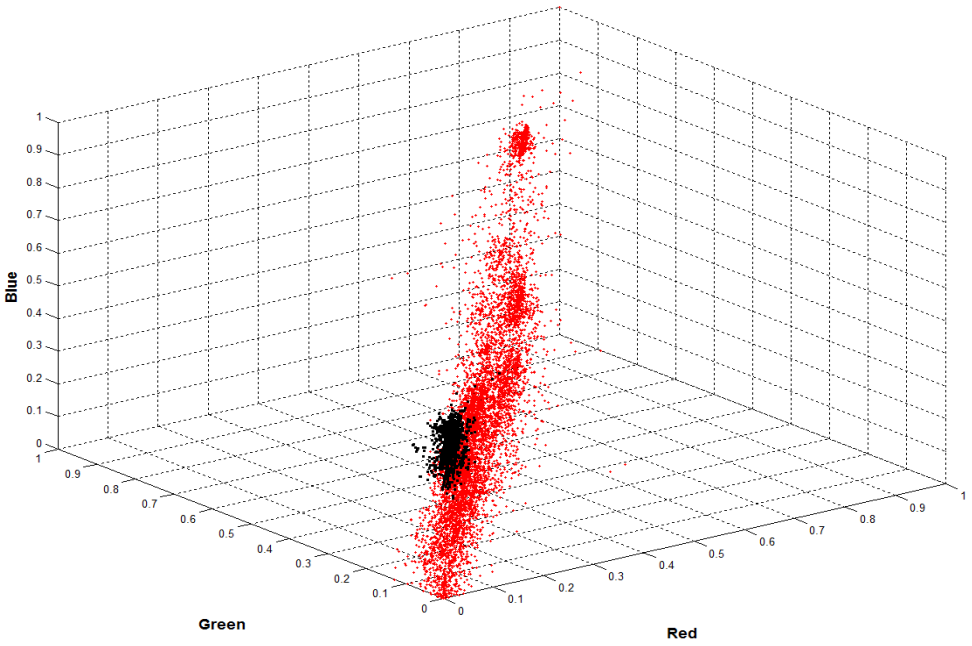

Figure 16: Feature Space of Fire Scene in RGB

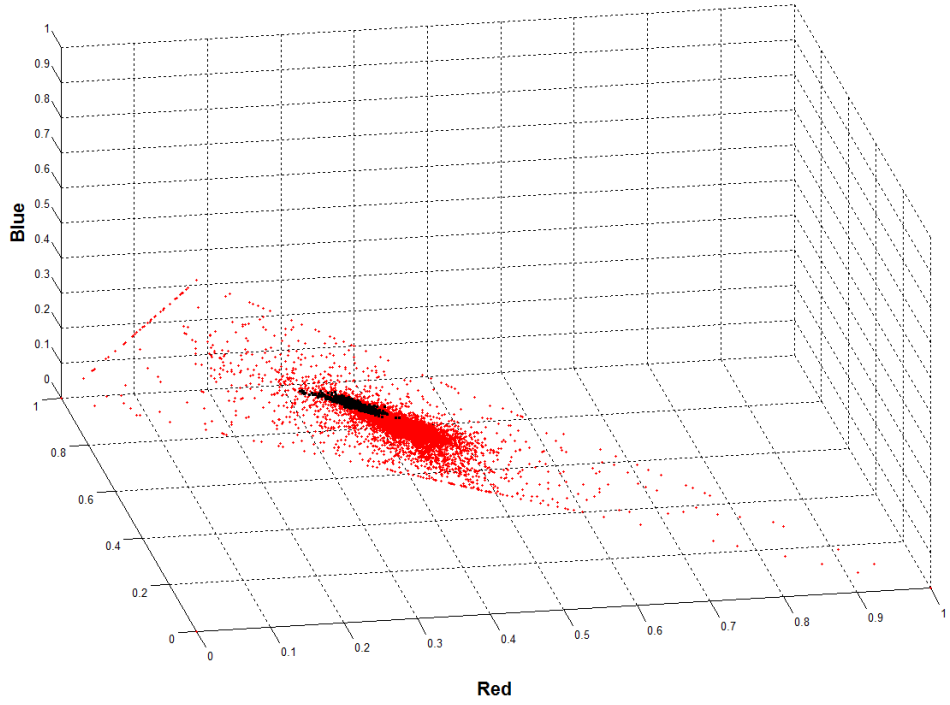

Figure 17: Feature Space of Fire Scene in normalized rgb

4.3 HSV Feature Space

Using the transformation equations defined in chapter 2.1, the features of the smoke can also be compared in the Hue, Saturation, and Value feature space. Below we have identified the

hue range of smoke pixels in comparison to the hue of all other pixels in figure 18. The hue is also compared to saturation and value in figures 19 and 20.

