

**EARLY FOREST FIRE DETECTION USING TEXTURE, BLOB THRESHOLD,
AND MOTION ANALYSIS OF PRINCIPAL COMPONENTS**

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

Georges F. Moussa

December 2012

© 2012

Georges F. Moussa

ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Early Forest Fire Detection using Texture, Blob Threshold, and Motion Analysis of Principal Components

AUTHOR: Georges F. Moussa

DATE SUBMITTED: December 2012

COMMITTEE CHAIR: Dr. John Saghri, Professor

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

COMMITTEE MEMBER: Dr. Jane Zhang, Associate Professor

ABSTRACT

EARLY FOREST FIRE DETECTION USING TEXTURE, BLOB THRESHOLD AND MOTION ANALYSIS OF PRINCIPAL COMPONENTS

Georges Moussa

Forest fires constantly threaten ecological systems, infrastructure and human lives. The purpose behind this study is minimizing the devastating damage caused by forest fires. Since it is impossible to completely avoid their occurrences, it is essential to accomplish a fast and appropriate intervention to minimize their destructive consequences. The most traditional method for detecting forest fires is human based surveillance through lookout towers. However, this study presents a more modern technique. It utilizes land-based real-time multispectral video processing to identify and determine the possibility of fire occurring within the camera's field of view. The temporal, spectral, and spatial signatures of the fire are exploited. The methods discussed include: (1) Range filtering followed by entropy filtering of the infrared (IR) video data, and (2) Principal Component Analysis of visible spectrum video data followed by motion analysis and adaptive intensity threshold. The two schemes presented are tailored to detect the fire core, and the smoke plume, respectively.

Cooled Midwave Infrared (IR) camera is used to capture the heat distribution within the field of view. The fire core is then isolated using texture analysis techniques: first, range filtering applied on two consecutive IR frames, and then followed by entropy filtering of their absolute difference.

Since smoke represents the earliest sign of fire, this study also explores multiple techniques for detecting smoke plumes in a given scene. The spatial and temporal variance of smoke plume is captured using temporal Principal Component Analysis, PCA. The results show that a smoke plume is readily segmented via PCA applied on the visible Blue band over 2 seconds sampled every 0.2

seconds. The smoke plume exists in the 2nd principal component, and is finally identified, segmented, and isolated, using either motion analysis or adaptive intensity threshold.

Experimental results, obtained in this study, show that the proposed system can detect smoke effectively at a distance of approximately 832 meters with a low false-alarm rate and short reaction time. Applied, such system would achieve early forest fire detection minimizing fire damage.

Keywords: Image Processing, Principal Component Analysis, PCA, Principal Component, PC, Texture Analysis, Motion Analysis, Multispectral, Visible, Cooled Midwave Infrared, Smoke Signature, Gaussian Mixture Model.

ACKNOWLEDGMENTS

I acknowledge the following people for their efforts and contributions to my thesis. It would not have been completed without them.

Special thanks to my thesis advisors, Dr. John Saghri and Dr. John Jacobs, for their guidance, patience, support, encouragement and stimulation in my graduate study.

I thank Dr. Jane Zhang for her willingness and graceful acceptance to serve as a committee member on such a short notice.

I thank Raytheon Employees: Steve Botts, Marc Bauer, Chris Tracy, and Dana Schneider for their support during the June 2011 Fire Test as well as Gary Hughes from FLIR for his help with IR cameras.

I thank my parents, siblings, family, and friends for their encouragement and support in helping me complete my thesis.

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES	x
Chapter 1 Introduction.....	1
1.1 Motivation	1
1.2 Background.....	1
1.3 Proposed Early Forest Fire Detection.....	4
1.4 Thesis Outline	5
Chapter 2 Data Acquisition and Preprocessing.....	6
2.1 Data Acquisition	6
2.2 Preprocessing	6
Chapter 3 Fire Core Detection using Texture Analysis on Cooled Midwave IR Frames.....	7
3.1 Introduction.....	7
3.2 Range Filtering:.....	8
3.3 Standard Deviation Filtering:.....	10
3.4 Entropy Filtering:.....	12
3.5 Absolute Difference:.....	15
3.6 Combination of Multiple Filter Types:.....	16
Chapter 4 Visible Preprocessing.....	17
Chapter 5 Principal Component Analysis.....	20
5.1 Principal Component Analysis (PCA) Introduction	20
5.2 Principal Component Analysis Theory/Procedure	20
5.3 Adaptive Principal Component Analysis (PCA).....	36
5.4 Comparing Adaptive to Non-Adaptive Principal Component (PC2).....	41
Chapter 6 Segmenting Foreground and Background.....	44
6.1 Introduction.....	44
6.2 Tracking Objects Using Gaussian Mixture Models	44
6.2.1 Introduction.....	44
6.2.2 Theory.....	46
6.2.3 Results	48
6.3 Threshold and Median Filtering	50
6.3.1 Introduction.....	50

6.3.2	Method.....	50
6.3.3	Results	52
Chapter 7	Principal Component Analysis Modification	55
7.1	Introduction.....	55
7.2	Blur Theory	57
7.2.1	Gaussian Blur Mask	57
7.2.2	Median Filter Applied to Gaussian Blurred Image	58
7.3	Modification of PC Image to Enhance Smoke Plume Detection	59
7.4	Spectral-Temporal Principal Component Analysis	62
Chapter 8	Conclusion	64
8.1	Final Results for Extracting Smoke Plumes.....	64
8.1.1	Gaussian Mixture Model Results.....	64
8.1.2	Adaptive Threshold and Median Filtering Results	66
8.2	Future Work	68
	Bibliography.....	69
	Appendix A: MATLAB Functions and Scripts	72

LIST OF TABLES

Table 1: Comparison between Different Video Based Smoke Detection Methods	4
Table 2: Camera/Sensor Specifications	6
Table 3: Range Filtering Sample Calculation Method	8
Table 4: Range Filtering Sample Calculation Results.....	9
Table 5: Standard Deviation Filtering Sample Calculation Results.....	10
Table 6: Standard Deviation Filtering Sample Calculation Method	11
Table 7: Entropy Filtering Sample Calculation Method.....	13
Table 8: Entropy Filtering Sample Calculation Results	14
Table 9: Full Dynamic Range Scaling Example.....	32

LIST OF FIGURES

Figure 1: Proposed Early Forest Fire Detection System	5
Figure 2: Original Midwave IR Image “Oats” Frame 2500.....	7
Figure 3: Output Image of Range Filter Applied to the Original Midwave IR Image.....	9
Figure 4: Output Image of Standard Deviation Filter Applied to the Original Midwave IR Image.....	11
Figure 5: Output Image of an Entropy Filter Applied to the Original Midwave IR Image	14
Figure 6: Original Midwave IR Image on the left, and Absolute Difference Image on the right.....	16
Figure 7: Midwave IR Frame 2500 on the Left, and Output Image showing Fire Plume Location Frame 2500 on the Right.....	16
Figure 8: RGB Visible “Capture 13” Frame 8003	17
Figure 9: Red, Green, and Blue Band of Frame 8003, Respectively	18
Figure 10: RGB Visible “Capture 13” Frame 8003 with Small FOV around BBQ Pit	18
Figure 11: RGB Visible “Capture 13” Frame 8003 with Small FOV around BBQ Pit,	19
Figure 12: Blue Band Visible “Capture 13” Frame 8003 with Small FOV around BBQ Pit.....	19
Figure 13: Frame Shifts per PCA Computation.....	22
Figure 14: 9 Consecutive Blue Band “Capture 13” Starting Frame 8003 used in One PCA Computation	24
Figure 15: Non Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	29
Figure 16: Histogram for Non Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	30
Figure 17: Non Adaptive Principal Components, After Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	34
Figure 18: Histogram for Non Adaptive Principal Components, After Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	35
Figure 19: Adaptive PCA Sub-Block Procedure.....	36
Figure 20: Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	37
Figure 21: Histogram for Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	38

Figure 22: Adaptive Principal Components, After Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9.....	39
Figure 23: Histogram for Adaptive Principal Components, After Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	40
Figure 24: Non Adaptive PC2 Frame 8000	41
Figure 25: Adaptive PC2 Frame 8000	41
Figure 26: Original Visible Frame, Absolute Difference of Two Consecutive Visible Frames, and Scaled PC2 of Blue Band, from Left to Right, Respectively.....	42
Figure 27: Temporal PC2 of Red, Green, and Blue Bands, From Left to Right, Respectively, at some instant of time	43
Figure 28: Temporal PC2 of Red, Green, and Blue Bands, From Left to Right, Respectively, at another instant of time	43
Figure 29: Pixels Intensities are modeled by a Mixture of Gaussian Distributions.....	46
Figure 30: Temporal Blue Band PC2 Frame 7911 “Oats1”	49
Figure 31: Mask Produced Using GMM Frame 7911 “Oats1”	49
Figure 32: Temporal Blue Band PC2 Frame 7831 “Oats1”	49
Figure 33: Mask Produced Using GMM Frame 7831 “Oats1”	49
Figure 34: Temporal Blue Band PC2 Frame 7832 “Oats1”	49
Figure 35: Mask Produced Using GMM Frame 7832 “Oats1”	49
Figure 36: Temporal Blue Band PC2 Frame 7890 “Oats1”	50
Figure 37: Histogram of the above Image using MATLAB Contrast/Threshold Tool	50
Figure 38: Segmenting the Dark Smoke Plume Pixels.....	51
Figure 39: Histogram Showing the Threshold Window that Segments the Dark Smoke Plume Pixels	51
Figure 40: Segmenting the Bright Smoke Plume Pixels.....	51
Figure 41: Histogram Showing the Threshold Window that Segments the Bright Smoke Plume Pixels	51
Figure 42: Invert the Polarity of Figure 38	52
Figure 43: Add Both Windows of the Segmented Smoke Plume to Produce the Full Smoke Plume (Figure42+Figure40)	52
Figure 44: Superimposed Mask using Threshold over both Respective PC2 and RGB Image.....	52
Figure 45: Superimposed Mask using Adaptive Threshold over both Respective PC2 and RGB Image	54

Figure 46: On the Left, Instantaneous Total Segmented Area per frame using Adaptive Threshold Technique; On the Right, 20 Frames Moving Average of Total Segmented Area per Frame, using Adaptive Threshold Technique	54
Figure 47: PC2 Frame 7888 “Oats1” on the left, and its Histogram on the Right	56
Figure 48: PC2 Frame 7890 “Oats1” on the left, and its Histogram on the Right	56
Figure 49: Gaussian Distribution Kernel Standard Deviation 1.0	57
Figure 50: Blue Band Frame 7888 “Capture 13” “Oats1”	57
Figure 51: Gaussian Blur applied to Blue Band Frame 7888 “Capture 13” “Oats1”	57
Figure 52: Blue Band Frame 7888 “Capture 13” “Oats1”	58
Figure 53: Final Blurred Blue Band Frame 7888 “Capture 13” “Oats1”	58
Figure 54: Absolute PC2 Blue Band Frame 7908 “Capture 13”	60
Figure 55: Histogram for the Image displayed on the left	60
Figure 56: Absolute PC2 Blue Band Median Blur size 3x3 Frame 7980.....	61
Figure 57: Histogram For the Image displayed on the left.....	61
Figure 58: Absolute PC2 Blue Band Median Blur size 10x10 Frame 7980.....	61
Figure 59: Histogram For the Image displayed on the left.....	61
Figure 60: Spectral-Temporal Principal Component Analysis, Using RGB; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9	63
Figure 61: Absolute Blue Band PC2 of Frame 7901 “Oats1”	65
Figure 62: GMM Mask of the Image on the Left	65
Figure 63: Absolute Blue Band PC2 of Frame 7977 “Oats1”	65
Figure 64: GMM Mask of the Image on the Left	65
Figure 65: Adaptive Threshold of Figure 61	66
Figure 66: 10x10 Median Filter of Image on the Left	66
Figure 67: Adaptive Threshold of Figure 63	66
Figure 68: 10x10 Median Filter of Image on the Left	66
Figure 69: Mask in Figure 66 Superimposed onto Figure 61.....	67
Figure 70: Mask in Figure 66 Superimposed onto the respective RGB Captured Image	67
Figure 71: Total Area of Segmented Blobs using Adaptive Threshold and Median Filtering of Absolute Temporal Blurred PC2 per Frame.....	68

Chapter 1 Introduction

1.1 Motivation

Forest fires have always been a natural source of destruction. They cause ecological and economical damage, along with risking humans' lives. The development of an Early Forest Fire Detection System can reduce property damage, pollution, and it can also save lives. For timely emergency response, it is necessary to detect forest fires in the early stages of combustion [5]. The United States spends on average \$900 million/year to fight forest fires [12]. These forest fires cause \$733 million/year in property and crop damage [13]. Eliminating this cost can help reduce both the state and national budget deficits.

1.2 Background

Conventional point-based smoke sensors detect the presence of smoke particles by ionization or photometry [7]. Since smoke particles must be transported from the fire to the detector, such system fails to give an early fire warning in the outdoors and open areas; hence, it cannot be used to detect forest fires. Wildfire detection was traditionally based on human surveillance from lookout towers at an effective vantage point. Modern technology offers more advanced alternatives. Smoke detection using surveillance cameras has become more active. They are based on machine vision, image processing, and pattern recognition techniques. Developing these algorithms has been a hot topic in computer vision over the last decade. Video-based smoke detection systems provide advantages over conventional methods, such as fast response, non-contact, and lack of spatial limits [15].

Land-based infrared (IR) and/or visible cameras may be installed in a step-stare-mode configuration in potential fire-prone areas [2]. The sequences of video frames from each camera may then be digitally processed to determine if there is a fire within the cameras' Field Of View (FOV). A group from the University of Nevada, Reno is blending old and new schemes by installing a remote

360-degree, solar powered camera that streams video to a public website, where thousands of volunteers can check on it, and manually raise an alarm [10]. Similarly, Integral Forest Fire Monitoring System, INPAS, is a Croatian land-based system relying on high resolution cameras. It requires constant human interaction and real-time meteorological data for processing [2].

Satellite solutions can work, but they have their own limitations. They can suffer from atmospheric conditions and coarse resolution in space and time, resulting in high false alarm rates. The existing land/satellite based detection systems rely on measurement and discrimination of fire plume directly from its infrared, and its visible reflectance imagery. These satellite based detection systems tend to be relatively expensive, and they are unable to perform 24/7 surveillance of the same geographical area. For example, the well-known MODIS algorithm, the subject of many current papers, implemented on the TERRA satellite and launched in 1999, has a spatial resolution of 250 m: hardly suitable for detecting small forest fires. According to a study on forest fires in Brazil, satellite systems are better suited to conservative fire detection algorithms that detect 30-40% of fires, accounting for 80-99% of burned biomass. Early detection for tactical response can only effectively be done from ground-based systems. In general, more domains and features, if combined judiciously, can add robustness to automatic fire detection system. Recent schemes tend to exploit multiple dimensions, i.e., temporal, spectral, and spatial, to detect forest fires. Unfortunately, these come at the cost of increased sensing system requirements. If an adequate system could be demonstrated to detect smoke or heat plume from a single camera, it would reduce cost and complexity, be more easily maintained, more light-weight, and more widely accessible [1].

Balance between early fire warning and minimum false alarm is essential in fire detection systems. In order to reach such balance, better “fire signatures” must be identified and implemented. “Fire signatures” are the local environmental conditions that change between non-fire and fire conditions. The goal is selecting conditions for sensing that appear as early as possible, and are present

at levels sufficiently above those at normal, non-fire conditions, to minimize false alarms [6]. These changes of conditions are called fire signatures, such as fire core, heat plume, and smoke plume. In video based systems, the fire core represents the source or flames of the fire; the heat plume represents the surrounding pixels of the fire core where the air is still hot and can be detected using IR sensors; lastly, smoke plume is relatively cold air bursting from the fire, and drifting with the wind [1].

Video smoke detection still has great technical challenges, since its current performance is inferior to those of traditional particle-sampling based detectors, in terms of detection rate and false alarm rate, mainly due to the variability in smoke density, scene illumination, diverse background, interfering objects, and generally because smoke is difficult to model, and most of the common image processing methods do not characterize smoke well [7].

There are several existing video-based techniques to detect smoke; however, most of them are based on near distance smoke. Unfortunately, smoke at far distances exhibits different spatial and temporal characteristic than nearby smoke [8], [9]. This demands specific methods explicitly developed for smoke detection at far distances, rather than using existing nearby smoke detection methods.

There are several approaches on automatic detection of forest fires in the literature. Some of the approaches are directed towards detection of the flames, using infrared and/or visible-range cameras, and some others aim at detecting the smoke [16] [17], [18] [19]. In order to reduce false alarms, multiple characteristic of fires are taken into consideration. In the visible spectrum, most existing wildfires smoke detection algorithms can be split in two categories: Visual and Dynamic smoke characteristic. Visual smoke characteristics represent its morphological and chromatic properties; while Dynamic smoke characteristic describe the behavior of propagation of smoke plumes. The most prominent visual smoke characteristic is its color; however, since smoke can have a very wide range of pixel characteristics, making it difficult to distinguish smoke from image background based solely on the color information. In conclusion, the color criterion alone is not enough to develop an accurate

smoke detection algorithm [25], [26]. On the other hand, Dynamic smoke characteristics consist of features that change over time, such as the shape, expansion, motion, orientation, and/or variations in its color. Other frequently used fire detection techniques include energy analysis [27] [28] [29], and flicker detection [7] [14] [30] [31] [32]. Both approaches also rely on the temporal behavior of smoke and flames. However, the flicker frequency of smoke varies with time; therefore, smoke-flicker detection technique results in high false alarms. Some of the techniques that use visible range cameras are summarized in Table 1 below.

Table 1: Comparison between Different Video Based Smoke Detection Methods

	Color Space Detection	Shape Analysis	Motion Detection	Texture Analysis	Smoke Behavior Analysis	Contour Analysis	Smoke Detection	Fire Detection
[20]	No	Yes	Yes	No	No	Yes	Yes	No
[21]	RGB/HSI	Yes	Yes	No	Yes	No	Yes	No
[22]	RGB/YUV	Yes	Yes	Yes	Yes	Yes	Yes	No
[23]	No	Yes	Yes	Yes	Yes	Yes	Yes	No
[24]	No	No	Yes	No	Yes	No	Yes	No
Proposed Technique	RGB	No	Yes	Yes	Yes	No	Yes	Yes

RGB: color space made from Red, Green, and Blue light that are added together in various combinations to reproduce a various colors.

HSI: color space that consists of Hue, Saturation, and Intensity; it is mostly used in computer vision applications.

YUV: color space that takes into account human perception.

1.3 Proposed Early Forest Fire Detection

A land-based real-time multispectral video processing is proposed to identify, and determine the possibility of fire occurring within a constant camera’s field of view. The temporal, spectral, and spatial signatures of the fire are exploited. The methods discussed include: (1) Range filtering followed by entropy filtering of the infrared video data, and (2) Temporal Principal Component Analysis of

visible spectrum video data, followed by motion analysis and adaptive intensity threshold. The two schemes presented are tailored to detect the fire core, and the smoke plume, respectively.

The development of Early Forest Fire Detection System provides an automated first alert to forest fire, by digitally detecting smoke plume identified using its temporal and spatial signature, in order to reduce damage caused by forest fire. Figure 1 below illustrates the block diagram of the proposed technique.

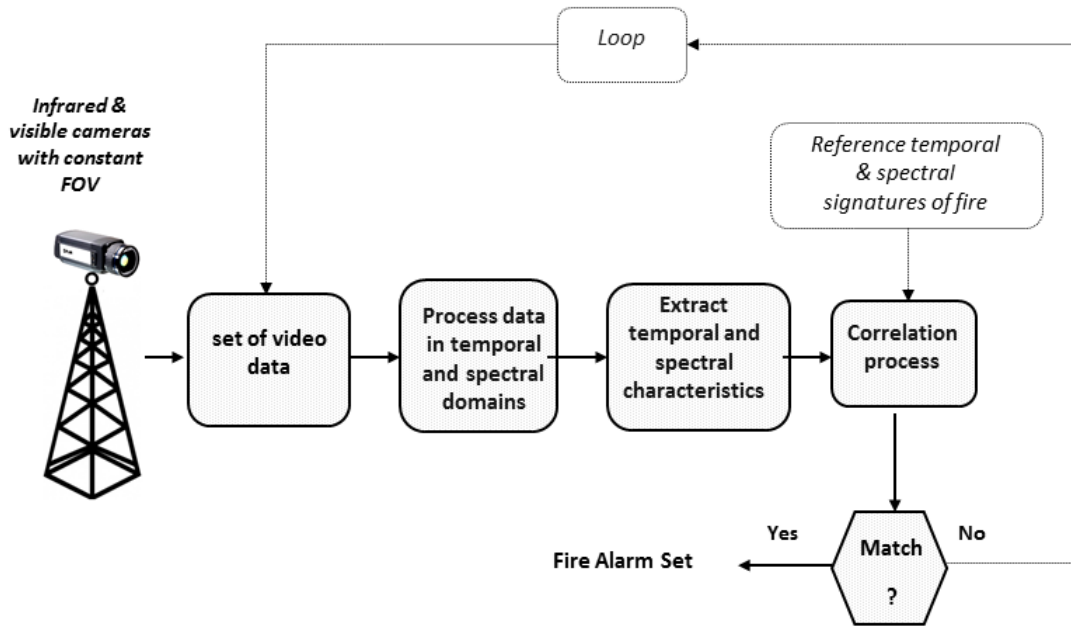


Figure 1: Proposed Early Forest Fire Detection System

1.4 Thesis Outline

The thesis is structured as follows: Chapter 2 focuses on the data acquisition and preprocessing stage of all the recorded videos. Chapter 3 introduces a method for detecting the fire core. Chapter 4 presents the preprocessing stage of the visible data. Chapter 5 illustrates Principal Component Analysis method to segment smoke plumes. Chapter 6 conveys two techniques for separating foreground from background on the second Principal Component. Chapter 7 offers a modification on Principal Component Analysis discussed in Chapter 5. Finally, Chapter 8 delivers final results and concludes the thesis by providing an overall summary of the results.

Chapter 2 Data Acquisition and Preprocessing

2.1 Data Acquisition

A controlled fire test was performed for the purpose of collecting temporal and multi-spectral smoke plumes and heat plumes data. Four different sensors were used: Visible, cooled mid-wave infrared, cooled long-wave infrared, and uncooled long-wave infrared. The sensors' specifications are summarized in the table below. The fire test generated over 300 Gigabytes of raw video. The fuels were incrementally added to simulate a growing fire, starting with a small amount of smoke plume that progressively increased covering a larger volume within the sensor's field of view.

Table 2: Camera/Sensor Specifications

Sensor	Visible	Cooled MWIR	Cooled LWIR	Uncooled LWIR
Resolution	1086 x 873	640 x 480	640 x 480	640 x 480
Bit Depth	8-bit	14-bit	14-bit	14-bit
FPS	10	30	30	30

2.2 Preprocessing

Since the four sensors did not have the same specifications, it was necessary to preprocess all the frames before any multi-spectral analysis. The mismatch in the resolution between the sensors resulted in different field of view for each sensor. This issue was solved by using image registration of all sensors to the visible frame, with resolution of 1086 x 876 along with multiple spatial control points (recognizable features within the field of view). Also, only the visible sensor was time stamped, a similar method was used to temporally align all the sensors. For more information on the fire test location, distance between the fire and the sensors, fuels that were burned, data format, time alignment, and field of view matching, please see [1],[3],[4].

Note that only the visible and Cooled MWIR frames were used in this study.

Chapter 3 Fire Core Detection using Texture Analysis on Cooled Midwave IR Frames

3.1 Introduction

Cooled MidWave IR cameras measure heat intensities within the camera's field of view, also known as thermal imaging. Such sensor can be used to identify hot spots and objects. For example, it can be used to detect fires; however, as seen in Figure 2 below, during the day, roof tops and buildings, can have the same range in pixel intensity values, due to pixel saturation. Therefore, intensity threshold and median averaging methods alone cannot isolate the fire location. Thus, this can cause high false fire detection rate. For this reason, instead of exclusively relying on pixel intensity values, it is important to investigate the image texture and its contents. Texture analysis can be helpful when objects in an image are more characterized by their texture than by intensity, and traditional threshold techniques cannot be as effective. Texture analysis uses the following methods: Range filtering, Standard Deviation filtering, and Entropy filtering. All these functions operate in a similar technique: they define a neighborhood around the pixel of interest, then calculate the statistics for that neighborhood, and finally use that value as the value of the pixel of interest in the output image. For pixels on the borders, these functions use symmetric padding. In symmetric padding, the values of padding pixels are a mirror reflection of the border pixels in the original image [1].



Figure 2: Original Midwave IR Image "Oats" Frame 2500

3.2 Range Filtering:

Range filtering is a method used in texture segmentation of images. It locates the boundaries and edges of objects within the field of view. This is done by examining the range in intensity values within a desired kernel size, and replacing the middle pixel by range intensity value of the neighborhood pixels. Within the desired kernel, the maximum and minimum intensity values are determined, and then the image's pixel overlaying the middle pixel of the kernel is replaced by the difference between the maximum and minimum intensity values. Figure 3 shows the output image after range filtering of Figure 2. As shown below, this method revealed the edges and texture of the image contents. Additionally, it highlights the fire location and increases the size of the heat plume. The kernel dimension used in this example is 3x3. The following example shows how to determine the pixel values in a Range filtered output image.

Table 3: Range Filtering Sample Calculation Method

			Maximum value in neighborhood	Minimum value in neighborhood		
Original Image						
0	74	0	255	0	0	0
0	50	55	0	214	125	189
124	255	0	48	0	26	255
0	0	0	0	0	0	0
0	178	27	0	0	0	0
255	213	184	0	77	0	0

Range = Max value – Min value = 255 – 0 = 255

Range Filtered Image						
74	74	255	255	255	214	
255	255	255	255	255		

Table 4: Range Filtering Sample Calculation Results

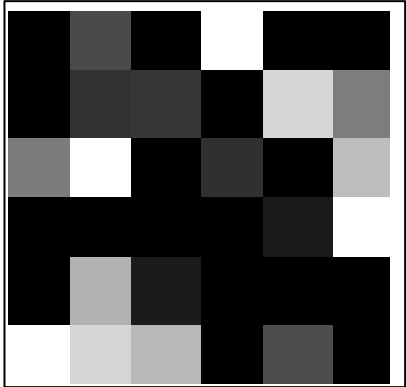
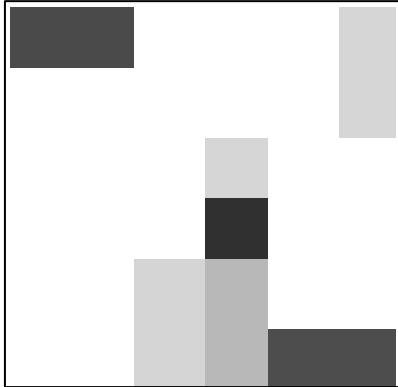
Original Image	Range Filtered Image
$\begin{bmatrix} 0 & 74 & 0 & 255 & 0 & 0 \\ 0 & 50 & 55 & 0 & 214 & 125 \\ 124 & 255 & 0 & 48 & 0 & 189 \\ 0 & 0 & 0 & 0 & 26 & 255 \\ 0 & 178 & 27 & 0 & 0 & 0 \\ 255 & 213 & 184 & 0 & 77 & 0 \end{bmatrix}$	$\begin{bmatrix} 74 & 74 & 255 & 255 & 255 & 214 \\ 255 & 255 & 255 & 255 & 255 & 214 \\ 255 & 255 & 255 & 214 & 255 & 255 \\ 255 & 255 & 255 & 48 & 255 & 255 \\ 255 & 255 & 213 & 184 & 255 & 255 \\ 255 & 255 & 213 & 184 & 77 & 77 \end{bmatrix}$
	



Figure 3: Output Image of Range Filter Applied to the Original Midwave IR Image

3.3 Standard Deviation Filtering:

Standard deviation filtering follows a similar approach to range filtering; however, the pixel of interest intensity value is replaced by the local standard deviation of the pixels within the kernel dimensions. This method also highlights edges and areas with high changes in intensity values. It indicates the degree of variability of pixel values in that region, whereas locations with high intensities can indicate fire locations, and a passing vehicle or flying bird, which can possibly misleadingly trigger a fire detection alarm. It is also noted that using this method, the heat plume burst forth from the fire flames. In Figure 4, the kernel used has dimensions of 3x3. The following example shows how to determine the pixel values in a Standard Deviation filtered output image.