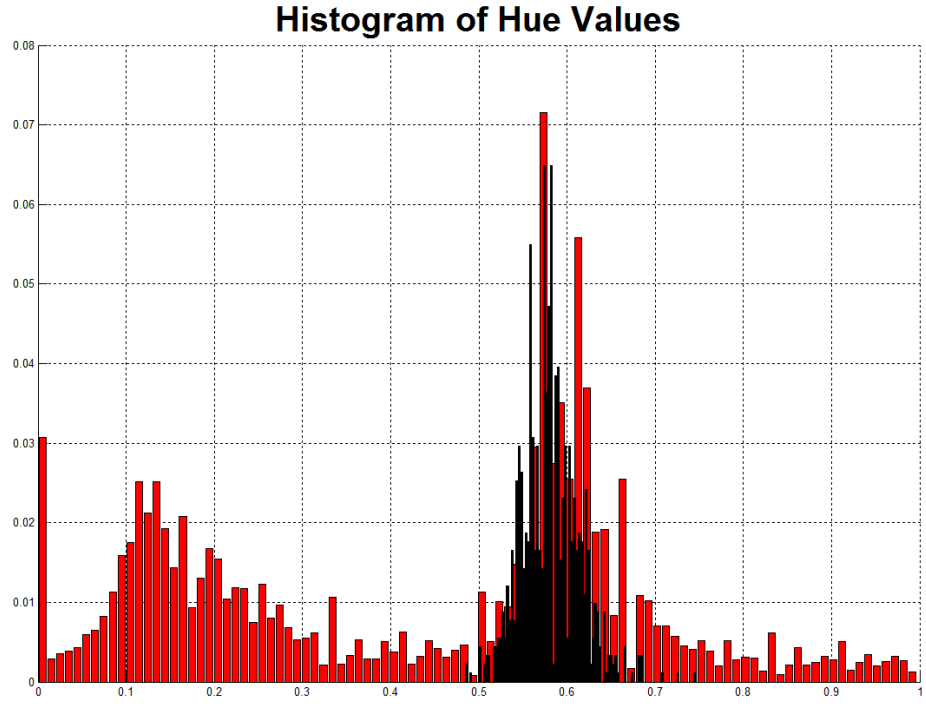

Figure 18: Histogram of Hue Values

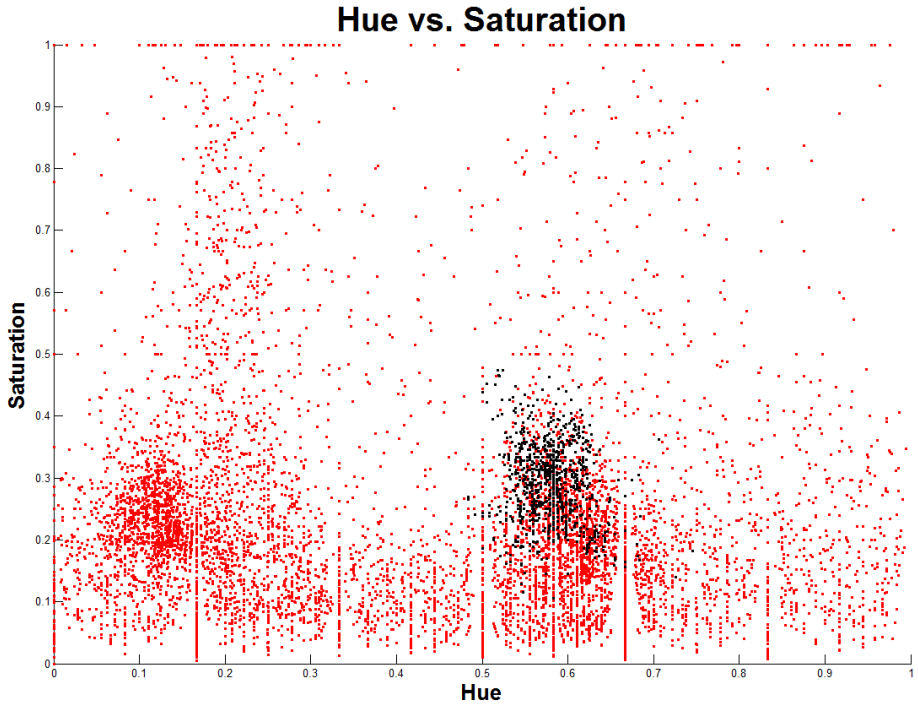

Figure 19: Hue vs. Saturation of Fire Scene

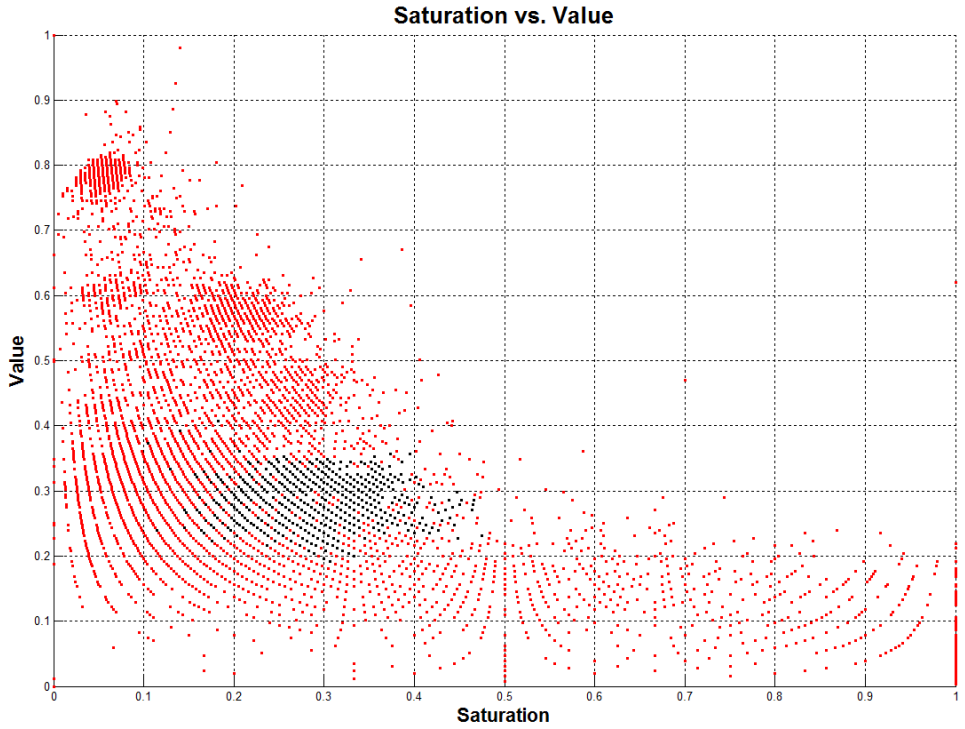

Figure 20: Saturation vs. Value of Fire Scene

In all of the feature space plots, the smoke pixels are localized to an area that can be defined as the spectral smoke signature. Unfortunately, other non-smoke features in the images are intermixed in the smoke signature region. This identification of a spectral smoke signature allows for classification of pixels up to a certain confidence but is susceptible to false positives because the smoke signature is not removed from other features.

Chapter 5: Difference Imaging Scheme

The next two chapters attempt to identify smoke by identifying and accentuating its temporal variance through time. Using a set of temporal images from a given exposure time and frame rate, many smoke false positives can be eliminated as they are stagnant in comparison to a moving smoke plume. The simplest way to identify a moving object in a set of images is to find the difference images. This process will be investigated in this chapter.

5.1 Visible Comparison of Smoke Signature

The goal of this section is to determine which of the visible spectral bands (Red, Green, or Blue) provides the strongest smoke signature. If one of the visible channels yields a more pronounced smoke signature, future processing can be simplified by only viewing data from that channel. Below, aligned images from the three visible channels and their respective histograms can be observed. Qualitatively, it can be argued that the smoke in the center of the frame is most explicit in the blue channel. Using the manually identified smoke in section 4.2, the feature location of the smoke can be identified in each individual channel in figure 21.

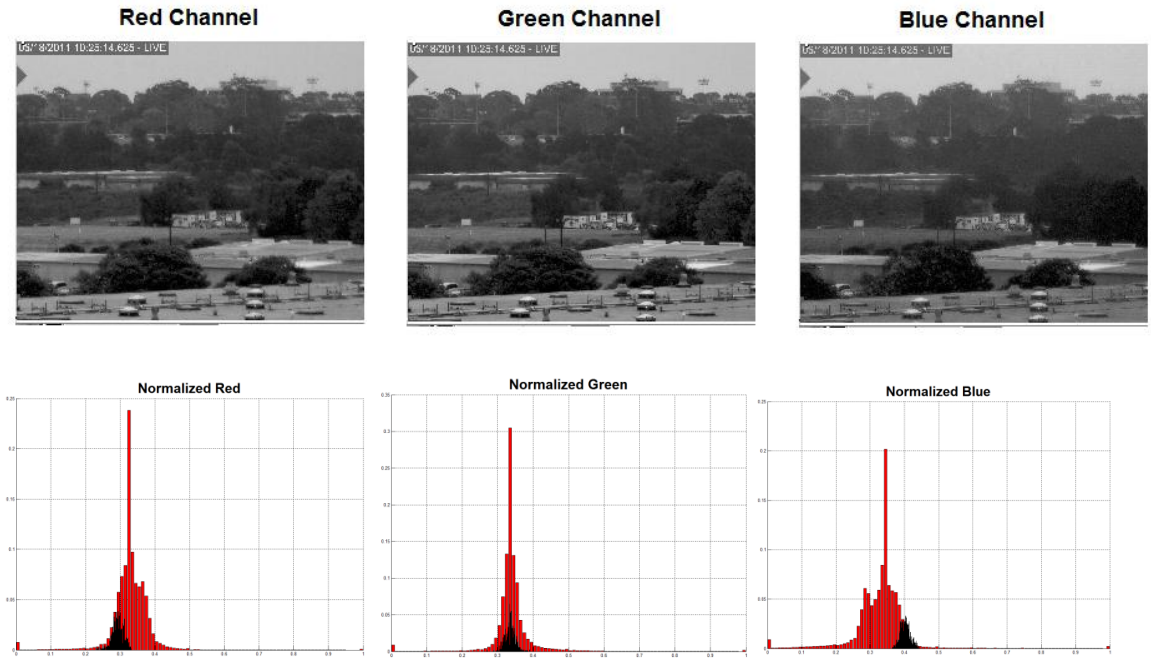

Figure 21: Image and Histogram Comparison of Fire Scene with Visible Bands

The normalized blue channel for this image has a notable separation in comparison to the red and green channels. Because the smoke pixels are more withdrawn from the distribution of other feature values in the blue channel, temporal analysis will be done on the blue channel for the remainder of this study.

5.2 Layout and setup

In this process, four difference images are created by differencing 5 tabulated temporal frames with equal exposure time. A cumulative difference image is created by adding the resulting frames together. Histogram equalization can be applied to improve the contrast of the cumulative image. In this image, pixels with high intensity values are associated with high temporal variance. In figure 22 below, the temporal spacing of the original tabulated images is 2 seconds.