Table 5: Standard Deviation Filtering Sample Calculation Results

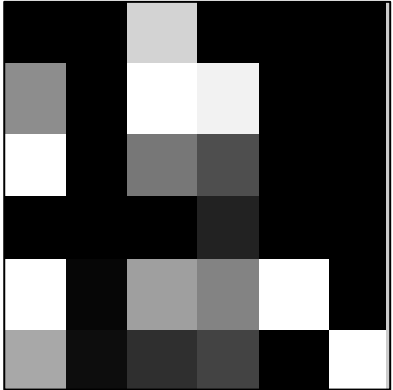
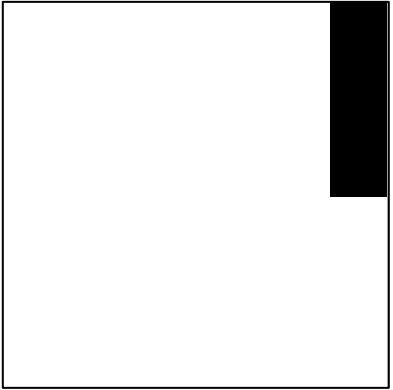
Original Image	Standard Deviation Filtered Image
$\begin{bmatrix} 0 & 0 & 211 & 0 & 0 & 0 \\ 141 & 0 & 255 & 242 & 0 & 0 \\ 255 & 0 & 119 & 78 & 0 & 0 \\ 0 & 0 & 0 & 34 & 0 & 0 \\ 255 & 6 & 159 & 131 & 255 & 0 \\ 168 & 13 & 47 & 67 & 0 & 255 \end{bmatrix}$	$\begin{bmatrix} 62 & 112 & 122 & 122 & 81 & 0 \\ 112 & 113 & 110 & 110 & 82 & 0 \\ 112 & 111 & 104 & 104 & 81 & 0 \\ 134 & 112 & 64 & 88 & 88 & 85 \\ 114 & 96 & 58 & 88 & 107 & 128 \\ 100 & 91 & 53 & 83 & 114 & 134 \end{bmatrix}$
	

Table 6: Standard Deviation Filtering Sample Calculation Method

Original Image					
0	0	211	0	0	0
141	0	255	242	0	0
255	0	119	78	0	0
0	0	0	34	0	0
255	6	159	131	255	0
168	13	47	67	0	255

$$Std = \sqrt{\frac{1}{8} \sum_{i=1}^9 (x_i - \bar{x})^2} \quad \text{where } \bar{x} = \frac{\sum_{i=1}^9 x_i}{9} = \frac{0+0+0+242+0+0+78+0+0}{9} = 35.6$$

$$Std = \sqrt{\frac{1}{8} ((242 - 35.6)^2 + (78 - 35.6)^2 + 7 \cdot (0 - 35.6)^2)} = 81$$

Standard Deviation Filtered Image					
62	112	122	122	81	0
112	113	110	110	81	

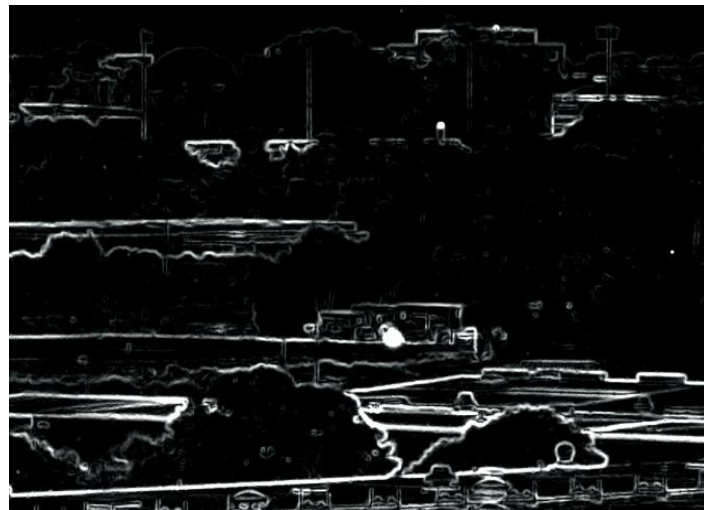


Figure 4: Output Image of Standard Deviation Filter Applied to the Original Midwave IR Image

3.4 Entropy Filtering:

Entropy filtering operates similarly to the previous two methods, first by defining a kernel/neighborhood around the pixel of interest and calculating the entropy at that pixel. Entropy is a statistical measure of randomness that can be used to characterize the texture of the neighborhood pixels. It is defined as:

$$\mathbf{Entropy} = - \sum h \times \log_2(h) \quad \text{Equation 1}$$

Where h is the neighborhood's histogram counts, also calculated as:

$$h = \frac{\mathbf{Count\ for\ a\ given\ Gray\ Level}}{\mathbf{Total\ Neighborhood\ Pixels}} \quad \text{Equation 2}$$

Once calculated, the intensity value is then assigned to the output pixel of interest. Entropy filtering consists of measuring the randomness of intensity values within a neighborhood, and then outputs either a logical 1 for locations with randomness, or a logical 0 for location without randomness. Finally, it was noted that the pixel intensities around the fire plume location are considered random. Note that, in Figure 5, the kernel used has dimensions of 9x9 around the pixel of interest. The following example shows how to determine the pixel values in Entropy filtered output image.

Table 7: Entropy Filtering Sample Calculation Method

Original Image									
0	0	0	0	0	255	255	255	0	0
0	0	0	0	0	255	255	255	0	255
0	0	0	0	0	255	255	255	255	255
0	0	0	0	0	255	255	255	255	255
0	0	0	0	0	255	255	255	255	0
0	0	0	0	0	255	255	255	255	0
0	0	0	0	0	255	255	255	0	255
0	0	0	0	0	255	255	255	255	255
0	0	0	0	0	255	255	255	0	255
0	0	0	0	0	255	255	255	0	0

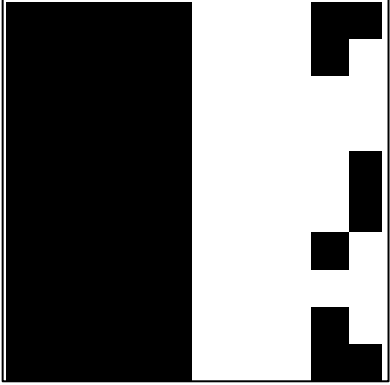
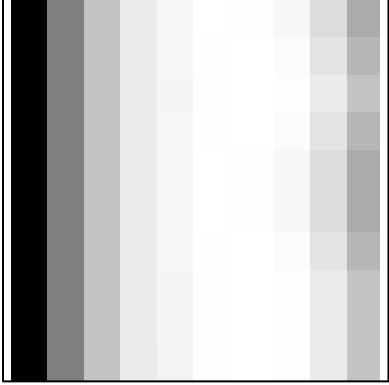
$$h = \frac{\text{Count for a given Gray Level}}{\text{Total Neighborhood Pixels}}$$

$$\text{entropy} = - \sum h \times \log_2(h) = - \left(\frac{32}{81} \log_2 \left(\frac{32}{81} \right) + \frac{49}{81} \log_2 \left(\frac{49}{81} \right) \right) = 0.968 \text{ ([0 1] range)}$$

$$\text{entropy} = 247 \text{ ([0 255] range)}$$

Entropy Filtered Image									
0	128	195	234	247	254	254	247	219	169
0	128	195	234	247	254	255	250	227	183
0	128	195	234	245	253	255	253	234	195
0	128	195	234	247	254	255	250	227	183
0	128	195	234	247					

Table 8: Entropy Filtering Sample Calculation Results

Original Image	Entropy Filtered Image
<pre> 0 0 0 0 0 255 255 255 0 0 0 0 0 0 0 255 255 255 0 255 0 0 0 0 0 255 255 255 255 255 0 0 0 0 0 255 255 255 255 255 0 0 0 0 0 255 255 255 255 0 0 0 0 0 0 255 255 255 255 0 0 0 0 0 0 255 255 255 0 255 0 0 0 0 0 255 255 255 255 255 0 0 0 0 0 255 255 255 0 255 0 0 0 0 0 255 255 255 0 0 </pre>	<pre> 0 128 195 234 247 254 254 247 219 169 0 128 195 234 247 254 255 250 227 183 0 128 195 234 245 253 255 253 234 195 0 128 195 234 247 254 255 250 227 183 0 128 195 234 247 254 254 247 219 169 0 128 195 234 247 254 254 247 219 169 0 128 195 234 247 254 255 250 227 183 0 128 195 234 247 254 255 250 227 183 0 128 195 234 245 253 255 253 234 195 0 128 195 234 245 253 255 253 234 195 0 128 195 234 242 253 255 253 234 195 </pre>
	

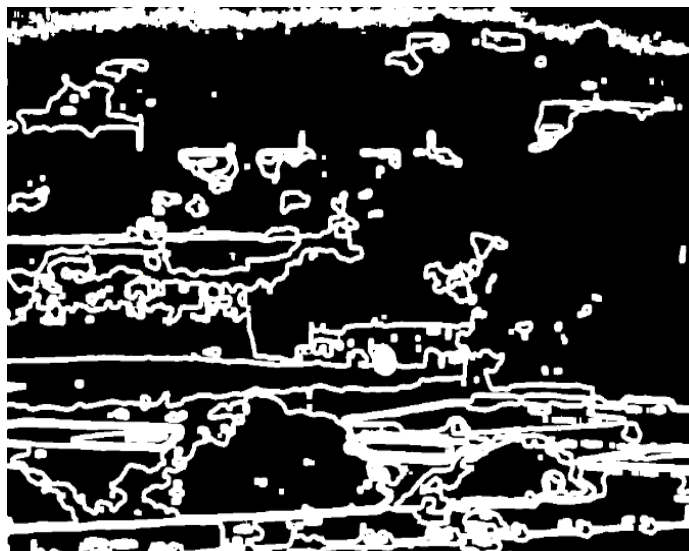


Figure 5: Output Image of an Entropy Filter Applied to the Original Midwave IR Image

3.5 Absolute Difference:

Absolute difference method is the simplest technique to detect variations between two frames temporally separated from each other. This method consists of calculating the absolute difference of pixels' intensity values for two frames temporally separated. For the purpose of early forest fire detection, a Cooled Midwave IR sensor can be used to detect the thermal distribution across a site. Such sensor detects with a frequency of 30 frames per second; then, the absolute difference image is produced using two frames 1/30 of a second apart. Figure 6 shows an original Midwave IR on the left, and the absolute difference image on the right. The difference image reveals the fire location; however, it contains static noise across the entire image, and it exposes any moving object within the sensor's field of view; this includes a moving vehicle, a flying bird, foliage, and any other moving objects. For example, let I_1 and I_2 be the images, of size $m \times n$, at frames one and two, respectively. Where:

$$I_1 = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \quad \text{and} \quad I_2 = \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix}$$

Then, the absolute difference image, I_{diff} is given by

$$I_{diff} = |I_2 - I_1| = |I_1 - I_2| \tag{Equation 3}$$

$$I_{diff} = \left| \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix} - \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \right|$$

$$I_{diff} = \begin{bmatrix} |b_{11} - a_{11}| & \cdots & |b_{1n} - a_{1n}| \\ \vdots & \ddots & \vdots \\ |b_{m1} - a_{m1}| & \cdots & |b_{mn} - a_{mn}| \end{bmatrix}$$



Figure 6: Original Midwave IR Image on the left, and Absolute Difference Image on the right

3.6 Combination of Multiple Filter Types:

After understanding how each filter affects the original image, different filtering combinations and orders were investigated. The best results were achieved using the following filter combination: First, applying range filtering to two temporally consecutive frames, then applying the absolute difference between the two output frames, and finally applying entropy filtering to the absolute difference. The final output results are shown in Figure 7 below. Without any threshold or adjusting any parameters, the final output exclusively reveals the fire core location. This method suppresses noise and moving objects, increasing the accuracy of the system's performance. However, additional tests and experiments must be performed for different types of fire and test conditions, such as time, location, weather, rain, and wind. In summary, the filter combination is as shown below.

Let I_1 and I_2 be two consecutive Midwave IR frames, then:

$$\text{Output Image} = \text{Entropy Filter} \{|\text{Range Filter}(I_1) - \text{Range Filter}(I_2)|\} \quad \text{Equation 4}$$

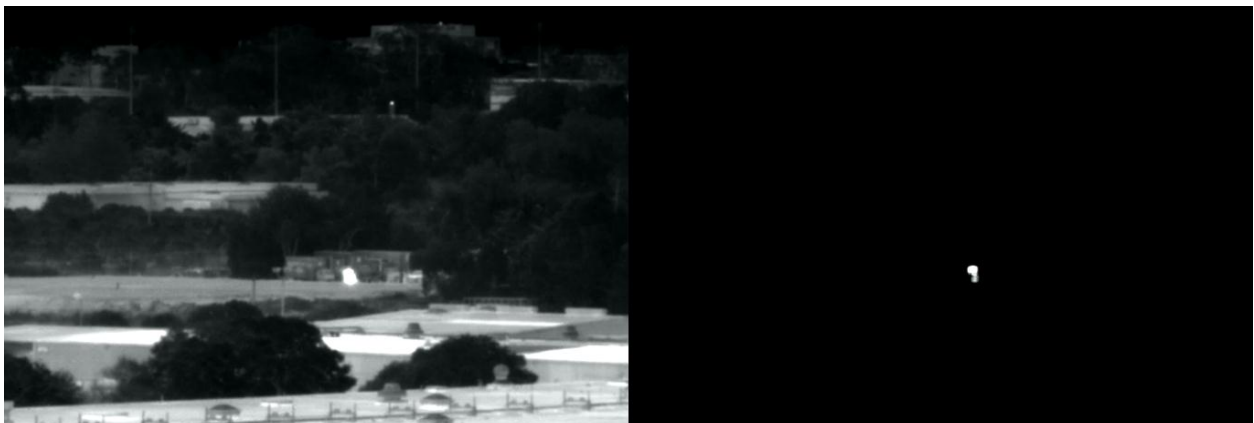


Figure 7: Midwave IR Frame 2500 on the Left, and Output Image showing Fire Plume Location Frame 2500 on the Right

Chapter 4 Visible Preprocessing

Visible sensors consist of three channels: Red, Green, and Blue. Each visible frame is composed of $M \times N \times 3$, a three dimensional matrix where $M \times N$ is the number of pixels within the camera's field of view, while the third dimension of the matrix is designated for each visible band: Red, Green, and Blue. The combination of the three bands forms a colored image. The visible camera used in the fire test has a resolution of 1086×873 ($M \times N$).

Figure 8 below shows a sample colored image recorded in the fire test. For the purpose of determining which band best reveals the smoke plumes and its characteristics, the three dimensional colored image was split into its individual bands. Then, a video of each band was generated, and it was determined visually that the Red band shows the least smoke plume activities, the Green band shows some smoke plume activities, and the Blue band shows the most smoke plume activities. For this reason, only the Blue band data was considered in this study. Figure 9 shows the Red, Green and Blue bands, respectively.



Figure 8: RGB Visible "Capture 13" Frame 8003



Figure 9: Red, Green, and Blue Band of Frame 8003, Respectively

Since the smoke plume covers relatively a small area of the camera's field of view, the frames were cropped to a smaller dimension covering the region of interest. This region is the enclosed area of the red rectangle in Figure 10 below. It includes a high smoke plume activity area, a walking pedestrian region, and two regions with high vehicle activities. The main purpose for cropping the original frames is to reduce computation time and storage requirements, which in result facilitates faster prototyping. Figure 11 shows the cropped colored image, and Figure 12 is the Blue band of the cropped image which includes the characteristics of smoke plume and field of view used in the rest of this study.



Figure 10: RGB Visible "Capture 13" Frame 8003 with Small FOV around BBQ Pit

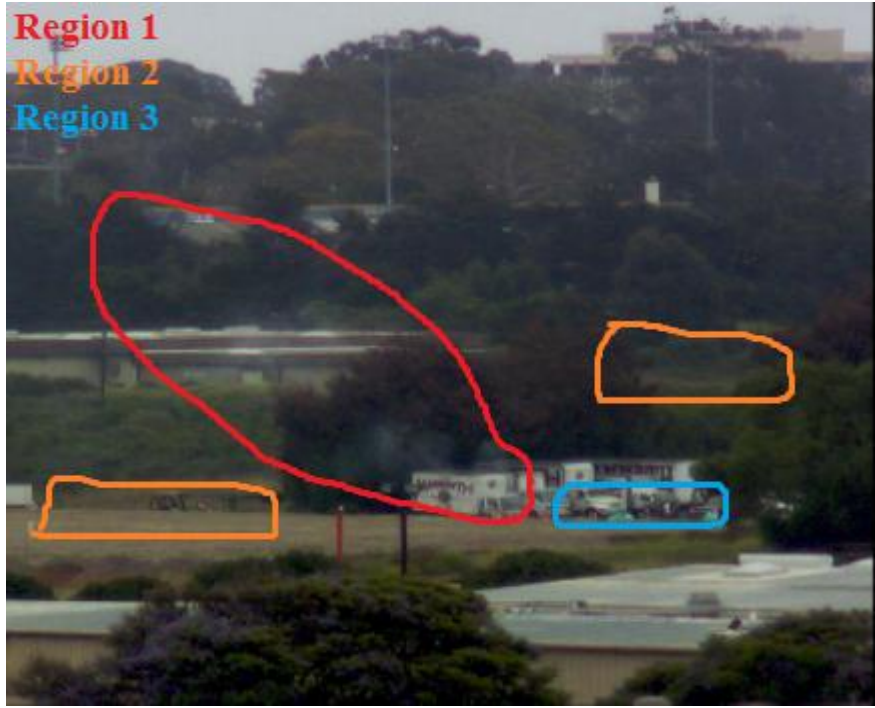


Figure 11: RGB Visible "Capture 13" Frame 8003 with Small FOV around BBQ Pit,
Region 1: Smoke Plume
Region 2: Vehicle Movements
Region 3: Pedestrians



Figure 12: Blue Band Visible "Capture 13" Frame 8003 with Small FOV around BBQ Pit

Chapter 5 Principal Component Analysis

5.1 Principal Component Analysis (PCA) Introduction

Principal Component Analysis is a mathematical method invented by Karl Pearson in 1901. It uses orthogonal transformation to reduce multi-dimensional data into lower dimensions presented in Principal Components (PC). This transformation converts the original data of possibly correlated variables into a set of values of linearly uncorrelated variables. PCA is a method of identifying patterns in data, and it reveals their similarities and differences. The greatest variance by any projection of the data corresponds to the first principal component, PC1; the 2nd greatest variance corresponds to PC2, and so on. Principal component transform is also known as Karhunen–Loève transform (KLT). PCA is a useful statistical technique that has found application in fields such as “Change Detection”, image and bandwidth compression, face recognition, and many other digital image processing purposes. It is a common technique for finding patterns in data of high dimension. In this paper Principal Component Analysis and Principal Component Transform are used as a change detection technique in order to segment the smoke plume within the Blue band of the visible frames. Since the movement is over time, this technique can be referred to as a multi-temporal PCA [33], [34], [35].

5.2 Principal Component Analysis Theory/Procedure

Principal Component Analysis simply is a method that maps data in input space, \mathbf{X} , to data in Principal Component space, \mathbf{Y} , using Principal Component Transform \mathbf{H} . This simple relation is shown below:

$$\mathbf{Y} = (\mathbf{H}^T \cdot \mathbf{X}^T)^T \quad \text{Equation 5}$$

Where \mathbf{X} and \mathbf{Y} are the input and output matrices of dimensions (M, N), and \mathbf{H} is the Transform matrix of dimensions (N, N). Please note that in temporally separated data, N represents the number of instances included in each PCA computation. For example, if temporal PCA is being performed on 5 instances of time at once, then N = 5. Additionally, M is the size of the input data of the 5 instances reshaped into a vector. M is described more in the next paragraph. The character T represents the transpose of the matrix.

Principal Component Analysis consists of the following steps:

1) Data collection (images):

In image processing, PCA is used as a change detection technique between temporally separated frames, of dimensions (m, n), with the same field of view. Let \mathbf{F}_1 be the frame at time t = 1, and \mathbf{F}_2 be the frame at time t = 2, and so on such that \mathbf{F}_N be the frame at time t = N.

Where:

$$\mathbf{F}_1 = \begin{pmatrix} a_{11_1} & \cdots & a_{1n_1} \\ \vdots & \ddots & \vdots \\ a_{m1_1} & \cdots & a_{mn_1} \end{pmatrix}, \mathbf{F}_2 = \begin{pmatrix} a_{11_2} & \cdots & a_{1n_2} \\ \vdots & \ddots & \vdots \\ a_{m1_2} & \cdots & a_{mn_2} \end{pmatrix}, \dots,$$

$$\mathbf{F}_N = \begin{pmatrix} a_{11_N} & \cdots & a_{1n_N} \\ \vdots & \ddots & \vdots \\ a_{m1_N} & \cdots & a_{mn_N} \end{pmatrix} \quad \text{Equation 6}$$

In this study, the original data are nine temporally separated Blue band of the visible frame (single band) shown in Figure 14. The visible sensor detects frames at a rate of 10 Hz; that is 10 frames per second; however, since smoke changes insignificantly at this rate, only every other frame is used for PCA. Since a total of 9 frames is used per PCA computation, these frames were as follows for the first computation: $F_1, F_3, F_5, F_7, F_9, F_{11}, F_{13}, F_{15},$ and F_{17} ; then the first frame, F_1 is dropped from the left side, and an additional frame, F_{19} , is added on the right end side for the second PCA computation; and so on. Therefore, each PCA computation requires $17Frames \times \frac{1\ sec}{10\ Frames} = 1.7\ seconds$ of real time data, while the update rate is only $2\ Frames \times \frac{1\ sec}{10\ Frames} = 0.2\ seconds$. The required real time data and the update rate values were adjusted accordingly in order to optimize and produce the best results. This procedure is described in the following diagram shown in Figure 13 below.

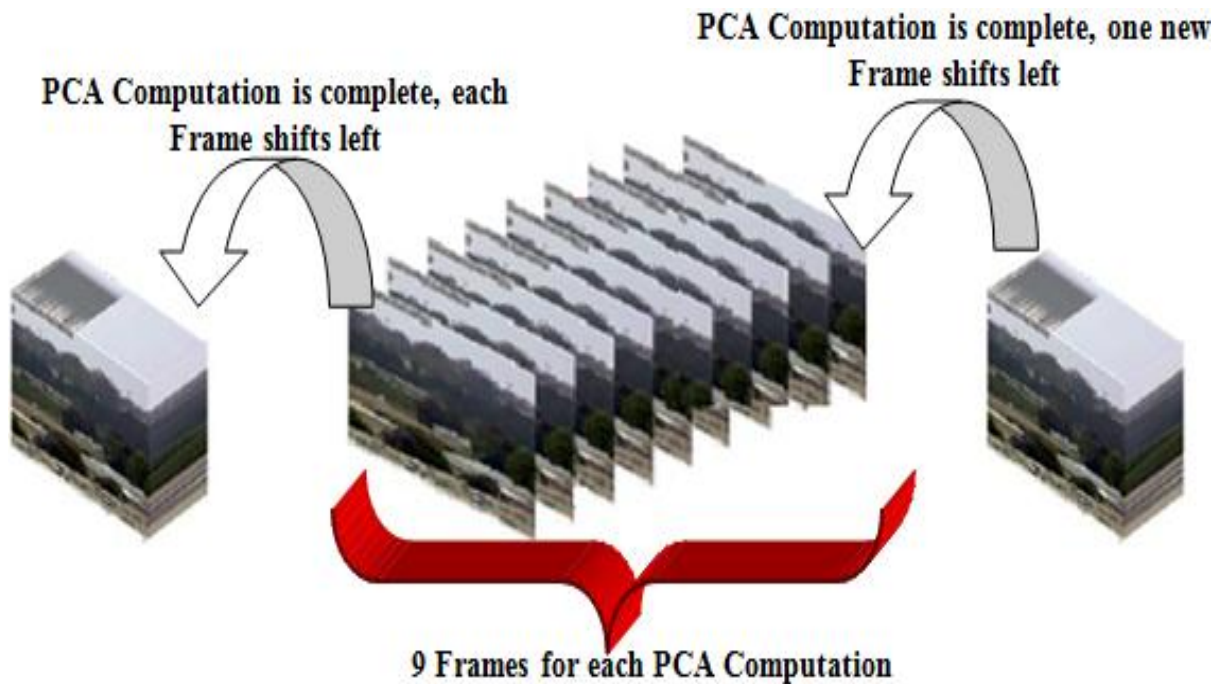


Figure 13: Frame Shifts per PCA Computation

Digital images consist of pixel locations and an intensity value at that location; pixel locations are identified by the intersection of rows and columns. However, spatial relation of pixels does not have any effect on PCA transformation matrix since the main purpose of PCA, in this case, is to extract temporal information (change detection). For this reason, and as mentioned earlier, the frames described above must first be reshaped into a vector with the correct dimensions. For each frame, the rows and columns are grouped together to form a vector of dimensions $(M, 1) = (m \cdot n, 1)$; but since 9 frames are considered for each PCA computation, the reshaped vector forms the input data vector \mathbf{X} of dimension $(M, N) = (m \cdot n, 9)$. The reshape procedure is described in the equations below.

$$\mathbf{X} = \text{reshape}([\mathbf{F}_1 \ \mathbf{F}_2 \ \cdots \ \mathbf{F}_N], M, N) \quad \text{Equation 7}$$

$$\mathbf{X} = \text{reshape} \left(\left[\begin{pmatrix} a_{11_1} & \cdots & a_{1n_1} \\ \vdots & \ddots & \vdots \\ a_{m1_1} & \cdots & a_{mn_1} \end{pmatrix} \begin{pmatrix} a_{11_2} & \cdots & a_{1n_2} \\ \vdots & \ddots & \vdots \\ a_{m1_2} & \cdots & a_{mn_2} \end{pmatrix} \cdots \begin{pmatrix} a_{11_N} & \cdots & a_{1n_N} \\ \vdots & \ddots & \vdots \\ a_{m1_N} & \cdots & a_{mn_N} \end{pmatrix} \right], M, N \right)$$

$$\mathbf{X} = \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{pmatrix}$$

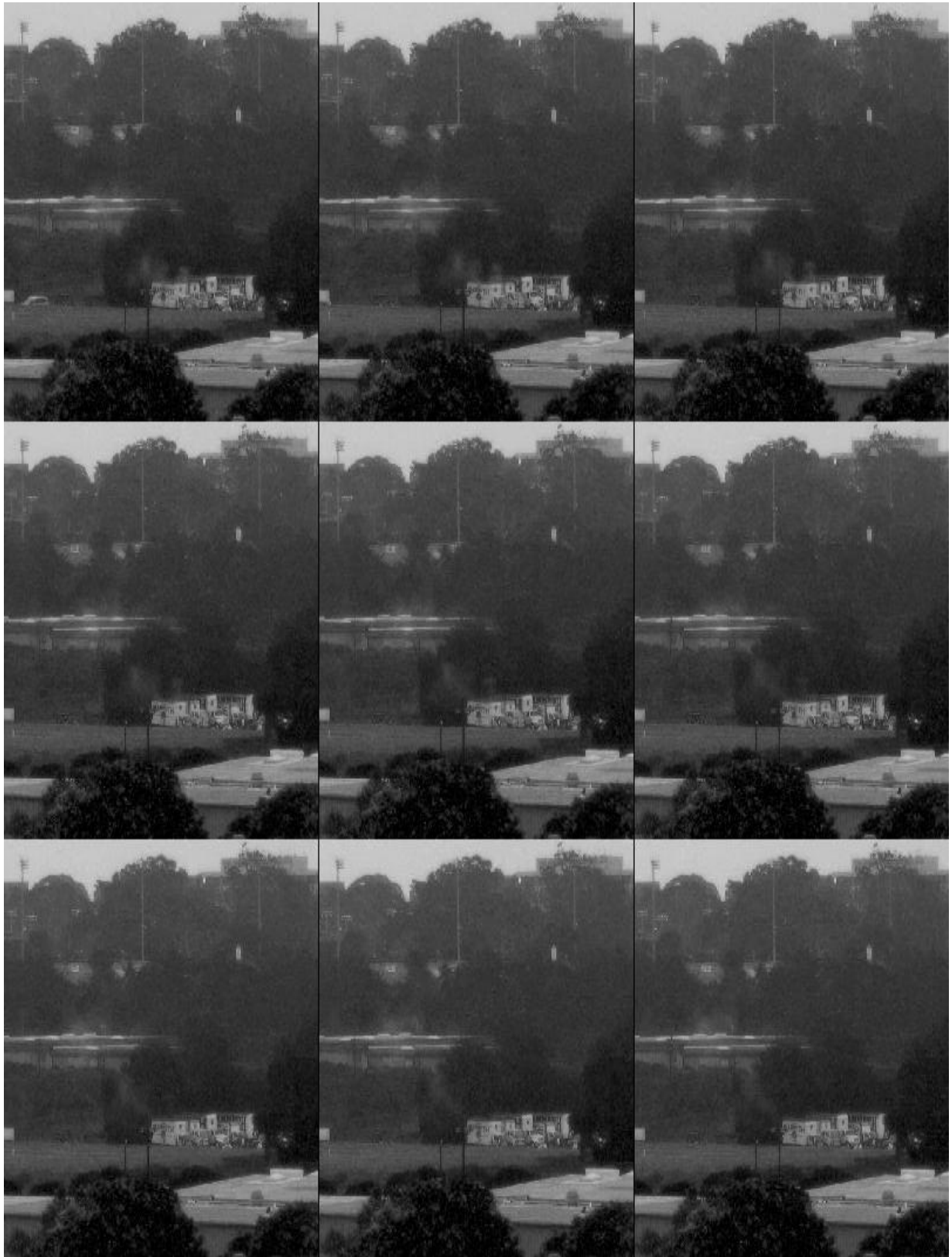


Figure 14: 9 Consecutive Blue Band "Capture 13" Starting Frame 8003 used in One PCA Computation

2) Mean subtraction:

The data set must have zero mean in all dimensions since the Principal Coefficients are sensitive to scale. Therefore, for PCA to work properly, the input data \mathbf{X} must be normalized to its Z scores. Z score normalization delivers both zero means and unity variance to the input data \mathbf{X} . This is done by subtracting the data by each column's mean μ_C , and then dividing by the column's standard deviation σ_C . Please note that C represents a specific column in \mathbf{X} while R represents a specific row in \mathbf{X} . This is done for each of the data dimensions.

$$\mathbf{X}_{\text{Norm}C} = \frac{\mathbf{X}_{RC} - \mu_C}{\sigma_C}, \text{ such that } C \in [1, N] \quad \text{Equation 8}$$

Where

$$\mu_C = \frac{1}{M} \sum_{R=1}^M \mathbf{X}_{RC}, \text{ such that } R \in [1, M] \quad \text{Equation 9}$$

$$\sigma_C = \sqrt{\frac{1}{M} \sum_{R=1}^M (\mathbf{X}_{RC} - \mu_C)^2} \quad \text{Equation 10}$$

Therefore,

$$\mathbf{X}_{\text{Norm}} = \begin{pmatrix} \frac{a_{11} - \mu_1}{\sigma_1} & \dots & \frac{a_{1N} - \mu_N}{\sigma_N} \\ \vdots & \ddots & \vdots \\ \frac{a_{M1} - \mu_1}{\sigma_1} & \dots & \frac{a_{MN} - \mu_N}{\sigma_N} \end{pmatrix} \quad \text{Equation 11}$$

3) Covariance matrix calculation:

The covariance matrix of the data set is simply calculated as follows:

$$\text{Cov}(\mathbf{X}_{\text{Norm}}) = \frac{1}{M} \mathbf{X}_{\text{Norm}} \cdot \mathbf{X}_{\text{Norm}}^T \quad \text{Equation 12}$$

$$\text{Cov}(\mathbf{X}_{\text{Norm}}) = \begin{pmatrix} c_{11} & \cdots & c_{1N} \\ \vdots & \ddots & \vdots \\ c_{N1} & \cdots & c_{NN} \end{pmatrix}$$

4) Eigenvectors and Eigenvalues of the covariance matrix calculation:

The Eigenvectors, \mathbf{V}_i , and the Eigenvalues, λ_i , of the Covariance matrix, $\text{Cov}(\mathbf{X}_{\text{Norm}})$, are calculated. The Eigenvectors are arranged in a decreasing order of associated Eigenvalues. This step is done to extract lines that characterize the data. Please note that the subscript i represents a specific Eigenvector, and i can be anywhere from $[1, N]$.

$$\mathbf{V} = [\mathbf{V}_1 \quad \mathbf{V}_2 \quad \cdots \quad \mathbf{V}_N] \quad \text{Equation 13}$$

$$\boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{pmatrix} \quad \text{Equation 14}$$

Please note that the $\boldsymbol{\lambda}$ diagonal matrix above is the covariance matrix in Principal Component space. Additionally, the zero values in $\boldsymbol{\lambda}$ ensure that the dimensions in Principal Component space have zero correlation, or no correlation, with each other.

5) Choosing components and forming a feature vector:

This step allows the possibility of both dimension reductions and data compression. The feature vector consists of the desired Eigenvectors (matrix of vectors). Excluding Eigenvectors from the feature vector will result in decreasing the dimensions of the final output data. This step is what makes a compression via PCA lossy. Small Eigenvalues can be excluded since they represent little information. However, in this study, all Eigenvectors are used to ensure that all the original information and input data are considered in PCA. Therefore, Principal Component Transform matrix consists of all the Eigenvectors calculated in step 4 above.

$$\mathbf{H} = \begin{pmatrix} \mathbf{V}_1 \\ \vdots \\ \mathbf{V}_N \end{pmatrix} \quad \text{Equation 15}$$

6) Deriving the new data set:

The new data set, also known as the Principal Components, is calculated as shown in the following equations. The mean subtracted data (step 2) is multiplied by the feature vector (step5).

$$\mathbf{Y} = (\mathbf{H}^T \cdot \mathbf{X}_{\text{Norm}}^T)^T \quad \text{Equation 16}$$

$$\mathbf{Y} = \left(\begin{pmatrix} \mathbf{V}_1 \\ \vdots \\ \mathbf{V}_N \end{pmatrix}^T \cdot \begin{pmatrix} \frac{a_{11} - \mu_1}{\sigma_1} & \dots & \frac{a_{1N} - \mu_N}{\sigma_N} \\ \vdots & \ddots & \vdots \\ \frac{a_{M1} - \mu_1}{\sigma_1} & \dots & \frac{a_{MN} - \mu_N}{\sigma_N} \end{pmatrix}^T \right)^T$$

\mathbf{Y} , dimensions $M \times N$, is the uncorrelated pixel vectors forming the Principal Components. There are N Principal Components in \mathbf{Y} . In order to view each Principal Component, the order of pixels must be restored and reshaped to fit the original dimensions of $m \times n$. Recall that $M = m \cdot n$ and in this study $N = 9$, so there are 9 Principal Components: $\text{PC}_1, \text{PC}_2, \dots$, and PC_9 .

$$\mathbf{Y} = \begin{pmatrix} y_{11} & \cdots & y_{1N} \\ \vdots & \ddots & \vdots \\ y_{M1} & \cdots & y_{MN} \end{pmatrix} = (\mathbf{PC}_1 \quad \cdots \quad \mathbf{PC}_N) \quad \text{Equation 17}$$

Where,

$$\mathbf{PC}_1 = \mathit{reshape} \begin{pmatrix} y_{11} \\ \vdots \\ y_{M1} \end{pmatrix} = \begin{pmatrix} y_{11_1} & \cdots & y_{1n_1} \\ \vdots & \ddots & \vdots \\ y_{m1_1} & \cdots & y_{mn_1} \end{pmatrix}$$

$$\vdots \qquad \qquad \qquad \vdots \qquad \qquad \qquad \vdots \qquad \qquad \qquad \text{Equation 18}$$

$$\mathbf{PC}_N = \mathit{reshape} \begin{pmatrix} y_{1N} \\ \vdots \\ y_{MN} \end{pmatrix} = \begin{pmatrix} y_{11_N} & \cdots & y_{1n_N} \\ \vdots & \ddots & \vdots \\ y_{m1_N} & \cdots & y_{mn_N} \end{pmatrix}$$

The procedure described above is referred to as non-adaptive Principal Component Analysis.

The Principal Components of the sample images presented in step 1 Figure 14 above are displayed along with their histograms in Figures 15 and 16, respectively.

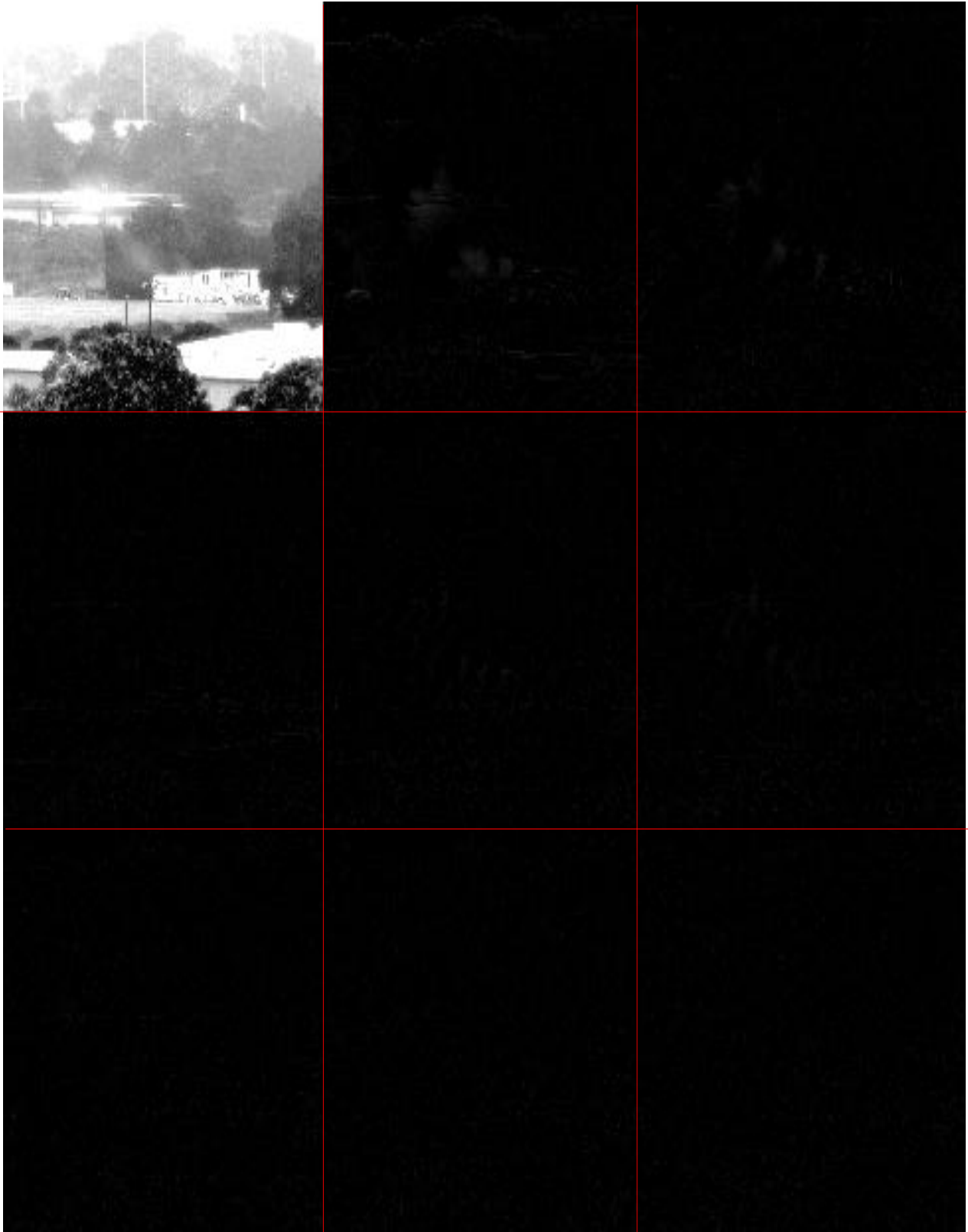


Figure 15: Non Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible "Capture 13" Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

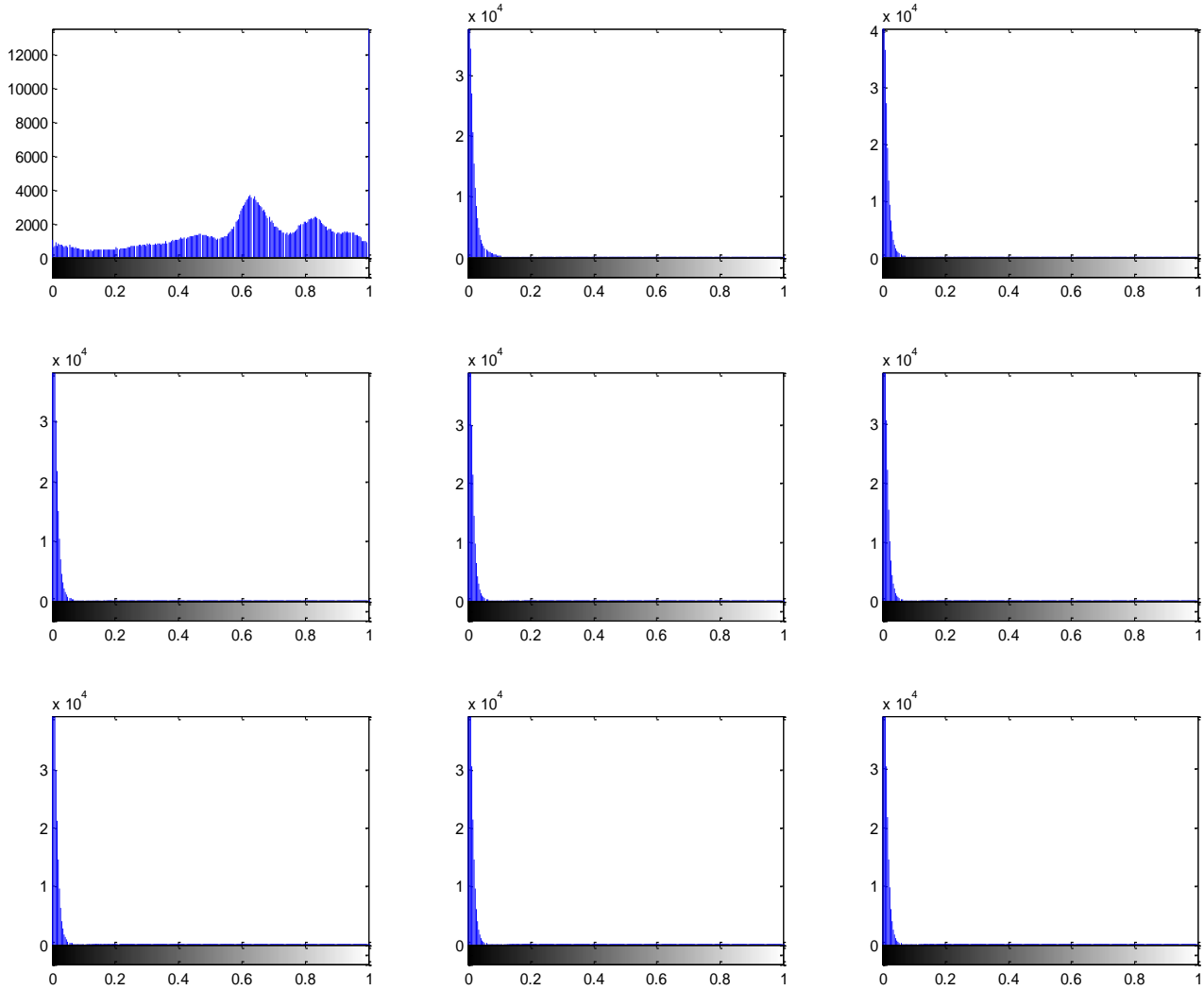


Figure 16: Histogram for Non Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

7) Scaling to Full Dynamic Range

Scaling to full dynamic range is a form of intensity mapping. It enhances features in an image by adjusting its contrast. It is mostly used when pixels intensity values lay within a small range, and when the data spans both positive and negative values, in which case negative data cannot be plotted or displayed. In result, scaling to full dynamic range enhances the information content in an image making it more observable to human viewers.

By examining the histograms shown in Figure 16 above, it is evident that the data is condensed to a small range of the data type; therefore, and in order to display better visual results for the Principal

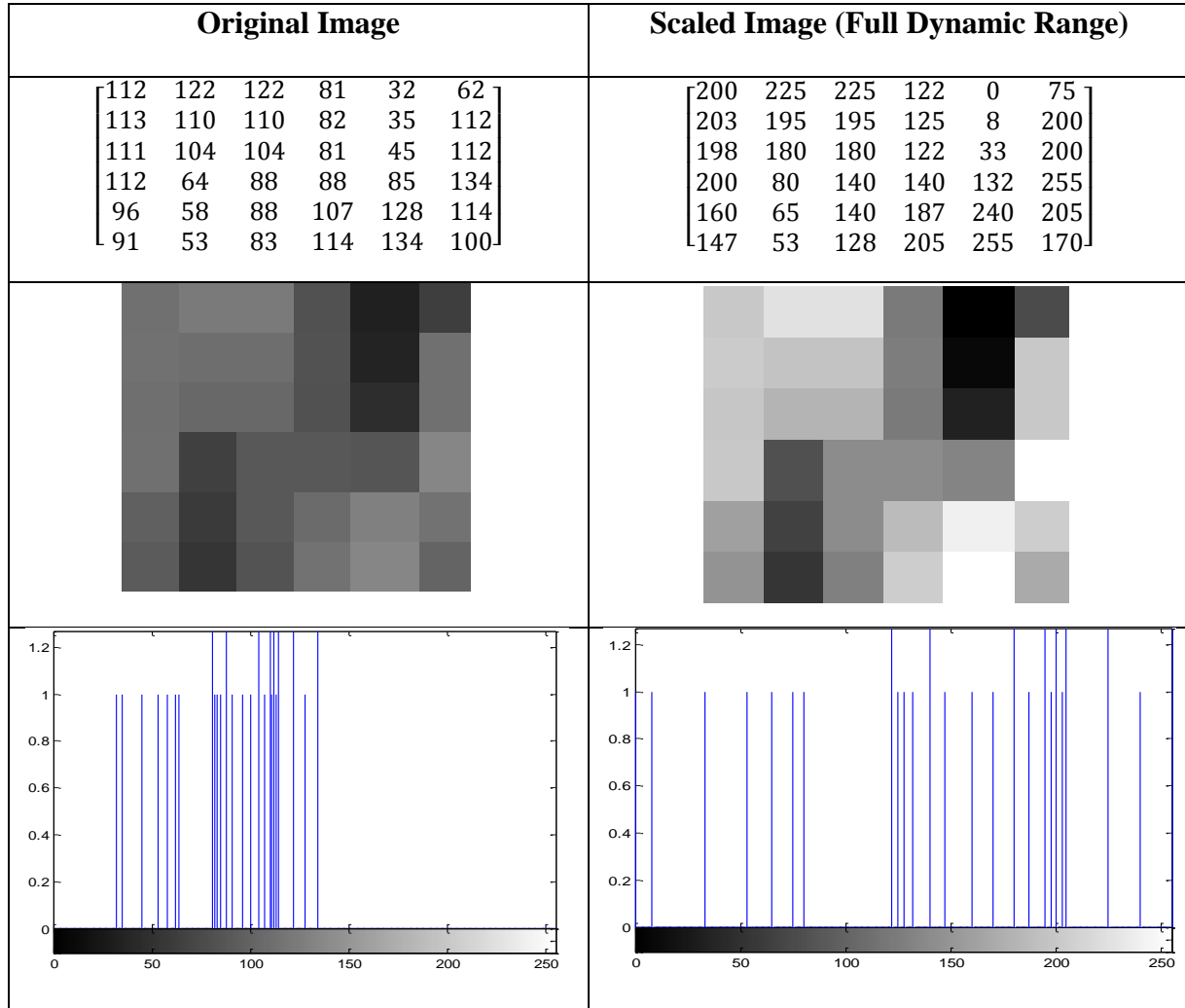
Components, the data can be scaled across the entire dynamic range, that is 0 to 1 in type DOUBLE data, or 0 to 255 in type UINT8 data. In this study both types were used, so some histograms display a data range between 0 to 1, while others display a data range between 0 and 255.

In order to scale data Z to full dynamic range, first, one must determine the minimum and maximum values in the input data; let these two values be **min** and **max**, respectively. Then, one must also know the full dynamic range, and let that range be [**a**, **b**]. Finally, in order to linearly scale the input data from their minimum and maximum values to their full dynamic range [**a**, **b**], the following linear transformation equation must be used:

$$Z_{\text{Scaled}} = \mathbf{a} + \frac{Z - \mathbf{min}}{(\mathbf{max} - \mathbf{min})} \times (\mathbf{b} - \mathbf{a}) \quad \text{Equation 19}$$

The table below shows a sample calculation of a random 6x6 UINT8 image and the result of scaling the intensities to the full dynamic range [0 255], and its effect on the histogram.

Table 9: Full Dynamic Range Scaling Example



Scaling the Principal Components' Data to their full dynamic range produces clearer results. As shown in Figure 17 below, the first Principal Component image, PC1, captures the common features in the nine temporally separated images shown in Figure 14 above, such as buildings, roads, towers, and any other stable objects. PC2 and few of the lower order Principal Components reveal features that changed over time, such as smoke plumes, moving vehicles, some foliage movements, and/or pedestrians. The higher order Principal Components such as 7th, 8th, and 9th are responsible for minor information and can be neglected for the purpose of Change Detection.

By examining the lower order Principal Components, more specifically PC2 and PC3, it is evident that PCA effectively captures the growing smoke plume, which is a decent indicative of a fire, because of its unique temporal variation. Therefore this unique temporal variation can be considered to be the signature of the smoke plume.

The following section of this study discusses another form of PCA, known as Adaptive PCA. It is studied for the purpose of identifying which method, Adaptive PCA or Non-Adaptive PCA, produces better results in highlighting and capturing the features that change temporally within the input frames. In other words, whichever PCA method better segments the smoke plume will be the candidate in the final design of an Early Forest Fire Detection System.

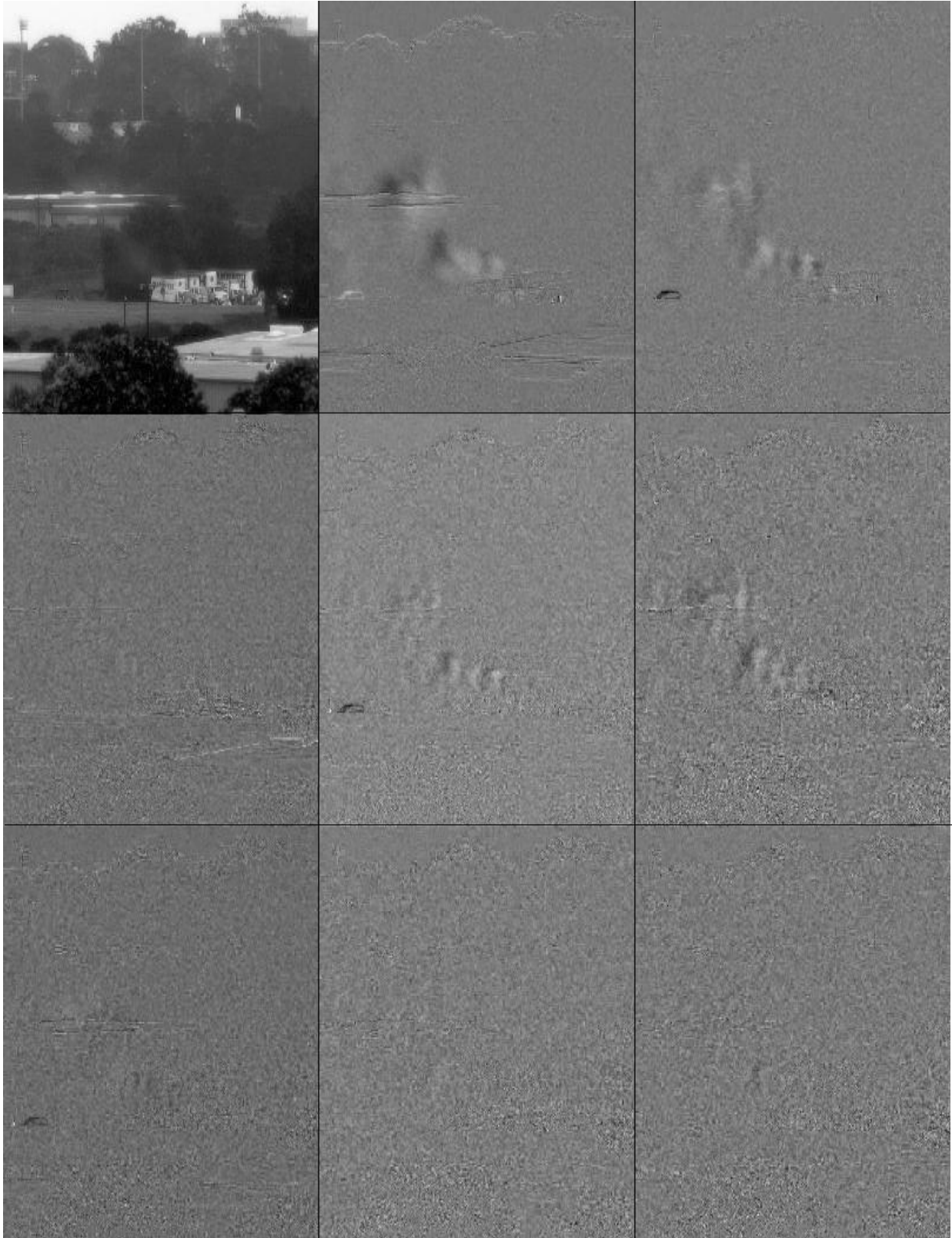


Figure 17: Non Adaptive Principal Components, After Full Dynamic Range Scaling, Visible "Capture 13" Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

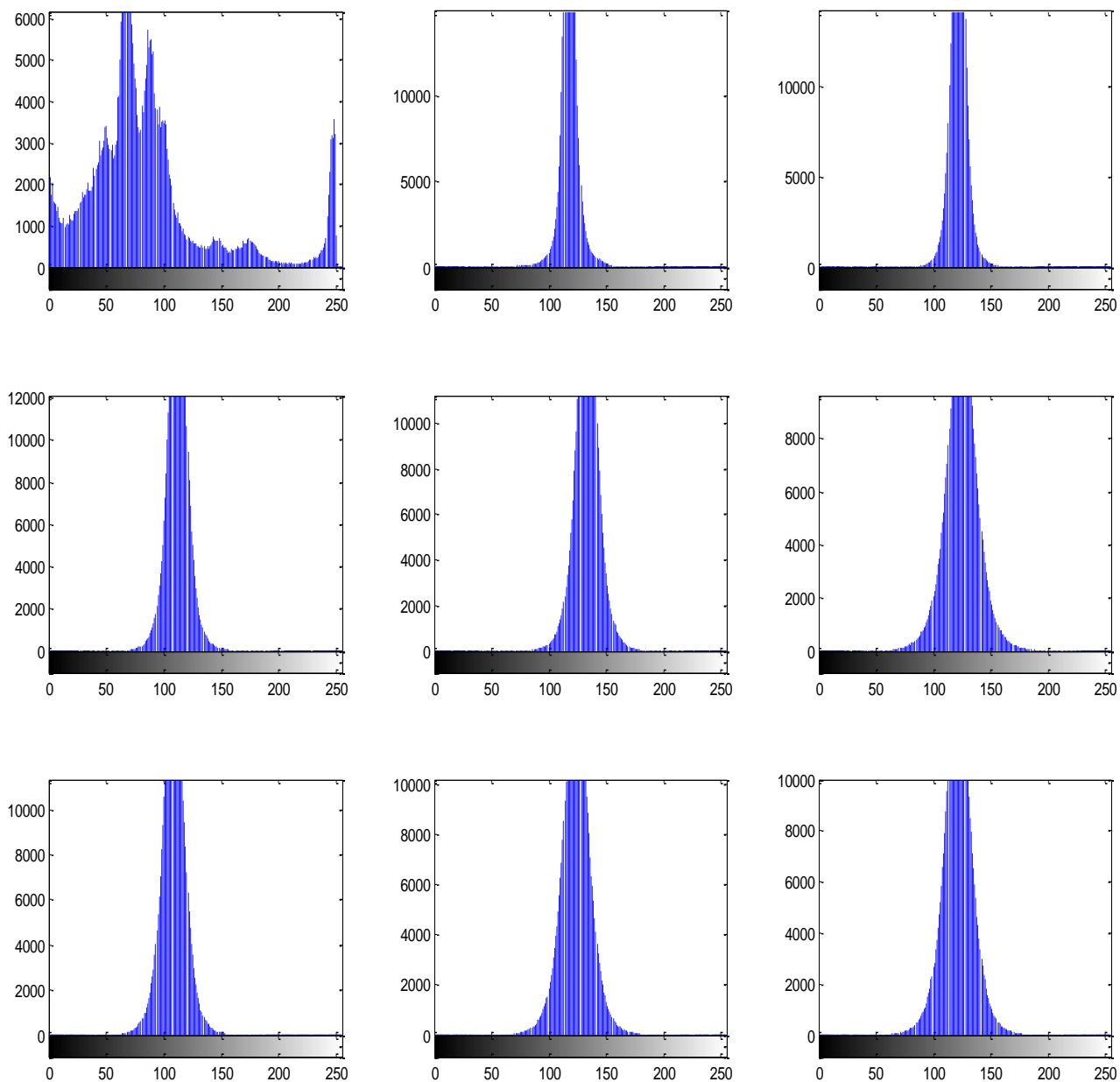


Figure 18: Histogram for Non Adaptive Principal Components, After Full Dynamic Range Scaling, Visible “Capture 13” Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

5.3 Adaptive Principal Component Analysis (PCA)

Non-Adaptive or so called normal Principal Component Analysis is the procedure described in the section above. PCA input data are the image frames with the full dimensions. Each image is reshaped into the correct format as described in the PCA procedure above. However, in Adaptive PCA, the original frames are broken into smaller blocks, or sub-images, and each sub-block becomes the input data of a normal PCA procedure. The size of sub-blocks can be adjusted depending on the field of view and the focal depth of the camera. In this study, the sub-blocks have a size of 150 x 150 pixels, while the full frame size used is 650 x 650 pixels. In this case, the case where the size of the full frame is not an integer multiple of the sub-block size, the border pixels are still treated with a similar fashion however their dimensions are different than the desired sub-block dimensions. The following diagram represents the described sub-blocks procedure.