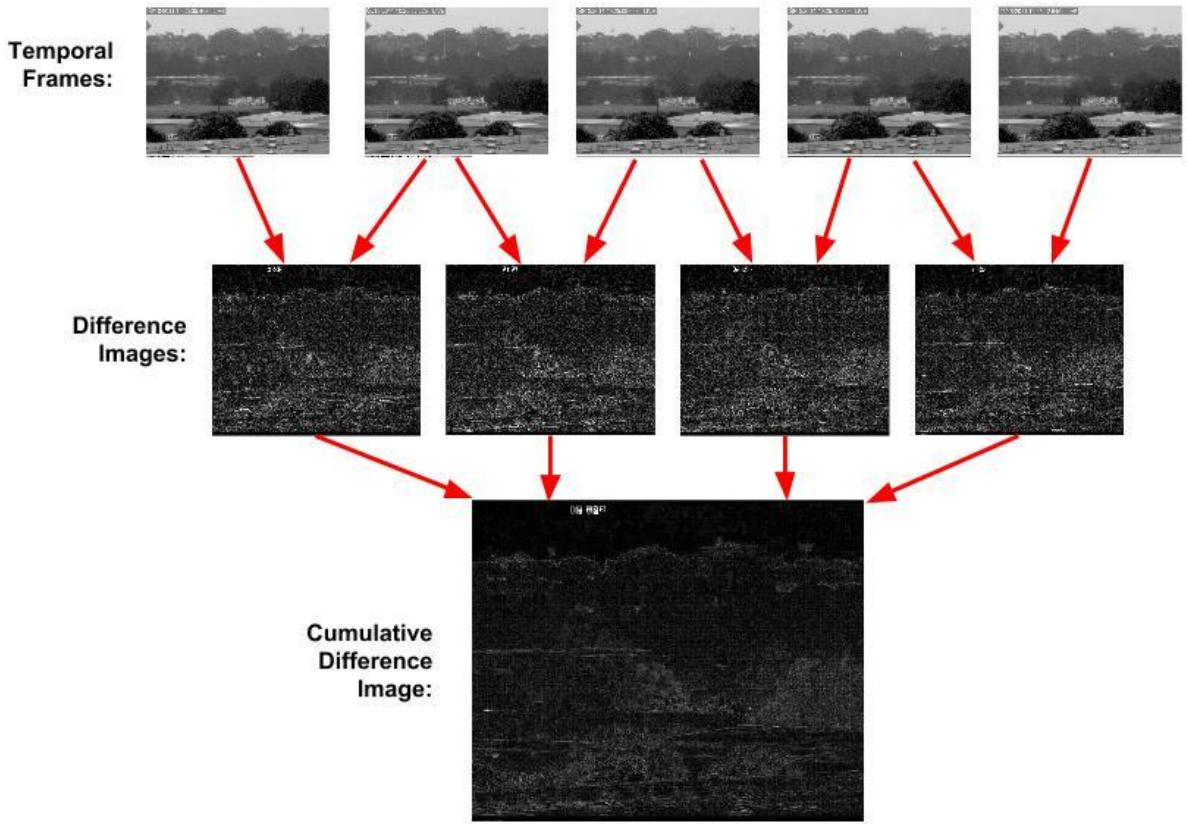

Figure 22: Construction of Cumulative Difference Image

5.3 Image Enhancement and Selective Threshold Adjustment

After difference images have been summed together, blurring is used to smooth out the intensities of the image's features. A selective threshold adjustment tool in Photoshop is then used to separate areas with high intensities from areas with low intensities. Median filtering is used to identify large blobs of variance and eliminate small and irrelevant locations of variance. The block diagram in figure 23 below shows the processing scheme on four difference images.

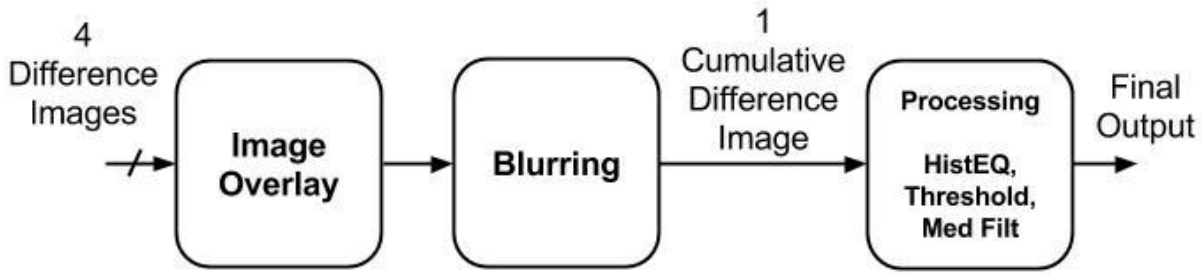

Figure 23: Block Diagram of Difference Image Threshold Scheme

5.4 Results from different intervals

Using 5 temporal images in each scenario, the image capture period is adjusted to see its effect on results. Below, results from capture intervals of 0.3 seconds, 2 seconds, and 5 seconds can be observed.

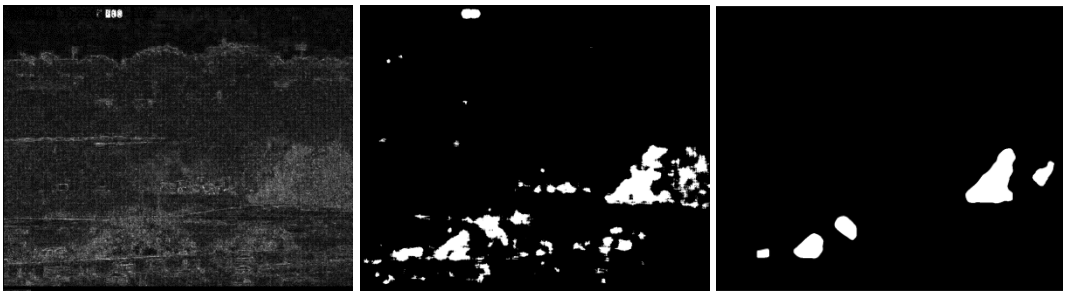

Figure 24: Difference Imaging Scheme with Delta = .3sec

Figure 25: Difference Imaging Scheme with Delta = 2sec

Figure 26: Difference Imaging Scheme with Delta = 5sec

5.5 Conclusion

A differencing image scheme using five temporal images was created in attempt to locate moving smoke by identifying temporal variance. Using blurring, selective threshold adjustment, and median filtering, areas of large temporal variance are segmented from areas with low temporal variance. Unfortunately, small movements within the scene throughout the exposure time can contribute a comparably high temporal variance to that of the moving smoke. Moreover, vibrations in the tower from wind contribute strong problematic temporal edges within the difference images.

Based on the results in figures 24, 25, and 26 above, it is determined that the temporal spacing of images needs to be at least 5 seconds in length in order for the smoke's temporal variance to over-power that of the false positives. It also is important to note that the threshold adjustment value used to produce the results in section 5.4 is not consistent throughout all test cases and was manually chosen to produce the best result possible. If an autonomous system were to be implemented, the choice of the threshold value would provide further complications to this method. Overall, this study concludes that the difference imaging scheme for identifying smoke by its temporal variance is ineffective especially with high frequencies of frame rates.

Chapter 6: Selective Threshold Adjustment on Principal Component Images

Chapter 6 investigates an alternate and more successful method used to locate smoke by segmenting its temporal variance. Principal component images resulting from a principal component transform of temporal frames are processed using similar techniques to those used in chapter 5. The success of method motivates a technique to develop an autonomous threshold value for each principal component image. This chapter investigates the resulting effects of different parameters throughout the process and also provides results from two different data sets containing various levels of smoke.

The primary data set contains images with heavy smoke, while a secondary data set contains images with small amounts of smoke. The data was initially thought to contain no features of smoke, however results from the algorithm suggest that there is still faint amounts of smoke. The first data set will be referred as “Smoke” while the second data set is referred to as “NoSmoke”.

6.1 Principal Component Transform of Five Temporal Blue Images

Just as in chapter 5, five temporal images with a given frame rate are selected for temporal processing. A principal component transformation is applied to the temporal images and the resulting principal component images can be seen in figure 27. In the resulting principal component images, areas with large temporal variance have intensity values outside of the mean value of the image. Section 6.2 discusses the autonomous process to segment the temporal smoke in the resulting principal component images.