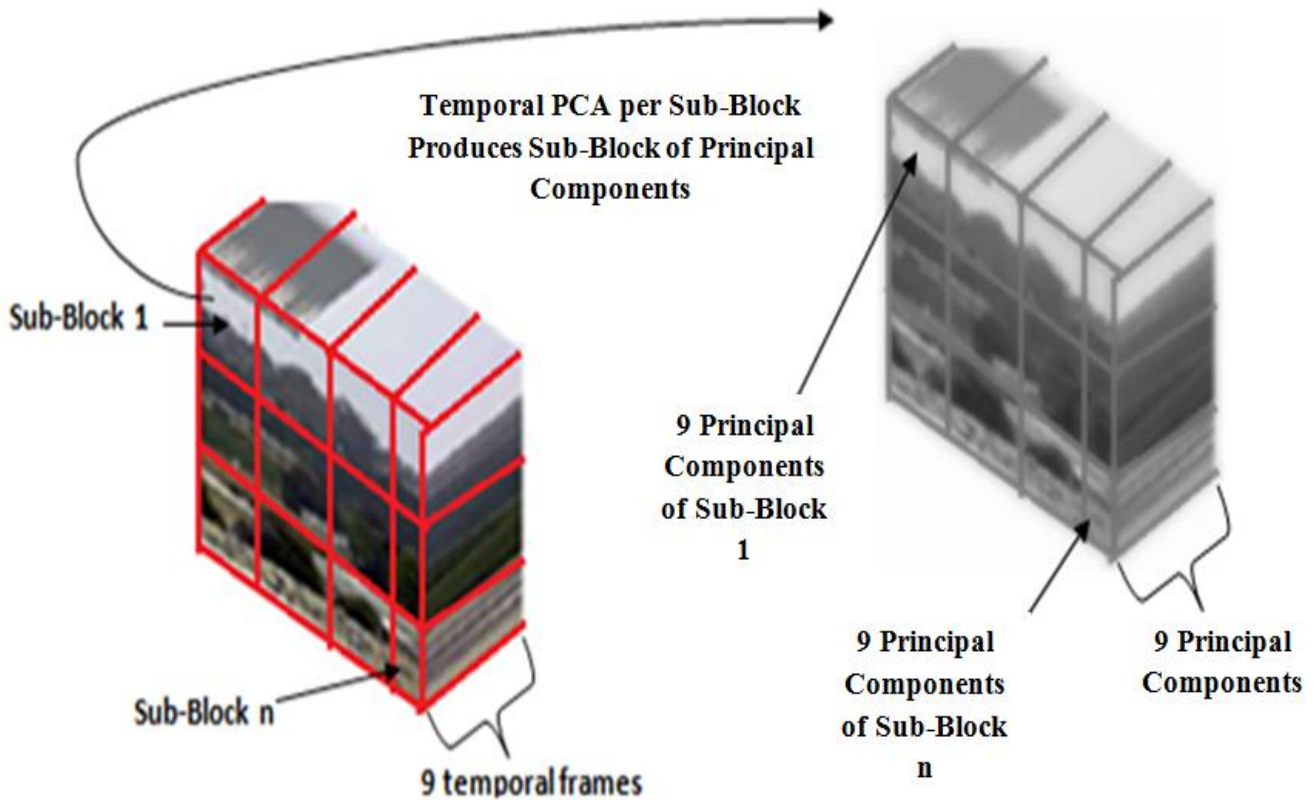


Figure 19: Adaptive PCA Sub-Block Procedure

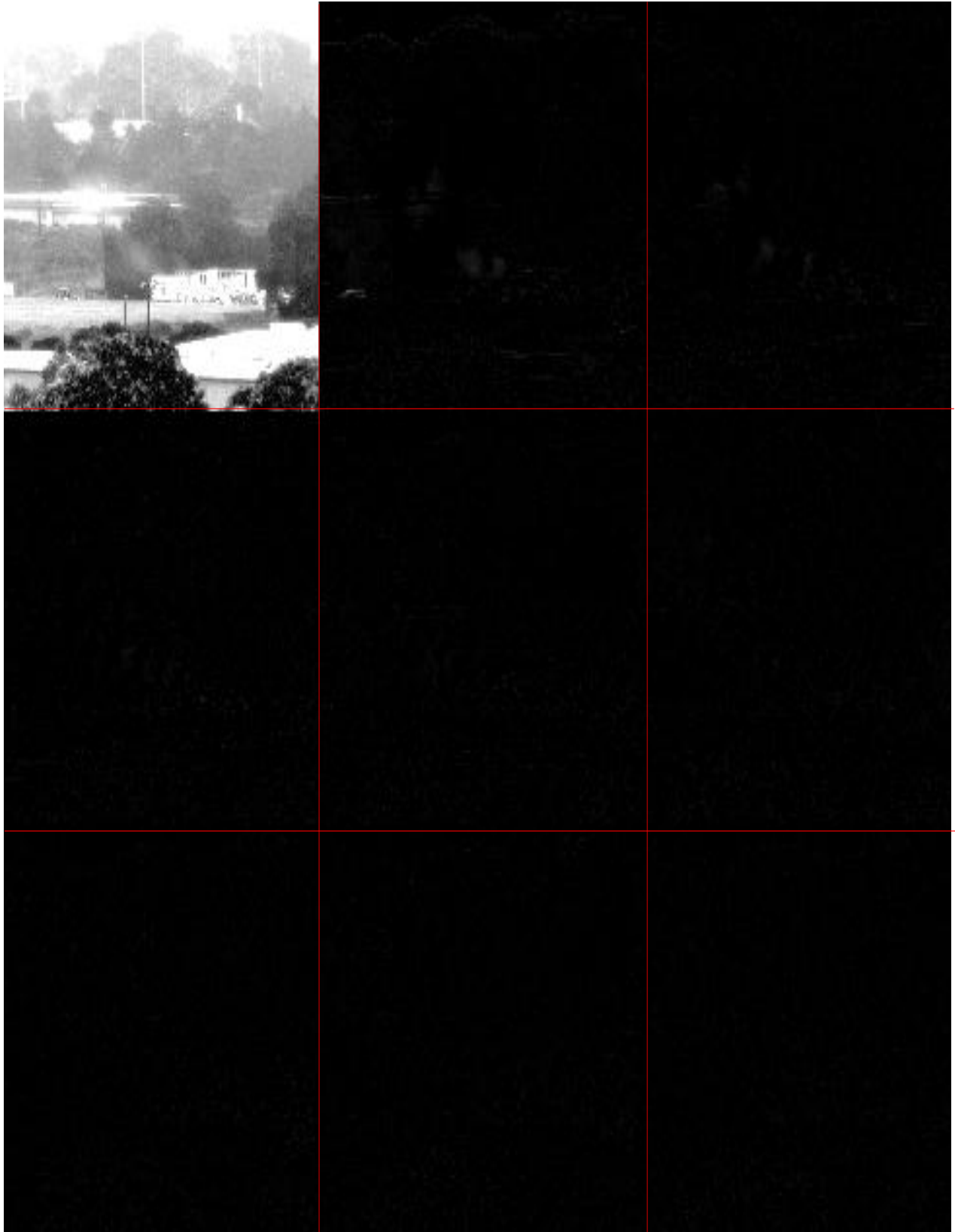


Figure 20: Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible "Capture 13" Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

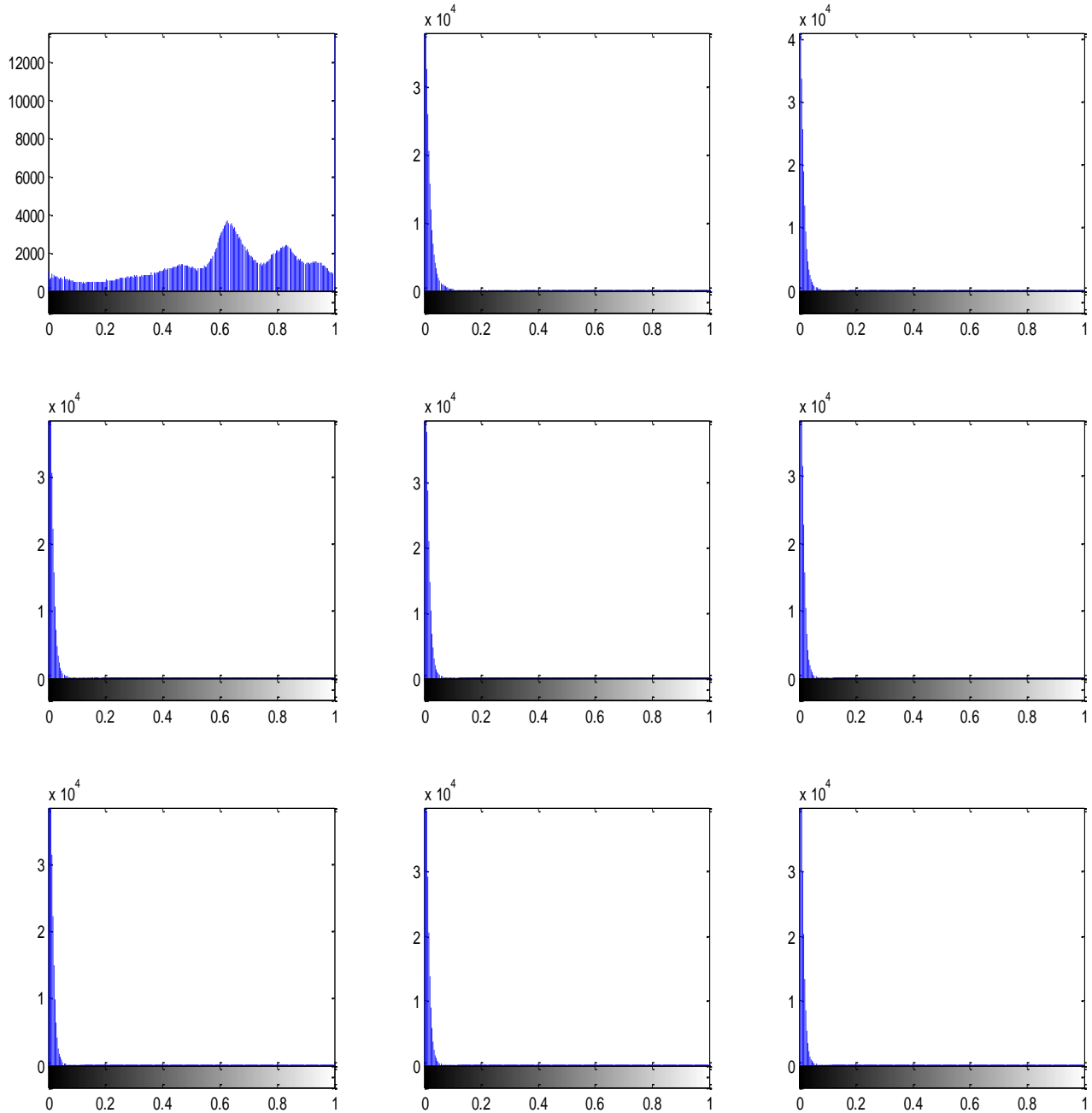


Figure 21: Histogram for Adaptive Principal Components, Before Full Dynamic Range Scaling, Visible "Capture 13" Frame 8003, Displaying Only Positive Intensity Values; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

Again, the Principal Components' histograms produced from the Adaptive PCA procedure show that the data does not cover the entire dynamic range. Therefore, Principal Components' data must undergo the same scaling procedure described in the Non-Adaptive PCA case. The final results of Adaptive Principal Components and their histograms are presented in Figures 22 and 23, respectively.

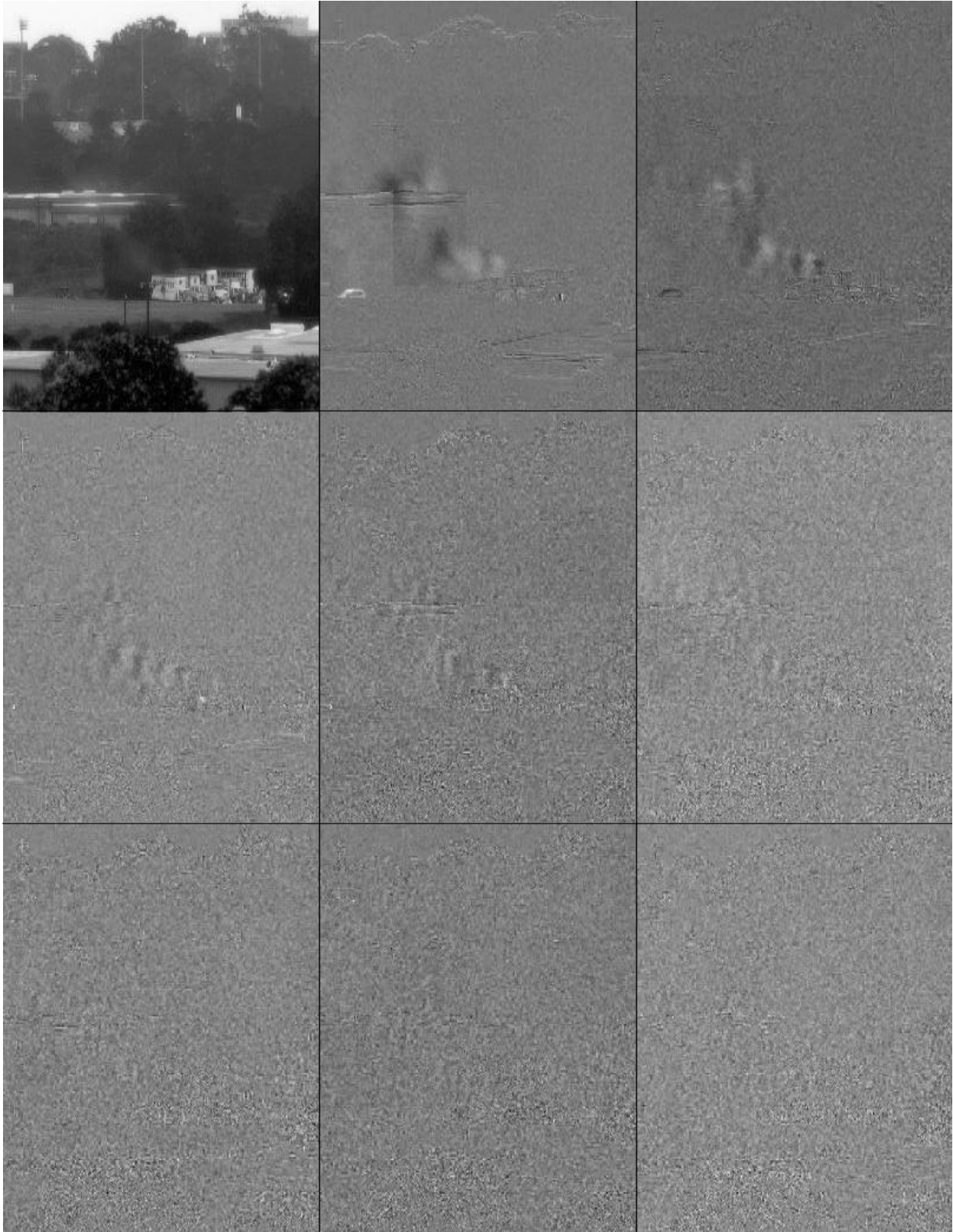


Figure 22: Adaptive Principal Components, After Full Dynamic Range Scaling, Visible "Capture 13" Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

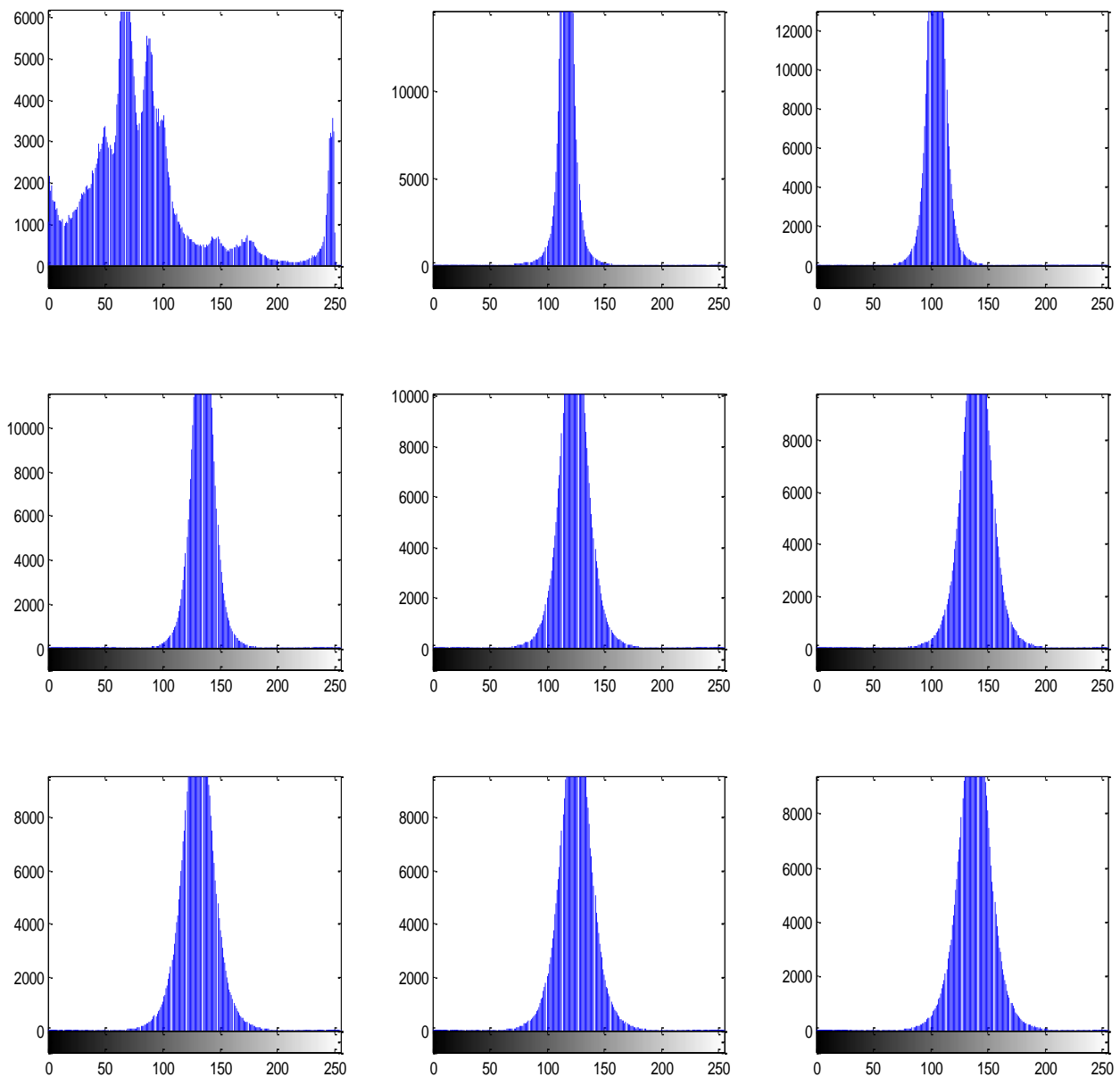


Figure 23: Histogram for Adaptive Principal Components, After Full Dynamic Range Scaling, Visible "Capture 13" Frame 8003; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

5.4 Comparing Adaptive to Non-Adaptive Principal Component (PC2)

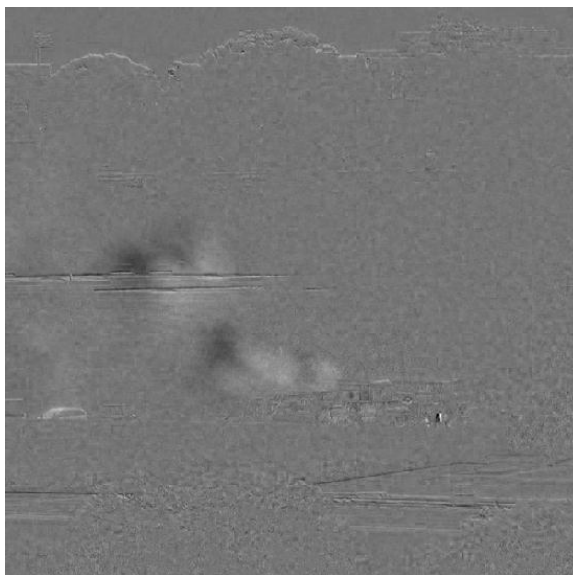


Figure 24: Non Adaptive PC2 Frame 8000

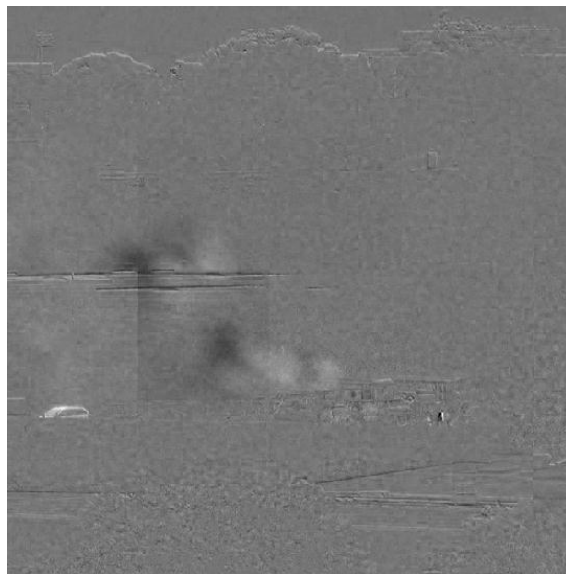


Figure 25: Adaptive PC2 Frame 8000

As shown in Figures 24 and 25 above, both adaptive and non-adaptive PCA methods segment and capture the temporal features within the original frames, including: the smoke plume, a passing vehicle, a pedestrian, and the border lines of initially thought to be stable objects, such as buildings, poles, and shrubby borders. The reason behind the existence of these border lines in PC2 is due to Jitter. Jitter is a source of noise formed from physical vibrations of the sensing device, which can be caused by wind. Since the sensing device is going through some minimal vibrations, the sensor's field of view in return vibrates, and therefore the current recorded frame's field of view is slightly a shifted version of the previous frame's field of view. For this reason PC2 captures the edges of non-moving objects. Please note that this effect can be minimized by adding an additional procedure in the preprocessing stage. In order to reduce the effect of Jitter, the original frames must first go through a video stabilizer process before applying Principal Component Analysis.

By examining the differences in PC2 obtained from the two PCA methods, it is evident that the adaptive approach shows more details, in the second Principal Component PC2, than the non-adaptive

approach. Hence, the brighter vehicle shape and the darker smoke plume shape in the adaptive PC2. On the down side, in the adaptive case, since each Principal Component's sub-block is individually scaled to its full dynamic range, which in return causes the edges of the sub-blocks to show up making it more difficult to identify and isolate the smoke signals.

In conclusion, Principal Component Analysis works well as change/movement detection technique. But instead of examining all the Principal Components (9 PCs in this study), we can just examine the 2nd Principal Component, PC2, since it will always capture the most temporal change. Doing so will decrease the required computational time, and it results in a quicker smoke detection. As shown above, two PCA techniques were experimented: Adaptive and Non-Adaptive PCA. Adaptive PCA introduced the blocking edge effect, which is an additional source of image pollution and noise, since it makes it more difficult to extract the smoke plume. For this reason, in the rest of this study, only Non-Adaptive PCA method, more specifically Non-Adaptive PC2, is considered the better candidate for change detection and segmenting a larger smoke plume, compared to a simple absolute difference method, shown in Figure 26 below. The following chapter introduces two methods that are used to identify smoke plumes in PC2 rather than all the other non-smoke signals present in PC2 such as vehicles, pedestrians and building edges due to camera jitter.

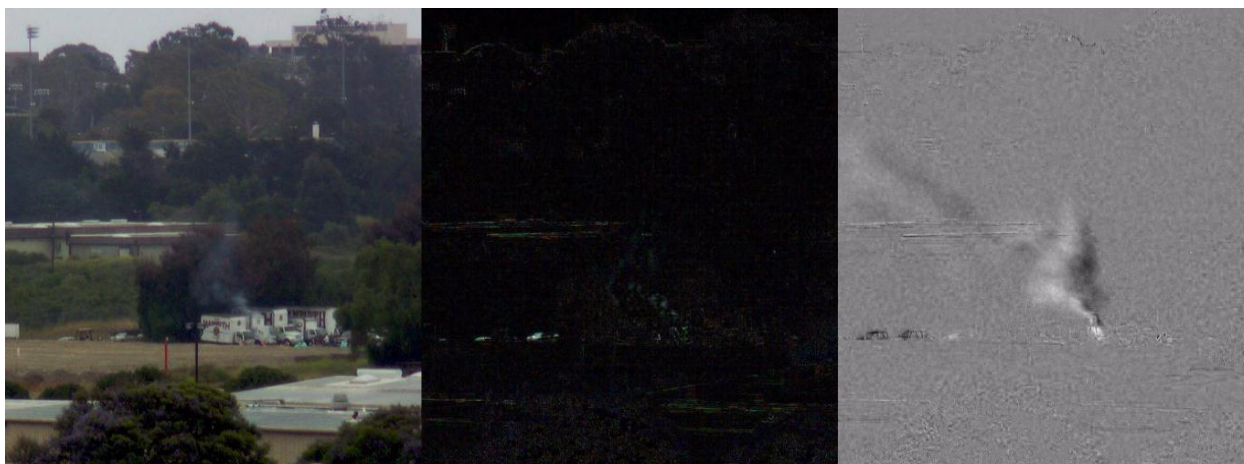


Figure 26: Original Visible Frame, Absolute Difference of Two Consecutive Visible Frames, and Scaled PC2 of Blue Band, from Left to Right, Respectively

As stated earlier in this study, Blue band showed the most smoke activity compared to both Red and Green bands. This conclusion was based on a simple visual examination and comparison of a video showing all three bands, Red, Green, and Blue bands, side by side, respectively. In order to confirm this conclusion, a more reliable method was considered. Instead of simply examining the three visible bands in their original form, temporal principal component analysis was performed on each band simultaneously and side by side for the purpose of identifying which band best reveals the smoke plume. Figures 27 and 28 below show the temporal 2nd Principal Component, PC2, of each visible band at two distinct instances of time. By examining these results, it is more evident that the Blue band shows the most smoke activity and can be a reliable band to segment smoke plumes.