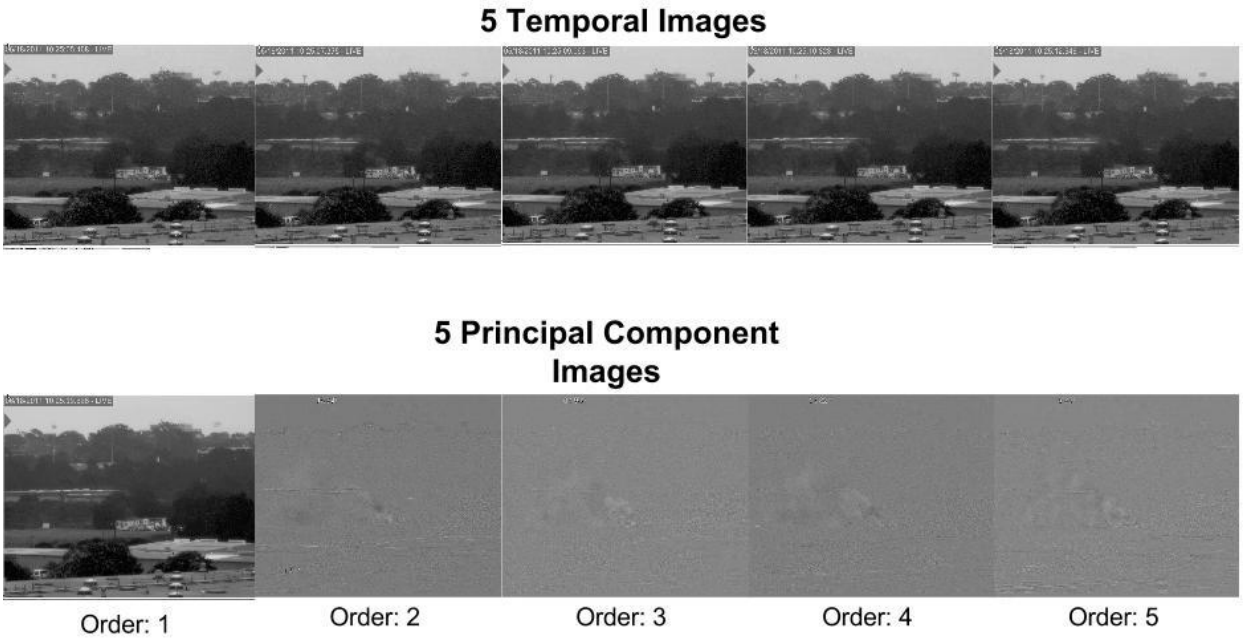

Figure 27: 5 Temporal Blue Images and Principal Component Images

6.2 Determining the Autonomous Threshold Value

In order to segment the temporal variance of the smoke, principal component images are blurred and then mapped to a best fit Gaussian function. An autonomous threshold adjustment value is then determined based off of the mean and standard deviation of the best fit Gaussian. The automatic threshold value for each principal component image is determined by the following relationship:

$$Theshold = \mu + A * \sigma \quad (8)$$

Where μ is the mean intensity value and σ is the standard deviation of the intensity value distribution of a given principal component image. An experimental tuning constant A is included in the autonomous threshold adjustment to provide flexibility across different types of fire scenes. The following experiments show successful results with a tuning constant value of A=4.5.

Figure 28 shows a block diagram of the principal component image analysis. Figures 29 and 31 show images throughout the process on a principal component image 2 of 5, with an exposure rate of one image every 2 seconds. Figure 30 demonstrates the Gaussian mapping of the histogram from the principal component image used in this example. Lastly, figure 32 displays the results of the automatic threshold adjustment scheme on all the principal component images from this example.

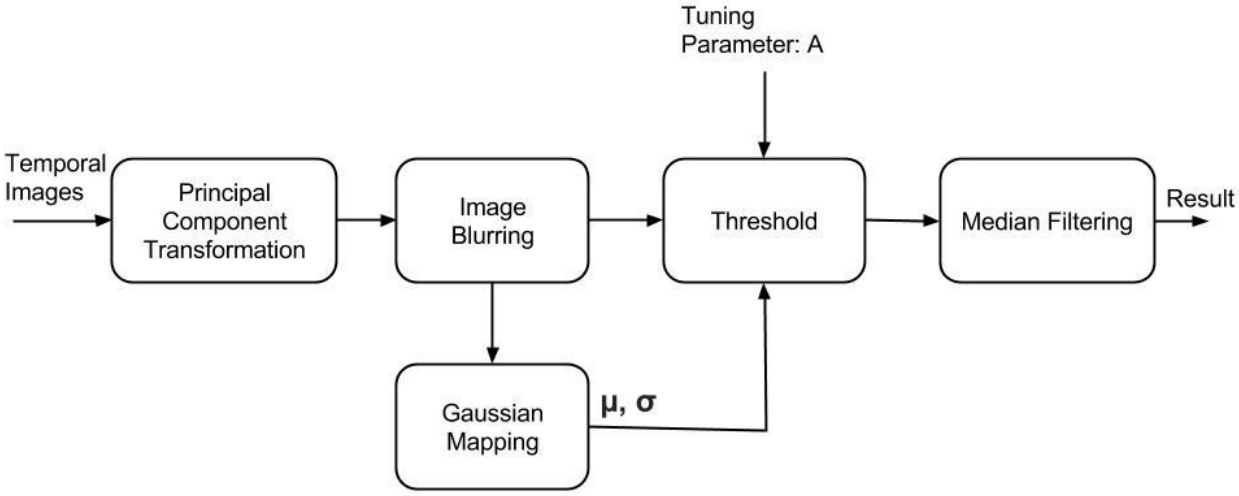

Figure 28: Block Diagram for Selective Threshold Adjustment on PC Images

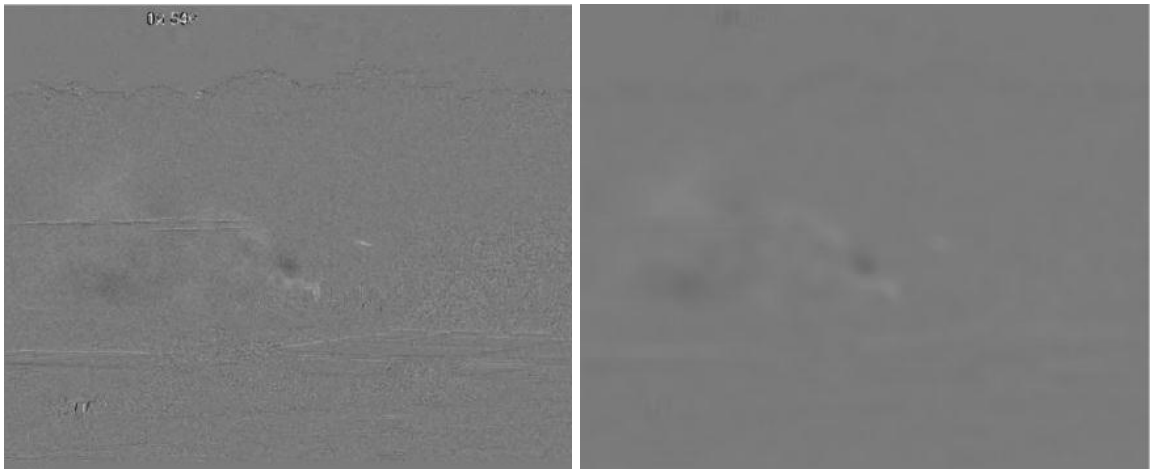

Figure 29: Principal Component Image (left) Blurred Principal Component Image (right)

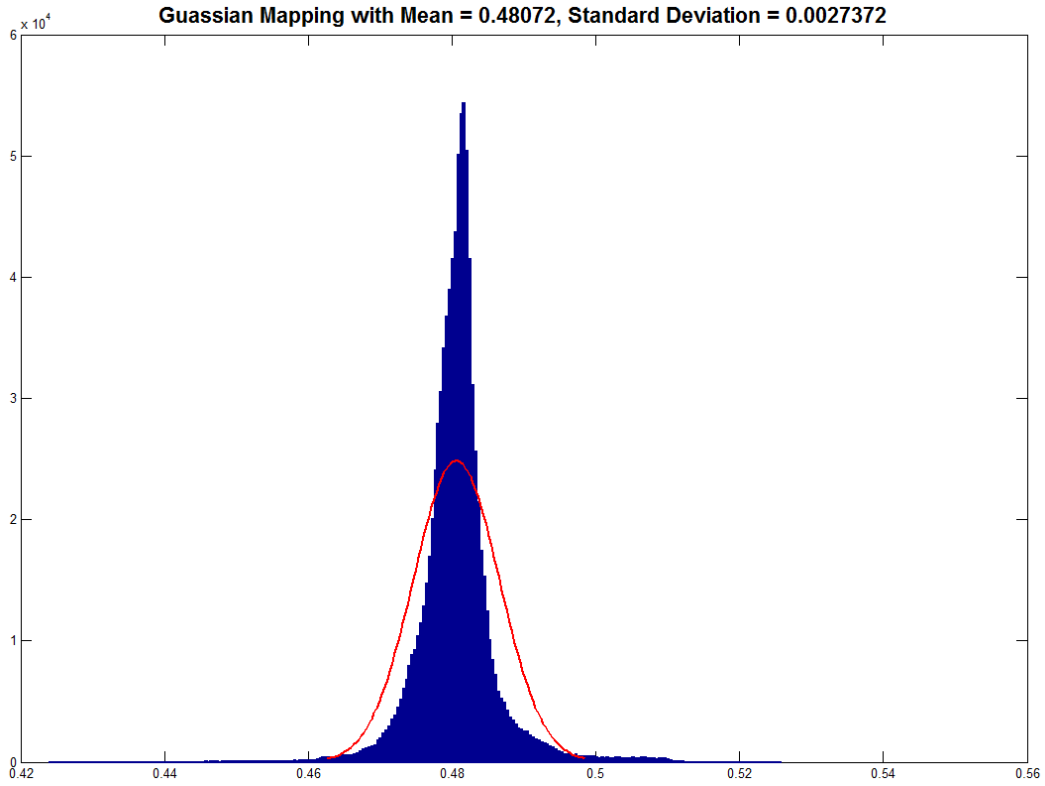

Figure 30: Gaussian Mapping to Histogram of Blurred Principal Component Image

Figure 31: Threshold Adjustment (Left) Median Filtered Result (Right)