Figure 27: Temporal PC2 of Red, Green, and Blue Bands, From Left to Right, Respectively, at some instant of time



Figure 28: Temporal PC2 of Red, Green, and Blue Bands, From Left to Right, Respectively, at another instant of time

Chapter 6 Segmenting Foreground and Background

6.1 Introduction

The previous chapter presented a method to amplify the smoke signal that is captured using a visible camera. The amplification technique used is temporal Principal Component Analysis; it was performed on a single visible band: Blue band. The results showed that the second Principal Component, PC2, captured an amplified smoke signal compared to what was originally captured in the visible domain. However, PCA cannot be used alone to determine if smoke is detected and therefore raise the flag. For this reason, more analysis must be performed in order to find a true indicator of smoke, or so called smoke signature.

This chapter will discuss two methods for separating foreground from background in consecutive (temporal) PC2's obtained in the previous chapter. The smoke signal and any moving objects will determine the foreground of the 2nd Principal Component, while most of the other pixels will form the background. The first technique to do so is by representing each pixel as Gaussian Mixture Models; while the second technique is simply threshold and median filtering.

6.2 Tracking Objects Using Gaussian Mixture Models

6.2.1 Introduction

There are numerous approaches used for real-time moving object detection, such as background subtraction, frame difference and optical flow. However, background subtraction is the most commonly used technique. It is based on calculating a reference image as the background. The Gaussian Mixture Model proposed in [36] is an effective technique in modeling a background scene

with repetitive motions. Modeling the background as Gaussian mixtures is more coherent and stable algorithm compared to others. Its principle is to model the values of a particular pixel as a mixture of Gaussians, each Gaussian component in the mixture has a weight. During the algorithm process, both the mean value and covariance of the Gaussian distributions get updated for each pixel which causes it to adapt to new inputs [37]. Adaptive GMM cannot overcome sudden illumination changes in the scene since the algorithm cannot update the background in time; however, this technique has become a successful solution for motion detection in the outdoors, since it incorporates illumination with slow variations as they occur during the day [37].

This analysis is performed on the assumption that the scene under observation is recorded using a fixed camera producing a fixed field of view without any jitter. The goal is to detect moving objects in the video scene. Therefore, if this method is to be applied on the original visible video sequence, the result will be a detection of anything moving within the camera's field of view, including walking pedestrians, moving vehicles, smoke plumes, and other moving objects. However, since this technique is being applied on the second Principal Component, PC2, ideally, only smoke plumes will be detected rather than all the other objects stated above. The reason behind this analogy is the fact that when Principal Component Analysis is performed, moving cars might be present in at most two or three frames per PCA computation, depending on the cars' moving speed. Hence, the cars will show up as flashing objects in few of PC2 frames, but they do not continuously and temporally move between consecutive frames. For this reason, when watching PC2 temporal frames sequence, cars will show up randomly in some frame, then disappear for the following few frames, and finally reappear in another location. Since the cars aren't continuously present in consecutive frames, Gaussian Mixture Model tracking technique won't detect cars. On the other hand, since smoke plumes are continuously moving throughout the video sequence, PC2 will as a result capture a growing and temporally moving smoke plume, making it feasible to be detected and tracked using Gaussian Mixture Model technique.

6.2.2 Theory

The probability that a certain pixel in a specific frame has a value I_t at time t is given by the following relation:

$$P(I_t) = \sum_{k=1}^K \gamma_{k,t} \cdot \boldsymbol{\varphi}(I_t, \mu_{k,t}, \sigma_{k,t}^2) \quad \text{Equation 20}$$

Where $\boldsymbol{\varphi}(I_t, \mu_{k,t}, \sigma_{k,t}^2)$ is the Gaussian probability density function described by Equation 21 below.

$$\boldsymbol{\varphi}(I_t, \mu_{k,t}, \sigma_{k,t}^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(I_t-\mu)^2}{\sigma^2}} \quad \text{Equation 21}$$

K represents the number of available Gaussian distributions. This value is limited by the available memory and the computational power. But in this study $3 \leq K \leq 5$ is used.

$\gamma_{k,t}$ represents the weight parameter of the k^{th} Gaussian component in the mixture at time t .

$\mu_{k,t}$ is the mean of the k^{th} Gaussian component in the mixture in a certain frame at time t .

$\sigma_{k,t}^2$ is the covariance of the k^{th} Gaussian component in the mixture in a certain frame at time t .

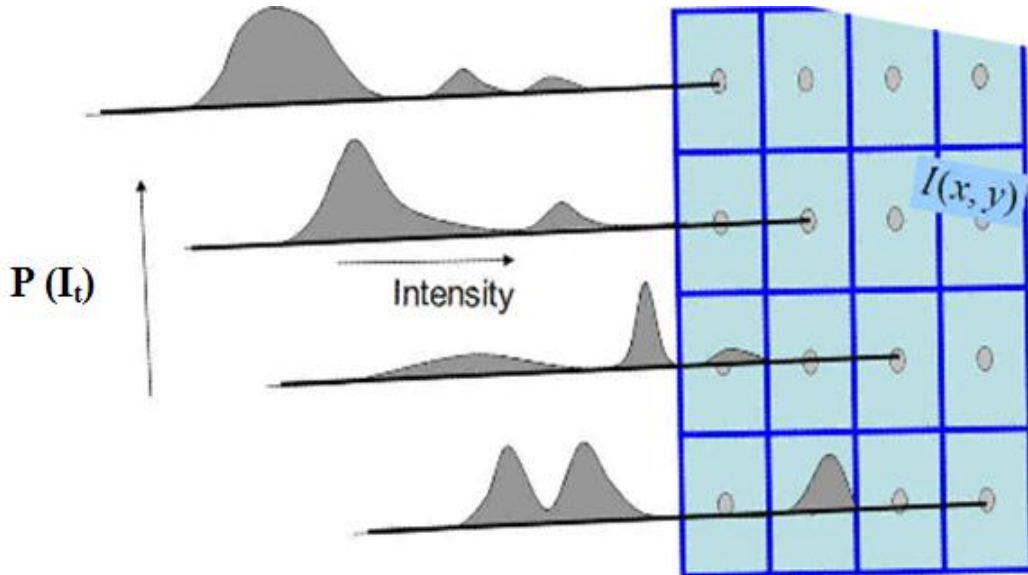


Figure 29: Pixels Intensities are modeled by a Mixture of Gaussian Distributions

In real-time computation, all new pixels intensity values, I_t , are individually checked against the K existing Gaussian distributions, and a match is determined for every pixel that has an intensity value within 2.5 standard deviation of any of the available Gaussian distributions.

If no match is found for the current pixel value, the mean value, $\mu_{k,t}$, of the least probable distribution is updated and replaced by the current pixel value. Additionally, this new Gaussian distribution will initially have high variance, and low prior weight $\gamma_{k,t}$. The prior weights, $\gamma_{k,t}$, of the K Gaussian distributions are updated at time t according to the following relation:

$$\gamma_{k,t} = (1 - r) \cdot \gamma_{k,(t-1)} + r\beta_{k,t} \quad \text{Equation 22}$$

Where r is the learning rate, and

$$\beta_{k,t} = \begin{cases} 1 & \text{for the matched model} \\ 0 & \text{for the remaining models} \end{cases} \quad \text{Equation 23}$$

Then $\gamma_{k,t}$ is renormalized, and finally, the mean and standard deviation of the unmatched distributions remain unchanged, while the respective values for the matching distribution are updated according to the following relations:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho I_t \quad \text{Equation 24}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(I_t - \mu_t)^2 \quad \text{Equation 25}$$

$$\rho = \alpha \cdot \boldsymbol{\Phi}(I_t, \mu_{k,t}, \sigma_{k,t}^2) \quad \text{Equation 26}$$

The Gaussian distributions are ordered based on the ratio of γ/σ . The first N distributions accounting for a proportion T of the observed data are defined as background, N .

$$N = \arg \min_n \left(\sum_{k=1}^n \gamma_k > T \right) \quad \text{Equation 27}$$

6.2.3 Results

As expected, Gaussian Mixture Model separated foreground from background in PC2. It was able to extract and capture most of the smoke plume that was originally segmented in PC2. On the down side, it is noted that the extracted smoke plume is smaller than the segmented area in PC2. More precisely, Gaussian Mixture Model was able to capture big smoke plumes as they were bursting from the fire source and drifting with the air; however, as smoke plumes got farther and farther from the fire source, Gaussian Mixture Model technique failed to capture the smaller smoke plumes. This behavior can be seen in Figures 30 and 31 below. In Figure 30, human eyes can perceive some of the smoke plumes that drifted from the fire location, towards the top left of the image, but GMM was not able to distinguish and capture them as part of the foreground. Additionally, there are some minor Salt and Pepper type noise introduced in GMM background and foreground separation, but can easily be removed using a simple 10x10 median filter.

Another issue that was noticed is the fact that when there is no obvious temporal movement between two consecutive PC2 frames, meaning smoke plume is stationary, GMM fails to extract the entire smoke plume. Additionally, when PC2 background changes intensity between two consecutive frames, GMM does not overcome this issue since its algorithm cannot update the background in time, due to sudden illumination variations as described earlier in this chapter. This behavior is noticeable in Figures 32 through 35 below. Figures 32 and 34 present a case where the background of PC2 varies in intensity between two consecutive frames; as seen below, Figure 34 has a slightly brighter background than Figure 32. In result, GMM algorithm cannot update its background in time, and the extracted foreground in Figure 35 does not represent the smoke plume.



Figure 30: Temporal Blue Band PC2 Frame 7911 "Oats1"

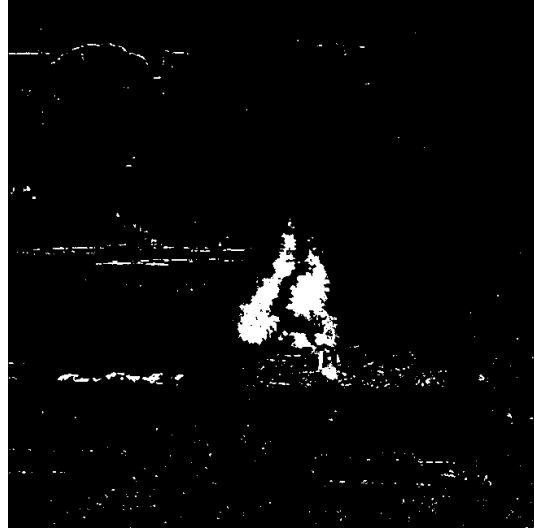


Figure 31: Mask Produced Using GMM Frame 7911 "Oats1"

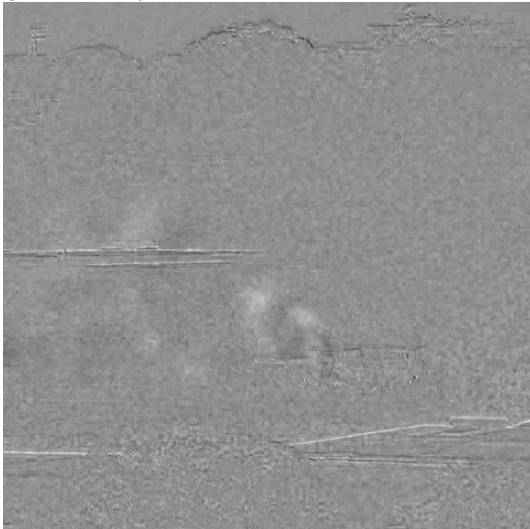


Figure 32: Temporal Blue Band PC2 Frame 7831 "Oats1"

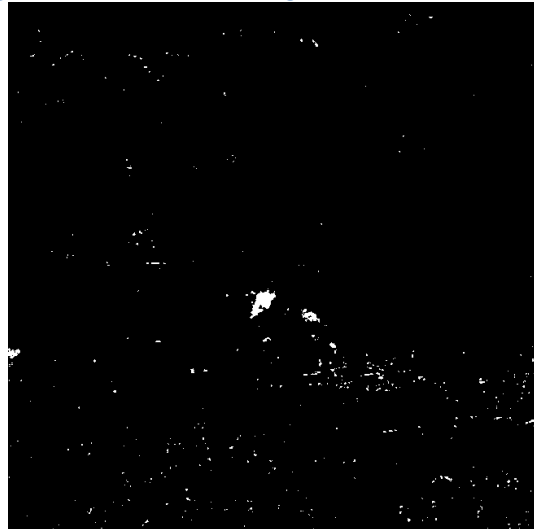


Figure 33: Mask Produced Using GMM Frame 7831 "Oats1"

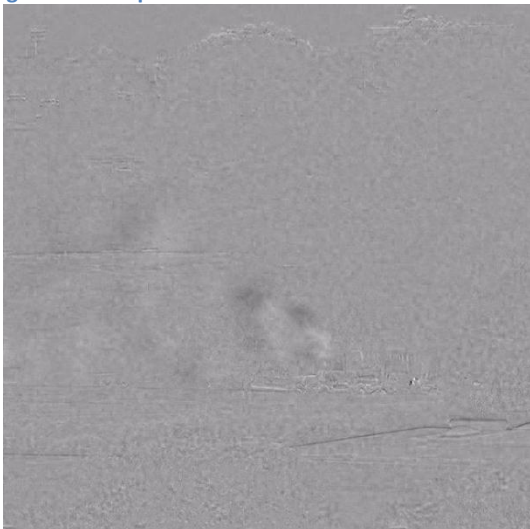


Figure 34: Temporal Blue Band PC2 Frame 7832 "Oats1"

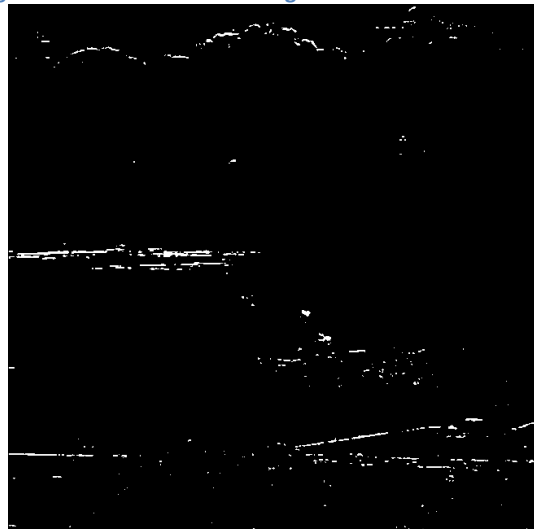


Figure 35: Mask Produced Using GMM Frame 7832 "Oats1"

6.3 Threshold and Median Filtering

6.3.1 Introduction

In the area of computer vision, blob detection refers to graphic modules that are designed to detect points and/or regions in the image that differ in properties, like brightness or color, compared to the surrounding. There are two main classes of blob detectors: (1) differential methods based on derivative expressions, and (2) methods based on local extrema in the intensity landscape. This section will only discuss second class methods, such as intensity threshold.

6.3.2 Method

Figure 36 displays a sample temporal blue band PC2 showing the smoke plume, a passing vehicle, and the background. The goal is to extract the entire smoke plume in this image. MATLAB's Contrast/Threshold tool allows the option of manually adjusting the histogram window of an image. Figure 37 shows the histogram with full dynamic range of Figure 36.



Figure 36: Temporal Blue Band PC2 Frame
7890 "Oats1"

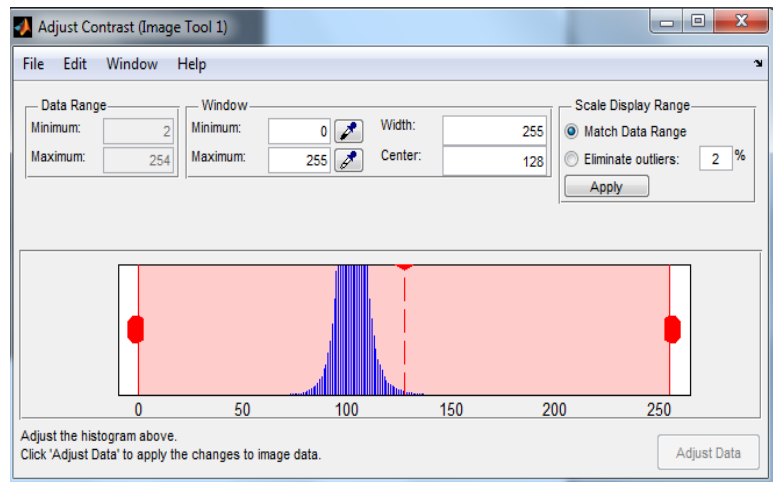


Figure 37: Histogram of the above Image using MATLAB
Contrast/Threshold Tool

As seen above, the smoke plume is made out of darker pixels and brighter pixels compared to the background of the image. In order to extract the entire smoke plume, two threshold cases must occur: one that extracts the dark pixels, and another that extracts the bright pixels. The following figures display the results along with the respective threshold window.

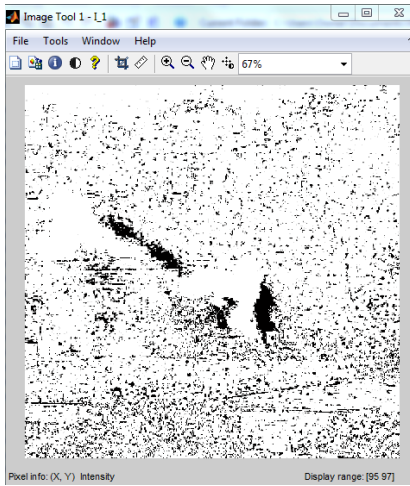


Figure 38: Segmenting the Dark Smoke Plume Pixels

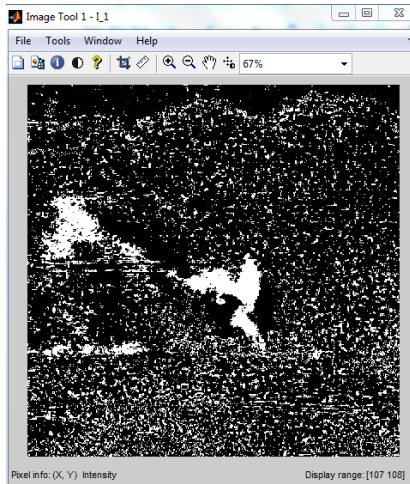


Figure 40: Segmenting the Bright Smoke Plume Pixels

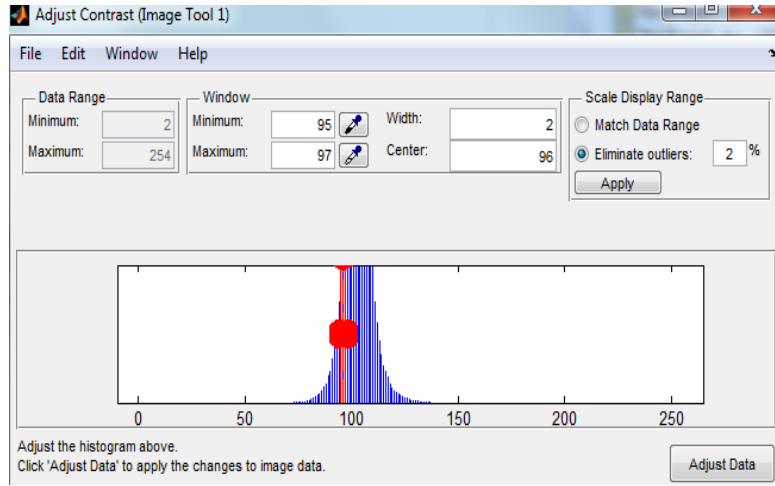


Figure 39: Histogram Showing the Threshold Window that Segments the Dark Smoke Plume Pixels

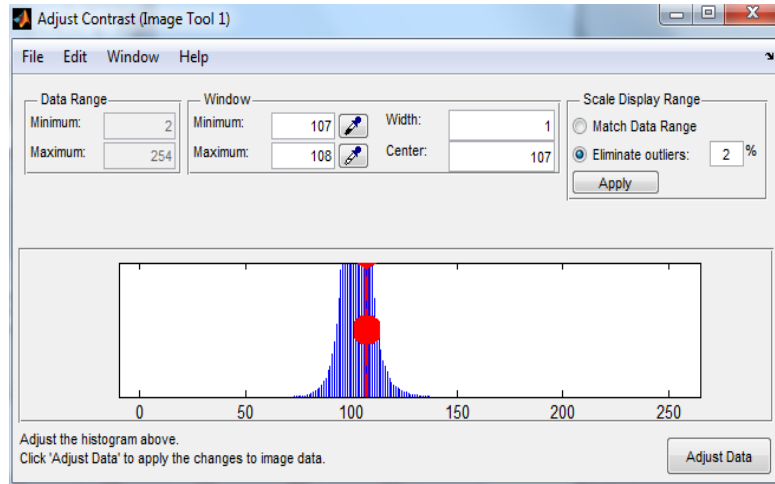


Figure 41: Histogram Showing the Threshold Window that Segments the Bright Smoke Plume Pixels

The next step is to combine both extracted plumes into one single image. But since the polarity of the two extracted plumes are opposite of each other, the polarity of one of them must first be inverted. This was done on Figure 38 and is displayed in Figure 42 below. After doing so, both extracted smoke plumes can be added together in order to form a single image, shown in Figure 43.



Figure 42: Invert the Polarity of Figure 38



Figure 43: Add Both Windows of the Segmented Smoke Plume to Produce the Full Smoke Plume (Figure42+Figure40)

6.3.3 Results

For the image displayed in Figure 36 above, the smoke plume was extracted using a simple threshold technique. However, when the same exact threshold windows were used throughout the entire frame sequence, the results weren't adequate. Figure 44 shows a sample frame using a constant threshold window obtained in the previous section. The green shapes represent the mask obtained using constant threshold level, and then superimposed on both PC2 and RGB images. Therefore, using constant threshold level does not produce adequate results of extracting the smoke plume.

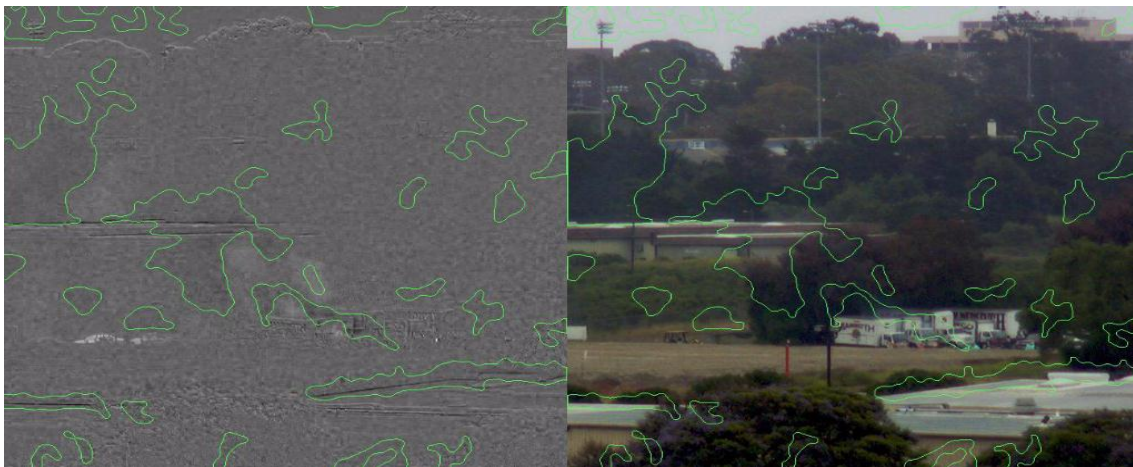


Figure 44: Superimposed Mask using Threshold over both Respective PC2 and RGB Image

Solving this issue can be done by finding a method that dynamically adjusts the threshold window, and therefore adapts to every frame. It was noted that in each PC2 frame, the highest count in the histogram belongs to the pixels forming the background of the image. So in order to adaptively adjust the threshold window, another algorithm was written. The algorithm computes the intensity value, I_T of the highest histogram counts in each frame; then, it assigns a window at ± 10 around I_T . Therefore the threshold window per frame follows the following relation:

$$I_T - 10 < I_F < I_T + 10 \quad \text{Equation 28}$$

Where I_F represents all of the pixels intensities within the specified window.

Finally, a size 10x10 median filter is applied on the image in order to remove salt and pepper noise. Figure 45 displays a sample result of the adaptive threshold technique. After running this algorithm throughout the entire frame sequence, the results are similar to the case displayed in Figure 45. Most of the smoke plume is correctly extracted in each frame, but there are still some missing blobs, for example near the fire source in Figure 45 below. Additionally, the cars are also extracted along with other minor small blobs.

The instantaneous total segmented area per frame is measured and plotted for the entire video. In order to smooth out the curve, a 20 frames moving average was implemented and applied on the instantaneous curve. These results are displayed in Figure 46 below.



Figure 45: Superimposed Mask using Adaptive Threshold over both Respective PC2 and RGB Image

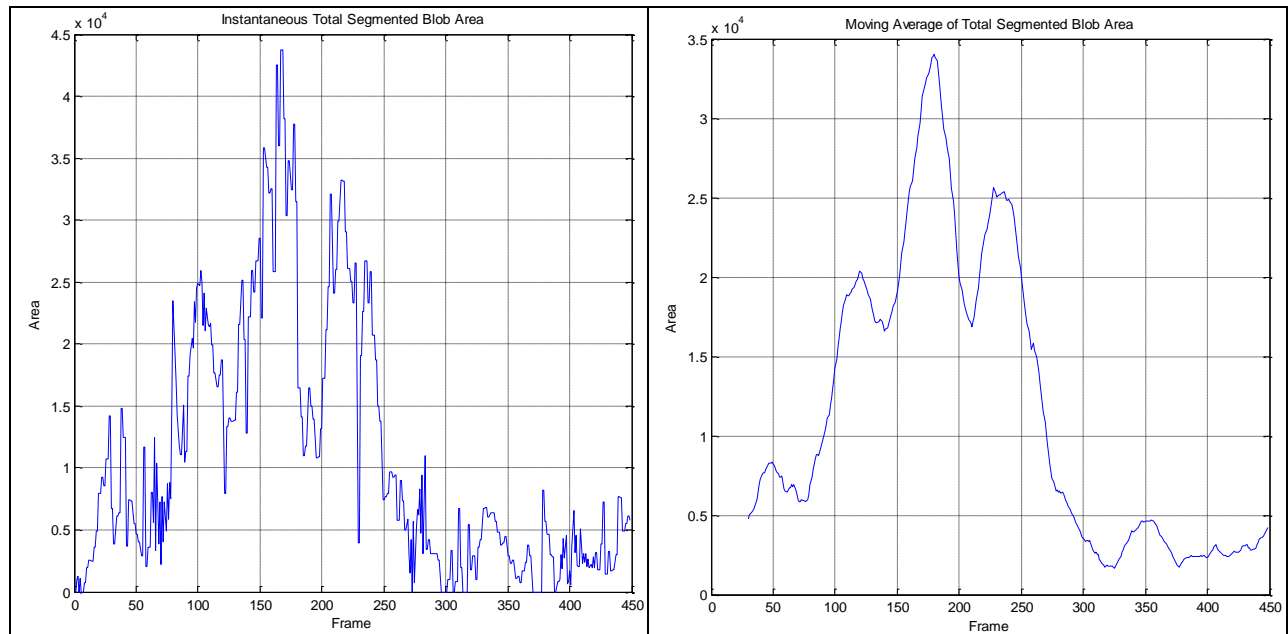


Figure 46: On the Left, Instantaneous Total Segmented Area per frame using Adaptive Threshold Technique; On the Right, 20 Frames Moving Average of Total Segmented Area per Frame, using Adaptive Threshold Technique

Chapter 7 Principal Component Analysis Modification

7.1 Introduction

In the previous chapter, it was difficult to completely isolate and extract the smoke plume in all of the video frames. This issue was determined to be caused by the sudden change in background intensity values between consecutive PC2 frames. Figures 47 and 48 illustrate two consecutive PC2 frames and their respective histograms. The background in Figure 47 has a brighter intensity color compared to the one in Figure 48. This seen assumption is confirmed by analyzing the histogram of each of the two images. The first histogram, shown in Figure 47, spans the range between 130 and 200 on a [0 255] scale, while the second histogram, shown in Figure 48 spans the range between 60 and 150 on the same scale as the previous case. This behavior raised the question: why is the background changing in PC2 and how can it be fixed? For this reason, a further investigation was performed on how PC2 is being generated.