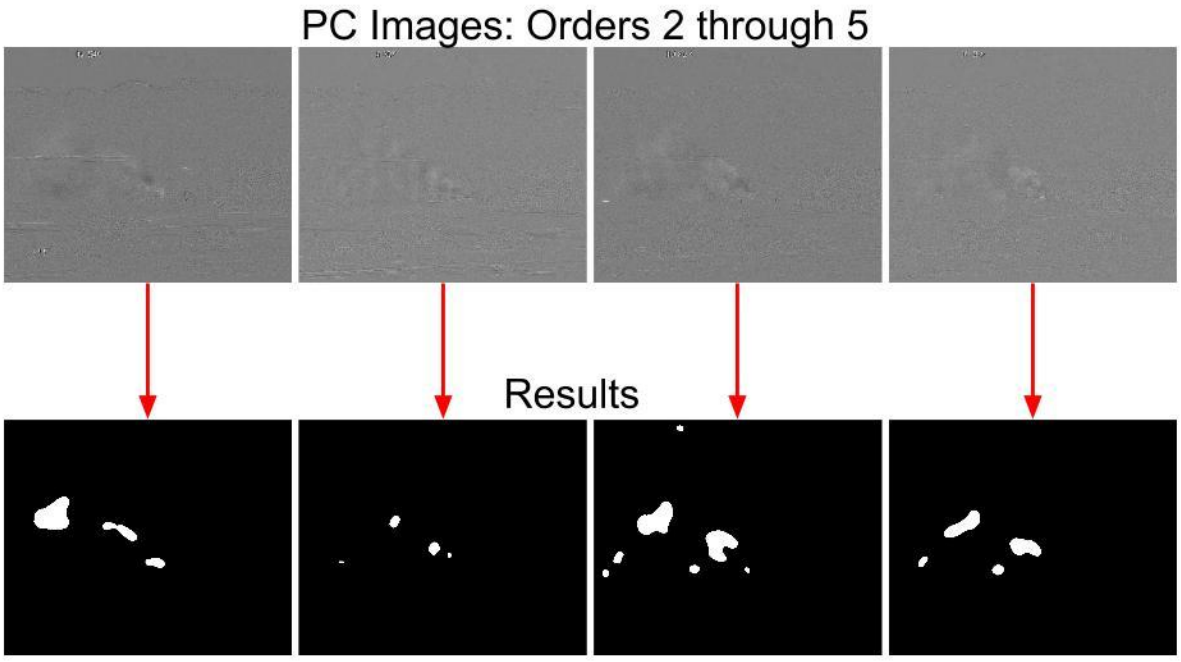

Figure 32: Selective Threshold Adjustment Process on Principal Components 2 through 5

6.3 Results from Different Intervals

Results from this process are compared using different intervals between five temporal images. Below in figure 33, we can observe that results grow stronger as exposure period increase. This makes sense because a longer capture period allows for more explicit temporal variance from moving smoke in a fire scene. This study concludes that 2 seconds per temporal image is the minimum required frame rate to produce successful results.

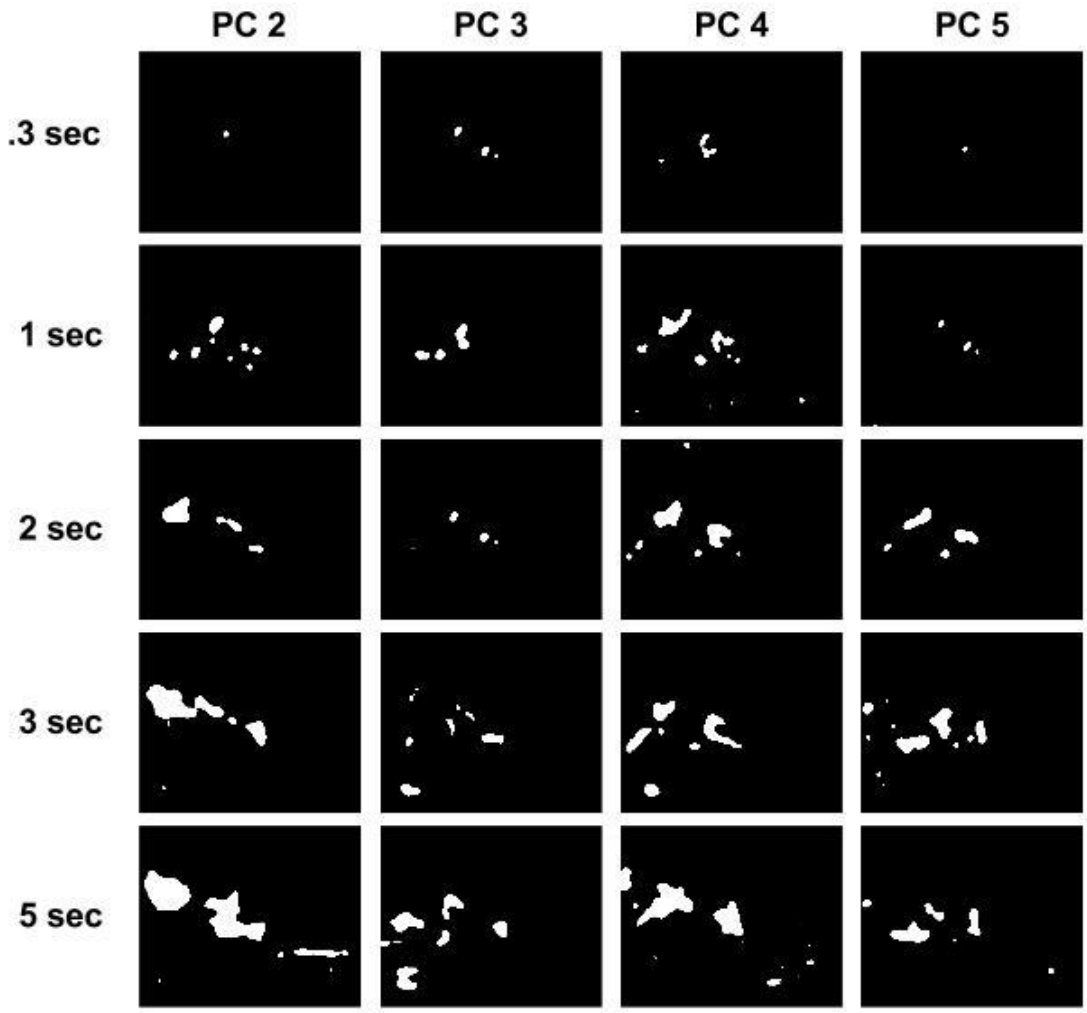

Figure 33: Selective Threshold Adjustment Results on 5 Blue Images, Varying Intervals

6.4 Results from Different Spectral Channels

Using five temporal images with 2 seconds in between each temporal image, results can be compared using different channels from the visible spectrum. Below in figure 34, the effects of using the red, green, and blue channels can be compared to one another. The following results reinforce that the blue channel provides the strongest spectral and temporal smoke signature.