Another issue is the appearance of building edges, tree lines, and other non-moving objects in PC2. Theoretically, none of these constant objects should appear in PC2; however, since the camera went through small vibrations due to wind, jitter was introduced to the video footage. This behavior was explained earlier in this study, and a solution was suggested by applying a video stabilizer before computing Principal Component Analysis. But as an alternative, one can just blur the original captured images, and then apply Principal Component Analysis.

Finally, earlier in this study, it was proven that the Blue band captures the most smoke signal if used exclusively in temporal Principal Component Analysis. But, this decision was made before considering spectral PCA. Therefore, in this chapter a spectral PCA is performed to explore the possibility of segmenting a better and larger smoke signal.

In summary, this chapter analyzes three issues: (1) the blurring process of the original images, (2) the modification process of Principal Component Analysis to eliminate sudden intensity changes of PC2 background, and (3) reconsider the exclusive use of the Blue band.

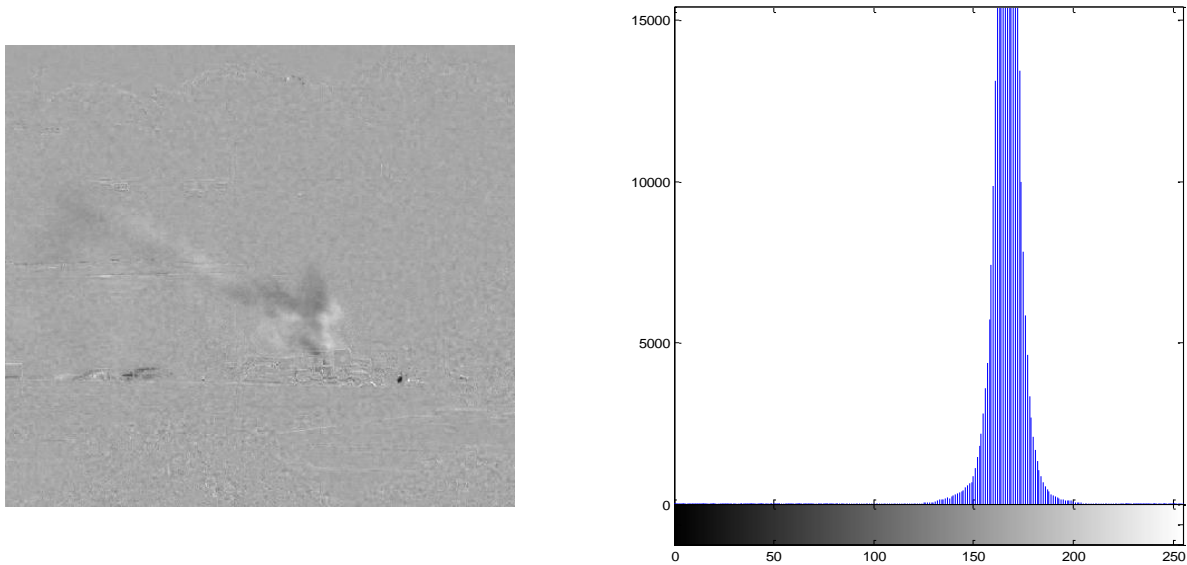


Figure 47: PC2 Frame 7888 "Oats1" on the left, and its Histogram on the Right

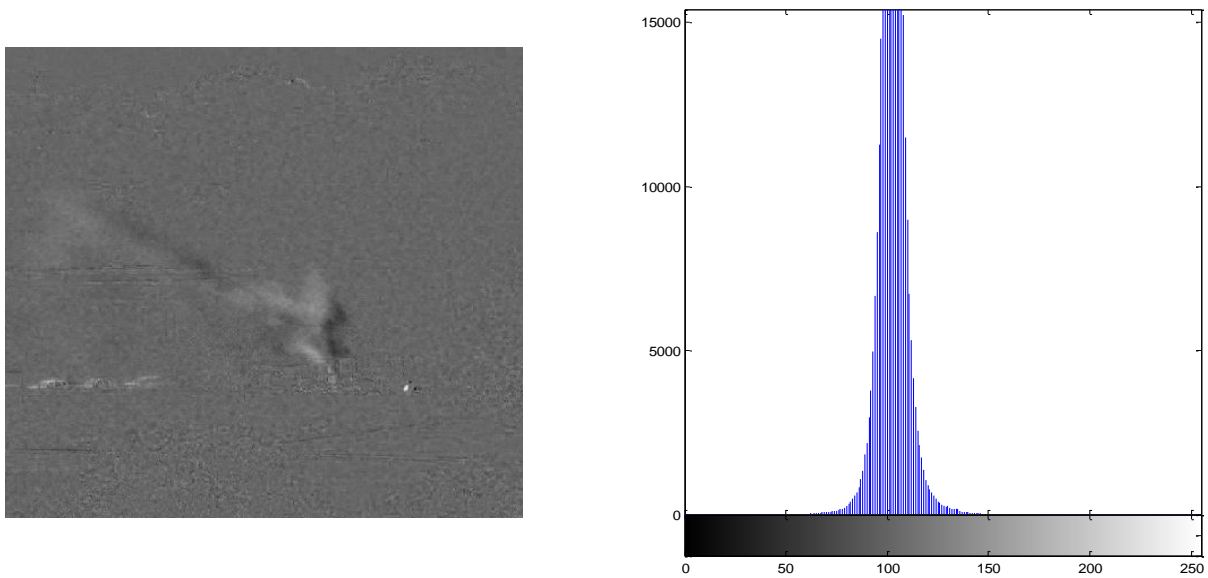


Figure 48: PC2 Frame 7890 "Oats1" on the left, and its Histogram on the Right

7.2 Blur Theory

7.2.1 Gaussian Blur Mask

Gaussian filter is a windowed filter of linear class. It operates in a similar fashion to range filtering, entropy filtering, and standard deviation filtering discussed in Chapter 3 of this study. It assigns a 2D neighborhood, and then calculates the pixel of interest intensity value according to the Gaussian distribution. Two dimensional Gaussian distribution is given by the following expression:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{Equation 29}$$

Where x and y are the pixel location, and σ is the standard deviation.

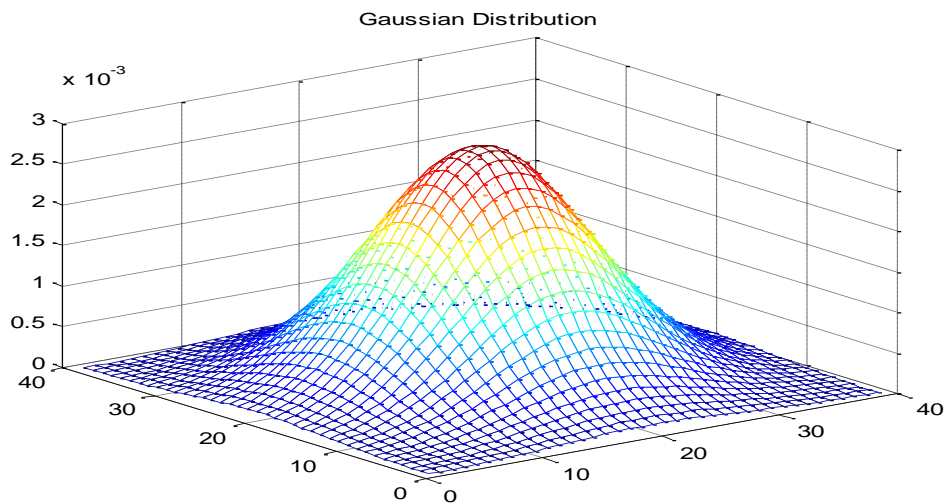


Figure 49: Gaussian Distribution Kernel Standard Deviation 1.0



Figure 50: Blue Band Frame 7888 "Capture 13" "Oats1"



Figure 51: Gaussian Blur applied to Blue Band Frame 7888 "Capture 13" "Oats1"

7.2.2 Median Filter Applied to Gaussian Blurred Image

In order to further blur the image, the same median filter, described in section 6.3.2 of this thesis was used. In this case, a size 10x10 median filter was applied on the Gaussian blurred image, shown in Figure 51. Finally, the output blurred image is presented in Figure 53 below. Since the image is blurred, the jitter that is introduced due to camera vibrations should be less than before; however, and on the down side, the smoke plume is also blurred, and it will be difficult to get its exact shape, especially on its borders and edges. Because the shape of smoke is not considered in this study and since the size of the smoke plume was not affected, this procedure can be applied.



Figure 52: Blue Band Frame 7888 "Capture 13" "Oats1"



Figure 53: Final Blurred Blue Band Frame 7888 "Capture 13" "Oats1"

7.3 Modification of PC Image to Enhance Smoke Plume Detection

After a full investigation of each step in the process of PCA described in Chapter 5 of this study, it was determined that the step that introduced the background intensity variations issue between consecutive PC2 frames is caused by the scaling to full dynamic range, described in step 7 of section 5.2 of this study.

PC2 captures most of the pixels that change within the original 9 frames that are considered in each PCA computation. However, some pixels that varied are represented as a positive value, and others are represented as negative values. This differentiation between the two types of changes affected the minimum value existent in PC2. Recall, that scaling to full dynamic range requires knowing both minimum and maximum values in PC2. These values are then linearly mapped to the full dynamic range [0 255]. The minimum value in PC2 is mapped to 0; the maximum value in PC2 is mapped to 255; and the other pixel intensities are linearly mapped accordingly in between [0 255]. Since the minimum value per PC2 frame is affected by the negative change, the background intensity shifts accordingly, causing the background to look like it is flashing between consecutive frames. This issue can be solved by calculating the absolute change, instead of having both positive and negative changes. This can be done by taking the absolute value of all PC2 pixels before mapping them to the full dynamic range. Again, the same temporal PCA procedure is applied except this time the absolute value of the Principal Components is calculated before scaling to the full dynamic range.

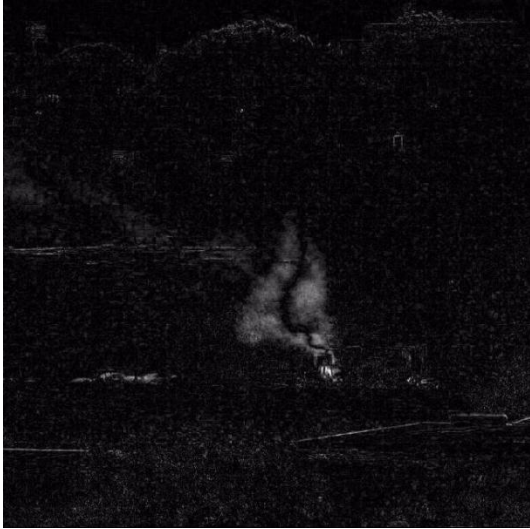


Figure 54: Absolute PC2 Blue Band Frame 7908 "Capture 13"

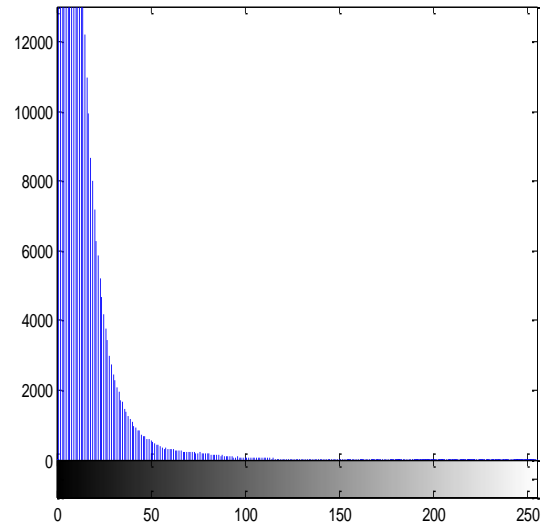


Figure 55: Histogram for the Image displayed on the left

The figure above is an example of what PC2 looks like after applying the modification on PCA. It was noted that throughout the entire frames' sequence, the background intensity values remained constant and no obvious flashing occurred. Note that no blurring was performed on the input frames before PCA in the case described above, and that's why PC2 has the random salt and pepper noise along with the tree lines and buildings.

The following images below display PC2 obtained using the modified version of PCA applied on blurred original images. In the top row, the original frames were first blurred using a Gaussian Blur filter with unity standard deviation, then blurred using a median filter of size 3x3. Similarly in the second row, but the only alteration is a bigger size median filter. The median filter used is of size 10x10.

In both cases, the background remains constant. But it is noted that as the size of the median filter is increased from 3x3 to 10x10, the smoke plume signal becomes larger and more separable from the background. This adjustment should ease the separation between the foreground and background step, in order to extract the smoke plume from the rest of the image.

Therefore, absolute Blue band PC2 obtained using the 10x10 median filtered original images creates the best candidate for segmenting the largest and clearest smoke plume. For this reason, this method will be considered in the final Early Forest Fire System design.



Figure 56: Absolute PC2 Blue Band Median Blur size 3x3
Frame 7980

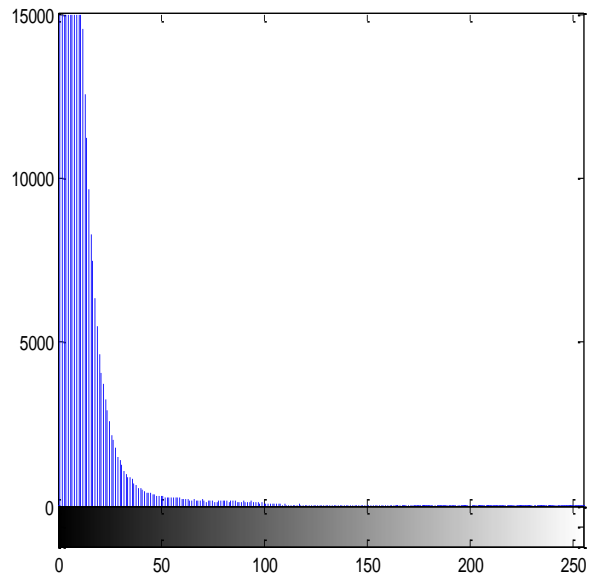


Figure 57: Histogram For the Image displayed on the left

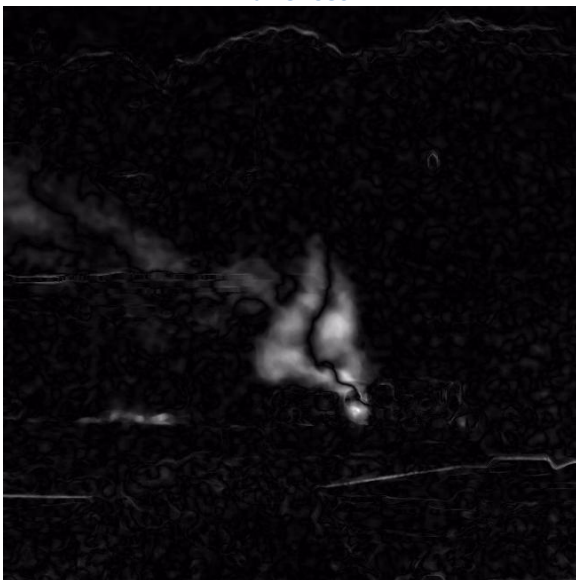


Figure 58: Absolute PC2 Blue Band Median Blur size 10x10
Frame 7980

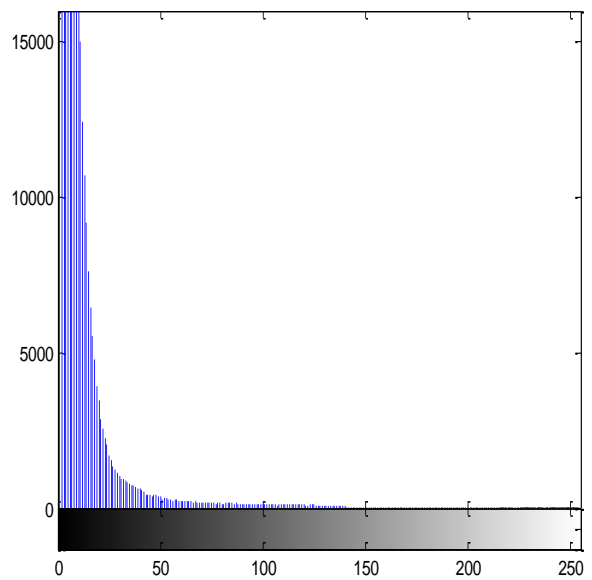


Figure 59: Histogram For the Image displayed on the left

7.4 Spectral-Temporal Principal Component Analysis

Up until now, only temporal analysis on single band is considered. Earlier in this study, it was proven that the Blue band captures the most smoke signal if used exclusively in temporal Principal Component Analysis. This section considers spectral-temporal PCA applied on RGB bands.

The only difference between the single band temporal PCA applied earlier and this spectral-temporal PCA is the original input images. Previously, the nine images in each PCA computation were nine Blue band frames, at nine instances of time. In this section, nine images are still deliberated, but only three instances of time were considered; each instance offers three bands: Red, Green, and Blue. Therefore, when combined, they form nine original frames as follow: I_{R1} , I_{G1} , I_{B1} , I_{R2} , I_{G2} , I_{B2} , I_{R3} , I_{G3} , and I_{B3} . Where $I_1 = (I_{R1}; I_{G1}; I_{B1})$ represents an RGB image at instance 1.

Figure 60 displays a case of all nine Principal Components produced using spectral-temporal PCA described above. It is noted that the smoke plume is mostly segmented in the 4th Principal Component, PC4, but also the 5th Principal Component, PC5, segments smaller smoke signal. The results do not appear to be better than the single Blue band temporal PCA applied earlier. Even if they were slightly better, considering all three bands triples the computational time. Again, the decision of solely using the single Blue band in this analysis holds.

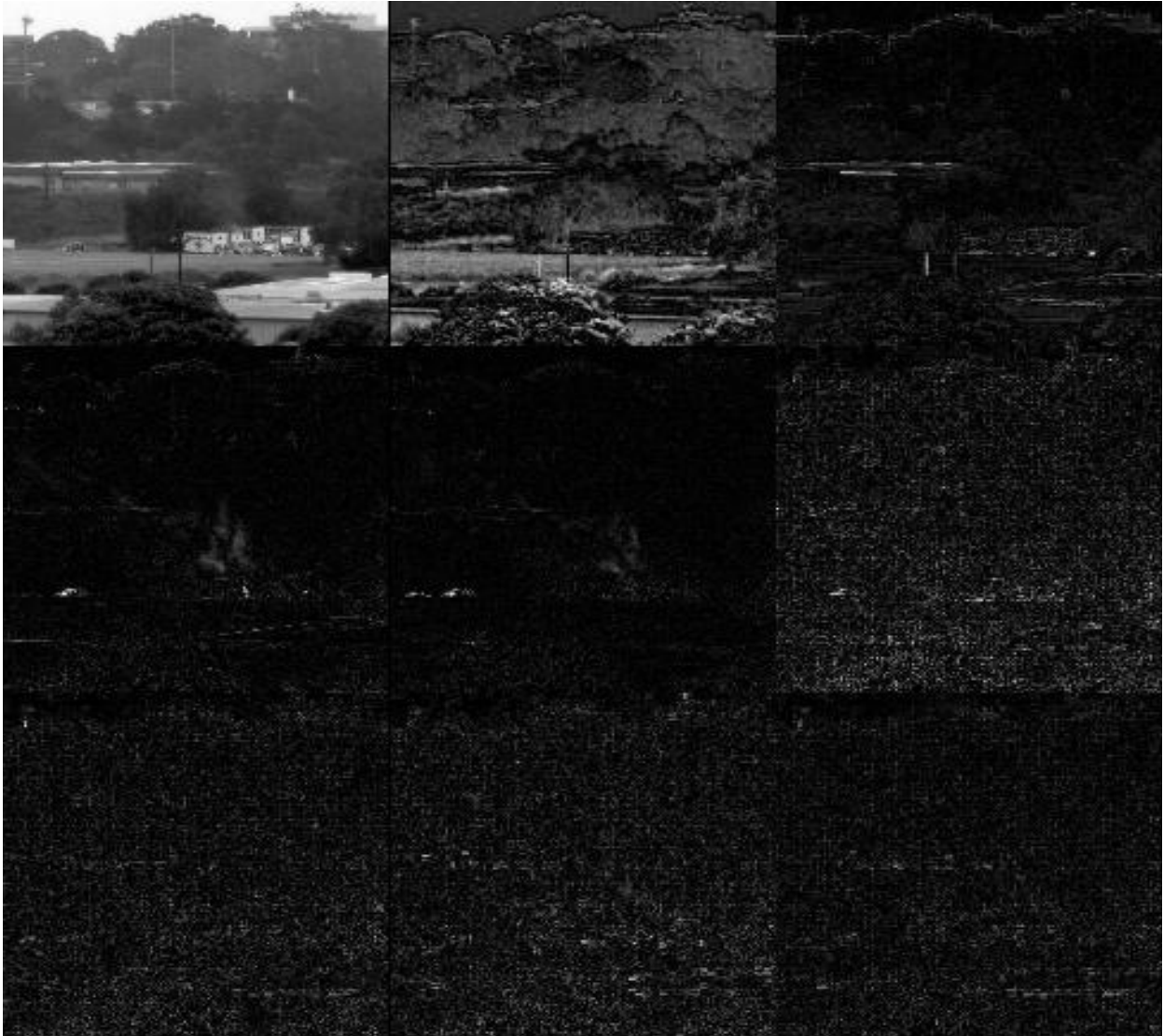


Figure 60: Spectral-Temporal Principal Component Analysis, Using RGB; From Left to Right, and Top to Bottom, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, and PC9

Chapter 8 Conclusion

8.1 Final Results for Extracting Smoke Plumes

In the previous chapter, better segmentation of smoke plume was obtained using the modified PCA technique, along with solving the background flashing and intensity varying issue. The following two sections present the results of applying the two techniques described in Chapter 6 for extracting the smoke plume from PC2; however, this time the two techniques will be applied on the modified version of PC2, also referred to as Absolute Blurred PC2.

8.1.1 Gaussian Mixture Model Results

Applying the same Gaussian Mixture Model algorithm, used earlier in this study, to the Absolute Blurred PC2 frame sequence produces an improved version of the results obtained earlier. Larger smoke plume is extracted compared to the earlier method, throughout the video. However, the Absolute Blurred PC2 continues showing a larger smoke plume than the one being extracted via Gaussian Mixture Model. Additionally, it was also noted that the background flashing issue was solved. Therefore, throughout the entire video, the algorithm was able to update the background in time, since there is no sudden change in the illumination.

Additionally, it is noted that the pixels above the fire source are not get detected. The explanation for this behavior is the fact that Gaussian Mixture Model detects pixels of moving objects only; however, the pixels at the fire source are always present in the same location, with the same intensity, which makes them look stationary. For this reason, these pixels are not detected using Gaussian Mixture Model method.

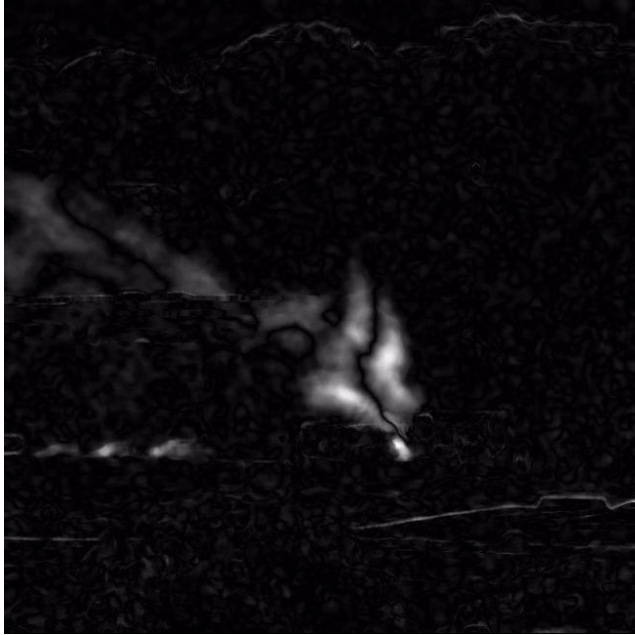


Figure 61: Absolute Blue Band PC2 of Frame 7901 "Oats1"

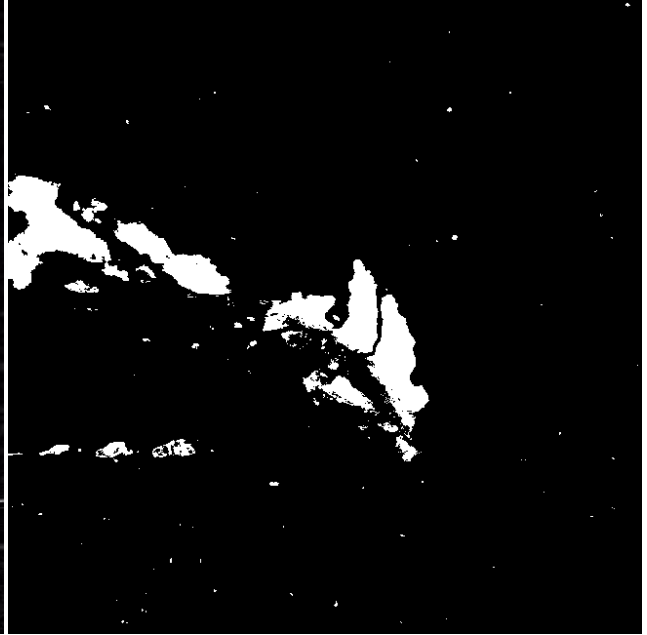


Figure 62: GMM Mask of the Image on the Left

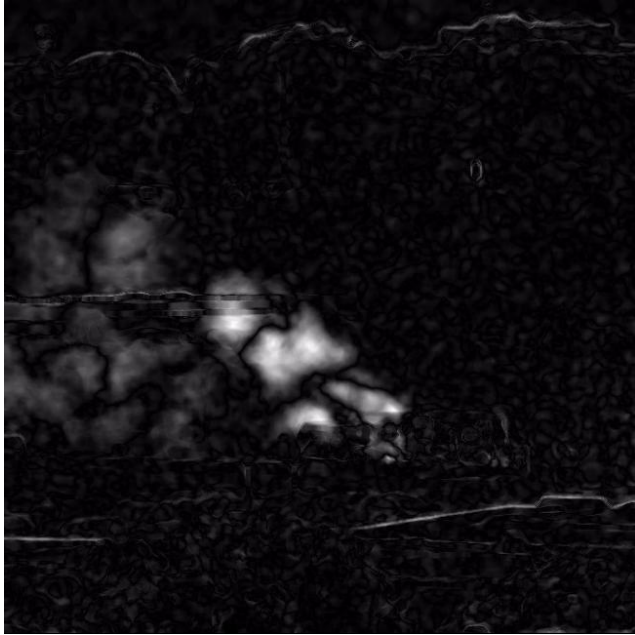


Figure 63: Absolute Blue Band PC2 of Frame 7977 "Oats1"

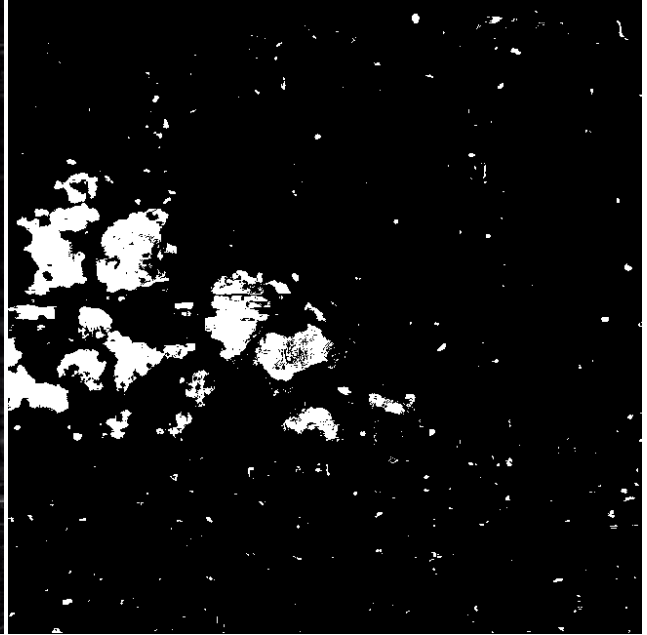


Figure 64: GMM Mask of the Image on the Left

8.1.2 Adaptive Threshold and Median Filtering Results

Similarly, applying the same adaptive threshold technique, described in section 6.3 of this study, produces results similar to the ones displayed in Figures 69 and 71. The smoke plume is fully extracted using this technique; however, other pixels are captured as well, such as the tree line, moving cars, building edges, and other random pixels. Obtaining a better and cleaner image is possible by applying a 10x10 median filter. Figures 70 and 72 display the final smoke plume mask.



Figure 65: Adaptive Threshold of Figure 61



Figure 66: 10x10 Median Filter of Image on the Left

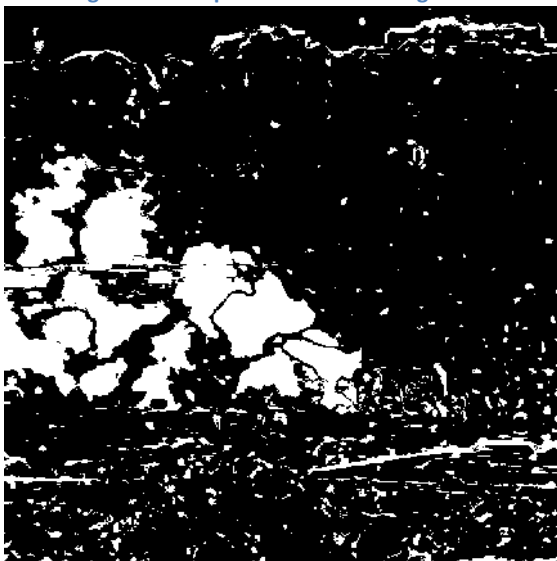


Figure 67: Adaptive Threshold of Figure 63

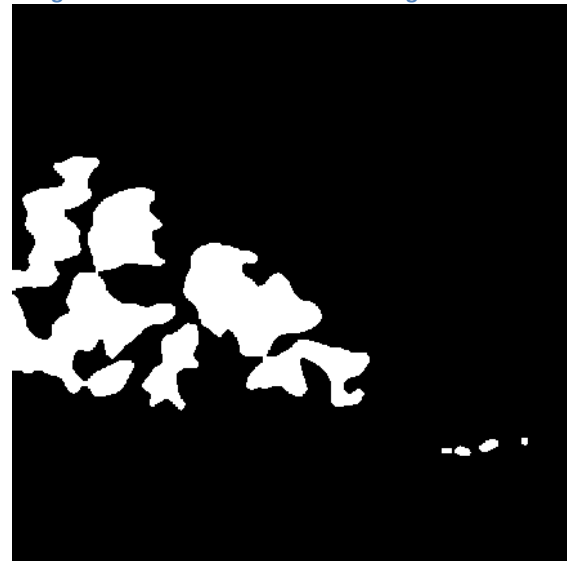


Figure 68: 10x10 Median Filter of Image on the Left

Then superimposing the final Mask onto the respective Absolute Blurred PC2 and the respective RGB colored image produces the following images.

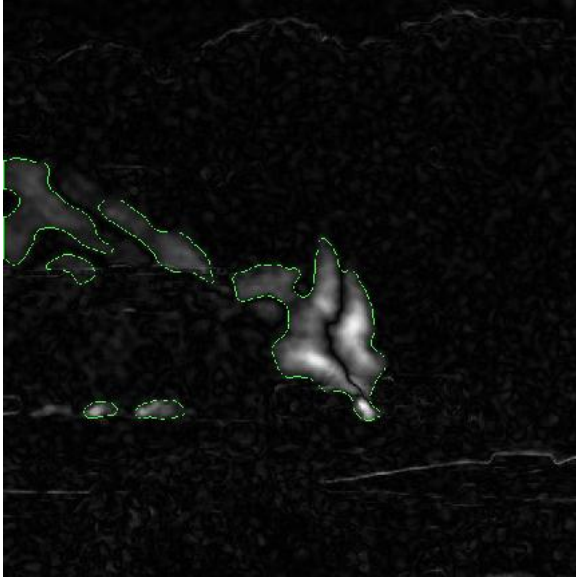


Figure 69: Mask in Figure 66 Superimposed onto Figure 61



Figure 70: Mask in Figure 66 Superimposed onto the respective RGB Captured Image

Running this algorithm throughout the entire video produced promising results; however, it had some minor anomalies. In order to reduce the false alarm rate, the total segmented area in green is recorded and plotted per frame. Then, a 20 frames moving average filter was implemented in order to smooth the curve and reduce the effects of the minor glitches. Finally, the alarm sensitivity can be controlled by setting a threshold line. If the moving average total segmented blob area crosses the threshold, an alarm is set, denoting the presence of fire. On one hand, if a car is segmented, it is most likely going to be present in few frames at most, and therefore its presence won't have major effects, since it will be averaged with the previous 20 frames. On the other hand, if smoke is increasing throughout the frame sequence, the area of the segmented blob will add up and increase until it crosses the threshold line, before triggering the alarm. The following plot displays the final results throughout the entire frame sequence. This plot proves that this improved method detects larger smoke plume than the earlier technique. In this case, the moving average plot reaches a higher maximum value than the one in Figure 46.

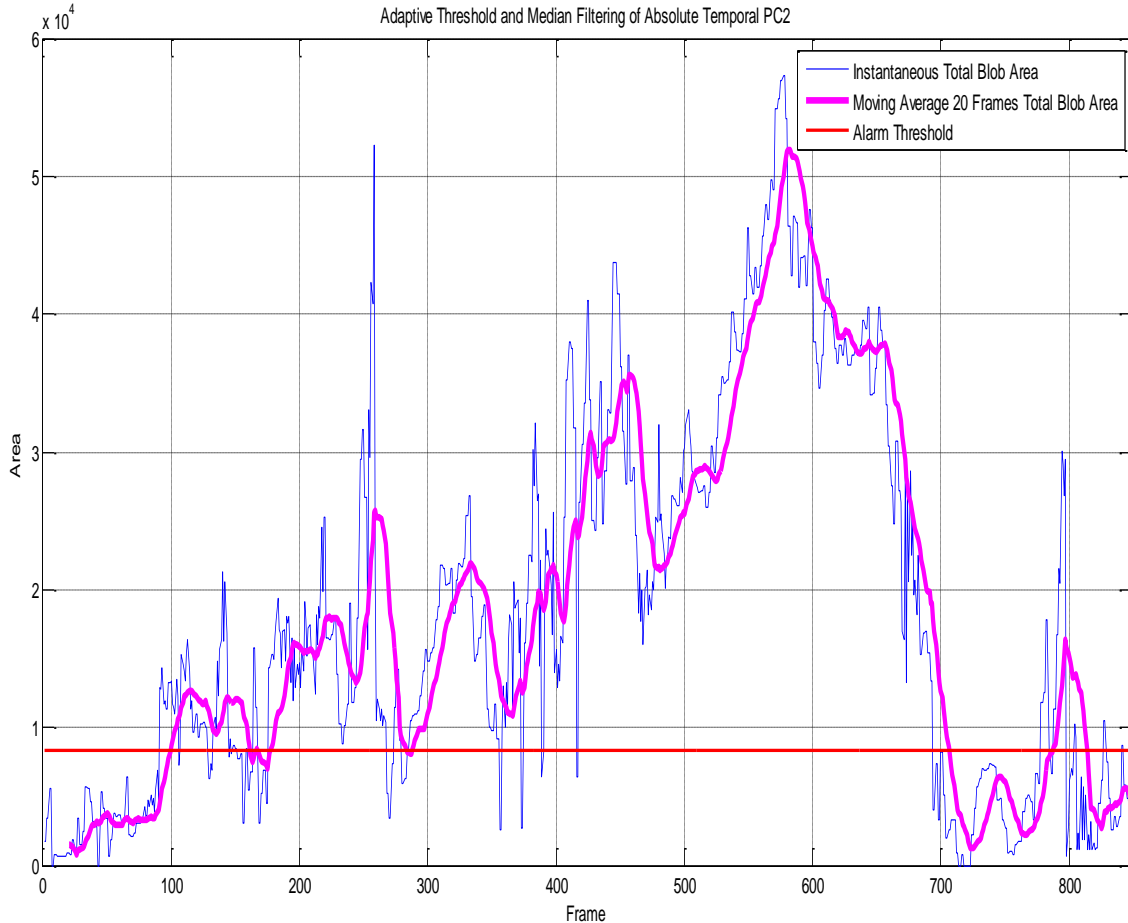


Figure 71: Total Area of Segmented Blobs using Adaptive Threshold and Median Filtering of Absolute Temporal Blurred PC2 per Frame

8.2 Future Work

The two schemes presented were tailored to detect the fire core, and the smoke, respectively. Collectively they represent a promising land-based early forest fire detection system that can help reduce the fire damage significantly. The proposed system is still in the testing and development stages. Ongoing and future work include: (1) varying fire fuel type and intensity, (2) experimenting with other types of IR cameras, (3) varying the distance from the camera to the fire site, (4) varying the composition of the features which form the fire background in the field of view, (5) performing test at night time, (6) performance optimization in terms of higher probability of detection and a lower false alarm rate, (7) assessing computation complexity and implementation cost [1].

Bibliography

- [1] John A. Saghri ; John T. Jacobs ; Daniel Kohler ; Tim Davenport ; Georges Moussa; Characterization and identification of smoke plume for early forest fire detection. Proc. SPIE 8499, Applications of Digital Image Processing XXXV, 84991J (October 15, 2012); doi:10.1117/12.930755.
- [2] John A. Saghri ; Ryan Radjabi ; John T. Jacobs; Early forest fire detection using principal component analysis of infrared video. Proc. SPIE 8135, Applications of Digital Image Processing XXXIV, 81351A (September 23, 2011); doi:10.1117/12.895512.
- [3] Timothy M. Davenport; Early Forest Fire Detection Using Texture Analysis of Principal Components from Multispectral Video; Thesis, EE Department, Cal Poly, SLO, June 2012
- [4] Daniel G. Kohler; Study of Statistical and Computational Intelligence Methods of Detecting Temporal Signature of Forest Fire Heat Plume from Single-Band Ground-Based Infrared Video; Thesis, EE Department, Cal Poly, SLO, June 2012
- [5] A. Ochoa-Brito; L. Millan-Garcia; G. Sanchez-Perez; K. Toscano-Medina; M. Nakano-Miyatake; , "Improvement of a Video Smoke Detection Based on Accumulative Motion Orientation Model," *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE* , vol., no., pp.126-130, 15-18 Nov. 2011; doi: 10.1109/CERMA.2011.27
- [6] Richard W. Bukowski ; *Techniques for Fire Detection*; USA, National Bureau of Standards, Center for Fire Research.
- [7] Z. Xiong; R. Caballero; H. Wang; A. Finn; M. A. Lelic; P. Peng; "Video-based smoke detection: possibilities, techniques, and challenges", *Suppression and Detection Research and Applications. In: A Technical Working Conference (SUPDET 2007). March 5-8 (2007) Orlando, Florida.*
- [8] B. Toreyin, Y. Dedeoglu, and A. Cetin. Flame detection in video using hidden markov models. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 1230-1233, 2005.
- [9] B. Toreyin, Y. Dedeoglu, U. Gudukbay, and A. Cetin. Computer vision based system for real-time fire and flame detection. *Pattern Recognition Letters*, 27:49-58, 2006.
- [10] "New forest fire detection system prototype installed at Lake Tahoe", PHYS.ORG, <http://phys.org/news179400560.html>, 2009.
- [11] T.A. Ollero; B.C. Arrue; J.R. Martinez; J.J. Murrillo; "Techniques for reducing false alarms in infared forest-fire automatic detection systems," *Control Engineering Practice*, Volume 7, issue 1, 123-131, 1999.

- [12] J.R. Martinez-de Diosa; T.A. Ollero; B.C. Arrue; “Distributed Intelligent Automatic Forest-Fire Detection System,” Presented at INNOCAP Conference, Grenoble, France, 1999.
- [13] D. Stipanicev; T. Vuko; D. Krstinic; M. Stula; L. Bodrozcic; “Forest Fire protection by advanced Video Detection System- Croatian Experiences,” In Third TIEMS Workshop-Improvement of Disaster Management System. Trogir, 2006.
- [14] X. Qi and J. Ebert, A computer vision-based method for fire detection in color videos, *International Journal of Imaging*, vol.2, no.S09, pp.22-34, 2009.
- [15] C. Yu, Y. Zhang, J. Fang and J. Wang, Video smoke recognition based on optical flow, *Proc. Of the 2nd International Conference on Advanced Computer Control*, vol.2, pp.16-21, 2010.
- [16] J. M. de Dios, B. Arrue, A. Ollero, L. Merino, and F. Gomez-Rodriguez. Computer vision techniques for forest fire perception. *Image and Vision Computing*, 26(4), 2008.
- [17] J. Li, Q. Qi, X. Zou, H. Peng, L. Jiang, and Y. Liang. Technique for automatic forest fire surveillance using visible light image. In *Proceedings of the International Geoscience and Remote Sensing Symposium*, volume 5, pages 31-35, 2005.
- [18] I. Bosch, S. Gomez, L. Vergara, and J. Moragues. Infrared image processing and its application to forest fire surveillance. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 283-288, 2007.
- [19] P. Guillemant and J. Vicente. Real-time identification of smoke images by clustering motions on a fractal curve with a temporal embedding method. *Optical Engineering*, 40:554-563, 2001.
- [20] Z. Xiong, R. Caballero, H. Wang, A. M. Finn, M. A. Lelic, P.-Y. Peng, Video-based smoke detection: possibilities, techniques, and challenges, IFPA, Fire Suppression and Detection Research and Applications – A Technical Working Conference (SUPDET), Orlando, FL, 2007.
- [21] T. H. Chen, Y. H. Yin, S. F. Huang, Y. T. Ye, The Smoke Detection for Early Fire- Alarming System Base on Video Processing, *Intelligent Information Hiding and Multimedia Signal Processing, IHH-MSP*, 2006, pp. 427-430.
- [22] B. U. Treyin, Y. Dedeoglu, A. E. Cetin, Wavelet Based Real-Time Smoke Detection in Video, *EUSIPCO '05*, 2005.
- [23] Z. Xu, J. Xu, Automatic Fire Smoke Detection Based on Image Visual Features, *Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops*, 2007, pp. 316-319.
- [24] P. Guillemant, J. Vicente, Real-time identification of smoke images by clustering motions on a fractal curve with a temporal embedding method, *Optical Engineering*, Vol. 40, No. 4, 2001, pp. 554-563.

- [25] T. Jakovcevic, Wildfire-Smoke Detection Based on Visible-Spectrum Image Analysis, doctoral dissertation, University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, 2011.
- [26] T. Jakovcevic, D. Stipanicev, D. Krstinic, False Alarms Reduction in Forest Fire Video Monitoring System, Proc. of [3]2. Int. Conference MIPRO 2009, Opatija, May 2009, pp. 265-270.
- [27] B. U. Toreyin, Y. Dedeoglu and A. E. Cetin, Contour based smoke detection in video using wavelets, Proc. of the 14th European Signal Processing Conference, 2006.
- [28] P. Piccinini, S. Calderara and R. Cucchiara, Reliable smoke detection in the domains of image energy and color, Proc. Of the 15th IEEE International Conference on Image Processing, pp.1376-1379, 2008.
- [29] S. Calderara, P. Piccinini and R. Cucchiara, Smoke detection in video surveillance: A MoG model in the wavelet domain, Proc. of the 6th International Conference on Computer Vision Systems, pp.119-128, 2008.
- [30] B. U. Toreyin, Y. Dedeoglu and A. E. Cetin, Wavelet based real-time smoke detection in video, Proc. of the 13th European Signal Processing Conference, 2005.
- [31] B. U. Toreyin, Y. Dedeoglu, U. Gudukbay and A. E. Cetin, Computer vision based method for real-time fire and flame detection, Pattern Recognition Letters, vol.27, no.1, pp.49-58, 2006.
- [32] G. Marbach, M. Loepfe and T. Brupbacher, An image processing technique for fire detection in video images, Fire Safety Journal, vol.41, no.4, pp.285-289, 2006.
- [33] M.J. Carlotto; "Detection and Analysis of Change in Remotely-Sensed Imagery with Application to Wide Area Surveillance", IP (6), No. 1, 1(1997), pp. 189-202.
- [34] J.A. Saghri, A.G. Tescher and J.T. Reagan, "Practical transform coding of multispectral imagery," IEEE SP Magazine, Vol. 12, pp. 32-43, Jan. 1995
- [35] J. Shlens; A Tutorial on Principal Component Analysis.
- [36] Chris Stauffer, W.E.L Grimson. "Adaptive background mixture models for real-time tracking", CVPR, Pages 246-252(1999).
- [37] Peng Suo, Yanjiang Wang. "An Improved Adaptive Background Modeling Algorithm Based on Gaussian Mixture Model". ICSP2008 Proceedings, IEEE, pp.1436-1439(2008).

Appendix A: MATLAB Functions and Scripts

```
%% Matlab Function -----  
%  
% Function : scaledata.m  
% Author : Georges Moussa  
% Date : 07/02/12  
  
function dataout = scaledata(datain,minval,maxval)  
%  
% function to linearly scale the values of a matrix between a user specified  
% minimum and a user specified maximum  
  
dataout = datain - min(datain(:));  
dataout = (dataout/range(dataout(:)))*(maxval-minval);  
dataout = dataout + minval;  
  
end
```

```
%% Matlab Function -----  
%  
% Function : PCA_1.m  
% Author : Georges Moussa  
% Date : 07/05/12  
  
function [ PC ] = PCA_1( input_images )  
%PCA_1 performs principal component analysis on input images and outputs  
%the principal components (PC). please determine the desired PC output  
% input_images : is a vector consisting of all Single Band input images  
% for principal component analysis  
  
% Read in 9 consecutive Frames but only Extract Blue Color Band  
I_1 = input_images(:,:,1);  
I_2 = input_images(:,:,2);  
I_3 = input_images(:,:,3);  
I_4 = input_images(:,:,4);  
I_5 = input_images(:,:,5);  
I_6 = input_images(:,:,6);  
I_7 = input_images(:,:,7);  
I_8 = input_images(:,:,8);  
I_9 = input_images(:,:,9);  
  
% Reshape to the correct format  
[M N K] = size(input_images);  
I = zeros(M,N,K);  
I(:,:,1)=I_1;  
I(:,:,2)=I_2;  
I(:,:,3)=I_3;  
I(:,:,4)=I_4;  
I(:,:,5)=I_5;  
I(:,:,6)=I_6;  
I(:,:,7)=I_7;
```

```

I(:,:,8)=I_8;
I(:,:,9)=I_9;

% Reshape into vector
I_vec = reshape(I,M*N,K);

% Get the transformation matrix
[fprincompT fPCT ev] = fPCA(I_vec);

% Transformation matrix
T = fPCT;

% Calculate the Principal Components
PCvec = PCT(I_vec,T);

% Reshape back to an image size and shape
PC = reshape(PCvec,M,N,K);

% Individual Principal Components
PC1 = (PC(:,:,1));
PC2 = (PC(:,:,2));
PC3 = (PC(:,:,3));
PC4 = (PC(:,:,4));
PC5 = (PC(:,:,5));
PC6 = (PC(:,:,6));
PC7 = (PC(:,:,7));
PC8 = (PC(:,:,8));
PC9 = (PC(:,:,9));

end

```

```

%% Matlab Script -----%
% Script : tracking.m
% Author : Georges Moussa
% Date : 08/03/12
%
% Description : it tracks moving object in a video using Gaussian Mixtture
% Models

%% Tracking Objects Using Gaussian Mixture Models

% Create a System object to read video from avi file.
filename = 'PC2.avi';
hvfr = vision.VideoFileReader(filename, ...
    'ImageColorSpace', 'RGB');

% Create color space converter System object to convert the image from RGB
% to intensity format.
hcsc = vision.ColorSpaceConverter('Conversion', 'RGB to intensity');

% Create a System object to detect foreground using Gaussian mixture
% models.
hof = vision.ForegroundDetector(...)

```

```

        'NumTrainingFrames', 5, ...    % only 5 because of short video
        'InitialVariance', (30/255)^2); % initial standard deviation of
30/255

% Create a blob analysis System object to segment objects in the video.
hblob = vision.BlobAnalysis( ...
    'CentroidOutputPort', false, ...
    'AreaOutputPort', true, ...
    'BoundingBoxOutputPort', true, ...
    'OutputDataType', 'single', ...
    'MinimumBlobAreaSource', 'Property', ...
    'MinimumBlobArea', 150, ...
    'MaximumBlobAreaSource', 'Property', ...
    'MaximumBlobArea', 3600, ...
    'MaximumCount', 80);

% Create and configure two System objects that insert shapes, one for
% drawing the bounding box around the detected objects and the other for
% drawing the motion vector lines.

hshapeins1 = vision.ShapeInserter( ...
    'BorderColor', 'Custom', ...
    'CustomBorderColor', [0 255 0]);
hshapeins2 = vision.ShapeInserter( ...
    'Shape','Lines', ...
    'BorderColor', 'Custom', ...
    'CustomBorderColor', [255 255 0]);

% Create and configure a System object to write the number of objects being
% tracked.
htextins = vision.TextInserter( ...
    'Text', '%4d', ...
    'Location', [1 1], ...
    'Color', [255 255 255], ...
    'FontSize', 12);

% Create System objects to display the results.
sz = get(0, 'ScreenSize');
pos = [20 sz(4)-300 200 200];
hVideoOrig = vision.VideoPlayer('Name', 'Original', 'Position', pos);
pos(1) = pos(1)+220; % move the next viewer to the right
hVideoFg = vision.VideoPlayer('Name', 'Foreground', 'Position', pos);
pos(1) = pos(1)+220;
hVideoRes = vision.VideoPlayer('Name', 'Results', 'Position', pos);

line_row = 23; % Define region of interest (ROI)

% Stream Processing Loop
% Create a processing loop to track the objects in the input video. This loop
% uses the previously instantiated System objects.
% When the VideoFileReader object detects the end of the input file,
% the loop stops.

while ~isDone(hvfr)

```

```

image = step(hvfr);           % Read input video frame
y = step(hcsc, image);       % Convert color image to intensity

% Remove the effect of sudden intensity changes due to camera's
% auto white balancing algorithm.
y = im2single(y);
y = y-mean(y(:));
% hmedianfilt = vision.MedianFilter([5 5]);
y = medfilt2(y, [3 3]); % Median Filtering of the foreground

fg_image = step(hof, y); % Detect foreground

% Estimate the area and bounding box of the blobs in the foreground
% image.
[area, bbox] = step(hblob, fg_image);

image_out = image;
image_out(22:23, :, :) = 255; % Count objects only below this white line
image_out(1:15, 1:30, :) = 0; % Black background for displaying count

Idx = bbox(:,2) > line_row; % Select boxes which are in the ROI.

% Based on dimensions, exclude objects which are not smoke. When the
% ratio between the area of the blob and the area of the bounding box
% is above 0.3 (30%) classify it as a smoke plume.
ratio = zeros(length(Idc),1);
ratio(Idc) = single(area(Idc,1))./single(bbox(Idc,3).*bbox(Idc,4));
ratiob = ratio > 0.3;
count = int32(sum(ratiob)); % Number of objects
bbox(~ratiob,:) = int32(-1);

% Draw bounding rectangles around the detected objects.
image_out = step(hshapeins1, image_out, bbox);

% Display the number of object tracked and a white line showing the ROI.
image_out = step(htextins, image_out, count);

step(hVideoOrig, image); % Original video
step(hVideoFg, fg_image); % Foreground
step(hVideoRes, image_out); % Bounding boxes around Smoke
end

% Close the video file
release(hvfr);

```

```

%% Matlab Script -----%
% Function : Adapt_PCA.m
% Author : Georges Moussa
% Date : 10/13/12

%% Adaptive PCA for Visible Capture 13
begin_frame = 7400;
last_frame = 8248;

```

```

% User Input
ncols      = 650;%873;
nrows      = 650;%1068;
frmstart   = begin_frame;
frmstep    = 1;
frmend     = last_frame;
vidfps     = 10;

% generate movie object
disp('creating movie object')
frminfo    = [num2str(frmstart) '-' num2str(frmstep) '-' num2str(frmend)];
vidinfo    = ['_at' num2str(vidfps) 'fps' ];
rawMovObj  = VideoWriter('PC2.avi');
rawMovObj.FrameRate = vidfps;
open(rawMovObj);

for j=begin_frame:1:last_frame,

% Open in 9 consecutive visible frames
file_name1 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j) '.BMP'];
file_name2 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+2) '.BMP'];
file_name3 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+4) '.BMP'];
file_name4 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+6) '.BMP'];
file_name5 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+8) '.BMP'];
file_name6 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+10) '.BMP'];
file_name7 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+12) '.BMP'];
file_name8 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+14) '.BMP'];
file_name9 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+16) '.BMP'];

% Read the 9 consecutive frames
I_1 = im2double(imread(file_name1));
I_2 = im2double(imread(file_name2));
I_3 = im2double(imread(file_name3));
I_4 = im2double(imread(file_name4));
I_5 = im2double(imread(file_name5));
I_6 = im2double(imread(file_name6));
I_7 = im2double(imread(file_name7));
I_8 = im2double(imread(file_name8));
I_9 = im2double(imread(file_name9));

% Only Extract Blue Color Band and a smaller FOV
I_1 = I_1(101:750,176:825,3);
I_2 = I_2(101:750,176:825,3);
I_3 = I_3(101:750,176:825,3);

```



```

I_4 = I_4(101:750,176:825,3);
I_5 = I_5(101:750,176:825,3);
I_6 = I_6(101:750,176:825,3);
I_7 = I_7(101:750,176:825,3);
I_8 = I_8(101:750,176:825,3);
I_9 = I_9(101:750,176:825,3);

% Display the 9 consecutive Frames before PCA
l = size(I_1,1);
k = size(I_1,2);
figure(10)
imshow([I_1 zeros(l,1) I_2 zeros(l,1) I_3 ;...
        I_4 zeros(l,1) I_5 zeros(l,1) I_6 ;...
        I_7 zeros(l,1) I_8 zeros(l,1) I_9 ]);
title('9 Consecutive Blue Band Frames')

% Reshape for PCA
input_images = zeros(size(I_1,1),size(I_1,2),9);
input_images(:,:,1) = I_1;
input_images(:,:,2) = I_2;
input_images(:,:,3) = I_3;
input_images(:,:,4) = I_4;
input_images(:,:,5) = I_5;
input_images(:,:,6) = I_6;
input_images(:,:,7) = I_7;
input_images(:,:,8) = I_8;
input_images(:,:,9) = I_9;

% % Blocks for Blue Band (PCA)
% Non Adaptive
fun1 = @(block_struct) ...
        (PCA_1(block_struct.data)) ; % 873 1068
PC = blockproc(input_images,[650 650], fun1);

% Adaptive
fun2 = @(block_struct) ...
        (PCA_1(block_struct.data)) ; % 873 1068
PC_adapt = blockproc(input_images,[150 150], fun2);

% Before Scaling to Full Dynamic Range
l1 = size(PC(:,:,1),1);
k1 = size(PC(:,:,1),2);
figure(30)
imshow([PC(:,:,1) zeros(l1,1) PC(:,:,2) zeros(l1,1) PC(:,:,3) ;...
        PC(:,:,4) zeros(l1,1) PC(:,:,5) zeros(l1,1) PC(:,:,6) ;...
        PC(:,:,7) zeros(l1,1) PC(:,:,8) zeros(l1,1) PC(:,:,9) ]);
title('Non Adaptive Principal Components')

% Display Histograms for Non adaptive before Full Dynamic Range Scaling
figure(31)
subplot(3,3,1)
imhist(PC(:,:,1))
subplot(3,3,2)