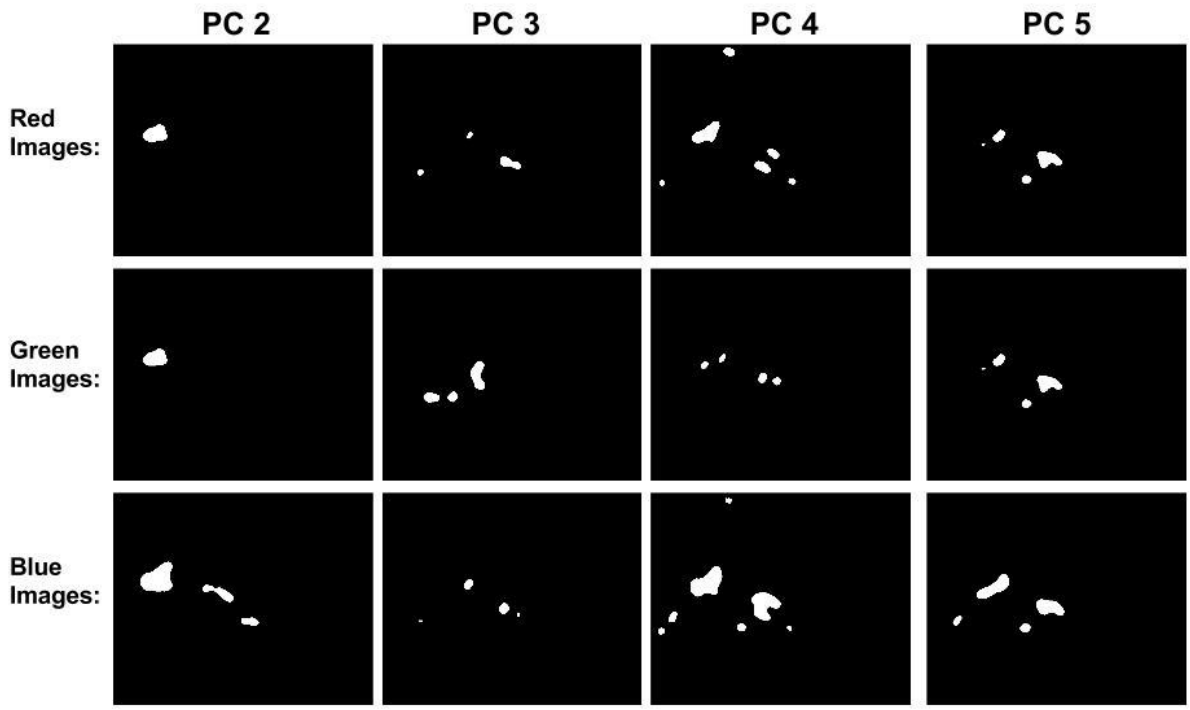

Figure 34: Comparison of Selective Threshold Adjustment on Visible Channels

6.5 Results from Different Numbers of Temporal Images

Figures 35, 36, and 37 display results on principal components generated from 3, 5, and 8 temporal images respectively from the Smoke and "NoSmoke" data set. Each temporal image is separated by 2 seconds in time. Results show that as the number of temporal images increase, results become stronger. However, an increased number of temporal images results in longer exposure time and a longer processing time. Although using different numbers of temporal image provides successful results, this study concludes that the use of 5 temporal images is a happy medium between exposure and processing time and consistency.

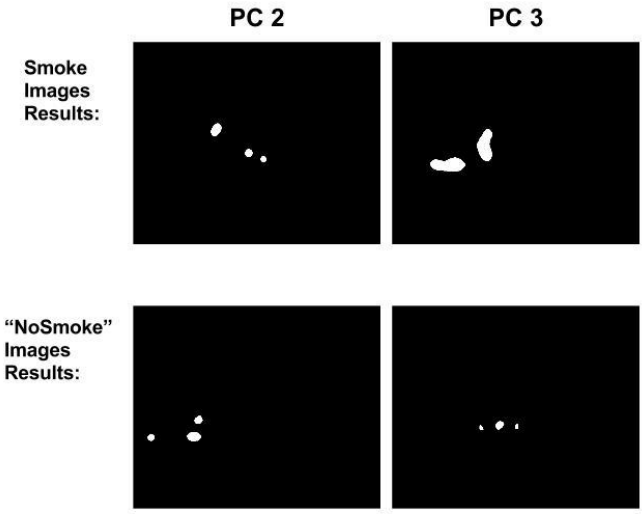

Figure 35: Results from 3 Temporal Images from Smoke and "NoSmoke" Data

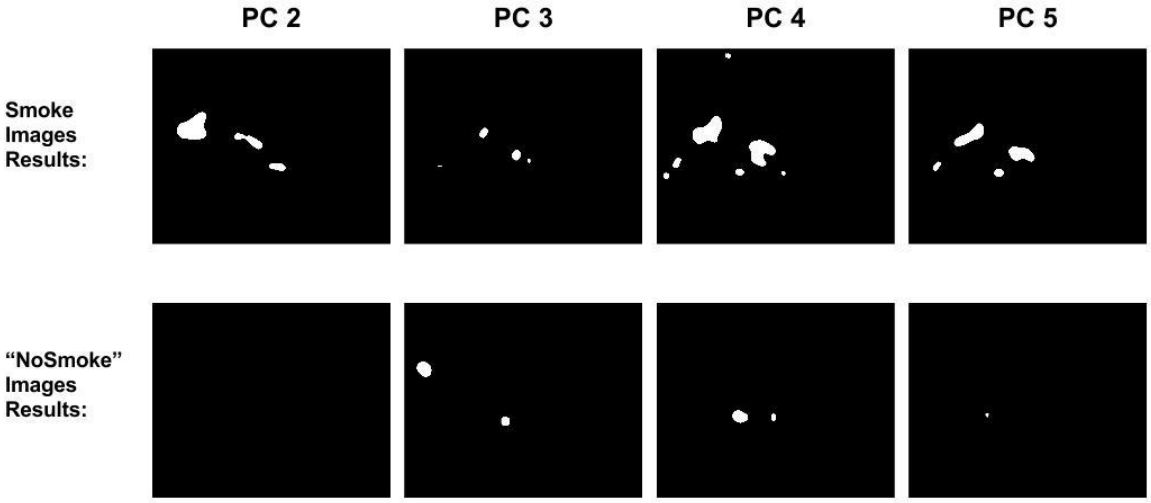

Figure 36: Results from 5 Temporal Images from Smoke and "NoSmoke" Data

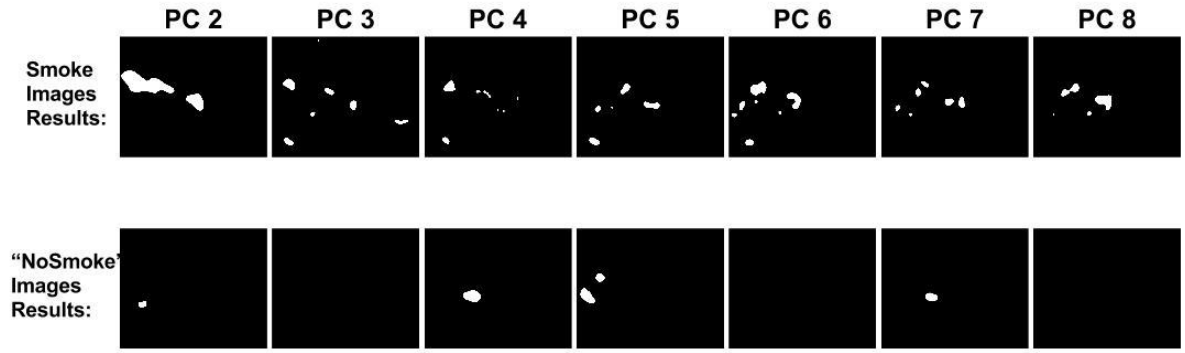

Figure 37: Results from 8 Temporal Images from Smoke and "NoSmoke" Data

6.6 Temporal Smoke signature from eigenvector

Using principal component analysis, an experimental temporal signature for moving smoke can be determined. Since all principal component images are linear combination of the original temporal images, the eigenvector associated with the principal component image that displays the temporal smoke most can be identified. The following temporal signature or "recipe" is based off of principal component image 2 of 5 based off of images 2 seconds apart.

$$I_{Temporal\ Signature} = -.49 * I_1 + -.05 * I_2 + -.27 * I_3 - .01 * I_4 + .82 * I_5 \quad (9)$$

Which can be further simplified to:

$$I_{Temporal\ Signature} = -.49 * I_1 + -.27 * I_3 + .82 * I_5 \quad (10)$$

This temporal smoke signature can be used to bypass a complete principal component transformation for a lowered processing time.

6.7 Conclusion: Discussion of tuning constant, wind, and POV

Chapter 6 provides a method to autonomously segment the temporal variance of smoke from sequences of images in time. Principal component images are generated from a principal component transform from a set of temporal images. The intensity spectrum of each principal component image is mapped to a Gaussian distribution to

determine the expectation and standard deviation and a threshold value is determined using the mean, standard deviation, and a tuning constant based on the fire scene. Results are then cleaned using median filtering. From this study, the best results come from 5 temporal images 2 seconds apart with a tuning constant of 4.5.

The tuning constant A, ultimately determines the accepted level of temporal variance. A fire scene with song amounts of wind resulting in large movement of smoke would require a large value for the tuning constant. On the other hand, smoke in a fire scene that is very far away occupies much less space in the field of view of the camera and effectively results in less temporal variance. A lower tuning constant would allow for the sensitivity of smoke in this case.

Chapter 7: Conclusion

7.1 Results

Throughout this study, there have been many strong conclusions. In chapter 4, the infrared channels are shown to contain little spectral features associated with smoke. Chapter 4 and chapter 6 also show that the strongest smoke signature results from the use of the blue spectral band. These two results are in conjunction with each other as the blue spectral band is the furthest away from the infrared spectral band in frequency. The feature space location of the smoke is determined to be unique; however, many other objects in the frame also occupy and overlap through this feature space. From this, it is determined that smoke can only be identified up to a certain confidence based on its spectral features.

Difference image analysis in chapter 5 suggests at the ability to successfully segment the temporal variance of the smoke. However, this requires long periods of time between successive temporal frames (5 seconds minimum) and this process is susceptible to many false alarm rates as is shown in chapter 5.

An autonomous selective threshold adjustment scheme is shown to successfully identify smoke in chapter 6. The process requires 5 temporal images with a minimum period of 2 seconds between each image. The eigenvector associated with principal component images that greatly accentuate the smoke is treated as the temporal smoke signature, and can be used as a recipe to combine temporal images and reduce processing time by avoiding an entire principal component transform.

7.2 Weaknesses

Ultimately, the success in identifying the location of smoke is based on the smoke's temporal variance. It is not guaranteed that the smoke will be the only moving object in the fire scene. Birds, cars, and vibrations provide sources for false positives in the temporal domain. A saving grace of the data collected in this fire scene is that the size of the smoke plume is much

larger compared to other moving object in the field of view. Median filtering helps to filter out small areas of temporal variance. Nonetheless, the algorithmic process proposed in this study is still somewhat susceptible to large areas of temporal variance resulting from moving objects other than smoke.

7.3 Future Work

There is potential for more confidence in the identification of smoke by combing methods presented in this study. Areas with large temporal variance can be verified in the feature space as smoke or something different. This combats the weaknesses addressed in section 7.2 by eliminating possible false positives that have the same temporal signature of smoke but a different spectral signature. In order for smoke to be confidently identified, the temporal and spectral variance must be correct.

The verification of the selective threshold adjustment scheme on data that contains no smoke in the atmosphere would prove to be useful. This study attempted to run processes on data containing no smoke in chapter 6, but positive results from this data and further visual investigation suggests that there are still small traces of smoke located in the atmosphere. The success of the algorithmic process would become much more valuable when true negatives have been verified.

BIBLIOGRAPHY

[1] Stipanicev, D.; Bugaric, M.; Bodrozic, L.; , "Integration of forest fire video monitoring system and Geographic Information System," ELMAR, 2009. ELMAR '09. International Symposium , vol., no., pp.49-52, 28-30 Sept. 2009

[2] Changwoo Ha; Gwanggil Jeon; Jechang Jeong; , "Vision-Based Smoke Detection Algorithm for Early Fire Recognition in Digital Video Recording System," Signal-Image Technology and Internet-Based Systems (SITIS), 2011 Seventh International Conference on , vol., no., pp.209-212, Nov. 28 2011-Dec. 1 2011

[3] Mustafa, M.; Taib, M.N.; Murat, Z.H.; Lias, S.; , "GLCM texture feature reduction for EEG spectrogram image using PCA," Research and Development (SCOReD), 2010 IEEE Student Conference on , vol., no., pp.426-429, 13-14 Dec. 2010

[4] Jing Huang; Jianhui Zhao; Weiwei Gao; Chengjiang Long; Lu Xiong; Zhiyong Yuan; Shizhong Han; , "Local binary pattern based texture analysis for visual fire recognition," Image and Signal Processing (CISP), 2010 3rd International Congress on , vol.4, no., pp.1887-1891, 16-18 Oct. 2010

[5] Vasyukov, V.; Kalennikova, E.; , "An adaptive procedure of smoke and background discrimination in the early fire detection video system," Strategic Technology (IFOST), 2011 6th International Forum on , vol.2, no., pp.844-847, 22-24 Aug. 2011

[6] Schowengerdt, Robert A. (2007). Remote sensing: models and methods for image processing (3rd ed.). Academic Press. p. 2. ISBN 978-0-12-369407-2

[7] Angayarkkani, K.; Radhakrishnan, N.; "Efficient Forst Fire Detection System: A Spatial Data Mining and Image Processing Based Approach," Computer Science and Network Security, 2009., IJCSNS International Journal of, vol. 9 , no. 9, pp. 100-106, Mar 2009

[8] Calle, A.; Casanova, J.L.; Moclan, C.; Romo, A.J.; Cisbani, E.; Costantini, M.; Zavagli, M.; Greco, B.; , "Latest algorithms and scientific developments for forest fire detection and monitoring using MSG/SEVIRI and MODIS sensors," Recent Advances in Space Technologies, 2005. RAST 2005. Proceedings of 2nd International Conference on , vol., no., pp. 118- 123, 9-11 June 2005

[9] Mazzeo, G.; Marchese, F.; Filizzola, C.; Pergola, N.; Tramutoli, V.; , "A Multitemporal
Robust Satellite Technique (RST) for Forest Fire Detection," Analysis of Multi-temporal Remote Sensing Images, 2007. MultiTemp 2007. International Workshop on the , vol., no., pp.1-6, 18-20 July 2007

[10] Krstinić, D.;, "Automatic Forest Fire detection in Visible Spectra, " Information and Intelligent Systems, 2010., Proceedings of the 21st Central European Conference on, vol., no.,pp.369-375, 22-24 Sept 2010

[11] Cappellini, V.; Mattii, L.; Mecocci, A.; , "An intelligent system for automatic fire detection in forests," Image Processing and its Applications, 1989., Third International Conference on , vol., no., pp.563-570, 18-20 Jul 1989

[12] Jones, T.A.; Christopher, S.A.; , "Multispectral Analysis of Aerosols Over Oceans Using Principal Components," Geoscience and Remote Sensing, IEEE Transactions on , vol.46, no.9, pp.2659-2665, Sept. 2008

[13] Viktor Zubko; Yoram J. Kaufman; Richard I. Burg; J. Vanderlei Martins; , "Principal Component Analysis of Remote Sensing of Aerosols Over Oceans," Geoscience and Remote Sensing, IEEE Transactions on , vol.45, no.3, pp.730-745, March 2007

[14] Davenport Tim, "Early Forest Fire Detection Using Texture Analysis of Principal Components from Multispectral Video," California Polytechnic State University, San Luis Obispo, CA, MS Thesis 2012.

[15] Davenport Tim, "EFFD Summary," unpublished.

[16] Liu, H., Motoda H. (1998) Feature Selection for Knowledge Discovery and Data Mining., Kluwer Academic Publishers. Norwell, MA, USA. 1998.

[17] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (PDF). Philosophical Magazine 2 (11): 559–572.

[18] Charles A. Poynton (2003). Digital Video and HDTV: Algorithms and Interfaces. Morgan Kaufmann. pp. 260, 630. ISBN 1-55860-792-7.

[19] P. E. Trahanias and A. N. Venetsanopoulos, "Color image enhancement through 3-D histogram equalization," in Proc. 15th IAPR Int. Conf. Pattern Recognition, vol. 1, pp. 545–548, Aug.-Sep. 1992.

[20] M. Strianese (2015). Firefighting Applications [Online]. Available: http://pr-infrared.com/about-thermal-imaging/firefighting-applications/

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; tictime = tic;
%-------------------------------------------------------------------------%
%% User Input: Source & Destination directory locations
% Source location: raw video files
vidfldr = ['F:\TestFolder1\'];
% Target directory: raw frames will be stored here
frmfldr = 'F:\Fire_Data_Target_Directory69';
%% Initialize & Allocate
MWcols = 640;
LWcols = 640;
nrows = 480;
frmnum = 0;
fps = 30;

% filename format: date_timestamp_fuel.raw
vidfiles = {'2011-06-18_102125_LW-MW-14bit_charcoal1.raw'};
vidlabels = {'charcoal1'};

% mkdir(LWfrmfldr);
% mkdir(MWfrmfldr);

read_attempt = 0;
%% Video Processing
for i=1:1:numel(vidfiles) %step through each stream raw file
```

```
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 'CLWIR/' vidlabel '/'];
MWsavefldr = [frmfldr 'CMWIR/' vidlabel '/'];

mkdir(LWsavefldr); % make folder for each lwir video
mkdir(MWsavefldr); % make folder for each mwir video
fid = fopen(vidpath,'r'); % open file for read access

display('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)
% 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)53
% t(4/60) | 3 | 3 <=
fseek(fid,2*(MWcols+LWcols)*nrows,'cof');