```

```

imhist(PC(:,:,2))
subplot(3,3,3)
imhist(PC(:,:,3))
subplot(3,3,4)
imhist(PC(:,:,4))
subplot(3,3,5)
imhist(PC(:,:,5))
subplot(3,3,6)
imhist(PC(:,:,6))
subplot(3,3,7)
imhist(PC(:,:,7))
subplot(3,3,8)
imhist(PC(:,:,8))
subplot(3,3,9)
imhist(PC(:,:,9))

l2 = size(PC_adapt(:,:,1),1);
k2 = size(PC_adapt(:,:,1),2);
figure(40)
imshow([PC_adapt(:,:,1) zeros(l2,1) PC_adapt(:,:,2) zeros(l2,1)
PC_adapt(:,:,3) ;...
        PC_adapt(:,:,4) zeros(l2,1) PC_adapt(:,:,5) zeros(l2,1)
PC_adapt(:,:,6) ;...
        PC_adapt(:,:,7) zeros(l2,1) PC_adapt(:,:,8) zeros(l2,1)
PC_adapt(:,:,9) ]]);
title('Adaptive Principal Components')

% Display Histograms for Non adaptive before Full Dynamic Range Scaling
figure(41)
subplot(3,3,1)
imhist(PC_adapt(:,:,1))
subplot(3,3,2)
imhist(PC_adapt(:,:,2))
subplot(3,3,3)
imhist(PC_adapt(:,:,3))
subplot(3,3,4)
imhist(PC_adapt(:,:,4))
subplot(3,3,5)
imhist(PC_adapt(:,:,5))
subplot(3,3,6)
imhist(PC_adapt(:,:,6))
subplot(3,3,7)
imhist(PC_adapt(:,:,7))
subplot(3,3,8)
imhist(PC_adapt(:,:,8))
subplot(3,3,9)
imhist(PC_adapt(:,:,9))

% Rescale to Full Dynamic Range [0 1]
I_9 = im2uint8(scaledata(I_9,0,1)); % Original Blue Band
PC1_Scaled = im2uint8(scaledata(PC(:,:,1),0,1)); % PC1
PC2_Scaled = im2uint8(scaledata(PC(:,:,2),0,1)); % PC2
PC3_Scaled = im2uint8(scaledata(PC(:,:,3),0,1)); % PC3
PC4_Scaled = im2uint8(scaledata(PC(:,:,4),0,1)); % PC4
PC5_Scaled = im2uint8(scaledata(PC(:,:,5),0,1)); % PC5

```

```

PC6_Scaled = im2uint8(scaledata(PC(:, :, 6), 0, 1)); % PC6
PC7_Scaled = im2uint8(scaledata(PC(:, :, 7), 0, 1)); % PC7
PC8_Scaled = im2uint8(scaledata(PC(:, :, 8), 0, 1)); % PC8
PC9_Scaled = im2uint8(scaledata(PC(:, :, 9), 0, 1)); % PC9

PC1_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 1), 0, 1)); % Adaptive PC1
PC2_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 2), 0, 1)); % Adaptive PC2
PC3_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 3), 0, 1)); % Adaptive PC3
PC4_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 4), 0, 1)); % Adaptive PC4
PC5_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 5), 0, 1)); % Adaptive PC5
PC6_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 6), 0, 1)); % Adaptive PC6
PC7_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 7), 0, 1)); % Adaptive PC7
PC8_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 8), 0, 1)); % Adaptive PC8
PC9_Scaled_Adapt = im2uint8(scaledata(PC_adapt(:, :, 9), 0, 1)); % Adaptive PC9

% Display PC2 for Non Adaptive PCA
figure(1)
imshow(PC2_Scaled)
% Display PC2 for Adaptive PCA
figure(2)
imshow(PC2_Scaled_Adapt)

% Display all Principal Components
l1 = size(PC1_Scaled, 1);
k1 = size(PC1_Scaled, 2);
figure(3)
imshow([PC1_Scaled zeros(l1, 1) PC2_Scaled zeros(l1, 1) PC3_Scaled ;...
        PC4_Scaled zeros(l1, 1) PC5_Scaled zeros(l1, 1) PC6_Scaled ;...
        PC7_Scaled zeros(l1, 1) PC8_Scaled zeros(l1, 1) PC9_Scaled ]);
title('Non Adaptive Principal Components')

l2 = size(PC1_Scaled_Adapt, 1);
k2 = size(PC1_Scaled_Adapt, 2);
figure(4)
imshow([PC1_Scaled_Adapt zeros(l2, 1) PC2_Scaled_Adapt zeros(l2, 1)
PC3_Scaled_Adapt ;...
        PC4_Scaled_Adapt zeros(l2, 1) PC5_Scaled_Adapt zeros(l2, 1)
PC6_Scaled_Adapt ;...
        PC7_Scaled_Adapt zeros(l2, 1) PC8_Scaled_Adapt zeros(l2, 1)
PC9_Scaled_Adapt ]);
title('Adaptive Principal Components')

% After Scaling to Full Dynamic Range
% Display Histograms for Non adaptive before Full Dynamic Range Scaling
figure(51)
subplot(3, 3, 1)
imhist(PC1_Scaled)
subplot(3, 3, 2)
imhist(PC2_Scaled)
subplot(3, 3, 3)
imhist(PC3_Scaled)
subplot(3, 3, 4)
imhist(PC4_Scaled)
subplot(3, 3, 5)
imhist(PC5_Scaled)
subplot(3, 3, 6)

```

```

imhist(PC6_Scaled)
subplot(3,3,7)
imhist(PC7_Scaled)
subplot(3,3,8)
imhist(PC8_Scaled)
subplot(3,3,9)
imhist(PC9_Scaled)

% Display Histograms for Non adaptive before Full Dynamic Range Scaling
figure(61)
subplot(3,3,1)
imhist(PC1_Scaled_Adapt)
subplot(3,3,2)
imhist(PC2_Scaled_Adapt)
subplot(3,3,3)
imhist(PC3_Scaled_Adapt)
subplot(3,3,4)
imhist(PC4_Scaled_Adapt)
subplot(3,3,5)
imhist(PC5_Scaled_Adapt)
subplot(3,3,6)
imhist(PC6_Scaled_Adapt)
subplot(3,3,7)
imhist(PC7_Scaled_Adapt)
subplot(3,3,8)
imhist(PC8_Scaled_Adapt)
subplot(3,3,9)
imhist(PC9_Scaled_Adapt)

%% make movie
disp('Reading in frames to movie object')

img = im2uint8(PC2_Scaled);% zeros(size(PC9_Scaled,1),2) PC9_Scaled
zeros(size(PC9_Scaled,1),2) PC9_Scaled_Adapt]);
index = ceil(j/frmstep);
writeVideo(rawMovObj, img);
clc
end

%% close file
disp('closing movie object')
close(rawMovObj);

```

```

%% Matlab Script -----%
% Function : Blob_Detect.m
% Author : Georges Moussa
% Date : 07/25/12

%% Blob Detection

% Detect Blobs on Non Adaptive PC2
regions = detectSURFFeatures(im2double(PC2_Scaled));

```

```

% Display Current Image
figure(5)
imshow(I_9)

% Display Non Adaptive Scaled PC2 with Blobs
figure(6)
imshow(PC2_Scaled); hold on;
plot(regions)

% Detect Blobs on Adaptive PC2
regions_Adapt = detectSURFFeatures(im2double(PC2_Scaled_Adapt));

% Display Adaptive Scaled PC2 with Blobs
figure(7)
imshow(PC2_Scaled_Adapt); hold on;
plot(regions_Adapt)

% Before Scaling
regions1 = detectSURFFeatures(im2double(PC(:,:,2)));
figure(11)
imshow(PC(:,:,2)); hold on;
plot(regions1)

regions1_Adapt = detectSURFFeatures(im2double(PC_adapt(:,:,2)));
figure(12)
imshow(PC_adapt(:,:,2)); hold on;
plot(regions1_Adapt)

```

```

%% Matlab Script -----%
% Function : Heat_Plume.m
% Author : Georges Moussa
% Date : 03/12/12

begin_frame = 2000;
last_frame = 3000;

%% User Input
ncols      = 873;
nrows     = 1068;
bandname  = 'CMWIR';
testname  = 'oats1';
prefix    = [bandname '_' testname '_'];
datapath  = 'C:\Users\Owner\Documents\Original Data\oats1\';
filepath  = [datapath bandname '_' testname '_'];
frmstart  = begin_frame;
frmstep   = 1;
frmend    = last_frame;
vidfps    = 20;

%% generate movie object
disp('creating movie object')
frminfo   = [num2str(frmstart) '-' num2str(frmstep) '-' num2str(frmend)];

```

```

vidinfo = ['_at' num2str(vidfps) 'fps' ];
rawMovObj = VideoWriter('Original_PC2.avi');
rawMovObj.FrameRate = vidfps;
open(rawMovObj);
%%
for j=begin_frame:1:last_frame,
    file_name = ['C:\Users\Owner\Documents\Original Data\oats1\CMWIR_oats1_'
num2str(j) '_fov.raw'];
    file_name2 = ['C:\Users\Owner\Documents\Original Data\oats1\CMWIR_oats1_'
num2str(j+1) '_fov.raw'];

% Open The First Frame
fid1 = fopen(file_name,'r');
x1= (uint16(fread(fid1,[1068 873],'uint16')))/2^14;
x11=double(imadjust((x1(1:853,:))));

% Open the Second Consecutive Frame
fid2 = fopen(file_name2,'r');
x2= (uint16(fread(fid2,[1068 873],'uint16')))/2^14;
x22=im2double(imadjust(x2(1:853,:)));

figure(1)
output_frame = entropyfilt(abs(rangefilt(x2)-rangefilt(x1)));

%% Trying other Combinations
% output_frame = abs(x2-x1);
% output_frame = (x2-x1); % difference final minus initial
% output_frame1 = (stdfilt(x1(1:853,:))); % standard deviation
% output_frame2 = (stdfilt(output_frame1)); %variance
% output_frame3 = (rangefilt(x1)); % maximum and minimum within 3x3 pixels
% output_frame4 = (entropyfilt(x1)); % measure of disorder/variation within
3x3 pixels
% output_frame5 = (stdfilt(output_frame)); % standard deviation of the
difference
% output_frame6 = (stdfilt(output_frame5)); % variance of the difference
% output_frame7 = (rangefilt(output_frame)); % rangefilt of the difference
% output_frame8 = (entropyfilt(output_frame)); % entropyfilt of the
difference
% output_frame9 = (x1-x2); % difference initial minus final
% output_frame10 = abs((stdfilt(x2)-stdfilt(x1))); %difference of the
standard deviations
%output_frame11= entropyfilt(output_frame10);
% output_frame11 = (stdfilt(stdfilt(x2))-stdfilt(stdfilt(x1)));
%difference of the variances
% output_frame12 = (rangefilt(x2)-rangefilt(x1)); % difference of the
rangefilts
% output_frame13 = entropyfilt(x2)-entropyfilt(x1); % difference of the
entropyfilts

%% Imadjust the frames
% output_frame = ((imadjust(output_frame(1:853,:))));
% output_frame1 = (imadjust((output_frame1(1:853,:))));
% output_frame2 = (imadjust(uint16(output_frame2(1:853,:))));
% output_frame3 = (imadjust(uint16(output_frame3(1:853,:))));
% output_frame4 = (imadjust(uint16(output_frame4(1:853,:))));

```

```

% output_frame5 = (imadjust(uint16(output_frame5(1:853, :))));
% output_frame6 = (imadjust(uint16(output_frame6(1:853, :))));
% output_frame7 = (imadjust((output_frame7(1:853, :))));
% output_frame8 = (imadjust(uint16(output_frame8(1:853, :))));
% output_frame9 = (imadjust(uint16(output_frame9(1:853, :))));
% output_frame10 = (imadjust(uint16(output_frame10(1:853, :))));
% output_frame11 = (imadjust(uint16(output_frame11(1:853, :))));
% output_frame12 = (imadjust((output_frame12(1:853, :))));
% output_frame13 = (imadjust(uint16(output_frame13(1:853, :))));
%imshow(imadjust(output_frame(1:853, :)));

% Threshold
%figure(4)
%output_frame(output_frame > 190) = 65536;
%imshow(output_frame);

%% make movie
disp('reading in frames to movie object')
%for i = frmstart:frmstep:frmend

    img = im2uint8([x11 output_frame]);
    index = ceil(j/frmstep);
    fclose(fid1);
    fclose(fid2);
    writeVideo(rawMovObj, img);
    clc

end
%% close file
disp('closing movie object')
close(rawMovObj);

```

```

%% Matlab Script -----%
% Function : Blob_Segmentation.m
% Author : Georges Moussa
% Date : 11/14/12

begin_frame = 1;%400;
last_frame = 848;%449;%417;
total = zeros(1,last_frame - begin_frame + 1);

%L = struct(zeros(1,last_frame - begin_frame + 1));
%H = zeros(1,last_frame - begin_frame + 1);
begin_frame_color = 7800;
last_frame_color = 8248;

% User Input
ncols = 650;%480;%873;
nrows = 650;%584;%1068;
bandname = 'Visible Blue';
testname = 'Capture_13';
prefix = [bandname '_' testname '_'];
datapath = 'C:\Users\Owner\Documents\Original Data\oats1\';
filepath = [datapath bandname '_' testname '_'];
frmstart = begin_frame;

```

```

frmstep    = 1;
frmend     = last_frame;
vidfps     = 5;
%
% generate movie object
disp('creating movie object')
frminfo    = [num2str(frmstart) '-' num2str(frmstep) '-' num2str(frmend)];
vidinfo    = ['_at' num2str(vidfps) 'fps' ];
rawMovObj  =
VideoWriter('Blurred_1_Med_10_PC2_B_abs_1F_thre_15_Overlay_Moving_Average.avi
');
rawMovObj.FrameRate = vidfps;
open(rawMovObj);

for j=begin_frame:1:last_frame,
% Open images
% file_name1 = ['C:\Users\Owner\Documents\MATLAB\PC2_B_1F_abs\PC2_B_1F_abs_'
num2str(j) '.BMP'];
% file_name1 =
['C:\Users\Owner\Documents\MATLAB\PC2_Blue_Thresh_2_Inv\PC2_Blue_Thresh_2_Inv_'
num2str(j) '.BMP'];
% file_name1 = ['C:\Users\Owner\Documents\MATLAB\PC2_Blue_Scaled (Visible
Capture 13 Oats1 Frame 7800 to Frame 8248)\PC2_Blue_Scaled_' num2str(j)
'.BMP'];
% file_name2 = ['C:\Users\Owner\Documents\MATLAB\PC2_Blue_Scaled (Visible
Capture 13 Oats1 Frame 7800 to Frame 8248)\PC2_Blue_Scaled_' num2str(j+1)
'.BMP'];
% file_name3 = ['C:\Users\Owner\Documents\MATLAB\PC2_Blue_Scaled (Visible
Capture 13 Oats1 Frame 7800 to Frame 8248)\PC2_Blue_Scaled_' num2str(j+2)
'.BMP'];

%file_name1 =
['C:\Users\Owner\Documents\MATLAB\Blurred_1_Med_10_PC2_B_abs_1F\Blurred_1_Med
_10_PC2_B_abs_1F_' num2str(j) '.BMP'];
%file_name1 =
['C:\Users\Owner\Documents\MATLAB\GMM_Blurred_Abs_PCA\GMM_Blurred_Abs_PCA_'
num2str(j) '.BMP'];
file_name1 =
['C:\Users\Owner\Documents\MATLAB\Blurred_1_Med_10_PC2_B_abs_1F\Blurred_1_Med
_10_PC2_B_abs_1F_' num2str(j) '.BMP'];
file_name9 = ['C:\Users\Owner\Documents\Original
Data\Visible\VISIBLE_capture-13_' num2str(j+7416) '.BMP'];
file_name3 =
['C:\Users\Owner\Documents\MATLAB\Moving_Average_20_Total_Area_Abs_PC2\Moving
_Average_20_Total_Area_Abs_PC2_' num2str(j) '.BMP'];

% % Read the Original colored Image
I_9 = im2uint8(imread(file_name9));
I_9 = I_9(101:749,176:825,:); %smaller FOV

% Read Absolute PC2 of Blurred Original Image
% I_2 = im2uint8(imread(file_name2));
% I_2 = I_2(1:649,(:,1)); %smaller FOV

% Read Plot Frames
I_3 = im2uint8(imread(file_name3));

```



```

% Zero pad in order to match the same size
B = padarray(I_3, [153;107]);
% I_cropped = imread(file_name1);
% I_cropped = (I_cropped(:,:,1));

I_1 = im2uint8(imread(file_name1));
I_1 = I_1(1:649, :, 1);
% I_2 = im2uint8(imread(file_name2));
% I_2 = I_2(:,:,1);
% I_3 = im2uint8(imread(file_name3));
% I_3 = I_3(:,:,1);

%% Blob Segmentation
%I_eq = adapthisteq(I_cropped);
% figure(1)
% imshow(I_1)

%% Gaussian Blur Filter
% h = fspecial('gaussian', size(I_1), 1.0); % Make a Gaussian Mask
% I_Blurred = imfilter(I_1, h, 'symmetric'); % Apply Gaussian Mask to Blur
Image
% figure(2)
% imshow(I_Blurred)

% H = fspecial('disk',10);
% blurred = imfilter(I,H,'replicate');
%% median filtering
m = 25; % Dimension of median filter
n = 25; % Dimension of median filter
B1 = medfilt2(I_1, [m n], 'symmetric'); % Apply Median Filter
% figure(3)
% imshow(B1)

%% Threshold
[M N]=size(B1);

T1 = median(median(B1))+15;% 5 lower bound
% T2 = median(median(B1))+10;% 10 upper bound
% % T1 = 98;
% % T2 = 113;
% NewImage1 = uint8(255*(B1 >= T1));% To capture the dark portions of PC2;
% NewImage2 = uint8(255*(B1 >= T1));% To capture the bright portions of PC2;
% % NewImage3 = uint8(255*(B1 >= T1 & B1 >= T2));
%
% figure(4)
% imshow(NewImage1)
% figure(5)
% imshow(NewImage2)

% figure(6)
% imshow(NewImage3)

% % Invert NewImage1 to have the same polarity as NewImage1
% NewImage1_inv = imcomplement(NewImage1);
% figure(6)

```

```

% imshow(NewImage1_inv)

% % Add NewImage2_inv to NewImage1 to segment the entire smoke plume in PC2
% Mask = imadd(NewImage2,NewImage1_inv);
% figure(11)
% imshow(Mask)
Mask = uint8(255*(B1 >= T1));
%% Remove Blobs with small area
% Cleaning up and then overlaying the perimeter on the Original PC2 image.
bw2 = imfill(Mask, 'holes');
% figure(12)
% imshow(bw2)

bw3 = imopen(bw2, ones(5,5));
% figure(13)
% imshow(bw3)

bw4 = bwareaopen(bw3, 578);% 578 Remove the cars and small blobs;
% figure(14)
% imshow(bw4)

%% median filtering
m = 15; % Dimension of median filter
n = 15; % Dimension of median filter
B2 = medfilt2(bw4, [m n], 'symmetric'); % Apply Median Filter

bw4_perim = bwperim(B2,4);
overlay1 = imoverlay(I_1, bw4_perim, [.3 1 .3]);% Change this to I_1 instead
of I_2
% figure(15)
% imshow(overlay1)

% Overlay Green color Mask on original Colored Image
bw4_perim = bwperim(bw4,4);
overlay2 = imoverlay(I_9, bw4_perim, [.3 1 .3]);
figure(16)
imshow(overlay2)

%% Calculate Blob total area

total(j) = bwarea(bw4);

%% Fourier Transform of PC2 in Attempt to Blur the image using LPF
% z1 = fftshift(fft2(I_1));
% % figure(10)
% % imagesc(20.*log10(abs(z1))), colormap(gray), axis image, colorbar;
% % title('fft2 shifted of original Magnitude Response (Log Scale)');
%
% [m n]=size(z1);
% % Generate a 2D Low Pass Filter
% c = lpf2D([m,n],0.1,100);
%
% % filter The image in frequency domain
% p = (z1.*c);
% w=im2uint8(scaledata((abs(iff2(p))),0,1));

```

```

%
% figure(11)
% imshow(w);
% title('Blurred');

% T9 = median(median(w))-15;% lower bound
% T10 = median(median(w))+15;% upper bound
% NewImage5 = uint8(255*(w >= T9 & w <= T10));
% NewImage5 = imcomplement(NewImage5);

%bw = im2bw(I_eq, graythresh(I_eq));

% % inverts the colors in the image
% NewImage1 = imcomplement(NewImage1);
% % figure(2)
% % imshow(NewImage1)

% T3 = median(median(I_2))-15;% lower bound
% T4 = median(median(I_2))+15;% upper bound
% NewImage2 = uint8(255*(I_2 >= T3 & I_2 <= T4));
% % inverts the colors in the image
% NewImage2 = imcomplement(NewImage2);
%
% T5 = median(median(I_3))-15;% lower bound
% T6 = median(median(I_3))+15;% upper bound
% NewImage3 = uint8(255*(I_3 >= T5 & I_3 <= T6));
% % inverts the colors in the image
% NewImage3 = imcomplement(NewImage3);
%
% NewImage4 = zeros(M,N);
%% Time moving Average per pixel
% for m=1:1:M,
%     for n = 1:1:N,
%         NewImage4(m,n) = (1/3)*NewImage1(m,n) + (1/3)*NewImage2(m,n)...
%             + (1/3)*NewImage3(m,n);
%     end
% end
% NewImage4 = im2uint8(NewImage4);

% figure(1)
% imshow(NewImage1)
%
% figure(2)
% imshow(I_1)

% figure(2)
% imshow(NewImage4)

%% median filtering
% m = 27;
% n = 27;
%
% B1 = medfilt2(NewImage1, [m n]);
% % figure(2)
% % imshow(I_eq)

```

```

%
% bw = B1;
% figure(3)
% imshow(bw)

% % Cleaning up and then overlaying the perimeter on the Original PC2 image.
% bw2 = imfill(bw,'holes');
%
% bw3 = imopen(bw2, ones(5,5));
% % figure(4)
% % imshow(bw3)
% bw4 = bwareaopen(bw3, 400);%40);
% % figure(5)
% % imshow(bw4)
%
% bw4_perim = bwperim(bw4,4);
% overlay1 = imoverlay(I_1, bw4_perim, [.3 1 .3]);

% mask = zeros(size(bw4_perim,1),size(bw4_perim,2),3);
% mask(:,:,1) = bw4_perim(:,:,);
% mask(:,:,2) = bw4_perim(:,:,);
% mask(:,:,3) = bw4_perim(:,:,);

% figure(7)
% imshow(overlay1)

% figure(5)
% imshow([bw bw2 bw3 bw4])

% mask_em = imextendedmax(I_eq, 30);
% figure(4)
% imshow(mask_em)

% % Let's clean that up and then overlay it.
%
% mask_em = imclose(mask_em, ones(5,5));
% mask_em = imfill(mask_em, 'holes');
% mask_em = bwareaopen(mask_em, 40);
% overlay2 = imoverlay(I_eq, bw4_perim | mask_em, [.3 1 .3]);
% figure(5)
% imshow(overlay2)

% Dimensions of the frame
l1 = size(I_9,1);
k1 = size(I_9,2);
B2_Mask = zeros(l1,k1,3);
B2_Mask(:,:,1) = B2(:,:,);
B2_Mask(:,:,2) = B2(:,:,);
B2_Mask(:,:,3) = B2(:,:,);
B2_Mask = im2uint8(B2_Mask);
%% make movie
disp('Reading in frames to movie object')

img = im2uint8([B2_Mask overlay1 overlay2 B]);
index = ceil(j/frmstep);

```

```

        writeVideo(rawMovObj, img);
        clc
    j
    %%
    % figure(100)
    % plot(total)
    % axis([0 (last_frame-begin_frame+1) 0 6.0*10^4])
    % grid on
    % title('Instantaneous Total Segmented Blob Area')
    % xlabel('Frame');
    % ylabel('Area');
    % hold on
    %
    % % Set a Threshold line (Alarm On if Total area is greater than this
    % % threshold)
    % z = 8300.*ones(1,last_frame - begin_frame + 1);
    % plot(z,'r')
    % hold off
    % L(j) = getframe;
    %
    % output = tsmovavg(total, 's', 20);
    % figure(101)
    % plot(output)
    % axis([0 (last_frame-begin_frame+1) 0 6.0*10^4])
    % grid on
    % title('Moving Average of Total Segmented Blob Area')
    % xlabel('Frame');
    % ylabel('Area');
    % hold on
    %
    % % Set a Threshold line (Alarm On if Total area is greater than this
    % % threshold)
    % y = 8300.*ones(1,last_frame - begin_frame + 1);
    % plot(y,'r')
    % hold off
    % H(j)=getframe;

end
% numtimes=1;
% fps=5;
% figure(100)
% movie(L,numtimes,fps)
% figure(101)
% movie(H,numtimes,fps)
% %Save video/Animations
% movie2avi(L, 'Instantaneous_Total_Area_Abs_PC2.avi','compression',
'none');
% movie2avi(H, 'Moving_Average_20_Total_Area_Abs_PC2.avi','compression',
'none');

% figure(100)
% plot(total)
% grid on
% %title('Instantaneous Total Segmented Blob Area')
% xlabel('Frame');

```

```

% ylabel('Area');
% %
% output = tsmovavg(total, 's', 30);
% figure(101),plot(output)
% grid on
% title('Moving Average of Total Segmented Blob Area')
% xlabel('Frame');
% ylabel('Area');
% hold on
% %
% % Set a Threshold line (Alarm On if Total area is greater than this
% % threshold)
% y = 8300.*ones(1,last_frame - begin_frame + 1);
% plot(y,'r')

% close file
disp('closing movie object')
close(rawMovObj);

```

```

begin_frame = 1;
last_frame = 448;

% User Input
ncols      = 650;%873;
nrows      = 650;%1068;
bandname    = 'Visible Blue';
testname    = 'Capture_13';
prefix      = [bandname '_' testname '_' ];
datapath    = 'C:\Users\Owner\Documents\MATLAB\PC2_Blue_Scaled (Visible Capture
13 Oats1 Frame 7800 to Frame 8248)';
filepath    = [datapath bandname '_' testname '_' ];
frmstart    = begin_frame;
frmstep     = 1;
frmend      = last_frame;
vidfps      = 5;

% generate movie object
disp('creating movie object')
frminfo     = [num2str(frmstart) '-' num2str(frmstep) '-' num2str(frmend)];
vidinfo     = ['_at' num2str(vidfps) 'fps' ];
rawMovObj   = VideoWriter('PC2_Blue_Thres5_Med5_1_2_5_7_10.avi');
rawMovObj.FrameRate = vidfps;
open(rawMovObj);
%% Matlab Script -----%
% Function : PC2_Blue_Difference.m
% Author : Georges Moussa
% Date : 11/14/12

for j=begin_frame:1:last_frame,

% Open in 2 consecutive visible frames

```

```

file_name1 = ['C:\Users\Owner\Documents\MATLAB\PC2_Blue_Scaled (Visible
Capture 13 Oats1 Frame 7800 to Frame 8248)\PC2_Blue_Scaled_' num2str(j)
'.BMP'];
%file_name2 = ['C:\Users\Owner\Documents\MATLAB\PC2_Blue_Scaled (Visible
Capture 13 Oats1 Frame 7800 to Frame 8248)\PC2_Blue_Scaled_' num2str(j+1)
'.BMP'];

% Read the 2 consecutive frames
I_1 = im2double(imread(file_name1));
%I_2 = im2double(imread(file_name2));

I_1 = im2uint8(I_1(:,:,1));
%I_2 = I_2(:,:,1);
% % Compute the Absolute Difference in PC2
% I_diff = abs(I_2 - I_1);
%
% % Rescale Absolute Difference to Full Dynamic Range [0 1]
% I_diff_Scaled = im2uint8(scaledata(I_diff,0,1));

% Dimensions of the frame
l1 = size(I_1,1);
k1 = size(I_1,2);
%
% I_2 = im2uint8(I_2);
% I_diff = im2uint8(I_diff);

% % Display Figures
% figure(1)
% imshow(I_1)
%
% figure(2)
% imshow(I_2)
%
% figure(3)
% imshow(I_diff)
%
% figure(4)
% imshow(I_diff_Scaled)

%% Threshold
T1 = median(median(I_1))-5;% lower bound
T2 = median(median(I_1))+5;% upper bound
NewImage1 = uint8(255*(I_1 >= T1 & I_1 <= T2));
%
% inverts the colors in the image
IM1 = imcomplement(NewImage1);

% median filtering
m = 7;
n = 7;

B1 = medfilt2(IM1, [m n]);
B2 = medfilt2(B1, [m n]);
B3 = medfilt2(B2, [m n]);
B4 = medfilt2(B3, [m n]);

```

```

B5 = medfilt2(B4, [m n]);
B6 = medfilt2(B5, [m n]);
B7 = medfilt2(B6, [m n]);
B8 = medfilt2(B7, [m n]);
B9 = medfilt2(B8, [m n]);
B10 = medfilt2(B9, [m n]);
%% make movie
disp('Reading in frames to movie object')

    img = im2uint8([I_1 zeros(11,2) B1 zeros(11,2) B2;
        B5 zeros(11,2) B7 zeros(11,2) B10]);
    index = ceil(j/frmstep);
    writeVideo(rawMovObj, img);
    clc
j
end

% close file
disp('closing movie object')
close(rawMovObj);

```