% separate bands (lw on left, mw on right) &
% divide each pixel by 4 to right shift twice
LWframe = dualfrm(1:640,:)./4;
MWframe = dualfrm(641:1280,:)./4;
% save individual frame with the following file name
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); % close file
```

```
 fclose(MWfsav); % close file
 end
 end

 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 of Script ------------------------------------------------------%
toc(tictime)
%--------------------------------------------------------------------%
```

```
%% Matlab Script ----------------------------------------------------%
% Script : UCLWIRvid2frame.m
% Author : Tim Davenport
% Date : 10/1/11
%
% Description : Extracts individual frames of the Uncooled Long Wave
% InfraRed (UCLWIR) .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. Resolution is 640x480
% Frame Rate is 30 fps
% Frame size is 640x480x2 = 614400 bytes
%
% Revision(s) : 01/20/12 - cleaned code: updated description, comments,
and
% superfluous syntax to be more elegant/succinct.
%-------------------------------------------------------------------%
close all; clear; clc; tictime = tic;
%-------------------------------------------------------------------%

%% User Input: Source & Destination directory locations
% Source location: raw video files
vidfldr = ['F:\TestFolder\'];
% Target location: raw frames will be stored here
frmfldr = 'F:Fire_Data_Target_Directory69';

%% Initialize
band = 'UCLWIR';
ncols = 640;
nrows = 480;
fps = 30;
frmnum = 0;

% filename format: date_timestamp_fuel.raw
vidfiles = {%'2011-06-18_101351_setup.raw'
'2011-06-18_102316_charcoal1.raw'
'2011-06-18_102646_oats1.raw'};
vidlabels = {%'setup'
'charcoal1'
'oats1'};

% mkdir(frmfldr);

%% Video Processing (frame extraction)
for i=1:1:numel(vidfiles) % step thru each video stream raw file
```

```
vidtic = tic;
disp(['Opening Video : ' vidfiles{i}]);

% Update file names & locations
vidname = vidfiles{i};
vidlabel = vidlabels{i};
vidpath = [vidfldr vidname];
prefix = [band '_' vidlabel '_'];
savefldr = [frmfldr '/' band '/' vidlabel '/'];

mkdir(savefldr); % make folder for each video (or test)
fid = fopen(vidpath,'r'); % open file for read access

disp('Extracting frames')
% Extract successive frames from each raw video stream until eof
while ~feof(fid)

% Read in raw video frame of 640x480 = 307200 16-bit pixels
frm = uint16(fread(fid,[ncols, nrows],'uint16'));

if ~isempty(frm)
frmnum = frmnum + 1;

% divide each pixel by 4 to right shift twice
frm = frm./4;

% save individual frame with the following file name
savepath = [savefldr prefix int2str(frmnum) '.raw'];

fsav = fopen(savepath,'w'); % open file for write access
fwrite(fsav,frm(:,:),'uint16'); % write frame with 16-bit data
fclose(fsav); % close file
end
end

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

frmnum = 0;
fclose(fid);
end

%% End of Script ------------------------------------------------------%
toc(tictime)
%--------------------------------------------------------------------------%
```

```
%% Matlab Script ---------------------------------------------------------%
% Script : visvid2frame.m
% Author : Tim Davenport
% Date : 10/31/2011
%
% Description : this script extracts all the frames from the visible
video.
%
% Notes :
% Revision(s) : 10/31/11 -- added get frame num
% 01/28/12 -- cleaned code
%-------------------------------------------------------------------------%
close all; clear; clc; tictime = tic;
%-------------------------------------------------------------------------%

%% Initialize and Allocate
nVISrows = 480;
nVIScols = 587;
fps = 30;

% Source location: raw video files
vidfldr = 'E:\Camtasia Videos\';
frmfldr = 'E:\Fire_Data_Target_DirectoryVisiblecapture-13new';
% mkdir(frmfldr);
vidfiles = {'capture-13'};

for j = 1:numel(vidfiles)
    vidtic = tic;
    disp(['Opening Video : ' vidfiles{j}]);

    visMov = VideoReader([vidfldr vidfiles{j} '.avi']);
    frmnum = get(visMov, 'NumberOfFrames');

    savefldr = [frmfldr vidfiles{j} '/'];
    mkdir(savefldr);

    %% Extract each frame
    disp('Extracting frames');
    for i = 1:frmnum

        % clc
        % disp(['extracting frame' ' ' num2str(i)]);
        % read ith frame from visible video
        frm = read(visMov, [i i]);
        % write visible frame
        savepath = [savefldr 'VISIBLE_' vidfiles{j} '_' num2str(i)
'.bmp'];
        imwrite(frm,savepath,'bmp');
    end

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

```
%% End of Script ------------------------------------------------------%
toc(tictime)
%--------------------------------------------------------------------%
```

```
%% Matlab Script ----------------------------------------------------%
% Script : alignCLWIRFOV.m
% Author : Tim Davenport
% Date : 10/31/2011
%
% Description : this script extracts all the frames from the visible
video.
%
% Notes :
% Revision(s) : 10/31/11 -- added get frame num
% 01/28/12 -- cleaned code
%-------------------------------------------------------------------%
close all; clear; clc; tictime = tic;
%-------------------------------------------------------------------%
%% User Input: Source & Destination directory locations
% Source location: raw un registered frame files
unRegfrmfldr = 'F:\Fire_Data_Target_DirectoryUCLWIR\';

% Target location: raw frames will be stored here
Regfrmfldr =
'F:\Fire_Data_Target_DirectoryUCLWIR_SpacialAllignment_test';

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

vidlabels = { 'charcoal1'
    'oats1'
    };

%% 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 ]-2';
VIS_y = [577 573 571 566 291 639 625 157 146 751 721 846 801 433 805
646 565 436 685 328 ]-27';
n     = length(VIS_x);

% the length of all the correlation lists
A      = ones ([n 3]);
A(:,1) = VIS_x;
A(:,2) = VIS_y;
cIR_x  = [299 330 343 357 381 408 158 195 486 458 131 280 397 210 112
532 461 326 358 243 ]'; % 578
cIR_y  = [361 357 356 354 239 388 379 180 178 437 419 476 459 296 457
392 349 298 406 253 ]'; % 424
abc_IR = A\cIR_x;
def_IR = A\cIR_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);
```

```
visible_x = visible_x(:);
visible_y = visible_y(:);

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

% 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 the
% [a b c] and [d e f] transformations calculated above
% 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
resmpl_IR_xy = ([abc_IR';def_IR']*[visible_x visible_y
ones(nVISrows*nVIScols,1)]')';
resmpl_IR_x = reshape(resmpl_IR_xy(:,1),nVISrows,nVIScols);
resmpl_IR_y = reshape(resmpl_IR_xy(:,2),nVISrows,nVIScols);

for i=1:length(vidlabels)
    disp(['Aligning Frames from : ' vidlabels{i} ' test']);

    % Update file names & locations
    testname  = vidlabels{i};
    unRegPath = [unRegfrmfldr testname '\']; % made change here
    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
        fid = fopen([unRegPath prefix num2str(k) '.raw'],'r');
%source file
        unRegfrm = uint16(fread(fid,[nIRcols, nIRrows],'uint16'))';
        fclose(fid);

        % 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')
%end file
        fwrite(fid,Regfrm(:,:)','uint16');
        fclose(fid)
        disp(['Frames Registered : ' num2str(k)]);
    end
end
%% End of Function ------------------------------------------------%
```

```
function img_agc = agc (img, n_rows, n_cols)


    % perform Plateau AGC on unsigned 14-bit image
    max_bit = 2^14;
    hist_bins = zeros (max_bit, 1);
    img_agc = uint8 (zeros (n_rows, n_cols));

    % set up the histogram bins, one for each 16-bit grayscale level
    for i_bin = 1:max_bit
        hist_bins(i_bin) = i_bin - 1;
    end

    % matlab calculates the image histogram
    img_hist = histc (img(:), hist_bins);

    % for Plateau Equalization, clip the bins to some plateau level
    img_hist(img_hist > 150) = 150;

    % calculate the cumulative frequency histogram from the clipped
    % histogram
    cum_hist = cumsum (img_hist);

    % scale the cumulative frequency histogram to 8-bit grayscale
    cum_hist = uint8 (255 * cum_hist / cum_hist(max_bit));

    % use the scaled cumulative frequency histogram as an intensity
    % transform table to map the original 16-bit data to 8-bit level
    for i_row = 1:n_rows
        for j_col = 1:n_cols
            img_agc(i_row,j_col) = cum_hist(img(i_row,j_col) + 1);
        end
    end

return
```